발 간 번 호
11-1360000-100251-01

2026 기상기후데이터 카탈로그

개정판

기상청 Korea Meteorological Administration
국가기후데이터센터

발 간 번 호
11-1360000-100251-01

2026 기상기후데이터 카탈로그

개정판

기상청 Korea Meteorological Administration
국가기후데이터센터



기상기후데이터 카탈로그
프롤로그

기상기후데이터는 이제 국민의 일상과 안전을 지키는 기초 정보를 넘어, 산업 혁신과 과학 연구, 인공지능(AI) 기술 발전을 이끄는 국가 핵심 데이터 자산으로 자리매김하고 있습니다.

기상청은 국민 누구나 기상기후데이터를 쉽고 편리하게 찾아 활용할 수 있도록 2015년 105종의 데이터를 수록한 『기상기후데이터 카탈로그』를 처음 발간하였습니다. 이후 데이터 개방 확대와 제공체계 정비에 발맞추어 2018년 196종, 2020년 270종으로 수록 범위를 넓혀 왔으며, 이번에 발간하는 『2026년 기상기후데이터 카탈로그』에는 307종을 수록하였습니다.

이번에 발간하는 『2026년 기상기후데이터 카탈로그』는 그동안 달라진 데이터 제공환경과 이용자 수요를 반영하여 새롭게 개정한 것입니다. 기상관측, 기상위성, 레이더, 세계기상전문(GTS), 기상예보, 수치모델, 기후, 응용기상, 지진·화산, 기상연구, 기후·지진 통계, 역사기록, 메타데이터 등 분야별로 체계적으로 정리하였습니다.

이번 개정판은 단순한 데이터 목록 안내를 넘어, 이용자가 자료의 성격과 활용 가능성을 한눈에 파악할 수 있도록 구성하였습니다. 첫째, 데이터명, 데이터요소, 제공범위, 제공기간, 생산·갱신주기, 제공형식, 제공경로 등 핵심 정보를 최신 내용으로 반영하였습니다. 둘째, 생산 종료된 자료는 현행 제공자료와 구분하여 수록함으로써 이용자의 혼선을 줄이는 동시에 과거자료의 보존·활용 가치도 함께 담았습니다. 셋째, 분야별 활용사례를 새롭게 추가하여 데이터를 실제 업무와 연구에 어떻게 활용할 수 있는지 구체적으로 안내하였습니다.

이 카탈로그가 국민, 공공기관, 산업계, 학계는 물론 AI 데이터 활용 분야에 이르기까지 다양한 이용자가 필요한 기상기후데이터를 정확하게 찾고 폭넓게 활용하는 든든한 길잡이가 되기를 기대합니다. 기상청은 앞으로도 기상기후데이터의 접근성과 활용성을 높이기 위해 카탈로그를 지속적으로 정비하고, 이용자의 목소리에 귀 기울여 나가겠습니다.

2026년 10월
국가기후데이터센터


기상관측


기상위성


세계기상전문(GTS)


레이더


기상예보


수치모델


기후


응용기상


지진화산


기상연구


기후지진통계


역사기록


메타데이터


데이터 활용사례

자료 이용 안내

이 책에서 필요한 자료를 찾은 뒤 제공경로의 웹사이트에서 조회 조건과 이용조건을 확인해 주세요. 제공기간은 자료 생산·보유기간과 서비스에서 조회할 수 있는 기간이 다를 수 있으며, 지점·요소·자료주기별로도 차이가 있습니다.

- 웹 조회·다운로드: 제공 화면에서 기간·지점·영역·요소와 자료주기를 선택합니다. 장기간 자료는 파일셋이나 자료 신청 경로를 이용하며, 다운로드 한도와 로그인 필요 여부는 해당 화면을 확인해 주세요.

- API 이용: 기상청 API허브(https://apihub.kma.go.kr) 또는 공공데이터포털(https://www.data.go.kr)에서 회원가입·인증키 발급과 해당 API의 활용신청을 진행합니다. API허브의 인증키 요청인자는 authKey입니다. 공공데이터포털의 인증키·요청인자는 그 서비스의 명세를 적용하며 서로 바꾸어 쓰지 않습니다.

- 신청형 자료: 공공데이터 제공신청이나 보유부서 문의는 즉시 다운로드하는 경로가 아닙니다. 자료명·기간·지점 또는 영역·요소·형식·이용목적을 기재하고, 보유 여부·제공 가능 범위와 이용조건을 안내받아 이용해 주세요.

- 시각과 단위: 국내 관측·예보에도 UTC를 사용하는 자료가 있습니다. KST는 UTC보다 9시간 빠르며, 자료마다 명세의 시간 기준을 적용합니다. 관측·발표·모델 기준시각·예측 유효시각을 구분하고 단위·누적기간·좌표계를 확인해 주세요.

- 오류와 결측: HTTP 응답이 성공해도 본문의 결과코드나 자료 유무를 확인합니다. 결측값·품질표지와 스케일계수는 해당 명세를 따릅니다. 현상이 없거나 제공 계절이 아닌 경우의 빈 자료와 통신·권한 오류를 구분하여야 합니다.

- 자료 상태와 출처: 생산 종료된 자료는 과거자료의 보유·공개 범위 안에서 이용합니다. 예측자료·관측자료·기후변화 시나리오·통계자료를 구분하고, 보고서나 결과물에는 제공기관(기상청)·자료명 등 출처를 밝혀 주세요.

기상기후데이터
카탈로그

CONTENTS



데이터

01. 기상관측 19

1.지상	종관기상관측(ASOS)	20
	방재기상관측(AWS)	21
	농업기상관측(AAOS)	22
	북한기상관측	23
	공공기관기상관측	24
	계절관측	25
	날씨 제보	26
	황사관측(PM10)	27
	낙뢰관측자료	28
2.해양	해양기상부이	29
	등표기상관측	30
	파고부이	31
	연안방재관측	32
	표류부이	33
3.고층	레윈존데	34
	연직바람관측	35
4.항공	공항기상관측(AMOS)	36
	저층급변풍경고(LLWAS)	37
	공항기상라이다(TDWL)	38
5.황사	LIDAR	39

02. 기상위성 41

1.정지궤도 기상위성	(생산종료)천리안위성 1호 기본/기상산출물(L1,L2)	42
	천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)	43
	천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)	44
	HIMAWARI9 기본/기상산출물(L1,L2)	45
	(생산종료)HIMAWARI8 기본/기상산출물(L1,L2)	46
	(생산종료)FY-2B 기본산출물(L1)	47
	(생산종료)FY-2C 기본/기상산출물(L1,L2)	48
	(생산종료)FY-2D 기본산출물(L1)	49
	(생산종료)FY-2E 기본산출물(L1)	50
	FY-2H 기본산출물(L1)	51
	FY-4A 기본/기상산출물(L1,L2)	52
	FY-4B 기본산출물(L1)	53

2.저궤도 기상위성	(생산종료)NOAA-18 기본/기상산출물(L1,L2)	54
	(생산종료)NOAA-19 기본/기상산출물(L1,L2)	55
	NOAA-20 기본/기상산출물(L1,L2)	56
	NOAA-21 기본산출물(L1)	57
	S-NPP 기본/기상산출물(L1,L2)	58
	(생산종료)METOP-A 기본/기상산출물(L1,L2)	59
	METOP-B 기본산출물(L1)	60
	METOP-C 기본산출물(L1)	61
	AQUA 기본/기상산출물(L1,L2)	62
	TERRA 기본/기상산출물(L1,L2)	63
3.기타	GNSS 기본/기상산출물(총지연량,가강수량)	64

03. 레이더 65

1.사이트	원시자료	66
	품질관리자료(QC)	67
	대기수상체(HCI)	68
	다중 고도각 기반 레이더 추정 강수량(HSR)	69
2.합성	레이더 강수량	70
	레이더 강수량 실황예측	71
	레이더 바람장	72
	레이더 우박	73
	레이더 수상체	74
	레이더 눈/비	75
	레이더 3차원	76



CONTENTS



04. 세계기상전문(GTS) 77

1.관측	세계기상전문(GTS) 관측	78
2.예특보	세계기상전문(GTS) 예특보	80

05. 기상예보 81

1.날씨예보	초단기실황	82
	초단기예보	83
	단기예보	84
	중기예보	85
	영향예보	86
2.기상특·정보	기상특보	87
	기상정보	88
3.태풍예보	태풍정보	89
	태풍목록	90
	태풍 경로도	91
	태풍 발생표	92
	태풍 베스트트랙	93
	열대저압부정보	94
4.항공예보	공항예보(TAF, 국내)	95
	이륙예보	95
	SIGWX(중요기상예보)	96
	SIGMET(위험기상정보)	97
	AIRMET(저고도 위험기상정보)	97
	공항경보(WARNING)	98
5.편집일기도	편집일기도(지상 24시간 예상)	99
	편집일기도(지상 12시간)	100
	(생산종료)편집일기도(지상 3시간)	101
	(생산종료)편집일기도(고층)	102
	(생산종료)편집일기도(해상 24시간 예상)	103

06. 수치모델 105

1.수치분석 일기도	분석일기도(3시간, 지상)	106
	분석일기도(3시간, 지상기압변화)	107
	분석일기도(6시간, 지상)	108
	분석일기도(보조분석도)	109
	분석일기도(6시간, 고층)	110
	분석일기도(단열선도)	111
	분석일기도(3시간, 황사)	112
	분석일기도(한반도 일기도)	113
	분석일기도(상세실황분석장)	114
2.단·중기모델	전지구예보모델(KIM-GDAPS) 예측자료	115
	전지구예보모델(KIM-GDAPS) 수치예상일기도	116
	국지예보모델(KIM-LDAPS) 예측자료	117
	국지예보모델(KIM-LDAPS) 수치예상일기도	118
	지역예보모델(KIM-RDAPS) 예측자료	119
	지역예보모델(KIM-RDAPS) 수치예상일기도	120
	(생산종료)전지구예보모델(UM-GDAPS) 예측자료	121
	(생산종료)전지구예보모델(UM-GDAPS) 수치예상일기도	122
	(생산종료)전지구예보모델(UM-GDAPS) 지역영역 예측자료 ((구)RDAPS 영역)	123
	(생산종료)국지예보모델(UM-LDAPS) 예측자료	124
	(생산종료) 국지예보모델(UM-LDAPS) 수치예상일기도	125
	(생산종료)지역예보모델(UM-RDAPS) 예측자료	126
	(생산종료) 지역예보모델(UM-RDAPS) 수치예상일기도	127
	(생산종료)지역예보모델(KWRF) 예측자료	128
	(생산종료)지역예보모델(KWRF) 수치예상일기도	129



기상기후데이터
카탈로그

CONTENTS



3.초단기모델	초단기모델(KLAPS) 분석(KL05) 자료	130
	초단기모델(KLAPS) 분석(KL05) 수치분석일기도	131
	초단기모델(KLAPS) 예측(KLFS) 자료	132
	초단기모델(KLAPS) 예측(KLFS) 수치예상일기도	133
	(생산종료)초단기모델 (VDAPS) 수치예상일기도	134
4.앙상블모델	(생산종료)전지구앙상블모델(UM-EPSG) 예측자료	135
	(생산종료)전지구앙상블모델(UM-EPSG) 수치예상일기도	136
	(생산종료)국지앙상블모델(UM-LENS) 예측자료	137
	(생산종료)국지앙상블모델(UM-LENS) 수치예상일기도	138
	전지구앙상블모델(KIM-EPSG) 예측자료	139
	전지구앙상블모델(KIM-EPSG) 수치예상일기도	140
	국지앙상블모델(KIM-LENS) 예측자료	141
	국지앙상블모델(KIM-LENS) 수치예상일기도	142
5.파랑모델	전지구파랑모델(GWW3) 예측자료	143
	전지구파랑모델(GWW3) 수치예상일기도	144
	지역파랑모델(RWW3) 예측자료	145
	지역파랑모델(RWW3) 수치예상일기도	146
	지역앙상블파랑모델(EWW3) 예측자료	147
	지역앙상블파랑모델(EWW3) 수치예상일기도	148
	국지연안파랑모델(CWW3) 예측자료	149
	국지연안파랑모델(CWW3) 수치예상일기도	150
	초단기파랑모델(KWW3) 예측자료	151
	초단기파랑모델(KWW3) 수치예상일기도	152
6.해구별예측	대해구 예측정보	153
	소해구 예측정보	154
7.폭풍해일모델	지역폭풍해일모델(RTSM) 예측자료	155
	지역폭풍해일모델(RTSM) 수치예상일기도	156
	국지연안폭풍해일모델(CTSM) 수치예상일기도	157
	지역앙상블폭풍해일모델(ETSM) 예측자료	158
	지역앙상블폭풍해일모델(ETSM) 수치예상일기도	159

8.황사모델	황사·연무모델(ADAM3) 예측자료	160
	황사·연무모델(ADAM3) 수치예상일기도	161
	(생산종료)황사모델(ADAM2) 예측자료	162
	(생산종료)황사모델(ADAM2) 수치예상일기도	163
9.기후모델 (GLOS)	기후예측시스템(GloSea) 예측자료(GRIB)	164
	기후예측모델(GloSea) 전지구 예측일기도(이미지)	165
10.해양순환 모델 (MOHID)	지역·국지 해양순환모델 예측자료	166
11.항공모델	한국형 항공난류 예측자료	167
	전지구 한국형 항공난류 예측자료	168
	제주공항 특화 예측일기도	169



CONTENTS



07. 기후 171

1.지구대기감시	온실가스	172
	반응가스	173
	에어로졸	174
	대기복사	175
	성층권오존	176
	자외선	177
	총대기침적	178
	총자외선지수(관측)	179
2.기후변화 시나리오	SRES 시나리오	180
	RCP 시나리오	181
	SSP 시나리오	183
3.기후예측	1개월 전망	185
	3개월 전망	186
	기후전망	187
4.수문기상	유역별 강수량	188
	유역별 증발산량	189
	유역별 토양수분	190
	유역별 강수량 예측정보	191
	토양수분량·증발산량 예측정보	192
5.가뭄정보	표준강수지수	193
	강수평년비	194
	기상가뭄예보	195
	표준강수증발산지수	196
6.해양정보	이안류 위험 예측정보	197
	너울 위험 예측정보	198

08. 응용기상 199

1.기상지수	생활기상지수	200
	보건기상지수	202
2.융합서비스	농업주산지 기상정보	204
	관광코스별 기상정보	205
	기상청 도로날씨정보	206
	서리발생예측정보	207
	전력기상지수	208
	도로위험기상정보	209

3.기상자원지도	위성기반 태양광자원지도	210
	풍력자원지도	211
4.항공	레이더 급변풍 탐지 일기도	212

09. 지진화산 213

1.지진화산 특·정보	지진해일특보	214
	지진해일정보	215
	지진해일 상세정보	216
	지진정보	217
	미소지진	218
	지진속보	219
	지진조기경보	220
	국외지진조기경보	221
	국외지진정보	222
	화산정보·화산재특보	223

10. 기상연구 225

1.표준기상	보성표준기상관측소 연직관측	226
2.연구용 관측 캠페인	(생산종료)하계집중관측	227
	(생산종료)강원영동집중관측	228
	(생산종료)평창집중관측	229
3.해양기상	기상관측선 관측	230
	아르고 플로트 관측	231
4.지진화산	지구자기	232
5.응용기상	도시기상관측	233
	꽃가루농도 수동관측	234
	산림연료·토양수분관측	235
	농업서리자동관측	236
	흑구온도	237
6.구름물리	구름물리관측	238
	챔버실험(구름물리챔버관측)	239

CONTENTS



11. 기후지진통계 241

1.통계연월보	일기상통계표	242
	기상월표원부	243
	기상연표원부	244
	기상월보	245
	기상연보	246
	방재기상관측월보	247
	방재기상관측연보	248
	농업기상월보	249
	해양기상월보	250
	고층기상월보	251
	고층기상연보	252
	항공기상월보	253
	항공기상연보	254
	낙뢰연보	255
	북한기상연보	256
	지진연보	257
2.평년값	우리나라 기후평년값	258
	북한 기후평년값	259
	세계 기후평년값	260
3.통계분석	바람장미	261
	강수일수	262
	황사일수	263
	폭염일수	264
	열대야일수	265
	24절기	266
	순위값	267
	장마	268
4.응용기상	체감온도	269
	적산온도	270
	ICAO 데이터 카탈로그 (공항표준온도, 공항평균최저온도)	271
5.간행물	한국기후표	273
	한국기후도	274
	북한기상20년보	275
	북한기상30년보	276
	장마백서	277
	태풍백서	278
	항공기후정보	279
	지구대기감시보고서	280

12. 역사기록 281

1.자기기록지 자기기록지 282
자기기록지설명자료 283

2.종이일기도 종이일기도 284

3.역사자료 해관기상관측자료(19~20세기) 285
측우기자료(조선시대) 286
조선왕조실록 기상기록 287
각사등록(조선시대) 288
역사지진 기록 289

4.기상기록집 기상천문지진기록 290
한반도 역사지진기록 291
관상감에 기록한 17세기 밤하늘 292
기상인이 말하는 중앙관상대 293
농사와 측우기 기록 294

13. 메타데이터 295

1.지점정보 관측지점정보 296
기상예보구역코드 297
기상특보구역코드 298

2.데이터정보 데이터 카탈로그 299
기후통계 메타데이터 300

기상기후데이터
카탈로그

CONTENTS



데이터 활용사례 50

기상관측 303

지상
1. 종관기상관측(ASOS) - 306
1-①기온·강수 특성과 냉난방도일 분석 306
1-②기온·기압·시정의 일변화 분석 311
2. 방재기상관측(AWS) - 집중호우와 강풍의 지역 분포 분석 319
3. 계절관측 - 지역별 개화 시기 비교 327
4. 황사관측(PM10) - 관측망별 농도 변화 비교 332
5. 낙뢰관측자료 - 발생 위치와 시간대 분석 337

해양
6. 해양기상부이 - 해역별 파고와 바람 비교 343
7. 파고부이 - 지점별 파고 변화 비교 348

고층
8. 레윈존데 - 353
8-①고도별 기온·습도·바람 분석 353
8-②대기 불안정도 비교 358
9. 연직바람관측 - 시간·고도별 바람 변화 분석 366

항공
10. 공항기상관측(AMOS) - 시정·운고·바람 변화 분석 370
11. 공항기상라이다(TDWL) - 공항 주변 바람 분포 분석 377

기상위성 387

정지궤도 기상위성
12. 천리안위성 2A호(GK2A) - 388
12-①적외영상과 안개 분포 비교 388
12-②산불위험도와 화점 분포 분석 394
13. 천리안위성 2A호(GK2A) - 우주 입자와 자기장 변화 분석 403

레이더 413

사이트
14. 원시자료 – 관측 범위와 빔고도 확인 414
15. 레이더 추정 강수량(HSR) - 반사도·에코고도·강수량 비교 419

합성
16. 레이더 강수량 – 격자별 강수강도 분석 429

세계기상전문(GTS) 437

관측
17. 지상(SYNOP) - 지역별 관측값 비교 438
18. 공항 정시기상관측보고(METAR) - 공항별 시정과 구름 상태 비교 444

기상예보 455

날씨예보
19. 초단기실황 – 지역별 현재 날씨 비교 456
20. 초단기예보 20-①시간별 강수 변화 분석 461
20- ②기온·강수·바람 공간 분포 비교 468
21. 단기예보 – 야외활동 시간대별 날씨 비교 479
22. 중기예보 – 날짜별 기온과 강수확률 비교 488
23. 영향예보 – 지역·분야별 위험수준 비교 496

기상특·정보
24. 기상특보 – 지역별 발표 현황 확인 502

태풍예보
25. 태풍정보 – 이동 경로와 접근 시간 분석 506
26. 태풍 베스트트랙 – 과거 태풍 경로와 강도 비교 514

항공예보
27. 공항예보(TAF, 국내) - 공항별 바람과 비행기상 변화 비교 520

수치모델 527

단·중기모델
28. 전지구예보모델(KIM-GDAPS) 예측자료 – 예측시간별 기온·바람 비교 528
29. 국지예보모델(KIM-LDAPS) 예측자료 536
29-①경로별 기온 연직단면 분석 536
29-②지역별 예측요소 분포 분석 542
29-③격자 연직 프로파일 552

초단기모델
30. 초단기모델(KLAPS) 분석(KL05) 자료 – 예측시간별 강수 분포 비교 560

파랑모델
31. 지역파랑모델(RWW3) 예측자료 – 과거 해구별 파고와 바람 비교 563

해구별예측
32. 대·소해구 예측정보 – 과거 해구별 파고 변화 비교 570

기후 575

지구대기감시
33. 온실가스 – 감시소별 농도 변화 비교 576
34. 자외선 – 시간대별 자외선 변화와 누적량 분석 579

기후변화 시나리오
35. SSP 시나리오 – 미래 해수면온도 변화 비교 588

가뭄정보
36. 표준강수지수 – 지역별 가뭄 정도 비교 591

CONTENTS



응용기상 595

기상지수
37. 자외선지수 – 지역별 자외선 위험단계 비교 596
38. 꽃가루농도위험지수 – 지역·종류별 위험단계 비교 602

융합서비스
39. 기상청 도로날씨정보 – 노선별 안개·비·눈 분포 분석 607
40. 서리발생예측정보 – 지역별 서리 발생확률 비교 614
41. 전력기상지수 – 지역별 지수 변화와 체감온도 비교 618
42. 도로위험기상정보 – 노선별 위험구간 확인 623

지진화산 629

지진화산 특정보
43. 지진해일정보 – 특보 발표 이력과 대상지역 확인 630
44. 지진정보 – 발생 위치·규모·진도 분석 633

기상연구 639

해양기상
45. 기상관측선 관측 – 항적별 해양기상 변화 분석 640

기후지진통계 647

평년값
46. 우리나라 기후평년값 – 지역별 기후 특성과 평년기간별 변화 비교 648

통계분석
47. 강수일수 – 지역별 발생 빈도와 장기 변화 분석 653
48. 폭염일수 – 지역별 발생 빈도와 장기 변화 분석 660
49. 열대야일수 – 일최저기온 참고지표 비교 667

메타데이터 675

지점정보
50. 관측지점정보 – 관측망 분포와 지점 간 거리 분석 676

기상기후데이터 카탈로그

데이터

Data



01

기상관측

Meteorological Observation



1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

종관기상관측(ASOS)



개요

자료상태 : 현행

종관기상관측이란 정해진 시각의 대기 상태를 파악하기 위해 모든 관측소에서 같은 시각에 실시하는 지상관측을 말합니다. 기온, 강수, 습도, 풍향, 풍속 등의 대기 상태를 종관기상관측장비(ASOS, Automated Synoptic Observing System)를 이용해 자동으로 관측하며 구름, 일기현상 등 일부 요소는 목측으로 관측합니다.

주요요소	기온, 강수, 기압, 습도, 풍향, 풍속, 일사, 일조, 적설, 구름, 시정, 지면·초상온도 등	
제공범위	105지점 ※ 기상자료개방포털 제공 기준('26.8.), 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	1904년 4월 ~ 현재(지점별로 다름)	
생산·갱신주기	분, 시간, 일, 월, 연 자료	
제공형식	CSV(포털 다운로드), ASCII·XML·JSON(API별 응답형식 상이)	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 종관기상관측(ASOS)
	XML	기상청 API허브(https://apihub.kma.go.kr) → 지상관측 → 종관기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. API를 이용할 때에는 시간·일·월자료 등 목적에 맞는 서비스를 선택합니다.	
간행물	기상월보, 기상연보를 각각 매월·매년 정기적으로 발간 매 10년마다 지난 30년에 대해 한국기후표, 한국기후도를 발간	

방재기상관측(AWS)



개요

자료상태 : 현행

방재기상관측이란 태풍 · 홍수 · 가뭄 등 기상현상에 따른 자연재해를 막기 위해 실시하는 지상관측을 말합니다. 관측 공백 해소 및 국지적인 기상 현상을 파악하기 위하여 자동기상관측장비(AWS, Automatic Weather System)를 설치하여 자동으로 관측합니다.

주요요소	기온, 강수, 풍향, 풍속 등	
제공범위	554지점 ※ 기상자료개방포털 제공 기준('26.8.), 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	1989년 7월 ~ 현재(지점별로 다름)	
생산·갱신주기	분, 시간, 일, 월, 연 자료	
제공형식	CSV(포털 다운로드), ASCII 등(API별 응답형식 상이)	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 방재기상관측(AWS)
	XML	기상청 API허브(https://apihub.kma.go.kr) → 지상관측 → 방재기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. 전국 지점(stn=0)의 AWS 매분자료는 최대 10분 구간으로 나눠 요청합니다. 1개 지점은 하루 이내로 요청하고, 결합 시 중복 시각을 제거합니다. 요청시각은 KST를 사용합니다.	
간행물	방재기상월보, 방재기상연보를 각각 매월·매년 정기적으로 발간	

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

농업기상관측(AAOS)



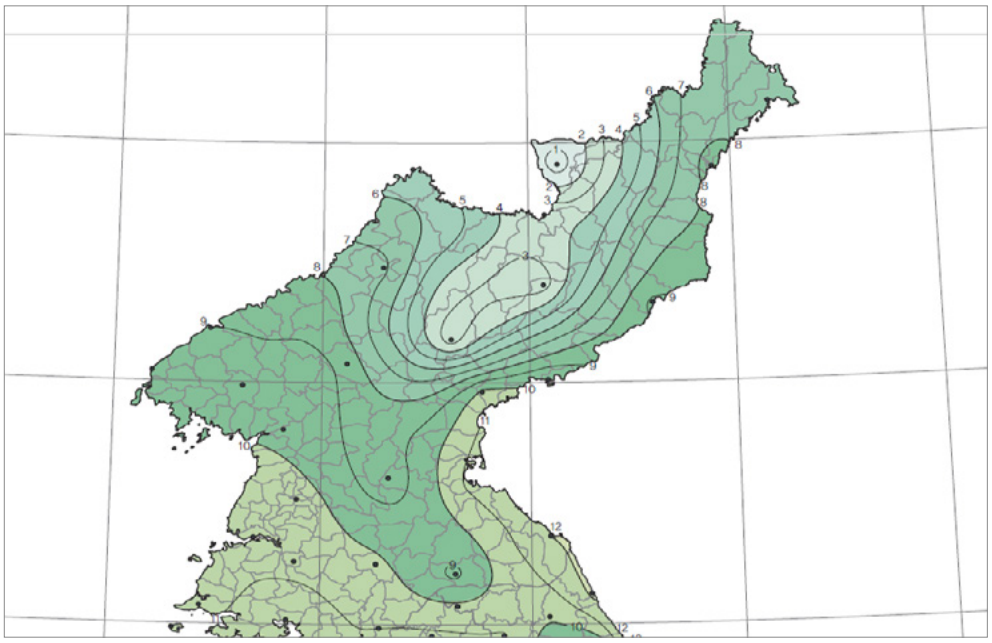
개요

자료상태 : 현행

농업기상관측이란 자연환경에서 시시각각으로 변하는 기상요소 중 농업에 필요한 요소를 관측하는 것을 말합니다. 특히, 농작물 생육에 영향을 주는 토양 수분, 지중온도, 복사 등을 농업기상관측장비(AAOS, Automated Agricultural Observing System)를 이용하여 관측합니다.

주요요소	기온, 습도, 풍속, 지면·초상·지중온도, 토양수분, 일사, 일조 등
제공범위	11지점 ※ 기상자료개방포털 제공 기준('26.8.)
제공기간	2002년 9월 ~ 현재(지점별로 다름)
생산·갱신주기	시간, 일 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 농업기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. 토양 관련 요소는 측정 깊이와 관측방법을 확인합니다.
간행물	농업기상월보(2018년 1월 이후 발간 중단)

북한기상관측



개요

자료상태 : 현행

세계기상통신망(GTS)을 통해 수집한 북한 지역 관측지점의 기상자료입니다. 자료 수신 상황에 따라 누락이나 지연이 있을 수 있으며, 기상청 국내 관측자료와 관측 주기·요소가 다를 수 있습니다.

주요요소	기온, 강수, 바람, 기압, 습도, 구름, 시정
제공범위	27지점 ※ 기상자료개방포털 제공 기준('26.8.), 폐쇄 지점 포함으로 현행 지점 수와 차이 있음
제공기간	1973년 1월 ~ 현재(지점별로 다름)
생산·갱신주기	시간, 일 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 북한기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. GTS 자료의 UTC 시각과 강수 누적기간을 확인합니다.

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

공공기관 기상관측



개요

자료상태 : 현행

「기상관측표준화법」에 따라 기상관측업무를 수행하는 기관에서 생산하는 기상관측자료입니다. 2026년 현재 국가기관, 지방자치단체, 공공기관 등 26개 관측기관 4,530개소의 기상관측시설의 기상관측자료를 기상자료개방포털을 통하여 통합 제공하고 있으며 '기상관측망 구축 및 관리계획'에 근거한 기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동됩니다.

주요요소	기온, 풍향, 풍속, 습도, 기압, 강수량
제공범위	4,530지점 ※ 기상관측업무를 수행하는 국가기관, 지자체, 공공기관 운영 기상관측시설
제공기간	2009년 9월 ~ 현재(지점별로 다름)
생산·갱신주기	분(1·2·5·10·60분) 자료 ※ 관측기관별 관측목적에 따라 관측자료 생산주기 차이
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 공공기관기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. 여러 기관의 자료를 결합할 때 기관·지점 식별자를 함께 보관합니다.

계절관측



개요

자료상태 : 현행

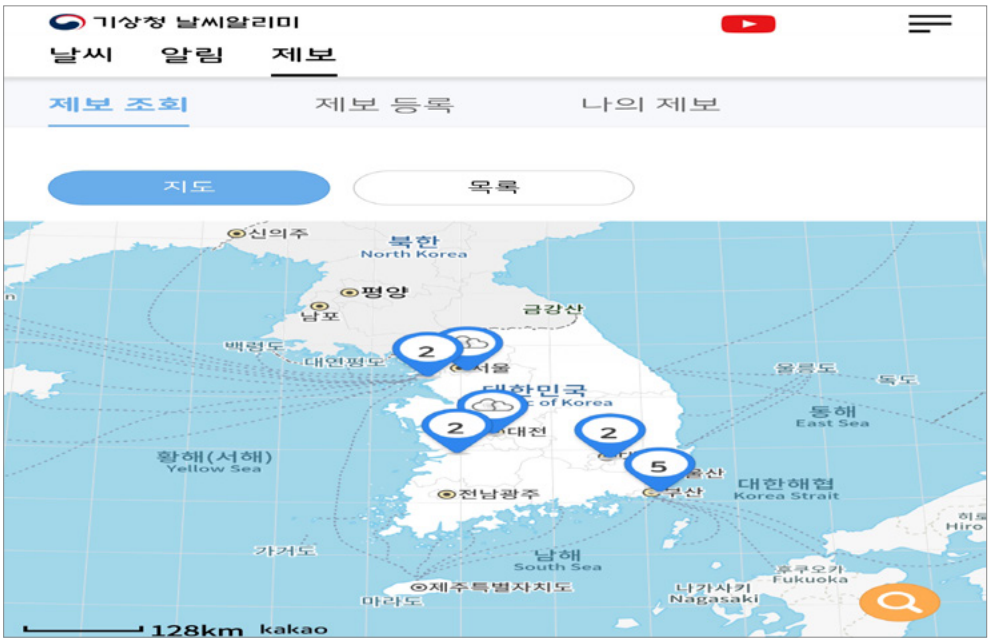
계절관측이란 계절을 대표할 수 있는 현상을 관측하고, 통계 분석하여 기후의 변화 추이를 파악하기 위해 관측장소에서 관측자가 지정된 식물, 동물, 기후계절을 관측하는 것을 말합니다. 2026년 기준 23개 기상관서에서 식물 10종, 동물 6종, 기후계절 5종을 관측하고 있으며, 군락지(18곳), 유명산 단풍(21곳)도 매년 관측하고 있습니다.

주요요소	• 식물계절관측: 개나리, 진달래, 벚나무, 단풍나무 등, 군락지 3종, 유명산 단풍 • 동물계절관측: 제비, 개구리, 나비, 잠자리, 빼꾸기, 매미 • 기후계절관측: 눈, 서리, 얼음, 관설, 강·하천 결빙 및 해빙
제공범위	87지점 ※ 기상자료개방포털 제공 기준('26.8.), 관측 중지 지점 포함으로 현행 지점 수와 차이 있음
제공기간	1904년 ~ 현재(요소별, 지점별로 다름)
생산·갱신주기	연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 계절관측
이용방법	대상 생물·현상, 지점과 연도를 선택해 관측일을 조회합니다. 개화·만발·단풍 등의 정의를 구분하고, 미관측과 현상 미발생을 같은 값으로 처리하지 않습니다.

1.지상
• 종관기상관측(ASOS)
• 방재기상관측(AWS)
• 농업기상관측(AAOS)
• 북한기상관측
• 공공기관기상관측
• 계절관측
• 날씨 제보
• 황사관측(PM10)
• 낙뢰관측자료

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

날씨 제보



개요

자료상태 : 현행

날씨 제보란 국민이 일상생활에서 직접 경험한 비, 눈, 안개, 황사 등의 날씨 현상과 계절 현상을 사진·동영상 등으로 제보하고 공유하는 자료를 말합니다. 날씨 현상, 계절 제보 및 지진제보로 구성되며, 날씨알리미 앱을 통해 운영하고 있습니다.

주요요소	• 날씨 제보 28종(비, 눈, 황사 등) • 계절 제보 2종(벚나무, 단풍) • 지진제보(진도 12단계)
제공범위	전국
제공기간	날씨알리미 앱 통합 운영(2021.1.~)
생산·갱신주기	수시(국민 참여에 따른 실시간 제보)
제공형식	사진, 동영상, 텍스트
제공경로	기상청 날씨알리미 앱 > 제보
이용방법	날씨알리미 앱의 제보 메뉴에서 위치·시각·현상과 사진 등을 등록하거나 제공 화면을 확인합니다. 제보의 위치·시각과 사실 여부를 확인하여 참고자료로 이용합니다.

황사관측(PM10)



개요

자료상태 : 현행

황사관측(PM10)은 대기 중에 부유하는 에어로졸 중 직경이 10㎛ 이하인 입자의 농도를 연속 측정합니다. 먼지(황사 포함)가 필터에 침적되고, 동위원소 C-14에서 방출되는 베타선을 필터 여지에 쏘아 감쇄된 베타선을 검출기로 측정하여 황사의 농도를 산출합니다.

주요요소	부유분진 농도
제공범위	27지점 ※ 기상자료개방포털 제공 기준('26.8.), 운영 종료 지점 포함으로 현행 지점 수와 차이 있음
제공기간	2003년 4월 ~ 현재(지점별로 다름)
생산·갱신주기	분(5분 주기), 시간 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 황사관측(PM10) 기상청 API허브(https://apihub.kma.go.kr) → 지상관측 → 황사관측(PM10)
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. 황사 분석에는 황사 관측기록과 기상상황을 함께 확인합니다.

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

1.지상
- 종관기상관측(ASOS)
- 방재기상관측(AWS)
- 농업기상관측(AAOS)
- 북한기상관측
- 공공기관기상관측
- 계절관측
- 날씨 제보
- 황사관측(PM10)
- 낙뢰관측자료

낙뢰관측자료



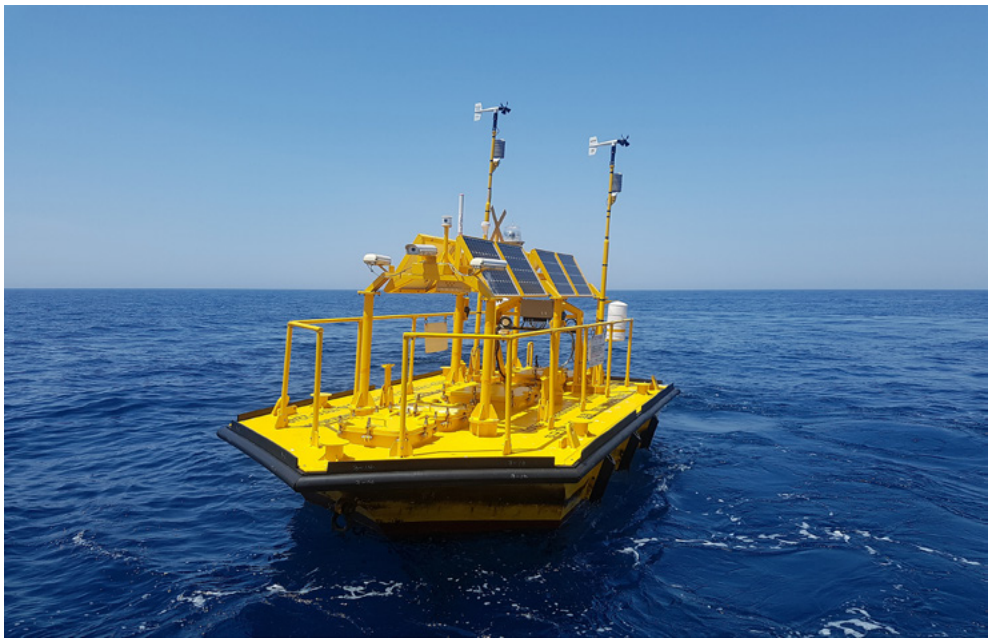
개요

자료상태 : 현행

발달하는 적란운 구름 내부에 분리 축적된 음전하와 양전하 사이에서 번개현상이 발생하고 구름 하단과 지면 사이에 발생하는 방전 현상을 낙뢰라고 합니다. 기상청 낙뢰관측자료는 전국 21개소 낙뢰관측장비를 이용해서 탐지·분석된 낙뢰의 발생시각, 위치, 극성(+, -), 강도 등을 관측한 자료입니다.

주요요소	관측시각, 위경도, 강도, 낙뢰감지센서수, 낙뢰타입 등	
제공범위	한반도 주변 낙뢰 정보(기상청 낙뢰관측소 21개소 정보를 분석하여 생산) ※ 낙뢰관측소(21개소): 백령도, 파주, 인천, 서해기지, 보령, 춘천, 충주, 간성, 대관령, 울진, 울릉도, 추풍령, 포항, 고창, 거창, 흑산도, 완도, 여수, 부산, 제주, 서귀포	
제공기간	1988년 1월 ~ 현재	
생산·갱신주기	분(5분 주기) 자료	
제공형식	PNG, CSV	
제공경로	PNG	공공데이터포털(https://www.data.go.kr) → '낙뢰분포도 조회서비스' 검색
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 낙뢰관측
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 지상 → 낙뢰관측
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 낙뢰관측
이용방법	기간·영역을 설정해 낙뢰 발생자료 또는 분포영상을 조회합니다. 시각 기준, 좌표·극성·강도 단위를 확인하고 개별 탐지자료와 시간별 집계영상의 차이를 구분합니다.	

해양기상부이



개요

자료상태 : 현행

해양기상부이는 해상에서 해양기상현상을 측정하고, 측정한 값을 통신장비를 이용해 전송하는 장비입니다. 파고와 파주기는 해수면에서 부이 몸체의 움직이는 가속도를 측정하여 스펙트럼 방법을 통해 유의파고와 파주기 등을 산출합니다.

주요요소	기온, 풍향, 풍속, 기압, 습도, 파고, 수온 등	
제공범위	49지점 ※ 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	1996년 7월 ~ 현재(지점별로 다름)	
생산·갱신주기	시간, 일 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 해양 → 해양기상부이
	XML	기상청 API허브(https://apihub.kma.go.kr) → 해양관측 → 해양기상부이·파고부이관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다. 파랑 통계량을 비교할 때 유의·최대·평균파고를 구분합니다.	

2.해양

- 해양기상부이
- 등표기상관측
- 파고부이
- 연안방재관측
- 표류부이

등표기상관측



개요

자료상태 : 현행

등표기상관측장비는 등표나 관측탑 등의 해양 구조물에 기상관측장비를 설치하여 관측기기에서 측정한 값을 통신장비를 이용하여 전송하는 장비입니다. 해양의 자연물 또는 구조물을 이용함에 따라 발생하는 구조적 관측환경으로부터 영향을 최소화할 수 있는 곳에 설치합니다.

주요요소	기온, 풍향, 풍속, 기압, 습도	
제공범위	10지점 ※ 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	2001년 12월 ~ 현재(지점별로 다름)	
생산·갱신주기	시간, 일 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 해양 → 등표기상관측
	XML	기상청 API허브(https://apihub.kma.go.kr) → 해양관측 → 등표기상관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다.	

파고부이



2.해양
• 해양기상부이
• 등표기상관측
• 파고부이
• 연안방재관측
• 표류부이

개요

자료상태 : 현행

파고부이는 해양기상부이 설치가 용이하지 않거나, 지형적으로 복잡한 연안바다에서 국지적으로 해면상태를 측정하고 측정한 값을 통신장비를 이용하여 전송하는 장비입니다. 파고를 측정하기 위해 가속도계로 구성된 파고계를 몸체 중심부에 설치하여 측정합니다.

주요요소	수온, 파고, 파주기	
제공범위	84지점 ※ 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	2009년 1월 ~ 현재(지점별로 다름)	
생산·갱신주기	시간, 일 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 해양 → 파고부이
	XML	기상청 API허브(https://apihub.kma.go.kr) → 해양관측 → 해양기상부이·파고부이관측
이용방법	관측지점·기간·요소와 자료주기를 선택해 조회·다운로드합니다. 지점 변경 이력, 단위와 결측·품질표지를 확인하고 분석에서 결측을 0으로 처리하지 않습니다.	

2.해양

- 해양기상부이
- 등표기상관측
- 파고부이
- 연안방재관측
- 표류부이

연안방재관측



개요

자료상태 : 현행

연안방재관측는 서해안의 기상해일 및 동해안의 너울성 파도 등 장주기파에 의해 발생하는 수난사고 예방을 위한 실시간 해양감시장비입니다. 해안지역의 기상실황감시를 위해 CCTV도 함께 설치되어 있습니다. 마이크로파를 해수면에 수직으로 송신한 뒤 반사되는 파를 수신하여 해수면까지의 거리의 변화(수위의 변화)를 측정하는 방법으로 관측합니다.

주요요소	풍향, 풍속, 기압, 기온, 습도, 수위 등
제공범위	18지점
제공기간	2003년 8월 ~ 현재(지점별로 다름)
생산·갱신주기	분, 시간, 일 자료
제공형식	CSV
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청
이용방법	관측지점·기간·요소를 정해 보유자료를 신청합니다. 해수면 또는 수위의 기준면, 관측간격·단위와 품질표지를 확인하여 연안 기상 변화를 분석합니다.

표류부이



개요

자료상태 : 현행

표류부이는 해양기상관측장비의 설치 및 유지·관리가 어려운 우리나라의 먼바다나 한·중 중간수역, 한·일 중간수역 등에 투하하여 원해 기상관측자료를 생산하고 위성통신을 이용해 자료를 송신합니다. 특히, 태풍진로 추적과 서해안 및 영동지방 겨울철 대설 등 해양 위험기상을 관측하고 감시하는 데 활용합니다.

주요요소	위경도, 유의파고, 파주기, 수온, 기압 등
제공범위	특별관측으로 운영되어 관측위치는 매번 다르며, 이동형 관측장비임
제공기간	2013년 8월 ~ 현재(지점별로 다름)
생산·갱신주기	분(20분 주기) 자료
제공형식	CSV
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청
이용방법	부이 식별번호·관측기간·해역을 정해 보유자료를 신청합니다. 관측시각별 위치를 함께 읽고 품질표지와 단위를 확인합니다.

3.고층

• 레윈존데
• 연직바람관측

레윈존데



개요

자료상태 : 현행

라디오존데를 기구에 매달아 비양시켜 지상으로부터 약 35km까지의 기압, 기온, 풍향·풍속 등을 일정한 시간 간격으로 관측합니다. 관측자료는 무선 송수신장치를 통해 지상으로 전송되고 지상 수신장치에서 처리됩니다.
기상청은 일 2회 정기 고층기상관측(9시, 21시)을 하고 우리나라에 태풍, 집중호우 등과 같은 위험기상 현상이 발생 또는 예측될 때는 정기 관측시각 이외에 3시, 15시에 특별기상관측을 합니다.

주요요소	기압, 고도, 기온, 이슬점온도, 풍향, 풍속	
제공범위	13지점 ※ 기상자료개방포털 제공 기준(2026.8.), 운영 종료 지점 포함으로 현행 지점 수와 차이 있음	
제공기간	1957년 4월 1일 ~ 현재(지점별로 다름)	
생산·갱신주기	시간 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 고층 → 레윈존데
	XML	기상청 API허브(https://apihub.kma.go.kr) → 고층관측 → 레윈존데
이용방법	지점·관측시각·기압면 또는 고도를 선택합니다. 관측시각의 UTC/KST 표기, 고도·기압 단위와 품질표지를 확인하고 연직자료를 기압 또는 고도 순으로 정렬하여 이용합니다.	
간행물	고층기상월보, 고층기상연보를 각각 매월·매년 정기적으로 발간	

연직바람관측



개요

자료상태 : 현행

연직바람관측장비(Wind Profiler Radar)는 초단파, 극초단파(UHF, 300~3000 MHz) 전파빔을 상층대기로 송신하여 바람과 함께 이동하는 난류에서 산란되어 오는 전파신호를 수신해 바람을 관측합니다. 관측자료는 최대빔 송신거리값 차이에 따라 ULC와 UHC로 구분하고, 10분주기로 생성됩니다.

※ ULC: low mode(약 5km 이하), UHC: high mode(약 12km 이하)

주요요소	바람, 고도, 동서·남북·연직 바람성분	
제공범위	15지점	
제공기간	2004년 ~ 현재(지점별로 다름)	
생산·갱신주기	분(10분 주기) 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 고층 → 연직바람관측
		기상청 API허브(https://apihub.kma.go.kr) → 고층관측 → 연직바람관측
	XML	공공데이터포털(https://www.data.go.kr) → '고층정보조회서비스' 검색
이용방법	지점·시각·고도별 바람을 조회합니다. 풍향·풍속 단위, 고도의 기준과 품질표지를 확인한 뒤 연직 바람 분포를 분석합니다.	

4.항공

• 공항기상관측(AMOS)
• 저층급변풍경고(LLWAS)
• 공항기상라이다(TDWL)

공항기상관측(AMOS)





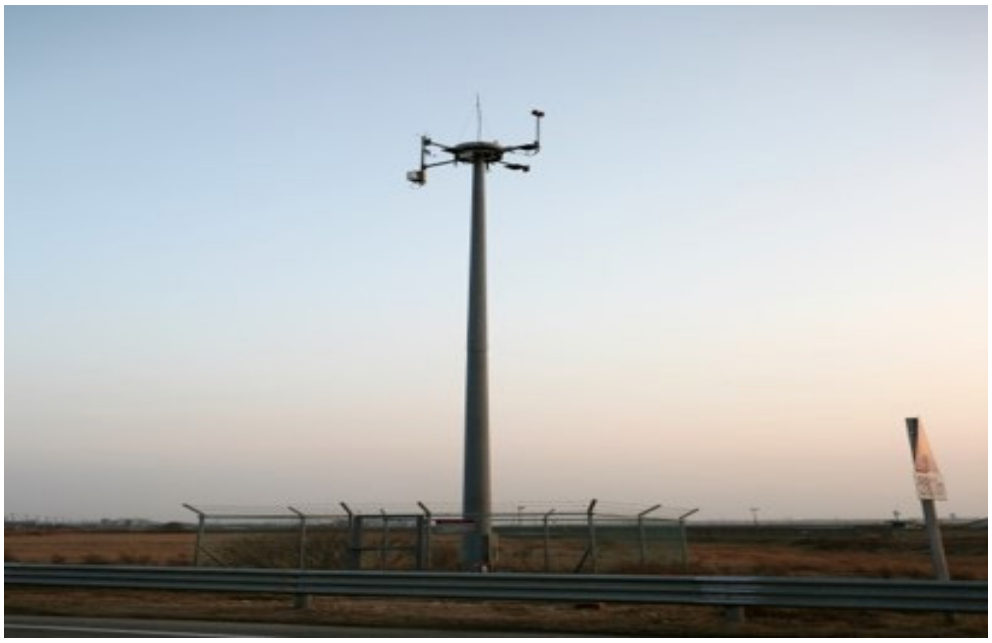
개요

자료상태 : 현행

항공기의 안전한 운항을 지원하기 위해 전국의 7개소 공항에서 공항기상관측을 실시합니다.
공항기상관측장비(Aerodrome Meteorological Observation System)는 항공기의 안전한 이·착륙에 필요한 활주로 부근의 기상실황을 실시간으로 관측하여 제공하는 항공기상관측의 기본 장비입니다.

주요요소	기온, 강수, 바람, 기압, 습도, 적설, 구름, 시정, 일기현상	
제공범위	7개소 공항 ※ 인천, 김포, 제주, 울산, 무안, 여수, 양양	
제공기간	인천공항(113): 2005-01-13 13:15:00 ~ 현재 김포공항(110): 2005-01-13 13:15:00 ~ 현재 제주공항(182): 2003-12-11 00:00:00 ~ 현재 울산공항(151): 2005-01-13 13:14:00 ~ 현재 무안공항(163): 2007-10-11 21:28:00 ~ 현재 여수공항(167): 2005-01-13 13:14:30 ~ 현재 양양공항(92): 2005-01-13 13:14:40 ~ 현재	
생산·갱신주기	분, 시간, 일 자료	
제공형식	CSV, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 항공 → 공항기상관측
	XML	기상청 API허브(https://apihub.kma.go.kr) → 항공기상 → 공항기상관측(AMOS)
이용방법	공항·기간·요소를 선택해 조회합니다. 공항 내 센서 위치와 관측 주기를 확인하고, 전문 자료와 결합할 때 시각 기준 및 풍속·시정 단위를 맞춥니다.	
간행물	공항기후자료를 정기적으로 발간	

저층급변풍경고(LLWAS)



개요

자료상태 : 현행

저층급변풍경고장비(Low Level Wind Shear Alert System, LLWAS)는 활주로와 활주로 인접지역의 저층에서 발생하는 급변풍(Wind shear) 및 마이크로버스트(Microburst)를 탐지하여, 항공기에 공항 주변 지역과 이·착륙 지역에서 발생하는 돌풍 현상을 경고(Alert)하기 위한 시스템입니다.

주요요소	풍향, 풍속, 윈드시어, 마이크로버스트
제공범위	3개소 공항 ※ 인천, 제주, 양양
제공기간	인천공항(113): 2004-12-02 16:06:00 ~ 현재 제주공항(182): 2012-02-27 15:45:00 ~ 현재 양양공항(92): 2011-08-08 06:04:00 ~ 현재
생산·갱신주기	분 단위
제공형식	CSV
제공경로	공공데이터포털(https://www.data.go.kr) → 기상청 항공기상 정보 → 저층윈드시어 경고(LLWAS)
이용방법	공항·기간·센서 또는 경고 항목을 정해 보유 자료를 확인합니다. 센서별 바람과 급변풍 경고를 구분하고 활주로 방향·경고 유효시간을 함께 확인합니다.

4.항공

• 공항기상관측(AMOS)
• 저층급변풍경고(LLWAS)
• 공항기상라이다(TDWL)

공항기상라이다(TDWL)



개요

자료상태 : 현행

공항기상라이다(Terminal Doppler Wind Lidar, TDWL)란 레이저 빔을 방사하여 미세에어로졸(Micro Sized Aerosol)로부터 후방 산란되어 오는 신호를 수신하여 바람을 관측하는 장비입니다. 공항 라이다는 항공기 이·착륙 경로(Glide Path)를 따라 스캔하여 급변풍과 같은 돌풍 현상을 탐지하고, 급변풍 경고와 마이크로버스트 경고를 제공합니다.

주요요소	공항기상라이다 고도별 수평바람장 (위도, 경도, 풍향, 풍속 / 고도: 300-600-900-1200-1500-1800ft)
제공범위	제주공항
제공기간	2026년 1월 28일 ~ 현재
생산·갱신주기	5분
제공형식	XML
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 항공기상 → LIDAR 관측자료
이용방법	API허브에서 활용신청 후 관측지점·시각·스캔 종류 등 요청인자를 설정합니다. XML 응답의 오류코드와 자료 유무를 확인하고 좌표·고도·품질표지를 적용합니다.

LIDAR



개요

자료상태 : 현행

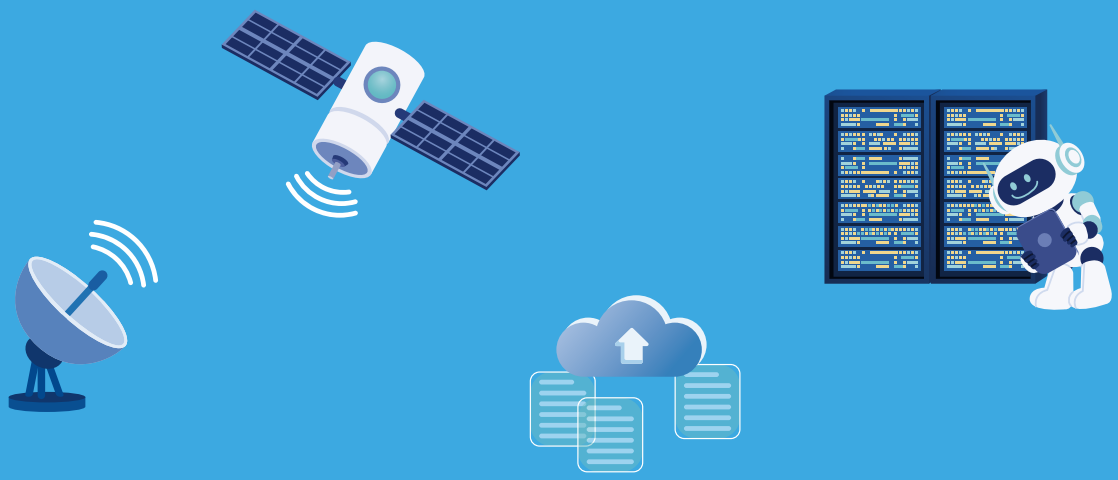
대기 중으로 일정한 파장의 레이저를 발사하여 대기 중에 부유하는 황사를 포함한 에어로졸(분자, 먼지입자, 구름입자 등)에 의해 후방 산란되는 빛을 수신·분석하여 황사의 연직 분포와 두께를 측정합니다. 또한 편광된 빛을 측정하여 에어로졸 입자의 형태를 구분합니다.

주요요소	에어로졸 소산계수, 비편광도
제공범위	안면도, 파주(종료), 백령도(종료), 군산(종료), 강화(종료)
제공기간	안면도: 2010년 12월 ~ 현재 파주: 2006년 12월 ~ 2015년 2월(종료) 백령도: 2005년 1월 ~ 2015년 2월(종료) 군산: 2004년 1월 ~ 2015년 4월(종료) 강화: 2004년 1월 ~ 2006년 11월(종료)
생산·갱신주기	일 자료 (안면도 자료)
제공형식	PNG
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량 날씨누리(https://weather.go.kr)
이용방법	국립기상과학원 지구대기감시연구과 문의(064-780-6648) (안면도 자료)

02

기상위성

Meteorological Satellite





1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

(생산종료)천리안위성 1호 기본/기상산출물 (L1,L2)



개요

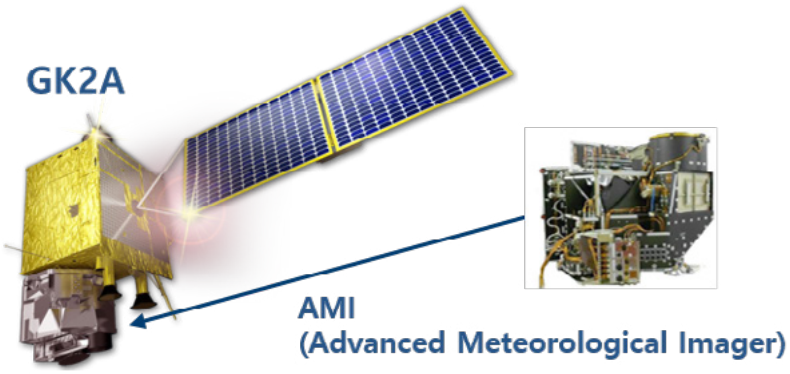
자료상태 : 생산종료

2010년에 발사된 천리안위성(COMS, Communication Ocean and Meteorological Satellite)은 지구 적도 상공 36,000km 고도, 동경 128.2°에 위치하여 기상관측, 해양관측, 통신서비스 등의 임무를 수행하는 우리나라 최초의 정지궤도 복합위성입니다.
천리안위성 1호는 5개의 기본채널 관측을 통해 16종의 기상산출물을 생산하며, 생산된 기상산출물들은 기상예보, 수치예보 및 기후변화 예측 등에 활용합니다.

주요요소	(L1) 가시영상, 단파적외영상, 수증기영상, 적외1영상, 적외2영상 (L2) 구름탐지, 해수면온도, 지표면온도, 해빙/적설탐지, 표면도달일사량, 상층수증기량, 가강수량, 구름분석, 안개, 강우강도, 지구방출복사량, 에어로졸탐지, 에어로졸 광학두께, 대기운동벡터, 청천복사, 대기불안정
제공범위	전구, 북반구, 동아시아, 한반도
관측시각	• 북반구: 매시 00, 15, 30, 45분 • 한반도: 매시 13, 28, 43, 58분 • 전구: 3시간마다 매시 15분
제공기간	2011년 4월 ~ 2020년 3월
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1) he5, bin.gz, png (L2) jpg, 이진(BIN), HDF, REST(API)
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 기상위성 → 천리안위성 1호 공공데이터포털(http://www.data.go.kr) → '천리안' 검색 → '기상청 기상위성 정보(파일데이터)' 기상청 API허브(https://apihub.kma.go.kr) ※ 이용 문의: 국가기상위성센터(kmasod@korea.kr)
이용방법	기상자료개방포털(http://data.kma.go.kr) ※ 소통과 참여 → 자료실 → '천리안 위성 Binary 파일 정보' 참조 기상청 API허브(https://apihub.kma.go.kr) ※ 이용 문의: 국가기상위성센터(kmasod@korea.kr)
품질정보	'2019년 천리안위성 1호 복사·위치 보정 품질분석 보고서' 참고

천리안위성 2A호 기본/기상산출물 (L1,L2,L3,L4)

1.정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)



개요

자료상태 : 현행

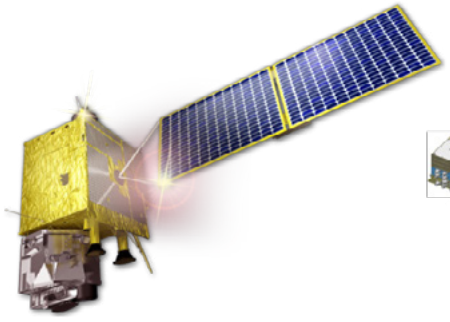
2018년 12월 5일에 발사된 천리안위성 2A호는 천리안위성 1호의 기상관측 역할을 승계하는 차세대 정지궤도 기상위성으로 기상 및 우주기상 관측임무를 수행합니다.
천리안위성 2A호는 16개의 기본채널 관측을 통해 52종의 기상산출물을 생산하며, 국내에서 개발한 우주기상 탑재체로 우주기상 감시 및 위성 운영과 관련되는 현상을 관측합니다. 이로써 우리나라의 기상관측, 수치예보, 기후감시, 우주기상감시의 역량을 강화했습니다.

주요요소	(L1) 가시영상(4개), 단파적외영상(1개), 근적외영상(2개), 수증기영상(3개), 적외영상(6개) (L2) AI기반 모의레이더(AI-RDR), AI기반 일사량(AI-DSR), AI기반 자외선지수(AI-UVI), AI기반 전운량(AI-CA), 강우강도(RR), 고해상도 대기운동벡터(AMV), 구름분석(CLA), 구름상부정보(CTPS), 구름탐지(CLD), 대류권계면 접힘 난류탐지, 다중융합 황사탐지(ADPS_DEBRA), 단파복사(SWRAD), 대기/지형 보정(ATCOR), 대기불안정지수(AII), 대기안정도 지수(ANN), 대기운동벡터(AMV), 대기정보(AAP), 다중위성융합 대기정보(MULTI_AAP), 딥러닝 기반 대류운발생탐지(CI), 확률기반 대류운발생탐지(CI), 배경지표반사율(BSR), 복사량(RAD), 산불탐지(FF), 산불열원강조(FFE), 성층권침투대류운탐지(OT), 식생정보(VGH), 안개(FOG), 도로안개(Road-FOG), 야간구름광학두께(NCOT), 양방향반사율분포도(BRDF), 에어로졸 시정(AVIS), 에어로졸특정산출물(APPS), 에어로졸탐지(ADPS), 클러스터링 기반 운형(CTCLUST), 이산화황 탐지(SO2D), 장파복사(LWRAD), 적설/해빙(SCSI), 적설깊이(SD), 정량적강수예측(QPN), 정량적강수상세예측(QPN), 주간구름광학특성(DCOEW), 지표면반사도(SAL) (L3) 1일 평균 적설/해빙(SCSI), 1일 평균 해상풍정보(ASWIND), 10일 평균 해수면온도(SST), 1일 평균 해수면온도(SST), 5일 평균 해수면온도(SST) (L4) 잠재 증발(산)량, 지상황사농도(SFC_DC), 토양수분(SM), 해수면온도(SST), 해양 선박착빙(OCN-VICING), 해양 수온변동지수(SSTA), 해양 에디:소용돌이(OCN_EDDY), 해양 연안용승(OCN_UA), 해양 전선(OCN_OF), 해양열용량(TCHP), 황사배출량(DF), 가뭄감시(DROUGHT), 다중 위성 강우강도(RR), 산불 방사열 에너지(FRP), 산불위험도(FR), GEO 에어로졸 두께(AOD), LEO 에어로졸 두께(AOD)
제공범위	전구, 동아시아, 한반도
관측시각	• 전구/동아시아: 매시 정각부터 10분 간격 • 한반도: 매시 정각부터 2분 간격
생산·갱신주기	2019년 7월 25일 ~ 현재
생산주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1, L2, L3, L4) PNG, NetCDF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 기상위성 → 천리안위성 2A호 공공데이터포털(http://www.data.go.kr) → '천리안' 검색 → '기상청 기상위성 정보(파일데이터)' 기상청 API허브(https://apihub.kma.go.kr) ※ 이용 문의: 국가기상위성센터(kmasod@korea.kr) DCPC 웹사이트(http://dcpc.nmsc.kma.go.kr/)
이용방법	기상청 API허브(https://apihub.kma.go.kr) ※ 이용 문의: 국가기상위성센터(kmasod@korea.kr)

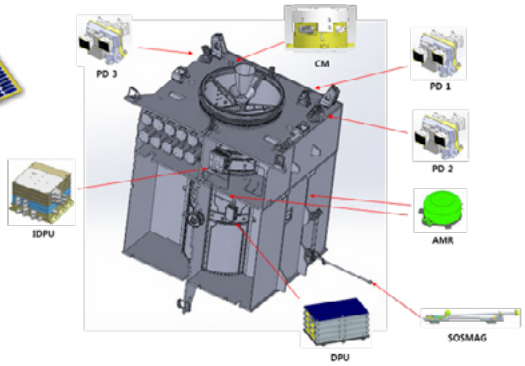
1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- **천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)**
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

천리안위성 2A호 우주기상 기본/기상산출물 (L1,L2)



GK2A



KSEM
(Korean Space wEather Monitor)

개요

자료상태 : 현행

2018년 12월에 발사된 천리안위성 2A호의 우주기상탑재체(KSEM, Korean Space Weather Monitor)를 통해 국내 최초 정지궤도에서 우주기상 관측을 수행하고 있습니다. KSEM은 입자측정기(PD, Particle Detector), 자력계(SOSMAG, Magnetometer), 위성 대전감시기(CM, Charging Monitor)로 이루어진 3종의 센서가 탑재되어 있습니다. 입자측정기는 지구로 유입되는 양성자 및 전자의 유입량(플럭스)을 측정하며, 자력계는 지구자기장의 변화량을 측정하고, 위성 대전감시기는 천리안위성 2A호의 내부대전량을 측정합니다. KSEM에서 관측된 자료는 우주위험기상의 실시간 감시 및 조기탐지, 위성체의 안정적인 운영을 위한 정보제공, 우주과학기술의 발전을 위한 학술연구 등에 활용되고 있습니다.

주요요소	(L1) 양성자/전자 플럭스, 자기장, 위성대전량 (L2) 위성대전지수, 자기권입자분포, Kp 지수 예측값, 정지궤도입자플럭스, Dst 지수 예측값 * L2 자료는 미제공
제공범위	지구 적도 상공 36,000km 고도, 동경 128.2°
관측시각	1일 간격
제공기간	2019년 7월 25일 ~ 현재
생산·갱신주기	관측시각과 동일
제공형식	NetCDF, txt
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 위성 ※ 이용 문의: 국가기상위성센터(kmasod@korea.kr) 국가기상위성센터 사용자포털(DCPC) 웹사이트(http://dcpc.nmsc.kma.go.kr/)
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

HIMAWARI9 기본/기상산출물 (L1,L2)



개요

자료상태 : 현행

2016년 11월 2일에 발사된 Himawari-9 위성은 일본 기상청(JMA)의 차세대 정지궤도 기상위성으로, 동경 약 140.7° 적도 상공에서 동아시아 및 서태평양 지역을 지속적으로 관측하고 있습니다. Himawari-8과 동일한 Advanced Himawari Imager(AHI)를 탑재하여 가시광 3개, 근적외 3개, 적외 10개 등 총 16개 관측밴드로 구름·수증기·대기 및 지표면 상태 등을 고해상도·고빈도로 관측하며, 전구영역을 10분 간격으로 관측할 수 있습니다. 태풍·집중호우·강한 대류 등 위험기상 감시와 초단기 및 수치예보, 기후·환경 감시, 자연재해 대응 등에 활용되고 있습니다. 2017년 3월 10일부터 Himawari-8의 백업위성으로 운영되다가 2022년 12월 13일 Himawari-8의 임무를 승계하여 주 관측위성으로 전환되었으며, 현재 정상 운영중입니다.

주요요소	(L1B) 16개 채널자료(가시(3), 근적외(3), 단파(1), 수증기(3), 적외(6)) (L2) RGB, 전운량, 안개, 황사, 청천복사량, 대기운동정보
제공범위	전구, 한반도
관측시각	10분마다 생산
제공기간	2022년 9월 26일 ~ 현재
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B) bin, png(L2) nc, bin, png
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

1.정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- **(생산종료)HIMAWARI8 기본/기상산출물(L1,L2)**
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

(생산종료)HIMAWARI8 기본/기상산출물 (L1,L2)



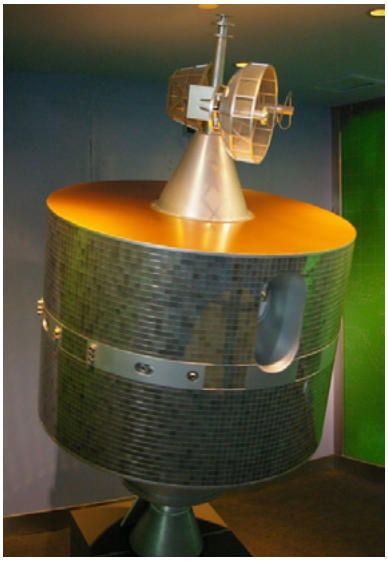
개요

자료상태 : 생산종료

2014년 10월 7일에 발사된 Himawari-8 위성은 일본 기상청(JMA)의 차세대 정지궤도 기상위성으로, 동경 약 140.7° 적도 상공에서 동아시아 및 서태평양 지역을 관측하였습니다. Advanced Himawari Imager(AHI)를 탑재하여 가시광 3개, 근적외 3개, 적외 10개 등 총 16개 관측밴드로 구름·수증기·대기 및 지표면 상태 등을 관측하며, 전구영역(Full Disk)을 10분 간격으로 관측할 수 있음. 향상된 시·공간 및 분광 관측성능을 기반으로 태풍·집중호우·강한 대류·화산재 등 위험기상 및 자연재해 감시와 기상예보, 기후·환경 감시 등에 활용되었습니다. 2015년 7월 7일부터 주 관측위성으로 운영되었으며, 2022년 12월 13일 Himawari-9으로 주 관측임무를 전환한 이후 현재는 Himawari-9의 백업위성으로 대기 운영 중입니다.

주요요소	(L1B) 16개 채널자료(가시(3), 근적외(3), 단파(1), 수증기(3), 적외(6)) (L2) RGB, 전운량
제공범위	전구, 한반도
관측시각	10분마다 생산
제공기간	2015년 5월 29일 ~ 2022년 12월 13일 14:50KST
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B, L2) bin, png
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

(생산종료) FY-2B 기본산출물 (L1)



개요

자료상태 : 생산종료

2000년 6월 25일에 발사된 Fengyun-2B(FY-2B) 위성은 중국의 1세대 정지궤도 기상위성인 FY-2 계열의 두 번째 시험위성으로, 동경 105°에 위치하여 기상관측 임무를 수행하였습니다. 3채널 가시·적외 자전주사복사계(VISSR-I)와 우주환경감시기(SEM)를 탑재하여 주간 가시광 구름영상, 주·야간 적외 구름영상 및 수증기 분포 등을 관측하였으며, 기상현상과 우주환경 감시 등에 활용되었습니다. FY-2B의 운영을 통해 축적된 기술과 경험은 이후 FY-2 업무용 기상위성의 개발 및 안정적인 운영 기반 마련에 기여하였으며, 2004년 9월 1일 운영을 종료하였습니다.

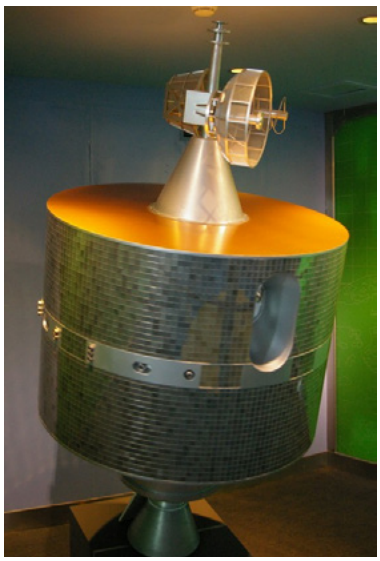
주요요소	SVISSR 센서 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2003년 3월 1일 ~ 2005년 2월 25일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	TXT, PNG
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

1. 정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본 산출물(L1)
- **(생산종료)FY-2C 기본/기상산출물(L1,L2)**
- (생산종료)FY-2D 기본 산출물(L1)
- (생산종료)FY-2E 기본 산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

(생산종료)FY-2C 기본/기상산출물(L1,L2)



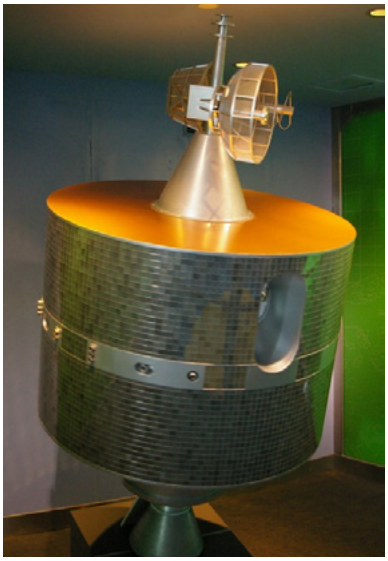
개요

자료상태 : 생산종료

2004년 10월 19일에 발사된 Fengyun-2C(FY-2C) 위성은 중국의 1세대 정지궤도 기상위성인 FY-2 계열 최초의 업무용 위성입니다. 기존 FY-2A/B보다 성능이 향상된 5채널 가시·적외 자전주사복사계(VISSR-II)와 우주환경감시기(SEM)를 탑재하여 가시광·적외 및 수증기 영역의 지구와 대기를 지속적으로 관측하였습니다. 구름 및 대기 상태 분석과 태풍·집중호우 등 위험기상 감시, 기상예보 및 방재업무 지원 등에 활용되었으며, 중국 정지궤도 기상위성이 시험단계에서 본격적인 업무운영 단계로 전환하는 데 중요한 역할을 수행하였습니다. 2009년 11월 25일 운영을 종료하였습니다.

주요요소	SVISSR 센서 L1B 자료 * L2 자료는 미제공
제공범위	전구
관측시각	1시간
제공기간	2006년 6월 7일 ~ 2008년 1월 24일
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	TXT, GIF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

(생산종료) FY-2D 기본산출물 (L1)



개요

자료상태 : 생산종료

2006년 12월 8일에 발사된 Fengyun-2D(FY-2D) 위성은 중국의 1세대 정지궤도 업무용 기상위성으로, 2007년부터 동경 86.5°에 위치하여 기상관측 임무를 수행하였습니다. 5채널 가시·적외 자전주사복사계(VISSR-II)와 우주환경감시기(SEM)를 이용하여 가시광·적외영상과 수증기 분포 등을 관측하였으며, FY-2C와의 복수위성 운영을 통해 관측 안정성과 연속성을 향상시켰습니다. 특히 두 위성을 활용한 15분 간격의 강화관측을 통해 집중호우·태풍 등 위험기상의 발달과 이동을 보다 연속적으로 감시하여 기상예보와 방재업무에 활용되었으며, 2015년 6월 30일 운영을 종료하였습니다.

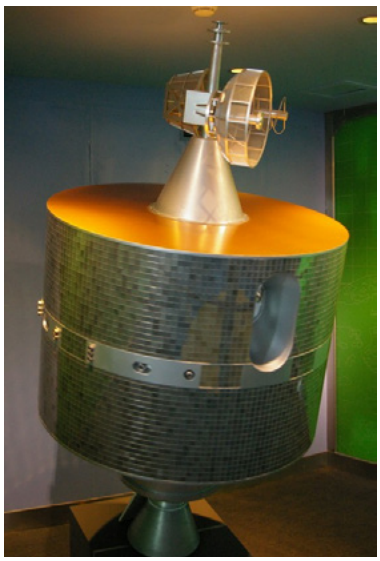
주요요소	SVISSR 센서 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2008년 3월 17일 ~ 2015년 7월 22일
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	BIN, GIF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

1.정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본 산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본 산출물(L1)
- **(생산종료)FY-2E 기본 산출물(L1)**
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

(생산종료)FY-2E 기본산출물 (L1)



개요

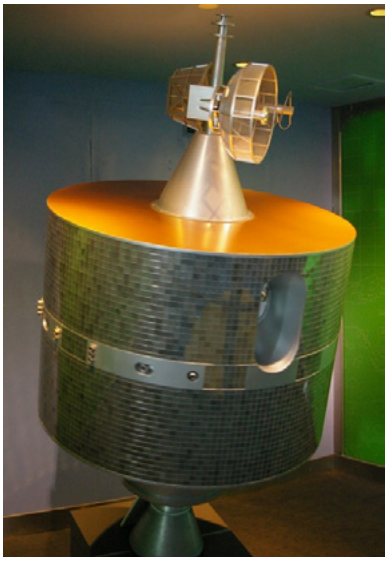
자료상태 : 생산종료

2008년 12월 23일에 발사된 Fengyun-2E(FY-2E) 위성은 중국의 1세대 정지궤도 업무용 기상위성으로, 초기 동경 105°에서 운영되다가 2015년 7월부터 동경 86.5°로 이동하여 관측 임무를 수행하였습니다. 5채널 가시·적외 자전주사복사계(VISSR-II)와 우주환경감시기(SEM)를 탑재하여 가시광·적외영상과 수증기 분포를 관측하였으며, 구름 및 대기 상태 분석, 태풍·집중호우 등 위험기상 감시와 기상예보 지원 등에 활용되었습니다. 또한 기상·수문·해양 관측자료 수집과 우주환경 감시 등의 임무를 수행하여 FY-2 계열의 안정적인 업무관측과 자료 연속성 확보에 기여하였으며, 2019년 1월 11일 운영을 종료하였습니다.

주요요소	SVISSR 센서 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2015년 7월 16일 ~ 2019년 1월 25일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	BIN, GIF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

FY-2H 기본산출물 (L1)

1.정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본 산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본 산출물(L1)
- (생산종료)FY-2E 기본 산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물 (L1,L2)
- FY-4B 기본산출물(L1)



개요

자료상태 : 생산종료

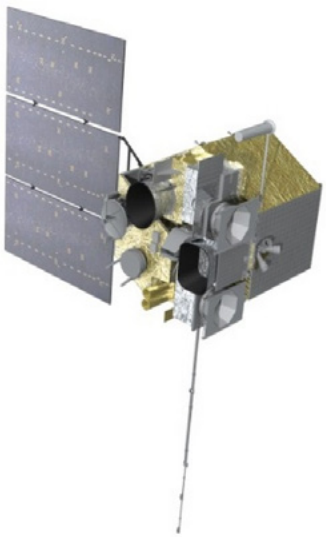
2018년 6월 5일에 발사된 Fengyun-2H(FY-2H) 위성은 중국의 1세대 정지궤도 기상위성인 FY-2 계열의 후속 업무용 위성으로, 동경 79° 인도양 상공에 위치하여 현재 기상관측 임무를 수행하고 있습니다. 가시·적외 자전주사복사계(VISSR-II)와 우주환경감시기(SEM)를 이용하여 가시광·적외영상과 수증기 분포 및 우주환경을 관측하며, 구름·대기 상태 분석과 태풍·강한 대류 등 위험기상 감시, 기상예보 및 방재업무 등에 활용되고 있습니다. 특히 79°E의 관측위치를 기반으로 아시아·인도양 및 주변지역에 대한 광범위한 기상감시와 위성자료 서비스를 지원하고 있습니다.

주요요소	SVISSR 센서 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2019년 2월 27일 ~ 2023년 3월 3일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	BIN, GIF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

1.정지궤도 기상위성

- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본 산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본 산출물(L1)
- (생산종료)FY-2E 기본 산출물(L1)
- FY-2H 기본산출물(L1)
- **FY-4A 기본/기상산출물 (L1,L2)**
- FY-4B 기본산출물(L1)

FY-4A 기본/기상산출물 (L1,L2)



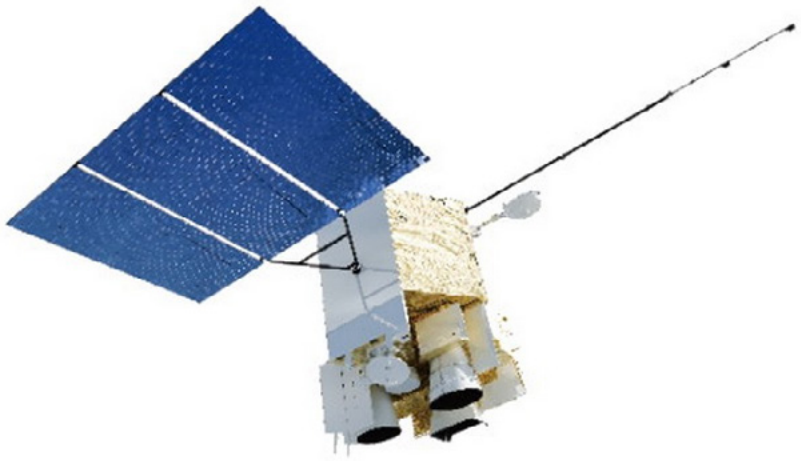
개요

자료상태 : 현행

2016년 12월 11일에 발사된 Fengyun-4A(FY-4A) 위성은 중국의 2세대 정지궤도 기상위성인 FY-4 계열의 첫 번째 위성으로, 기존 FY-2 계열보다 향상된 시·공간 및 분광 관측성능을 기반으로 지구와 대기를 관측하고 있습니다.
고성능 정지궤도 복사영상기(AGRI), 정지궤도 간섭계 적외탐측기(GIIRS), 번개탐지기(LMI) 및 우주환경감시장비 등을 이용하여 다채널 영상과 대기 연직정보, 번개 및 우주환경 등을 종합적으로 관측하였습니다. 구름·수증기·지표면 및 대기 상태 분석, 태풍·집중호우·강한 대류 등 위험기상 감시와 초단기예보, 수치예보 및 재해·환경 감시 등에 활용되고 있습니다.

주요요소	(L1B) GIIRS 초분광 복사휘도 * L2 자료 미제공
제공범위	전구
관측시각	1시간
제공기간	2020년 10월 14일 ~ 2023년 7월 31일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

FY-4B 기본산출물 (L1)



개요

자료상태 : 현행

2021년 6월 3일에 발사된 Fengyun-4B(FY-4B) 위성은 중국의 2세대 정지궤도 기상위성인 FY-4 계열의 두 번째 위성이자 첫 번째 본격적인 업무용 위성으로, 현재 동경 105°에 위치하여 기상관측 임무를 수행하고 있습니다.

고성능 정지궤도 복사영상기(AGRI), 정지궤도 간섭계 적외탐측기(GIIRS), 고속영상기(GHI) 및 우주환경감시장비 등을 이용하여 구름·대기 및 지표면을 고해상도·고빈도로 관측하였습니다. 특히 특정 지역에 대한 신속한 집중관측 능력을 바탕으로 태풍·집중호우·강한 대류 등 빠르게 발달하는 위험기상을 정밀하게 감시하며, 초단기예보·수치예보·환경 및 재해감시 등에 활용되고 있습니다. 현재 정상 운영 중에 있습니다.

주요요소	(L1B) GIIRS 초분광 복사휘도
제공범위	전구
관측시각	1시간
제공기간	2023년 3월 1일 ~
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

1.정지궤도 기상위성
- (생산종료)천리안위성 1호 기본/기상산출물(L1,L2)
- 천리안위성 2A호 기본/기상산출물(L1,L2,L3,L4)
- 천리안위성 2A호 우주기상 기본/기상산출물(L1,L2)
- HIMAWARI9 기본/기상산출물(L1,L2)
- (생산종료)HIMAWARI8 기본/기상산출물(L1,L2)
- (생산종료)FY-2B 기본산출물(L1)
- (생산종료)FY-2C 기본/기상산출물(L1,L2)
- (생산종료)FY-2D 기본산출물(L1)
- (생산종료)FY-2E 기본산출물(L1)
- FY-2H 기본산출물(L1)
- FY-4A 기본/기상산출물(L1,L2)
- FY-4B 기본산출물(L1)

2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- (생산종료)METOP-A 기본/기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

(생산종료)NOAA-18 기본/기상산출물 (L1,L2)



개요

자료상태 : 생산종료

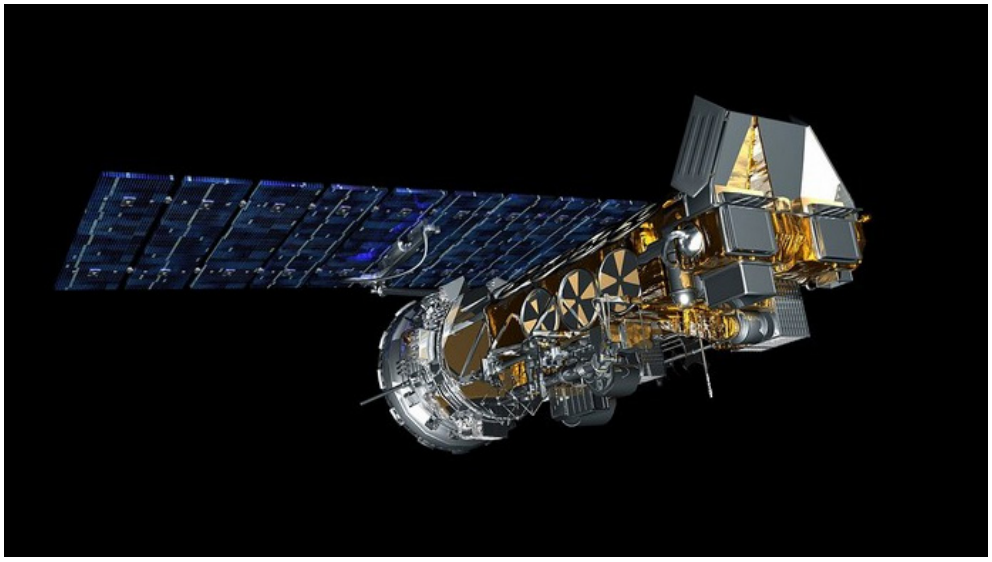
2005년 5월 20일에 발사된 NOAA-18 위성은 미국 NOAA의 POES 프로그램에 속하는 극궤도 기상위성으로, 약 850 km 고도의 극궤도를 운항하며 장기간 전 지구 기상·환경관측 임무를 수행하였습니다.

AVHRR, AMSU-A, HIRS, MHS 등의 관측장비를 이용하여 구름·해수면 및 지표면 온도, 대기 온·습도 연직분포 등을 관측하였으며, 기상예보와 수치예보, 기후 및 환경감시 등에 활용합니다. 또한 SARSAT 장비를 통한 조난신호 탐지·중계 등 수색구조 업무도 지원하였습니다.

장기간의 임무 수행 후 S-band 송신기의 복구 불가능한 고장으로 인해 2025년 6월 6일 17:40 UTC에 공식적으로 운영을 종료하였습니다.

주요요소	(L1B, L2) AMSU-A, HIRS 센서
제공범위	전구
관측시각	1시간
제공기간	(L1B) 2005년 5월 20일 ~ 2005년 5월 29일, 2005년 6월 1일 ~ 2005년 11월 26일 (L2) 2018년 9월 1일 ~ 2025년 6월 6일
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B) DAT, BIN (L2) TXT, BIN
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

(생산종료)NOAA-19 기본/기상산출물 (L1,L2)



개요

자료상태 : 생산종료

2009년 2월 6일에 발사된 NOAA-19 위성은 미국 NOAA의 POES 프로그램에 속하는 극궤도 기상위성으로, 전 지구의 대기·해양·육지 및 우주환경을 지속적으로 관측하는 임무를 수행하였습니다. AVHRR, AMSU-A, HIRS, MHS, SBUV 등의 관측장비를 이용하여 구름·해수면 및 지표면 온도, 대기 온·습도 연직분포와 오존 등을 관측하였으며, 기상 및 수치예보, 기후·환경감시와 장기간 기후자료 구축 등에 활용되었습니다.
또한 SARSAT 장비를 이용하여 조난신호 탐지와 수색구조 업무를 지원합니다. 2025년 6월 16일부터 사용자 대상 POES 운영자료 제공이 중단되었으며, 배터리 셀 고장 이후 2025년 8월 13일 16:55 UTC에 공식적으로 운영을 종료하였습니다.

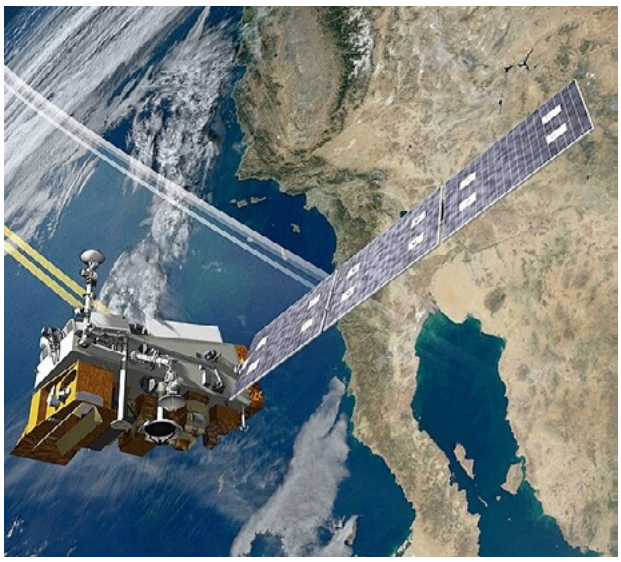
주요요소	(L1B, L2) AMSU-A, HIRS, MHS 센서
제공범위	전구
관측시각	1시간
제공기간	2009년 3월 25일 ~ 2025년 6월 18일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B) DAT, GIF, PNG, JPG, TXT, HDF (L2) HDF, GIF, NC
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

2.저궤도 기상위성
- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- (생산종료)METOP-A 기본/기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- (생산종료)METOP-A 기본/ 기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

NOAA-20 기본/기상산출물 (L1,L2)



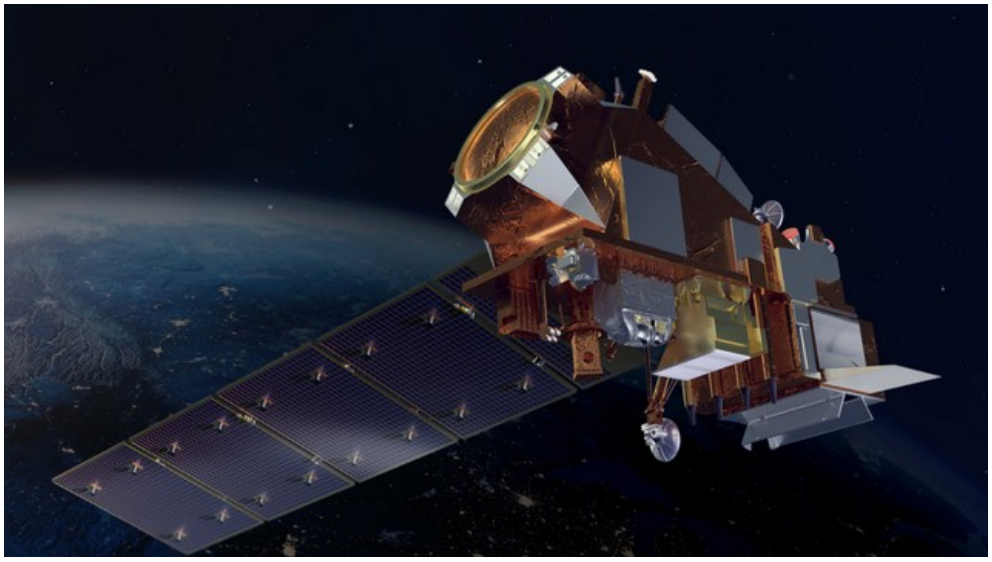
개요

자료상태 : 현행

2017년 11월 18일에 JPSS-1으로 발사된 NOAA-20 위성은 미국 NOAA의 차세대 극궤도 기상위성체계인 JPSS의 주요 위성으로, 전 지구 기상·기후 및 환경관측 임무를 수행하고 있습니다.
VIIRS, CrIS, ATMS, OMPS 및 CERES 등의 첨단 관측장비를 이용하여 구름·대기·해양·육지·빙권, 대기 온·습도 연직분포, 오존 및 지구복사수지 등을 관측 중에 있습니다.
특히 VIIRS를 이용한 구름·해수면 및 지표면 온도·산불·연기·적설·해빙 관측과 CrIS·ATMS를 이용한 대기 온·습도 탐측자료는 수치예보 정확도 향상과 위험기상 감시, 기후·환경 및 자연재해 감시 등에 폭넓게 활용되고 있습니다. 현재 NOAA-21과 함께 JPSS 극궤도 관측체계를 구성하여 정상 운영 중에 있습니다.

주요요소	(L1B, L2) ATMS, CrIS 센서
제공범위	전구
관측시각	1시간
보유기간	2019년 3월 7일 ~
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B, L2) HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

NOAA-21 기본산출물 (L1)



개요

자료상태 : 현행

2022년 11월 10일에 JPSS-2로 발사된 NOAA-21 위성은 미국 NOAA의 차세대 극궤도 기상위성체계인 JPSS의 세 번째 위성으로, 2023년부터 본격적인 업무운영을 수행하고 있습니다.
VIIRS, CrIS, ATMS 및 OMPS 등 첨단 관측장비를 탑재하여 구름·대기·해양·육지·빙권과 대기 온·습도 연직분포 및 오존 등을 전 지구적으로 관측하였습니다. 이를 통해 해수면 및 지표면 온도, 강수, 적설·해빙, 산불 위치와 연기, 대기 온도·수증기 등의 정보를 생산하여 수치예보와 기상예보 정확도 향상, 위험기상 감시, 기후변화 및 환경·재해감시 등에 활용되고 있습니다.
현재 JPSS 위성군의 주력(Primary) 위성으로 정상 운영 중에 있습니다.

주요요소	ATMS, CrIS 센서의 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2026년 3월 31일 ~
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- **S-NPP 기본/기상산출물 (L1,L2)**
- (생산종료)METOP-A 기본/ 기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

S-NPP 기본/기상산출물 (L1,L2)



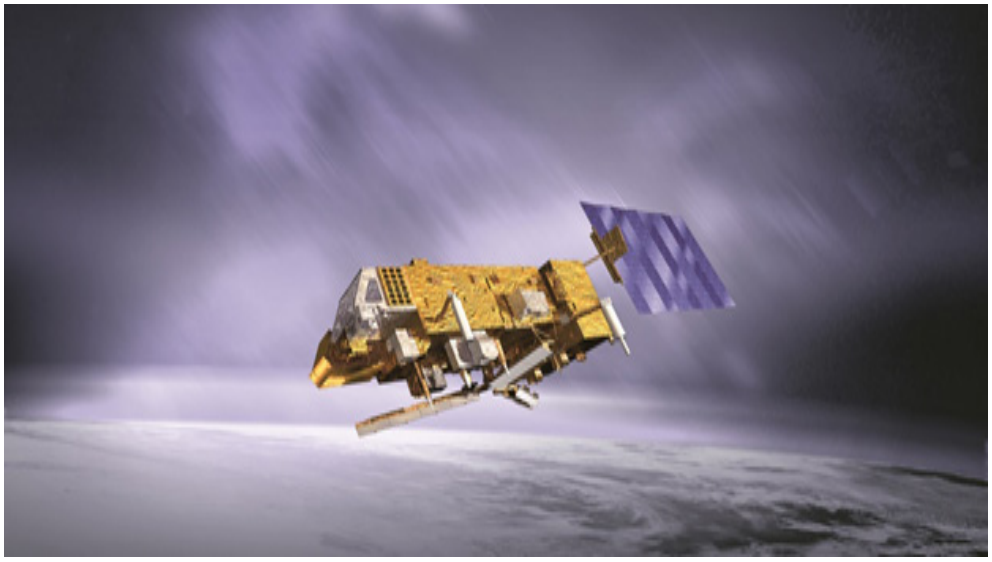
개요

자료상태 : 현행

2011년 10월 28일에 발사된 Suomi National Polar-orbiting Partnership(S-NPP) 위성은 미국 NASA와 NOAA가 공동으로 개발·운영한 차세대 극궤도 환경위성으로, 기존 NASA EOS 지구관측위성과 NOAA의 차세대 JPSS 위성체계를 연결하는 선행위성 역할을 수행하였습니다.
약 824 km 고도의 태양동기 극궤도에서 VIIRS, CrIS, ATMS, OMPS, CERES 등을 이용하여 구름·대기·해양·육지·빙권 및 지구복사환경을 관측하였습니다. VIIRS를 활용한 구름·해수면 및 지표면 온도·산불·연기·적설·해빙·야간불빛 관측과 CrIS·ATMS를 활용한 대기 온·습도 연직분포 관측 등을 통해 기상예보, 기후변화 연구 및 환경·재해감시에 활용되어 왔습니다.
현재 Tertiary(3순위) 위성으로 분류되어 있으며, NOAA는 2026년 11월 1일 13:00 UTC부터 잔여 과학자료 제공을 영구 종료할 예정입니다. CrIS 자료는 이에 앞서 2025년 12월 23일 운영 활용이 종료되었습니다.

주요요소	(L1B, L2) ATMS, CrIS 센서
제공범위	전구
관측시각	1시간
제공기간	(L1B) 2014년 1월 1일 ~ 현재 (L2) 2012년 5월 1일 ~ 2021년 5월 21일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B) HDF (L2) TXT, HDF, BIN
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

(생산종료)METOP-A 기본/기상산출물 (L1,L2)



개요

자료상태 : 생산종료

2006년 10월 19일에 발사된 Metop-A 위성은 유럽기상위성개발기구(EUMETSAT)의 EPS를 구성하는 유럽 최초의 업무용 극궤도 기상위성으로, 약 817 km 고도의 태양동기 극궤도에서 전 지구 기상·기후 및 환경관측 임무를 수행하였습니다.
IASI, AMSU-A, MHS, AVHRR/3, ASCAT, GOME-2, GRAS 등 다양한 관측장비를 이용하여 대기 온·습도 연직분포와 미량기체, 구름·지표 및 해수면 온도, 해상풍·토양수분·오존 등을 관측하였습니다.
관측자료는 수치예보 정확도 향상과 위험기상 예보, 기후변화 및 해양·육상 환경감시에 폭넓게 활용되었으며, 15년 이상의 장기간 임무를 수행한 후 2021년 11월 30일 공식적으로 퇴역하였습니다.

주요요소	(L1B, L2) AVHRR, ATOVS 센서
제공범위	전구
관측시각	1시간
제공기간	2013년 11월 4일 ~ 2019년 3월 20일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B) HDF (L2) TXT, TAR
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

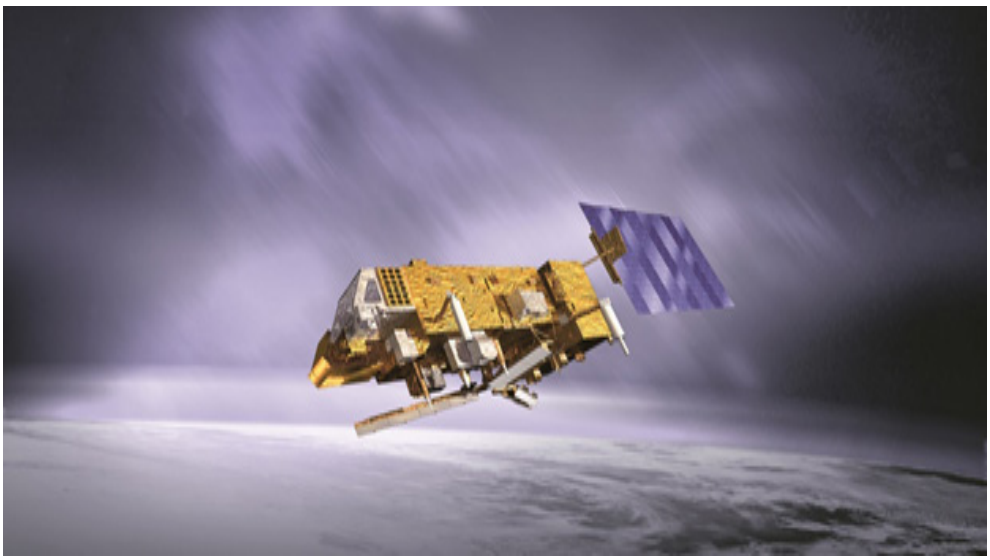
2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- **(생산종료)METOP-A 기본/기상산출물(L1,L2)**
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물(L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물(L1,L2)
- (생산종료)METOP-A 기본/기상산출물(L1,L2)
- **METOP-B 기본산출물(L1)**
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물(L1,L2)
- TERRA 기본/기상산출물(L1,L2)

METOP-B 기본산출물(L1)



개요

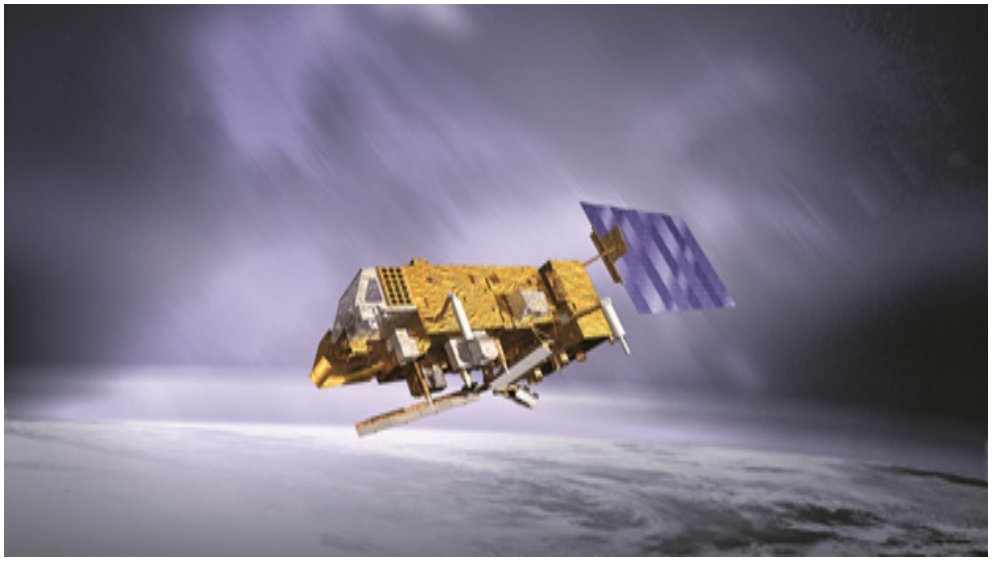
자료상태 : 현행

2012년 9월 17일에 발사된 Metop-B 위성은 EUMETSAT의 1세대 EPS를 구성하는 두 번째 극궤도 기상위성으로, 약 817 km 고도의 태양동기 극궤도에서 전 지구 대기·해양·육지 및 기후환경을 관측하고 있습니다.

IASI, AMSU-A, MHS, AVHRR/3, ASCAT, GOME-2 및 GRAS 등 다양한 관측장비를 이용하여 대기 온·습도 연직분포와 미량기체, 구름·지표 및 해수면 온도, 해상풍·토양수분·오존 등을 관측하였습니다. 특히 고정밀 대기탐측자료와 마이크로파 관측자료는 전 세계 수치예보모델의 초기자료로 활용되어 기상예보 정확도 향상에 중요한 역할을 수행하고 있으며, 해양·육상 및 기후환경 감시에도 폭넓게 활용되고 있고, 현재 관측하고 있습니다.

주요요소	AVHRR 센서의 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2026년 9월 2일 ~
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

METOP-C 기본산출물(L1)



개요

자료상태 : 현행

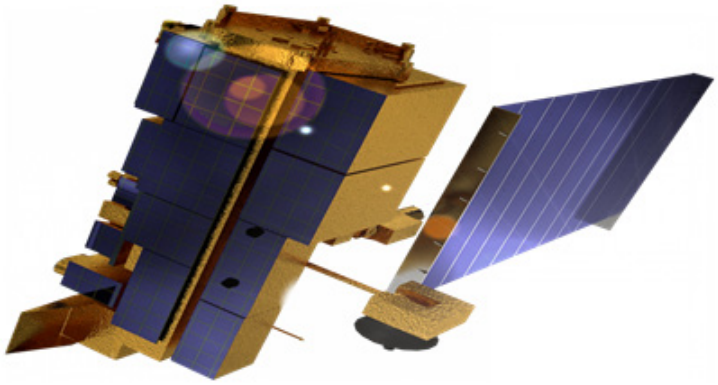
2018년 11월 7일에 발사된 Metop-C 위성은 EUMETSAT의 1세대 EPS를 구성하는 세 번째이자 마지막 극궤도 기상위성으로, Metop-A/B에서 이어진 장기간의 전 지구 기상·기후 및 환경관측을 지속하고 있습니다.
IASI, AMSU-A, MHS, AVHRR/3, ASCAT, GOME-2 및 GRAS 등의 관측장비를 이용하여 대기 온·습도 연직분포와 미량기체, 구름·지표 및 해수면 온도, 해상풍·토양수분·오존 등을 관측하고 있습니다.
Metop-B와 함께 복수 극궤도 관측체계를 구성하여 관측자료의 시간적·공간적 가용성을 높이고 있으며, 생산된 자료는 수치예보, 위험기상 예측, 기후변화 연구와 해양·육상 및 대기환경 감시 등에 활용되고 있습니다. 현재 정상 운영 중에 있습니다.

주요요소	AVHRR 센서의 L1B 자료
제공범위	전구
관측시각	1시간
제공기간	2026년 9월 2일 ~
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

2.저궤도 기상위성

- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- (생산종료)METOP-A 기본/ 기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- **AQUA 기본/기상산출물 (L1,L2)**
- TERRA 기본/기상산출물 (L1,L2)

AQUA 기본/기상산출물 (L1,L2)



개요

자료상태 : 현행

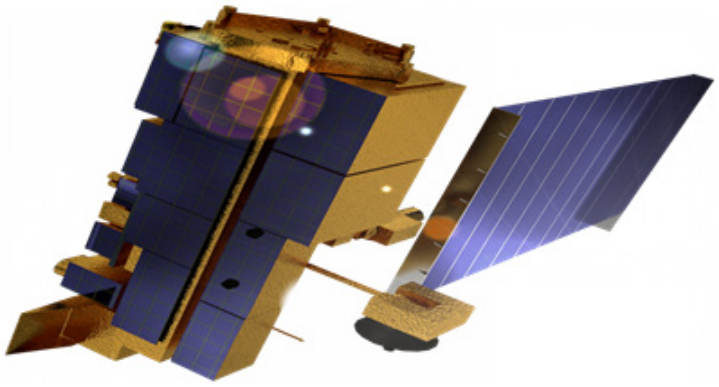
2002년 5월 4일에 발사된 Aqua 위성은 미국 NASA의 지구관측시스템(EOS)을 구성하는 극궤도 지구관측위성으로, 지구의 물 순환과 대기·해양·육지 및 에너지 수지 등을 종합적으로 관측하는 임무를 수행하고 있습니다.

약 705 km 고도의 태양동기궤도를 운항하며, AIRS, AMSU-A, MODIS, CERES 등의 관측장비를 이용하여 대기 온·습도 연직분포, 구름, 수증기, 해수면 및 지표면 온도, 에어로졸, 해빙·적설 등 다양한 지구환경 요소를 관측하고 있습니다.

특히 AIRS와 AMSU-A 관측자료는 대기의 온도·습도 구조 분석과 수치예보 등에 활용되며, MODIS 자료는 구름·해양·육지·산불·에어로졸 및 환경감시 등에 폭넓게 활용되고 있습니다. 현재 장기간의 연장임무를 수행 중에 있습니다.

주요요소	(L1B) MODIS센서 L1B 자료 (L2) MODIS센서 L2 자료
제공범위	전구
관측시각	1시간
제공기간	(L1B) 2002년 12월 20일 ~ (L2) 2006년 12월 4일 ~ 2019년 1월 19일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B, L2) HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

TERRA 기본/기상산출물 (L1,L2)



2.저궤도 기상위성
- (생산종료)NOAA-18 기본/기상산출물(L1,L2)
- (생산종료)NOAA-19 기본/기상산출물(L1,L2)
- NOAA-20 기본/기상산출물 (L1,L2)
- NOAA-21 기본산출물(L1)
- S-NPP 기본/기상산출물 (L1,L2)
- (생산종료)METOP-A 기본/기상산출물(L1,L2)
- METOP-B 기본산출물(L1)
- METOP-C 기본산출물(L1)
- AQUA 기본/기상산출물 (L1,L2)
- TERRA 기본/기상산출물 (L1,L2)

개요

자료상태 : 현행

1999년 12월 18일에 발사된 Terra 위성은 미국 NASA의 지구관측시스템(EOS)을 대표하는 극궤도 지구관측위성으로, 지구의 대기·육지·해양과 이들 사이의 상호작용을 장기간 종합적으로 관측하는 임무를 수행하고 있습니다.
약 705 km 고도의 태양동기궤도를 운항하며, MODIS, ASTER, MISR, CERES, MOPITT 등을 탑재하였습니다. 이를 통해 구름·에어로졸·식생·산불·적설 및 해빙, 지표면 온도와 토지피복, 대류권 일산화탄소 및 지구복사수지 등을 관측하여 기상·기후변화 연구와 환경·재해 감시 등에 활용되고 있습니다.
장기간 지구관측자료를 생산하여 기후변화와 지구환경 변화 분석에 중요한 역할을 수행하고 있습니다.

주요요소	(L1B) MODIS센서 L1B 자료 (L2) MODIS센서 L2 자료
제공범위	전구
관측시각	1시간
제공기간	(L1B) 2004년 2월 10일 ~ 2008년 8월 16일, 2019년 7월 19일 ~ 2020년 6월 22일 (L2) 2006년 11월 16일 ~ 2019년 1월 22일
생산·갱신 주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(L1B, L2) HDF
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

3.기타

- AQUA 기본/기상산출물 (총지연량,가강수량)

GNSS 기본/기상산출물 (총지연량,가강수량)



개요

자료상태 : 현행

GNSS(Global Navigation Satellite System)는 미국 GPS, 러시아 GLONASS, 유럽 Galileo, 중국 BeiDou 등 전 지구 위성항법시스템을 통칭하며, 항법·측위뿐만 아니라 대기 및 기상관측에도 활용되고 있습니다.

GNSS 위성에서 송신된 전파가 지구 대기권을 통과하면서 발생하는 굴절·지연 특성을 분석하여 대기의 온도·습도·기압 및 수증기 정보를 산출할 수 있으며, 특히 저궤도 위성에서 GNSS 신호의 전파엄폐(Radio Occultation)를 관측하여 대기 연직구조를 고해상도로 분석할 수 있습니다.

이러한 GNSS 기반 대기관측자료는 수치예보모델의 초기자료, 대기 온·습도 분석, 기상예보 및 장기간 기후변화 연구 등에 활용되고 있습니다. 또한 지상 GNSS 관측망에서 산출되는 대류권 신호지연을 이용하여 대기 중 수증기량을 추정할 수 있어 집중호우 등 위험기상 감시와 예보에도 활용되고 있습니다.

주요요소	총지연량, 가강수량 (Bernese v5.0 및 v5.2 버전으로 산출)
제공범위	한반도
관측시각	15분, 1시간
제공기간	(v5.0 1시간) 2016. 7. 1 ~ (v5.0 15분) 2017. 2. 22. ~ (v5.2 1시간) 2020. 6. 1. ~ (v5.2 15분) 2022. 12. 09. ~
생산·갱신주기	관측시각과 동일(모든 관측에 대해 자료생산)
제공형식	(총지연량) TRO (가강수량) BUFR
제공경로	국가기상위성센터(kmasod@korea.kr) 별도 문의
이용방법	국가기상위성센터(kmasod@korea.kr) 별도 문의

03

레이더

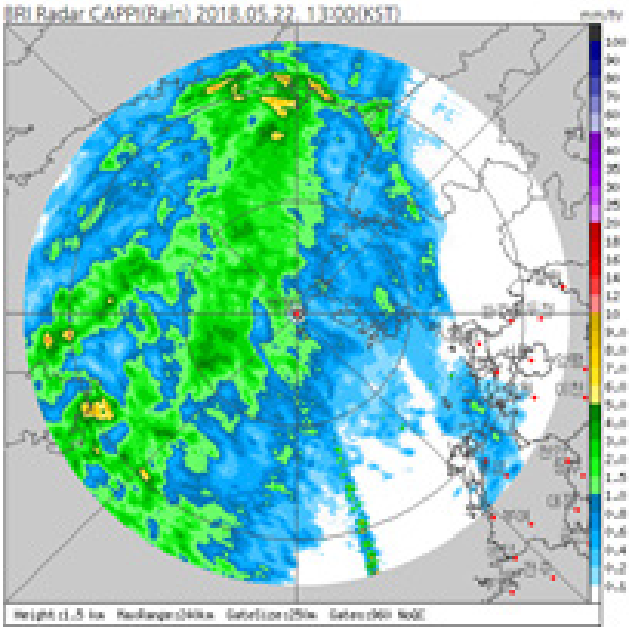
Radar



1.사이트

• 원시자료
• 품질관리자료(QC)
• 대기수상체(HCI)
• 다중 고도각 기반 레이더 추정 강수량(HSR)

원시자료



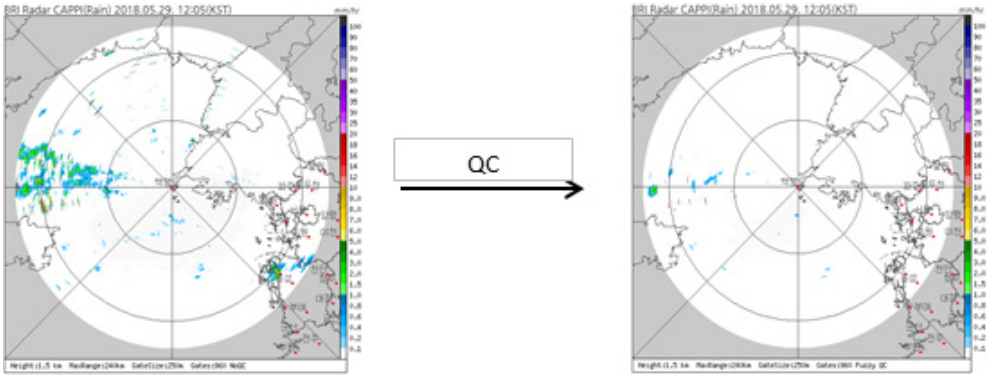
개요

자료상태 : 현행

기상레이더는 전자파를 발사, 대기 중의 물방울에 부딪혀 되돌아오는 반사파를 분석하여 강수의 지역과 세기 등을 관측하는 장비입니다. 주로 집중호우, 대설, 우박 등 위험기상과 한반도로 접근하는 태풍을 추적하고 감시합니다. 전국 19개 지점(대형 17소, TDWR 2소)에서 기상레이더를 운영합니다.

주요요소	보정전반사도(DZ), 보정후반사도(CZ), 시선속도(VR), 스펙트럼폭(SW), 차등반사도(DR), 교차상관계수(RH), 차등위상차(PH), 비차등위상차(KD)	
제공범위	관악산, 오성산, 구덕산, 고산, 강릉, 백령도, 진도, 광덕산, 면봉산, 인천공항, 성산, 제주공항, 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산	
제공기간	• 관악산, 오성산, 구덕산, 고산, 강릉: 1998년 ~ 현재 • 백령도, 진도, 인천공항(TDWR): 2001년 ~ 현재 • 광덕산: 2003년 ~ 현재 • 면봉산: 2005년 ~ 현재 • 성산: 2006년 ~ 현재 • 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산: 2026년 ~ 현재 • 제주공항(TDWR): 2026년 ~ 현재	
생산·갱신주기	분(5, 10분 주기) 자료	
제공형식	NetCDF, UF, PNG	
제공경로	NetCDF UF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 사이트
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더원시자료
	PNG	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 사이트
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.	

품질관리자료(QC)



개요

자료상태 : 현행

레이더 자료에는 대기 중의 기상현상에 의한 강수에코뿐만 아니라 주변의 지형, 건물, 파랑, 채프, 생물체 등이 관측되어 나타납니다. 이러한 비강수에코를 제거하고 품질 좋은 관측자료를 제공하기 위해 품질관리를 수행하고 있습니다.

기상청은 2006년부터 ORPG 품질관리 알고리즘을 개발하여 현업에서 활용하였고, 2014년부터 이중편파 퍼지논리(fuzzy logic)기반 품질관리 알고리즘을 개발하여 적용하였으며, 2019년에 자체 개발한 품질관리기술(CLEANER)을 현업에 적용하고 있습니다.

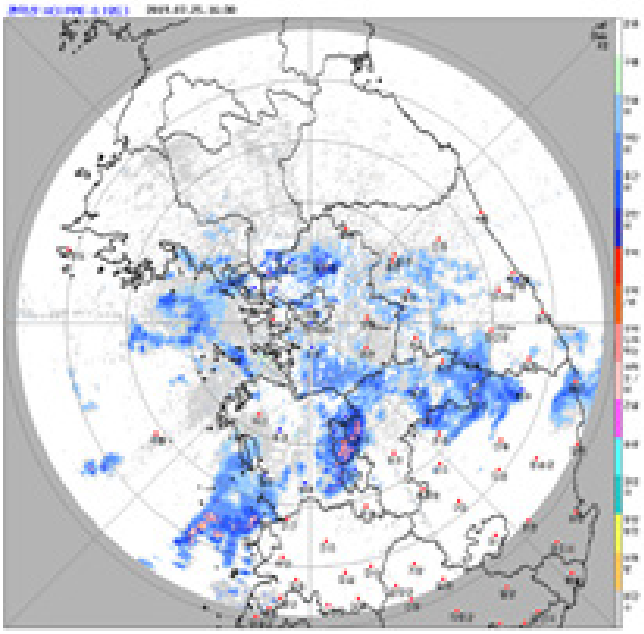
※ 품질관리 요소 : 반사도, 시선속도, 차등반사도, 교차상관계수, 차등위상차, 비차등위상차

주요요소	보정전 반사도(DZ), 보정후 반사도(CZ), 시선속도(VR), 스펙트럼 폭(SW) 차등반사도(DR), 교차상관계수(RH), 차등위상차(PH), 비차등위상차(KD)	
제공범위	관악산, 오성산, 구덕산, 고산, 강릉, 백령도, 진도, 광덕산, 면봉산, 인천공항, 성산, 제주공항, 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산	
제공기간	• 관악산, 오성산, 구덕산, 고산, 강릉, 백령도, 진도, 광덕산, 면봉산, 인천공항, 성산: 2006년 ~ 현재 • 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산, 제주공항: 2026년 ~ 현재	
생산·갱신 주기	분(5, 10분 주기) 자료	
제공형식	NetCDF, PNG	
제공경로	NetCDF UF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 사이트
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더원시자료
	PNG	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 사이트
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.	

1.사이트

- 원시자료
- 품질관리자료(QC)
- 대기수상체(HCI)
- 다중 고도각 기반 레이더 추정 강수량(HSR)

대기수상체(HCI)



값	NCAR 분류	기상레이더센터 분류	사용 명칭
0	-	자료없음	무강수
1	구름(CL)	비	비
2	이슬비(DRZ)		
3	약한비(LR)		
4	중간비(MR)		
5	강한비(HR)	강한비	
6	우박(HA)	비+우박	우박
7	비+우박(RH)		
8	싸락눈+작은우박(GSH)		
9	싸락눈+비(GR)		
10	건설(DS)	건설	눈
11	습설(WS)	습설	눈비
14	과냉각수적(SLD)		
12	등방 빙정(IC)	빙정	빙정
13	비등방 빙정(IIC)		
50	-	비기상 에코	비기상

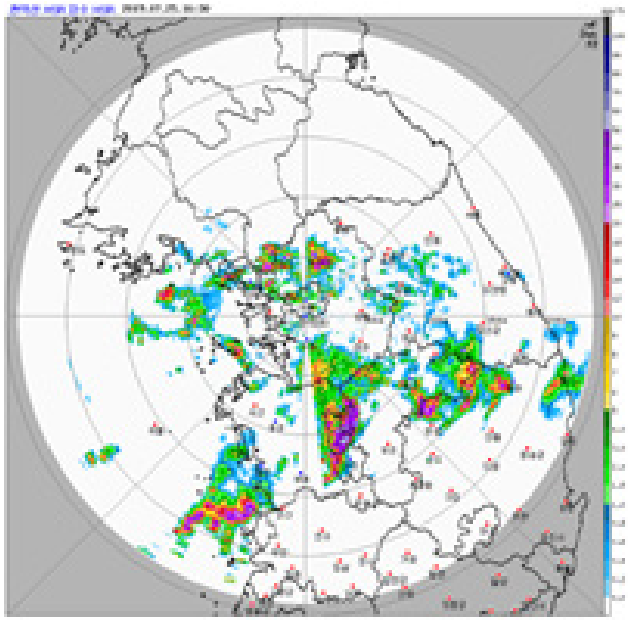
개요

자료상태 : 현행

여러 레이더의 관측을 고도별로 합성한 HCI 자료입니다. 선택한 고도의 반사도 등 합성요소를 분석하는 데 활용하며, 지상 강수량과 동일한 값으로 해석하지 않습니다.

주요요소	대기수상체
제공범위	관악산, 오성산, 구덕산, 고산, 백령도, 진도, 광덕산, 면봉산, 성산, 인천공항, 제주공항 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산
제공기간	• 관악산, 구덕산, 백령도, 진도, 면봉산 : 2016년 ~ 현재 • 고산, 광덕산: 2017년 ~ 현재 • 오성산, 성산: 2018년 ~ 현재 • 인천공항: 2024년 ~ 현재 • 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산, 제주공항: 2026년 ~ 현재
생산·갱신주기	분(5분 주기) 자료
제공형식	NetCDF
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량(HSR) → 레이더사이트
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.

다중 고도각 기반 레이더 추정 강수량(HSR)



개요

자료상태 : 현행

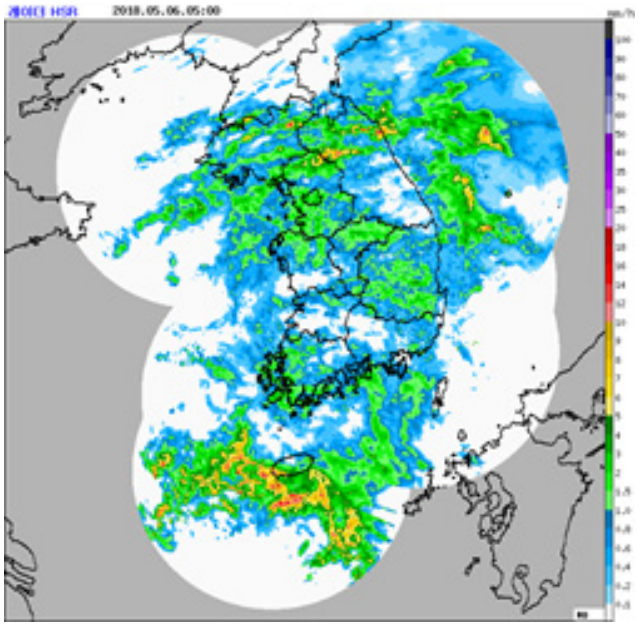
지형차폐의 영향이 없는 지상에 가장 가까운 고도각 자료로 추정한 레이더 강우량(HSR: Hybrid Surface Rainfall)으로, 지형에코와 비기상에코 등의 영향을 최소화하여 레이더 강우량의 정확도를 향상시켰고, 에코고도를 함께 제공하여 분석능력을 강화하였습니다.

주요요소	반사도(CZ), 차등반사도(DR), 비차등위상차(KD), 고도각, 고도값
제공범위	관악산, 오성산, 구덕산, 고산, 강릉, 백령도, 진도, 광덕산, 면봉산, 인천공항, 성산 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산
제공기간	• 관악산, 오성산, 구덕산, 고산, 강릉, 백령도, 광덕산, 성산: 2014년 ~ 현재 • 진도, 면봉산: 2016년 ~ 현재 • 인천공항: 2023년 ~ 현재 • 감악산, 예봉산, 가리산, 소백산, 비슬산, 서대산, 모후산: 2026년 ~ 현재
생산·갱신주기	분(5분 주기) 자료
제공형식	NetCDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 사이트
	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량(HSR) → 레이더사이트
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.

2.합성

- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원

레이더 강수량



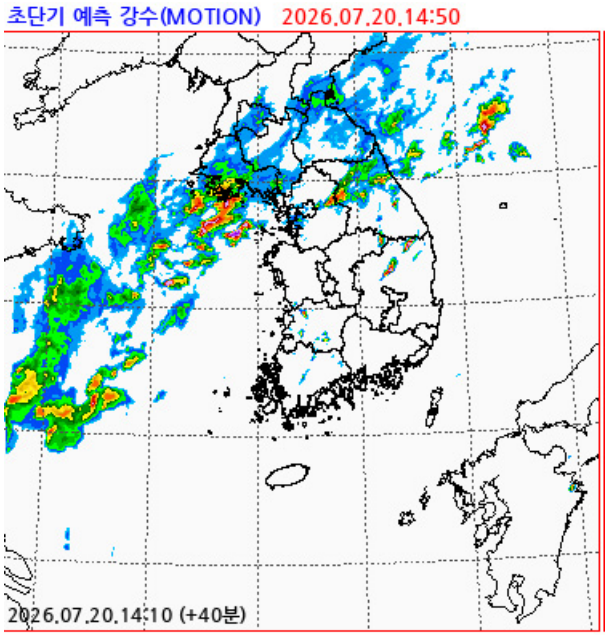
개요

자료상태 : 현행

레이더 강수량은 각각의 레이더 관측소에서 생산된 관측자료를 합성하여 우리나라 전역의 레이더 합성영상자료를 산출하고, 이를 활용하여 우리나라에 영향을 주는 강수의 범위와 이동경향에 대한 감시 및 분석에 이용합니다.

주요요소	강수량	
제공범위	전국	
제공기간	1998년 1월 ~ 현재	
생산·갱신주기	분(5분 주기) 자료	
제공형식	BIN, PNG	
제공경로	BIN	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 합성
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량
	PNG	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 레이더 → 합성
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.	

레이더 강수량 실황예측



개요

자료상태 : 현행

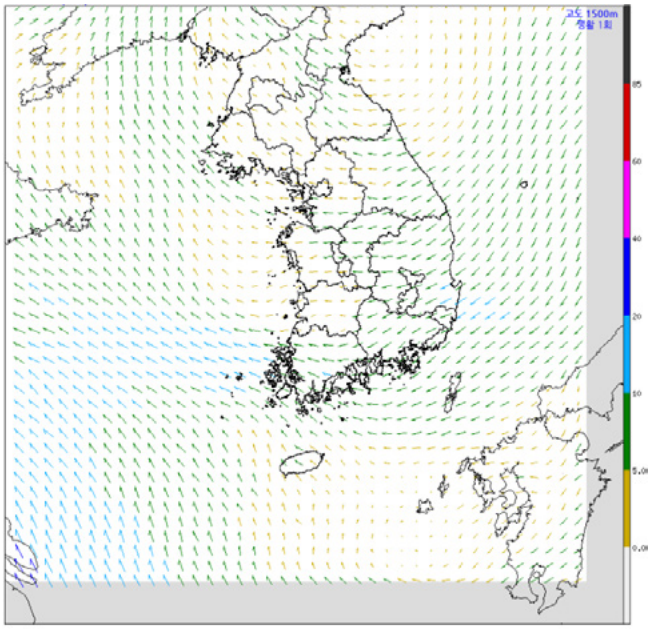
레이더 강수량 실황예측자료는 독자기술로 개발한 외삽 기반 레이더 초단기 강수예측모델(MOTION)을 이용하여 10분 단위로 6시간까지 레이더 강수량을 예측한 자료입니다.

주요요소	강수량	
제공범위	전국	
제공기간	2018년 5월 ~ 현재	
생산·갱신주기	실시간(10분 간격 갱신)	
제공형식	BIN, PNG	
제공경로	BIN	공공데이터포털(https://www.data.go.kr) → 제공신청 → 공공데이터 제공신청
	PNG	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량 → (그래픽)레이더분포도 조회 → 초단기 강수예측 분포도
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다. 관측 기반시각과 예측 유효시각을 구분합니다.	

2.합성

- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원

레이더 바람장



개요

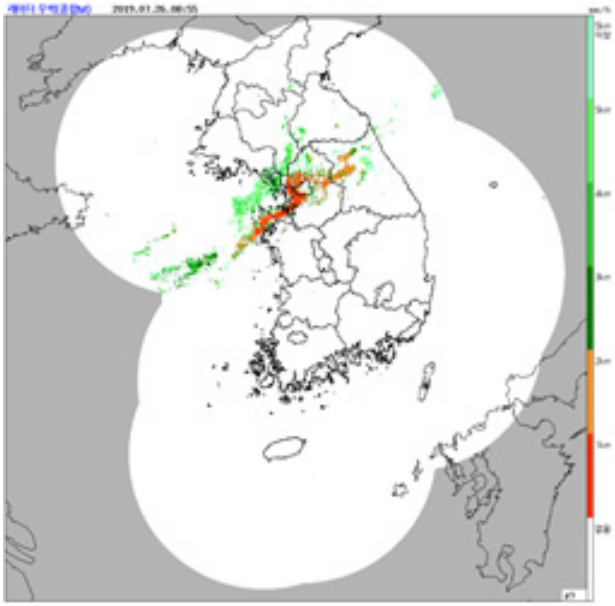
자료상태 : 현행

여러 레이더 지점에서 관측된 시선속도·반사도 자료, 지상관측자료 등을 입력하여 WISSDOM(WInd Synthesis System using Doppler Measurements) 분석을 통해 산출한 3차원 레이더 바람장 영상입니다.

주요요소	동-서 바람성분, 남-북 바람성분, 연직 바람, 합성반사도	
제공범위	전국	
제공기간	2012년 11월 ~ 현재	
생산·갱신주기	실시간(10분 간격 갱신)	
제공형식	BIN, PNG	
제공경로	BIN	공공데이터포털(https://www.data.go.kr) → 제공신청 → 공공데이터 제공신청
	PNG	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량 → (그래픽)레이더분포도 조회, 레이더 → 레이더강수량 → (그래픽)레이더합성영상 조회
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다. 바람성분과 고도의 기준을 확인한 후 풍향·풍속을 계산합니다.	

레이더 우박

2.합성
- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원



개요

자료상태 : 현행

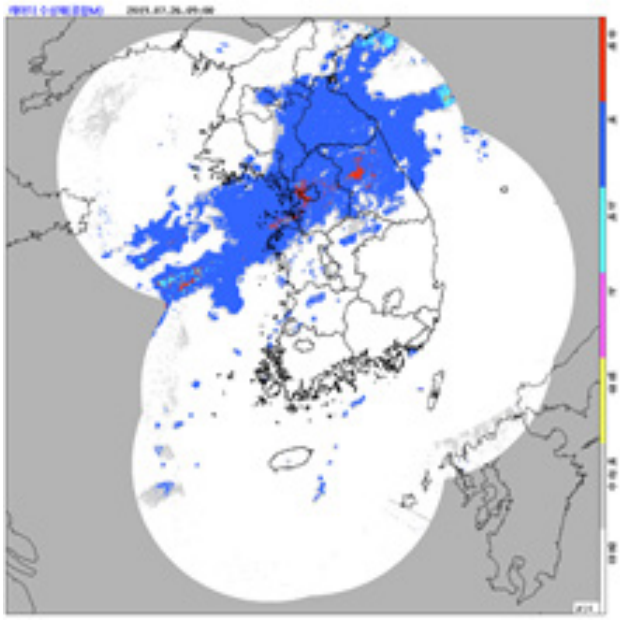
레이더 관측을 바탕으로 우박과 관련된 에코의 위치·고도 등을 산출한 자료입니다. 우박 탐지 산출물은 지상에서 관측한 우박의 크기나 피해 발생 기록과 구분하여 해석합니다.

주요요소	강수량, 우박에코 최저고도	
제공범위	전국	
제공기간	2018년 1월 ~ 현재	
생산·갱신주기	분(5분 주기) 자료	
제공형식	BIN, PNG	
제공경로	BIN	공공데이터포털(https://www.data.go.kr) → 제공신청 → 공공데이터 제공신청
		기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량 → 레이더 합성파일 목록
	PNG	공공데이터포털(https://www.data.go.kr)※ 제공신청 → 공공데이터 제공신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다. 탐지 고도나 분류값을 지상 우박 발생 횟수로 직접 집계하지 않습니다.	

2.합성

- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원

레이더 수상체



값	NCAR 분류	기상레이더센터 분류	사용 명칭
0	-	자료없음	무강수
1	구름(CL)	비	비
2	이슬비(DRZ)		
3	약한비(LR)		
4	중간비(MR)		
5	강한비(HR)	강한비	
6	우박(HA)	비+우박	우박
7	비+우박(RH)		
8	싸락눈+작은우박(GSH)		
9	싸락눈+비(GR)		
10	건설(DS)	건설	눈
11	습설(WS)	습설	눈비
14	과냉각수적(SLD)		
12	등방 빙정(IC)	빙정	빙정
13	비등방 빙정(IIC)		
50	-	비기상 에코	비기상

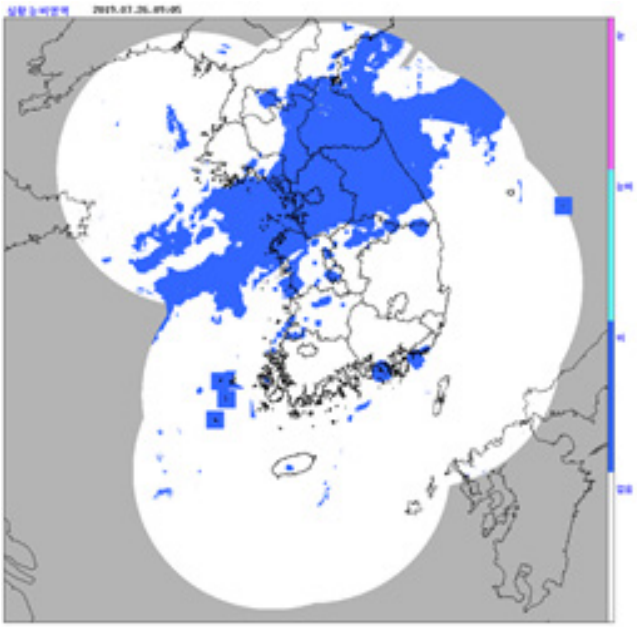
개요

자료상태 : 현행

레이더 수상체는 기상레이더를 통해 관측되는 강수 에코를 14종의 대기수상체로 분류한 자료입니다.

주요요소	레이더 수상체
제공범위	전국
제공기간	2018년 1월 ~ 현재
생산·갱신주기	분(5분 주기) 자료
제공형식	BIN, PNG
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 레이더 → 레이더강수량 → 레이더 합성파일 목록, 레이더 → 레이더강수량 → (그래픽)레이더합성영상 조회
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다. 분류코드는 코드표에 따라 해석하고 평균하지 않습니다.

레이더 눈/비



개요

자료상태 : 현행

레이더 눈/비 자료는 관측자료와 수치예보자료를 3차원 객관분석 방법을 이용하여 판별한 눈/비 영역자료입니다.

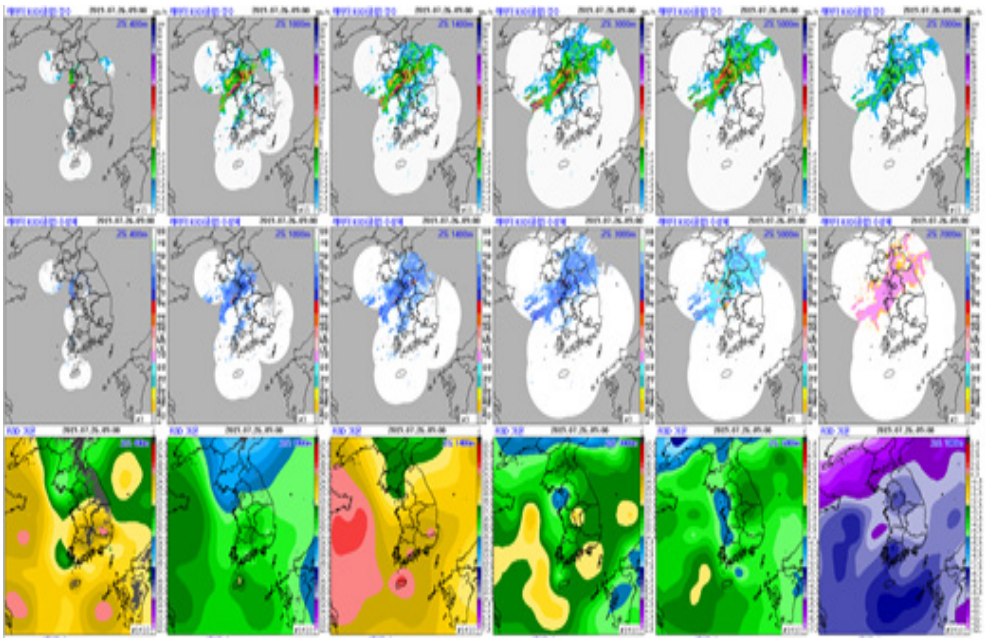
주요요소	강수유형, 눈비가능영역, 습구온도, 레이더강수량, AWS 15분 강수량
제공범위	전국
제공기간	2018년 1월 ~ 현재
생산·갱신주기	분(5분 주기) 자료
제공형식	BIN
제공경로	공공데이터포털(https://www.data.go.kr) → 제공신청 → 공공데이터 제공신청
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다. 강수형태 코드는 코드표에 따라 해석합니다.

2.합성
- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원

2.합성

- 레이더 강수량
- 레이더 강수량 실황예측
- 레이더 바람장
- 레이더 우박
- 레이더 수상체
- 레이더 눈/비
- 레이더 3차원

레이더 3차원



개요

자료상태 : 현행

레이더 관측자료를 3차원 격자로 구성한 산출물입니다. 수평 위치와 연직 고도별 에코 분포를 분석할 수 있으며, 격자 좌표·고도·결측값을 확인하여 이용합니다.

주요요소	반사도(CZ), 교차상관계수(RH), 차등반사도(DR), 비차등위상차(KD), 기온(ta), 이슬점(td), 기압(pa), 대기수상체(HCI, HC)
제공범위	전국
제공기간	• CZ, DR, RH : 2016년 1월 ~ 현재 • HCI, HC, KD, ta, td, pa : 2018년 1월 ~ 현재
생산·갱신주기	분(5분 주기) 자료
제공형식	BIN
제공경로	공공데이터포털(https://www.data.go.kr) → 제공신청 → 공공데이터 제공신청
이용방법	지점 또는 합성영역·관측시각·산출물·요소를 선택하여 파일목록을 조회한 뒤 다운로드합니다. 파일의 좌표·격자·단위·스케일계수와 결측 영역을 확인합니다. BIN·UF·NetCDF는 해당 형식을 지원하는 분석 도구로 읽습니다.

04

세계기상전문(GTS)

Global Telecommunication System



1.관측

- 지상(SYNOP)
- 고층풍(PILOT)
- 상층풍 고층(PIBAL)
- 육상이동관측소 고층풍 (PILOT MOBILE)
- 고층(TEMP)
- 육상이동관측소 고층 (TEMP MOBILE)
- 해상관측소(SHIP)
- 부이(BUOY)
- 해상관측소 고층풍(PILOT SHIP)
- 선박 고층(TEMP SHIP)
- 드롭존데(TEMP DROP)
- 항공기 고층기상 (CODAR)
- 항공기 기상관측 보고 (AIREP, AMDAR)
- 공항 정시기상관측보고 (METAR)
- 공항 특별기상관측보고 (SPECI)

세계기상전문(GTS) 관측

개요

자료상태 : 현행

기상전문이란 기상관측에 의해 생산 · 수집된 자료를 교환하기 위해 송신에 적합하게 WMO에 의해서 명시된 국제적 규정에 따라 만든 자료를 말합니다.
세계기상통신망인 GTS(Global Telecommunication System)을 통해 자료를 유통합니다.
기상전문은 기상요소를 표시하는 문자와 숫자로 구성되어 있으며, 이러한 문 · 숫자는 관측한 기상요소의 값, 또는 상태를 나타냅니다.

※ GTS 전문 종류 첨부

주요요소	관측 평균 · 최저 · 최고 기온, 평균 습도, 평균 해면기압, 중하층 · 평균 전운량, 합계 강수량, 평균 풍속, 평균 이슬점온도, 평균 현지기압 등	
제공기간	1996년 1월 ~ 현재(요소별로 다름)	
생산·갱신주기	일, 시간 자료	
제공형식	TXT, XML	
제공경로	TXT	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청
		기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상관측 → 세계기상전문(GTS)
	XML	항공기상청(https://amo.kma.go.kr) → 정보마당 → 공공데이터개방 → AFTN항공기상전문(METAR, TAF)
이용방법	WMO-No. 306 Manual on Codes 참조 XML 포맷은 오픈API 서버를 통해 자료를 제공(단, 일부 전문에 한함)	

GTS전문 종류

1	SYNOP	육상관측소의 지상관측
2	PILOT	육상고정관측소의 상층풍관측
3	PIBAL	측풍기구관측
4	PILOT MOBILE	이동식 육상관측소의 상층풍관측
5	TEMP	육상고정관측소의 상층기압, 기온, 습도와 바람관측
6	TEMP MOBILE	이동식 육상관측소의 상층기압, 기온, 습도와 바람관측
7	SHIP	해상관측소의 지상관측
8	BUOY	부이관측
9	PILOT SHIP	해상관측소의 상층풍관측
10	TEMP SHIP	해상관측소의 상층기압, 기온, 습도와 바람관측
11	TEMP DROP	운반용기구 또는 항공기에서 낙하된 라디오존데에 의한 상층기압, 기온, 습도와 바람관측
12	CODAR	항공기의 상층기상관측(기상감시용 항공기 제외)
13	AIREP, AMDAR	항공기 기상관측(항공기 기상자료중계)
14	METAR	항공기상정시관측
15	SPECI	항공기상특별관측
16	SATEM	기상위성의 상층기압, 기온 및 습도 보고
17	SAREP	기상위성 구름자료의 종관해석
18	SARAD	기상위성의 청천복사량관측
19	SATOB	기상위성의 바람, 지표면온도, 구름, 습도, 복사관측
20	CLIMAT	육상관측소의 일·월기후자료값
21	CLIMAT TEMP	기후 고층
22	RADOB	레이더기상관측

1.관측

- 지상(SYNOP)
- 고층풍(PILOT)
- 상층풍 고층(PIBAL)
- 육상이동관측소 고층풍 (PILOT MOBILE)
- 고층(TEMP)
- 육상이동관측소 고층 (TEMP MOBILE)
- 해상관측소(SHIP)
- 부이(BUOY)
- 해상관측소 고층풍(PILOT SHIP)
- 선박 고층(TEMP SHIP)
- 드롭존데(TEMP DROP)
- 항공기 고층기상 (CODAR)
- 항공기 기상관측 보고 (AIREP, AMDAR)
- 공항 정시기상관측보고 (METAR)
- 공항 특별기상관측보고 (SPECI)

2.예특보

- 공항예보(TAF, 국내외)
- 태풍예보(TYP)
- 화산재예보(FV)
- 쓰나미경보(WE)
- SIGMET(위험기상정보)(WS)
- 태풍경보(WT)
- 지진·해일특보(SE)
- 화산재구름경보(WV)
- 경보 및 날씨정보(WW)

세계기상전문(GTS) 예특보

개요

자료상태 : 현행

기상전문이란 기상관측에 의해 생산 · 수집된 자료를 교환하기 위해 송신에 적합하게 WMO에 의해서 명시된 국제적 규정에 따라 만든 자료를 말합니다.
세계기상통신망인 GTS(Global Telecommunication System)을 통해 자료를 유통합니다.
기상전문은 기상요소를 표시하는 문자와 숫자로 구성되어 있으며, 이러한 문 · 숫자는 관측한 기상요소의 값, 또는 상태를 나타냅니다.

※ GTS 전문 종류 첨부

주요요소	예보 풍향, 풍속, 시정, 날씨, 운량, 운고, 태풍위치 등	
제공기간	1996년 1월 ~ 현재(요소별로 다름)	
생산·갱신주기	일, 시간 자료	
제공형식	TXT, XML	
제공경로	TXT	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청
	XML	항공기상청(http://amo.kma.go.kr) → 정보마당 → 공공데이터개방 → AFTN항공기상전문(METAR, TAF)
이용방법	WMO NO.-306 Manual on Codes 참조 XML 포맷은 오픈API 서버를 통해 자료를 제공(단, 일부 전문에 한함)	

GTS전문 종류

1	FV	화산재경보예보
2	FK	열대저기압경보예보
3	SE	지진정보
4	TAF	공항예보(국내외)
5	TYP	태풍정보
6	WC	열대저기압 경보(SIGMET)
7	WD, WW	특보 기상 요약
8	WE	쓰나미경보
9	WH	해양·해안의 홍수 경보
10	WS	SIGMET(항공기상특보)
11	WT	태풍경보
12	WV	화산관측경보 (화산재 구름)

05

기상예보

Meteorological Forecast



1.날씨예보
• 초단기실황
• 초단기예보
• 단기예보
• 중기예보
• 영향예보

초단기실황



개요

자료상태 : 현행

초단기 실황정보란 단기예보 구역(5km 해상도 격자)에 대한 대표 AWS 관측값을 의미합니다.

※ 2018년 7월11일 이전 제공값에 대해서는 AWS 관측값이 아닌 분석값을 제공

주요요소	하늘상태, 기온, 습도, 동서바람, 남북바람, 강수형태, 1시간 강수량, 풍향, 풍속	
제공범위	5km 해상도 격자	
발표시각	10분 간격으로 생산	
제공기간	• 2010년 6월 15일 17:00KST(시행일 기준) • 하늘상태, 낙뢰: 2010년 6월 15일 17:00KST(시행일 기준) ~ 2018년 10월 10일 10:00KST(서비스종료 기준)	
제공형식	XML, CSV	
제공경로	XML	공공데이터포털(https://www.data.go.kr) → '기상청 날씨예보 정보 검색
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 동네예보
이용방법	단기예보 격자 좌표(nx, ny), 기준일·기준시각과 응답형식을 설정합니다. 조회 가능한 최근 자료와 과거 보관자료의 경로를 구분하고, 응답 오류코드·전체 건수·페이지를 확인합니다. 발표시각과 자료 대상시각을 구분합니다. 실황값은 해당 격자의 값으로 해석합니다.	

초단기예보



개요

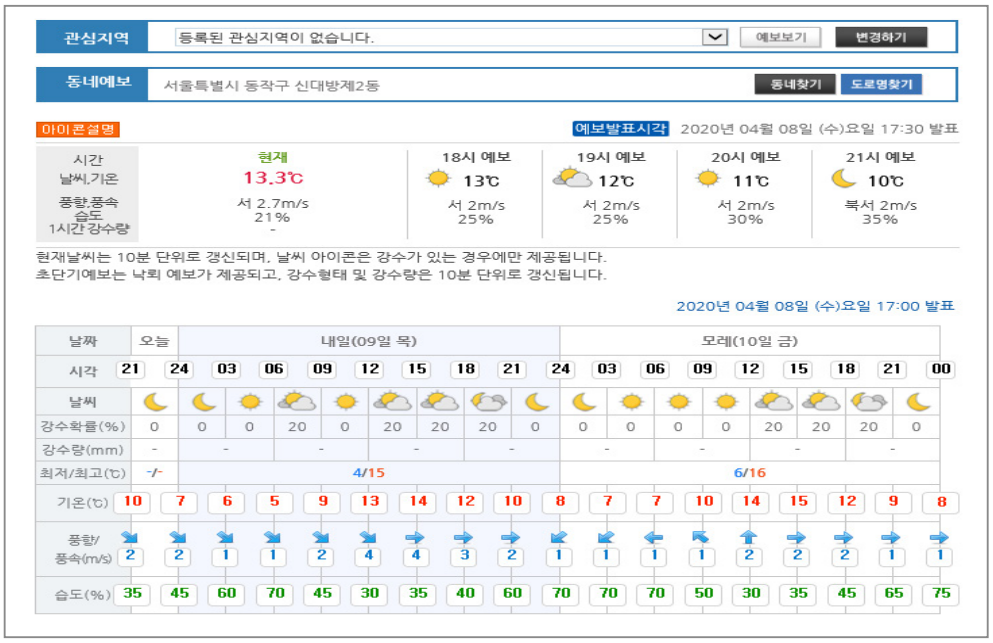
자료상태 : 현행

초단기 예보란 예보시점부터 6시간 이내에 대하여 행하는 예보를 말합니다. 짧은 시간에 발생·소멸하는 위험기상에 신속하게 대처하고, 예보 간 연속성을 확보하기 위해 발표시간 1시간 후부터 최대 6시간까지의 예보를 제공합니다.

주요요소	하늘상태, 기온, 습도, 동서바람, 남북바람, 강수형태, 1시간 강수량, 풍향, 풍속	
제공범위	5km 해상도 격자	
발표시각	10분 간격 생산	
유효시간	발표시각 1시간 이후부터 최대 6시간	
예보간격	1시간 간격	
제공기간	2010년 06월 15일 17:30KST(시행일 기준) ~ 현재	
제공형식	XML, CSV	
제공경로	XML	공공데이터포털(https://www.data.go.kr) → 기상청 날씨예보 정보 검색
	CSV	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 기상예보 → 동네예보 → 초단기예보
이용방법	단기예보 격자 좌표(nx, ny), 기준일·기준시각과 응답형식을 설정합니다. 조회 가능한 최근 자료와 과거 보관자료의 경로를 구분하고, 응답 오류코드·전체 건수·페이지를 확인합니다. 발표시각과 자료 대상시각을 구분합니다. 발표 후 제공 지연을 고려하여 조회시각을 선택합니다.	

1.날씨예보
- 초단기실황
- 초단기예보
- 단기예보
- 중기예보
- 영향예보

단기예보



개요

자료상태 : 현행

단기예보란 오늘부터 모레(글피)까지의 날씨를 3시간 단위로 세분화하고, 전국을 5km 격자로 나누어 지역별 기상상황을 상세하게 제공하는 예보입니다.

주요요소	하늘상태, 풍향, 풍속, 강수유무, 강수확률, 최저기온, 최고기온, 유의파고	
제공범위	5km 해상도 격자	
발표시각	3시간 간격(02시부터 23시까지) 일 8회	
유효시간	오늘, 내일, 모레, 글피	
제공기간	오전, 오후	
생산·갱신주기	2010년 06월 15일 17:30KST(시행일 기준) ~ 현재	
제공형식	XML, CSV	
제공경로	XML	공공데이터포털(https://www.data.go.kr) → 기상청 날씨예보 정보 검색
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 동네예보
이용방법	단기예보 격자 좌표(nx, ny), 기준일·기준시각과 응답형식을 설정합니다. 조회 가능한 최근 자료와 과거 보관자료의 경로를 구분하고, 응답 오류코드·전체 건수·페이지를 확인합니다. 발표시각과 자료 대상시각을 구분합니다. 강수량·적설량의 문자형 범주값은 숫자 0으로 바꾸지 않고 기준표에 따라 처리합니다.	

중기예보



육상예보 | 중기예보

도움말 | 예보용어보기 | 인쇄

전국 | 서울·경기도 | 강원도 | 충청남·북도 | 전라남·북도 | 경상남·북도 | 제주특별자치도

기상전망 — 발표시각 2020년 04월 08일 (수)요일 06:00 발표 (총괄예보관: 선지호)

- 전국
○ (강수) 12일(일)은 강원영동과 제주도, 17일(금)은 전국, 18일(토)은 강원영동에 비가 오겠습니다.
※ 12일(일)은 대기가 불안정하여 내륙을 중심으로 소나성 비가 내릴 가능성이 있습니다.
○ (건조) 비가 오는 날이 적어 대기가 매우 건조하겠으니, 산불 등 화재예방에 각별히 유의하기 바랍니다.
○ (기온) 낮 기온은 어제(7일, 13~22도)와 비슷하거나 조금 높은 12~23도로 분근하겠습니다.
○ (주말전망) 전국이 가끔 구름많겠으나, 12일(일)은 강원영동과 제주도에 비가 오겠습니다.
낮 기온은 12~19도로 분근하겠으나, 낮과 밤의 기온차가 10도 이상으로 크겠습니다.

육상날씨
- 16일~18일은 오전, 오후 구분 없이 일별로 예보합니다.
- 2019년 4월 30일부터 중기 육상날씨 신뢰도 정보 대신 강수확률 정보를 제공합니다.

지역	11일(토)		12일(일)		13일(월)		14일(화)		15일(수)		16일(목)	17일(금)	18일(토)
	오전	오후	오전	오후	오전	오후	오전	오후	오전	오후			
서울 인천 경기도	20%	30%	20%	40%	10%	10%	0%	0%	0%	0%	40%	60%	40%
강원도 영서	10%	20%	20%	40%	10%	10%	0%	0%	0%	10%	40%	70%	40%
강원도 영동	10%	20%	60%	60%	20%	10%	0%	10%	10%	10%	40%	80%	70%
대전 세종 충청남도	30%	20%	20%	40%	10%	10%	0%	0%	0%	0%	40%	60%	40%
충청북도	30%	20%	20%	40%	10%	10%	0%	0%	0%	0%	40%	60%	20%
광주 전라남도	30%	40%	20%	40%	20%	20%	0%	0%	0%	0%	40%	70%	30%
전라북도	30%	30%	20%	40%	10%	10%	0%	0%	0%	0%	40%	60%	20%
부산 울산 경상남도	40%	40%	40%	40%	20%	20%	0%	0%	0%	10%	40%	60%	30%
대구 경상북도	40%	40%	40%	40%	20%	20%	0%	0%	0%	0%	40%	60%	30%

개요

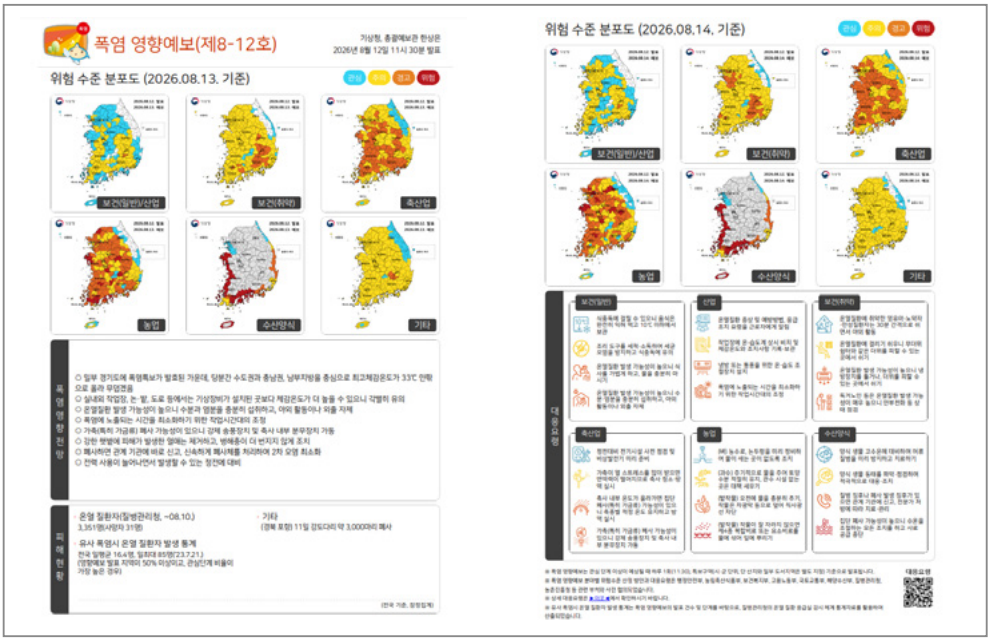
자료상태 : 현행

중기예보는 발표일로부터 6일째 되는 날부터 11일째 되는 날까지의 기간에 대한 예보입니다. 중기예보 통보문에서는 6~8일까지는 오전과 오후로 구분하여 예보하고, 9~11일까지는 일 단위로 구분합니다. 2026년 11월 12일부터는 시공간 해상도 향상, 강수가능성 등 격자 형태의 디지털 중기예보 체계를 바탕으로 6~8일은 3시간 간격, 9~11일은 6시간 단위로 상세히 구분하여 예보자료를 제공합니다.

주요요소	기온, 최고기온, 최저기온, 하늘상태, 강수확률, 강수형태, 유의파고, 강수가능성, 강수변동성, 기온 확률 범위	
제공범위	서울·인천·경기도, 강원도 영서, 강원도 영동, 충청북도, 대전·세종·충청남도, 전라북도, 광주·전라남도, 대구·경상북도, 부산·울산·경상남도, 제주도	
발표시각	일 2회 발표(06시, 18시) 2026년 11월 11일까지 일 1회 발표(18시) 2026년 11월 12일부터	
유효시간	발표일로부터 6일째 되는 날부터 11일째 되는 날	
제공기간	2012년 12월 18일 18:00KST(시행일기준) ~ 현재 (2013년 10월 15일 18:00KST부터 7일 → 10일 예보기간 연장운영) ※ 신뢰도: 2013년 10월 15일 18:00KST(시행일 기준) ~ 2019년 4월 30일 06:00KST(서비스종료 기준) ※ 강수확률, 최고·최저기온 상·하한 범위: 2019년 4월 30일 18:00KST(시행일 기준) ~ 현재	
예보간격	12~24시간 간격(2026년 11월 11일까지), 3~6시간 간격(2026년 11월 12일부터)	
제공형식	XML, CSV	
제공경로	XML	공공데이터포털(https://www.data.go.kr) → 기상청 날씨예보 정보 검색
	CSV	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 기상예보 → 동네예보 → 중기예보
이용방법	조회할 요소에 맞는 중기 예보구역코드와 발표시각을 선택합니다. 기온·육상·해상·개황 서비스의 코드와 예측기간을 구분하며, 발표시각(KST)과 예측 대상 날짜를 함께 저장합니다.	

1.날씨예보
- 초단기실황
- 초단기예보
- 단기예보
- 중기예보
- 영향예보

영향예보



개요

자료상태 : 현행

영향예보는 날씨 뿐만 아니라 시간과 장소에 따라 달라지는 날씨의 영향을 고려하여 기상 현상별 위험수준에 따른 분야별 상세 영향정보와 대응요령을 제공합니다.
이를 통해 유관기관에 실효적 정보를 제공하여 방재업무를 지원하고, 기상재해로부터 국민의 안전을 보호하고자 합니다.

주요요소	폭염, 한파(영향 전망, 피해 현황, 분야별 위험수준 및 대응요령)	
제공범위	전국 235개 특보구역	
발표시각	일 1회 발표(11시 30분)	
유효시간	예보일로부터 1일	
제공기간	2019년 6월 ~ 현재	
예보간격	24시간 간격(발표 기준에 부합되지 않을 때 미발표)	
제공형식	PDF, JSON, XML, CSV	
제공경로	PDF	방재기상플랫폼(https://dmdw.kma.go.kr) → 예‖특보 → 특·정보 → 영향예보
	JSON+XML	공공데이터포털(https://www.data.go.kr) → 기상청 영향예보 조회서비스 검색
	CSV	공공데이터포털(https://www.data.go.kr) → 기상청 영향예보 검색
		기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 영향예보
이용방법	발표일·지역·영향 분야를 선택해 위험수준과 대응요령을 확인합니다. 예보 대상기간과 단계별 기준을 함께 기록하고, 기상특보의 발효 상태도 별도로 확인합니다.	

기상특보



개요

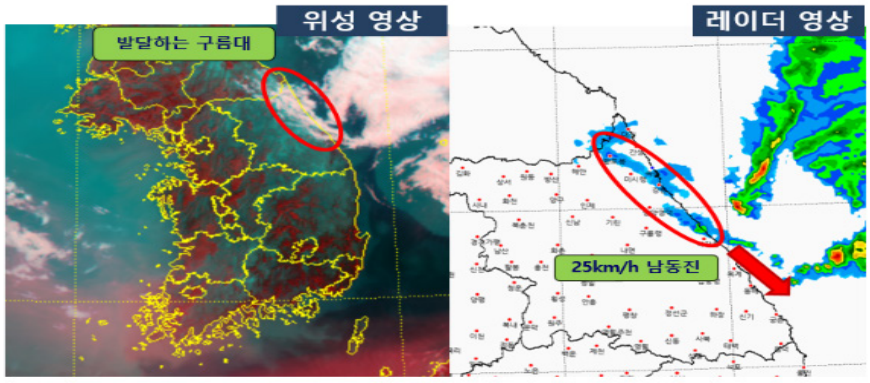
자료상태 : 현행

호우, 대설, 폭풍해일 등 11개 기상현상으로 인해 중대한 재해발생이 예상될 때 해당 지역에 대하여 기상특보의 발표 기준에 따라 주의보 및 경보로 구분하여 발표합니다.(단 폭염은 주의보, 경보, 중대경보로 구분하여 발표하고, 열대야는 주의보만 발표)
기상특보는 육상과 해상을 대상으로 발표하여 국민의 생명과 재산을 지키는데 최선을 다합니다.

주요요소	강풍, 풍랑, 호우, 대설, 건조, 폭풍해일, 한파, 태풍, 황사, 폭염, 열대야	
제공범위	전국 235개 시·군 단위와 44개 해역(2026년 6월 기준)	
발표시각	특보를 발표한 시각	
유효시간	발효시각: 발표된 특보가 효력을 발휘하는 최초시각	
제공기간	2004년 6월 30일 ~ 현재	
생산·갱신주기	수시	
제공형식	TXT, XML, CSV	
제공경로	TXT	전자민원시스템(http://minwon.kma.go.kr) 신청
	XML	공공데이터포털(https://www.data.go.kr) → 기상특보정보조회서비스 조회
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 기상특보
이용방법	기간·구역·특보 종류를 정해 발표 이력을 조회합니다. 발표·발효·변경·해제시각을 구분하고, 최신 상태 조회와 과거 이력 집계를 목적에 맞게 선택합니다.	

기상정보

제04-9호 | 기상정보 "강원동해안 약한 비"



< 위성영상과 레이더영상(8일 09시) >

□ 기상상황 및 전망
○ 현재(09시 30분), 전국이 맑으나 강원영동에는 구름많고 산발적으로 빗방울이 떨어지는 곳이 있습니다.

○ 동해상에서 비구름대가 해안으로 유입되어 시속 25km로 남동진하고 있습니다. 이 비구름대의 영향으로 강원동해안에는 오늘(8일) 오전(12시)까지 비가 조금(1mm 내외) 내리는 곳이 있겠고, 강원산지에는 산발적으로 빗방울이 떨어지거나 눈이 날리는 곳이 있겠습니다.

개요

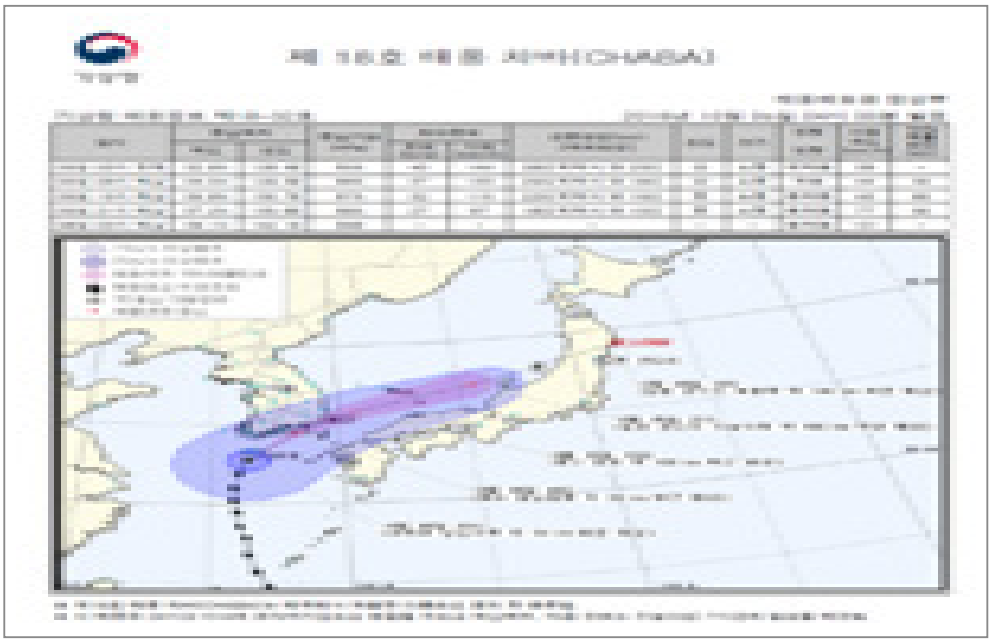
자료상태 : 현행

갑작스러운 기상변화가 예상되거나, 국민과 관계기관에 날씨변화 및 위험기상 상황을 보다 상세하게 알릴 필요가 있을 때 발표하는 자료입니다. 예▮특보 외에 제공하는 기상정보에는 위험기상 상황을 설명하는 기상정보, 예보기간의 주요날씨와 전망, 발생 원인 등을 상세하게 설명하는 날씨해설, 급변하는 기상상황을 신속하게 전달하기 위한 기상속보 등이 있습니다.

주요요소	기상상황과 전망, 위험기상 발생 및 배경, 유의사항 등
제공범위	전국 171개 시·군 단위와 33개 해역
발표시각	정보를 발표한 시각
제공기간	2001년 8월 7일 ~현재 ※ 날씨해설은 2019년 4월 30일 ~ 현재
생산·갱신주기	수시
제공형식	XML
제공경로	공공데이터포털(https://www.data.go.kr) → 기상특보정보조회서비스 조회
이용방법	발표일·지역·정보 종류를 선택해 원문과 첨부자료를 확인합니다. 수정된 발표가 있으면 최신 원문을 적용하고, 내용의 대상기간을 확인합니다.

태풍정보

3.태풍예보
- 태풍정보
- 태풍목록
- 태풍 경로도
- 태풍 발생표
- 태풍 베스트트랙
- 열대저압부정보



개요

자료상태 : 현행

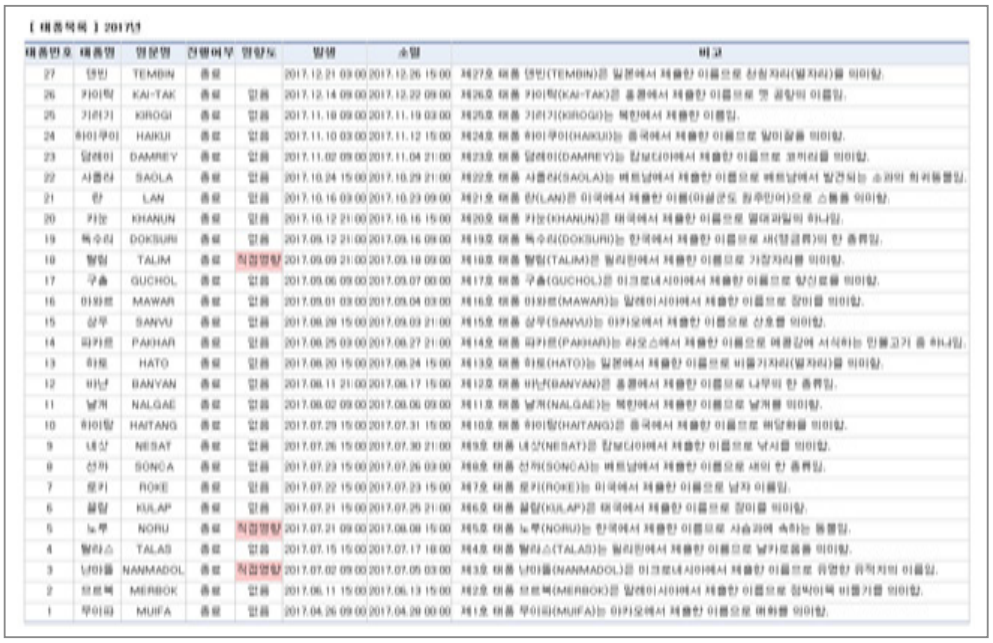
태풍의 현재 상황과 향후 진로와 강도 등에 대한 종합적인 정보(중심위치, 중심기압, 중심부근 최대풍속, 강풍반경, 폭풍반경, 진행방향, 이동속도 등)를 제공합니다.

주요요소	태풍중심위치, 중심기압, 최대풍속, 강풍반경, 강도, 크기, 진행방향, 이동속도	
제공범위	북서태평양 지역	
발표시각	4, 10, 16, 22시 및 필요 시 수시 발표	
제공기간	2001년부터 현재 태풍까지(단, 날씨누리는 현재 활동중인 태풍에 대해 제공)	
제공형식	PNG, XML, CSV, PDF, TXT	
제공경로	PNG	기상청 날씨누리(https://www.weather.go.kr) ※ 위험기상 → 태풍 → 태풍통보문
	XML	공공데이터포털(https://www.data.go.kr) → 태풍정보조회서비스 검색
	CSV, PDF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 태풍예보 → 태풍정보
	TXT	기상청 API허브(https://apihub.kma.go.kr) → 태풍 → 태풍정보
이용방법	연도·태풍번호 또는 기간을 선택합니다. 발표시각·위치·중심기압·최대풍속과 시간 기준을 확인하고, 예보경로·당시 발표자료·사후 분석자료를 구분합니다.	

3.태풍예보

- 태풍정보
- 태풍목록
- 태풍 경로도
- 태풍 발생표
- 태풍 베스트트랙
- 열대저압부정보

태풍목록



[태풍목록] 2017년

태풍번호	태풍명	영문명	진행여부	영향도	발생	소멸	비고
27	뎬빈	TEMBIN	종료		2017.12.21 03:00	2017.12.26 15:00	제27호 태풍 뎬빈(TEMBIN)은 일본에서 제출한 이름으로 산청자리(별자리)를 의미함.
26	카이탁	KAI-TAK	종료	영향	2017.12.14 09:00	2017.12.22 09:00	제26호 태풍 카이탁(KAI-TAK)은 홍콩에서 제출한 이름으로 맛 공항의 이름임.
25	기러기	KIROGI	종료	영향	2017.11.18 09:00	2017.11.19 03:00	제25호 태풍 기러기(KIROGI)는 북한에서 제출한 이름임.
24	하이쿠이	HAIKUI	종료	영향	2017.11.10 03:00	2017.11.12 15:00	제24호 태풍 하이쿠이(HAIKUI)는 중국에서 제출한 이름으로 일이삼을 의미함.
23	담레이	DAMREY	종료	영향	2017.11.02 09:00	2017.11.04 21:00	제23호 태풍 담레이(DAMREY)는 캄보디아에서 제출한 이름으로 코끼리를 의미함.
22	사올라	SAOLA	종료	영향	2017.10.24 15:00	2017.10.29 21:00	제22호 태풍 사올라(SAOLA)는 베트남에서 제출한 이름으로 베트남에서 발견되는 소과의 희귀동물임.
21	란	LAN	종료	영향	2017.10.16 03:00	2017.10.23 09:00	제21호 태풍 란(LAN)은 미국에서 제출한 이름(아설군도 원주민어)으로 스톰을 의미함.
20	카눈	KHANUN	종료	영향	2017.10.12 21:00	2017.10.16 15:00	제20호 태풍 카눈(KHANUN)은 태국에서 제출한 이름으로 열대과일의 하나임.
19	독수리	DOKSURI	종료	영향	2017.09.12 21:00	2017.09.16 09:00	제19호 태풍 독수리(DOKSURI)는 한국에서 제출한 이름으로 새(천금류)의 한 종류임.
18	탈림	TALIM	종료	직접영향	2017.09.09 21:00	2017.09.18 09:00	제18호 태풍 탈림(TALIM)은 필리핀에서 제출한 이름으로 가장자리를 의미함.
17	구촐	GUCHOL	종료	영향	2017.09.06 09:00	2017.09.07 00:00	제17호 태풍 구촐(GUCHOL)은 미크로네시아에서 제출한 이름으로 탁신로를 의미함.
16	마와르	MAWAR	종료	영향	2017.09.01 03:00	2017.09.04 03:00	제16호 태풍 마와르(MAWAR)는 말레이시아에서 제출한 이름으로 장미를 의미함.
15	산우	SANVU	종료	영향	2017.08.29 15:00	2017.09.03 21:00	제15호 태풍 산우(SANVU)는 마카오에서 제출한 이름으로 산호를 의미함.
14	파카르	PAKHAR	종료	영향	2017.08.25 03:00	2017.08.27 21:00	제14호 태풍 파카르(PAKHAR)는 라오스에서 제출한 이름으로 메콩강에 서식하는 민물고기 중 하나임.
13	하토	HATO	종료	영향	2017.08.20 15:00	2017.08.24 15:00	제13호 태풍 하토(HATO)는 일본에서 제출한 이름으로 비둘기자리(별자리)를 의미함.
12	바난	BANYAN	종료	영향	2017.08.11 21:00	2017.08.17 15:00	제12호 태풍 바난(BANYAN)은 홍콩에서 제출한 이름으로 나무의 한 종류임.
11	날개	NALGAE	종료	영향	2017.08.02 09:00	2017.08.06 09:00	제11호 태풍 날개(NALGAE)는 북한에서 제출한 이름으로 날개를 의미함.
10	하이탕	HAITANG	종료	영향	2017.07.29 15:00	2017.07.31 15:00	제10호 태풍 하이탕(HAITANG)은 중국에서 제출한 이름으로 해당화를 의미함.
9	네삿	NESAT	종료	영향	2017.07.26 15:00	2017.07.30 21:00	제9호 태풍 네삿(NESAT)은 캄보디아에서 제출한 이름으로 낚시를 의미함.
8	손까	SONCA	종료	영향	2017.07.23 15:00	2017.07.26 03:00	제8호 태풍 손까(SONCA)는 베트남에서 제출한 이름으로 새의 한 종류임.
7	로키	ROKE	종료	영향	2017.07.22 15:00	2017.07.23 15:00	제7호 태풍 로키(ROKE)는 미국에서 제출한 이름으로 남자 이름임.
6	쿨랍	KULAP	종료	영향	2017.07.21 15:00	2017.07.25 21:00	제6호 태풍 쿨랍(KULAP)은 태국에서 제출한 이름으로 장미를 의미함.
5	노루	NORU	종료	직접영향	2017.07.21 09:00	2017.08.08 15:00	제5호 태풍 노루(NORU)는 한국에서 제출한 이름으로 사슴과에 속하는 동물임.
4	탈라스	TALAS	종료	영향	2017.07.15 15:00	2017.07.17 18:00	제4호 태풍 탈라스(TALAS)는 필리핀에서 제출한 이름으로 날카로움을 의미함.
3	난마돌	NANMADOL	종료	직접영향	2017.07.02 09:00	2017.07.05 03:00	제3호 태풍 난마돌(NANMADOL)은 미크로네시아에서 제출한 이름으로 유명한 유적지의 이름임.
2	므르복	MERBOK	종료	영향	2017.06.11 15:00	2017.06.13 15:00	제2호 태풍 므르복(MERBOK)은 말레이시아에서 제출한 이름으로 참복이목 비늘기를 의미함.
1	무이파	MUIFA	종료	영향	2017.04.26 09:00	2017.04.29 00:00	제1호 태풍 무이파(MUIFA)는 마카오에서 제출한 이름으로 며하를 의미함.

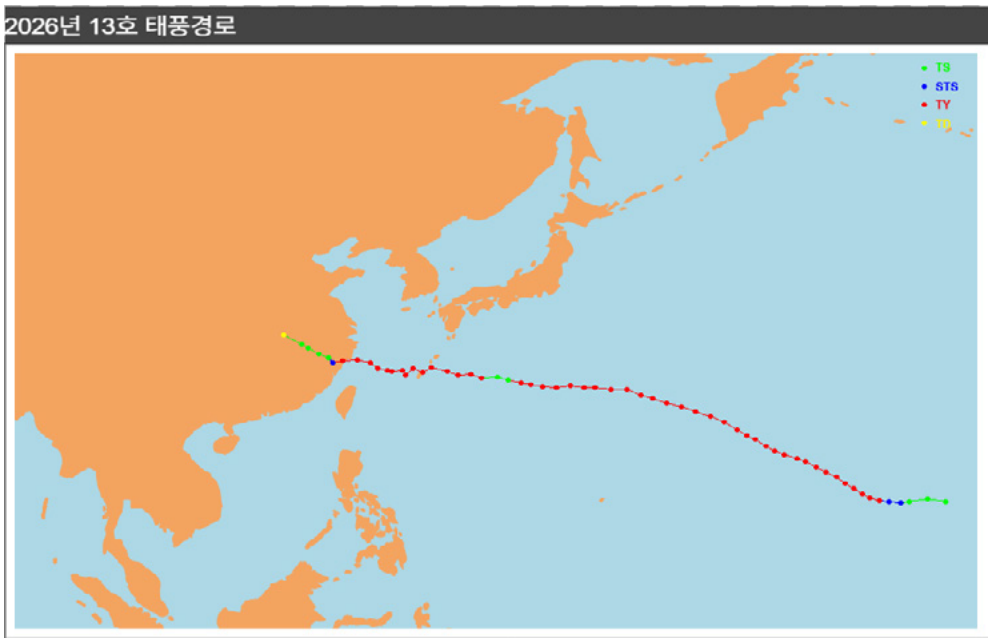
개요

자료상태 : 현행

태풍목록은 발생한 태풍의 태풍번호와 태풍명(영문명 포함), 진행여부, 영향도, 발생일과 소멸일에 대한 정보를 연도별로 보여줍니다.

주요요소	태풍번호, 태풍명, 영문명, 진행여부, 영향도, 발생시각, 소멸시각, 비고(태풍이름의 정의)	
제공범위	북서태평양 지역	
제공기간	2001년부터 현재 태풍까지	
생산·갱신주기	연 자료	
제공형식	XML, CSV	
제공경로	XML	공공데이터포털(https://www.data.go.kr) → 태풍정보조회서비스 검색
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 태풍예보 → 태풍목록
이용방법	연도별 태풍 목록에서 태풍번호·명칭을 확인합니다. 연도와 번호를 함께 식별자로 사용하고, 발생 태풍과 우리나라 영향 태풍의 구분 기준을 확인합니다.	

태풍 경로도



개요

자료상태 : 현행

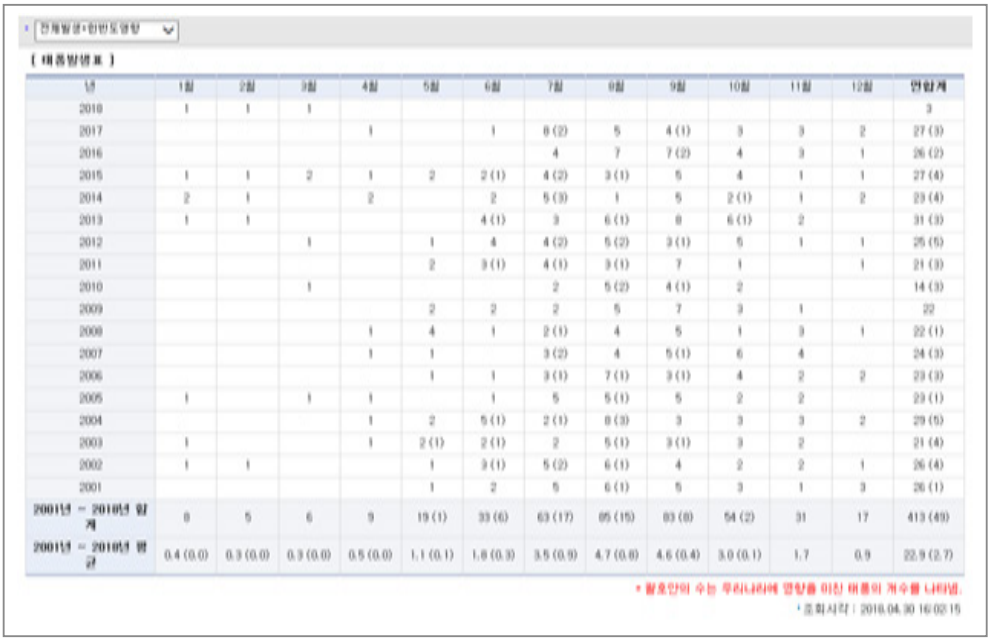
북서태평양에서 과거에 발생했던 태풍들의 발생부터 종료까지의 경로를 확인합니다.

주요요소	태풍경로(태풍발생일과 태풍종료일 표시)	
제공범위	북서태평양 지역	
제공기간	2001년부터 현재 태풍까지	
생산·갱신주기	연 자료(각 태풍별)	
제공형식	PNG, CSV, TXT	
제공경로	PNG, CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 태풍예보 → 태풍경로
이용방법	연도·태풍번호 또는 기간을 선택합니다. 발표시각·위치·중심기압·최대풍속과 시간 기준을 확인하고, 예보경로·당시 발표자료·사후 분석자료를 구분합니다.	

3.태풍예보

- 태풍정보
- 태풍목록
- 태풍 경로도
- 태풍 발생표
- 태풍 베스트트랙
- 열대저압부정보

태풍 발생표



(태풍발생표)

년	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월	연합계
2018	1	1	1										3
2017				1		1	8 (2)	5	4 (1)	3	3	2	27 (3)
2016							4	7	7 (2)	4	3	1	26 (2)
2015	1	1	2	1	2	2 (1)	4 (2)	3 (1)	5	4	1	1	27 (4)
2014	2	1		2		2	5 (3)	1	5	2 (1)	1	2	23 (4)
2013	1	1				4 (1)	3	6 (1)	8	6 (1)	2		31 (3)
2012			1		1	4	4 (2)	5 (2)	3 (1)	5	1	1	25 (5)
2011					2	3 (1)	4 (1)	3 (1)	7	1		1	21 (3)
2010			1				2	5 (2)	4 (1)	2			14 (3)
2009					2	2	2	5	7	3	1		22
2008				1	4	1	2 (1)	4	5	1	3	1	22 (1)
2007				1	1		3 (2)	4	5 (1)	6	4		24 (3)
2006					1	1	3 (1)	7 (1)	3 (1)	4	2	2	23 (3)
2005	1		1	1		1	5	5 (1)	5	2	2		23 (1)
2004				1	2	5 (1)	2 (1)	0 (3)	3	3	3	2	29 (5)
2003	1			1	2 (1)	2 (1)	2	5 (1)	3 (1)	3	2		21 (4)
2002	1	1			1	3 (1)	5 (2)	6 (1)	4	2	2	1	26 (4)
2001					1	2	5	6 (1)	5	3	1	3	26 (1)
2001년 ~ 2018년 합계	8	5	6	9	19 (1)	33 (6)	63 (17)	85 (15)	83 (8)	54 (2)	31	17	413 (49)
2001년 ~ 2018년 평균	0.4 (0.0)	0.3 (0.0)	0.3 (0.0)	0.5 (0.0)	1.1 (0.1)	1.8 (0.3)	3.5 (0.9)	4.7 (0.8)	4.6 (0.4)	3.0 (0.1)	1.7	0.9	22.9 (2.7)

* 괄호안의 수는 우리나라에 영향을 미친 태풍의 개수를 나타냄.
* 조회시각 : 2018.04.30 16:02:15

개요

자료상태 : 현행

태풍발생표는 월별, 연도별 북서태평양에서 발생한 태풍과 한반도에 영향을 준 태풍의 개수를 파악할 수 있습니다.

주요요소	태풍발생개수, 영향 개수	
제공범위	북서태평양 지역	
제공기간	1951년부터 현재(날씨누리) 2001년부터 현재(기상자료개방포털)	
생산·갱신주기	연 자료	
제공형식	TXT, CSV	
제공경로	TXT	기상청 날씨누리(https://www.weather.go.kr) → 위험기상 → 태풍 → 태풍발생통계
	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 태풍예보 → 태풍발생표
이용방법	조회 연도와 발생·영향 등 통계 구분을 선택합니다. 통계의 집계 기준과 자료 갱신일을 확인하며, 서로 다른 기준의 통계를 합산하지 않습니다.	

태풍 베스트트랙



개요

자료상태 : 현행

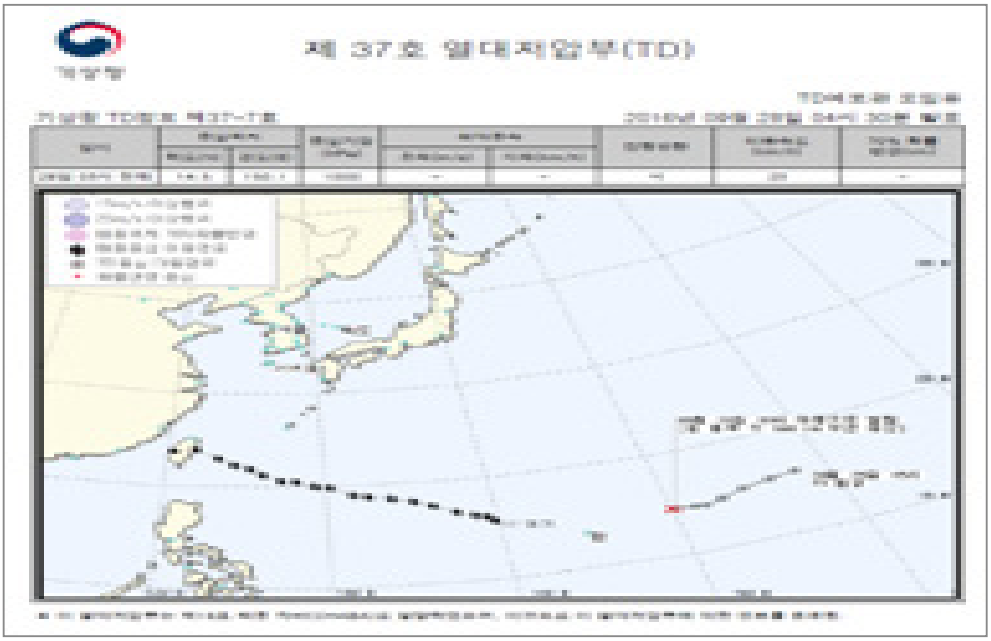
태풍의 위치·중심기압·최대풍속 등을 사후 분석하여 정리한 베스트트랙 자료입니다. 실시간 태풍 예보경로나 당시 발표자료와 차이가 있을 수 있으므로 연구 목적에 맞게 구분하여 이용합니다.

주요요소	등급, 태풍호수, 날짜(년월일시), 위·경도, 중심최대풍속, 중심기압, 강풍·폭풍 장반경, 단반경, 단반경방향, 태풍이름	
제공범위	북서태평양	
제공기간	2015년부터 전년도 태풍	
생산·갱신주기	매년	
제공형식	TXT, PDF	
제공경로	TXT	기상청 API허브(https://apihub.kma.go.kr) → 태풍 → 태풍 베스트트랙
	PDF	기상청 누리집(https://kma.go.kr) → 알림·자료 → 기상간행물 → 베스트트랙
이용방법	연도·태풍번호를 선택하고 제공되는 사후 분석자료를 내려받습니다. 보유기간과 분석 완료 여부를 확인하고, 자료가 없는 연도·태풍은 결측 또는 미제공으로 구분합니다. 실시간 진로 안내에는 이용하지 않습니다.	

3.태풍예보

- 태풍정보
- 태풍목록
- 태풍 경로도
- 태풍 발생표
- 태풍 베스트트랙
- 열대저압부정보

열대저압부정보



개요

자료상태 : 현행

태풍 사전단계 또는 태풍에서 약화된 뒤에도 우리나라에 영향을 줄 수 있는 열대저압부를 대상으로 중심위치, 예상경로 등의 종합적인 정보를 제공합니다.

※ 현재 진행 중인 열대저압부에 한하여 제공(태풍으로 발달 시 태풍정보로 제공)

주요요소	열대저압부 중심위치, 중심기압, 최대풍속, 진행방향, 이동속도, 24시간 예상경로	
제공범위	북서태평양 지역	
발표시각	4, 10, 16, 22시 30분 및 필요시 수시 발표	
제공기간	2013년부터 현재 태풍까지(단, 날씨누리는 현재 활동중인 태풍에 대해 제공)	
제공형식	PNG, CSV, PDF	
제공경로	PNG	기상청 날씨누리(https://www.weather.go.kr) → 위험기상 → 태풍 → 태풍통보문
	CSV, PDF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 기상예보 → 태풍예보 → 태풍정보(TD)
	TXT	기상청 API허브(https://apihub.kma.go.kr) → 태풍 → 태풍정보(TD)
이용방법	연도·태풍번호 또는 기간을 선택합니다. 발표시각·위치·중심기압·최대풍속과 시간 기준을 확인하고, 예보경로·당시 발표자료·사후 분석자료를 구분합니다.	

4.항공예보
• 공항예보(TAF, 국내)
• 이륙예보
• SIGWX(중요기상예보)
• SIGMET(위험기상정보)
• AIRMET(저고도 위험 기상정보)
• 공항경보(WARNING)

공항예보(TAF, 국내)

개요

자료상태 : 현행

공항의 일정 유효기간에 대한 바람·시정·일기·구름 등의 예보를 부호화한 TAF 전문입니다. 실제 관측 전문인 METAR/SPECI와 구분하며, 변경·수정 전문과 변화군의 유효시간을 확인합니다.

주요요소	풍향, 풍속, 시정, 일기현상, 운량, 운고, 기온(최고·최저)
제공범위	인천, 김포, 제주, 무안, 울산, 김해, 청주, 대구, 여수, 양양, 광주, 포항경주, 사천, 원주, 군산
발표시각	일 4회 (05, 11, 17, 23 UTC)
유효시간	발표시각 1시간 이후 ~ 30시간 이내
제공기간	2021년 6월 18일 ~ 현재
제공형식	XML
제공경로	API허브(https://apihub.kma.go.kr) → 항공기상 → 공항예·특보
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.

이륙예보

개요

자료상태 : 현행

항공기 이륙 지원을 위해 제공하는 공항의 기상예보입니다. 해당 공항과 예보 유효시간에 맞는 바람·기온·기압 등 제공요소를 확인하여 이용합니다.

주요요소	풍향, 풍속, 기온, 기압
제공범위	인천, 김포, 제주, 무안, 울산, 김해, 청주, 대구, 여수, 양양, 광주, 포항경주, 사천, 원주
발표시각	매 정시
유효시간	3시간
제공기간	2022년 11월 2일 ~ 현재
제공형식	TXT
제공경로	공공데이터포털(https://www.data.go.kr) → 기상청 국내 공항 이륙예보 조회 서비스 검색 API허브(https://apihub.kma.go.kr) → 항공기상 → 공항예·특보
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.

4.항공예보
- 공항예보(TAF, 국내)
- 이륙예보
- SIGWX(중요기상예보)
- SIGMET(위험기상정보)
- AIRMET(저고도 위험 기상정보)
- 공항경보(WARNING)

SIGWX(중요기상예보)

개요

자료상태 : 현행

중요기상 예보(SIGWX)는 항로상에 영향을 미칠 수 있는 기상현상을 중·고고도(FL100~FL600), 저고도(10,000ft 이하)로 각각 나누어 발표합니다.
중·고고도 중요기상예보는 세계공역기상센터에서 발표한 자료이며, 저고도 중요기상예보는 항공기상청에서 발표합니다.

주요요소	모래폭풍, 적란운, 착빙(보통 또는 심한), 난류(보통 또는 심한), 화산분출(화산명 포함) 등	
제공범위	인천 비행정보구역	
발표시각	일 4회 (05, 11, 17, 23UTC)	
유효시간	발표시각 1시간 이후부터 6시간 후(12(+1), 18(+1), 00(+1), 06(+1)UTC) ※ +1은 1일 후(내일)을 의미함	
제공기간	2024년 2월 5일 ~ 현재	
제공형식	TXT, XML	
제공경로	TXT	공공데이터포털(https://www.data.go.kr) → 기상청_저고도 기상정보 조회 서비스 검색
	XML	API허브(https://apihub.kma.go.kr) → 항공기상 → 저고도 기상지원
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.	

4.항공예보
- 공항예보(TAF, 국내)
- 이륙예보
- SIGWX(중요기상예보)
- SIGMET(위험기상정보)
- AIRMET(저고도 위험기상정보)
- 공항경보(WARNING)

SIGMET(위험기상정보)

개요

자료상태 : 현행

SIGMET 정보는 운항하는 항공기에게 위험을 초래할 수 있는 기상 현상과 이러한 현상의 시간적 및 공간적 변화에 대한 정보입니다.

주요요소	천둥번개, 태풍, 심한 난류·착빙·산악파, 먼지·모래 폭풍, 화산재, 방사성 구름	
제공범위	인천 비행정보구역	
발표시각	기상현상 발생이 예상되는 시각으로부터 최대 4시간 전 (화산재 구름 및 태풍과 같은 특별한 경우는 발생 예상시각으로부터 최대 12시간 전)	
유효시간	4시간 이내(화산재 구름 및 태풍과 같은 특별한 경우는 6시간 이내)	
제공기간	2005년 3월 10일 ~ 현재	
제공형식	XML, CSV	
제공경로	CSV	공공데이터포털(https://www.data.go.kr) → 기상청_항공기상청_공역기상특보 검색
	XML	API허브(https://apihub.kma.go.kr) → 항공기상 → 공항예·특보
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.	

AIRMET(저고도 위험기상정보)

개요

자료상태 : 현행

AIRMET 정보는 10,000ft 이하의 저고도를 운항하는 항공기에 위험을 초래할 수 있는 기상 현상과 이러한 현상의 시간적 및 공간적 변화에 대한 정보입니다.

주요요소	지상풍속, 지상시정, 천둥번개, 산악차폐, 구름(적란운 또는 탑상적운), 보통 착빙, 보통 난류, 보통 산악파
제공범위	인천 비행정보구역
발표시각	기상현상 발생이 예상되는 시각으로부터 최대 4시간 전
유효시간	4시간 이내
제공기간	2007년 11월 8일 ~ 현재
제공형식	XML
제공경로	API허브(https://apihub.kma.go.kr) → 항공기상 → 공항예·특보
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.

4.항공예보
- 공항예보(TAF, 국내)
- 이륙예보
- SIGWX(중요기상예보)
- SIGMET(위험기상정보)
- AIRMET(저고도 위험 기상정보)
- 공항경보(WARNING)

공항경보(WARNING)

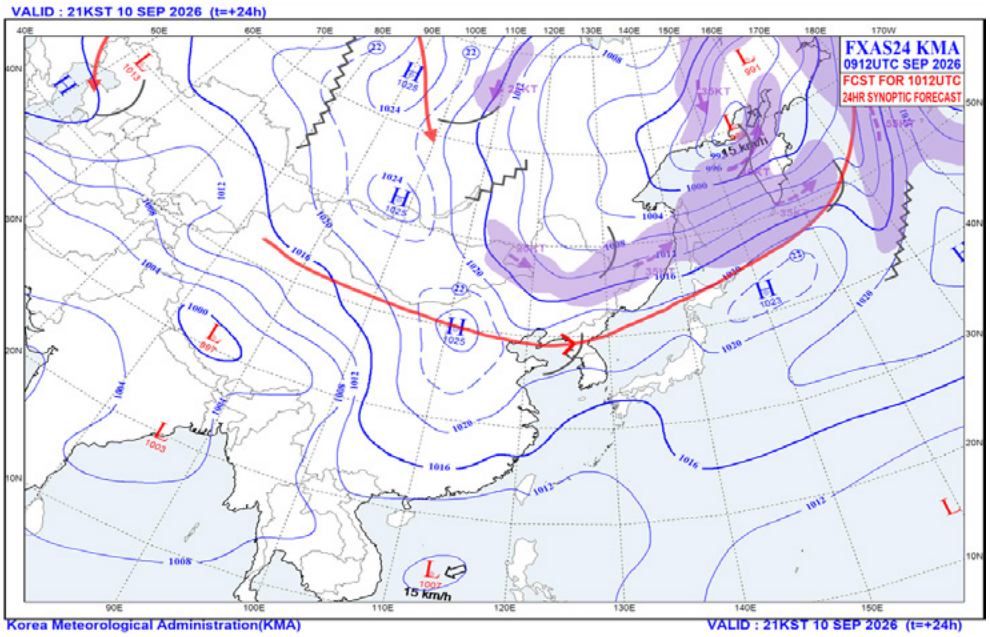
개요

자료상태 : 현행

공항시설과 지상 항공기 등에 영향을 줄 수 있는 기상현상에 대해 발표하는 공항경보입니다. 공항별 발표·수정·해제 정보와 유효기간을 확인하여 이용합니다.

주요요소	태풍, 천둥번개, 대설, 강풍, 구름고도, 저시정, 호우, 황사, 우박, 어는강수, 서리, 지진해일 등
제공범위	인천, 김포, 제주, 무안, 울산, 김해, 청주, 대구, 여수, 양양, 광주, 포항경주, 사천, 원주, 군산
발표시각	해당 공항의 기상현상이 특보발표 기준치에 해당 될 것으로 관측 및 예상시 발표
유효시간	해당 공항에 특보발표 기준치의 기상현상의 시작 예상시간을 유효 시작시간, 기상현상이 끝날 것으로 예상되는 시간을 유효 끝시간으로 설정 ※ 공항경보 발효시간 종료 후에도 특보기준에 해당하는 기상현상이 지속될 것으로 예상 시 연장발표, 더 이상 발생하지 않거나 발생하지 않을 것으로 예상될 시 해제발표)
제공기간	2007년 11월 17일 ~ 현재
제공형식	XML
제공경로	API허브(https://apihub.kma.go.kr) → 항공기상 → 공항예·특보
이용방법	공항 ICAO 코드 또는 비행정보구역과 발표·유효시각을 선택합니다. 전문의 UTC 시각, 항공기상 부호와 단위를 확인하고 변경·수정·취소 전문을 적용합니다.

편집일기도(지상 24시간 예상)



개요

자료상태 : 현행

편집일기도(지상 24시간 예상)는 기압계와 기상요소의 분포를 지도에 표시한 일기도 자료입니다. 분석·예상 여부, 지상·상층 구분과 유효시각을 확인하여 이용하며, 그림의 범례와 단위를 적용하여 해석합니다.

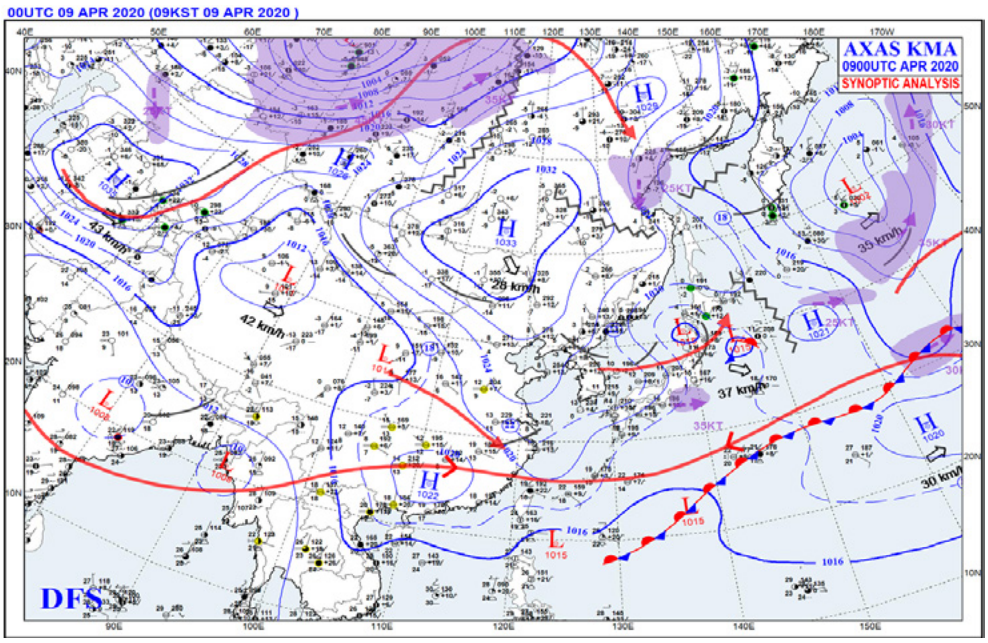
주요요소	지상 24시간 예상, 편집일기도
제공범위	아시아
발표시각	일 2회(00UTC, 12UTC)
제공기간	2004년 1월 1일 09:00 ~ 현재 ※ 12시간 지상일기도는 2014. 4. 30. 부터 '편집일기도'로 제공
생산·갱신주기	12시간
제공형식	PNG
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	일기도 종류·영역·고도·기상요소·시각을 선택해 이미지를 조회·다운로드합니다. 분석·예상 여부, 시간 기준과 범례를 확인하며, 정량 계산에는 별도의 수치자료를 이용합니다.

5.편집일기도
- 편집일기도 (지상 24시간 예상)
- 편집일기도(지상 12시간)
- (생산종료)편집일기도 (지상 3시간)
- (생산종료)편집일기도 (고층)
- (생산종료)편집일기도 (해상 24시간 예상)

5.편집일기도

- 편집일기도 (지상 24시간 예상)
- 편집일기도(지상 12시간)
- (생산종료)편집일기도 (지상 3시간)
- (생산종료)편집일기도 (고층)
- (생산종료)편집일기도 (해상 24시간 예상)

편집일기도(지상 12시간)



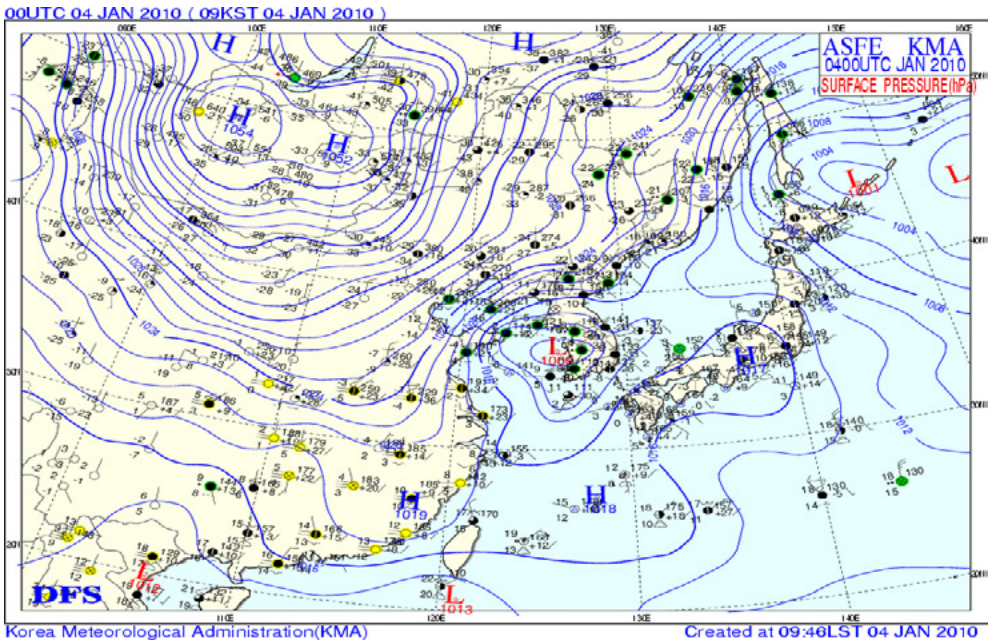
개요

자료상태 : 현행

편집일기도(지상 12시간)는 기압계와 기상요소의 분포를 지도에 표시한 일기도 자료입니다. 분석·예상 여부, 지상·상층 구분과 유효시각을 확인하여 이용하며, 그림의 범례와 단위를 적용하여 해석합니다.

종 류	지상 12시간, 편집일기도
제공범위	아시아
발표시각	일 2회(00UTC, 12UTC)
제공기간	2014년 4월 3일 ~ 현재
생산·갱신주기	12시간
제공형식	PNG
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	일기도 종류·영역·고도·기상요소·시각을 선택해 이미지를 조회·다운로드합니다. 분석·예상 여부, 시간 기준과 범례를 확인하며, 정량 계산에는 별도의 수치자료를 이용합니다.

(생산종료)편집일기도(지상 3시간)



개요

자료상태 : 생산종료

3시간 간격의 지상 기상요소(등압선, 지상기온, 바람, 운량, 현재날씨 등)를 한 장의 일기도에 표현한 자료입니다.

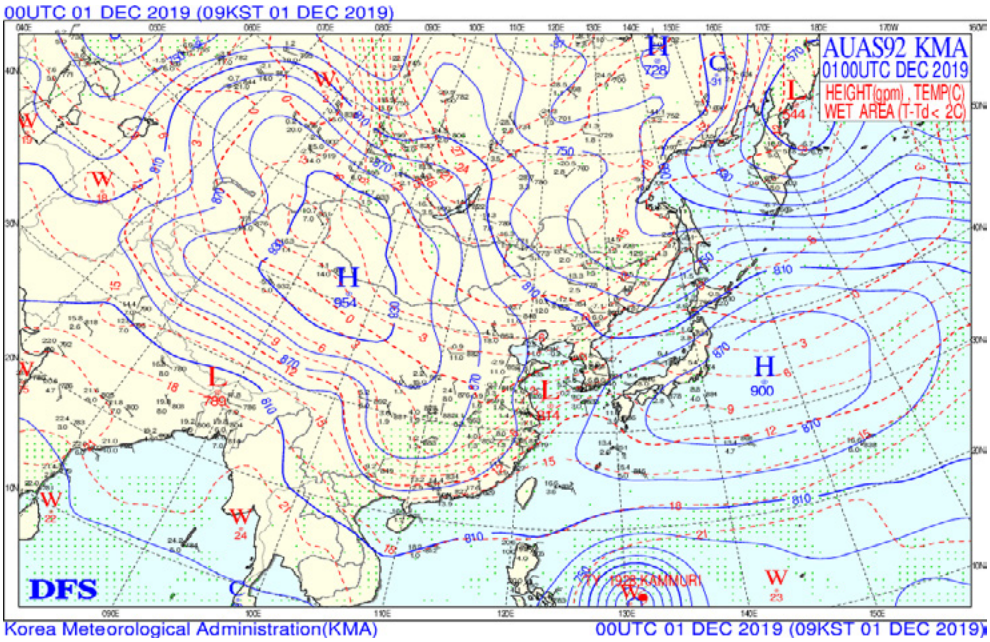
종 류	지상 3시간, 일기도
제공범위	동아시아
발표시각	일 8회(00, 03, 06, 09, 12, 15, 18, 21UTC)
제공기간	2004년 2월 12일 ~ 2013년 3월 5일 ※ 중단시점 이후 수치 분석일기도로 제공(파일명 rename 후 별도 저장 중) - (수치) kim_sfc3_{type1}_{type2}_{yyyymmddhh}.png → (rename) sfc3_{yyyymmddhh}.png
생산·갱신주기	3시간
제공형식	png
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	일기도 종류·영역·고도·기상요소·시각을 선택해 이미지를 조회·다운로드합니다. 분석·예상 여부, 시간 기준과 범례를 확인하며, 정량 계산에는 별도의 수치자료를 이용합니다.

5.편집일기도
- 편집일기도 (지상 24시간 예상)
- 편집일기도(지상 12시간)
- (생산종료)편집일기도 (지상 3시간)
- (생산종료)편집일기도 (고층)
- (생산종료)편집일기도 (해상 24시간 예상)

5.편집일기도

- 편집일기도 (지상 24시간 예상)
- 편집일기도(지상 12시간)
- (생산종료)편집일기도 (지상 3시간)
- **(생산종료)편집일기도 (고층)**
- (생산종료)편집일기도 (해상 24시간 예상)

(생산종료)편집일기도(고층)



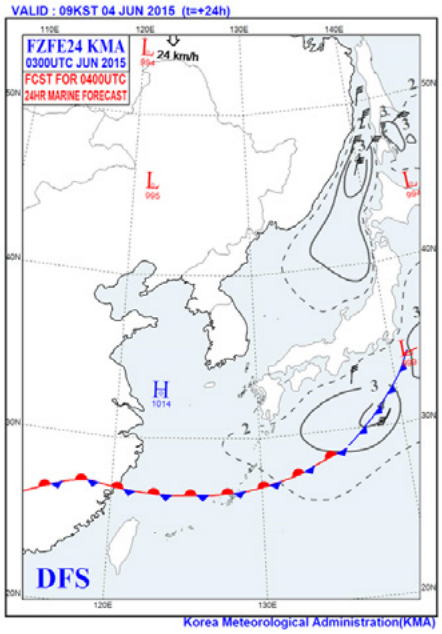
개요

자료상태 : 생산종료

대기 상층의 기압, 온도, 바람, 습도 등에 대한 자료를 표현한 자료입니다.

종 류	고층일기도
제공범위	동아시아
발표시각	일 2회(00, 12UTC)
제공기간	2004년 1월 1일 ~ 2013년 3월 5일 ※ 중단시점 이후 수치 분석일기도로 제공(파일명 rename 후 별도 저장 중) - (수치) kim_up{lev}_{type1}_{type2}_{yyyymmddhh}.png → (rename) up??_yyyymmddhh.png
생산·갱신 주기	12시간
제공형식	PNG
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	일기도 종류·영역·고도·기상요소·시각을 선택해 이미지를 조회·다운로드합니다. 분석·예상 여부, 시간 기준과 범례를 확인하며, 정량 계산에는 별도의 수치자료를 이용합니다.

편집일기도(해상 24시간 예상)



개요

자료상태 : 생산종료

24시간 예상: 24시간 이후 우리나라 주변 해상의 기상정보(기압배치, 전선, 해무 등)를 표현한 자료입니다.

종 류	해당 24시간 예상, 일기도
제공범위	한반도
발표시각	일 2회(00, 12UTC)
제공기간	해상 24시간 예상: 2013년 12월 4일 ~ 2016년 3월 1일
생산·갱신 주기	12시간
제공형식	png
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청 자료 보유·제공 여부를 검토하는 신청 절차로, 승인·제공 안내에 따라 이용합니다.
이용방법	일기도 종류·영역·고도·기상요소·시각을 선택해 이미지를 조회·다운로드합니다. 분석·예상 여부, 시간 기준과 범례를 확인하며, 정량 계산에는 별도의 수치자료를 이용합니다.

5.편집일기도
- 편집일기도 (지상 24시간 예상)
- 편집일기도(지상 12시간)
- (생산종료)편집일기도 (지상 3시간)
- (생산종료)편집일기도 (고층)
- (생산종료)편집일기도 (해상 24시간 예상)

06

수치모델

Numerical Weather Prediction Model



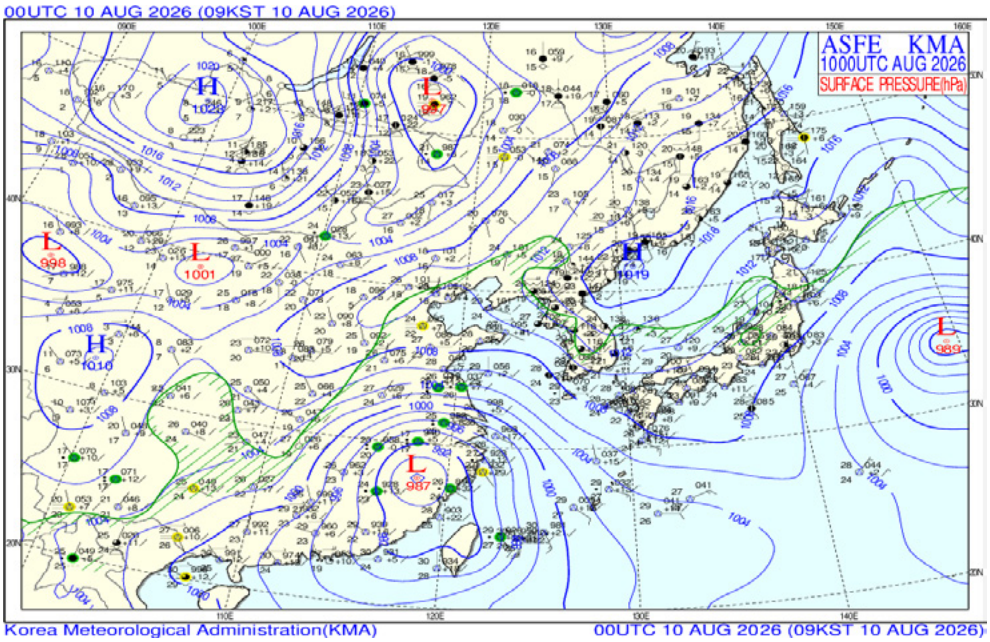
1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)

분석일기도(단열선도)

- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

분석일기도(3시간, 지상)



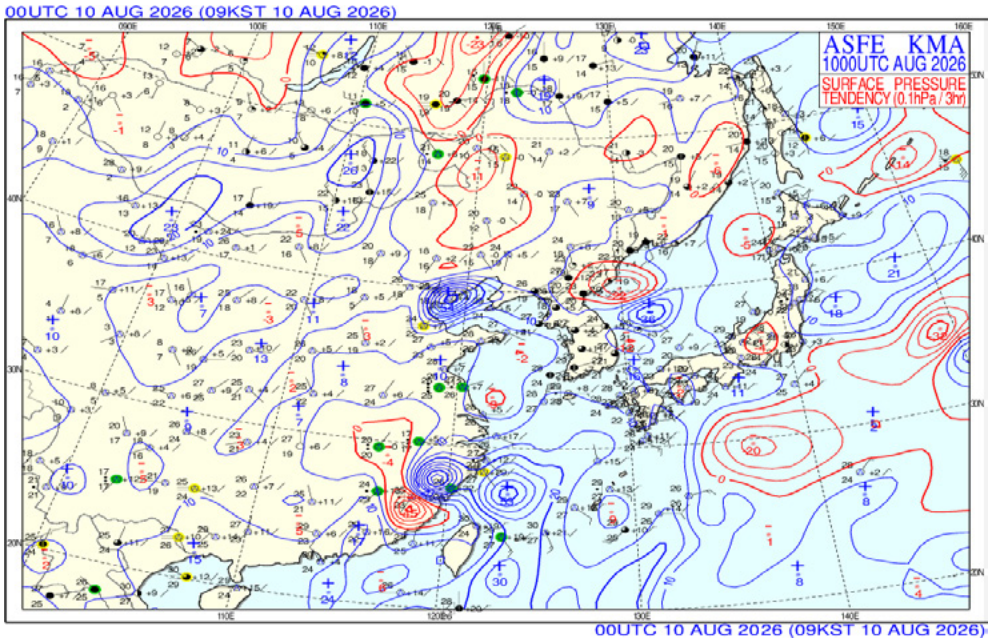
개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(3시간, 지상)는 3시간 간격의 지상일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기온, 기압 등
종 류	3시간 지상일기도
발표시각	일 8회(00UTC 기준 3시간마다)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도
	공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

분석일기도(3시간, 지상기압변화)



개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(3시간, 지상기압변화)는 3시간 간격의 지상기압변화일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기압변화
종 류	3시간 지상 기압변화일기도
발표시각	일 8회(00UTC 기준 3시간마다)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

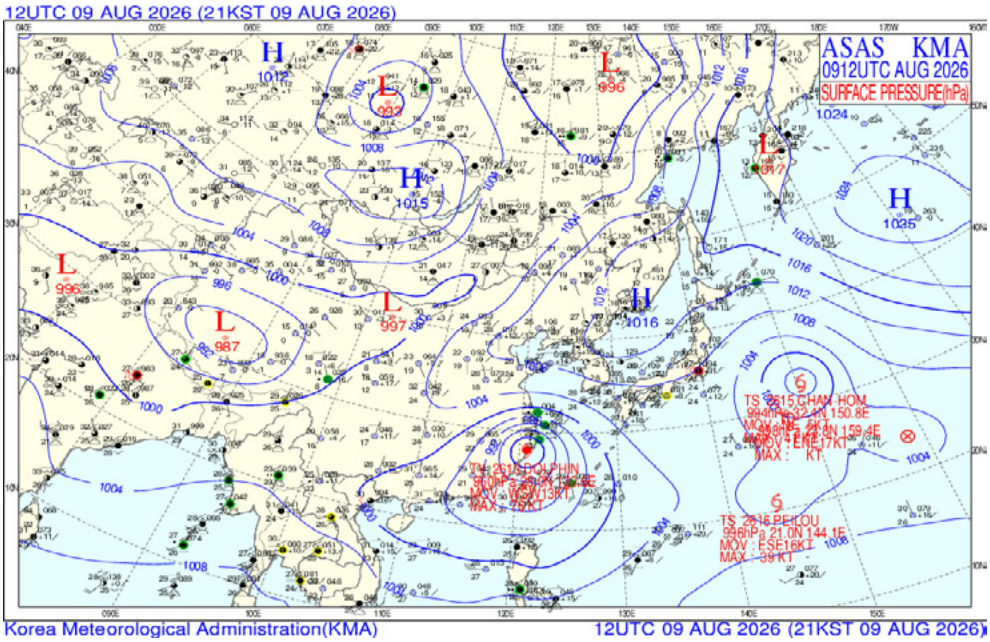
1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

분석일기도(6시간, 지상)



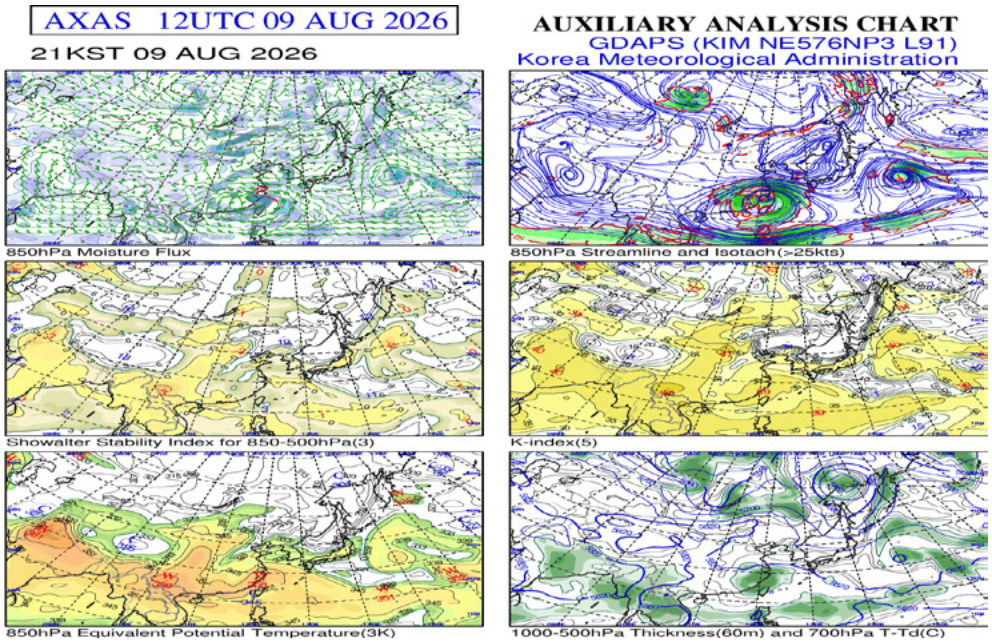
개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(6시간, 지상)는 6시간 간격의 지상일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기압
종 류	6시간 지상일기도
발표시각	일 4회(00, 06, 12, 18UTC)
제공기간	2004년 1월 ~ 현재
제공형식	GIF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

분석일기도(보조분석도)



자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(보조분석도)는 6시간 간격의 유선도, 습수도 등이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	바람, 습도 등
종 류	보조분석도
발표시각	일 4회(00, 06, 12, 18UTC)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 수치모델 → 분석일기도 → 보조일기도
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

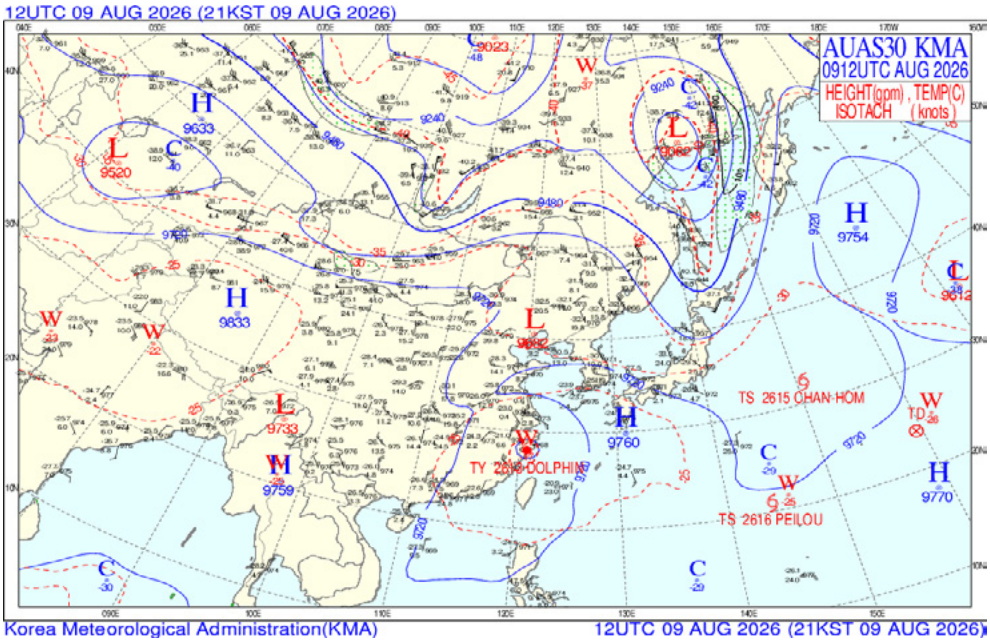
1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- **분석일기도(6시간, 고층)**
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

분석일기도(6시간, 고층)



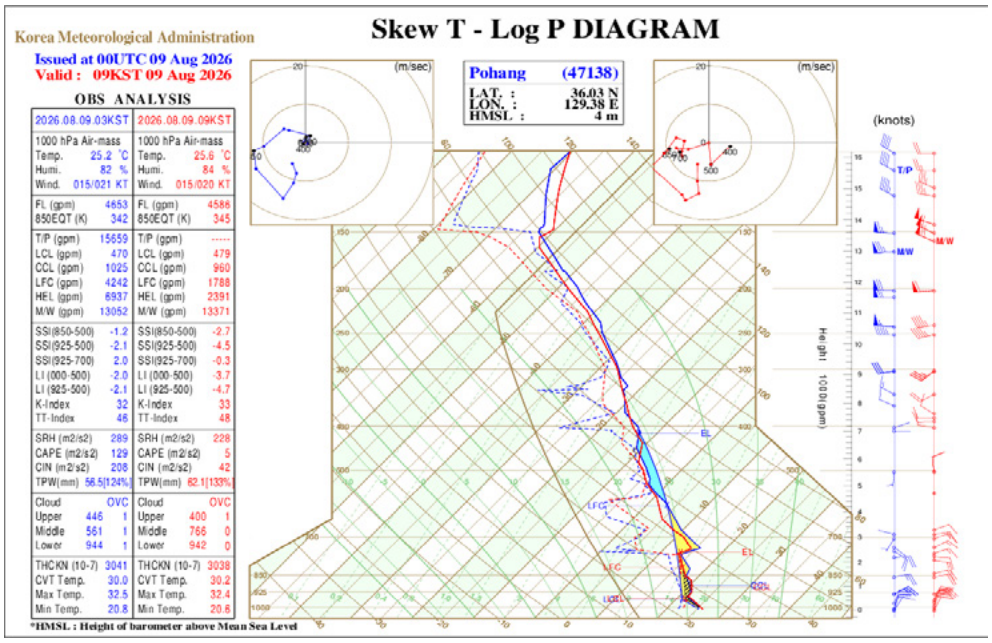
개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(6시간, 고층)는 6시간 간격의 고층일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기온, 기압
종 류	6시간 고층일기도
발표시각	일 4회(00, 06, 12, 18UTC)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도
	공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

분석일기도(단열선도)



1.수치분석 일기도
- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

자료상태 : 현행

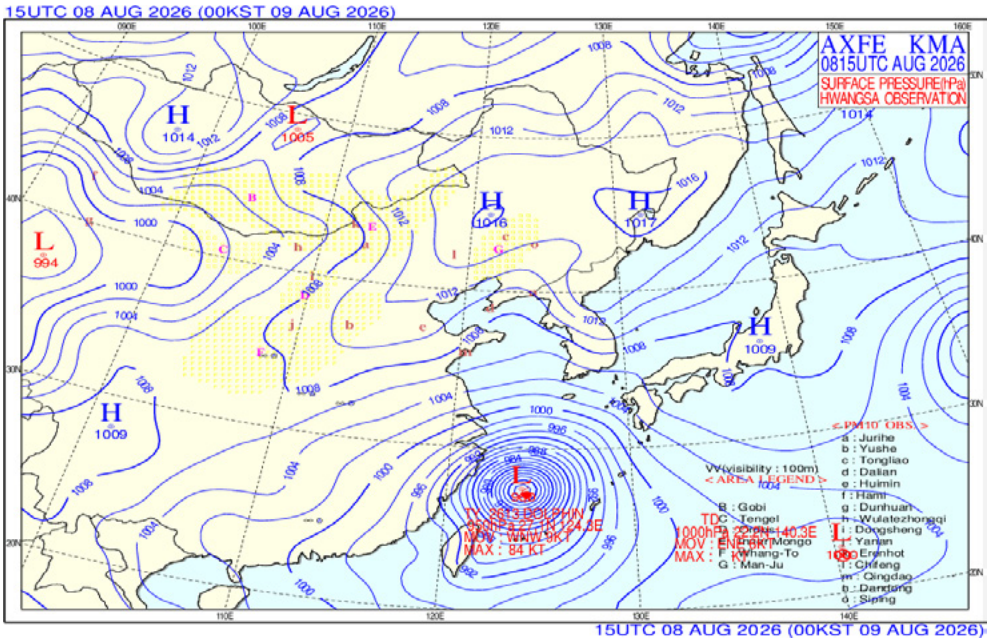
수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(단열선도)는 6시간 간격의 고층자료로 그린 단열선도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기온, 이슬점온도 등
종 류	단열선도
발표시각	일 4회(00, 06, 12, 18UTC)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- **분석일기도(3시간, 황사)**
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

분석일기도(3시간, 황사)



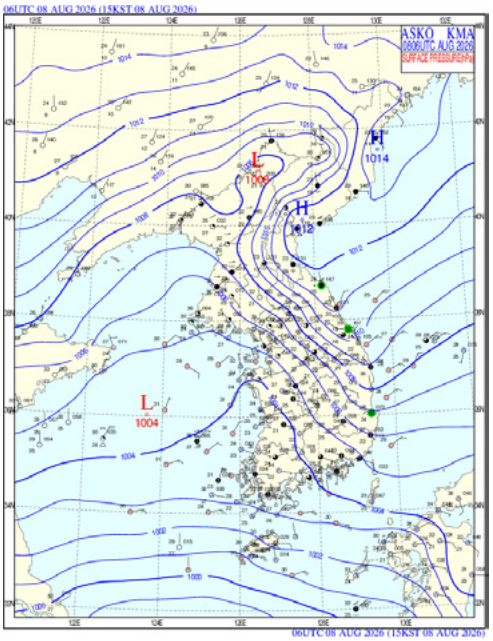
개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(3시간, 황사)는 3시간 간격의 고층일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	황사, 기압
종 류	3시간 황사일기도
발표시각	일 4회(00, 06, 12, 18UTC)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

분석일기도(한반도 일기도)



1.수치분석 일기도
- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

자료상태 : 현행

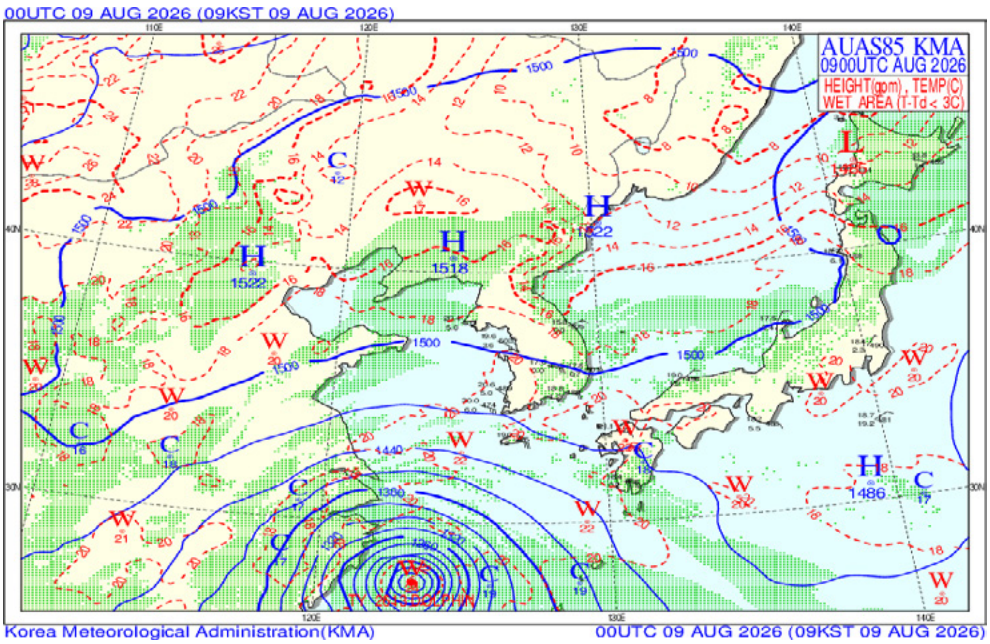
수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(한반도 일기도)는 1시간 간격의 한반도 주변에 대한 일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기압, 바람, 습도 등
종 류	한반도 일기도
발표시각	일 24회(매시간)
제공기간	2004년 1월 1일 ~ 현재
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

1.수치분석 일기도

- 분석일기도(3시간, 지상)
- 분석일기도(3시간, 지상기압 변화)
- 분석일기도(6시간, 지상)
- 분석일기도(보조분석도)
- 분석일기도(6시간, 고층)
- 분석일기도(단열선도)
- 분석일기도(3시간, 황사)
- 분석일기도(한반도 일기도)
- 분석일기도(상세실황분석장)

분석일기도(상세실황분석장)



개요

자료상태 : 현행

수치분석일기도는 수치예보시스템에서 생산된 수치자료 및 관측자료를 이용하여 기상 변수들에 대한 기호와 등치선 등으로 이루어진 일기도를 뜻합니다. 분석일기도(상세실황분석장)는 1시간 간격의 분석일기도이며, 일기예보를 위하여 예보관 및 사용자들에게 제공됩니다.

주요요소	기압, 기온, 바람, 습도 등
종 류	상세실황분석장
발표시각	일 24회(매시간)
제공기간	2024년 5월 14일 ~ 현재
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

전지구예보모델(KIM-GDAPS) 예측자료

2.단·중기 모델
• 전지구예보모델 예측자료
• 전지구예보모델 수치예상일기도
• 국지예보모델 예측자료
• 국지예보모델 수치예상일기도
• 지역예보모델 예측자료
• 지역예보모델 수치예상일기도
• 전지구예보모델 예측자료
• 전지구예보모델 수치예상일기도
• 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
• 국지예보모델 예측자료
• 국지예보모델 수치예상일기도
• 지역예보모델 예측자료
• 지역예보모델 수치예상일기도
• 지역예보모델 예측자료
• 지역예보모델 수치예상일기도

개요

자료상태 : 현행

한국형수치예보모델(KIM, Korean Integrated Model)은 우리나라의 기상현상 예측정확도를 향상시키고, 기술자립도를 높이기 위해 2011년부터 개발하여 2019년 12월에 개발 완료되었으며, 2020년 4월부터 기상청의 현업모델로 사용되고 있습니다. 수평해상도는 약 8km이며, 연직층수는 91층, 모델의 최상층은 0.01hPa(~80km)입니다. 일 4회(00, 06, 12, 18UTC) 수행되며, 00UTC와 12UTC는 288시간까지 예측자료를 생산하고, 06UTC와 18UTC에는 87시간까지 예측자료를 생산하고 있습니다.

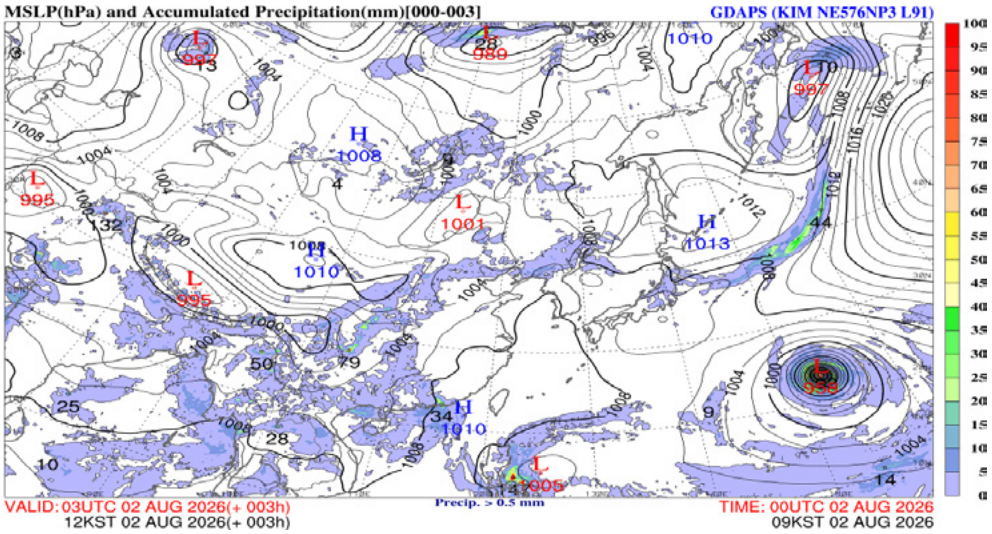
현재 기상청에서는 KIM-GDAPS 예측자료를 GRIB2 및 NetCDF4 형식으로 생산·저장, 제공하고 있으며 2026년 9월 GRIB2 형식 자료는 서비스를 종료하고 NetCDF4 형식(표준화 자료)으로 일원화하여 제공됩니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 12km/8km(2025년 5월 14일 ▮ 현재) · 연직층수 : 91층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 87시간 예측(06, 18UTC 발표) · 288시간 예측(00, 12UTC 발표)
제공기간	(GRIB) 2020년 4월 28일 ▮ 2026년 9월 30일 (NetCDF) 2026년 3월 17일 ▮ 현재 (과거자료(2025년 5월 14일 이후) 표준화 변환 진행중)
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB2, NetCDF4(표준화 자료)
제공경로	기상청 API허브(https://apihub.kma.go.kr/) → 수치모델 → 수치예보모델 공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	(GRIB) 1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다. (NetCDF) 1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 -h 옵션으로 헤더 정보를 -v 옵션으로 자료를 읽을 수 있습니다.

2.단·중기 모델

- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

전지구예보모델(KIM-GDAPS) 수치예상일기도



개요

자료상태 : 현행

한국형수치예보모델(KIM, Korean Integrated Model)은 우리나라의 기상현상 예측정확도를 향상시키고, 기술자립도를 높이기 위해 2011년부터 개발하여 2019년 12월에 개발 완료되었으며, 2020년 4월부터 기상청의 현업모델로 사용되고 있습니다. 수평해상도는 약 8km이며, 연직층수는 91층, 모델의 최상층은 0.01hPa(~80km)입니다. 일 4회(00, 06, 12, 18UTC) 수행되며, 00UTC와 12UTC는 288시간까지 예측자료를 생산하고, 06UTC와 18UTC에는 87시간까지 예측자료를 생산하고 있습니다.
현재 기상청에서는 KIM-GDAPS 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면 요소
해 상 도	· 수평분해능 : 12km/8km · 연직층수 : 91층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 87시간 예측(06, 18UTC 발표) · 288시간 예측(00, 12UTC 발표)
제공기간	2020년 4월 28일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	PNG, GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

국지예보모델(KIM-LDAPS) 예측자료

개요

자료상태 : 현행

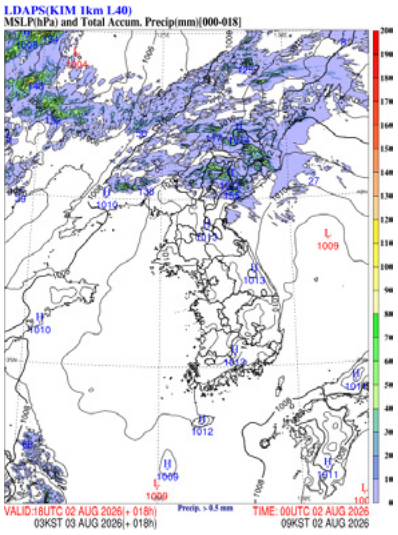
한국형모델 기반 국지예보시스템(LDAPS, Local Data Assimilation and Prediction System)은 2025년 11월부터 현업 운영을 시작하였습니다. 한반도 영역에 대해 일 4회 2일(48시간) 예측을 수행하며, 수평해상도는 1km이며 예측간격은 1시간입니다.
현재 기상청에서는 KIM-LDAPS 예측자료를 GRIB2 및 NetCDF4 형식으로 생산·저장, 제공하고 있으며 2026년 9월 GRIB2 형식 자료는 서비스를 종료하고 NetCDF4 형식(표준화 자료)으로 일원화하여 제공됩니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 1km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	48시간 예측
제공기간	(GRIB) 2025년 11월 14일 ~ 2026년 9월 30일 (NetCDF) 2026년 2월 9일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB2, NetCDF4(표준화 자료)
제공경로	기상청 API허브(https://apihub.kma.go.kr/) → 수치모델 → 수치예보모델 공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	(GRIB) 1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용 매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다. (NetCDF) 1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

2.단·중기 모델

- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

국지예보모델(KIM-LDAPS) 수치예상일기도



개요

자료상태 : 현행

한국형모델 기반 국지예보시스템(LDAPS, Local Data Assimilation and Prediction System)은 2025년 11월부터 현업 운영을 시작하였습니다. 한반도 영역에 대해 일 4회 2일(48시간) 예측을 수행하며, 수평해상도는 1km이며 예측간격은 1시간입니다.
현재 기상청에서는 KIM-LDAPS 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면 요소
해 상 도	· 수평분해능 : 1km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	48시간 예측
제공기간	2025년 11월 14일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	PNG, GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

지역예보모델(KIM-RDAPS) 예측자료

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

개요

자료상태 : 현행

한국형모델 기반 지역예보시스템(RDAPS, Regional Data Assimilation and Prediction System)으로 현업 운영 중인 한국형 지역예보시스템(KIM-Regional)은 2022년 5월 12일부터 현업 운영을 시작하였습니다. 동아시아 영역에 대해 수평해상도는 3km이다. 예측기간은 00과 12UTC는 5일, 06과 18UTC는 3일이며 예측간격은 1시간입니다.

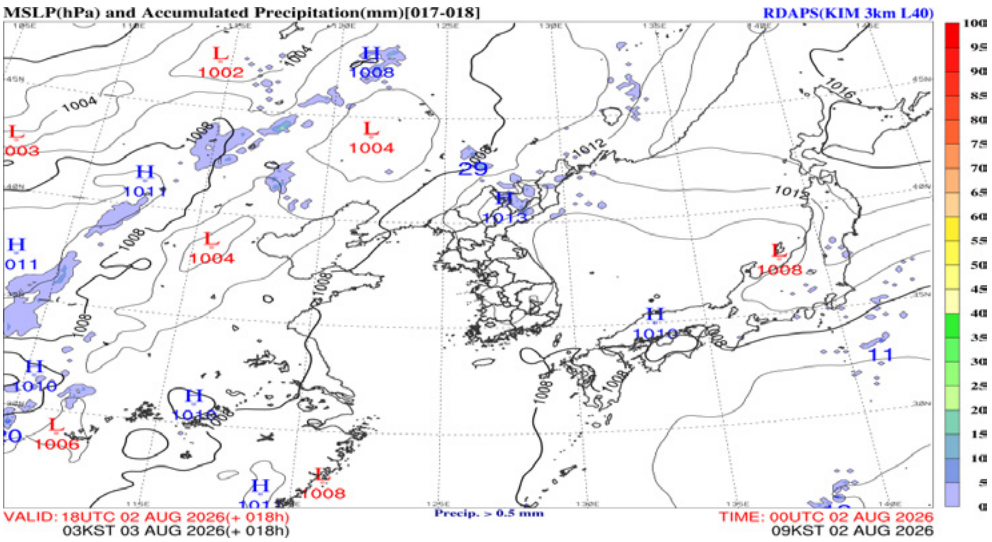
현재 기상청에서는 KIM-RDAPS 예측자료를 GRIB2 및 NetCDF4 형식으로 생산·저장, 제공하고 있으며 2026년 9월 GRIB2 형식 자료는 서비스를 종료하고 NetCDF4 형식(표준화 자료)으로 일원화하여 제공됩니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 3km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 72시간 예측(06, 18UTC 발표) · 120시간 예측(00, 12UTC 발표)
제공기간	(GRIB) 2022년 5월 12일 ▮ 2026년 9월 30일 (NetCDF) 2026년 2월 9일 ▮ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB2, NetCDF4(표준화 자료)
제공경로	기상청 API허브(https://apihub.kma.go.kr/) → 수치모델 → 수치예보모델 공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	(GRIB) 1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용 매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다. (NetCDF) 1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

2.단·중기 모델

- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

지역예보모델(KIM-RDAPS) 수치예상일기도



개요

자료상태 : 현행

한국형모델 기반 지역예보시스템(RDAPS, Regional Data Assimilation and Prediction System)으로 현업 운영 중인 한국형 지역예보시스템(KIM-Regional)은 2022년 5월 12일부터 현업 운영을 시작하였다. 동아시아 영역에 대해 수평해상도는 3km이다. 예측기간은 00과 12UTC는 5일, 06과 18UTC는 3일이며 예측간격은 1시간입니다.
현재 기상청에서는 KIM-RDAPS 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 3km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 72시간 예측(06, 18UTC 발표) · 120시간 예측(00, 12UTC 발표)
제공기간	2022년 5월 12일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	PNG, GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

(생산종료)전지구예보모델 (UM-GDAPS) 예측자료

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

개요

자료상태 : 생산종료

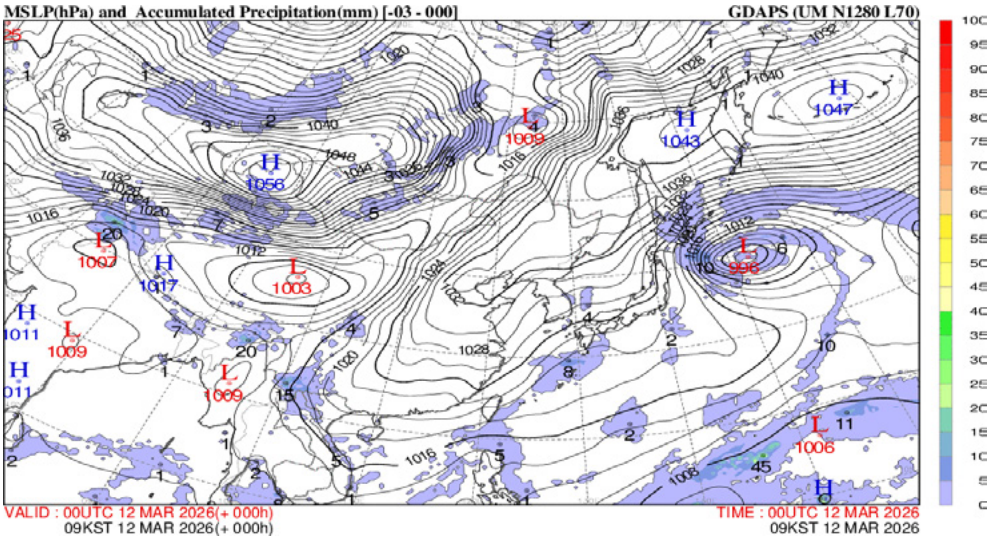
전지구 예보모델(GDAPS, Global Data Assimilation and Prediction System)은 전지구 날씨 예측, 단기예보, 중기예보 등을 목적으로 영국 통합모델(UM)을 기반 구축된 수치예보시스템입니다.
6시간 주기 순환예측에 필요한 각종 배경장을 생산하기 위한 일 4회 예측을 수행하며, 일 1회 06UTC에는 해수면온도, 해빙자료 및 동서평균 오존량을 갱신하기 위한 배경장 갱신과정을 별도로 수행합니다.
현재 UM 모델 종료로 UM-GDAPS 예측자료는 미생산 중입니다.

주요요소	등압면, 단일면
해상도	· 수평분해능 : 10km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 87시간 예측(06, 18UTC 발표) · 288시간 예측(00, 12UTC 발표)
제공기간	2011년 5월 23일 ~ 2026년 3월 31일
생산·갱신주기	6시간 주기 자료
제공형식	GRIB2
제공경로	기상청 API허브(https://apihub.kma.go.kr/) → 수치모델 → 수치예보모델 공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델 데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl 라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

2.단·중기 모델

- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- **전지구예보모델 수치예상일기도**
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

(생산종료)전지구예보모델 (UM-GDAPS) 수치예상일기도



개요

자료상태 : 생산종료

전지구 예보모델(GDAPS, Global Data Assimilation and Prediction System)은 전지구 날씨 예측, 단기예보, 중기예보 등을 목적으로 영국 통합모델(UM)을 기반 구축된 수치예보시스템입니다.
6시간 주기 순환예측에 필요한 각종 배경장을 생산하기 위한 일 4회의 예측을 수행하며, 일 1회 06UTC에는 해수면온도, 해빙자료 및 동서평균 오존량을 갱신하기 위한 배경장 갱신과정을 별도로 수행합니다.
현재 UM 모델 종료로 UM-GDAPS 예측자료 이미지는 미생산 중입니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 10km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	· 87시간 예측(06, 18UTC 발표) · 288시간 예측(00, 12UTC 발표)
제공기간	2011년 5월 24일 ~ 2026년 3월 31일
생산·갱신 주기	6시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

(생산종료)전지구예보모델(UM-GDAPS) 지역영역 예측자료((구)RDAPS 영역)

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

개요

자료상태 : 생산종료

전지구 예보모델(GDAPS, Global Data Assimilation and Prediction System)은 전지구 날씨 예측, 단기예보, 중기예보 등을 목적으로 영국 통합모델(UM)을 기반 구축된 수치예보시스템입니다.
6시간 주기 순환예측에 필요한 각종 배경장을 생산하기 위한 일 4회의 15시간 예측을 수행하며, 일 1회 06UTC에는 해수면온도, 해빙자료 및 동서평균 오존량을 갱신하기 위한 배경장 갱신과정을 별도로 수행합니다. 현재 기상청에서는 GDAPS 예측자료에서 동아시아 영역의 자료만 추출하여 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	등압면, 단일면, 모델면
해 상 도	• 수평: 12km • 연직: 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	87시간 예측
제공기간	2010년 3월 ~ 2026년 3월 31일
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB2
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치예보 → 단·중기예측 → 지역예보모델(RDAPS)
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델 데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl 라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

(생산종료)국지예보모델 (UM-LDAPS) 예측자료

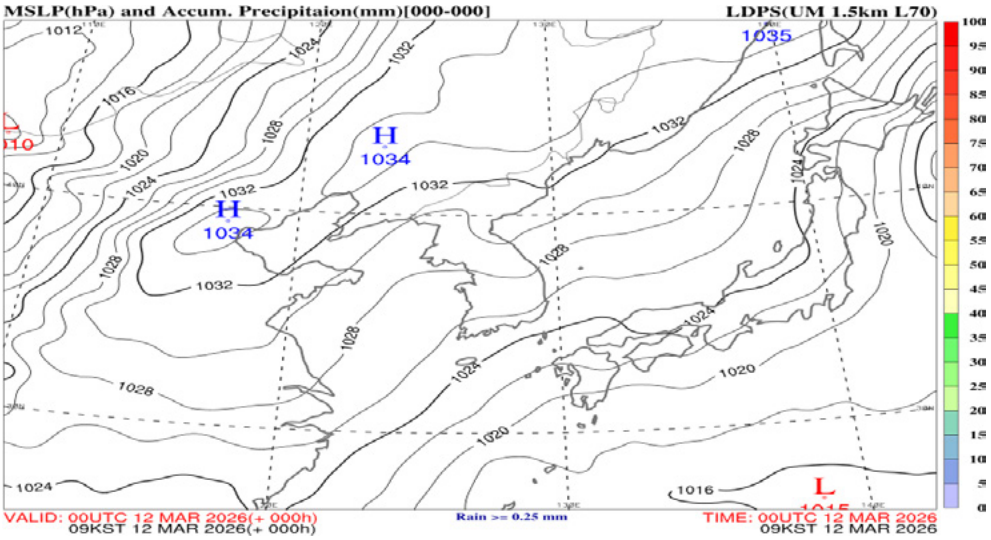
개요

자료상태 : 생산종료

국지예보모델(LDAPS, Local Data Assimilation and Prediction System)은 3시간 간격으로 전지구 예보모델로부터 경계장을 제공받아 1일 4회 예측을 수행하는 수치예보 시스템입니다.
3차원 변분자료 동화 기법을 이용하여 각각의 자체 분석-예측 순환 체계로 운영되고 있으며, 주로 한반도 날씨 예측에 활용됩니다. 현재 UM 모델 종료로 UM-LDAPS 예측자료는 미생산 중입니다.

주요요소	등압면, 단일면, 모델면
해 상 도	· 수평분해능 : 1.5 ~ 4km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	48시간 예측
제공기간	2012년 5월 15일 ~ 2026년 3월 31일
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB2
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치예보 → 단·중기예측 → 국지예보모델(LDAPS)
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델 데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl 라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

(생산종료)국지예보모델 (UM-LDAPS) 수치예상일기도



개요

자료상태 : 생산종료

국지예보모델(LDAPS, Local Data Assimilation and Prediction System)은 3시간 간격으로 전지구 예보모델로부터 경계장을 제공받아 1일 4회 예측을 수행하는 수치예보 시스템입니다.
3차원 변분자료 동화 기법을 이용하여 각각의 자체 분석-예측 순환 체계로 운영되고 있으며, 주로 한반도 날씨 예측에 활용됩니다.
현재 UM 모델 종료로 UM-LDAPS 예측자료 이미지는 미생산 중입니다.

주요요소	등압면, 단일면, 모델면
해상도	· 수평분해능 : 1.5 ~ 4km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	48시간 예측
제공기간	2015년 12월 1일 ~ 2026년 3월 31일
생산·갱신주기	6시간 주기 자료
제공형식	GIF, PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델 자료 활용 가이던스 참조

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

2.단·중기 모델

- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- **지역예보모델 예측자료**
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

(생산종료)지역예보모델 (UM-RDAPS) 예측자료

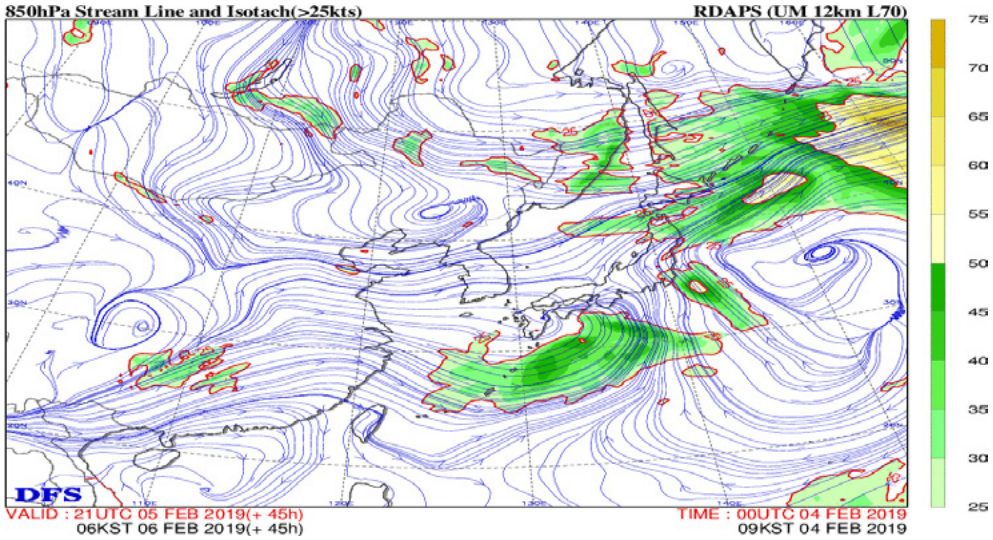
개요

자료상태 : 생산종료

지역예보모델(RDAPS, Regional Data Assimilation and Prediction System)은 6시간 간격으로 전지구 예보모델로부터 경계장을 제공받아 1일 4회 예측을 수행하는 수치예보 시스템입니다.
4차원 변분자료 동화 기법을 이용하여 각각의 자체 분석-예측 순환 체계로 운영되고 있으며, 주로 한반도 날씨 예측에 활용됩니다. 현재 UM-RDAPS 모델 종료로 예측자료는 미생산 중입니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 12km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	87시간 예측
제공기간	2016년 6월 29일 ~ 2019년 3월 31일
생산·갱신주기	6시간 주기 자료
제공형식	GRIB2
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

(생산종료)지역예보모델 (UM-RDAPS) 수치예상일기도



개요

자료상태 : 생산종료

지역예보모델(RDAPS, Regional Data Assimilation and Prediction System)은 6시간 간격으로 전지구 예보모델로부터 경계장을 제공받아 1일 4회 예측을 수행하는 수치예보 시스템입니다.
4차원 변분자료 동화 기법을 이용하여 각각의 자체 분석-예측 순환 체계로 운영되고 있으며, 주로 한반도 날씨 예측에 활용됩니다. 현재 UM-RDAPS 모델 종료로 예측자료를 활용한 일기도는 미생산 중입니다.

주요요소	등압면, 단일면
해상도	· 수평분해능 : 12km · 연직층수 : 70층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	87시간 예측
제공기간	2015년 12월 1일 ~ 2019년 2월 10일
생산·갱신주기	6시간 주기 자료
제공형식	GRIB2
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델 자료 활용 가이던스 참조

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

(생산종료)지역예보모델(KWRF) 예측자료

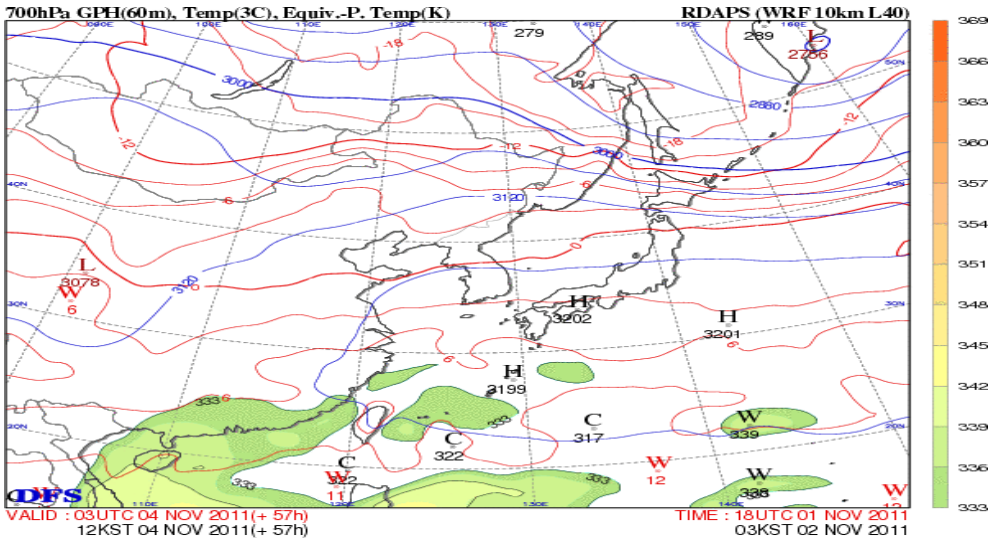
개요

자료상태 : 생산종료

지역예보모델(KWRF, Korea Weather Research and Forecasting)은 지역 날씨 예측, 동아시아 영역 단기예보 등을 목적으로 WRF 모델을 기반으로 구축된 수치예보시스템입니다. 12시간 주기 순환으로 일 2회의 60시간 예측을 수행합니다. 현재 KWRF 모델 종료로 예측자료는 미생산 중입니다.

주요요소	고도, 기온 등
해상도	· 수평분해능 : 10km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	72시간
제공기간	2011년 4월 1일 ~ 2015년 2월 5일
생산·갱신주기	6시간 주기 자료
제공형식	GRIB
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

(생산종료)지역예보모델(KWRF) 수치예상일기도



개요

자료상태 : 생산종료

지역예보모델(KWRF, Korea Weather Research and Forecasting)은 지역 날씨 예측, 동아시아 영역 단기예보 등을 목적으로 WRF 모델을 기반으로 구축된 수치예보시스템입니다. 12시간 주기 순환으로 일 2회의 60시간 예측을 수행합니다. 현재 KWRF 모델 종료로 예측자료를 활용한 이미지는 미생산 중입니다.

주요요소	고도, 기온 등
해 상 도	· 수평분해능 : 10km · 연직층수 : 40층
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	72시간
제공기간	2010년 5월 15일 ~ 2011년 11월 1일
생산·갱신 주기	6시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델 자료 활용 가이던스 참조

2.단·중기 모델
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 전지구예보모델 예측자료
- 전지구예보모델 수치예상일기도
- 전지구예보모델 지역영역 예측자료 ((구)RDAPS 영역)
- 국지예보모델 예측자료
- 국지예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도
- 지역예보모델 예측자료
- 지역예보모델 수치예상일기도

3.초단기모델

- 초단기모델(KLAPS) 분석(KL05) 자료
- 초단기모델(KLAPS) 분석(KL05) 수치분석일기도
- 초단기모델(KLAPS) 예측(KLFS) 자료
- 초단기모델(KLAPS) 예측(KLFS) 수치예상일기도
- (생산종료)초단기모델(VDAPS) 수치예상일기도

초단기모델(KLAPS) 분석(KL05) 자료

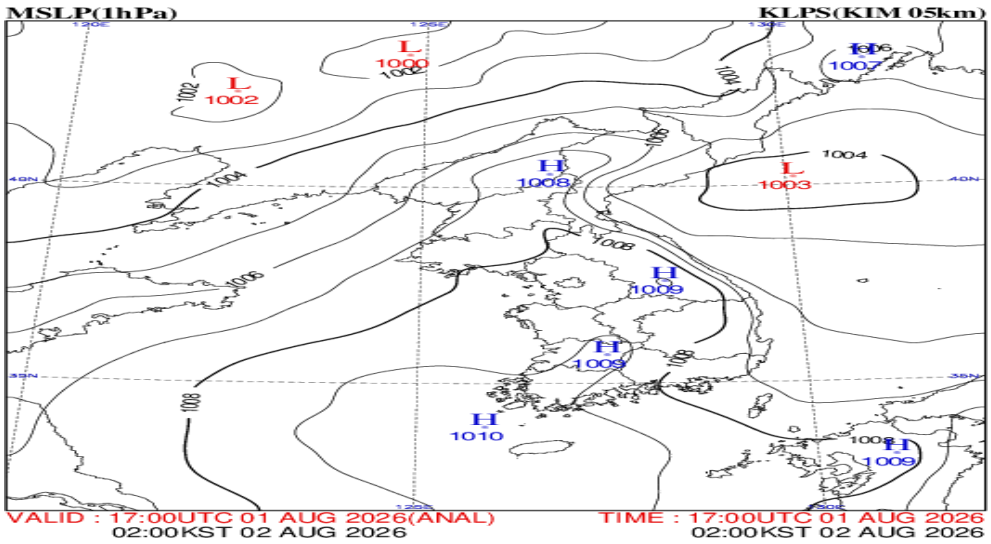
개요

자료상태 : 현행

초단기예측시스템(KLAPS, Korea Local Analysis and Prediction System)은 배경 및 초기장으로 활용되는 전지구예보시스템에 따라 KIM-KLAPS('23년 2월 현업 운영) 모델이 운영 중이다. 초단기 분석 및 예측 시스템은 한반도 영역에 대해 5km 수평해상도로 매 10분마다 원격, 지점, 종관 등 다양한 기상관측자료를 수집·분석하고 이를 초기장으로 활용하여 12시간까지 초단기 강수 및 예측정보를 생산하고 있다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 5km · 연직층수 : 40층
발표시각	일 144회(매 10분 간격)
제공기간	2019년 7월 16일 ~ 현재
생산·갱신 주기	1시간 주기 자료
제공형식	NetCDF4
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 초단기예측 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 KLAPS 분석장(KLPS) 파일 변수 목록 참조

초단기모델(KLAPS) 분석(KL05) 수치분석일기도



개요

자료상태 : 현행

초단기예측시스템(KLAPS, Korea Local Analysis and Prediction System)은 배경 및 초기장으로 활용되는 전지구예보시스템에 따라 KIM-KLAPS('23년 2월 현업 운영) 모델이 운영 중이다. 초단기 분석 및 예측 시스템은 한반도 영역에 대해 5km 수평해상도로 매 10분마다 원격, 지점, 종관 등 다양한 기상관측자료를 수집·분석하고 이를 초기장으로 활용하여 12시간까지 초단기 강수 및 예측정보를 생산하고 있습니다.

현재 기상청에서는 KIM-KLAPS 분석자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 5km · 연직층수 : 40층
발표시각	일 144회(매 10분 간격)
제공기간	2019년 7월 16일 ~ 현재
생산·갱신 주기	1시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델 자료 활용 가이던스 참조

3.초단기모델

- 초단기모델(KLAPS) 분석(KL05) 자료
- 초단기모델(KLAPS) 분석(KL05) 수치분석일기도
- 초단기모델(KLAPS) 예측(KLFS) 자료
- 초단기모델(KLAPS) 예측(KLFS) 수치예상일기도
- (생산종료)초단기모델(VDAPS) 수치예상일기도

초단기모델(KLAPS) 예측(KLFS) 자료

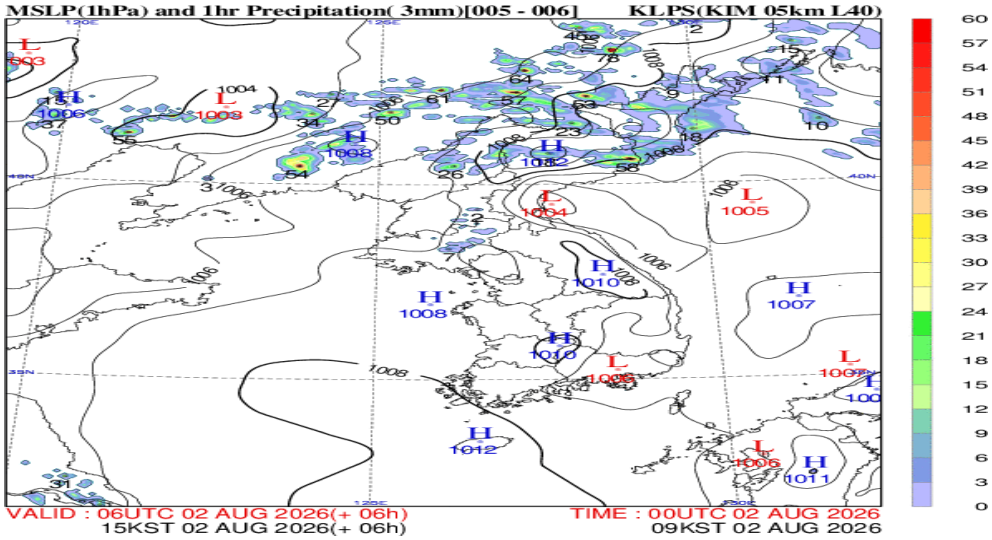
개요

자료상태 : 현행

한반도의 기상예측을 위해 수행하는 수치예보시스템을 의미하며, 매시간 Barnes 객관분석기법으로 3차원 변분자료 동화를 통해 생산됩니다. 주로 한반도영역의 3차원 분석·예측에 이용됩니다. 초단기예측(KLFS)은 한반도 영역을 대상으로 3차원 예측자료를 생산합니다.

주요요소	기압, 고도, 기온 등
해 상 도	· 수평분해능 : 5km · 연직층수 : 40층
발표시각	일 144회
예측시간	12시간 예측
제공기간	2019년 7월 16일 ~ 현재
생산·갱신주기	1시간 주기 자료
제공형식	NetCDF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

초단기모델(KLAPS) 예측(KLFS) 수치예상일기도



개요

자료상태 : 현행

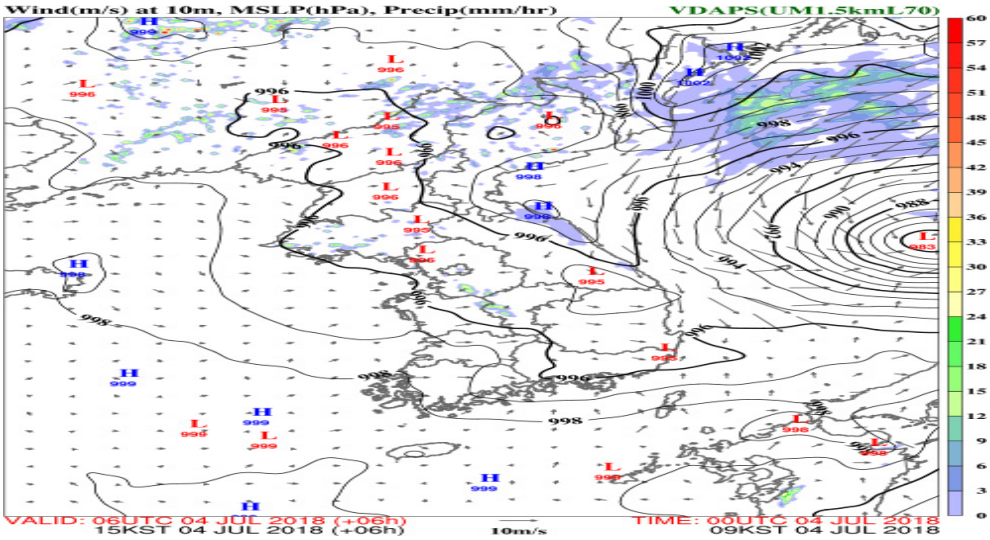
한반도의 기상예측을 위해 수행하는 수치예보시스템을 의미하며, 매시간 Barnes 객관분석기법으로 3차원 변분자료 동화를 통해 생산됩니다. 주로 한반도영역의 3차원 분석·예측에 이용됩니다. 초단기예측(KLFS)는 한반도 영역을 대상으로 3차원 예측자료를 생산합니다.
현재 기상청에서는 KIM-KLAPS 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	기압, 고도, 기온 등
해 상 도	· 수평분해능 : 5km · 연직층수 : 40층
발표시각	일 144회
예측시간	12시간 예측
제공기간	2019년 7월 16일 ~ 현재
생산·갱신 주 기	1시간 주기 자료
제공형식	GIF, PNG
제공경로	기상청 API허브(https://apihub.kma.go.kr/) → 수치모델 → 수치예보모델 공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델 자료 활용 가이던스 참조

3.초단기모델

- 초단기모델(KLAPS) 분석 (KL05) 자료
- 초단기모델(KLAPS) 분석 (KL05) 수치분석일기도
- 초단기모델(KLAPS) 예측 (KLFS) 자료
- 초단기모델(KLAPS) 예측 (KLFS) 수치예상일기도
- (생산종료)초단기모델 (VDAPS) 수치예상일기도

(생산종료)초단기모델(VDAPS) 수치예상일기도



개요

자료상태 : 생산종료

VDAPS(Very Short range Data Assimilation and Prediction System)는 통합모델 기반 초단기예보시스템으로 3차원 실황자료 및 초단기 예측자료를 생산, 제공합니다. 현재 VDAPS 모델 종료로 예측자료를 활용한 일기도는 미생산 중입니다.

주요요소	기압, 고도, 기온 등
해 상 도	· 수평분해능 : 1.5km · 연직층수 : 70층
발표시각	일 24회
예측시간	12시간 예측
제공기간	2017년 6월 21일 ~ 2021년 12월 28일
생산·갱신 주 기	1시간 주기 자료
제공형식	PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

(생산종료)전지구 앙상블모델 (UM-EPSG) 예측자료

4.앙상블모델

- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- (생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도
- 전지구앙상블모델(KIM-EPSG) 예측자료
- 전지구앙상블모델(KIM-EPSG) 수치예상일기도
- 국지앙상블모델(KIM-LENS) 예측자료
- 국지앙상블모델(KIM-LENS) 수치예상일기도

개요

자료상태 : 생산종료

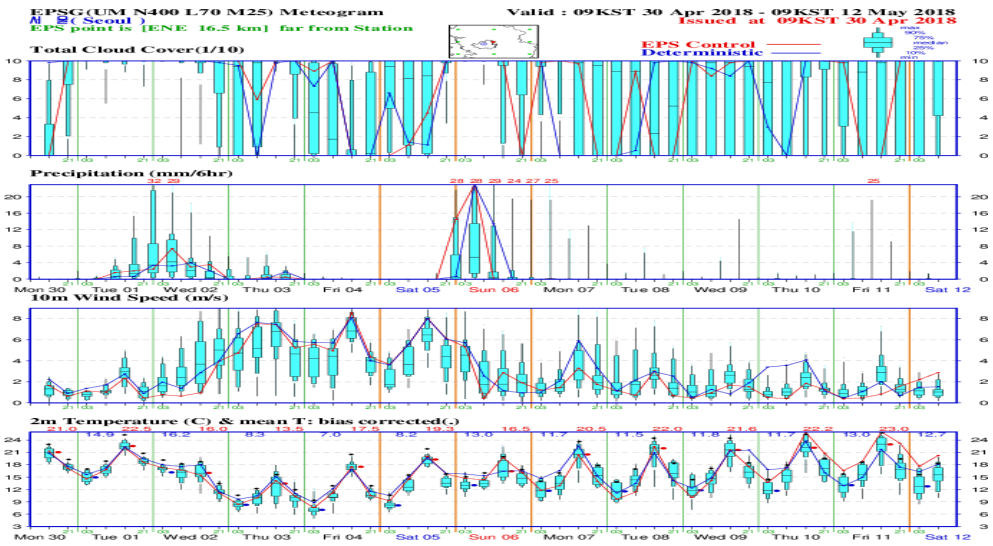
전지구 영역을 대상으로 일정 수 이상의 모델 구성을 통해 기상현상의 확률 예측 결과를 생산하는 수치예보모델 시스템이며, 전지구 영역의 확률 수치예측 자료를 생산하고 주간 예보에 이용됩니다. 수평해상도는 32km, 연직층수는 70층의 49멤버로 구성되어있습니다. 현재 UM 모델 종료로 UM-EPSG 예측자료(GRIB)은 미생산 중입니다.

주요요소	바람, 기온, 습도 등 기상요소
해 상 도	· 수평분해능 : 32km · 연직층수 : 70층
발표시각	일 2회(00, 12UTC)
예측시간	12일
제공기간	2015년 1월 3일 ~ 2026년 3월 31일
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

4.앙상블모델

- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- (생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도
- 전지구앙상블모델(KIM-EPSG) 예측자료
- 전지구앙상블모델(KIM-EPSG) 수치예상일기도
- 국지앙상블모델(KIM-LENS) 예측자료
- 국지앙상블모델(KIM-LENS) 수치예상일기도

(생산종료)전지구 앙상블모델 (UM-EPSG) 수치예상일기도



개요

자료상태 : 생산종료

전지구 영역을 대상으로 일정 수 이상의 모델 구성을 통해 기상현상의 확률 예측 결과를 생산하는 수치예보모델 시스템이며, 전지구 영역의 확률 수치예측 자료를 생산하고 주간 예보에 이용됩니다.
수평해상도는 32km, 연직층수는 70층의 49멤버로 구성되어있습니다.
현재 UM 모델 종료로 UM-EPSG 예측자료 이미지는 미생산 중입니다.

주요요소	바람, 기온, 습도 등 기상요소
해 상 도	· 수평분해능 : 32km · 연직층수 : 70층
발표시각	일 2회(00, 12UTC)
예측시간	12일
제공기간	2015년 12월 2일 ~ 2026년 3월 31일
생산·갱신 주기	12시간 주기 자료
제공형식	GIF, PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 기상청 날씨누리(https://www.weather.go.kr) → 알림자료 → 자료 → 기상간행물 → 손에 잡히는 예보기술(검색) 참조

(생산종료)국지앙상블모델 (UM-LENS) 예측자료

4.앙상블모델
- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- (생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도
- 전지구앙상블모델(KIM-EPSG) 예측자료
- 전지구앙상블모델(KIM-EPSG) 수치예상일기도
- 국지앙상블모델(KIM-LENS) 예측자료
- 국지앙상블모델(KIM-LENS) 수치예상일기도

개요

자료상태 : 생산종료

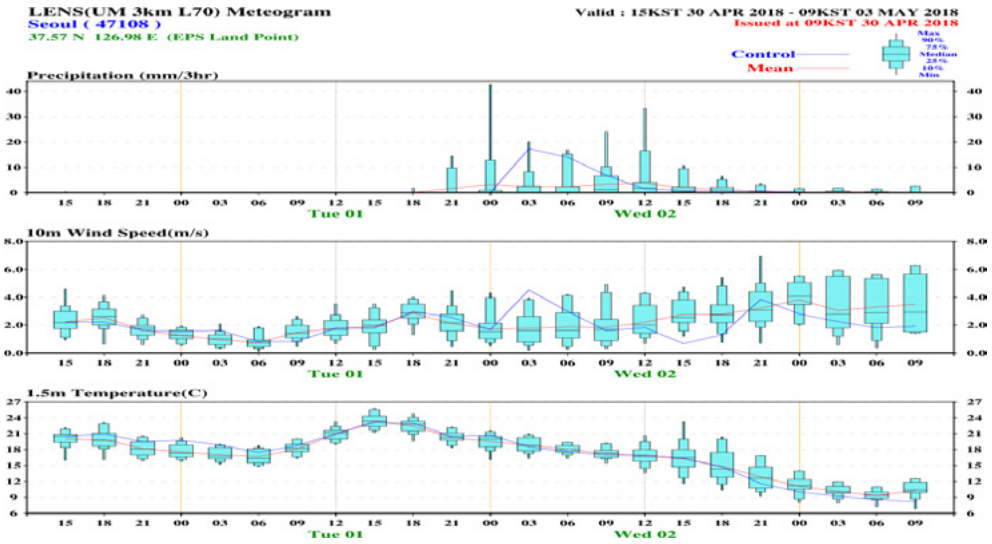
국지 영역을 대상으로 일정 수 이상의 모델 구성을 통해 기상현상의 확률 예측 결과를 생산하는 수치예보모델시스템이며, 국지 영역의 확률 수치예측 자료를 생산합니다. 3km 격자간격의 13멤버로 구성되어있습니다.
현재 UM 모델 종료로 UM-LENS 예측자료(GRIB)는 미생산 중입니다.

주요요소	등압면, 단일면, 모델면
해 상 도	· 수평분해능 : 3km · 연직층수 : 70층
발표시각	일 2회(00, 12UTC)
예측시간	72시간
제공기간	2015년 10월 29일 ~ 2026년 3월 31일
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다.

4.앙상블모델

- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- **(생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도**
- 전지구앙상블모델(KIM-EPSG) 예측자료
- 전지구앙상블모델(KIM-EPSG) 수치예상일기도
- 국지앙상블모델(KIM-LENS) 예측자료
- 국지앙상블모델(KIM-LENS) 수치예상일기도

(생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도



개요

자료상태 : 생산종료

국지 영역을 대상으로 일정 수 이상의 모델 구성을 통해 기상현상의 확률 예측 결과를 생산하는 수치예보모델시스템이며, 국지 영역의 확률 수치예측 자료를 생산합니다. 3km 격자간격의 13멤버로 구성되어있습니다.

현재 UM 모델 종료로 UM-LENS 예측자료 이미지는 미생산 중입니다.

주요요소	등압면, 단일면, 모델면
해상도	· 수평분해능 : 3km · 연직층수 : 70층
발표시각	일 2회(00, 12UTC)
예측시간	72시간
제공기간	2015년 12월 2일 ~ 2026년 3월 31일
생산·갱신주기	시간(12시간 주기) 자료
제공형식	PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상청 날씨누리(https://www.weather.go.kr) → 알림자료 → 자료 → 기상간행물 → 손에 잡히는 예보기술(검색) 참조

전지구앙상블모델(KIM-EPSG) 예측자료

개요

자료상태 : 현행

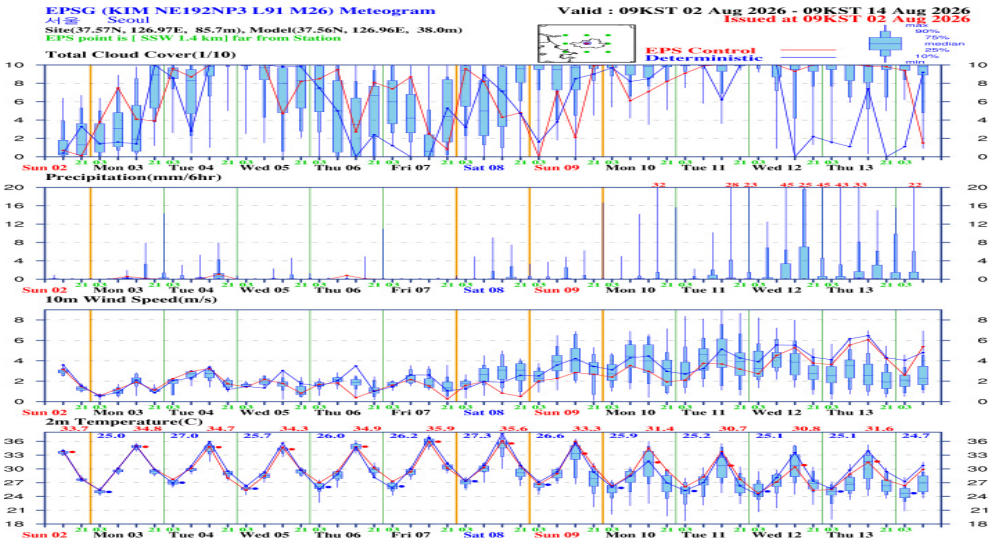
한국형모델 기반의 전지구앙상블예측시스템(EPSG, Ensemble Prediction System Global)은 32km/24km의 수평해상도와 91개의 연직층을 가지고 있으며, 총 26개의 앙상블 멤버를 활용하여 일 2회(00, 12UTC) 12일 예측을 생산한다. 또 일 4회(00, 06, 12, 18UTC) 50개의 앙상블멤버로부터 앙상블 에러 정보를 생산하는데, 이렇게 생산된 에러 정보는 4차원 앙상블 변분자료동화과정에서 전지구모델의 오차정보를 실시간 갱신하는데 활용된다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 32km/24km(2025년 5월 14일부터~현재) · 연직층수 : 91층
발표시각	일 2회(00, 12UTC)
예측시간	360시간 예측
제공기간	2021년 10월 13일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	NetCDF4 (2026년 6월 17일 이후부터는 표준화 자료로 제공)
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	(GRIB) 1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다. (NetCDF) 1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

4.앙상블모델

- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- (생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도
- 전지구앙상블모델(KIM-EPSG) 예측자료
- **전지구앙상블모델(KIM-EPSG) 수치예상일기도**
- 국지앙상블모델(KIM-LENS) 예측자료
- 국지앙상블모델(KIM-LENS) 수치예상일기도

전지구앙상블모델(KIM-EPSG) 수치예상일기도



개요

자료상태 : 현행

한국형모델 기반의 전지구앙상블예측시스템(EPSG, Ensemble Prediction System Global)은 32km/24km의 수평해상도와 91개의 연직층을 가지고 있으며, 총 26개의 앙상블 멤버를 활용하여 일 2회(00, 12UTC) 12일 예측을 생산하고 있습니다. 또 일 4회(00, 06, 12, 18UTC) 50개의 앙상블멤버로부터 앙상블 에러 정보를 생산하는데, 이렇게 생산된 에러 정보는 4차원 앙상블 변분자료동화과정에서 전지구모델의 오차정보를 실시간 갱신하는데 활용됩니다.
현재 기상청에서는 EPSG 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 32km/24km(2025년 5월 14일부터~현재) · 연직층수 : 91층
발표시각	일 2회(00, 12UTC)
예측시간	360시간 예측
제공기간	2021년 10월 13일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

국지앙상블모델(KIM-LENS) 예측자료

개요

자료상태 : 현행

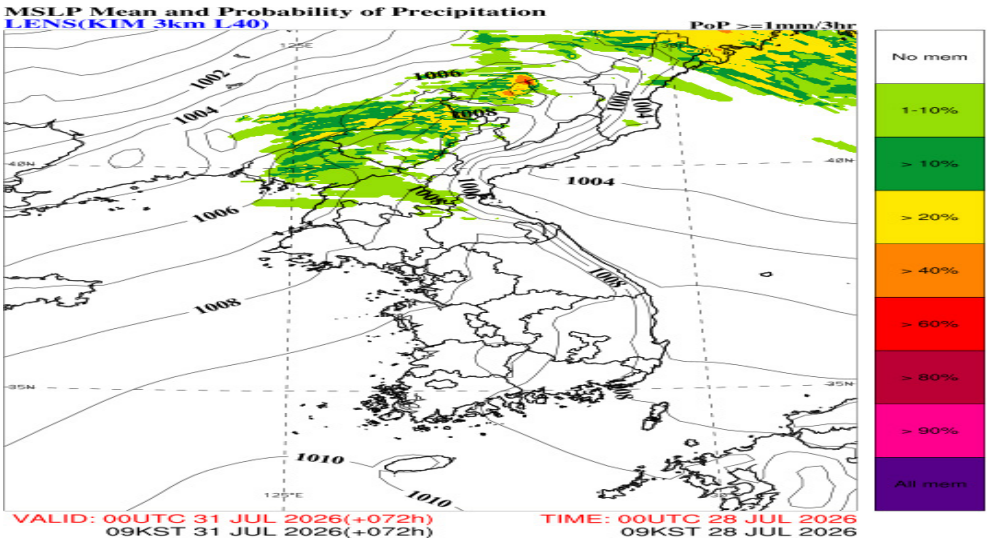
한국형모델 기반의 국지앙상블예측시스템(LENS, Local ENsemble prediction System)은 KIM 기반으로 각각 운영되고 있습니다. 13개의 앙상블 멤버를 활용하여 일 2회(00, 12UTC) 자료를 생산하며, KIM-LENS는 5일까지 예측이 이루어집니다.
현재 기상청에서는 KIM-LENS 예측자료를 GRIB2 및 NetCDF4 형식으로 생산·저장, 제공하고 있으며 2026년 9월 GRIB2 형식 자료는 서비스를 종료하고 NetCDF4 형식(표준화 자료)으로 일원화하여 제공됩니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 3km · 연직층수 : 40층
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	(GRIB) 2024년 5월 13일 ~ 2026년 9월 30일 (NetCDF) 2026년 6월 23일 ~ 현재 (표준화 자료)
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2, NetCDF4
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	(GRIB) 1. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치예보모델자료 활용 가이던스 참조 2. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델자료활용매뉴얼 참조 3. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 수치모델데이터 라이브러리를 활용하여 데이터를 읽을 수 있습니다. 4. 기상자료개방포털(http://data.kma.go.kr)-소통과 참여-자료실에서 g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있습니다. (NetCDF) 1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

4.앙상블모델

- (생산종료)전지구앙상블모델 (UM-EPSG) 예측자료
- (생산종료)전지구앙상블모델 (UM-EPSG) 수치예상일기도
- (생산종료)국지앙상블모델 (UM-LENS) 예측자료
- (생산종료)국지앙상블모델 (UM-LENS) 수치예상일기도
- 전지구앙상블모델(KIM-EPSG) 예측자료
- 전지구앙상블모델(KIM-EPSG) 수치예상일기도
- 국지앙상블모델(KIM-LENS) 예측자료
- **국지앙상블모델(KIM-LENS) 수치예상일기도**

국지앙상블모델(KIM-LENS) 수치예상일기도



개요

자료상태 : 현행

한국형모델 기반의 국지앙상블예측시스템(LENS, Local ENsemble prediction System)은 KIM 기반으로 각각 운영되고 있다. 13개의 앙상블 멤버를 활용하여 일 2회(00, 12UTC) 자료를 생산하며, KIM-LENS는 5일까지 예측이 이루어집니다.
현재 기상청에서는 KIM-LENS 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	등압면, 단일면
해 상 도	· 수평분해능 : 3km · 연직층수 : 40층
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	2024년 5월 12일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조 수치예보센터 홈페이지(https://www.kma.go.kr/nmc/html/main/index.jsp) → 정보마당 → 자료실 → 수치일기도 매뉴얼 참조

전지구파랑모델(GWW3) 예측자료

개요

자료상태 : 현행

전지구 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위하여 전지구 해상파랑 지역을 대상으로 수행하는 모델이며, 주로 단기예보 및 주간해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 GWW3 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능: 25km
발표시각	일 2회(00, 12UTC)
예측시간	12일 예측
제공기간	(UM) 2012년 11월 27일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 파랑모델 → 전지구파랑모델
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

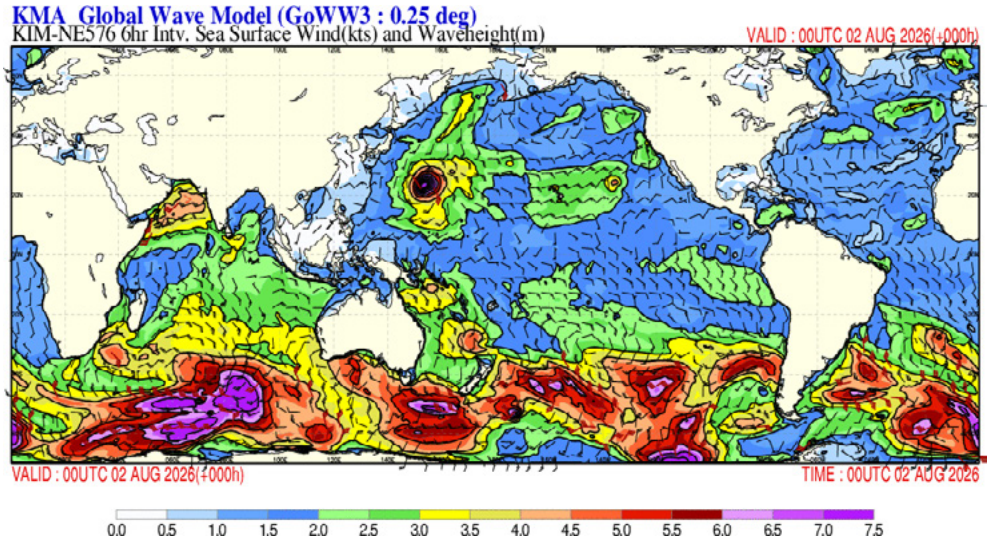
5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

전지구파랑모델(GWW3) 수치예상일기도



개요

자료상태 : 현행

전지구 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위하여 전지구 해상파랑 지역을 대상으로 수행하는 모델이며, 주로 단기예보 및 주간해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 GWW3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능: 25km
발표시각	일 2회(00, 12UTC)
예측시간	12일 예측
제공기간	(UM) 2015년 12월 2일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

지역파랑모델(RWW3) 예측자료

개요

자료상태 : 현행

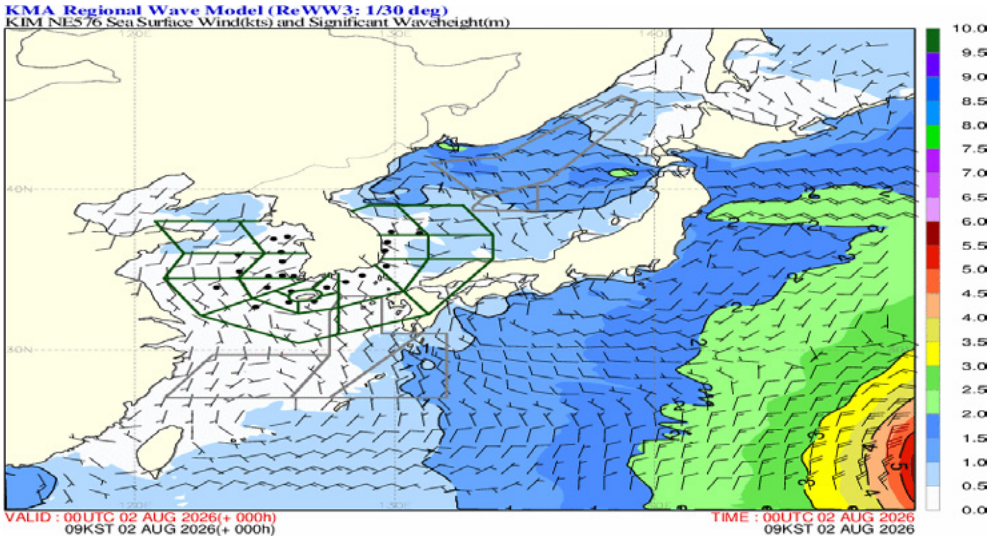
지역 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 주로 단기 해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 RWW3 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능 : 4km
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	(UM) 2012년 11월 27일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 파랑모델 → 지역 파랑모델
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- **지역파랑모델(RWW3) 수치예상일기도**
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

지역파랑모델(RWW3) 수치예상일기도



개요

자료상태 : 현행

지역 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 주로 단기 해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 RWW3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능 : 4km
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	(UM) 2015년 12월 2일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF, PNG
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도 → 지역파랑모델 수치예상 일기도
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

지역앙상블파랑모델(EWW3) 예측자료

개요

자료상태 : 현행

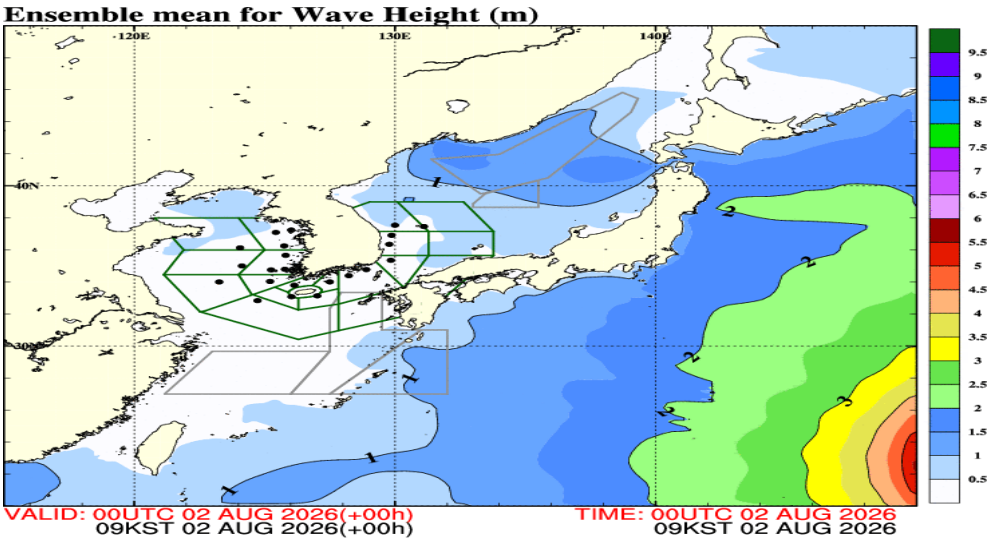
지역앙상블파랑모델은 전지구앙상블모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다.
현재 기상청에서는 KIM 모델 기반 EWW3 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	유의파고, 최대파주기, 평균파향 등의 예측확률 정보
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	(UM) 2018년 10월 31일 ~ 2026년 3월 31일 (KIM) 2025년 5월 22일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- **지역앙상블파랑모델(EWW3) 수치예상일기도**
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

지역앙상블파랑모델(EWW3) 수치예상일기도



개요

자료상태 : 현행

지역앙상블파랑모델은 전지구앙상블모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다.
현재 기상청에서는 KIM 모델 기반 EWW3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	유의파고, 최대파주기, 평균파향 등의 예측확률 정보
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간 예측
제공기간	(UM) 2018년 10월 31일 ~ 2026년 3월 31일 (KIM) 2025년 5월 22일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

국지연안파랑모델(CWW3) 예측자료

개요

자료상태 : 현행

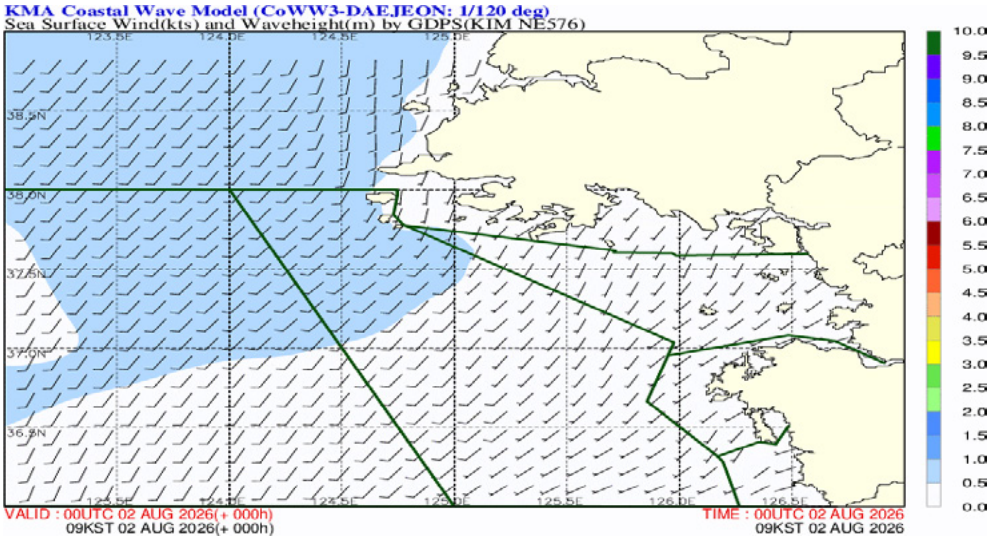
국지연안 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 상세 국지연안을 대상으로 단기 해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 CWW3 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능 : 1km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	(UM) 2012년 12월 4일 ~ 2026년 3월 31일 (KIM) 2021년 10월 7일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB2
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 파랑모델 → 국지연안파랑모델
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- **국지연안파랑모델(CWW3) 수치예상일기도**
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

국지연안파랑모델(CWW3) 수치예상일기도



개요

자료상태 : 현행

국지연안 파랑모델은 전지구 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 상세 국지연안을 대상으로 단기 해상예보에 이용됩니다. 현재 기상청에서는 KIM 모델 기반 CWW3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	해상풍, 유의파고, 파향, 파장, 파주기 등
해 상 도	수평분해능 : 1km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	(UM) 2015년 12월 2일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

초단기파랑모델(KWW3) 예측자료

5.파랑모델
- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- 초단기파랑모델(KWW3) 수치예상일기도

개요

자료상태 : 현행

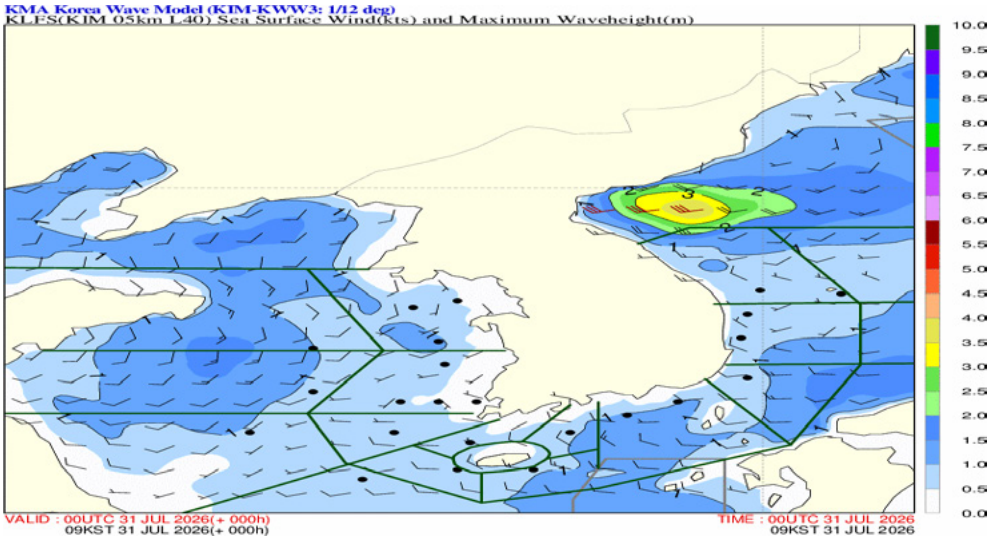
초단기 파랑모델은 초단기 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 현재 기상청에서는 KIM 모델 기반 KWW3 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	해상풍, 유의파고, 너울, 풍파 등
해 상 도	수평분해능 : 8km
발표시각	일 24회
예측시간	12시간 예측
제공기간	(UM) 2025년 11월 10일 ~ 2026년 3월 31일 (KIM) 2026년 2월 26일 ~ 현재
생산·갱신 주기	1시간 주기 자료
제공형식	GRIB
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

5.파랑모델

- 전지구파랑모델(GWW3) 예측자료
- 전지구파랑모델(GWW3) 수치예상일기도
- 지역파랑모델(RWW3) 예측자료
- 지역파랑모델(RWW3) 수치예상일기도
- 지역앙상블파랑모델(EWW3) 예측자료
- 지역앙상블파랑모델(EWW3) 수치예상일기도
- 국지연안파랑모델(CWW3) 예측자료
- 국지연안파랑모델(CWW3) 수치예상일기도
- 초단기파랑모델(KWW3) 예측자료
- **초단기파랑모델(KWW3) 수치예상일기도**

초단기파랑모델(KWW3) 수치예상일기도



개요

자료상태 : 현행

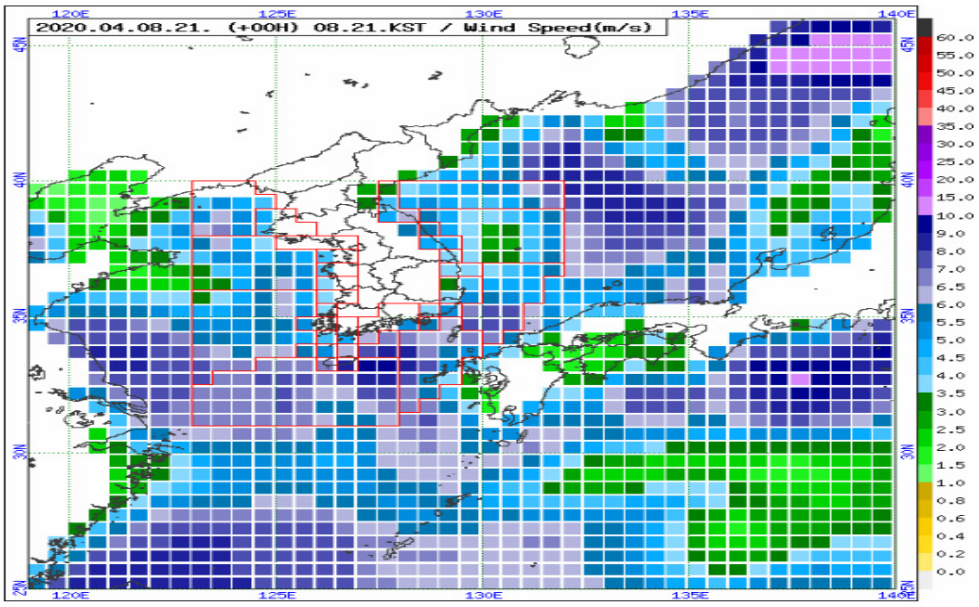
초단기 파랑모델은 초단기 예보모델의 기상 예측장(해상풍)을 활용하여 해상의 날씨를 예보하기 위한 수치예보시스템입니다. 현재 기상청에서는 KIM 모델 기반 KWW3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	해상풍, 유의파고, 너울, 풍파 등
해 상 도	수평분해능 : 8km
발표시각	일 24회
예측시간	12시간 예측
제공기간	(UM) 2019년 10월 23일 ~ 2026년 3월 31일 (KIM) 2026년 2월 26일 ~ 현재
생산·갱신 주기	1시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

대해구 예측정보

6.해구별예측

• 대해구 예측정보
• 소해구 예측정보



개요

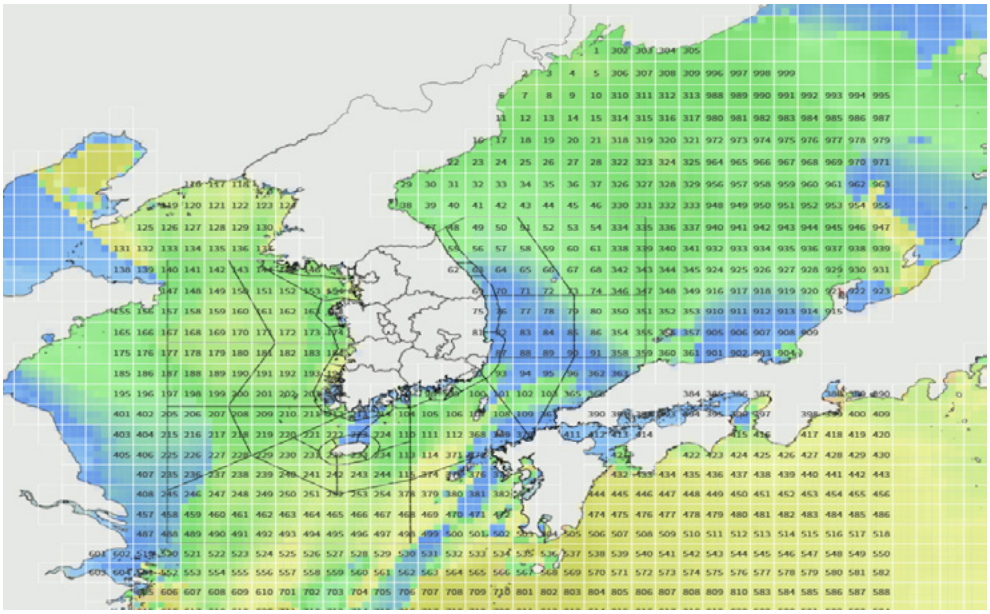
자료상태 : 현행

해구별 예측정보는 지역 파랑모델(RWW3)의 예측자료를 활용하여 산출한 50km(0.5°)격자의 해구별 바다 날씨에 관한 정보입니다. 한반도 주변 연근해 1331개의 해구를 대상으로 하며, 현재 기상청에서는 해구별 예측정보를 TXT 형태로 생산하여 제공하고 있습니다.

주요요소	유의파고, 파향, 파주기, 풍속, 풍향
해 상 도	50km(0.5°)
발표시각	일 2회(00, 12UTC)
예측시간	72시간(3시간 간격)
제공기간	2016년 7월 ~ 현재
생산·갱신 주기	시간(12시간 주기) 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 수치모델 → 파랑모델 → 해구별 예측자료
이용방법	해구번호·발표시각·예측시간과 요소를 선택합니다. 해구의 경계·대표 위치와 요소별 단위, 예측 유효시각을 확인합니다.

6.해구별예측 소해구 예측정보

• 대해구 예측정보
• 소해구 예측정보



개요

자료상태 : 현행

지역 파랑모델(RWW3), 장기기후모델(GloSea6), 지역 대기모델(RDPS) 등을 활용하여 산출한 소해구 해상 날씨 예측정보

※ 대해구: 우리나라 인근 해역을 위·경도 0.5°(약 50km×50km) 간격의 격자로 나누어 번호를 부여한 바다의 주소 체계
※ 소해구: 대해구를 다시 가로와 세로로 3등분(3×3)해 소해구로 구성

주요요소	유의파고, 파향, 파주기, 풍속, 풍향, 수온, 시정, 예상 강수량, 너울 위험 예측정보	
해 상 도	17km	
발표시각	일 2회	
예측시간	120시간 ※ 제공경로 및 요소에 따라 일부 상이함	
제공기간	2026년 7월 1일 ~ 현재	
생산·갱신주기	일 2회(12시간 주기)	
제공형식	웹이미지, ASCII	
제공경로	웹이미지	해양기상기후정보포털(marine.kma.go.kr)→해양기상예보→해구별 해양기상정보
	ASCII	API허브(https://apihub.kma.go.kr) → 수치모델 → 해구별 예측데이터 조회
이용방법	해구번호·발표시각·예측시간과 요소를 선택합니다. 해구의 경계·대표 위치와 요소별 단위, 예측 유효시각을 확인합니다.	

지역폭풍해일모델(RTSM) 예측자료

7.폭풍해일 모델
- 지역폭풍해일모델(RTSM) 예측자료
- 지역폭풍해일모델(RTSM) 수치예상일기도
- 국지연안폭풍해일모델(CTSM) 수치예상일기도
- 지역앙상블폭풍해일모델 (ETSM) 예측자료
- 지역앙상블폭풍해일모델 (ETSM) 수치예상일기도

개요

자료상태 : 현행

지역폭풍해일모델(RTSM, Regional Tide/Storm Surge Model)은 전지구 예보모델의 기상 예측 결과(해면기압, 해상풍)를 활용하여 한반도를 포함한 동아시아 해역의 조석과 해수면 높이를 계산하고, 폭풍해일고(해수면높이-조석)를 생산하는 모델입니다. 동아시아 및 한반도 주변 해역 폭풍해일 예보에 이용됩니다.

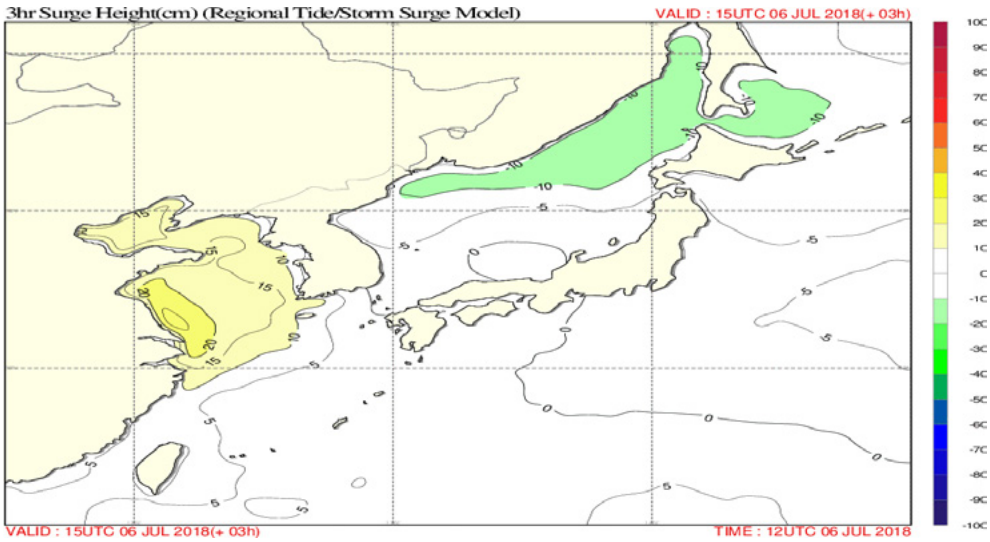
현재 기상청에서는 KIM 모델 기반 RTSM 예측자료를 GRIB2형식으로 만들어 저장, 제공하고 있습니다.

주요요소	폭풍해일고 등
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	(UM) 2017년 2월 27일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

7.폭풍해일 모델

- 지역폭풍해일모델(RTSM) 예측자료
- 지역폭풍해일모델(RTSM) 수치예상일기도
- 국지연안폭풍해일모델(CTSM) 수치예상일기도
- 지역앙상블폭풍해일모델 (ETSM) 예측자료
- 지역앙상블폭풍해일모델 (ETSM) 수치예상일기도

지역폭풍해일모델(RTSM) 수치예상일기도



개요

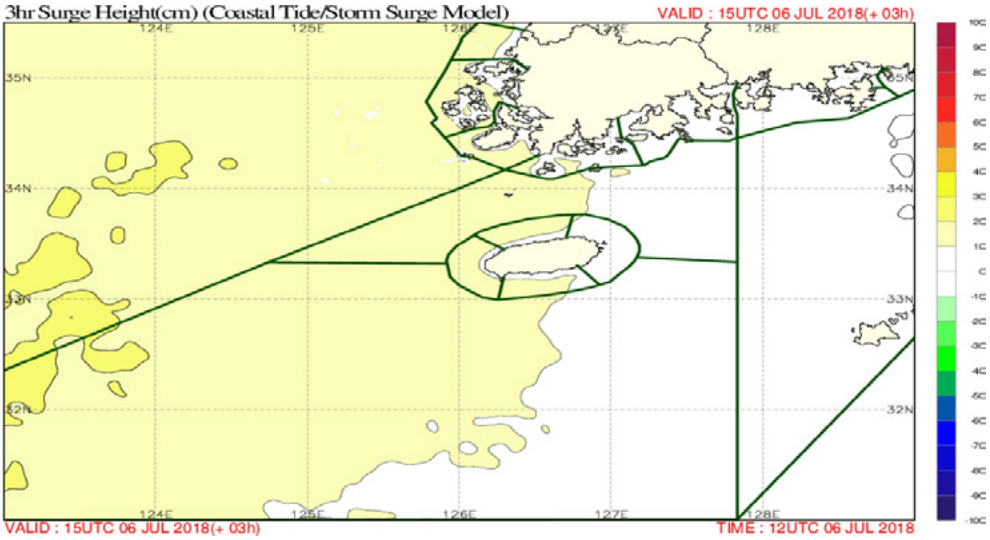
자료상태 : 현행

지역폭풍해일모델(RTSM, Regional Tide/Storm Surge Model)은 전지구 예보모델의 기상 예측 결과(해면기압, 해상풍)를 활용하여 한반도를 포함한 동아시아 해역의 조석과 해수면 높이를 계산하고, 폭풍해일고(해수면높이-조석)를 생산하는 모델입니다.
동아시아 및 한반도 주변 해역 폭풍해일 예보에 이용됩니다.
현재 기상청에서는 KIM 모델 기반 RTSM 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	폭풍해일고 등
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	(UM) 2015년 12월 2일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델 자료 활용 가이던스 참조

국지연안폭풍해일모델(CTSM) 수치예상일기도

7.폭풍해일 모델
- 지역폭풍해일모델(RTSM) 예측자료
- 지역폭풍해일모델(RTSM) 수치예상일기도
- 국지연안폭풍해일모델(CTSM) 수치예상일기도
- 지역앙상블폭풍해일모델 (ETSM) 예측자료
- 지역앙상블폭풍해일모델 (ETSM) 수치예상일기도



개요

자료상태 : 현행

국지연안폭풍해일모델(CTSM, Coastal Tide/Storm Surge Model)은 전지구 예보모델의 기상 예측 결과(해면기압, 해상풍)를 활용하여 국지연안의 조석과 해수면 높이를 계산하고, 폭풍해일고(해수면높이-조석)를 생산하는 모델이며, 국지연안 폭풍해일 예보에 이용됩니다.
현재 기상청에서는 KIM 모델 기반 CTSM 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	폭풍해일고 등
해 상 도	수평분해능 : 1km
발표시각	일 2회(00, 12UTC)
예측시간	72시간
제공기간	(UM) 2015년 12월 2일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GIF
제공경로	기상자료개방포털(http://data.kma.go.kr) → 데이터 → 수치모델 → 수치분석일기도
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델자료 활용 가이던스 참조

7.폭풍해일 모델

- 지역폭풍해일모델(RTSM) 예측자료
- 지역폭풍해일모델(RTSM) 수치예상일기도
- 국지연안폭풍해일모델(CTSM) 수치예상일기도
- 지역앙상블폭풍해일모델 (ETSM) 예측자료
- 지역앙상블폭풍해일모델 (ETSM) 수치예상일기도

지역앙상블폭풍해일모델(ETSM) 예측자료

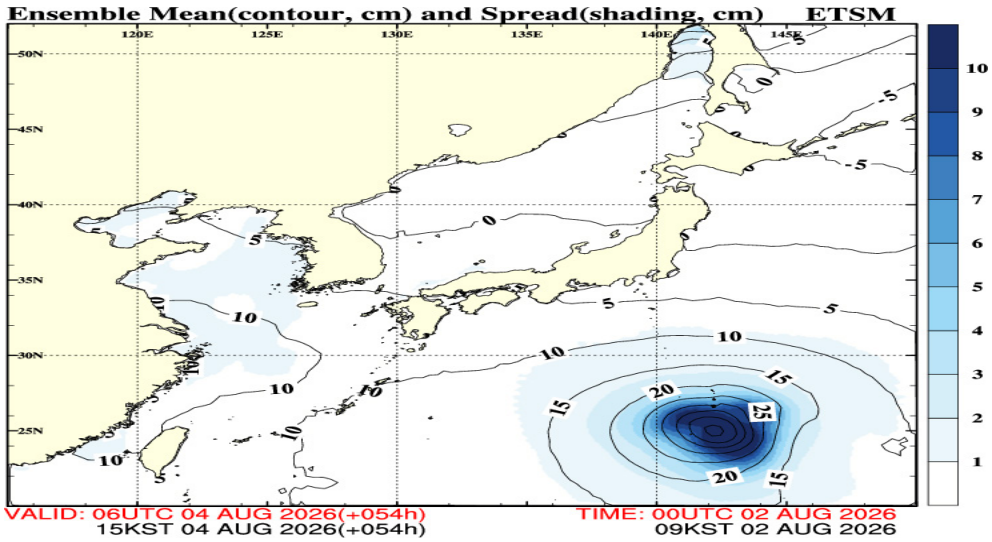
개요

자료상태 : 현행

지역앙상블 폭풍해일모델(ETSM, Ensemble Tide/Storm Surge Model)은 전구앙상블모델의 기상 예측 결과(해면기압, 해상풍)를 활용하여 한반도를 포함한 동아시아 해역의 조석과 해수면 높이를 계산하고, 폭풍해일고(해수면높이-조석)를 생산하는 모델입니다. 동아시아 및 한반도 주변 해역 폭풍해일 예보에 이용됩니다.
현재 기상청에서는 KIM 모델 기반 ETSM 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	폭풍해일고 등
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	2025년 5월 22일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	GRIB
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

지역앙상블폭풍해일모델(ETSM) 수치예상일기도



개요

자료상태 : 현행

지역앙상블 폭풍해일모델(ETSM, Ensemble Tide/Storm Surge Model)은 전구앙상블모델의 기상예측 결과(해면기압, 해상풍)를 활용하여 한반도를 포함한 동아시아 해역의 조석과 해수면 높이를 계산하고, 폭풍해일고(해수면높이-조석)를 생산하는 모델입니다 동아시아 및 한반도 주변 해역 폭풍해일 예보에 이용됩니다.

현재 기상청에서는 KIM 모델 기반 ETSM 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	폭풍해일고 등
해 상 도	수평분해능 : 8km
발표시각	일 2회(00, 12UTC)
예측시간	120시간
제공기간	2025년 5월 22일 ~ 현재
생산·갱신 주기	12시간 주기 자료
제공형식	PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델자료 활용 가이던스 참조

8. 황사모델

• 황사·연무모델(ADAM3) 예측자료
• 황사·연무모델(ADAM3) 수치예상일기도
• (생산종료)황사모델(ADAM2) 예측자료
• (생산종료)황사모델(ADAM2) 수치예상일기도

황사·연무모델(ADAM3) 예측자료

개요

자료상태 : 현행

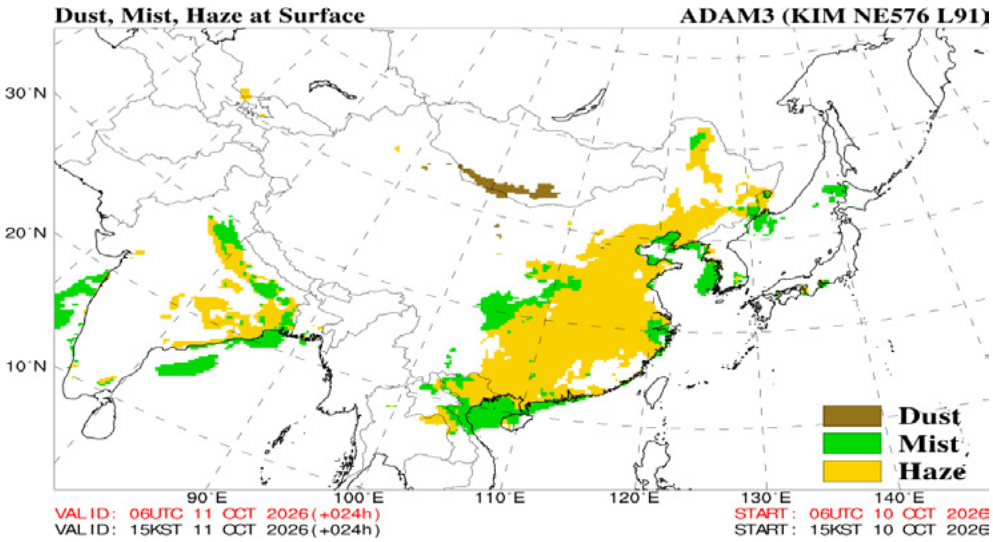
황사·연무 통합예측모델(ADAM3)을 통해 생성된 예측결과를 토대로 황사 및 연무 예보에 활용할 수 있도록 예측자료를 이미지 형태로 생산, 제공하고 있습니다.
현재 기상청에서는 KIM 모델 기반 ADAM3 예측자료를 NetCDF 형태로 생산하여 제공하고 있습니다.

주요요소	수평분포, 연직시계열 등
해 상 도	25km (49층)
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	168시간(00, 12UTC) 72시간(06, 18UTC)
제공기간	(UM) 2016년 6월 29일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	NetCDF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	1. NetCDF 는 범용 포맷으로, NetCDF 포맷에 대한 공식 홈페이지(https://www.unidata.ucar.edu/software/netcdf)에서 상세정보 참조 2. NetCDF 사용자 가이드(https://docs.unidata.ucar.edu/netcdf-c/current/) 참조 3. 공식 GitHub 소스 코드 저장소(https://github.com/Unidata/netcdf-c) 프로그램 참조 4. ncdump를 이용하여 –h 옵션으로 헤더 정보를 –v 옵션으로 자료를 읽을 수 있습니다.

황사·연무모델(ADAM3) 수치예상일기도

8. 황사모델

- 황사·연무모델(ADAM3) 예측자료
- 황사·연무모델(ADAM3) 수치예상일기도
- (생산종료)황사모델(ADAM2) 예측자료
- (생산종료)황사모델(ADAM2) 수치예상일기도



개요

자료상태 : 현행

황사·연무 통합예측모델(ADAM3)을 통해 생성된 예측결과를 토대로 황사 및 연무 예보에 활용할 수 있도록 예측자료를 이미지 형태로 생산, 제공하고 있습니다.
현재 기상청에서는 KIM 모델 기반 ADAM3 예측자료를 이미지 형태로 생산하여 제공하고 있습니다.

주요요소	수평분포, 연직시계열 등
해 상 도	25km (49층)
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	72시간(3시간 간격)
제공기간	(UM) 2016년 1월 10일 ~ 2026년 3월 31일 (KIM) 2020년 4월 28일 ~ 현재
생산·갱신 주기	6시간 주기 자료
제공형식	GIF, PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델자료 활용 가이던스 참조

8. 황사모델

- 황사·연무모델(ADAM3) 예측자료
- 황사·연무모델(ADAM3) 수치예상일기도
- (생산종료)황사모델(ADAM2) 예측자료
- (생산종료)황사모델(ADAM2) 수치예상일기도

(생산종료)황사·연무모델(ADAM2) 예측자료

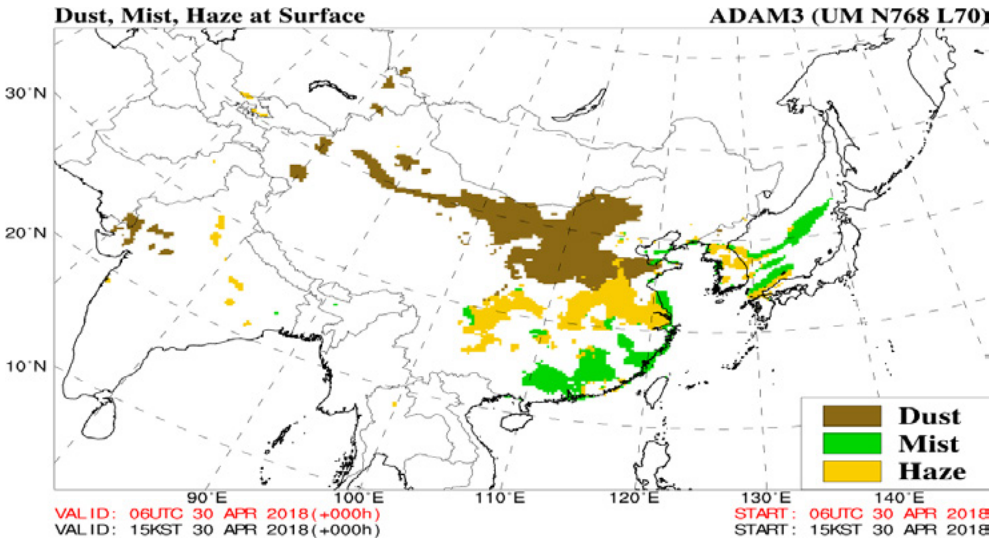
개요

자료상태 : 생산종료

황사·연무 통합예측모델(ADAM3)을 통해 생성된 예측결과를 토대로 황사 및 연무 예보에 활용할 수 있도록 예측자료를 이미지 형태로 생산, 제공하고 있습니다.

주요요소	수평분포, 연직시계열 등
해 상 도	25km (49층)
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	72시간(3시간 간격)
제공기간	2016년 6월 29일 ~ 2021년 12월 31일
생산·갱신 주기	6시간 주기 자료
제공형식	GRIB
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 1. 수치예보모델자료 활용 가이던스, 매뉴얼 참조 2. 수치모델데이터 라이브러리를 활용하여 데이터 읽기 가능 3. g2ctl라이브러리를 활용하여 Grads로 전환, 이미지를 생성할 수 있음

(생산종료)황사·연무모델(ADAM2) 수치예상일기도



개요

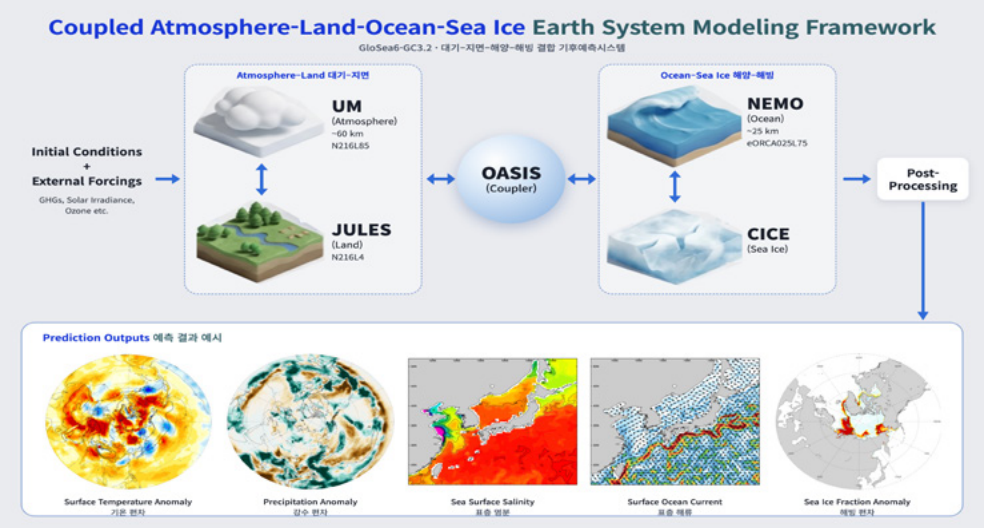
자료상태 : 생산종료

황사·연무 통합예측모델(ADAM3)을 통해 생성된 예측결과를 토대로 황사 및 연무 예보에 활용할 수 있도록 예측자료를 이미지 형태로 생산, 제공하고 있습니다.

주요요소	수평분포, 연직시계열 등
해 상 도	25km (49층)
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	72시간(3시간 간격)
제공기간	2016년 6월 29일 ~ 2021년 12월 28일
생산·갱신 주기	시간(6시간 주기) 자료
제공형식	GIF, PNG
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실 → 수치예보모델자료 활용 가이던스 참조

9.기후모델 (GLOS)
• 기후예측시스템(GloSea) 예측자료(GRIB)
• 기후예측모델(GloSea) 전지구 예측일기도(이미지)

기후예측시스템(GloSea) 예측자료(GRIB)



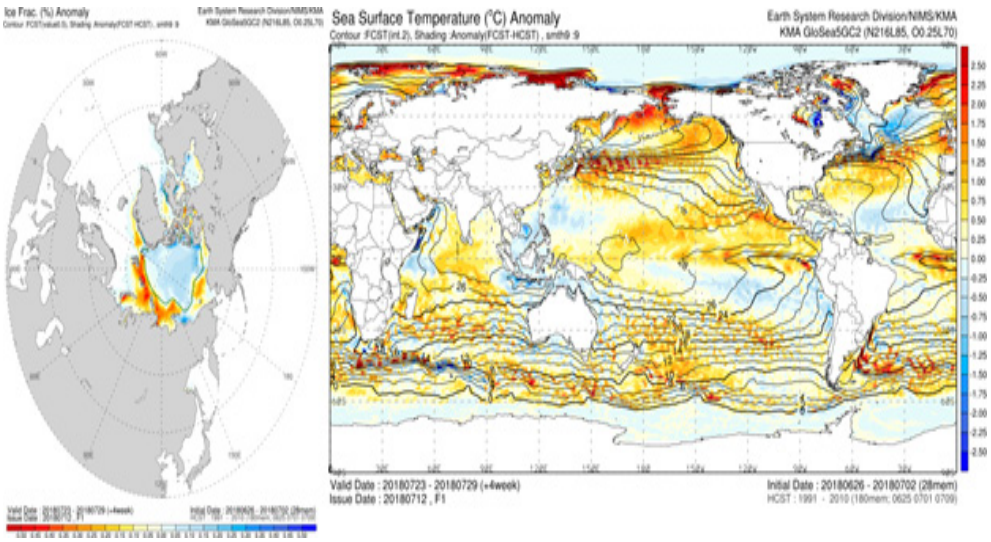
개요

자료상태 : 현행

기후예측시스템(GloSea6-GC3.2)은 대기· 지면과 해양 ·해빙이 결합된 모델로, 전지구 영역을 대상으로 1~6개월까지 장기변동 예측을 수행하는 모델입니다. 장기변동을 분석하기 위해 실시간 예측장(forecast)와 과거 재현 기후장(hindcast)를 동시에 수행합니다.

주요요소	예측장, 기후장
해 상 도	· 대기 : 60km(85층) · 해양 : 25km(75층)
발표시각	주 1회 (앙상블 예측장)
예측시간	· 1개월 예보용 : 7주(앙상블 주평균) · 3/6개월 예보용 : 6개월(앙상블 월평균)
제공기간	2022년 1월 ~ 현재 (기후장 기간 : 1993~2022년)
제공형식	GRIB2(대기변수), NetCDF(해양변수)
제공경로	공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	기상자료개방포털(http://data.kma.go.kr) → 소통과 참여 → 자료실에서 수치예보 모델자료 활용 가이던스 참조

기후예측모델(GloSea) 전지구 예측일기도(이미지)



개요

자료상태 : 현행

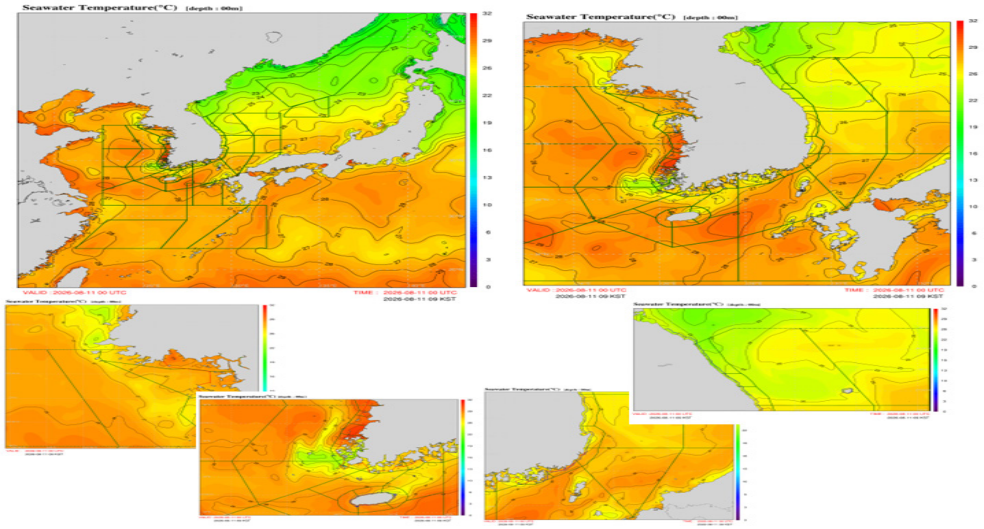
기후예측모델은 대기· 지면과 해양 ·해빙이 결합된 모델로, 전지구 영역을 대상으로 1~6개월까지 장기변동 예측을 수행하는 모델입니다. 장기변동을 분석하기 위해 실시간 예측장(forecast)와 과거 재현 기후장(hindcast)를 동시에 수행합니다.

주요요소	예측장, 기후장
해 상 도	· 대기 : 60km(85층) · 해양 : 25km(75층)
발표시각	주 1회 (앙상블 예측장)
예측시간	· 1개월 전망용 : 0~+7주(앙상블 주평균) · 3/6개월 전망용 : +1~+6개월(앙상블 월평균)
제공기간	2016년 4월 ~ 현재 (기후장 기간 : 1993~2022년)
제공형식	GRIB2(대기변수), NetCDF(해양변수)
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	모델·기상요소·고도·기준시각과 예측시간을 선택해 그림을 조회합니다. 범례와 단위, UTC 기준시각 및 유효시각을 확인합니다. 수치 계산에는 원자료 또는 수치 조회 서비스를 이용합니다. 실시간 예측(forecast)과 과거 재예측(hindcast)의 기준기간·멤버를 구분합니다.

10.해양순환모델 (MOHD)

- 지역·국지 해양순환모델 예측자료

지역·국지 해양순환모델 예측자료



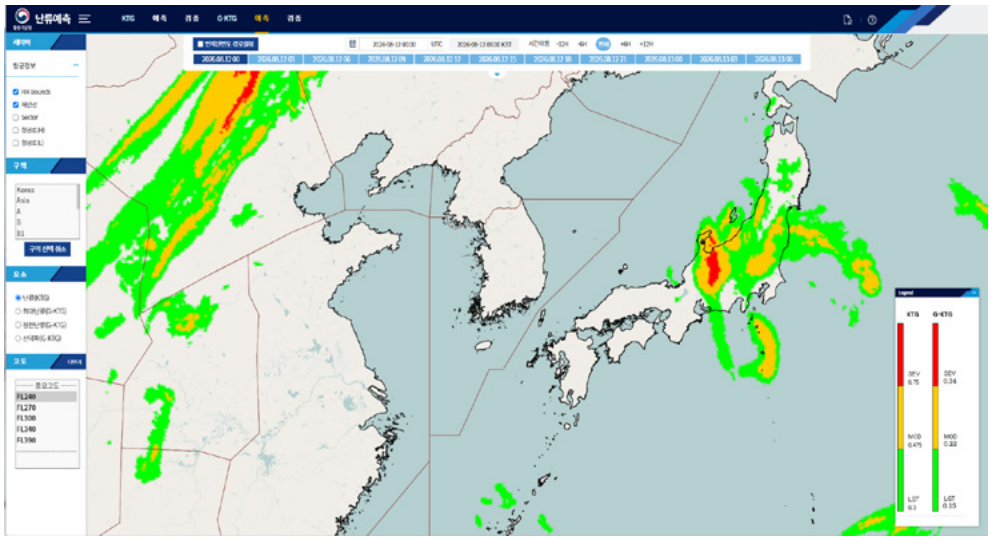
개요

자료상태 : 현행

해양순환모델로 계산한 수온·염분·해류 등의 예측자료입니다. 대상 해역별 수평격자와 수심별 연직구조를 가지며, 바다의 파고를 예측하는 파랑모델과 구분합니다.

주요요소	수온, 염분, 해류(U,V)
해 상 도	동아시아(8km), 한반도(3km), 지방청권역(500m/수도권·대전청, 광주·제주청, 부산·대구청, 강원·대구청)
발표시각	일 1회
예측시간	120시간
제공기간	2020년 1월 1일 ~ 현재
생산·갱신 주 기	일 1회(24시간 주기)
제공형식	hdf5
제공경로	요청시 ※ FTP 혹은 우편(저장매체 저장) 제공
이용방법	대상 해역·기준시각·예측시간·수심과 필요한 변수를 정하여 신청합니다. 수평·연직 좌표, 해류성분의 방향과 단위·결측값을 확인하여 이용합니다.

한국형 항공난류 예측자료



개요

자료상태 : 현행

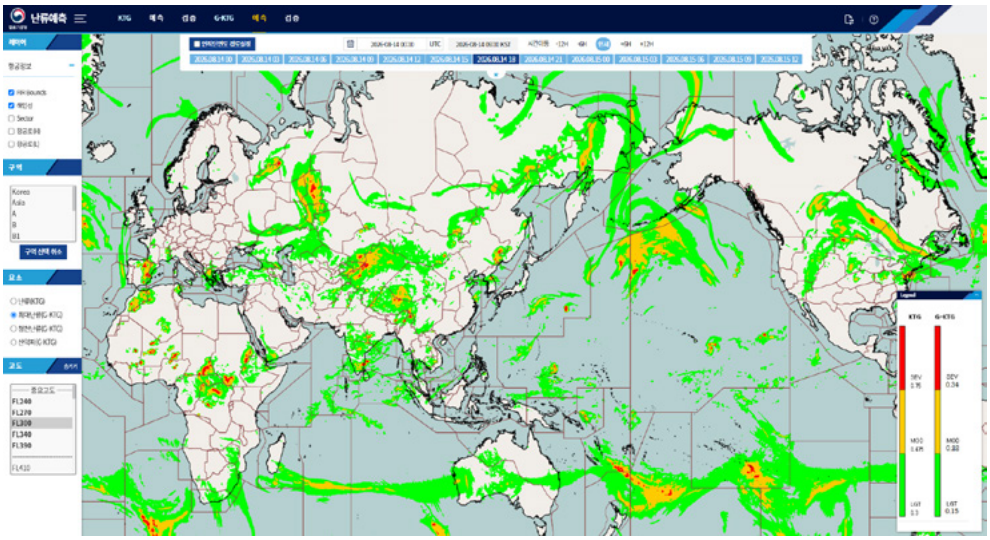
한국형 항공난류 예측모델(KTG, Korean Aviation Turbulence Guidance)은 항공기 안전운항을 위협하는 난류를 예측하기 위해 개발되었습니다.
2026년 4월 이후로는 한국형 수치예보모델(KIM)을 기반으로 여러 난류 진단지수를 산출한 뒤, 각 지수별 가중치를 적용해 0~1 사이의 단일 통합 청천난류(CAT, Clear Air Turbulence) 예측지수로 산출합니다.
공간 해상도는 수평 약 8km, 연직 42개 층(지상~41,000ft, 1,000ft 간격)으로 구성되어 있습니다.
예측 모델은 일 4회(00, 06, 12, 18 UTC) 작동하며, 3시간 간격으로 최대 30시간까지의 예측자료를 생산합니다.
이렇게 생산된 최신 일기도는 항공기상청 항공날씨 웹페이지(http://global.amo.go.kr/)를 통해 실시간으로 확인하실 수 있습니다.

주요요소	청천난류(CAT)	
제공범위	동아시아 영역	
발표시각	일 4회(00, 06, 12, 18UTC)	
예측시간	30시간 예측(3시간 간격)	
제공기간	2022년 1월 1일 ~ 현재	
제공형식	NetCDF	
제공경로	저고도 (0~10,000ft)	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
	중·고고도 (10,000ft초과)	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	항공기상청 차세대항공기상팀 문의(032-222-3066)	

11.항공모델

• 한국형 항공난류 예측자료
• 전지구 한국형 항공난류 예측자료
• 제주공항 특화 예측일기도

전지구 한국형 항공난류 예측자료



개요

자료상태 : 현행

전지구 한국형 항공난류 예측모델(GKTG, Global- Korean Aviation Turbulence Guidance)은 기존 한국형 항공난류 예측모델(KTG) 개발 이후, 전 세계적인 항공난류 예보 체계의 필요성에 따라 추가 개발되었습니다.

2026년 4월 이후로는 한국형수치예보모델(KIM)을 기반으로 여러 난류 진단지수를 계산한 뒤 가중치를 부여합니다. 이를 통해 청천난류(CAT)와 산악파난류(MWT)를 EDR 규모(0~1)의 통합 지수로 각각 산출하며, 두 값 중 최댓값인 최대난류(MAX)까지 포함해 총 3가지 난류 예측지수(CAT, MWT, MAX)를 최종 생산합니다.

공간 해상도는 수평 약 8km, 연직 40개 층(1,000ft~40,000ft, 1,000ft 간격)으로 구성되어 있습니다.

일 4회(00, 06, 12, 18 UTC) 작동하며, 3시간 간격으로 최대 36시간까지의 예측자료를 생산합니다.

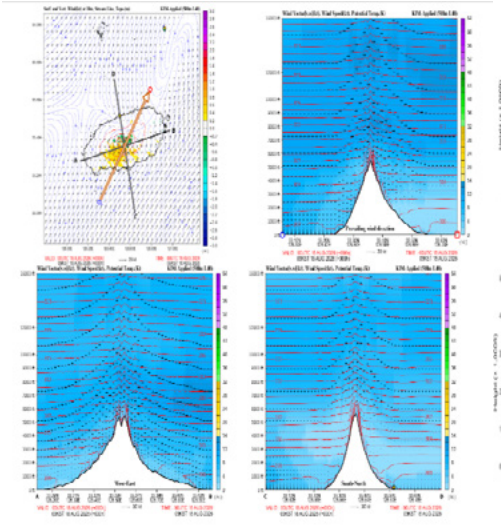
이렇게 생산된 최신 일기도는 항공기상청 항공날씨 웹페이지(http://global.amo.go.kr/)를 통해 실시간으로 확인하실 수 있습니다.

주요요소	청천난류(CAT), 산악파난류(MWT), 최대난류(MAX)
제공범위	한반도(과거 자료)
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	36시간 예측(3시간 간격)
제공기간	2022년 1월 1일 ~ 현재
제공형식	NetCDF
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	항공기상청 차세대항공기상팀 문의(032-222-3066)

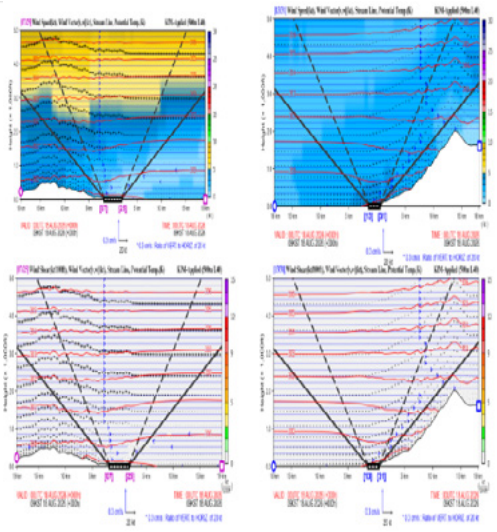
제주공항 특화 예측일기도

11.항공모델

• 한국형 항공난류 예측자료
• 전지구 한국형 항공난류 예측자료
• 제주공항 특화 예측일기도



<물뜀현상 일기도>



<연직급변풍(양배풍) 일기도>

개요

자료상태 : 현행

제주공항은 지형적 특성으로 인해 국내 공항 중 급변풍이 가장 빈번하게 발생하는 공항입니다. 이에 따라 항공기 안전운항을 지원하기 위해 고해상도 제주공항 특화 예측일기도를 개발하였습니다.
본 데이터는 고해상도 모델의 바람 성분 예측자료를 활용해 제주 물뜀현상*, 활주로 주변 풍속, 연직급변풍에 대한 연직단면도를 일기도 형태로 제공합니다.

*한라산을 넘어 활강하는 남풍 밑에서 발생하는 국지적 상승기류

공간 해상도는 수평 약 500m이며, 연직 표출 범위는 제주 물뜀현상 일기도의 경우 지상~약 5km (16,000ft), 풍속 및 연직급변풍 일기도는 지상~약 1.5km(5,000ft)입니다. 모델은 일 4회(00, 06, 12, 18 UTC) 수행되며, 1시간 간격으로 최대 48시간까지의 예측자료를 생산합니다.
이렇게 생산된 최신 일기도는 항공기상청 항공날씨 웹페이지(http://global.amo.go.kr/)를 통해 실시간으로 확인하실 수 있습니다.

주요요소	- 물뜀현상 일기도: 제주도 수평바람장, 풍향별(주풍향,동서,남북) 연직 단면도 - 연직급변풍(양배풍) 일기도: 주·보조 활주로별 풍속 및 연직급변풍의 연직단면도
제공범위	제주공항
발표시각	일 4회(00, 06, 12, 18UTC)
예측시간	48시간 예측(1시간 간격)
제공기간	2025년 1월 1일 ~ 현재
제공형식	png
제공경로	공공데이터포털(data.go.kr) 수동 제공 신청을 통해 데이터 제공 ※ 공공데이터포털 → 공공데이터 → 공공데이터 제공신청
이용방법	일기도 종류·기준시각·예측시간과 표출 고도를 선택합니다. 바람 및 급변풍의 단위·범례와 연직단면 위치를 확인하고, 실제 탐지·관측자료와 구분합니다.

07

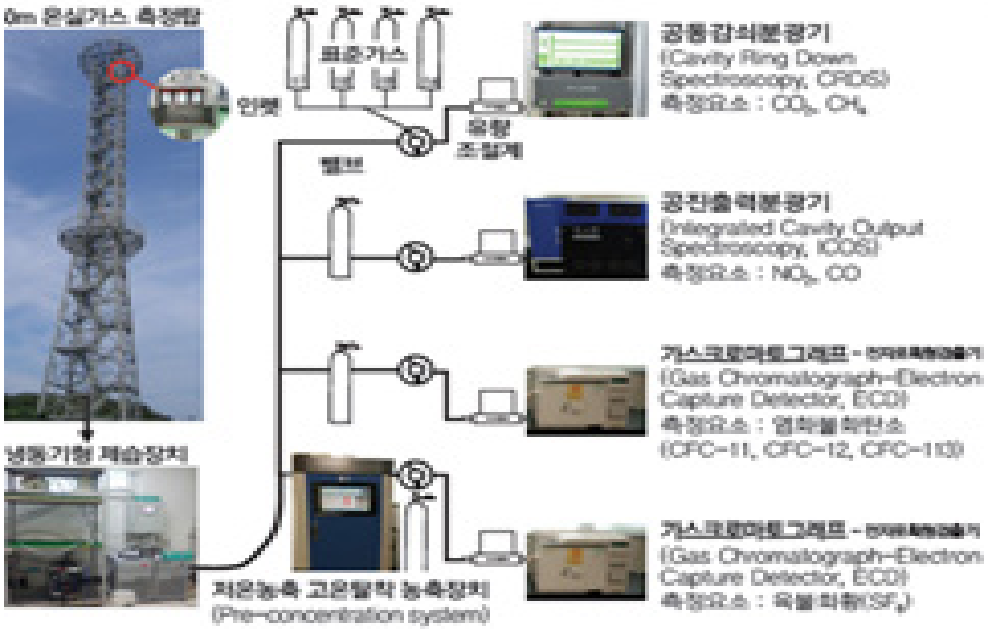
기후

Climate



1.지구대기 감시
• 온실가스
• 반응가스
• 에어로졸
• 대기복사
• 성층권오존
• 자외선
• 총대기침적
• 총자외선지수(관측)

온실가스



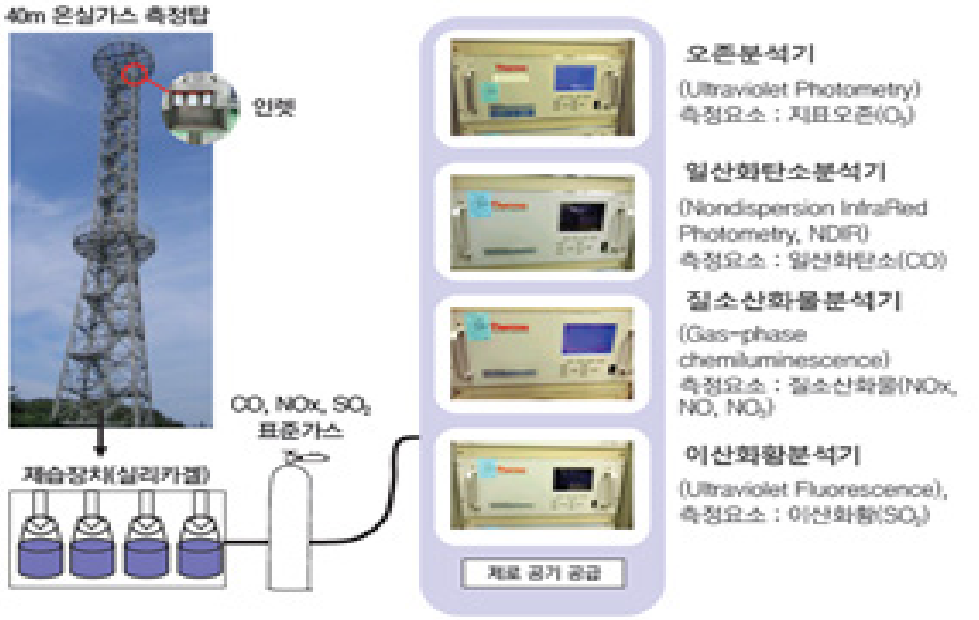
개요

자료상태 : 현행

온실가스는 기후변화의 주요 원인물질로 대기 중에 체류하면서 대부분의 태양복사를 투과시키고 지표면에서 방출되는 장파복사를 흡수하거나 재방출하여 온실효과를 일으킵니다.
대표적인 온실가스인 이산화탄소, 메탄 등 관측 자료는 온실가스가 지구온난화에 미치는 직접적인 영향을 추산하고, 배출원과 소멸원의 변화를 유추하기 위한 좋은 과학적 근거자료로 활용됩니다.

주요요소	이산화탄소(CO_2), 메탄(CH_4), 아산화질소(N_2O), 육불화황(SF_6), 염화불화탄소류(CFC-11, CFC-12, CFC-113)	
제공범위	• 안면도: 이산화탄소(CO_2), 메탄(CH_4), 아산화질소(N_2O), 육불화황(SF_6), 염화불화탄소류(CFC-11, CFC-12, CFC-113) • 제주고산: 이산화탄소(CO_2), 아산화질소(N_2O), 육불화황(SF_6) • 울릉도: 이산화탄소(CO_2), 육불화황(SF_6) • 독도: 이산화탄소(CO_2)	
제공기간	1999년 1월 ~ 현재	
생산·갱신주기	일, 월, 연 자료	
제공형식	TXT, CSV	
제공경로	TXT	기후정보포털(https://www.climate.go.kr) ※ 기후변화감시 → 기후변화감시자료 → 온실가스
	CSV	기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 온실가스(자료부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.	
간행물	지구대기감시보고서를 매년 정기적으로 발간	

반응가스



개요

자료상태 : 현행

반응가스는 지표오존, 일산화탄소, 질소산화물, 이산화황은 다른 가스 상 물질들과의 반응성이 높아 대기 중 체류시간이 수 초, 수개월 이내로 짧습니다.
지표오존은 강력한 온실기체 중 하나로 복사강제력에 직접적인 영향을 미치며, 그 외 물질들도 온실가스와 마찬가지로 대기조성비, 가스 상 물질의 대기 중 체류시간 등 지구온난화에 깊이 관여합니다.

주요요소	일산화탄소(CO), 질소산화물(NO), 지표오존(O_3), 이산화황(SO_2)	
제공범위	• 안면도, 제주고산 : CO, NO_X, O_3, SO_2 • 울릉도 : CO	
제공기간	2004년 1월 ~ 현재	
생산·갱신주기	일, 월, 연 자료	
제공형식	TXT, CSV	
제공경로	TXT	기후정보포털(https://www.climate.go.kr) ※ 기후변화감시 → 기후변화감시자료 → 반응가스
	CSV	기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 반응가스(자료부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.	
간행물	지구대기감시보고서를 매년 정기적으로 발간	

1.지구대기 감시
• 온실가스
• 반응가스
• 에어로졸
• 대기복사
• 성층권오존
• 자외선
• 총대기침적
• 총자외선지수(관측)

1.지구대기 감시
- 온실가스
- 반응가스
- 에어로졸
- 대기복사
- 성층권오존
- 자외선
- 총대기침적
- 총자외선지수(관측)

에어로졸

개요

자료상태 : 현행

에어로졸은 공기 중에 떠 있는 고체 또는 액체 상태의 입자로 보통 0.001~100μm 정도이며 황사, 화산재, 해염 같은 자연적 요인과 도시·산업시설 배출, 소각, 자동차 등의 인위적 요인에 의해 발생됩니다. 지표면으로 들어오는 태양복사에너지를 차단 또는 흡수하거나, 구름 형성과 물리특성을 변화시켜 기후변화에 영향을 미칩니다. 세계기상기구 지구대기감시프로그램 권고에 따라 다양한 장비를 통해 기후변화 원인물질 중 하나인 에어로졸의 물리·광학·화학 특성을 관측합니다.

주요요소 / 지점

요소		장비명/모델명	지점
물리특성	PM10 질량농도	부유분진측정기/ FH62C14	안면도, 제주고산, 울릉도
	크기별 수농도 (0.5~20μm)	공기역학입자계수기/ TSI3321	안면도, 제주고산, 울릉도
	미세입자 크기별수농도 (0.008~1.2μm)	전자기유도입자계수기/ TSI3034, GRIMM SMPS5.416	안면도
	응결핵 수농도 (0.01~3μm)	응결핵계수기/TSI3772	제주고산
광학특성	광산란계수	광산란계수측정기/ TSI3563	안면도
	광흡수계수	광흡수계수측정기/AE16, AE31	안면도
	광학깊이	정밀필터복사계/PFR	안면도, 제주고산, 울릉도
화학특성	이온성분	저용량미세먼지채취기/ PMS104	안면도
	원소성분		안면도

제공기간	2001년 1월 ~ 현재
생산·갱신주기	일, 월, 연 자료
제공형식	TXT, CSV
제공경로	TXT : 기후정보포털(https://www.climate.go.kr) ※ 기후변화감시 → 기후변화감시자료 → 에어로졸
	CSV : 기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 에어로졸(자료부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.
간행물	지구대기감시보고서를 매년 정기적으로 발간

대기복사



직달산란일사계 (Pyranometer)

- 측정요소 : 직달일사, 산란일사
- 측정영역 : 300~2800nm

※ 직달일사 : 대기에서 거의 산란되지 않고 직접 도달하는 태양복사
산란일사 : 대기, 구름 등에 의해 다중 산란 된 태양복사



태양복사계(Pyranometer), 지구복사계(Pyrgeometer)

- 측정요소 : 태양상하향복사, 지구상하향복사

※ 태양상향복사 : 에어로졸에 의해 산란되어 외계로 방출되는 일사량
태양하향복사 : 태양으로부터 지표에 도달하는 모든 일사량
지구상향복사 : 지표에서 반사되어 대기로 방출되는 일사량
지구하향복사 : 지구상향복사가 에어로졸에 의해 산란되어 지표에 도달하는 일사량



순복사계 (Pyranometer)

- 측정요소 : 태양 및 지구복사 수지

개요

자료상태 : 현행

대기복사는 기후시스템 내 일어나는 구름, 비와 같은 현상은 근원적으로 태양으로부터 받는 복사에너지의 양과 지구-대기권 사이의 상호작용 변동에 따라 발생하는 것으로, 대기복사수지를 감시하는 것은 기후변화를 이해하는 가장 중요한 부분입니다.
관측 자료는 기후학적으로 장기간 대기 복사량의 변화 파악과 원격탐사자료나 기후모델, 대기복사 모델의 검증 등에 활용됩니다.

주요요소	직달일사, 태양하향복사(전천일사), 태양상향복사, 산란일사, 지구상향복사, 지구하향복사, 순복사
제공범위	• 안면도 : 직달일사, 태양하향복사(전천일사), 태양상향복사, 산란일사, 지구상향복사, 지구하향복사, 순복사 • 제주고산 : 직달일사, 태양하향복사(전천일사), 산란일사
제공기간	1999년 2월 ~ 현재
생산·갱신주기	일, 월, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 대기복사
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.
간행물	지구대기감시보고서를 매년 정기적으로 발간

1.지구대기 감시
- 온실가스
- 반응가스
- 에어로졸
- 대기복사
- 성층권오존
- 자외선
- 총대기침적
- 총자외선지수(관측)

1.지구대기 감시
- 온실가스
- 반응가스
- 에어로졸
- 대기복사
- 성층권오존
- 자외선
- 총대기침적
- 총자외선지수(관측)

성층권 오존



Brewer MK III·IV
- 측정요소 : 오존전량, 오존연직분포
- 측정영역 : 286.5~363nm

개요

자료상태 : 현행

성층권 오존의 연직분포 또는 오존전량 등 오존 관측자료입니다. 오존의 농도·분압·전량은 서로 다른오존층은 성층권에는 지구 대기에 존재하는 오존의 약 90%가 존재하며, 대체로 15~30km 사이에 오존 농도가 높게 나타나는 층을 말합니다.
태양으로부터 유해 자외선 복사를 흡수하여 자외선으로부터 인간과 생태계가 유지될 수 있도록 보호하고 지구의 기후 조절에 중요한 역할을 합니다. 물리량이므로 관측방법과 요소별 단위를 확인하여 이용합니다.

주요요소	오존전량, 오존 연직분포	
제공범위	• 안면도, 제주고산, 포항: 오존전량, 오존 연직분포	
제공기간	• 오존 전량 : 1994년 ~ 현재 / (지점별 보유기간 다름) • 오존 연직분포 : 1995년 ~ 현재(지점별, 요소별 보유기간 다름)	
생산·갱신 주기	• 오존 전량: 월, 연 자료, • 오존 연직분포 : 주(2주 이내)	
제공형식	TXT, CSV	
제공경로	TXT	기후정보포털(https://www.climate.go.kr) ※ 기후변화감시 → 기후변화감시자료 → 성층권오존
	CSV	기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 성층권오존(자료부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.	
간행물	지구대기감시보고서를 매년 정기적으로 발간	

자외선



UV-Biometer 501
- 측정요소 : 자외선A, 자외선B
- 측정영역 : 자외선A(320~400nm), 자외선B(280~320nm)

개요

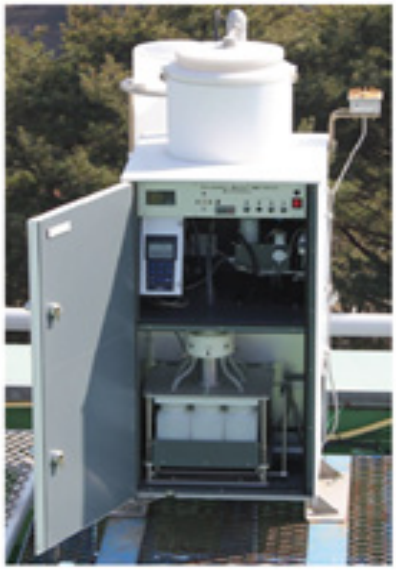
자료상태 : 현행

오존층은 성층권에는 지구 대기에 존재하는 오존의 약 90%가 존재하며, 대체로 15~30km 사이에 오존 농도가 높게 나타나는 층을 말합니다.
태양으로부터 유해 자외선 복사를 흡수하여 자외선으로부터 인간과 생태계가 유지될 수 있도록 보호하고 지구의 기후 조절에 중요한 역할을 합니다.

주요요소	자외선A, 홍반자외선B	
제공범위	안면도, 제주고산, 울릉도, 서울, 포항, 목포, 강릉	
제공기간	1999년 1월 ~ 현재	
생산·갱신주기	일, 월, 연 자료	
제공형식	TXT, CSV	
제공경로	TXT	기후정보포털(https://www.climate.go.kr) ※ 기후변화감시 → 기후변화감시자료 → 자외선
	CSV	기상자료개방포털(https://data.kma.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 자외선(자료부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.	
간행물	지구대기감시보고서를 매년 정기적으로 발간	

1.지구대기 감시
- 온실가스
- 반응가스
- 에어로졸
- 대기복사
- 성층권오존
- 자외선
- 총대기침적
- 총자외선지수(관측)

총대기침적


강수 자동 측정 시스템


이온크로마토그래프


산성도 , 전기전도도 측정기

측정요소
음이온 F^-, Cl^-, SO_4^{2-}, NO_3^-
양이온 Na^+, Ca^{2+}, Mg^{2+}, NH_4^+, K^+
산성도(pH)
전기전도도(EC)

개요

자료상태 : 현행

대기 중에 부유하는 오염물질은 비, 눈, 우박 등의 강수나 입자, 가스상 물질로 지면으로 떨어지는데 이런 강하물의 화학성분을 분석하면 대기 중의 강수성분과 오염물질을 감시할 수 있습니다.
분석된 자료는 강수성분 경향성 파악, 생태계의 산성 침적 피해 평가와 그 원인 규명을 위한 연구에 활용됩니다.

주요요소	산성도, 전기전도도, 강수이온성분 (F^-, Cl^-, SO_4^{2-}, NO_3^-, Na^+, NH_4^+, K^+, Mg^{2+}, Ca^{2+})
제공범위	안면도, 제주고산
제공기간	1997년 1월 ~ 현재
생산·갱신주기	월, 계절 자료
제공형식	CSV
제공경로	기후정보포털(https://www.climate.go.kr) ※ 데이터 → 기후 → 기후변화감시 → 총대기침적(자료 부분 제공 중)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.
간행물	지구대기감시보고서를 매년 정기적으로 발간

총자외선지수(관측)



개요

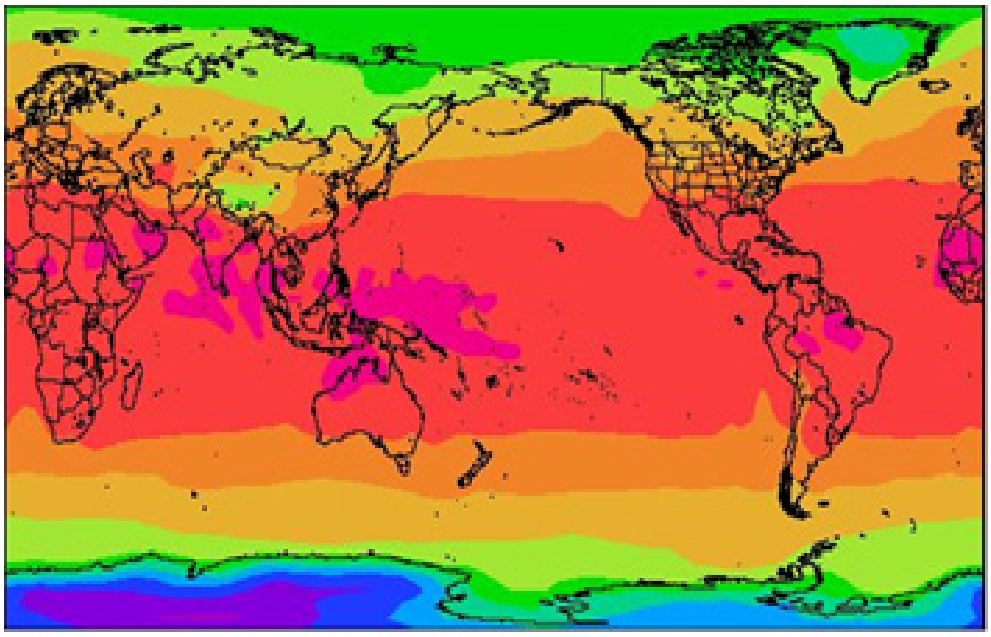
자료상태 : 현행

자외선 복사는 일반적으로 자외선A(315~400nm), 자외선B(280~315nm), 자외선C(100~280nm)로 나뉘며, 이 중 자외선B는 오존층에 일부가 흡수되고 그 나머지가 지표에 도달합니다.
특히 자외선B는 인체에 홍반을 발생시키기도 하며, 자외선B 값을 활용하여 자외선지수를 산출할 수 있습니다.

주요요소	관측된 자외선B를 활용한 자외선지수(UVI)
제공범위	안면도, 제주고산, 울릉도, 포항
제공기간	1999년 1월 ~ 현재(지점별 보유기간 다름)
생산·갱신주기	일, 월 자료
제공형식	CSV, PNG
제공경로	기후정보포털(https://www.climate.go.kr) ※ 지구대기감시 → 지구대기감시자료 → 자외선(일, 월 자료)
이용방법	관측지점·기간·요소와 시간·일·월 등의 집계 구분을 선택합니다. 요소별 단위, 시각 기준·품질표지와 잠정·확정 여부를 확인합니다. 농도·순간값·시간 평균·누적량을 구분하여 분석합니다.
간행물	지구대기감시보고서를 매년 정기적으로 발간

2.기후변화 시나리오
• SRES 시나리오
• RCP 시나리오
• SSP 시나리오

SRES 시나리오



개요

자료상태 : 현행

기후변화 시나리오는 온실가스, 에어로졸 등 인위적인 원인에 의한 기후변화가 언제, 어디서, 어떻게 일어날 지를 전망하기 위해 기후변화예측모델(지구시스템 모델)을 이용하여 계산한 미래기후(기온, 강수, 습도, 바람 등)에 대한 전망 정보를 말합니다.

SRES(Special Report on Emission Scenario)는 IPCC 4차 평가보고서 기반으로 개발된 미래 배출 시나리오 보고서로, 인위적인 기후변화 요인 중에서 온실가스와 에어로졸의 영향에 의한 강제력을 포함한 온실가스 시나리오입니다.

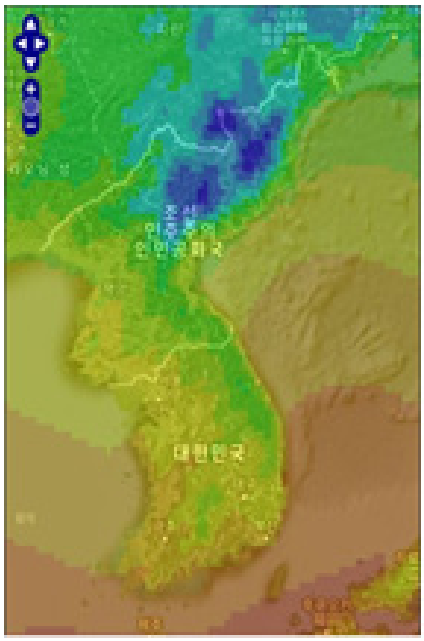
※ 시나리오 구분 별첨

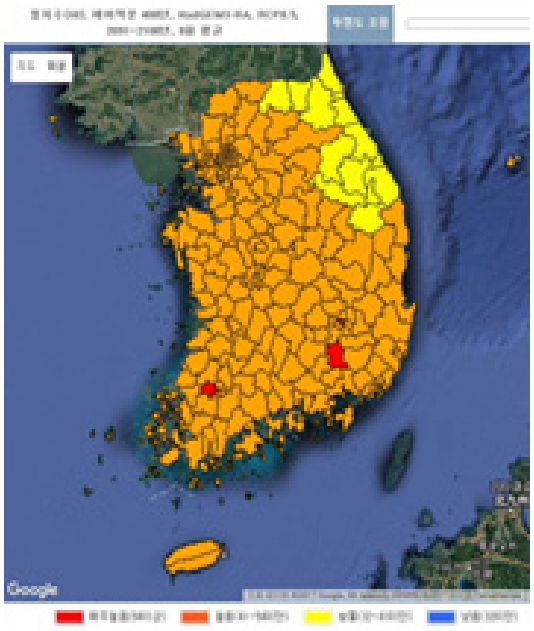
제공형식	BIN, ESRI ASCII GRID, ASCII, CSV
제공경로	기후정보포털(climate.go.kr) → 국가 기후변화 표준 시나리오 → 다운로드 → 데이터
이용방법	1. API 신청서 양식 다운로드 후 작성 2. 기후위기협력팀 이메일(kmaccc@korea.kr)로 신청서 제출 3. 인증키와 활용방법 회신 후 해당 API 링크를 통해 다운로드(2주 소요)

[SRES 시나리오 구분]

구분	시나리오	공간범위	시간범위	공간 해상도	시간 해상도	정보종류
전지구	20C3M, A1B, B1, A2	경도: 0~360 위도:-87~87	1860~2100년	400km	월	기온(최고, 최저, 평균), 강수, 상대습도
한반도	A1B	경도: 122~131 위도: 32~43	1971~2100년	27km	일, 월	기온(최고, 최저, 평균), 강수, 상대습도
남한상세	A1B	남한 지역	2000~2100년	10km	일, 월	기온(최고, 최저, 평균), 강수
극한기후지수	A1B	남한 지역	2000~2100년	10km	월, 년	기온지수(11종), 강수량지수(9종)
응용기후자료 (농림분야)	A1B	남한, 북한지역	1971~2100년 (10년 단위)	30m	월	기온(최고, 최저), 이차기후도, 분석기후도

RCP 시나리오





개요

자료상태 : 현행

RCP(Representative Concentration Pathways, 대표농도경로)시나리오는 인위적인 기후변화요인 중에서 온실가스와 에어로졸의 영향에 의한 강제력, 토지이용변화에 따른 영향까지 포함한 온실가스 시나리오입니다.
RCP시나리오는 IPCC 5차 평가보고서에 사용됐으며, 온실가스 농도 변화 경향을 반영하여 RCP 2.6, 4.5, 6.0, 8.5의 총 4가지로 분류하였습니다.
분야별 기후변화 응용정보는 기후변화 영향평가 및 적응대책 수립 지원, 각종 기후변화 연구지원을 위해 적응대책 분야별 응용정보를 생산·제공하고 있습니다.
보건·방재·농업·수자원 등의 분야별 응용정보를 생산해서 제공하고 있습니다.
※ 시나리오 구분 별첨

제공형식	BIN, ESRI ASCII GRID, ASCII, CSV
제공경로	기후변화 상황지도(www.climate.go.kr/atlas)→적응대책 분석지원 → 데이터 다운로드
이용방법	1. 기후변화 상황지도 로그인 후 데이터 다운로드 메뉴 접속 2. 원하는 시나리오 경로, 해상도, 자료형식, 자료종류 등 선택 3. 원하는 시나리오 요소 클릭 후 다운로드 클릭 4. 소속, 활용목적, 활용분야 등 선택 후 다운로드 완료 ※ 자세한 방법은 기후정보포털(climate.go.kr) → 국가 기후변화 표준 시나리오 → 다운로드 → 데이터 메뉴에서 확인

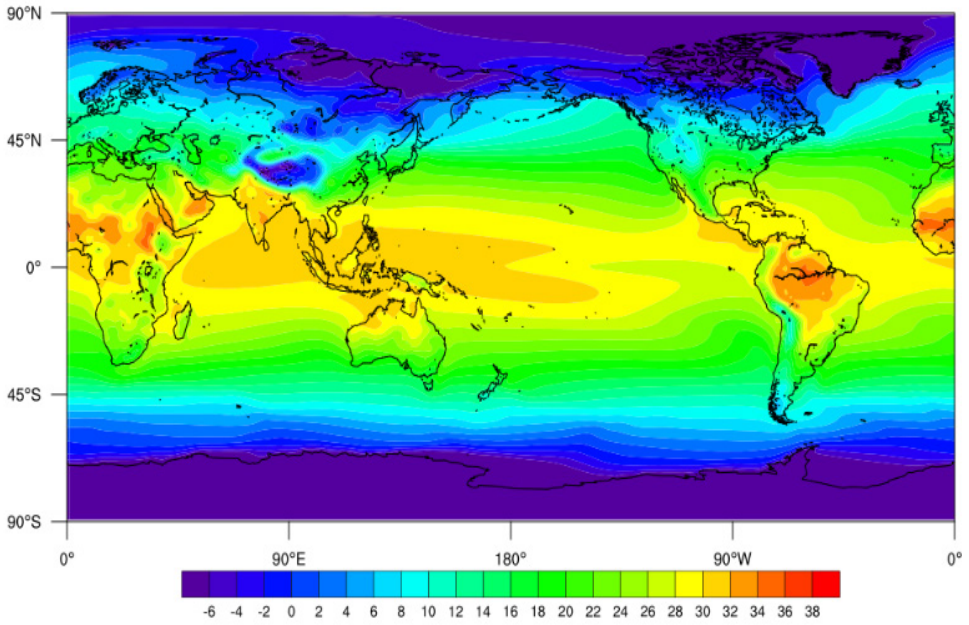
2.기후변화 시나리오

- SRES 시나리오
- RCP 시나리오
- SSP 시나리오

[RCP 시나리오 구분]						
구분	시나리오	공간범위	시간범위	공간 해상도	시간 해상도	정보종류
전지구	RCP(4종) 2.6/4.5/6.0/8.5	경도: 0~360 위도:-90~90	1860~2100년	135km	월	기온(최고,최저, 평균), 강수, 상대습도, 풍속
한반도	RCP(4종) 2.6/4.5/6.0/8.5	경도: 111~144 위도:26.5~48.9	1950~2100년	12.5km	일, 월	기온(최고,최저,평균), 강수, 상대습도, 풍속
남한상세	RCP(4종) 2.6/4.5/6.0/8.5	남한지역	2011~2100년	1km	일, 월	기온(최고,최저,평균), 강수
남한상세 (앙상블)	RCP(4종) 2.6/4.5/6.0/8.5	남한지역	2021~2100년	1km	일, 월	기온(최고,최저,평균), 강수
기후 (극한지수)	RCP(4종) 2.6/4.5/6.0/8.5	남한지역	2011~2100년	1km	년	기온극한지수(열대야, 폭염, 한파, 서리일수, 식물성장기간, 일교차, 온난일, 한랭야 등 16종) 강수극한지수(호우일수, 강수강도 등 4종)
행정구역별 자료	RCP(4종) 2.6/4.5/6.0/8.5	73개 지점 17개광역시도 230개시구군	기온, 강수량 : 2000~2100년 상대습도 등 : 1950~2100년	-	일, 월	기온(최고,최저, 평균), 강수, 상대습도, 현지·해면기압, 평균·최대풍속, 기온 및 강수극한지수

구분	수자원	보건	농업	방재	산림	동물생태
시나리오	RCP(4종)					
생산요소	중권역별 강수량, 유출량, 잠재증발산량, 유황분석정보	열지수, 불쾌지수, 체감온도(AT), 체감추위지수, 열체감지수, Net Effective Temperature (NET), 열사병발생 위험지수, 온열지수 (WBGT)	생육온도일수, 유효적산온도, 식물기간, 작물기간, 무상기간, 작물 저온요구도, 온습도지수, 기후생산력지수, 난방도일, 냉방도일, 기준증발산량, 엽면수분지속기간	표준강수지수 (SPI), 독립호우사상	최저기온지수, 유효강우지수, 건조강도지수	물새류월동 환경지수, 기후변화 심각도지수, 강우열량지수
시간범위	2000~2100년	1950~2005년 2013~2100년	1971~2100년 2011~2100년	2012~2100년	2011~2100년	2011~2100년
공간 해상도	중권역(104개), 73개지점	230개 시군구	230개 시군구, 격자(12.5km)	73개 지점	230개 시군구	230개 시군구
시간 해상도	일, 월	월	순, 10년평균	월	연	연

SSP 시나리오



개요

자료상태 : 현행

SSP(Shared Socio-economic Pathways, 공통사회 경제경로) 시나리오는 IPCC 제6차평가보고서를 위해 2100년 기준 복사강제력 강도와 함께 미래 기후변화 대비 수준에 따라 인구, 경제, 토지이용, 에너지사용 등의 미래 사회경제시스템의 변화를 적용한 시나리오입니다.
해상도에 따라 전지구(135km), 동아시아(25km), 남한상세(1km) 시나리오로 나뉘며 SSP1-2.6, SSP2-4.5, SSP3-7.0, SSP5-8.5의 총 4가지 경로를 제공 중입니다.
각 시나리오 경로별로 기온, 강수량 등 기후요소와 폭염일수, 호우일수 등 극한기후지수, 냉·난방도일, 체감온도 등 분야별 응용정보를 생산 및 제공하고 있습니다.
※ 시나리오 구분 별첨

제공형식	ASCII, NetCDF
제공경로	기후변화 상황지도(www.climate.go.kr/atlas)→적응대책 분석지원→데이터 다운로드
이용방법	기후변화 상황지도 로그인 후 데이터 다운로드 메뉴 접속 2. 원하는 시나리오 경로, 해상도, 자료형식, 자료종류 등 선택 3. 원하는 시나리오 요소 클릭 후 다운로드 클릭 4. 소속, 활용목적, 활용분야 등 선택 후 다운로드 완료 ※ 자세한 방법은 기후정보포털(climate.go.kr) → 국가 기후변화 표준 시나리오 → 다운로드 → 데이터 메뉴에서 확인

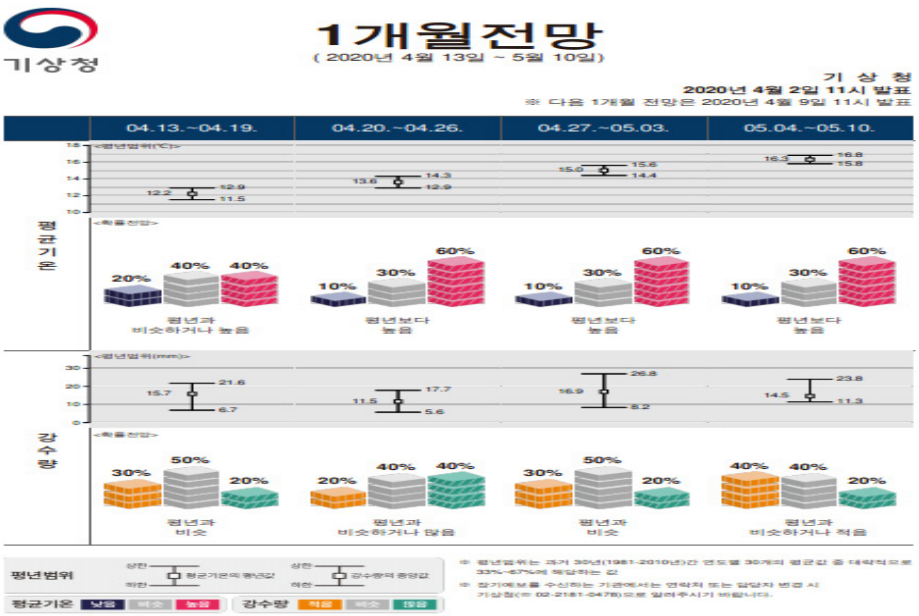
2.기후변화 시나리오

- SRES 시나리오
- RCP 시나리오
- SSP 시나리오

[SSP 시나리오 구분]				
	전지구(135km)	동아시아(대기/25km)	한반도(해양/8km)	남한상세(1km)
공간범위	[경도] 0°~360° [위도] -90°~90°	[경도] 55°~180° [위도] 0°~60°	[경도] 117°~159.6° [위도] 20°~53.2°	[경도] 124.5°~132° [위도] 33°~39°
시간범위	[과거] 1850~2014년 [미래] 2015~2100년	[과거] 1979~2014년 [미래] 2015~2100년	[과거] 1995~2014년 [미래] 2015~2100년	[과거] 2000~2019년 [미래] 2021~2100년

[SSP 시나리오 제공요소]	
영역(해상도)	기후요소
전지구 시나리오 (135km)	(대기) 평균기온, 평균최고기온, 평균최저기온, 강수량, 상대습도, 풍속 (해양) 해수면온도, 해수면높이, 표층염분, 해빙면적
동아시아·한반도 시나리오 (25km / 8km)	(대기) 평균기온, 평균최고기온, 평균최저기온, 강수량, 상대습도, 풍속, 최대풍속, 적설량 (해양) 해수면온도, 해수면높이, 표층염분
남한상세 시나리오(1km)	평균기온, 평균최고기온, 평균최저기온, 강수량, 상대습도, 풍속, 일사량
구분	**극한기후지수**
기온 극한기후지수	폭염일수, 열대야일수, 서리일수, 한파일수 등
강수 극한기후지수	호우일수, 강수강도, 1일최다강수량 등
분야	**응용정보 종류**
농업	생육온도일수, 식물기간, 작물기간, 재배적지 등
산림	최저기온지수, 건조지수, 유효강우지수 등
동물생태	물새류월동환경지수, 기후변화심각도지수 등
보건	열지수, 불쾌지수, 체감온도, 감각온도 등
방재	표준강수지수, 독립호우사상지수, 표준강수증발산지수 등
수자원	잠재증발산량 등
에너지	태양광발전잠재지수 등

1개월 전망



개요

자료상태 : 현행

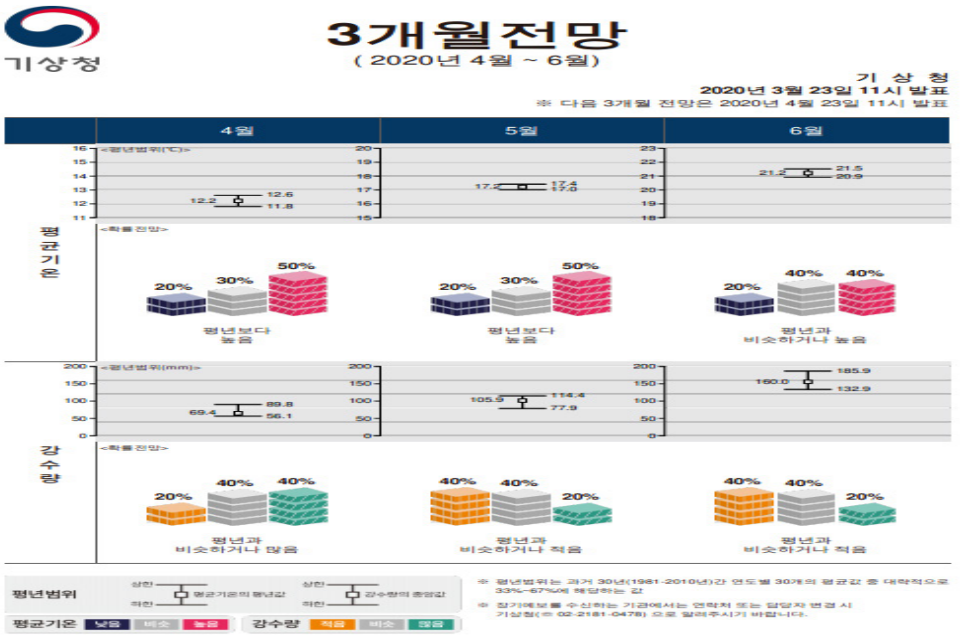
매주 목요일 발표하며, 발표일이 속한 다음 두번째 주(월~일요일)부터 다섯 번째 주까지 4주간에 대해 주별기온, 강수량 전망과 그 밖의 1개월전망에 필요한 사항을 예보합니다.

주요요소	주별 확률(기온, 강수량), 전망 요약, 주별 날씨전망, 주·지역별 확률전망(기온, 강수량)
제공범위	전국 및 한반도 12개 권역 ※ 전국, 서울·인천·경기도, 강원도 영서, 강원도 영동, 대전·세종·충청남도, 충청북도, 광주·전라남도, 전북특별자치도, 부산·울산·경상남도, 대구·경상북도, 제주도, 평안남북도·황해도, 함경남북도
발표시각	매주 목요일 11시
예측시간	발표일이 속한 주의 다음 두 번째 주부터 다섯 번째 주까지 4주간의 주별 전망
제공기간	2010년 7월 ~ 현재(PDF), 2015년 12월~ 현재(XML)
생산·갱신주기	매주 1회
제공형식	PDF: 2010년 7월 ~ 현재 XML: 2015년 12월 ~ 현재
제공경로	기상청 날씨누리(http://www.kma.go.kr) → 기후·기후변화 → 기후예측 → 1개월 전망
이용방법	발표일과 전망 대상기간을 선택하여 원문과 부속자료를 확인합니다. 평년 대비 높음·비슷·낮음 등의 전망과 확률을 구분하고, 특정 날짜의 날씨를 확정한 예보로 해석하지 않습니다.

3.기후예측

• 1개월 전망
• 3개월 전망
• 기후전망

3개월 전망



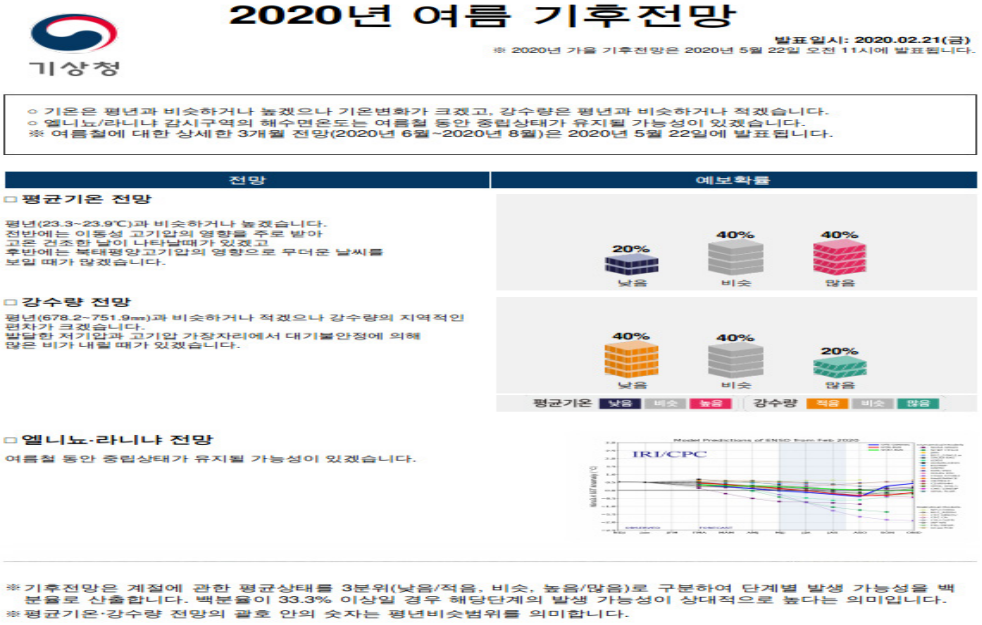
개요

자료상태 : 현행

매월 23일경 발표하며, 발표일이 속한 월의 다음월부터 3개월 동안의 기간에 대해 기압계 동향 및 전망, 월별 기온 및 강수량 전망과 그 밖의 3개월 전망에 필요한 사항을 예보합니다.

주요요소	월별 확률(기온, 강수량), 전망 요약, 월별 날씨전망, 월·지역별 확률전망(기온, 강수량)
제공범위	전국 및 한반도 12개 권역 ※ 전국, 서울·인천·경기도, 강원도 영서, 강원도 영동, 대전·세종·충청남도, 충청북도, 광주·전라남도, 전북특별자치도, 부산·울산·경상남도, 대구·경상북도, 제주도, 평안남북도·황해도, 함경남북도
발표시각	매월 23일 경 11시
예측시간	다음 월부터 3개월간의 월별 전망
제공기간	2010년 7월 ~ 현재(PDF), 2015년 12월~ 현재(XML)
생산·갱신주기	매월 1회
제공형식	PDF: 2010년 7월 ~ 현재 XML: 2015년 12월 ~ 현재
제공경로	기상청 날씨누리(http://www.kma.go.kr) → 기후·기후변화 → 기후예측 → 3개월 전망
이용방법	발표일과 전망 대상기간을 선택하여 원문과 부속자료를 확인합니다. 평년 대비 높음·비슷·낮음 등의 전망과 확률을 구분하고, 특정 날짜의 날씨를 확정한 예보로 해석하지 않습니다.

기후전망



2020년 여름 기후전망

기상청

발표일시: 2020.02.21(금)

※ 2020년 가을 기후전망은 2020년 5월 22일 오전 11시에 발표됩니다.

○ 기온은 평년과 비슷하거나 높겠으나 기온변화가 크겠고, 강수량은 평년과 비슷하거나 적겠습니다.
○ 엘니뇨/라니냐 감시구역의 해수면온도는 여름철 동안 중립상태가 유지될 가능성이 있겠습니다.
※ 여름철에 대한 상세한 3개월 전망(2020년 6월~2020년 8월)은 2020년 5월 22일에 발표됩니다.

전망	예보확률
□ 평균기온 전망 평년(23.3~23.9℃)과 비슷하거나 높겠습니다. 전반에는 이동성 고기압의 영향을 주로 받아 고온 건조한 날이 나타날때가 있겠고 후반에는 북태평양고기압의 영향으로 무더운 날씨를 보일 때가 많겠습니다.	20% 낮음 / 40% 비슷 / 40% 많음
□ 강수량 전망 평년(678.2~751.9㎜)과 비슷하거나 적겠으나 강수량의 지역적인 편차가 크겠습니다. 발달한 저기압과 고기압 가장자리에서 대기불안정에 의해 많은 비가 내릴 때가 있겠습니다.	40% 낮음 / 40% 비슷 / 20% 많음 평균기온 낮음 비슷 높음 강수량 적음 비슷 많음
□ 엘니뇨·라니냐 전망 여름철 동안 중립상태가 유지될 가능성이 있겠습니다.	IRI/CPC

※기후전망은 계절에 관한 평균상태를 3분위(낮음/적음, 비슷, 높음/많음)로 구분하여 단계별 발생 가능성을 백분율로 산출합니다. 백분율이 33.3% 이상일 경우 해당단계의 발생 가능성이 상대적으로 높다는 의미입니다.
※평균기온·강수량 전망의 괄호 안의 숫자는 평년비슷범위를 의미합니다.

개요

자료상태 : 현행

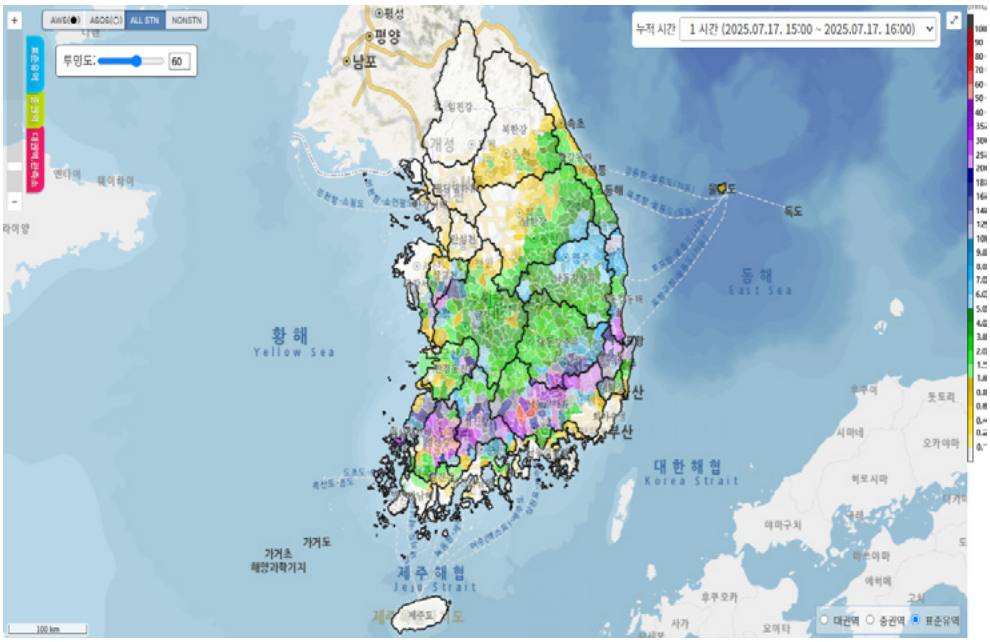
기후전망은 다음다음 계절 또는 다음 해에 대한 평균기온, 강수량, 엘니뇨 · 라니냐 전망을 말하며, 계절 기후전망은 연 4회, 연기후 전망은 1년에 한번 발표합니다.

주요요소	평균기온, 강수량, 엘니뇨·라니냐 전망
제공범위	전국(우리나라 전역)
발표시각	· 계절 기후전망 : 연 4회 발표, 발표일 다음다음 계절에 대한 기후전망 ※ 발표 날짜 봄철 : 11월 23일경 여름철 : 2월 23일경 가을철 : 5월 23일경 겨울철 : 8월 23일경 · 연기후 전망 : 연 1회 발표, 다음 해에 대한 기후전망 ※ 발표 날짜 : 2월 7일경
예측시간	(계절기후전망) 발표일 기준 이후 다음다음 계절 전망 (연기후전망) 발표일 기준 다음 해 전망
제공기간	2008년 5월(2008년 가을철 기후전망) ~ 현재
생산·갱신주기	(계절 기후전망) 매년 4회 / (연기후 전망) 매년 1회
제공형식	PDF
제공경로	기상청 날씨누리(http://www.kma.go.kr) → 기후·기후변화 → 기후예측 → 계절 기후 전망(연기후 전망)
이용방법	발표일과 전망 대상기간을 선택하여 원문과 부속자료를 확인합니다. 평년 대비 높음·비슷·낮음 등의 전망과 확률을 구분하고, 특정 날짜의 날씨를 확정한 예보로 해석하지 않습니다.

4.수문기상

- 유역별 강수량
- 유역별 증발산량
- 유역별 토양수분
- 유역별 강수량 예측정보
- 토양수분량·증발산량 예측정보

유역별 강수량



개요

자료상태 : 현행

지점별 관측자료를 기반으로 유역단위 매시간 면적강수량을 산출한 자료입니다.

주요요소	강수량
제공범위	대권역, 중권역, 표준유역
발표시각	매시간
제공기간	2017년 12월 ~ 현재
생산·갱신주기	일 24회
제공형식	GIS 분포도, 표, 시계열
제공경로	기상청 수문기상가뭄정보시스템(https://hydro.kma.go.kr) → 현황 → 강수량 → 유역별 면적강수량
이용방법	날짜 및 시각을 선택 후 유역 단위(대권역, 중권역, 표준유역)를 선택합니다. 선택한 유역을 마우스로 클릭하면 해당 유역의 면적강수량을 확인합니다.

유역별 증발산량



개요

자료상태 : 현행

천리안위성 2A호 산출물을 유역 경계와 중첩하여 중앙격자점 산술평균법으로 유역별 증발산량을 산출한 자료입니다.

주요요소	유역별 증발산량
제공범위	행정구역 167개 시·군, 대권역 26개, 중권역 117개, 표준유역 850개
발표시각	매일 14시 30분경(전일자료)
제공기간	최근 1년 보유
생산·갱신주기	1일 1회
제공형식	GIS 분포도, 표, 시계열
제공경로	기상청 수문기상가뭄정보시스템(https://hydro.kma.go.kr) → 현황 → 증발량/증발산량 → 유역별(GIS) 증발산량
이용방법	유역 또는 지역·기간·요소를 선택합니다. 유역 경계와 집계기간·단위·시각을 확인하고 관측 기반 현황과 예측정보를 구분하여 이용합니다. 증발량·증발산량과 실제·잠재 증발산량의 구분을 확인합니다.

4.수문기상

- 유역별 강수량
- 유역별 증발산량
- 유역별 토양수분
- 유역별 강수량 예측정보
- 토양수분량·증발산량 예측정보

유역별 토양수분



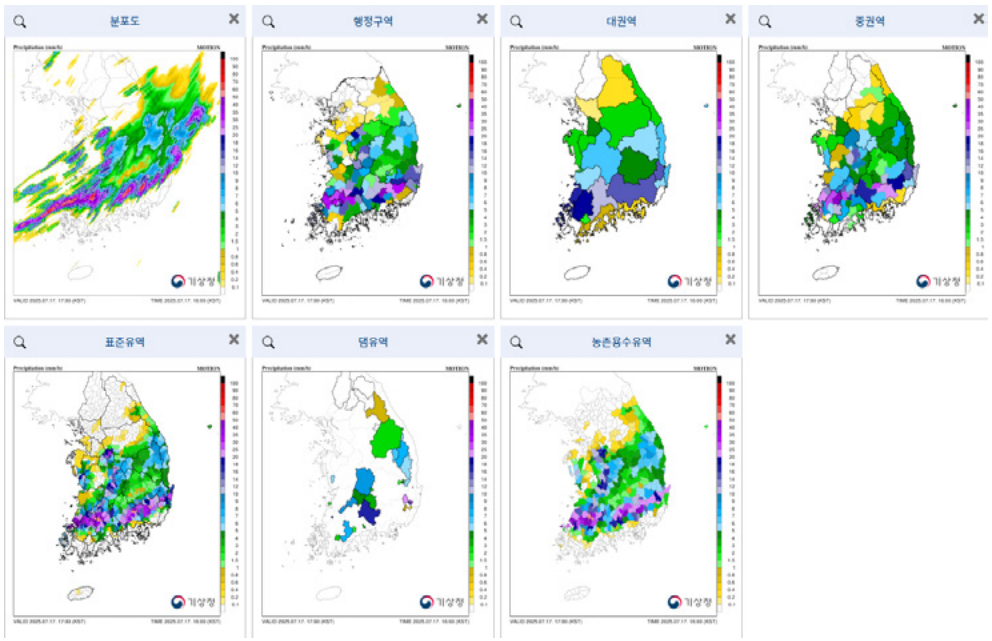
개요

자료상태 : 현행

천리안위성 2A호 산출물을 유역 경계와 중첩하여 중앙격자점 산술평균법으로 유역별 토양수분을 산출한 자료입니다.

주요요소	유역별 토양수분
제공범위	행정구역 167개 시·군, 대권역 26개, 중권역 117개, 표준유역 850개
발표시각	매일 14시 30분경(전일자료)
제공기간	최근 1년 보유
생산·갱신주기	1일 1회
제공형식	GIS 분포도, 표, 시계열
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/obsStat/obsHydroData.do) → 현황 → 토양수분 → 유역별(GIS) 토양수분량
이용방법	유역 또는 지역·기간·요소를 선택합니다. 유역 경계와 집계기간·단위·시각을 확인하고 관측 기반 현황과 예측정보를 구분하여 이용합니다. 토양수분의 단위와 대표 깊이를 확인합니다.

유역별 강수량 예측정보



개요

자료상태 : 현행

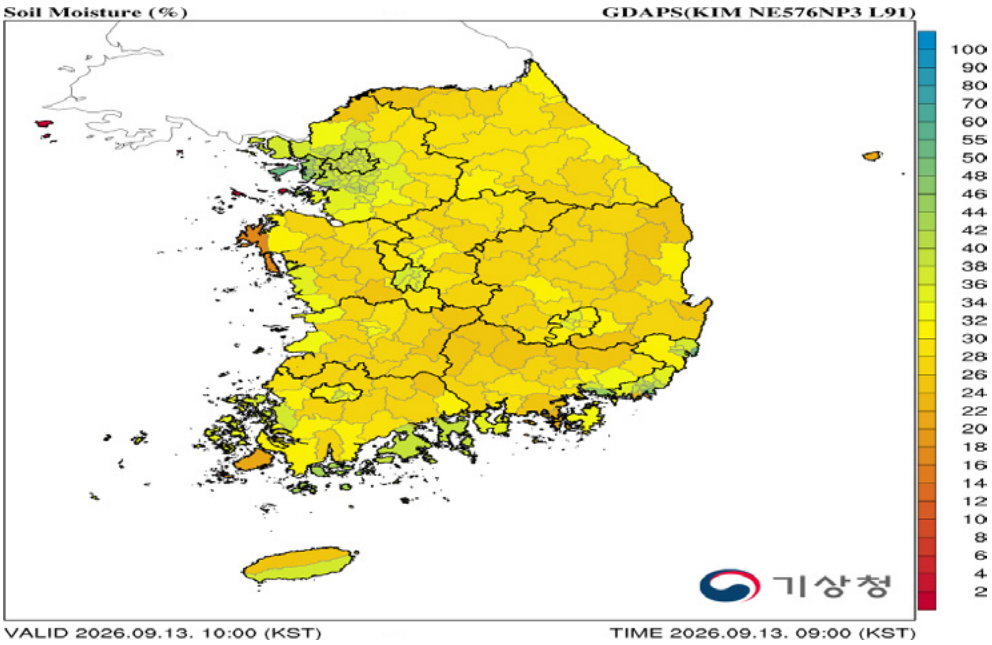
수치예보모델 및 단기예보 기반으로 유역별 면적강수량 예측정보입니다.

주요요소	강수량
제공범위	행정구역, 대권역, 중권역, 표준유역, 댐유역, 농촌용수유역, 저수지유역
발표시각	매시(일 24회)/09시,21시(일 2회)/02, 05, 08, 11, 14, 17, 20, 23시(일 8회) ※ 모델별 발표시각 상이
예측시간	최대 12일
제공기간	2022년 1월 ~ 현재
생산·갱신주기	1시간/3시간/12시간(모델별 주기 상이)
제공형식	분포도, 표, 시계열
제공경로	기상청 수문기상가뭄정보시스템(https://hydro.kma.go.kr) → 예측 → 수문기상 → 단일수치예보모델, 단기예보
이용방법	날짜 및 시각을 선택 후 유역 분포도에서 해당 유역(지역)을 마우스로 클릭하면 해당 유역(지역)의 예측 면적강수량을 확인합니다.

4.수문기상

- 유역별 강수량
- 유역별 증발산량
- 유역별 토양수분
- 유역별 강수량 예측정보
- 토양수분량·증발산량 예측정보

토양수분량·증발산량 예측정보



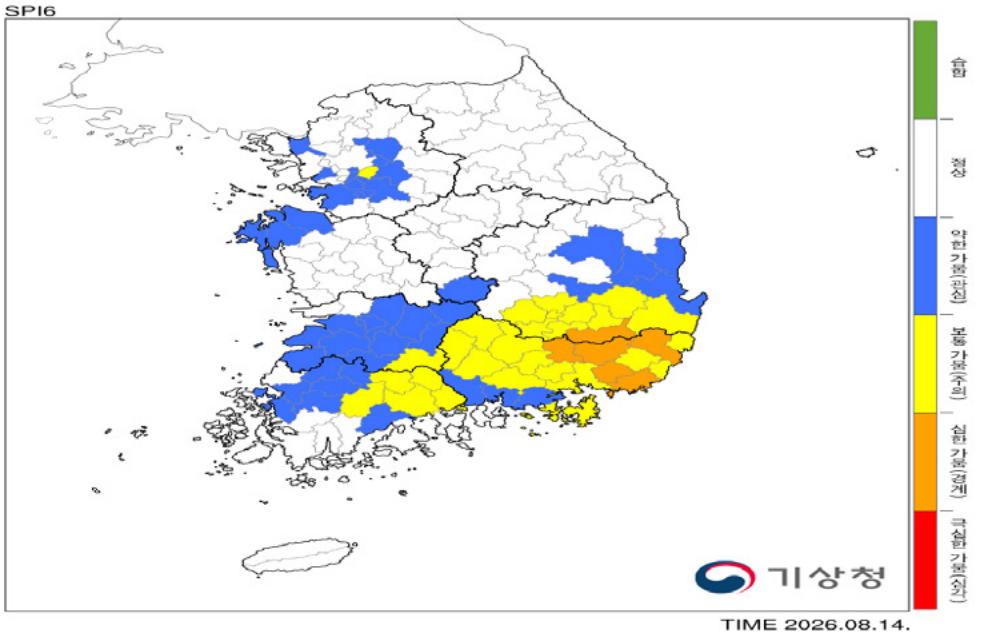
개요

자료상태 : 현행

홍수 등 수문기상 재해 대응과 물관리 관계기관 지원을 위해 한국형 전지구 수치예보모델(KIM) 자료를 유역별로 가공한 토양수분량·증발산량 예측정보입니다.

주요요소	토양수분량, 증발산량
제공범위	행정구역, 대권역·중권역·표준유역, 댐유역, 농촌용수유역
발표시각	매일 09시, 21시(1일 2회)
예측시간	최대 10일
제공기간	2026년 1월 ~ 현재
생산·갱신주기	1일 2회
제공형식	분포도, 표, 시계열
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/hydro/hydroFcstKim.do?referenceSet=prcp) → 로그인 후 예측 → 수문기상 → 단일 수치예보모델 → KIM 전지구모델 → 토양수분량/증발산량
이용방법	유역 또는 지역·기간·요소를 선택합니다. 유역 경계와 집계기간·단위·시각을 확인하고 관측 기반 현황과 예측정보를 구분하여 이용합니다. 로그인 또는 이용권한이 필요한 자료는 신청·승인 절차를 확인합니다.

표준강수지수



개요

자료상태 : 현행

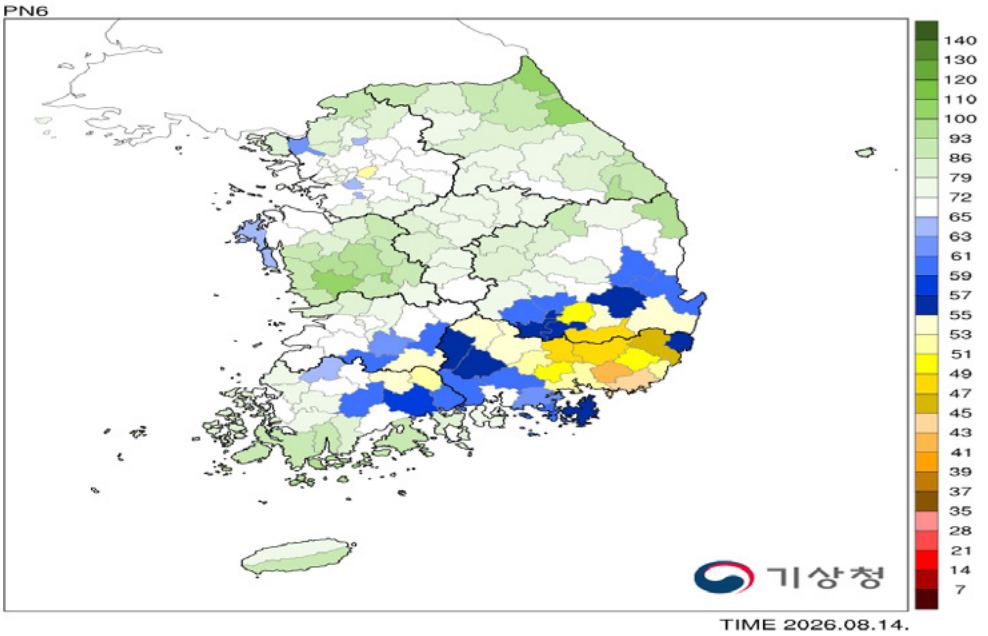
기상가뭄의 정도를 나타내기 위해 최근 누적강수량을 과거 동일 기간의 누적강수량 분포와 비교하여 표준화한 기상가뭄 지수입니다.

주요요소	SPI1~6·9·12·18·24, 분포도, 시계열
제공범위	행정구역 167개 시·군, 유역 117개 중권역
발표시각	매일 16시경(전일자료 반영)
제공기간	2016년 1월 ~ 현재
생산·갱신주기	1일 1회
제공형식	GIS 분포도, 표, 시계열(PNG 저장 가능)
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/obse/prcpt_idx.do) → 현황 → 기상가뭄 → 표준강수지수
이용방법	지역·기준일·누적기간(예: 1·3·6개월)을 선택합니다. SPI의 기준기간과 가뭄 단계표를 확인하고 같은 누적기간끼리 비교합니다. 토양수분량 자체를 나타내는 값으로 해석하지 않습니다.

5.가뭄정보

- 표준강수지수
- 강수평년비
- 기상가뭄예보
- 표준강수증발산지수

강수평년비



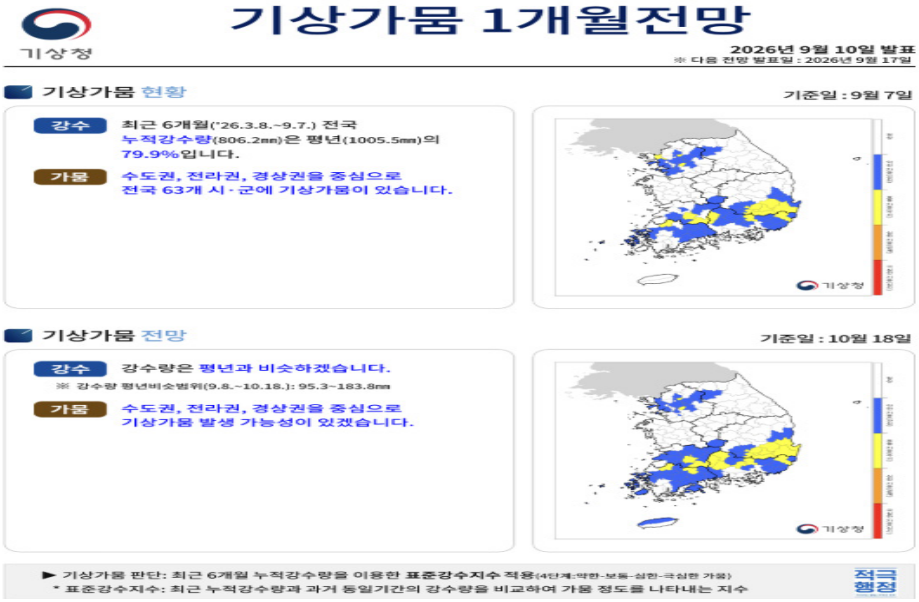
개요

자료상태 : 현행

평년 대비 강수의 부족·과다 정도를 나타내기 위해 일정 기간의 누적강수량을 같은 기간의 평년 누적강수량으로 나눈 비율입니다.

주요요소	PN1~6·9·12·18·24, 분포도, 시계열
제공범위	행정구역 167개 시·군, 유역 117개 중권역
발표시각	매일 16시경(전일자료 반영)
제공기간	2017년 12월 ~ 현재
생산·갱신주기	1일 1회
제공형식	GIS 분포도, 표, 시계열(PNG 저장 가능)
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/obse/prcpt_nmyr_idx.do) → 현황 → 기상가뭄 → 강수평년비
이용방법	지역·누적기간과 평년 기준기간을 선택합니다. 강수평년비의 단위(%)를 확인하고 같은 기간의 평년강수량을 기준으로 해석합니다.

기상가뭄예보



개요

자료상태 : 현행

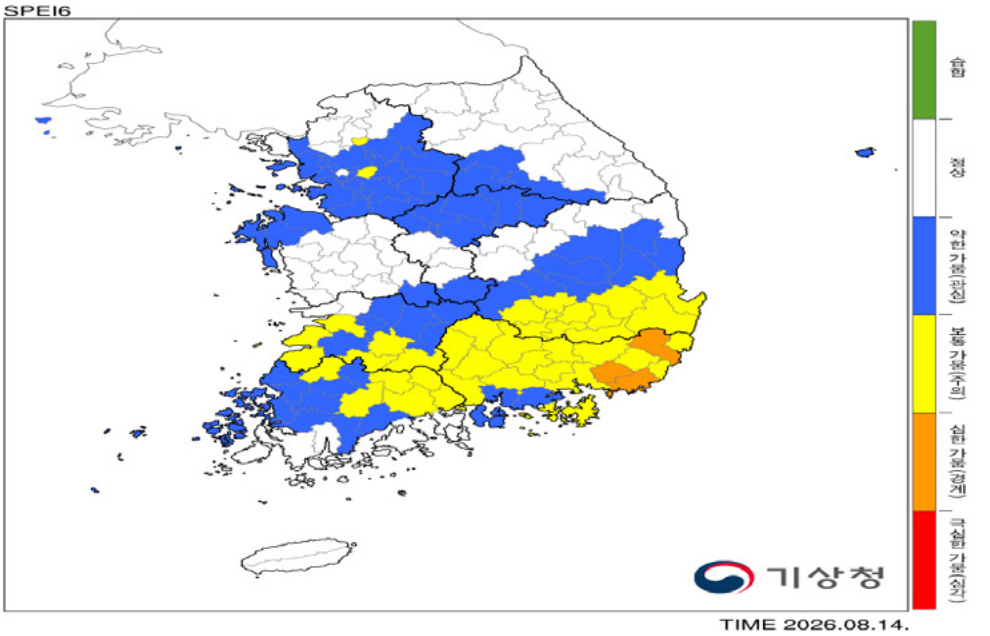
특정 지역의 강수 부족이 지속될 가능성과 기상학적 가뭄 상태를 1·3·6개월 단위로 전망하는 정보입니다.

주요요소	기상가뭄 현황 및 1개월·3개월·6개월전망
제공범위	행정구역 167개 시·군
발표시각	1개월: 매주 목요일 / 3개월: 매월 23일경 / 6개월: 2·5·8·11월 23일경
제공기간	2018년 11월 ~ 현재
생산·갱신주기	1개월 주 1회, 3개월 월 1회, 6개월 연 4회
제공형식	웹페이지, PDF
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/warn/yebo_news.do) → 예측 → 기상가뭄 → 기상가뭄 전망
이용방법	발표일·지역·전망기간을 선택합니다. 가뭄 단계와 전망의 기준을 확인하고 현재 가뭄 현황과 미래 전망을 구분합니다.

5.가뭄정보

- 표준강수지수
- 강수평년비
- 기상가뭄예보
- 표준강수증발산지수

표준강수증발산지수



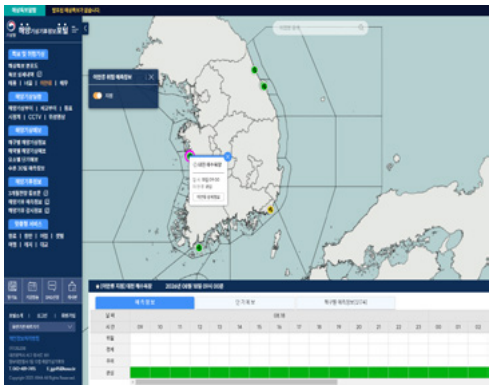
개요

자료상태 : 현행

일정 기간의 누적강수량과 잠재증발산량의 차이를 과거 같은 기간의 값과 비교하여 표준화한 기상가뭄 지수입니다.

주요요소	SPEI1~6·9·12·18·24, 분포도, 시계열
제공범위	행정구역 167개 시·군, 유역 117개 중권역
발표시각	매일 16시경(전일자료 반영)
제공기간	2018년 1월 ~ 현재
생산·갱신주기	1일 1회
제공형식	GIS 분포도, 표, 시계열(PNG 저장 가능)
제공경로	기상청 수문기상 가뭄정보 시스템(https://hydro.kma.go.kr/obse/prcpt_et_idx.do) → 현황 → 기상가뭄 → 표준강수증발산지수
이용방법	지역·기준일·누적기간을 선택합니다. 잠재증발산량의 계산방법과 기준기간을 확인하고 SPI와 비교할 때 누적기간을 맞춥니다.

이안류 위험 예측정보





개요

자료상태 : 현행

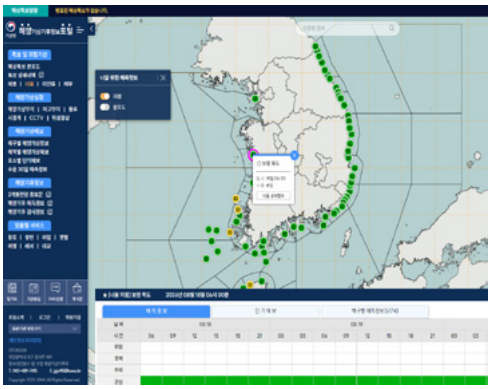
주요 8개 해수욕장 지점의 이안류 위험 예측정보를 4단계(관심, 주의, 경계, 위험)로 제공

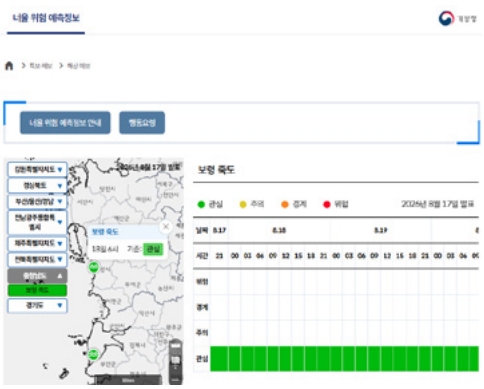
※ 예측정보 단계
- 위험 : 이안류 발생 가능성이 매우 높아 위험한 상태
- 경계 : 이안류 발생 가능성이 높아 경계가 필요한 상태
- 주의 : 이안류 발생 가능성이 있어 주의가 필요한 상태
- 관심 : 이안류 발생 가능성이 낮은 상태

※ 이안류: 해안에서 바다쪽으로 흐르는 폭이 좁고 유속이 빠른 해류를 말함

주요요소	이안류 위험 예측정보	
제공범위	해운대, 중문, 낙산, 대천, 강문, 경포, 안목, 명사십리 해수욕장	
발표시각	72시간(1시간 간격)	
제공기간	2026년 1월 1일 ~ 현재	
생산·갱신주기	일 1회(24시간 주기)	
제공형식	웹이미지, CSV	
제공경로	웹이미지	- 해양기상기후정보포털(marine.kma.go.kr) → 특보 및 위험기상 →이안류 - 날씨누리(weather.go.kr)→특보·예보→해상예보→이안류 위험 예측정보
	CSV	공공데이터포털(data.go.kr)
이용방법	대상 해수욕장 또는 해안지점·발표시각·예측시간을 선택해 위험단계와 행동요령을 확인합니다. 단계값은 위험분류이며 실제 유속이나 파고의 측정값과 구분합니다.	

너울 위험 예측정보





개요

자료상태 : 현행

국지파랑모델(CWW3)을 활용하여 너울에 취약한 동해, 서해, 남해, 제주해안 62개 지점과 소해구를 대상으로 하며 유의파고와 파주기에 따라 너울 위험 예측정보를 4단계(관심, 주의, 경계, 위험)로 제공

※ 예측정보 단계
- 위험 : 매우 높고 강한 너울이 유입되어 해안가에서의 활동이 위험한 상태
- 경계 : 높은 파고의 너울이 유입되어 해안가 활동에 경계가 필요한 상태
- 주의 : 파고가 높거나, 너울이 유입될 가능성이 있어 해안가 활동에 주의가 필요한 상태
- 관심 : 파도가 낮고, 너울 발생 가능성이 낮은 상태

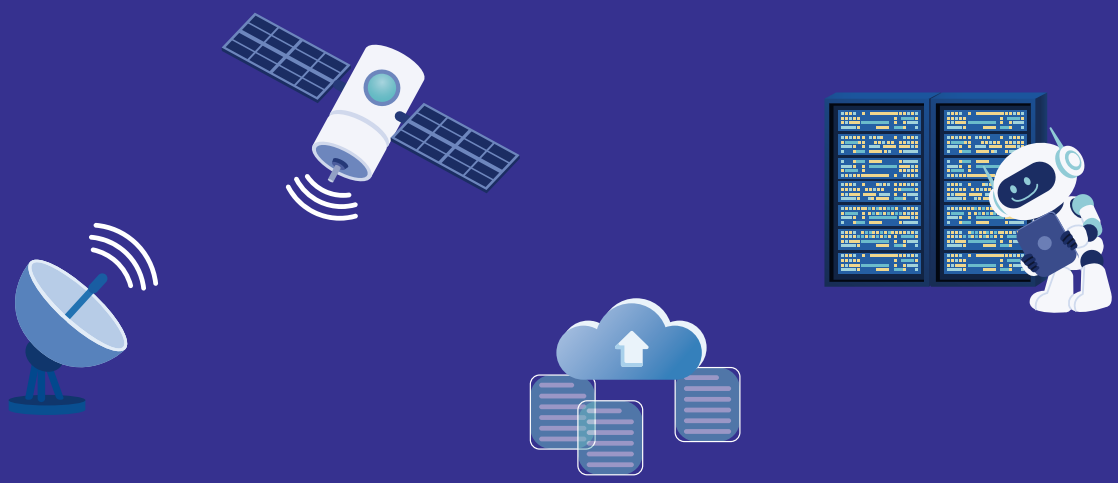
※ 너울 : 먼 해역에서 강한 풍파(Wind wave)가 긴 주기를 가진 채, 해안가로 전달되어 큰 피해로 이어질 수 있는 물결

주요요소	너울 위험 예측정보		
제공범위	동해안	30개소	고성군 거진읍 해안도로, 속초시 영랑동 해안도로, 양양군 현남면 광진리 해안, 강릉시 정동진 해안, 동해시 어달 해안도로, 삼척시 원평 해수욕장, 죽변, 후포, 영덕군 장사 해수욕장, 월포, 구룡포, 일산해수욕장, 진하해수욕장, 강동 몽돌해변, 임랑해수욕장, 일광해수욕장, 송정해수욕장, 명파해변, 송지호해변, 동호해변, 영진해변, 망상해변, 한섬해변, 삼척해변, 축산, 울진, 강구, 읍천, 오류고아라해변, 기성망양해수욕장
	남해안	14개소	학동 흑진주몽돌해변, 영도(남향대교 수변공원), 흥남해수욕장, 남해상주은모래비치, 고흥 남열해수욕장, 완도 신지해수욕장, 신안 짱뚱어해수욕장, 완도 청산도, 완도 여서도, 진도 밀매도, 여수 오동도 방파제, 익금해수욕장,보길도, 서도
	서해안	12개소	가거도, 흑산도, 홍도, 상태도, 만재도, 우이도, 안마도, 백길해수욕장, 대광해수욕장, 변산해수욕장, 보령 죽도, 인천연안부두
	제주해안	6개소	표선, 중문, 월정, 이호, 대정, 협재
발표시각	일 2회		
예측시간	72시간(3시간 간격)		
제공기간	2026년 1월 1일 ~ 현재		
생산·갱신주기	일 2회		
제공형식	웹이미지, ASCII		
제공경로	웹이미지	- 해양기상기후정보포털(marine.kma.go.kr)→특보 및 위험기상→너울 - 날씨누리(www.weather.go.kr)→특보·예보→해상예보→너울 위험 예측정보	
	ASCII	- API허브(https://apihub.kma.go.kr) → 수치모델 → 해구별 예측데이터 조회	
이용방법	대상 해수욕장 또는 해안지점·발표시각·예측시간을 선택해 위험단계와 행동요령을 확인합니다. 단계값은 위험분류이며 실제 유속이나 파고의 측정값과 구분합니다.		

08

응용기상

Applied Meteorology

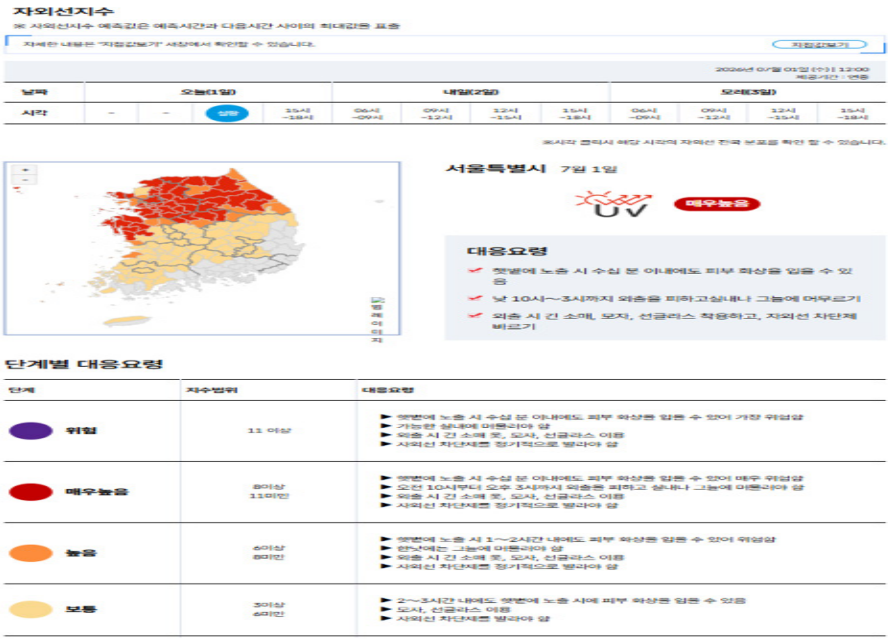




1.기상지수

• 생활기상지수
• 보건기상지수

생활기상지수



개요

자료상태 : 일부 현행

생활과 안전에 필요한 기상조건을 지수로 표현한 자료입니다. 지수별 제공기간·발표시각·예측기간과 단계 기준이 다릅니다. 생활기상지수의 체감온도 자료는 생산 중단 안내가 있으므로 현행 지수와 과거자료를 구분합니다.

주요요소	지점번호, 발표시간, 예측값	
제공범위	단기예보 지점(3,838지점)	
발표시각	지수별 상이(서비스 상세 안내의 발표시각 및 제공 지연 확인)	
제공기간	연중 또는 일정 기간(지수별로 상이)	
제공형식	CSV, JSON, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 응용기상 → 기상지수 → 생활기상지수
	JSON XML	공공데이터포털(https://www.data.go.kr) → '생활기상지수' 검색 → 오픈API 활용신청
이용방법	지수 종류·지역코드·발표시각을 선택하고 지수별 제공 계절·예측기간과 단계표를 확인합니다. API 활용신청 후 응답 오류코드·건수를 확인합니다. 체감온도 예측자료는 중단 공지와 보유 과거기간을 확인합니다.	

1.기상지수

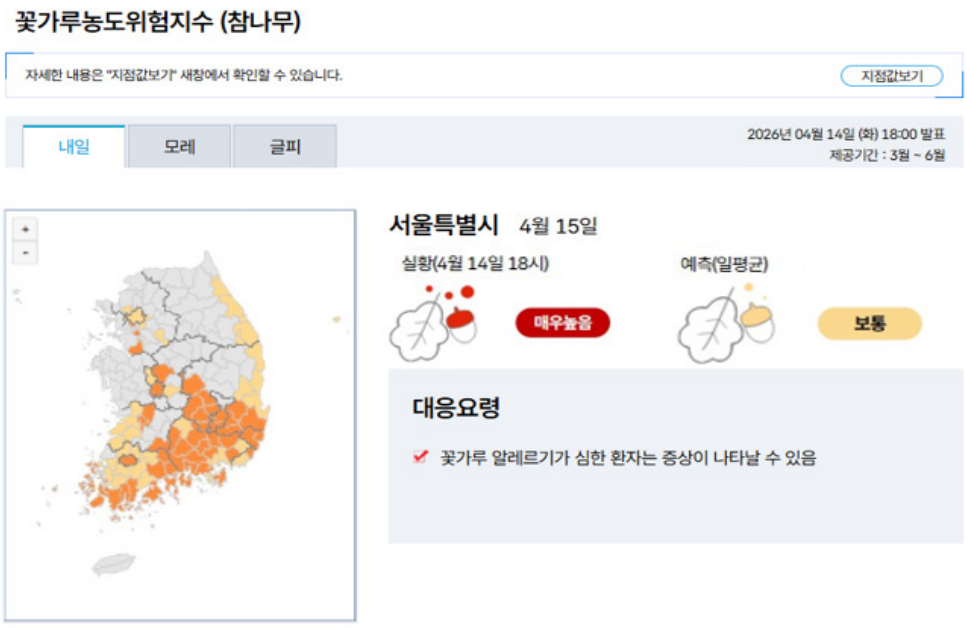
[생활기상지수 종류]				
번호	명칭	설명	생산주기	제공기간
1	자외선지수	태양복사가 최대인 시간대에 지표에 도달하는 자외선 복사량을 나타낸 지수	일 8회 (3시간 간격)	연중
2	동파가능지수 (생산종료)	겨울철 한파로 인해 발생되는 수도관 및 계량기의 동파가능성을 나타낸 지수	일 8회 (3시간 간격)	11월~3월
3	대기정체지수	발생된 오염물질이 대기상태에 의해 변화·소산될 수 있는 가능성을 나타낸 지수	일 8회 (3시간 간격)	연중
4	체감온도 (생산종료)	여름철 대상과 환경에 따라 느끼는 더위의 정도를 나타내는 맞춤형 지수	일 8회 (3시간 간격)	5.15~10.15 (여름철 방재기간)
5	불쾌지수 (생산종료)	여름철 기온과 습도에 따라 사람이 느끼는 불쾌 정도를 나타낸 지수	일 8회 (3시간 간격)	6월~9월
6	열지수 (생산종료)	기온과 습도에 따른 사람이 실제로 느끼는 더위 정도를 나타낸 지수	일 8회 (3시간 간격)	6월~9월
7	더위체감지수* (생산종료)	온열지수(WBGT)를 기반으로 개발된, 대상과 환경에 따라 차별화된 더위 정보를 제공하는 맞춤형 지수	일 2회 (06시, 18시)	5월~9월

* 폭염특보 기준 개선에 따라 더위체감지수를 대상·환경별 세분화된 체감온도 서비스로 변경('20.5.15.)

1.기상지수

• 생활기상지수
• 보건기상지수

보건기상지수



개요

자료상태 : 일부 현행

보건기상지수는 기상조건에 따른 질병 발생가능정도를 지수화하여 국민 건강보호에 활용할 수 있도록 개발된 정보입니다. 단기예보 지점별로 조회가 가능하며 오늘, 내일, 모레, 글피의 예측단계를 제공합니다.

주요요소	지점번호, 발표시간, 예측값	
제공범위	단기예보지점(3,838지점)	
발표시각	지수별 상이(서비스 상세 안내의 발표시각 및 제공 지연 확인)	
제공기간	연중 또는 일정 기간(지수별로 상이)	
제공형식	CSV, JSON, XML	
제공경로	CSV	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 응용기상 → 기상지수 → 보건기상지수
	JSON	공공데이터포털(https://www.data.go.kr) → '보건기상지수' 검색 → 오픈API 활용신청
	XML	
이용방법	지수 종류·지역코드·발표시각을 선택합니다.	

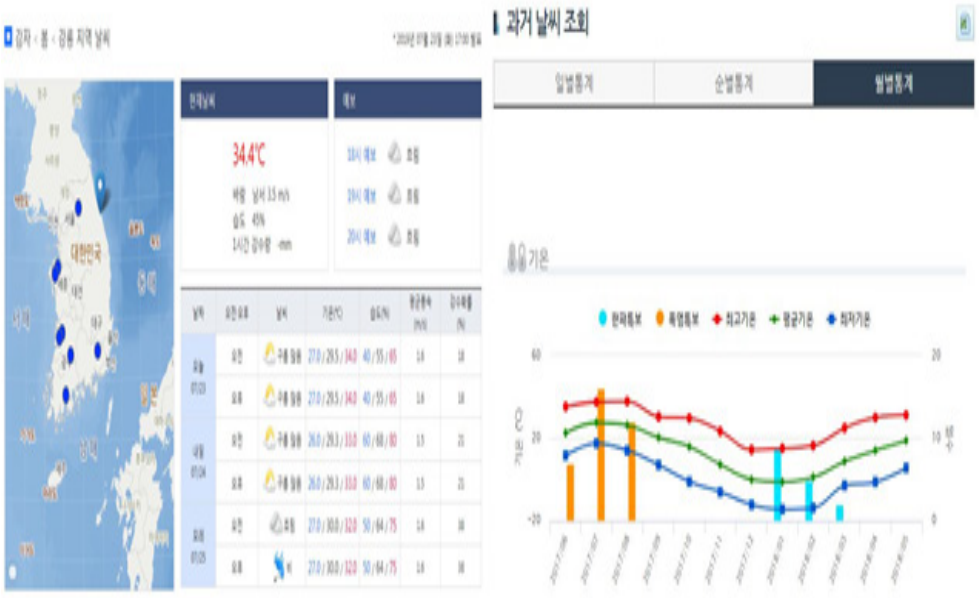
1.기상지수

[보건기상지수 종류]				
번호	명칭	설명	생산주기	제공기간
1	식중독지수 (생산종료)	날씨와 환경 변화에 따른 식중독 발생 위험도를 나타내는 지수	일 2회 (06시, 18시)	연중
2	감기가능지수 (생산종료)	기상조건(최저기온, 일교차, 현지기압, 상대습도)에 따른 감기 발생 가능성을 지수화	일 2회 (06시, 18시)	9월~4월
3	천식폐질환 가능지수 (생산종료)	기상조건(최저기온, 일교차, 현지기압, 상대습도)에 따른 천식·폐질환 발생 가능정도를 지수화	일 2회 (06시, 18시)	연중
4	뇌졸중 가능지수 (생산종료)	기상조건(최저기온, 일교차, 현지기압, 상대습도)에 따른 뇌졸중 발생 가능정도를 지수화	일 2회 (06시, 18시)	연중
5	꽃가루농도 위험지수	기상조건(평균기온, 최저기온, 최고기온, 상대습도, 풍속, 강수량 등)에 따른 수목류 및 잡초류 꽃가루농도를 예측하여 알레르기질환 발생 가능정도를 지수화	일 2회 (06시, 18시)	참나무: 3월~6월 소나무: 3월~6월 잡초류: 8월~10월
6	피부질환 가능지수 (생산종료)	기상조건(최고기온, 상대습도)에 따른 피부질환발생 가능정도를 지수화	일 2회 (06시, 18시)	연중

2.융합 서비스

- 농업주산지 기상정보
- 관광코스별 기상정보
- 기상청 도로날씨정보
- 서리발생예측정보
- 전력기상지수
- 도로위험기상정보

농업주산지 기상정보



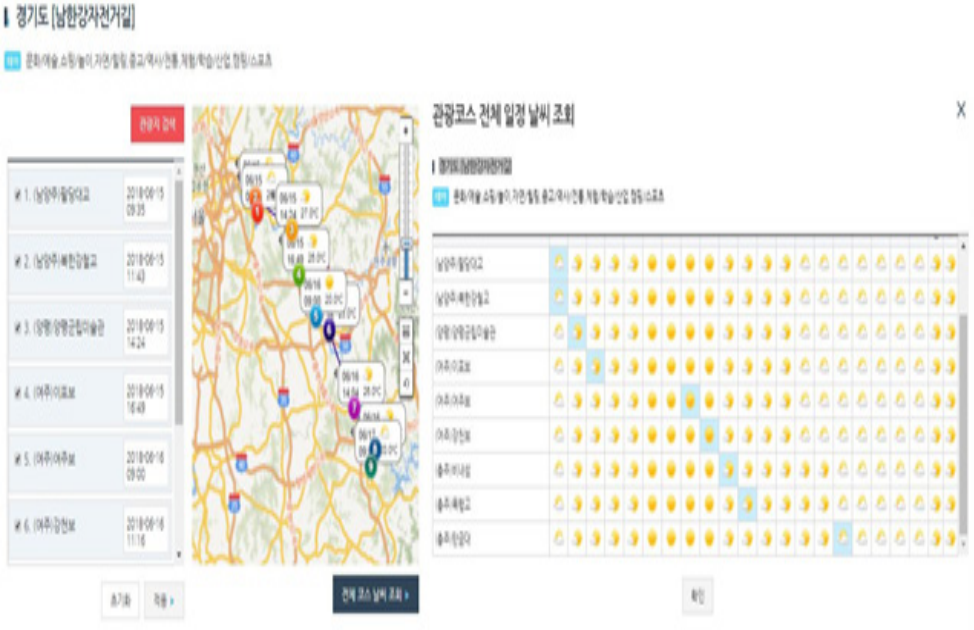
개요

자료상태 : 현행

농업주산지 기상정보는 36개 주요 작물이 주로 재배되는 주산지의 단기예보, 중기예보 그리고 과거 날씨 정보를 제공하는 맞춤형 기상기후 융합서비스로 기상기후 자료 활용 확산을 위한 모형으로 2016년부터 날씨마루(기상기후 빅데이터 분석 플랫폼) 웹과 공공 OpenAPI로 제공되고 있습니다.

주요요소	단기예보, 중기예보, 지상관측(통계)
제공범위	36개 주요 작물이 주로 재배되는 주산지 · 서비스 대상 작물 선정 : 한국농수산식품유통공사, 한국농촌경제연구원 · 농작물 생육 주기별 주요 기상정보 : 농촌진흥청
제공기간	2016년 6월 ~ 현재
생산·갱신주기	시간 자료 또는 일 자료(지수별로 다름)
제공형식	XML
제공경로	공공데이터포털(https://www.data.go.kr) → '농업 주산지 기상정보' 검색 → 오픈API 활용신청
이용방법	작물·주산지 코드와 조회기간을 선택합니다. 실황·단기·중기예보를 구분하고 지역코드와 대상시각을 확인합니다.

관광코스별 기상정보



개요

자료상태 : 현행

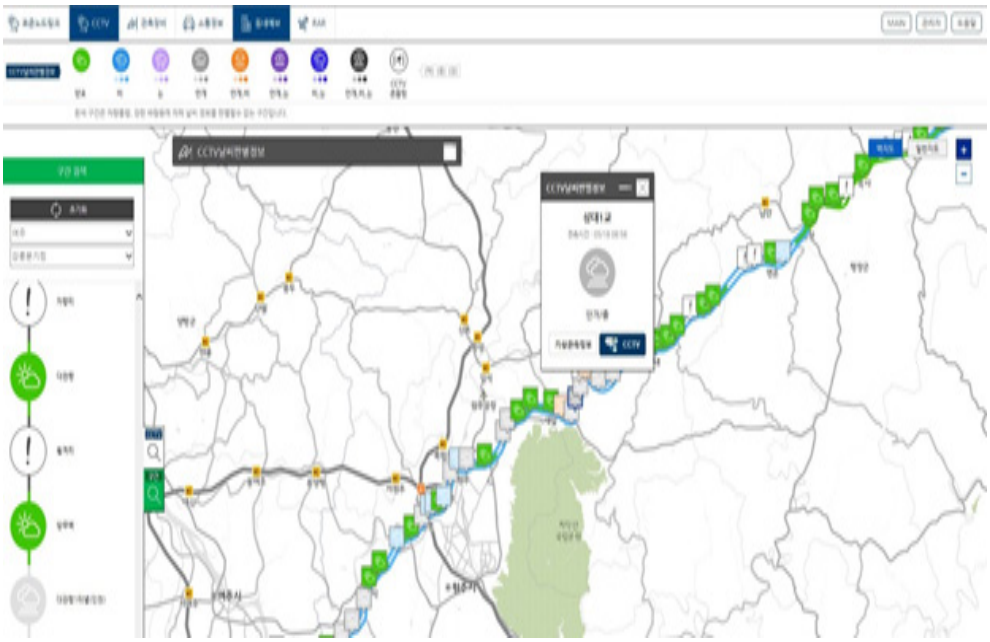
기상청 단기예보를 한국관광공사에서 추천하는 관광코스별(400여 개)로 제공하는 맞춤형 기상기후 융합서비스로 기상기후 자료 활용 확산을 위한 모형으로 2016년 부터 날씨마루(기상기후 빅데이터 분석 플랫폼) 웹과 공공 OpenAPI로 제공되고 있습니다.

주요요소	단기예보(3시간 기온, 습도, 풍속, 하늘상태, 강수확률)
제공범위	관광코스(400여 개)
제공기간	2016년 6월 ~ 현재
생산·갱신주기	6시간 주기
제공형식	XML
제공경로	공공데이터포털(https://www.data.go.kr) → '관광코스 기상정보' 검색 → 오픈API 활용신청
이용방법	관광코스 코드·발표일·예측시간을 선택합니다. 코스별 적용 위치와 기상요소·지수의 단위를 확인합니다. 제공되는 코스 목록과 운영기간을 먼저 조회합니다.

2.융합 서비스

- 농업주산지 기상정보
- 관광코스별 기상정보
- 기상청 도로날씨정보
- 서리발생예측정보
- 전력기상지수
- 도로위험기상정보

기상청 도로날씨정보



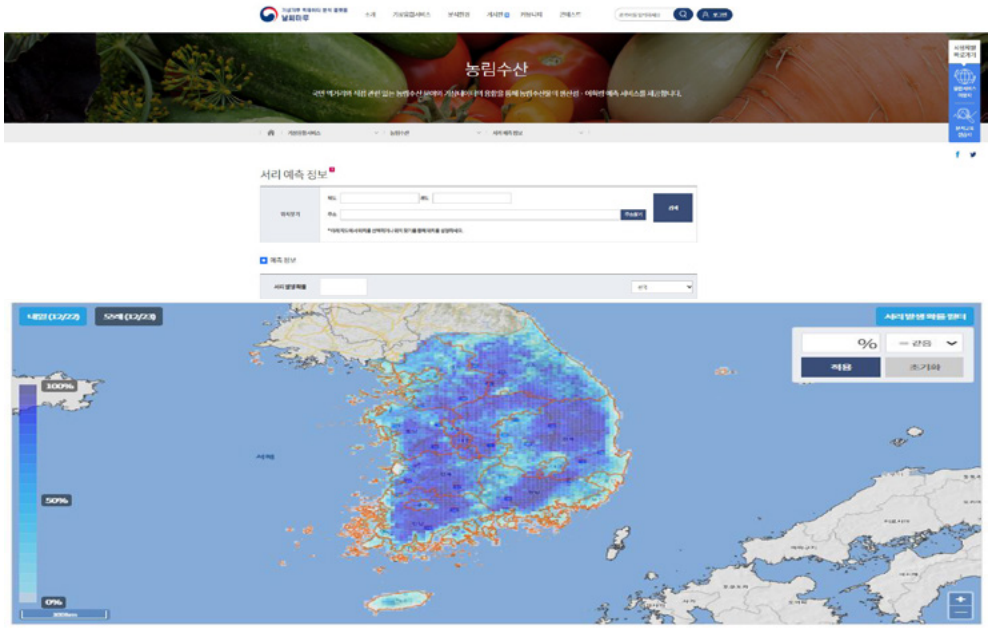
개요

자료상태 : 현행

도로날씨정보는 도로에 설치되어 있는 교통 CCTV 영상을 인공지능 기법으로 분석하여 생산하는 도로위 기상정보(비, 눈, 안개)로서 안전한 도로주행을 위해 제공하는 정보입니다.

주요요소	날씨정보(비, 눈, 안개, 양호)와 안개강도(강, 중, 약)
제공범위	경부선, 중부선, 영동선, 중앙선 등 총 11개 노선
발표시각	낮 시간대 실시간
유효시간	실시간
제공기간	2018년 1월 ~ 현재
생산·갱신주기	낮 시간대 10분 주기
제공형식	XML, JSON
제공경로	공공데이터포털(https://www.data.go.kr) → 'CCTV 기반 도로날씨정보서비스' 검색 → 오픈API 활용신청
이용방법	CCTV 또는 도로지점 코드·조회시각을 선택합니다. CCTV 영상분석 시각과 판별코드·품질정보를 확인하고, 결측·판별불가를 맑음으로 처리하지 않습니다.

서리발생예측정보



개요

자료상태 : 현행

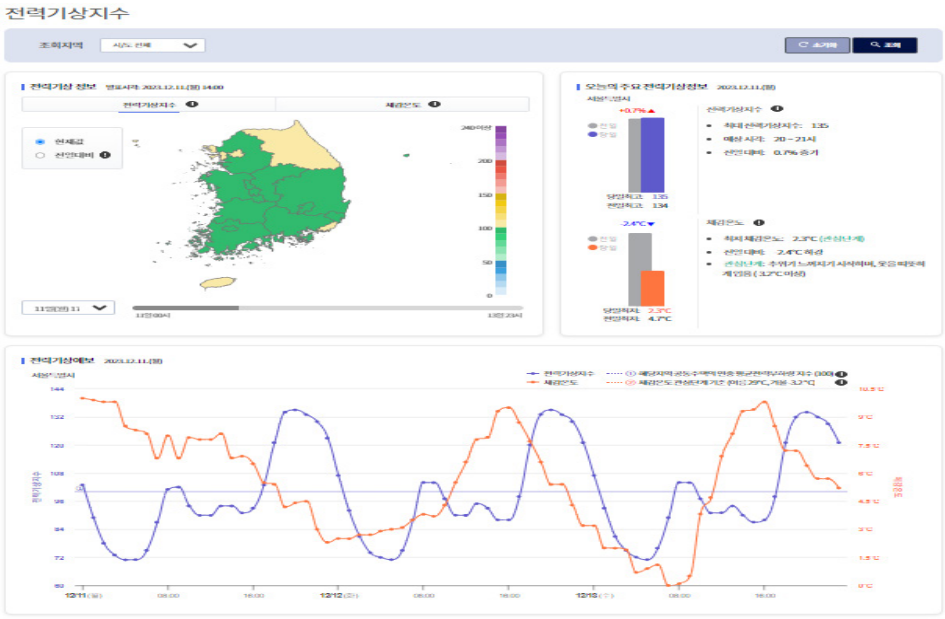
서리발생예측정보는 단기예보 기반의 서리 발생 가능성을 사전에 예측하고, 발생가능성을 수치화 및 시각화하여 서리예측정보를 확률로 제공하는 정보입니다.

주요요소	단기예보
제공범위	단기예보 격자(전국)
발표시각	일 2회
유효시간	-
제공기간	2018.12.~
생산·갱신주기	일 2회
제공형식	XML, JSON
제공경로	공공데이터포털(https://www.data.go.kr) → '서리발생예측정보' 검색 → 오픈API 활용신청
이용방법	지역·발표시각·예측 대상시각을 지정합니다. 확률(%) 또는 단계의 의미와 결측 기준을 확인하고 실제 서리 관측과 비교할 때 시간·위치를 맞춥니다.

2.융합 서비스

- 농업주산지 기상정보
- 관광코스별 기상정보
- 기상청 도로날씨정보
- 서리발생예측정보
- 전력기상지수
- 도로위험기상정보

전력기상지수



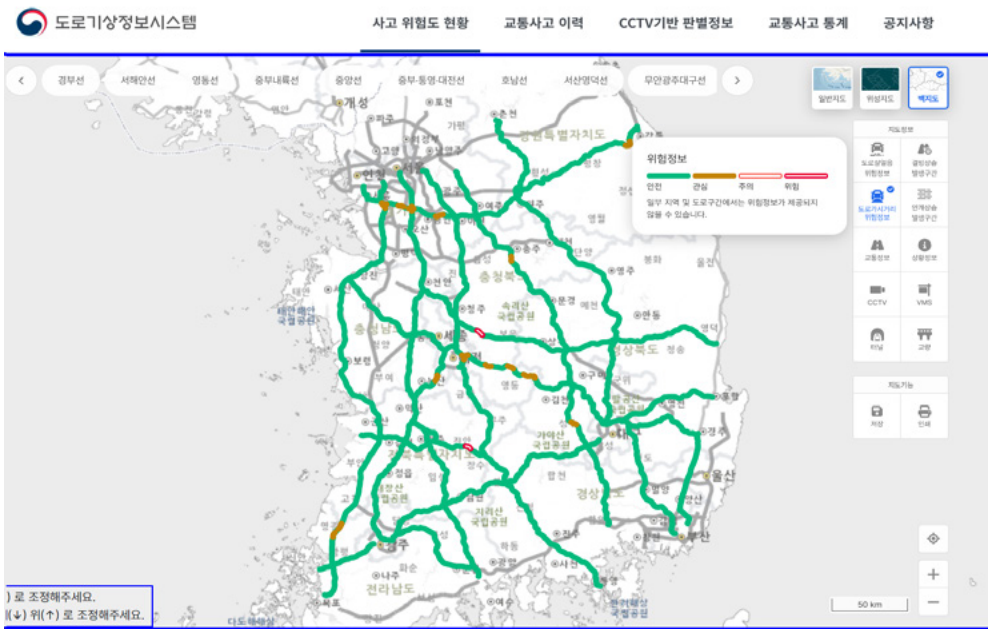
개요

자료상태 : 현행

전력기상지수는 기상변화에 따른 공동주택의 전력사용량 증감을 쉽게 인지할 수 있도록 체감온도와 전력사용의 시간적, 계절적 특성을 활용해 전력사용량 증감을 지수화한 것입니다.

주요요소	전력기상지수, 체감온도
제공범위	5km 해상도 단기예보 격자(남한)
발표시각	2, 5, 8, 11, 14, 17, 20, 23시 5분
유효시간	3일
제공기간	2024.1.~
생산·갱신주기	3시간
제공형식	JSON
제공경로	공공데이터포털(https://www.data.go.kr) → '전력기상지수' 검색 → 오픈API 활용신청
이용방법	지역·시각과 제공 지수 항목을 선택합니다. 지수의 기준값과 증가·감소 의미를 확인하며, 전력량(kWh)으로 직접 환산하지 않습니다.

도로위험기상정보



개요

자료상태 : 현행

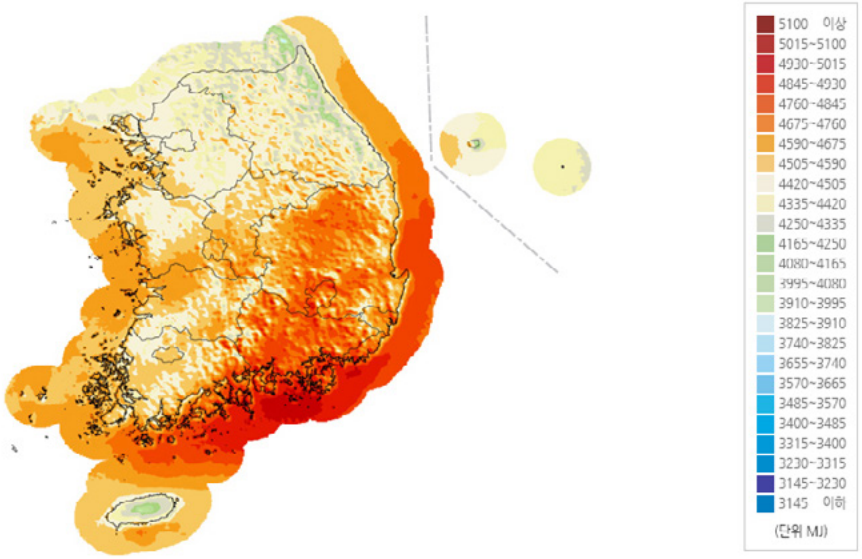
도로교통 안전관리 지원을 위해 도로살얼음 및 가시거리에 대한 위험도를 3단계(관심, 주의, 위험)로 분류하여 실시간 산출하는 자료입니다. 도로기상정보시스템 누리집(rwis.kma.go.kr)을 통해 시각화된 자료가 제공됩니다.

주요요소	도로살얼음 발생 가능정보, 도로 가시거리 위험정보
제공범위	- (제공중) 전국 12개 노선 ※ 경부선, 서해안선, 영동선, 중부내륙선, 중앙선, 통영-대전·중부선, 호남선, 서산영덕선, 무안광주·광주대구선, 순천완주선, 새만금-포항선, 호남선의 지선 - (2026년 12월 제공 예정) 남해선
발표시각	매시 00·05·10·15·20·25·30·35·40·45·50·55분 (일 288회, 5분 간격)
제공기간	2025.12.15.~현재 ※ 요소별 제공기간: (도로살얼음) 11.15.~3.15. / (안개) 연중
생산·갱신주기	일 288회, 5분 간격
제공형식	CSV
제공경로	기상청 API허브(apihub.kma.go.kr) → 융합기상 → 교통안전 → 도로위험기상정보
이용방법	API허브의 융합기상 → 교통안전에서 도로명·시각을 지정합니다. 도로살얼음·가시거리의 위험 단계와 제공시각을 확인하고 실제 노면 상태와 구분합니다.

3.기상자원 지도

• 위성기반 태양광자원지도
• 풍력자원지도

위성기반 태양광자원지도



개요

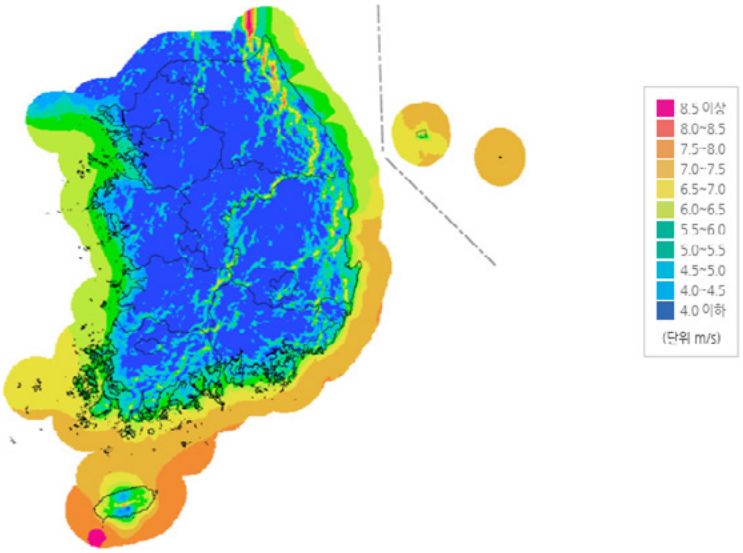
자료상태 : 현행

기상청의 천리안2A호 관측자료와 AI 기법을 활용하여 산출한 일사량 정보를 다양한 기간에 대해 통계처리한 자료

※ 통계기간: 2020.1.~2024.12.
※ 통계방법: 연평균, 월평균, 시간평균

주요요소	1시간 누적 수평면 일사량
제공범위	남한지역(인근 해상 포함)
생산·갱신주기	1회/년 또는 수시
제공형식	이미지, netcdf
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 응용기상 → 기상자원지도 → 태양기상자원지도 재생에너지 기상정보 플랫폼(https://energy.kma.go.kr → 기상자원서비스 → 태양광 → 기상자원지도
이용방법	자원 종류·대상 지역·통계기간·요소와 높이 등 조회조건을 선택합니다. 지도 해상도·좌표·단위와 관측 또는 재현자료의 산출방법을 확인합니다.

풍력자원지도



개요

자료상태 : 현행

관측자료와 수치예보모델 결과를 기상청에서 자체 개발한 AI-변분자료동화(variational data assimilation) 기술을 활용하여 재현한 재현바람장을 활용하여 요소별 통계처리(2021.6~2026.5)하여 바람특성을 지도 형태로 제공

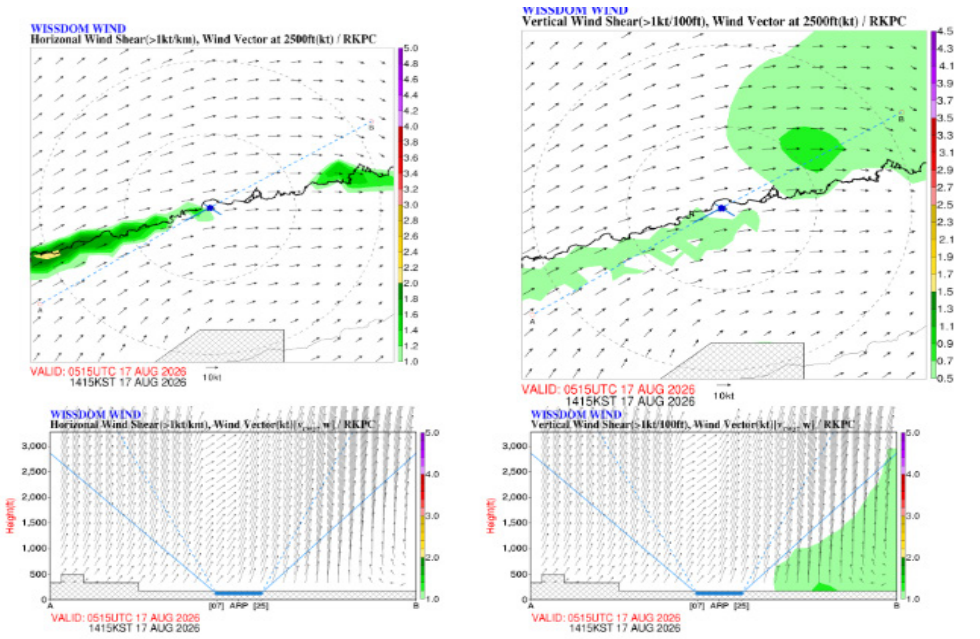
※ 통계기간은 매년 업데이트 예정

주요요소	고도별(10m, 80m, 140m, 220m) 평균풍속, 표준편차, 평균 일최대풍속, 주풍향, 주풍향비율, 주풍향지속시간, 가용풍속비율, 바람장미, Weibull 형상계수, Weibull 척도계수
제공범위	남한지역(인근 해상 포함)
생산·갱신주기	1회/년 또는 수시
제공형식	이미지, netcdf
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 응용기상 → 기상자원지도 → 풍력기상자원지도 재생에너지 기상정보 플랫폼(https://energy.kma.go.kr → 기상자원서비스 → 풍력 → 기상자원지도
이용방법	자원 종류·대상 지역·통계기간·요소와 높이 등 조회조건을 선택합니다. 지도 해상도·좌표·단위와 관측 또는 재현자료의 산출방법을 확인합니다.

4.항공

• 레이더 급변풍 탐지 일기도

레이더 급변풍 탐지 일기도



개요

자료상태 : 현행

제주공항은 지형적 특성으로 인해 국내 공항 중 급변풍이 가장 빈번하게 발생하는 공항입니다. 이에 따라 항공기 안전운항을 지원하기 위해 레이더 기반 급변풍 탐지 일기도를 개발하였습니다.
본 데이터는 레이더 3차원 바람장 자료(WISSDOM*)를 활용하여 공항 주변의 수평 급변풍(kt/1km)과 연직 급변풍(kt/100ft)을 탐지하고, 이를 일기도 형태로 제공합니다.

* WISSDOM: WInd Synthesis System using Doppler Measurements

공간 해상도는 수평 약 1km이며, 연직 표출 범위는 지상~약 1km 미만(지상~3,000ft)입니다. 자료는 5분 간격으로 실시간 생산됩니다.

이렇게 생산된 최신 일기도는 항공기상청 항공날씨 웹페이지(http://global.amo.go.kr/)를 통해 실시간으로 확인하실 수 있습니다.

주요요소	- 수평급변풍: 공항 주변(16km 반경) 수평급변풍 분포도, 이·착륙 경로 수평급변풍 연직 단면도 - 연직급변풍: 공항 주변(16km 반경) 연직급변풍 분포도, 이·착륙 경로 연직급변풍 연직 단면도
제공범위	제주공항
생산주기	매 5분
제공기간	2025년 1월 1일 ~ 현재
제공형식	png
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 제공 신청
이용방법	분석시각·고도·수평 또는 연직 급변풍 요소를 선택해 일기도를 조회합니다. 탐지값의 단위와 범례를 확인하며, 예측 일기도나 LLWAS 경고와 구분합니다.

09

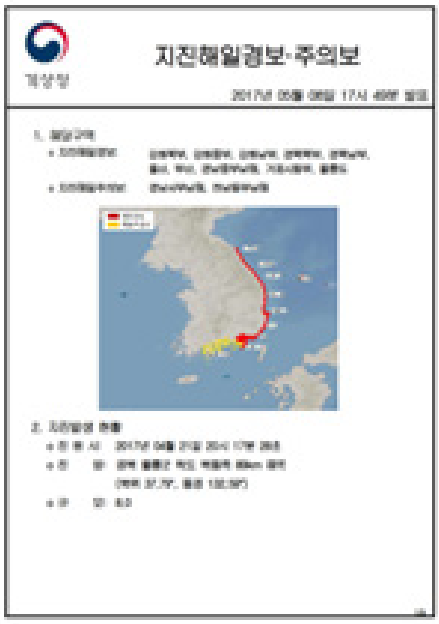
지진화산

Earthquakes and Volcanoes



1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

지진해일특보



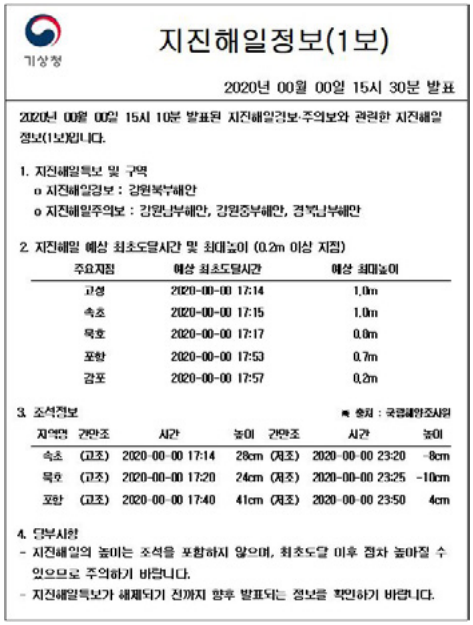
개요

자료상태 : 현행

지진해일특보는 지진해일주의보와 지진해일경보로 구분됩니다.
지진해일주의보는 한반도의 주변해역에서 규모 6.0 이상의 해저지진이 발생하여 우리나라 해안에 해일파고 0.5m 이상 1.0m 미만의 지진해일 내습이 예상될 때, 지진해일경보는 규모 6.0 이상의 해저지진이 발생하여 1.0m 이상의 지진해일 내습이 예상될 때 발표됩니다.

주요요소	지진해일특보 및 구역, 지진발생현황(발생시각, 발생위치, 발생깊이, 규모), 구역별 지진해일 예상 최초도달시간 및 최대높이
제공범위	지진해일 감시구역(21°N ~ 45°N, 110°E ~145°E)
발표시각	발생 시
제공기간	2005년 3월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

지진해일정보



기상청

지진해일정보(1보)

2020년 00월 00일 15시 30분 발표

2020년 00월 00일 15시 10분 발표된 지진해일경보·주의보와 관련한 지진해일 정보(1보)입니다.

1. 지진해일특보 및 구역
   - o 지진해일경보 : 강원북부해안
   - o 지진해일주의보 : 강원남부해안, 강원중부해안, 경북남부해안

2. 지진해일 예상 최초도달시간 및 최대높이 (0.2m 이상 지점)

주요지점	예상 최초도달시간	예상 최대높이
고성	2020-00-00 17:14	1.0m
속초	2020-00-00 17:15	1.0m
묵호	2020-00-00 17:17	0.8m
포항	2020-00-00 17:53	0.7m
감포	2020-00-00 17:57	0.2m

3. 조석정보 ※ 출처 : 국립해양조사원

지역명	간만조	시간	높이	간만조	시간	높이
속초	(고조)	2020-00-00 17:14	28cm	(저조)	2020-00-00 23:20	-8cm
묵호	(고조)	2020-00-00 17:20	24cm	(저조)	2020-00-00 23:25	-10cm
포항	(고조)	2020-00-00 17:40	41cm	(저조)	2020-00-00 23:50	4cm

4. 당부사항
   - 지진해일의 높이는 조석을 포함하지 않으며, 최초도달 이후 점차 높아질 수 있으므로 주의하기 바랍니다.
   - 지진해일특보가 해제되기 전까지 향후 발표되는 정보를 확인하기 바랍니다.

개요

자료상태 : 현행

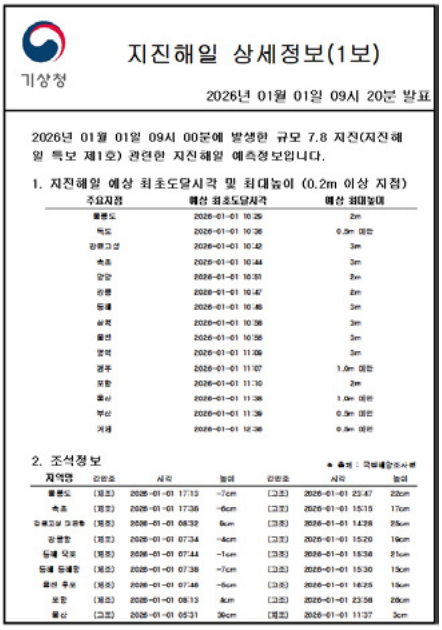
지진해일정보는 지진해일특보 기준에는 미치지 못하나
우리나라에 0.2m 이상 0.5m 미만의 지진해일이 예상되거나, 0.2m미만 예상되나 국외지진해일센터가 국내 영향을 예측한 경우 알릴 필요가 있을 때 발표됩니다.

주요요소	지진해일특보 및 구역, 지진발생현황(발생시각, 발생위치, 발생깊이, 규모), 주요지점 지진해일 예측정보, 조석정보, 지진해일 관측정보, 당부사항 등
제공범위	지진해일 감시구역(21°N ~ 45°N, 110°E ~ 145°E)
발표시각	발생 시
제공기간	2020년 4월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

1.지진화산 특·정보

- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

지진해일 상세정보


기상청

지진해일 상세정보(1보)

2026년 01월 01일 09시 20분 발표

2026년 01월 01일 09시 00분에 발생한 규모 7.8 지진(지진해일 특보 제1호) 관련한 지진해일 예측정보입니다.

1. 지진해일 예상 최초도달시각 및 최대높이 (0.2m 이상 지점)

주요지점	예상 최초도달시각	예상 최대높이
울릉도	2026-01-01 10:25	2m
독도	2026-01-01 10:36	0.5m 미만
강릉그섬	2026-01-01 10:42	3m
속초	2026-01-01 10:44	3m
양양	2026-01-01 10:33	2m
강릉	2026-01-01 10:47	2m
동해	2026-01-01 10:48	3m
삼척	2026-01-01 10:58	3m
울진	2026-01-01 10:56	3m
영덕	2026-01-01 11:09	3m
경주	2026-01-01 11:07	1.0m 미만
포항	2026-01-01 11:10	2m
울산	2026-01-01 11:36	1.0m 미만
부산	2026-01-01 11:39	0.5m 미만
거제	2026-01-01 12:36	0.5m 미만

2. 조석정보

지역명	관련조	시각	높이	관련조	시각	높이
울릉도	(제3)	2026-01-01 17:13	-7cm	(그3)	2026-01-01 23:47	22cm
속초	(제3)	2026-01-01 17:38	-6cm	(그3)	2026-01-01 15:15	17cm
강릉그섬	(제3)	2026-01-01 08:32	6cm	(그3)	2026-01-01 14:38	25cm
강릉항	(제3)	2026-01-01 07:34	-4cm	(그3)	2026-01-01 15:20	19cm
동해 묵호	(제3)	2026-01-01 07:44	-1cm	(그3)	2026-01-01 15:38	21cm
동해 동해항	(제3)	2026-01-01 07:38	-7cm	(그3)	2026-01-01 15:30	15cm
울산 우보	(제3)	2026-01-01 07:46	-5cm	(그3)	2026-01-01 16:25	15cm
포항	(제3)	2026-01-01 08:13	4cm	(그3)	2026-01-01 23:58	20cm
울산	(그3)	2026-01-01 05:31	39cm	(제3)	2026-01-01 11:37	3cm

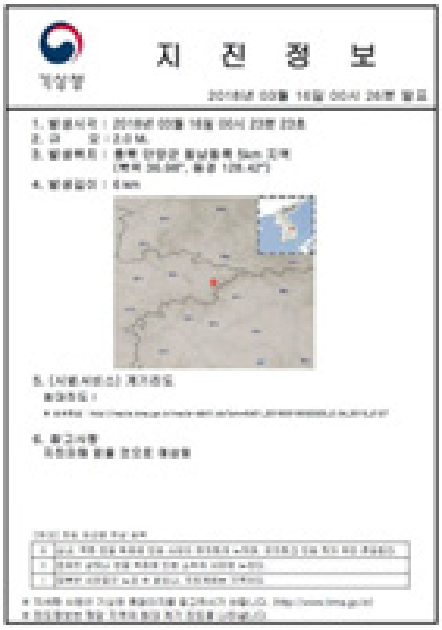
개요

자료상태 : 현행

지진해일 상세정보는 지진해일특·정보 발표 이후 주요 지점별 지진해일 예측정보 또는 실제 관측된 지진해일 자료 등의 추가 정보를 알릴 필요가 있을 때 발표됩니다.

주요요소	주요지점의 지진해일 예상 최초도달시각 및 최대높이, 관측지점에서 관측된 지진해일 최초도달시각 및 최대높이와 그 관측시각, 조석정보, 해수면 최대높이 등
제공범위	지진해일 감시구역(21°N ~ 45°N, 110°E ~ 145°E)
발표시각	발생 시
제공기간	2025년 12월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

지진정보



개요

자료상태 : 현행

지진정보는 국내에서 규모 2.0 이상인 지진이 발생한 경우 발표됩니다.

주요요소	발생시각, 발생위치, 규모, 발생깊이, 계기진도
제공범위	국내에서 발생한 지진
발표시각	발생 시
제공기간	2001년 3월 ~ 현재
생산·갱신주기	수시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
간행물	지진연보를 매년 정기적으로 발간
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

미소지진

개요

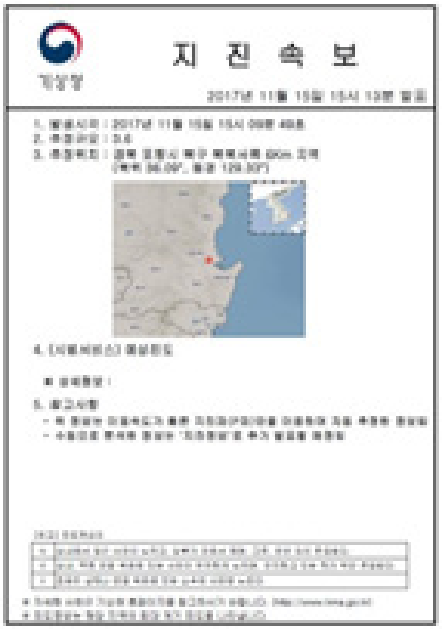
자료상태 : 현행

미소지진은 세기가 매우 작은 지진으로 보통 규모 2.0 미만의 지진을 의미합니다. 국내에서 규모 2.0 미만인 지진이 발생한 경우 홈페이지를 통해 공개됩니다.

주요요소	발생시각, 발생위치, 규모, 발생깊이, 계기진도
제공범위	국내에서 발생한 지진
발표시각	발생 시
제공기간	2012년 1월 ~ 현재
생산·갱신주기	수시
제공형식	CSV
제공경로	기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 지진조회 → 미소지진포함보기 기상청 기상자료개방포털(http://data.kma.go.kr)
이용방법	국내지진 목록에서 미소지진 포함 조건과 기간·지역을 설정합니다. 목록의 최종 분석값과 자료 갱신시점을 확인합니다.

지진속보

1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보


지 진 속 보

개요

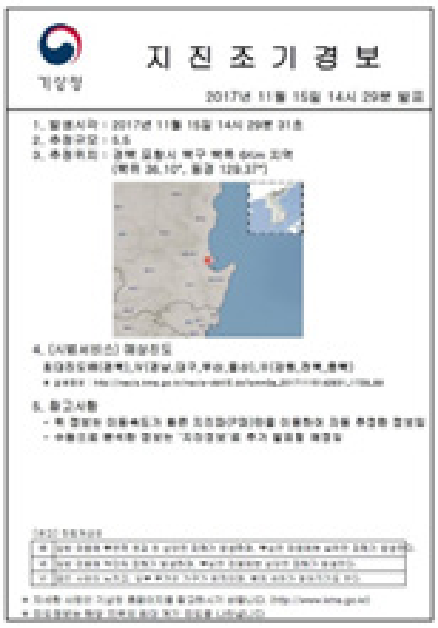
자료상태 : 현행

지진속보는 국내 지역에서 규모 3.5 이상 또는 국내 해역에서 규모 4.0 이상이며 규모 5.0 미만의 지진에 대하여 발표합니다.

주요요소	발생시각, 추정위치, 추정규모, 예상진도
제공범위	국내에서 발생한 지진
발표시각	발생 시
제공기간	2005년 3월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

지진조기경보



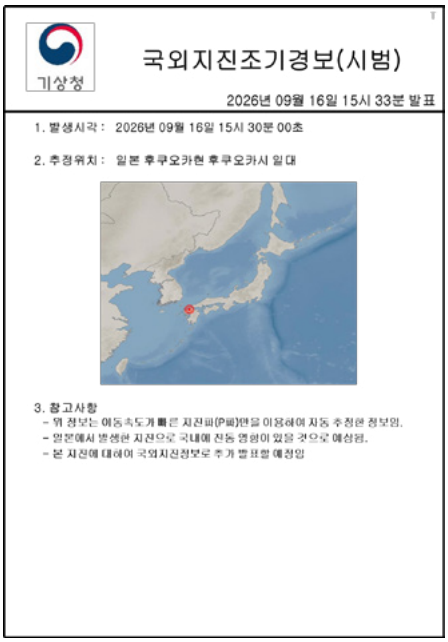
개요

자료상태 : 현행

지진조기경보는 규모 5.0 이상인 지진이 국내에서 발생하거나, 국외에서 발생한 지진이 국내에 상당한 영향을 미칠 것으로 예상될 때 발표됩니다.

주요요소	발생시각, 추정위치, 추정규모, 예상진도
제공범위	국내에서 발생한 지진 ※ 기타 국외지진 중 국내에서 많은 사람이 느낀 지진도 포함
발표시각	발생 시
제공기간	2015년 1월 22일 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

국외지진조기경보


국외지진조기경보(시범)

기상청

2026년 09월 16일 15시 33분 발표

1. 발생시각 : 2026년 09월 16일 15시 30분 00초

2. 추정위치 : 일본 후쿠오카현 후쿠오카시 일대

3. 참고사항

1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

개요

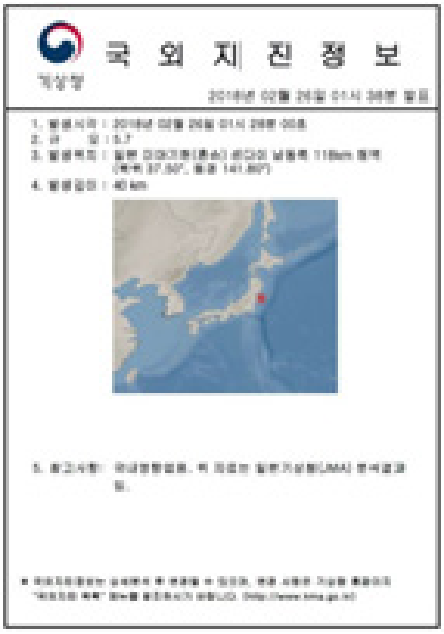
자료상태 : 현행

국외지진조기경보는 국외 지진조기경보 영역에서 발생한 규모 5.0 이상의 지진으로 국내(부산, 경남, 제주) 예상진도 IV 이상 되는 경우 발표됩니다.

주요요소	발생시각, 추정위치, 참고사항
제공범위	국외 지진조기경보 영역(일본 서남부 규슈 일대)
발표시각	발생 시
제공기간	2018년 5월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

1.지진화산 특·정보
- 지진해일특보
- 지진해일정보
- 지진해일 상세정보
- 지진정보
- 미소지진
- 지진속보
- 지진조기경보
- 국외지진조기경보
- 국외지진정보
- 화산정보·화산재특보

국외지진정보



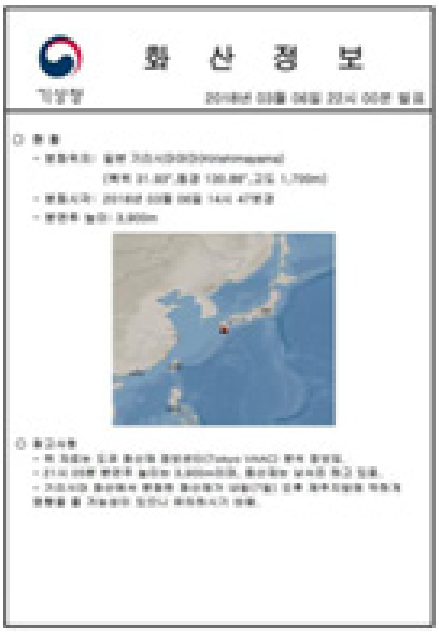
개요

자료상태 : 현행

국외지진정보는 감시구역 내에서 규모 5.0 이상 또는 진도 II 이상의 지진이 발생했을 경우, 구역 외에서는 규모 6.0 이상의 지진이 발생했을 경우 발표됩니다.

주요요소	발생시각, 발생위치, 규모, 발생깊이
제공범위	감시구역 내(북위 21°~45°, 동경 110°~145°) 및 그 외 국외지진
발표시각	발생 시
제공기간	2001년 3월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표 종류·기간을 선택하여 정보를 확인합니다. 발생시각·발표시각, 규모·진도 의미를 구분합니다.

화산정보·화산재특보



개요

자료상태 : 현행

화산정보는 한반도 및 그 주변 지역 또는 국외에서 발생한 화산현상에 관한 정보를 말합니다. 화산재특보에는 화산재주의보와 화산재경보가 있으며, 발표 기준은 화산재로 인하여 우리나라에 피해가 예상되는 정도입니다.

- 국내: (화산재특보) 한반도 내의 화산이 분화할 징조가 나타나거나 분화한 경우
- 국외: (화산재정보) 화산 분화로 인해 국내에 영향을 미칠 것으로 예상되거나, 전 지구적 대규모 화산 분화로 국민에게 알릴 필요가 있는 경우

(화산재특보) 우리나라에 영향을 줄 수 있는 화산이 분화하여 화산재에 의한 피해 예상 시

주요요소	화산정보: 현황(화산명, 분화위치, 분화시각, 분연주 높이 등) 화산재특보: 해당구역, 발효시각, 현황(화산명, 분화위치, 분화시각, 분연주 높이 등)
제공범위	국내외
발표시각	발생 시
제공기간	2010년 7월 ~ 현재
생산·갱신주기	발생 시
제공형식	PDF(통보문)
제공경로	(실시간) 기상청 날씨누리(http://www.weather.go.kr) → 지진·화산 → 발표정보 (과거자료) 공공데이터포털(http://www.data.go.kr) → 공공데이터 제공 신청
이용방법	발표일·화산 또는 대상영역을 선택해 원문·수정·해제 정보를 확인합니다. 분화정보와 화산재특보의 적용 지역·유효시각을 구분합니다.

10

기상연구

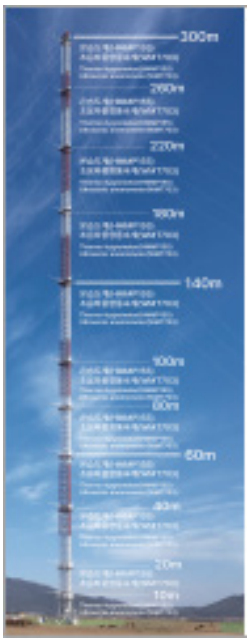
Meteorological Research



1.표준기상

• 보성표준기상관측소 연직관측

보성표준기상관측소 연직관측



개요

자료상태 : 현행

보성 표준기상관측소에 위치한 기상관측탑의 자료는 총 11개 층(10, 20, 40, 60, 80, 100, 140, 180, 220, 260, 300m)에서 관측되는 자료로 온·습도, 풍향·풍속, 기압, 플럭스 요소를 포함하고 있습니다.

주요요소	기온, 습도, 풍향, 풍속, 기압, 플럭스
제공범위	보성 기상관측탑
제공기간	(기온, 습도, 풍향, 풍속) 2015.1.1. ~ 2023. 3. 26., 2026.1.19. ~ (기압) 2026.3.11. ~ (플럭스) 2015.1.1. ~ 2022.8.29., 2026.5.28. ~
생산기간	(기온, 습도, 풍향, 풍속, 기압) 1분 자료/ (플럭스) 30분 자료
제공형식	ASCII
제공경로	전자메일 또는 직접방문을 통한 자료제공 ※ 국립기상과학원(nims.go.kr) → 기상기후연구자료
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.

(생산종료)하계집중관측



개요

자료상태 : 생산종료

한반도에서 발생하는 위험기상 현상(호우, 대설, 장마 등)에 대한 이해 증진 및 예측 능력 향상을 위해 비정기적으로 실시한 라디오존데 집중관측 자료입니다.

주요요소	관측시간, 기압, 기온, 습도, 풍속, 풍향
제공범위	해마다 관측지점 상이
제공기간	2005년 6월 25일 ~ 2019년 8월 17일(관측요소별로 다름)
생산기간	일 자료
제공형식	ASCII
제공경로	전자메일 또는 직접방문을 통한 자료제공 ※ 국립기상과학원(nims.go.kr) → 기상기후연구자료
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.

2.연구용 관측 캠페인
- (생산종료)하계집중관측
- (생산종료)강원영동집중관측
- (생산종료)평창집중관측

2.연구용 관측 캠페인

• (생산종료)하계집중관측
• (생산종료)강원영동집중관측
• (생산종료)평창집중관측

(생산종료)강원영동집중관측

개요

자료상태 : 일부 현행

기상관측차량을 이용한 고층 및 지상관측을 수행하며, 특히 지상관측은 고정과 이동 관측 자료를 생산합니다. 위험기상 발생 시 현장으로 이동 또는 인근에서 관측하며, 재해기상 모니터링 및 분석에 활용합니다. 또한 운고계, 광학우적계 등을 통해서 운고, 운량, 강우강도 등을 관측합니다.

주요요소	GNSS 가강수량	가강수량
	운고계	운량, 운고, 연직시정
	광학우적계	강우강도, 강우 입자 크기, 강우 입자 수
	모바일기상관측차량 지상관측자료	기온, 상대습도, 풍속, 풍향, 순간풍속, 순간풍향, 현지기압
	모바일기상관측차량 고층관측자료	기온, 상대습도, 풍속, 풍향, 현지기압, 위도, 경도, 고도, 이슬점온도
	모바일기상관측차량 노면관측자료	노면온도, 노면상태, 마찰계수, 수막, 대기온도, 대기습도
제공범위	해마다 관측지점 상이	
제공기간	(GNSS 가강수량, 운고계, 광학우적계) 2022. ~ 2025. (장비별 상이) (모바일기상관측차량) 2013. ~ 현재 (장비별 상이)	
제공형식	ASCII	
제공경로	전자메일 또는 직접방문을 통한 자료제공 ※ 국립기상과학원(nims.go.kr) → 기상기후연구자료	
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.	

(생산종료)평창집중관측





개요

자료상태 : 생산종료

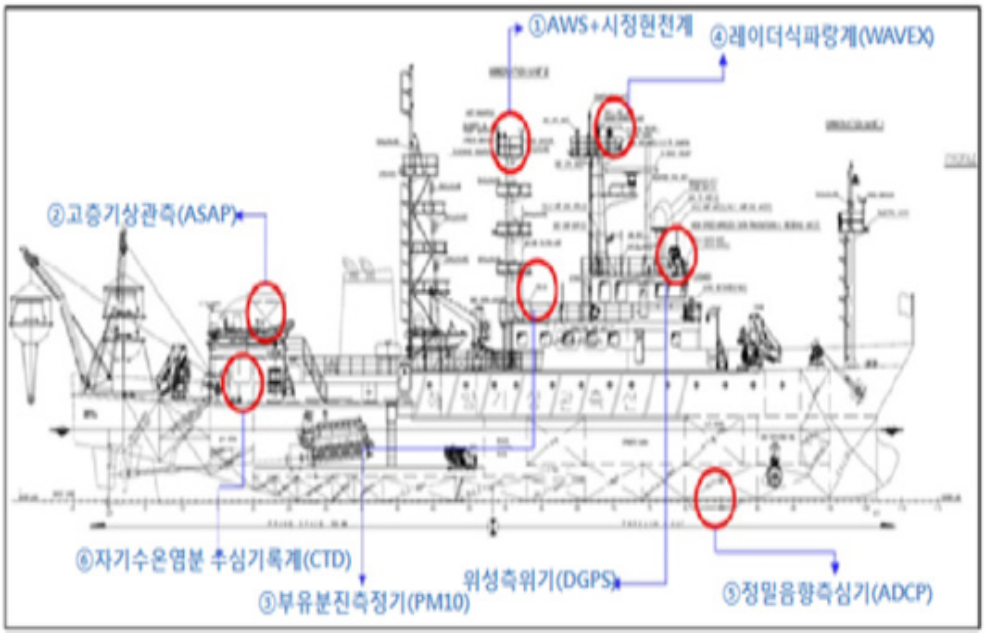
연직강우레이더, 광학우적계 등을 활용하여 위험기상 현상에 대한 강수구름 내의 강우강도, 액체수함량, 강우 입자 크기 등을 관측

주요요소	연직강우레이더	반사도, 강우강도, 낙하속도, 액체수함량
	광학우적계	강우강도, 강우 입자 크기, 강우 입자 수
	우량계	강수량
제공범위	해마다 관측지점 상이	
제공기간	2015.05. ~ 2018.05.	
제공형식	ASCII, BIN (관측장비마다 상이)	
제공경로	전자메일 또는 직접방문을 통한 자료제공 ※ 국립기상과학원(nims.go.kr) → 기상기후연구자료	
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.	

3.해양기상

• 기상관측선 관측
• 아르고 플로트 관측

기상관측선 관측



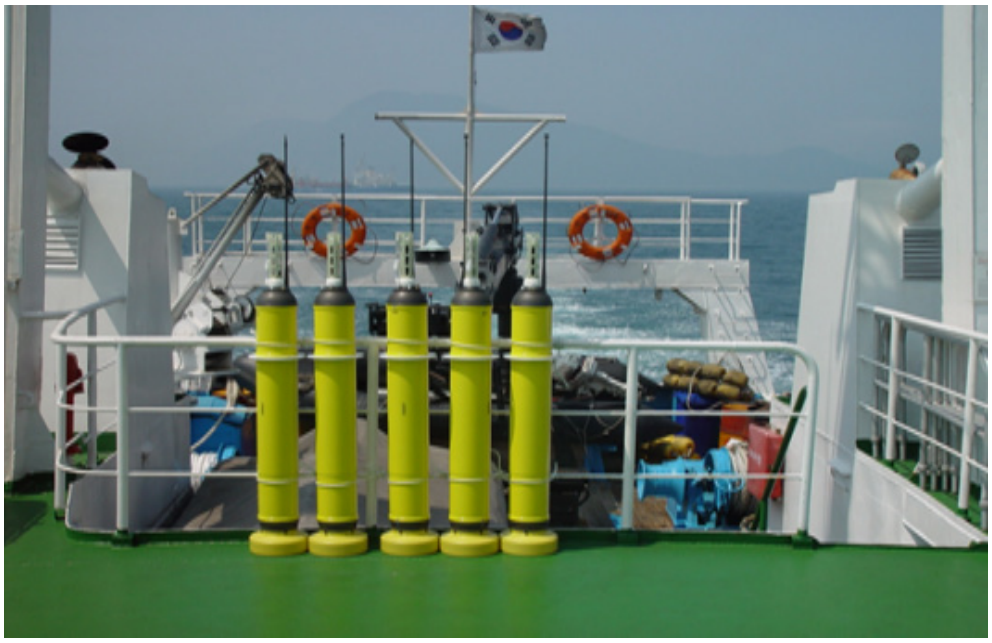
개요

자료상태 : 현행

기상관측선 기상1호는 2011년 5월 30일 취항하였습니다.
기상1호는 해양에서 고층·해상·해양·대기 환경등을 관측하는 이동기상대 역할을 수행하며, 위험기상 예상 시 관측효과를 극대화할 수 있는 지역에서 목표관측 및 장기간 고층관측위주의 연속적인 집중관측 업무를 수행 합니다.

주요요소	자동기상관측장비(AWS)	위경도, 파고, 파주기, 파향, 수온, 바람, 기온, 기압, 습도
	시정현천계(VPWS)	시정 및 현천(10m<범위<75km)
	고층기상관측장비(ASAP)	고도, 기온, 이슬점, 풍향풍속
	부유분진측정장비(PM10)	공기중 미세먼지 농도 측정
	레이더식 파랑계(WAVEX)	파고, 파향, 파주기
	초음파 해류관측(ADCP)	수층별 해류(유속, 유향, 700m)
	자기수온염분 수심기록계(CTD)	수압, 수온, 염분(3,000m)
제공범위	· 수평범위 : 서해, 남해, 동해 및 근해 구역 · 연직범위 : 수중 3,000m ~ 고층 20 km	
제공기간	2011년 6월 ~ 현재(관측 요소별로 다름)	
생산·갱신주기	시간, 일 자료	
제공형식	CSV	
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공	
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 관측연구부)	

아르고 플로트 관측



개요

자료상태 : 현행

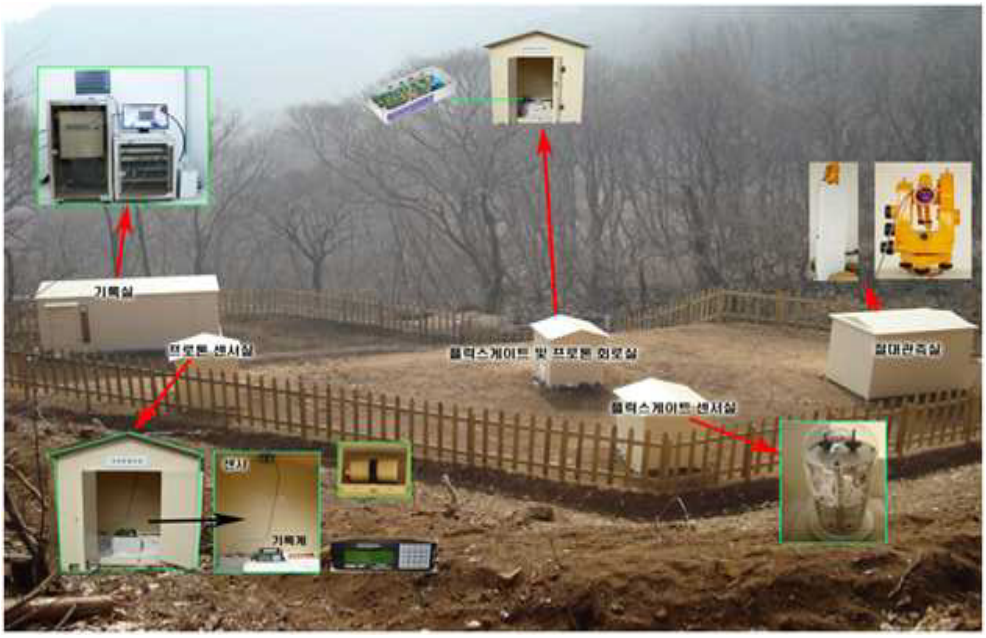
ARGO 플로트는 해양에서 일정한 수심까지 잠수하도록 설계되어 설정된 수심에서 해류를 따라 일정기간 표류하다가 플로트 내부의 동력에 의해 표층으로 부상하면서 수온과 염분을 연속적으로 관측하는 장비입니다. 관측된 자료는 위성을 통해 수신하여, 국립기상과학원의 실시간 품질관리를 거쳐 국제자료센터로 전송합니다.
국제자료센터에서 운영하는 FTP사이트를 통하여 ARGO 플로트 관측 자료를 제공하고 있습니다.

주요요소	수온, 염분, 용존산소
제공범위	· 2001년부터 매 년 지속적으로 투하 · 2019년 기준 총 247기 플로트 (동해 : 133기, 북서태평양 : 103기, 서해: 8기, 남해: 3기)
제공기간	2001년 11월 1일 ~ 현재 (지점별로 다름)
생산·갱신주기	수심 자료(7,10일 주기)
제공형식	NetCDF
제공경로	국립기상과학원 ARGO프로그램(http://argo.nimr.go.kr) 국제자료센터 FTP(ftp://ftp.ifremer.fr/ifremer/argo/dac)
이용방법	플로트 번호·관측기간·해역으로 프로파일을 조회합니다. 압력·수온·염분과 위치·시각, 품질표지(QC), 조정값(ADJUSTED) 및 자료모드(DATA_MODE)를 확인해 분석 목적에 맞는 값을 사용합니다.

4.지진화산

• 지구자기

지구자기



개요

자료상태 : 현행

청양 지구자기 관측소는 지구자기장 변동을 정밀하게 관측하고 국제 수준의 자료를 생산하기 위해 2009년 구축하였으며, 2013년 국내에서는 처음으로 국제지구자기관측망(INTERMAGNET: International Real-Time Magnetic Observatory Network)에 공식 가입·등록되었습니다. 관측점 위치의 지구자기 성분별 변화량을 생산하며, 한반도 지질구조 연구 및 우주기상 감시에 활용되는 표준 관측자료입니다.

주요요소	지구자기 성분별(X,Y,Z,F) 변동량
제공범위	청양 지구자기 관측소
제공기간	2009년 2월 ~ 현재
생산·갱신주기	초, 분, 일 자료
제공형식	ASCII
제공경로	국가지진종합정보시스템(http://necis.kma.go.kr) → 지구물리자료 → 지구자기장
이용방법	관측소·기간·자기장 성분을 선택합니다. 단위·시간분해능·성분 방향과 품질관리 상태를 확인하고 관측소 이동·장비 변경 이력을 고려합니다.

도시기상관측



개요

자료상태 : 현행

국립기상과학원에서 고해상도 도시기상서비스 기술개발을 위해 도시기상에 특화된 관측망을 구성하여 생산하는 관측자료입니다. 통합미기상, 도시에너지수지, 운고계, 라디오미터, 윈드라이다 등을 구축하여 운영하고 있습니다. 도시기상관측지점의 연구 목적에 따라 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동됩니다.

주요요소	기온, 풍향, 풍속, 습도, 기압, 에너지, 구름높이, 연직 바람, 온습도 프로파일 등
제공범위	수도권을 비롯한 전국 30개 지점
제공기간	2014년 ~ 현재(지점별로 다름)
생산·갱신 주기	분(1·2·5·10·60분) 자료 ※ 관측목적에 따라 관측자료 생산기간·갱신주기 상이
제공형식	CSV, NC파일 등 각 장비별 각 데이터 raw 형식
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

5.응용기상

• 도시기상관측
• 꽃가루농도 수동관측
• 산림연료·토양수분관측
• 농업서리자동관측
• 흑구온도

꽃가루농도 수동관측



개요

자료상태 : 현행

대기 중 알레르기 유발 꽃가루 농도 자료를 생산하고 있습니다. 관련 연구사업 정책에 근거한 해당 기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동될 수 있습니다.

주요요소	일별, 종류별 꽃가루 농도
제공범위	전국 12개 관측 지점 - 제공기준 · 본 연구자료는 비 영리 목적에 한하여 품질관리 및 품질검사 등의 과정을 거친 후 자료생산일로부터 최소 1년 후에 제공 가능(단, 특별관측 및 국제공동프로그램에 의해 생산된 자료는 별도 지정된 기간을 따름) · 기상청 및 그 소속기관과 국가연구개발사업을 수행 중인 기관에서 해당 사업 수행을 목적으로 연구자료를 요청할 경우에는 자료생산일로부터 1년 이내 자료도 제공 가능 · 영리 목적 사용인 경우 별도 협의 필요
제공기간	2018년 3월 ~ 현재
생산·갱신주기	일 자료 생산, 주 단위 갱신 ※ 관측지점별 관측자료 생산기간·갱신주기 상이
제공형식	CSV
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

산림연료·토양수분관측



개요

자료상태 : 현행

국가연구개발사업 관련법에 근거한 연구 활동에 의해 생산되는 자료로서 14개소의 산림토양 및 토양수분의 기상관측시설의 자료를 통합관측 및 제공하고 있습니다. 관련 연구사업 정책에 근거한 해당기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동될 수 있습니다.

주요요소	토양수분함유량, 전기전도도, 토양온도, 산림연료관습도, 산림연료계수분
제공범위	- 강원지역 14개 지점 ※ 기상관측업무를 수행하는 국가기관, 지자체, 공공기관 운영 기상관측시설과 대부분 연동 - 제공기준 · 공공데이터에 포함되지 않는 연구자료는 품질관리 및 품질검사 등의 과정을 거친 다음 자료생산일로부터 1년 후에 제공(단, 특별관측 및 국제공동프로그램에 의해 생산된 자료는 별도 지정된 기간을 따름) · 기상청 및 그 소속기관과 국가연구개발사업을 수행 중인 기관에서 해당 사업 수행을 목적으로 연구자료를 요청할 경우에는 자료생산일로부터 1년 이내 자료도 제공
생산·갱신주기	1시간 자료
제공형식	dat 파일 확장자
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

5.응용기상

- 도시기상관측
- 꽃가루농도 수동관측
- 산림연료·토양수분관측
- 농업서리자동관측
- 흑구온도

농업서리자동관측



개요

자료상태 : 현행

농업서리자동관측 시스템을 통한 서리 발생 자료를 생산하고 있습니다. 관련 연구사업 정책에 근거한 해당기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동될 수 있습니다.

주요요소	농업서리자동관측 시스템을 통한 농업서리 발생 유무
제공범위	전국 19개 관측 지점 - 제공기준 · 본 연구자료는 비 영리 목적에 한하여 품질관리 및 품질검사 등의 과정을 거친 후 자료생산일로부터 최소 1년 후에 제공 가능(단, 특별관측 및 국제공동프로그램에 의해 생산된 자료는 별도 지정된 기간을 따름) · 기상청 및 그 소속기관과 국가연구개발사업을 수행 중인 기관에서 해당 사업 수행을 목적으로 연구자료를 요청할 경우에는 자료생산일로부터 1년 이내 자료도 제공 가능 · 영리 목적 사용인 경우 별도 협의 필요
제공기간	2021년 8월 ~ 현재 ※ 관측지점별 관측자료 생산기간 상이
생산·갱신주기	1분, 10분, 1시간, 1일 자료생산 ※ 1시간 간격 갱신(매시): 1분, 10분, 1시간 자료 ※ 1일 간격 갱신(매일00시): 1일 자료
제공형식	dat
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

흑구온도



개요

자료상태 : 현행

흑구온도계에서 관측된 흑구온도 자료를 생산하고 있습니다. 관련 연구사업 정책에 근거한 해당기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동될 수 있습니다.

주요요소	흑구온도
제공범위	전국 12개 관측 지점 - 제공기준 · 본 연구자료는 비 영리 목적에 한하여 품질관리 및 품질검사 등의 과정을 거친 후 자료생산일로부터 최소 1년 후에 제공 가능(단, 특별관측 및 국제공동프로그램에 의해 생산된 자료는 별도 지정된 기간을 따름) · 기상청 및 그 소속기관과 국가연구개발사업을 수행 중인 기관에서 해당 사업 수행을 목적으로 연구자료를 요청할 경우에는 자료생산일로부터 1년 이내 자료도 제공 가능 · 영리 목적 사용인 경우 별도 협의 필요
제공기간	2024년 12월 ~ 현재 ※ 관측지점별 관측자료 생산기간 상이
생산·갱신주기	1분, 10분
제공형식	dat
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

6.구름물리

• 구름물리관측
• 챔버실험(구름물리챔버관측)

구름물리관측



개요

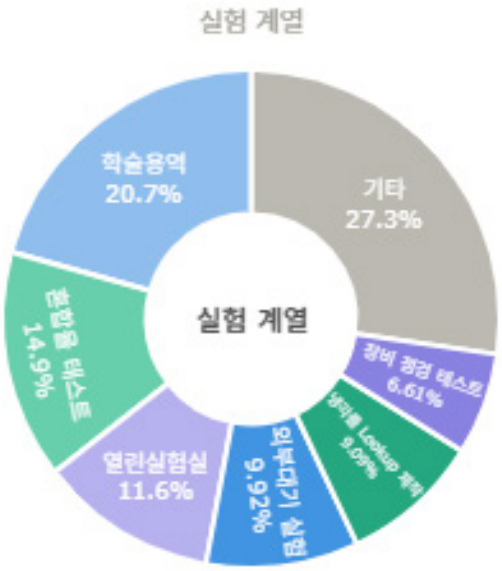
자료상태 : 현행

국가연구개발사업 관련법에 근거한 연구 활동에 의해 생산되는 자료로서 25개소의 구름물리 연구용 기상관측시설의 자료를 통합관측 및 제공하고 있습니다. 관련 연구사업 정책에 근거한 해당기상관측시설의 신규·폐지·이전·교체에 따라 제공 대상 지점이 변동될 수 있습니다.

주요요소	기온, 풍향, 풍속, 습도, 기압, 강수량, 강수 입자 및 수직 반사도, 액체 수함량 등
제공범위	- 강원지역 및 서해안을 포함한 25개 지점 ※ 기상관측업무를 수행하는 국가기관, 지자체, 공공기관 운영 기상관측시설과 대부분 연동 - 제공기준 · 공공데이터에 포함되지 않는 연구자료는 품질관리 및 품질검사 등의 과정을 거친 다음 자료생산일로부터 1년 후에 제공(단, 특별관측 및 국제공동프로그램에 의해 생산된 자료는 별도 지정된 기간을 따름) · 기상청 및 그 소속기관과 국가연구개발사업을 수행 중인 기관에서 해당 사업 수행을 목적으로 연구자료를 요청할 경우에는 자료생산일로부터 1년 이내 자료도 제공
생산·갱신주기	분(1·2·5·10·60분) 자료 ※ 관측기관별 관측목적에 따라 관측자료 생산주기 차이
제공형식	CSV, 각 데이터 raw 형식
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

챔버실험(구름물리챔버관측)

6.구름물리



개요

자료상태 : 현행

- 한국 구름물리실험챔버(K-CPEC)는 저온, 저압, 고압, 포화도 등 실제 대기 구름환경을 실험챔버에서 재현하여 구름 입자와 빙정의 생성, 응집 등에 대한 특성을 정량적으로 평가합니다.
- 한국 구름물리실험챔버(K-CPEC)은 2023년 시험 운영을 시작으로 기초 운영 기술을 확보하였고, '26년 현재 다양한 구름 물리 실험을 통해 표준 운영 체계를 마련 중입니다.
- 현재 생산 자료는 챔버 실험 목적, 상태 정보 등 기초자료이며, 향후 단계별 실험자료를 제공할 계획입니다.

주요요소	월 실험 운영 기초 정보(운영 일, 목적, 챔버 상태 정보)
제공범위	월별 운영 기초 정보
생산·갱신 주기	월별 운영(운영 계획), 실험 수행 기초정보 갱신
제공형식	CSV
제공경로	공공데이터포털(https://www.data.go.kr) → 공공데이터 → 공공데이터 제공신청 수동제공신청 접수 후 우편 등을 통한 저장매체의 물리적 이동에 따른 자료 제공
이용방법	관측 또는 실험의 대상·기간·위치·요소·자료주기를 정하여 보유부서에 제공 가능 및 이용방법을 문의합니다.(국립기상과학원 기상응용연구부)

11

기후지진통계

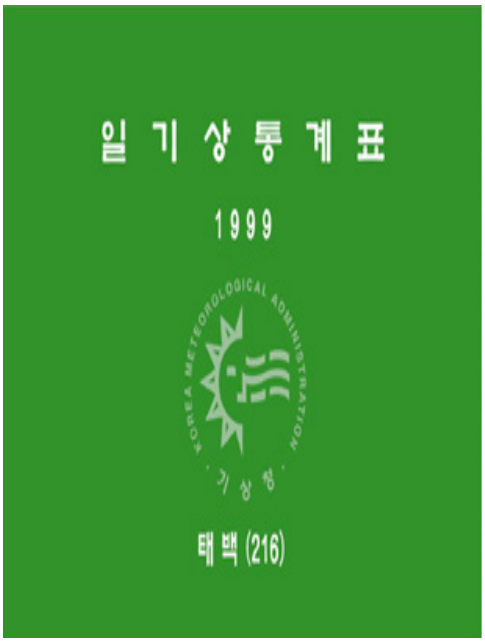
Climate and Earthquake Statistics

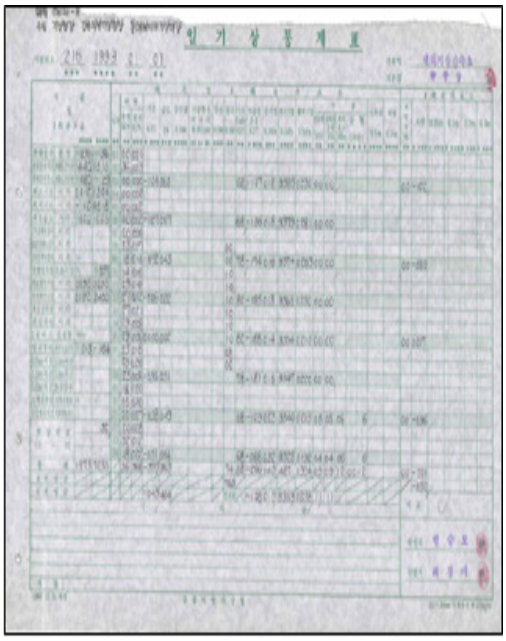


1.통계연월보

일기상통계표

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보





개요

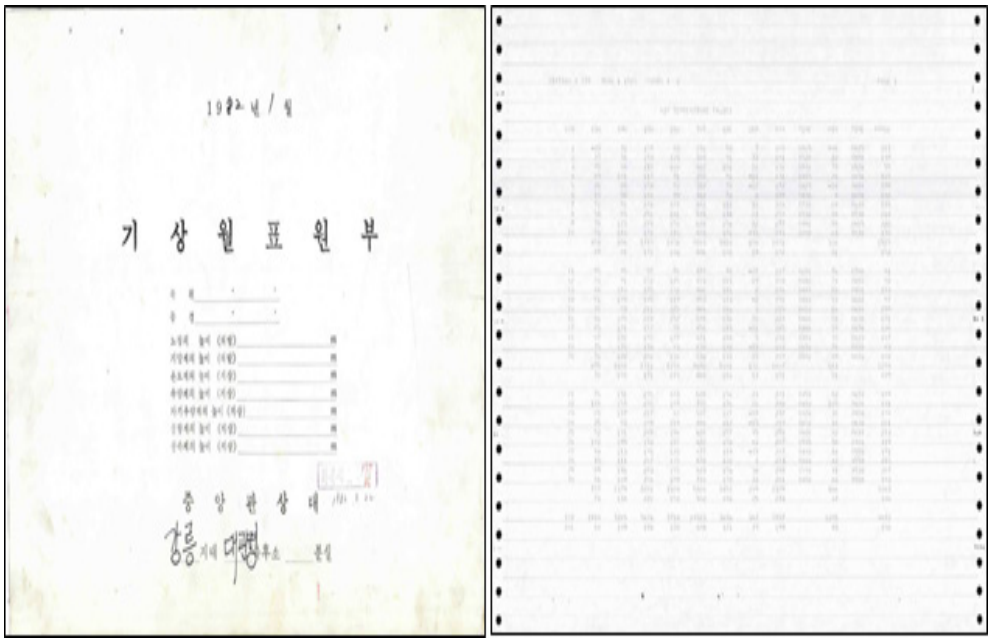
자료상태 : 생산종료

기상통계표란 기상요소에 따라 일 1회, 4회, 8회, 매 시각 등 해당 관서의 시간별 기상요소를 기입한 자료입니다.

주요요소	매시풍향 · 풍속 · 일사 · 일조 · 강수량, 3시간별 기온 · 습도 · 시정 · 이슬점온도 · 증기압 · 현지기압 · 해면기압 · 구름, 6시간별 지면온도 · 지중온도, 일최대풍향 · 풍속, 일최대순간풍향 · 풍속, 최고기온 · 최저기온, 최소습도, 최저초상온도, 증발량, 일기상현상 등
제공범위	110지점
제공기간	1962년 ~ 1999년(지점별로 다름)
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 통계원부류 → 일기상통계표
이용방법	관측관서·연월과 원부 종류를 선택하여 스캔 또는 제공 파일을 조회합니다. 당시 관측시간·요소·통계방법·단위와 수기 보정 기록을 확인하여 수치를 옮깁니다.

기상월표원부

1.통계연월보
- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보



개요

자료상태 : 생산종료

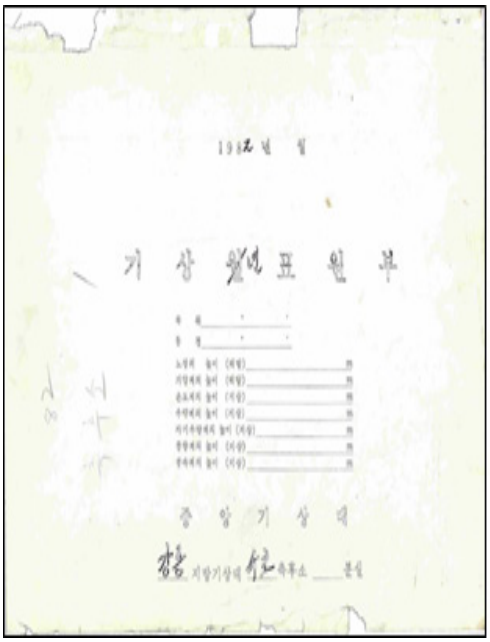
기상월표원부는 기상관측야장과 여러 종류의 자기기록지를 바탕으로 기상요소별, 계급별, 현상별, 기사란 등을 정리하여 월 통계와 극값·순위값 등을 작성한 자료입니다.

주요요소	매시풍향·풍속·일사·일조·강수량·시간별 기온·습도·시정· 이슬점온도·증기압·현지기압·해면기압·구름·지면온도·지중온도, 일최대풍향·풍속·최고기온·최저기온·최고기압 ·최저기압·일최다강수량, 최소습도, 적설, 증발량, 일기상현상 등
제공범위	66지점(1982년 기준)/총 96개 지점
제공기간	1904년 ~ 1985년(지점별로 다름)
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 통계원부류 → 월표원부
이용방법	관측관서·연월과 원부 종류를 선택하여 스캔 또는 제공 파일을 조회합니다. 당시 관측시간·요소·통계방법·단위와 수기 보정 기록을 확인하여 수치를 옮깁니다.

1.통계연월보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보

기상연표원부





개요

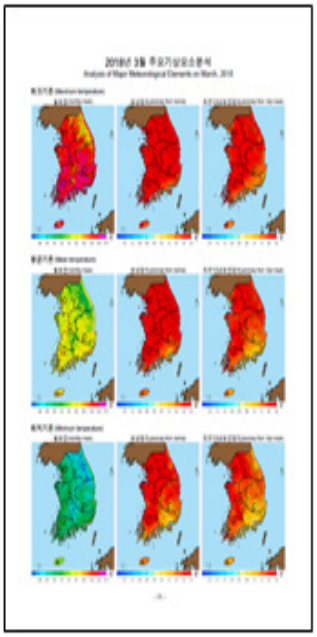
자료상태 : 생산종료

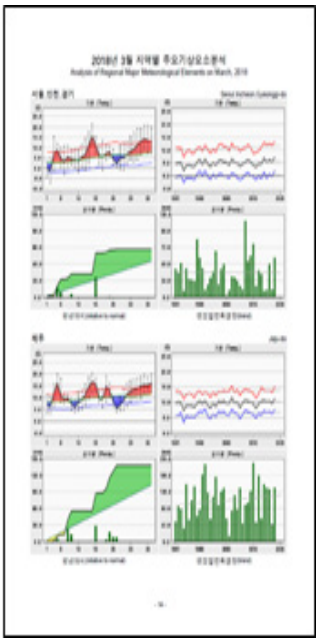
기상연표원부는 기상월표원부를 바탕으로 연 통계와 극값, 순위값 등을 생산하였으며, 기상월보와 연보의 기초자료로 활용하였습니다.

주요요소	월별 기상관측 자료: 기온, 습도, 증기압, 이슬점온도, 현지기압, 해면기압, 강수량, 증발량, 적설, 풍향, 풍속, 운량, 안개계속시간, 지중온도, 일조 등
제공범위	73지점(1982년 기준)
제공기간	1931년 ~ 1982년(지점별로 다름)
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 통계원부류 → 연표원부
이용방법	관측관서·연월과 원부 종류를 선택하여 스캔 또는 제공 파일을 조회합니다. 당시 관측시간·요소·통계방법·단위와 수기 보정 기록을 확인하여 수치를 옮깁니다.

기상월보







자료상태 : 현행

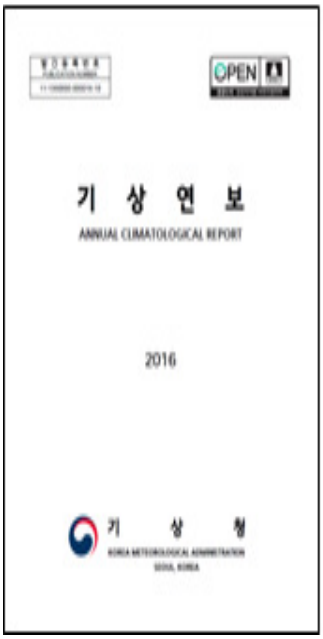
기상월보에서 매달 전국 97개 관측 지점의 일별 기상자료와 순·월 요약자료와 함께 주요기상 요소 분석, 극값순위, 바람장미, 저기압 및 태풍경로도 등을 확인 할 수 있으며, 세계주요도시 기후자료를 확인할 수 있습니다.

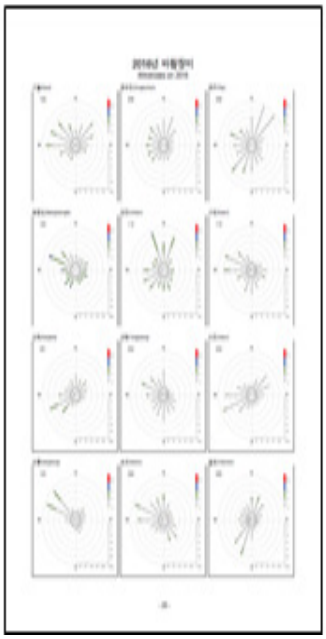
자료처리	월 통계 : 기온, 기압, 습도, 강수량, 증발량, 바람, 구름, 일조, 일사 등	
제공기간	1913년 1월 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 통계원부류 → 기상월보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 기상월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

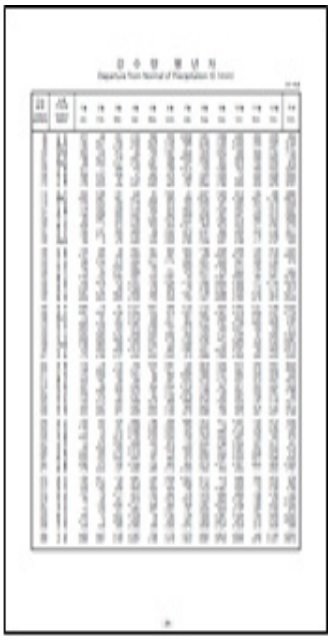
1.통계연월보

기상연보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보







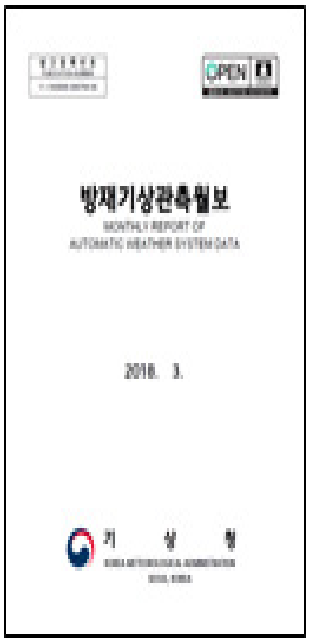
개요

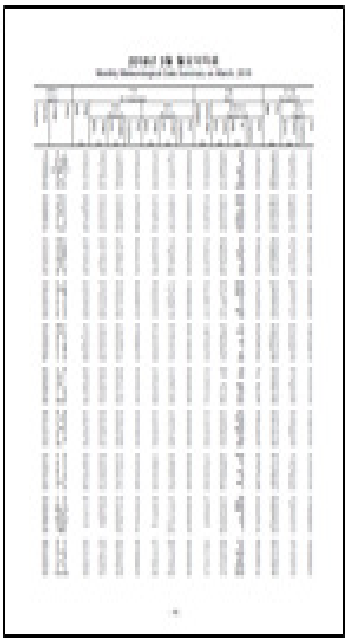
자료상태 : 현행

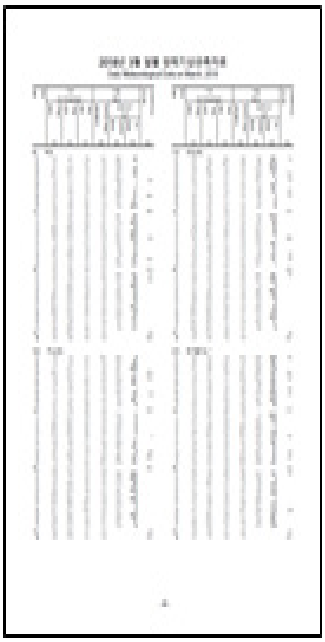
기상연보에서 지난 해 전국 97개 관측 지점의 월별 기상자료와 연 요약자료와 함께 주요기상 요소 분석, 월별 기상개요, 기후기록, 기상재해, 바람장미, 우리나라에 영향을 미친 태풍, 평균기온 및 강수량 편차 등을 확인할 수 있습니다.

자료처리	연 통계: 기온, 기압, 증기압, 습도, 강수량, 증발량, 바람, 구름, 일조, 일사 등	
제공기간	1904년 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 통계원부류 → 기상연보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 기상연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

방재기상관측월보







자료상태 : 현행

방재기상관측월보에서 방재기상관측장비(AWS)로 관측된 기온, 바람, 강수량의 월 통계 자료와 일 통계 자료를 확인할 수 있습니다.

※ 1993년 1월분부터 자동기상관측월보를 발간하기 시작하였고, 2010년 1월분부터 방재기상관측 월보로 명칭이 변경됨

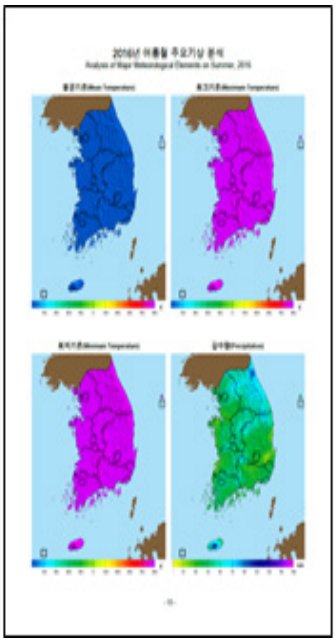
주요요소	월 통계 : 기온, 바람, 강수량	
제공기간	1993년 1월 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 방재간행물 → 방재기상관측월보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 방재기상관측월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

1.통계연월보

방재기상관측연보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보







개요

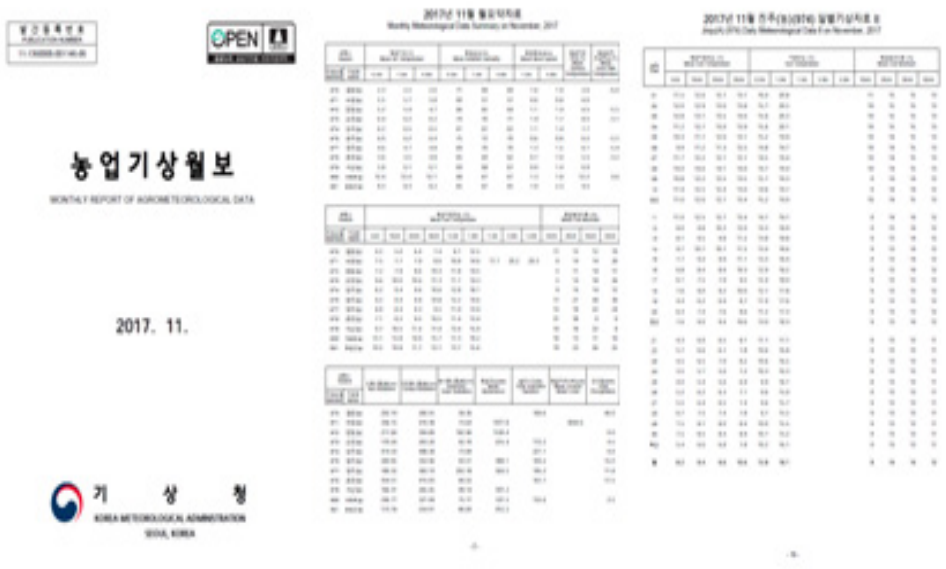
자료상태 : 현행

방재기상관측연보에서 방재기상관측장비(AWS)로 관측된 기온, 바람, 강수량의 연 통계 자료와 월 통계 자료를 확인할 수 있습니다.

※ 1993년분부터 자동기상관측연보를 발간하기 시작하였고, 2011년분부터 방재기상관측연보로 명칭이 변경됨
※ 기상청은 2017년 9월 1997년~2015년 방재기상관측연보를 개정함. 「2017년 개정판」은 국가기후자료 시스템(CDMS)에서 품질관리를 수행한 AWS 분자료를 개선된 통계산출 방식을 적용하여 재산출함.

주요요소	연 통계: 기온, 바람, 강수량	
제공기간	1993년 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 방재간행물 → 방재기상관측연보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 방재기상관측연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

농업기상월보



자료상태 : 생산종료

농업기상월보는 전국 11개 지점의 농업기상관측장비(AAOS, Automated Agricultural Observing System)로 관측된 기온, 지면·초상·지중온도, 토양수분, 복사 등 총 30개 요소의 농업기상관측자료를 수록한 정기간행물로 2015년부터 품질관리를 적용하여 일, 순, 월의 농업기후자료를 확인할 수 있습니다.

※ 서비스 대상: 11지점의 AAOS 관측자료(2015년 7월 ~ 2018년 1월)

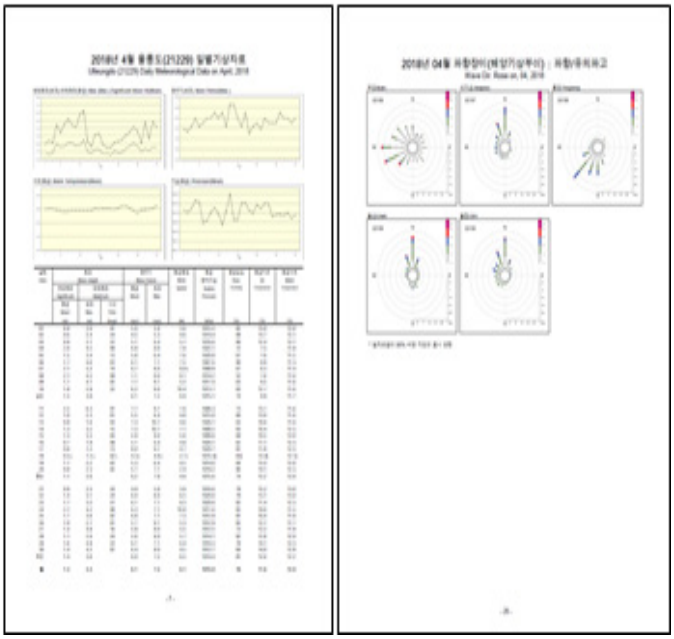
※ 농업기상관측 효율화 방안에 따라 농업기상관측장비(AAOS)를 종관(ASOS) 또는 방재(AWS) 기상관측장비와 통합·운영하여 2018년 2월분부터 농업기상월보 생산을 잠정 중단

주요요소	월 통계 : 기온, 습도, 풍속, 지면온도, 초상온도, 지중온도, 토양수분, 순복사, 전천복사, 반사복사, 조도, 일조시간, 강수량, 지하수위
제공기간	2015년 7월 ~ 2018년 1월
제공형식	E-BOOK, PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 농업간행물 → 농업기상월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

1.통계연월보 해양기상월보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보


해 양 기 상 월 보
MONTHLY REPORT OF MARINE DATA
2017. 1.
기 상 청



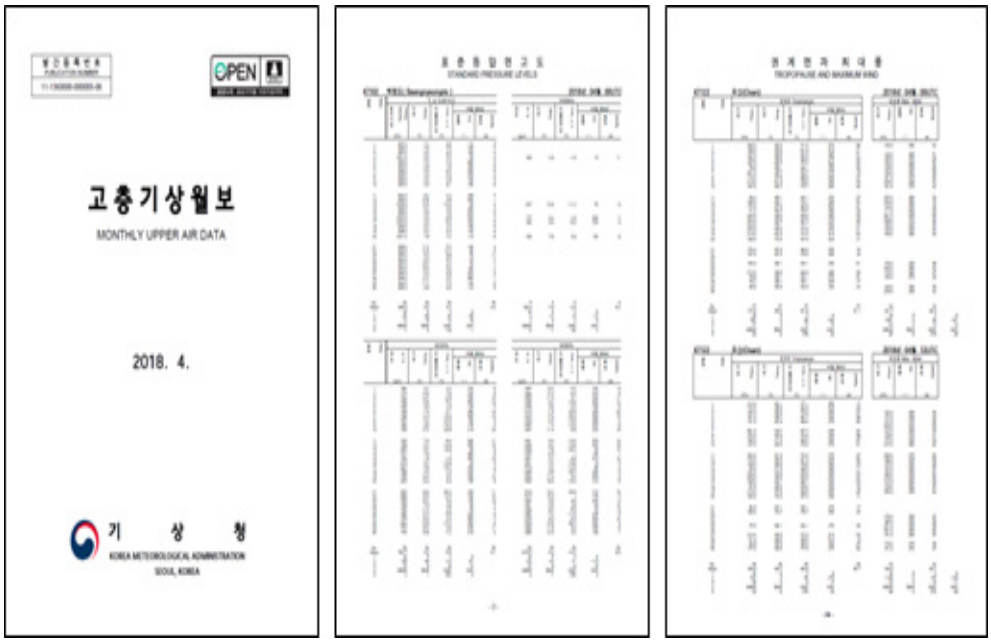
개요

자료상태 : 현행

해양기상월보에서 해양기상부이와 등표기상관측장비로 관측된 9개 요소에 대한 시간별 자료와 일통계 및 월통계를 확인할 수 있습니다.

주요요소	월 통계: 파고, 파주기, 파향, 풍향, 풍속, 기압, 습도, 기온, 수온 등	
제공범위	• 해양기상부이(23소): 울릉도, 덕적도, 칠발도, 거문도, 거제도, 동해 등 • 등표기상관측장비(8소): 서수도, 가대암, 십이동파, 갈매여, 해수서, 지귀도 등 • 파고부이(62소): 독도, 혈암, 구암, 신진도, 삽시도, 옥도, 진도, 연곡, 죽변 등 ※ 파고부이는 지점 수시 변동가능	
제공기간	2012년 1월 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 해양 → 해양간행물 → 해양기상월보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 해양기상월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

고층기상월보



자료상태 : 현행

고층기상월보에서 기상청과 공군에서 관측하여 보정된 일별 고층 기상자료를 확인할 수 있으며, UTC시각을 사용합니다.

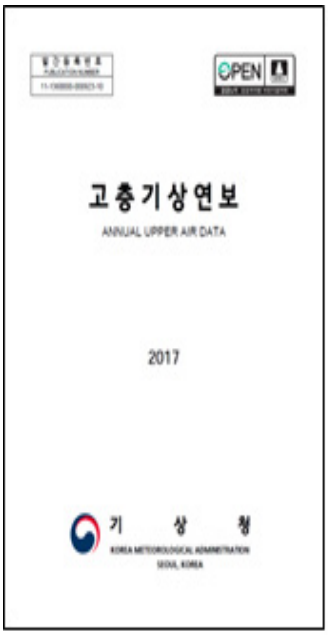
표준등압면 고도는 지상, 1000, 925, 850, 700, 500, 400, 300, 250, 150, 100, 70, 50, 30, 20, 10hPa입니다.

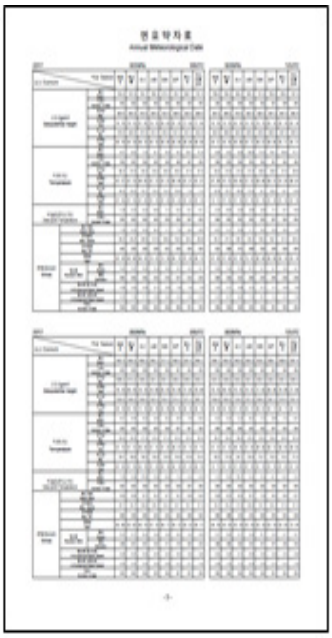
주요요소	월 통계: 기압, 기온, 이슬점온도, 풍향, 풍속	
제공범위	8지점(백령도, 북강릉, 오산, 포항, 창원, 광주, 흑산도, 국가태풍센터)	
제공기간	2011년 1월 ~ 현재(제공형식별로 다름)	
제공형식	PDF, 엑셀(xls), E-BOOK, XML	
제공경로	PDF	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 고층 → 고층간행물 → 고층기상월보
	엑셀(xls)	
	E-BOOK	
	XML	기상자료개방포털(https://data.kma.go.kr) → 오픈API → 고층기상월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.	

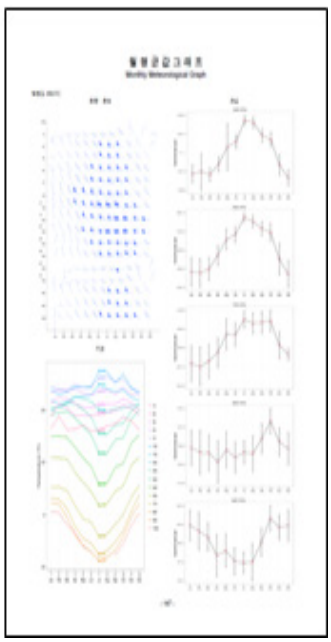
1.통계연월보

고층기상연보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보







개요

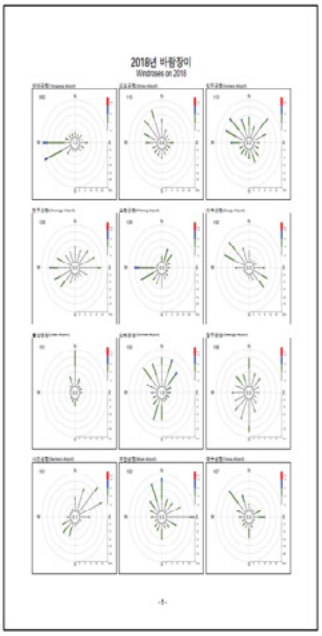
자료상태 : 현행

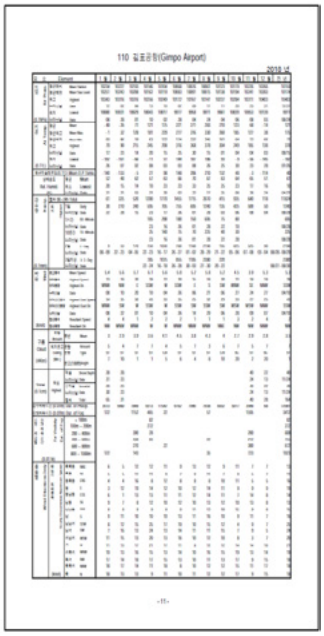
고층기상연보에서 기상청과 공군에서 관측하여 보정된 월별 고층 기상자료와 연 요약자료를 확인할 수 있으며, UTC시각을 사용합니다.
표준등압면 고도는 지상, 1000, 925, 850, 700, 500, 400, 300, 250, 150, 100, 70, 50, 30, 20, 10hPa입니다.

주요요소	연 통계: 고도, 기온, 이슬점온도, 풍향, 풍속
제공범위	8지점(백령도, 북강릉, 오산, 포항, 창원, 광주, 흑산도, 국가태풍센터)
제공기간	2014년 ~ 현재(제공형식별로 다름)
제공형식	PDF, E-BOOK
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 고층 → 고층간행물 → 고층기상연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

항공기상월보







자료상태 : 현행

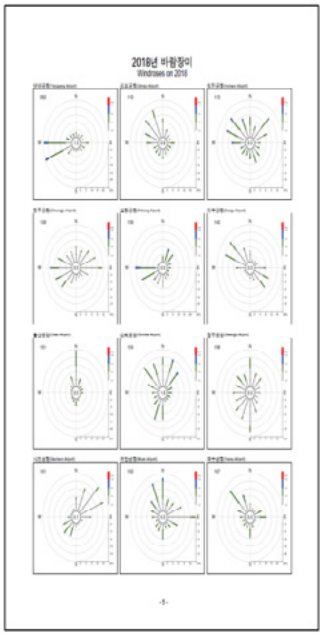
항공기상월보에서 13개 항공기상관측지점에서 관측한 일별기상자료와 바람장미를 확인할 수 있습니다.

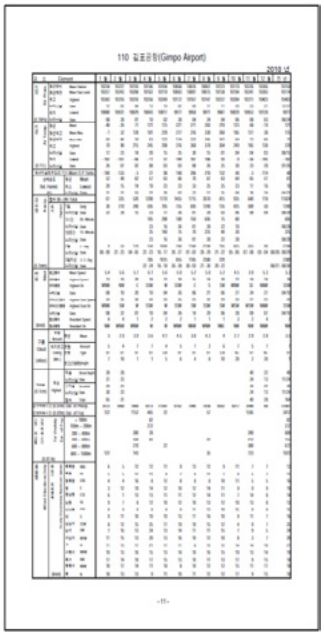
주요요소	월 통계: 기압, 기온, 이슬점온도, 상대습도, 바람, 강수량 등
제공범위	13지점(양양공항, 김포공항, 인천공항, 청주공항, 포항공항, 대구공항, 울산공항 등)
제공기간	1988년 1월 ~ 현재(제공형식별로 다름)
제공형식	PDF, E-BOOK
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 항공 → 항공기상연월보 → 항공기상월보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

1.통계연월보 항공기상연보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- **항공기상연보**
- 낙뢰연보
- 북한기상연보
- 지진연보







개요

자료상태 : 현행

항공기상연보에서 13개 항공기상관측지점에서 관측한 월별기상자료와 연요약자료, 바람장미를 확인할 수 있습니다.

주요요소	연 통계: 기압, 기온, 이슬점온도, 상대습도, 바람, 강수량 등
제공범위	13지점(양양공항, 김포공항, 인천공항, 청주공항, 포항공항, 대구공항, 울산공항 등)
제공기간	2004년 ~ 현재(제공형식별로 다름)
제공형식	PDF, E-BOOK
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 항공 → 항공기상연월보 → 항공기상연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

낙뢰연보



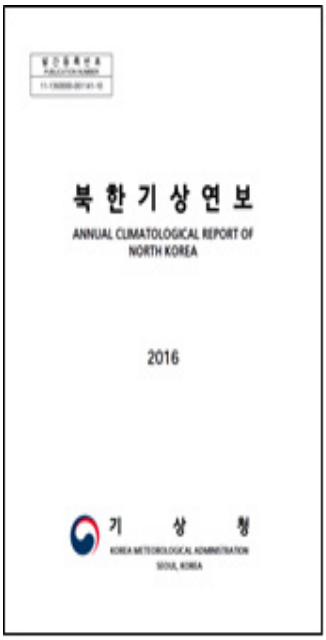
자료상태 : 현행

낙뢰연보는 당해 연도 낙뢰발생 횟수, 전국(광역시 · 도, 시 · 군 · 구) 낙뢰발생 횟수, 주요 낙뢰사례 분석 등을 확인할 수 있습니다.

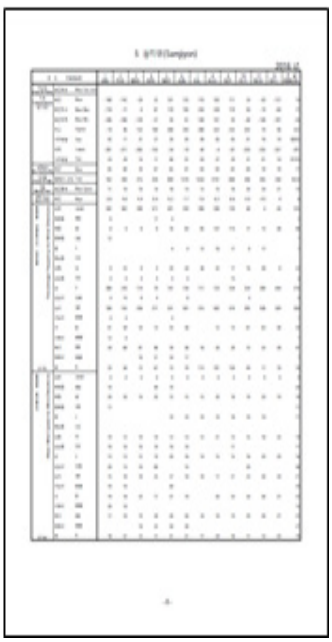
자료처리	기상청 관측 낙뢰에 대한 당해연도 지역별, 월별 발생횟수 등 통계
제공기간	1996년 ~ 현재
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 레이더 → 낙뢰연보 기상청 행정 누리집(https://kma.go.kr/kma) → 알림·자료 → 자료 → 기상간행물 → 낙뢰연보 기상레이더센터 누리집(https://radar.kma.go.kr) → 알림 → 자료실 → 간행물
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

1.통계연월보 북한기상연보

- 일기상통계표
- 기상월표원부
- 기상연표원부
- 기상월보
- 기상연보
- 방재기상관측월보
- 방재기상관측연보
- 농업기상월보
- 해양기상월보
- 고층기상월보
- 고층기상연보
- 항공기상월보
- 항공기상연보
- 낙뢰연보
- 북한기상연보
- 지진연보







개요

자료상태 : 현행

북한기상연보에서 세계기상기구(WMO)의 기상통신망(GTS)을 통해 입수한 북한 27개 지점의 기압, 기온, 강수량, 바람, 습도, 운량 등 총 6개 요소의 기후통계자료를 확인할 수 있습니다.

주요요소	연 통계: 기압, 기온, 강수량, 바람, 습도, 운량 등
제공범위	27지점
발표시각	해당 연도 이후 6개월 이내
제공기간	2014년 ~ 현재
생산·갱신주기	연 자료(매년)
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 북한간행물 → 북한기상연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

지진연보



자료상태 : 현행

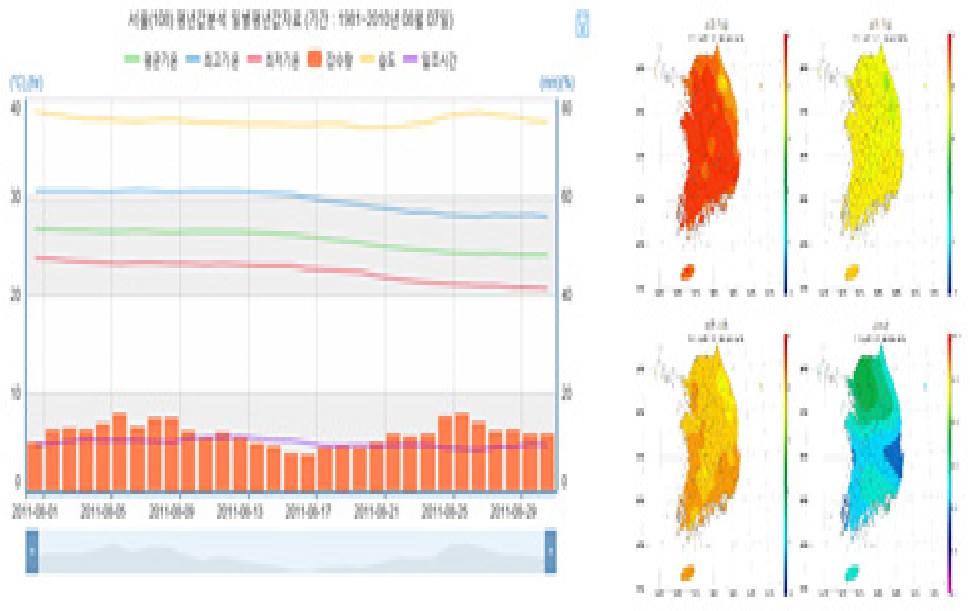
지진연보에서 지진발생현황, 지진관련 대응체계, 주요 연구 성과, 그 해의 주요 지진 정밀 분석 등을 확인할 수 있습니다.

자료처리	규모 3이상 지진의 연별, 지역별 통계
제공기간	2001년 ~ 현재
생산·갱신주기	년 자료
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지진 → 지진연보
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

2.평년값

• 우리나라 기후평년값
• 북한 기후평년값
• 세계 기후평년값

우리나라 기후평년값



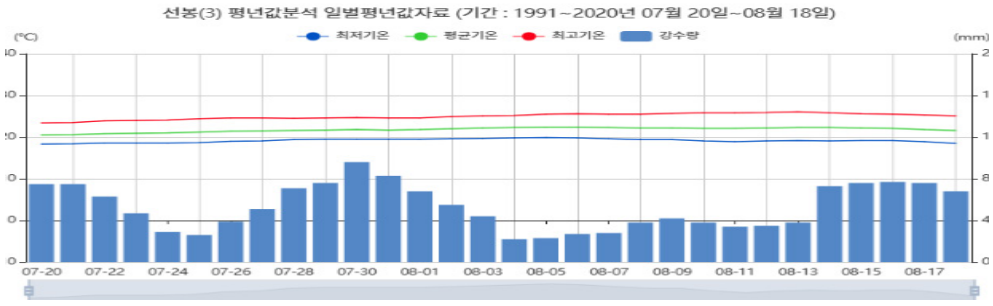
개요

자료상태 : 현행

평년값은 해당 지점을 대표하는 기후값으로 매 10년마다 이전 30년에 대해 산출하는 값입니다. 평년값 산출 대상 기간은 기본적으로 30년이지만, 10년 이상 연속적으로 관측된 자료가 존재하는 경우에도 산출할 수 있습니다. 현재 사용하고 있는 평년값은 2021년에 산출된 1991년~2020년에 대한 평년값입니다. 기상청에서는 2031년에 2001년~2030년에 대해 평년값을 산출할 예정입니다.

주요요소	평년값(기온, 강수량)
제공범위	연대별로 지점 수 다름(1991년 ~ 2020년 평년값: 219지점)
제공기간	1961년~1990년, 1971년~2000년, 1981년~2010년, 1991년~2020년
생산·갱신주기	• 그래프: 일, 월 자료 • 분포도: 일, 월, 년 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 평년값 → 우리나라 기후평년값
이용방법	기준기간·관측지점·통계요소와 일·순·월 등 집계 구분을 선택합니다. 통계방법·결측처리와 단위를 확인하고, 다른 평년기간의 값은 구분하여 비교합니다.

북한 기후평년값



지점명	일시	평균기온(℃)	최고기온(℃)	최저기온(℃)	강수량(mm)	평균풍속(m/sec)	습도(%)	전운량(1/10)
선봉	07-20	20.5	23.4	18.3	7.5	1.6	90.9	8.7
선봉	07-21	20.6	23.5	18.4	7.5	1.6	90.5	8.6
선봉	07-22	20.8	23.9	18.6	6.3	1.6	89.5	8.5
선봉	07-23	20.9	24.0	18.6	4.7	1.5	89.2	8.3
선봉	07-24	21.0	24.1	18.6	2.9	1.5	89.4	8.3
선봉	07-25	21.2	24.4	18.7	2.6	1.5	89.5	8.3
선봉	07-26	21.4	24.6	19.0	3.9	1.5	89.8	8.4
선봉	07-27	21.5	24.6	19.1	5.1	1.5	90.6	8.5
선봉	07-28	21.6	24.5	19.4	7.1	1.5	91.2	8.8
선봉	07-29	21.7	24.6	19.5	7.6	1.5	90.9	8.9

개요

자료상태 : 현행

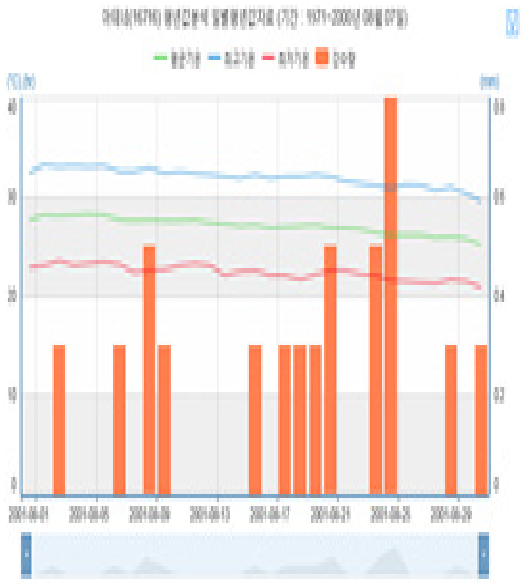
평년값은 해당 지점을 대표하는 기후값으로 매 10년마다 이전 30년에 대해 산출하는 값입니다. 현재 사용하고 있는 평년값은 2021년에 산출된 1991년~2020년에 대한 평년값입니다. 기상청에서는 2031년에 2001년~2030년에 대해 평년값을 산출할 예정입니다.

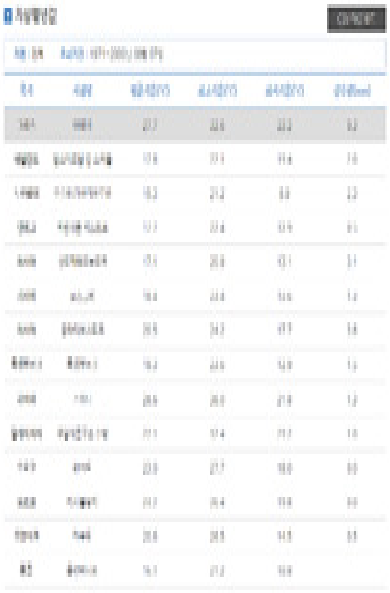
주요요소	평년값(기온, 기압, 강수량 등)
제공범위	27지점
제공기간	1973년~1994년, 1981년~2010년, 1991년~2020년
생산·갱신주기	일, 월 자료 ※ 평년값 기간 중 1973~1994년은 일별 자료만 제공
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 평년값 → 북한 기후평년값
이용방법	기준기간·관측지점·통계요소와 일·순·월 등 집계 구분을 선택합니다. 통계방법·결측처리와 단위를 확인하고, 다른 평년기간의 값은 구분하여 비교합니다.

2.평년값

• 우리나라 기후평년값
• 북한 기후평년값
• 세계 기후평년값

세계 기후평년값





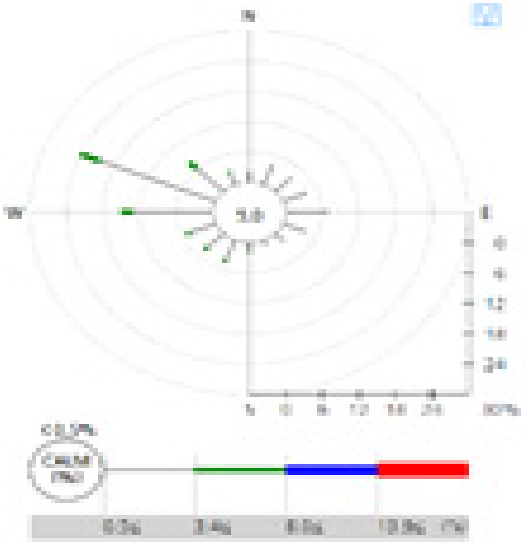
개요

자료상태 : 현행

기상자료개방포털에서 제공하고 있는 세계기후평년값은 기상청에서 자체적으로 모든 지점에 대해 동일한 기준을 적용해 분석·생산한 값으로(1991년, 2001년에 각각 1961년~1990년, 1971년~2000년을 대상으로 작성) 각 국 기상청에서 제공하고 있는 값과 다를 수 있음을 안내해드립니다.

주요요소	평년값(기온, 강수량)
제공범위	103개 지점
제공기간	1961년~1990년, 1971년~2000년, 1980~2010년, 1998~2010년, 2002~2010년, 2011~2020년
생산·갱신주기	10년
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 평년값 → 세계기후평년값
이용방법	기준기간·관측지점·통계요소와 일·순·월 등 집계 구분을 선택합니다. 통계방법·결측처리와 단위를 확인하고, 다른 평년기간의 값은 구분하여 비교합니다.

바람장미



개요

자료상태 : 현행

바람장미는 관측지점의 어느 기간에 대하여 각 방위별 풍향 출현 빈도를 방사 모양의 그래프에 나타낸 것으로, 출현빈도의 백분율(%)을 각각의 풍향에 대응하는 방위판 위에 방위선의 길이로 나타냅니다. 단, 정온(Calm, 풍속 0.5m/s 미만)의 출현빈도는 그래프의 중심에 표시합니다. 바람장미에는 각 풍향의 풍속 계급별 빈도도 함께 나타납니다.

주요요소	각 방위별 풍향 출현 빈도
제공범위	종관기상관측 전지점(폐쇄지점 포함), 방재기상관측지점 전지점(현재 운영중인 지점번호)
제공기간	지점별 관측개시일 ~ 현재
생산·갱신주기	일, 월, 계절, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 계급별일수 → 바람계급별일수(바람장미)
이용방법	지점·기간·풍속 계급을 선택해 바람장미 또는 계급별 일수를 조회합니다. 풍향은 바람이 불어오는 방향으로 해석하고 무풍의 처리와 빈도 분모를 확인합니다.

3.통계분석

• 바람장미
• 강수일수
• 황사일수
• 폭염일수
• 열대야일수
• 24절기
• 순위값
• 장마

강수일수



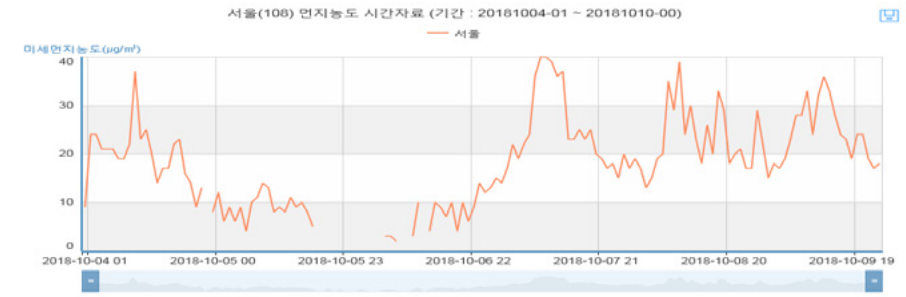
개요

자료상태 : 현행

강수일수는 일강수량이 0.1mm 이상인 날의 수입니다.

주요요소	강수일수
제공범위	종관기상관측 전지점(폐쇄지점 포함)/지역(전국 45지점+제주 2지점)
제공기간	지점별 관측개시일 ~ 현재/지역: 1973년 ~ 현재
생산·갱신주기	월, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 기상현상일수 → 강수일수
이용방법	지점·연도·월 또는 계절을 선택해 현상일수를 조회합니다. 현상별 정의와 집계기간을 확인하고, 결측이나 미관측을 발생일수 0으로 처리하지 않습니다. 강수량 문턱값을 확인합니다.

황사일수



미세먼지 농도 CSV 다운로드

자료구분 : 시간 조회기간 : 20181004 ~ 20181010
지점 : 서울 전체 158건 10

지점번호	지점명	시간	미세먼지 농도
108	서울	2018-10-04 01	9
108	서울	2018-10-04 02	24
108	서울	2018-10-04 03	24
108	서울	2018-10-04 04	21
108	서울	2018-10-04 05	21
108	서울	2018-10-04 06	21

개요

자료상태 : 현행

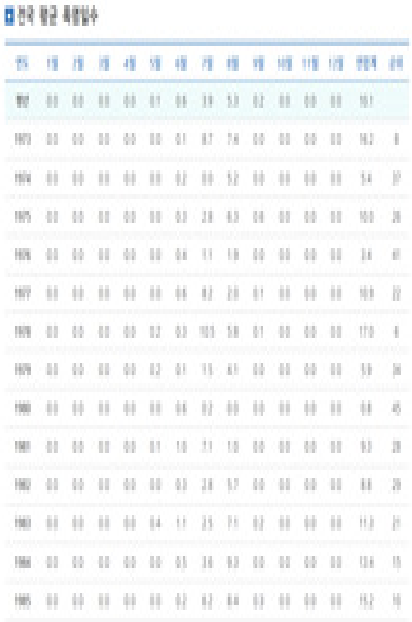
관측기준에 따라 황사가 관측된 날을 집계한 통계자료입니다. PM10 농도가 일황사일수는 황사 현상이 나타난 일수이며, 먼지농도는 부유분진측정기(PM10)로 관측한 PM10 질량농도(㎍/m^3)입니다.
황사는 먼지연무의 일종으로서 주로 대륙의 황토지대에서 불려 올라간 다량의 황토먼지가 온 하늘을 덮고 떠다니며 서서히 하강하는 현상입니다. 정 값 이상인 날을 단순 집계한 결과와 구분하여 이용합니다.

주요요소	황사일수, 미세먼지농도
제공범위	종관기상관측 전지점(폐쇄지점 포함)/지역(전국 13지점+제주 2지점) ※ 자료는 각 지점별로 관측기록이 있는 기간에 대해서만 표출됨
제공기간	지점별 관측개시일 ~ 현재
생산·갱신주기	월, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 기상현상일수 → 황사일수
이용방법	지점·연도·월 또는 계절을 선택해 현상일수를 조회합니다. 현상별 정의와 집계기간을 확인하고, 결측이나 미관측을 발생일수 0으로 처리하지 않습니다. 황사 관측기록을 이용합니다.

3.통계분석

- 바람장미
- 강수일수
- 황사일수
- 폭염일수
- 열대야일수
- 24절기
- 순위값
- 장마

폭염일수





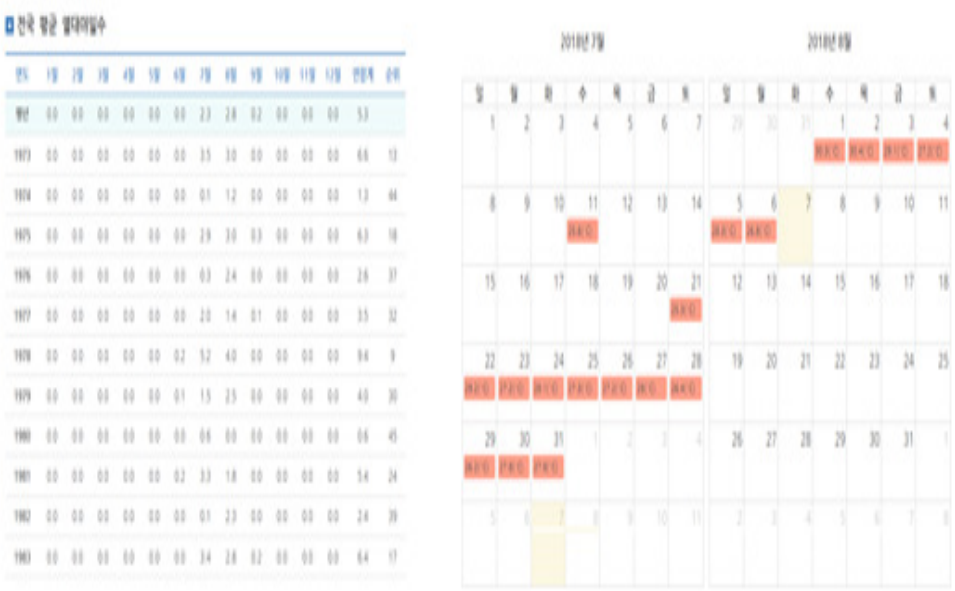
개요

자료상태 : 현행

폭염일수는 일 최고기온이 33°C 이상인 날의 수를 의미합니다. 지점별 폭염이 나타나는 날의 최고기온도 함께 알 수 있습니다.

주요요소	폭염일수/순위, 최장 폭염지속일수, 가장 빠른/늦은 폭염일
제공범위	종관기상관측 전지점(폐쇄지점 포함)/지역(전국 45지점+제주 2지점)
제공기간	지점별 관측개시일 ~ 현재/지역: 1973년 ~ 현재
생산·갱신주기	월, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 기상현상일수 → 폭염일수
이용방법	지점·연도·월 또는 계절을 선택해 현상일수를 조회합니다. 현상별 정의와 집계기간을 확인하고, 결측이나 미관측을 발생일수 0으로 처리하지 않습니다. 일최고기온 기준이며 폭염특보 발효일수와 다릅니다.

열대야일수



개요

자료상태 : 현행

열대야일수는 밤 최저기온이 25 °C 이상인 날로 정의합니다.
※ 밤시간: 당일 18:01 ~ 익일 09:00시까지
기온이 밤에도 25 °C 이하로 내려가지 않을 때에는 너무 더워서 사람이 잠들기 어렵기 때문에 더위를 나타내는 지표로 열대야를 사용합니다.

주요요소	열대야일수/순위, 최장 열대야지속일수, 가장 빠른/늦은 열대야일
제공범위	종관기상관측 전지점(폐쇄지점 포함)/지역(전국 45지점+제주 2지점)
제공기간	지점별 관측개시일 ~ 현재/지역: 1973년 ~ 현재
생산·갱신주기	월, 연 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 기상현상일수 → 열대야일수
이용방법	지점·연도·월 또는 계절을 선택해 현상일수를 조회합니다. 현상별 정의와 집계기간을 확인하고, 결측이나 미관측을 발생일수 0으로 처리하지 않습니다. 공식 열대야 통계를 이용하며, ASOS 일자료의 TA_MIN≥25로 대체 계산하지 않습니다.

3.통계분석

- 바람장미
- 강수일수
- 황사일수
- 폭염일수
- 열대야일수
- 24절기
- 순위값
- 장마

24절기



개요

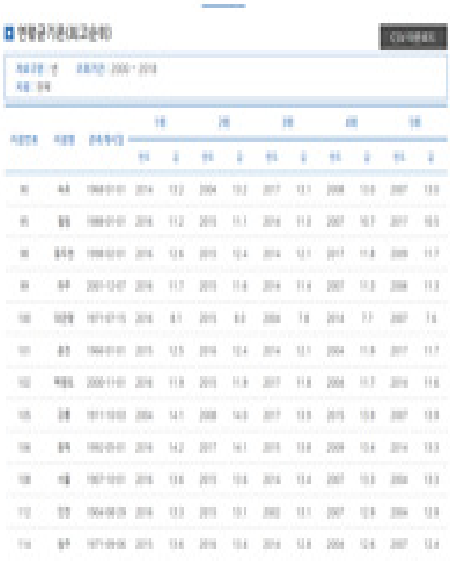
자료상태 : 현행

해당 연도에 해당하는 24절기를 조회하고, 그 절기에 해당하는 최근 30년 기상현상을 조회할 수 있습니다.

주요요소	평균기온, 최고기온, 최저기온, 일강수량, 최심적설량, 강수유무, 강설유무
제공범위	종관기상관측 전지점(폐쇄지점 포함)
제공기간	2010년 ~ 현재
생산·갱신주기	일 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 통계분석 → 24절기
이용방법	지점·기간과 통계요소를 선택해 조회합니다. 해당 통계의 정의·계산식·기준기간·단위를 확인하고 동일한 산출기준의 값을 비교합니다. 절기별 날짜를 확인합니다.

순위값





개요

자료상태 : 현행

관측지점·기간·요소별 기상값의 순위를 조회하는 통계자료입니다. 최고·최저 등 순위 기준과 비교 대상 기간을 함께 확인합니다.

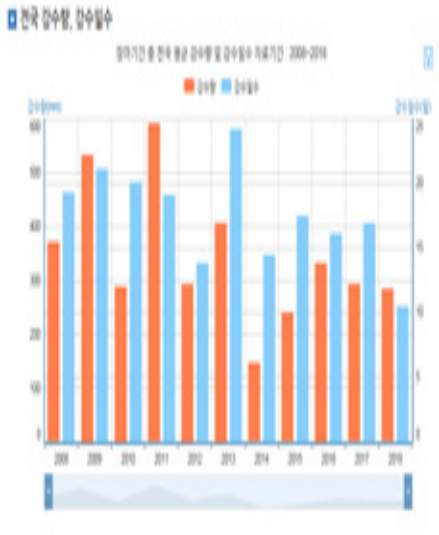
주요요소	기온(평균, 최고, 최저), 강수량(합계강수량, 1시간최다강수량), 풍속(평균, 최대, 최대순간), 적설(신적설, 적설)
제공범위	• 구역별: 종관기상관측 전지점(폐쇄지점 포함) • 지점별: 93지점(10년 이상 관측)
제공기간	지점별 관측 개시일 ~ 현재
생산·갱신주기	일, 순, 월, 계절, 년 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 통계분석 → 순위값
이용방법	지점·기간과 통계요소를 선택해 조회합니다. 해당 통계의 정의·계산식·기준기간·단위를 확인하고 동일한 산출기준의 값을 비교합니다. 순위 비교 대상기간과 동순위 처리 기준을 확인합니다.

3.통계분석

- 바람장미
- 강수일수
- 황사일수
- 폭염일수
- 열대야일수
- 24절기
- 순위값
- 장마

장마





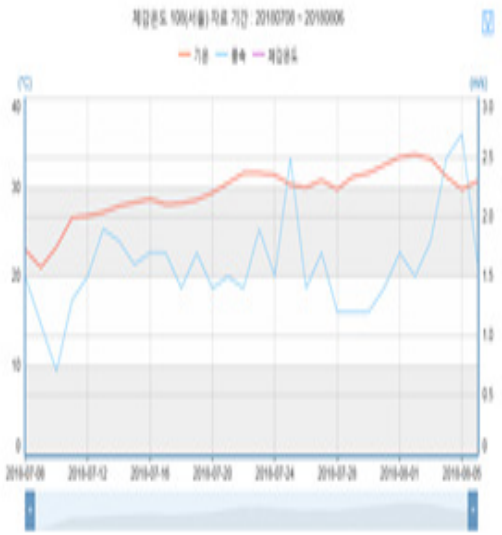
개요

자료상태 : 현행

장마는 우리나라 6월 하순부터 7월 하순까지 계속해서 많이 내리는 비로 기상학적으로는 장마전선의 영향을 받아 비가 오는 경우를 의미합니다.

주요요소	장마평년값, 강수량, 강수일수, 장마시작·종료일, 장마 기간
제공범위	66지점(전국 62지점+제주 4지점)
제공기간	1961년 ~ 현재
생산·갱신주기	년 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 통계분석 → 장마
이용방법	지점·기간과 통계요소를 선택해 조회합니다. 해당 통계의 정의·계산식·기준기간·단위를 확인하고 동일한 산출기준의 값을 비교합니다. 사후 확정된 지역별 장마기간을 적용합니다.

체감온도





개요

자료상태 : 현행

체감온도는 외부에 있는 사람이나 동물이 바람과 한기에 노출된 피부로부터 열을 빼앗길 때 느끼는 추운 정도를 나타내는 지수입니다. 바람이 강해질수록 피부의 열 손실률은 높아지게 되며 결국 내부체온을 떨어뜨립니다.

※ 체감온도 = $13.12 + 0.6215T - 11.37V^{0.16} + 0.3965TV^{0.16}$

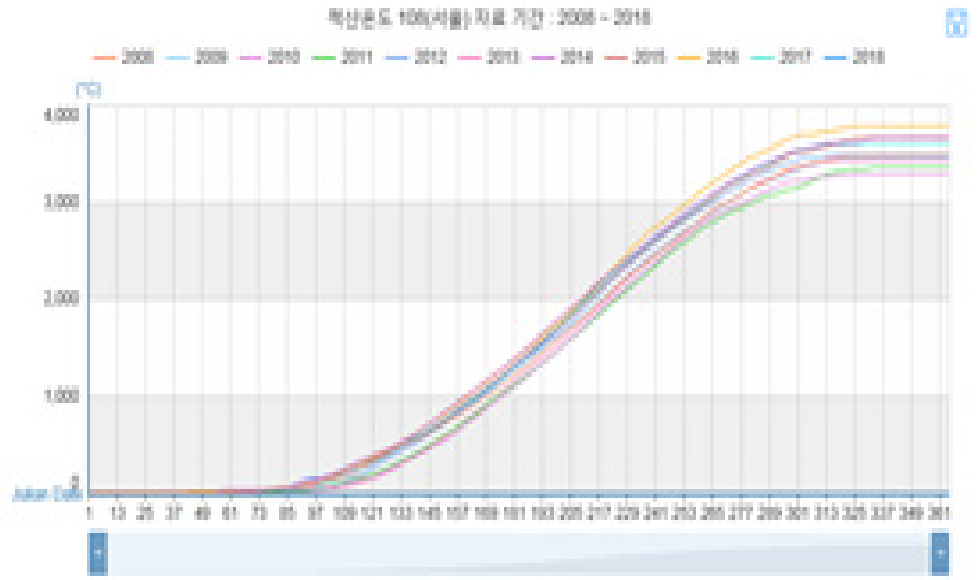
T : 기온(°C), V : 풍속(km/h)

주요요소	기온, 풍속, 체감온도
제공범위	종관기상관측 전지점(폐쇄지점 포함)
제공기간	지점별 관측 개시일 ~ 현재
생산·갱신주기	일 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 응용기상분석 → 체감온도
이용방법	지점·기간과 통계요소를 선택해 조회합니다. 해당 통계의 정의·계산식·기준기간·단위를 확인하고 동일한 산출기준의 값을 비교합니다. 계절별 체감온도 계산식의 차이를 확인합니다.

4.응용기상

• 체감온도
• 적산온도
• ICAO 데이터
카탈로그(공항표준온도, 공항평균최저온도)

적산온도



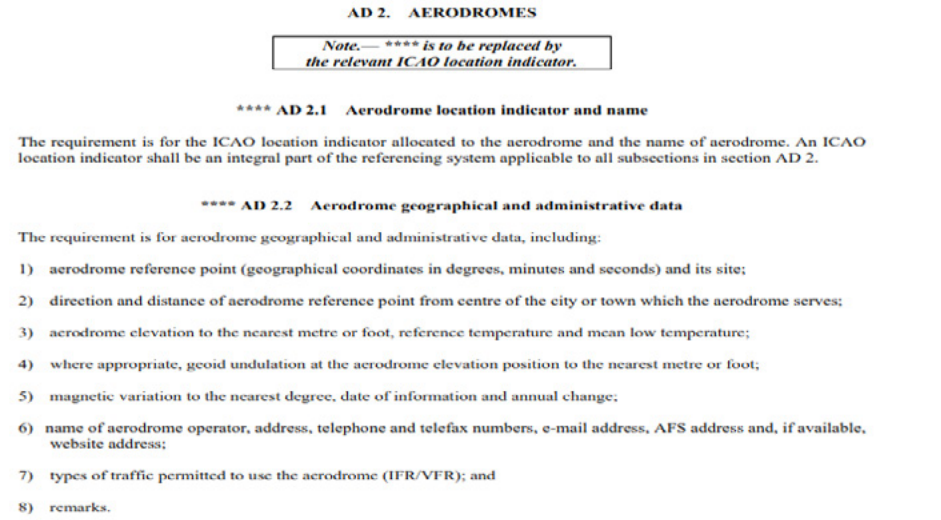
개요

자료상태 : 현행

적산온도는 작물의 생육에 필요한 열량을 나타내기 위한 것으로 생육 일수의 일평균기온을 적산한 것을 말하며, 적산온도를 계산할 때의 일평균기온은 해당 작물이 활동할 수 있는 최저 온도(기준 온도) 이상의 것만을 택합니다.
기준 온도는 작물에 따라 다르지만 가을 채소와 같이 저온에서도 자라는 것은 5 ℃, 일반적인 온대 지방의 여름철 작물은 10 ℃, 고온을 필요로 하는 작물은 15 ℃라고 알려져 있습니다.

주요요소	적산온도
제공범위	종관기상관측 전지점(폐쇄지점 포함)
제공기간	지점별 관측 개시일 ~ 현재
생산·갱신주기	일 자료
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 기후통계분석 → 응용기상분석 → 적산온도
이용방법	지점·기간과 통계요소를 선택해 조회합니다. 해당 통계의 정의·계산식·기준기간·단위를 확인하고 동일한 산출기준의 값을 비교합니다. 기준온도와 누적 시작·종료일을 확인합니다.

ICAO 데이터 카탈로그 (공항표준온도, 공항평균최저온도)



AD 2. AERODROMES

*Note.— **** is to be replaced by the relevant ICAO location indicator.*

****** AD 2.1 Aerodrome location indicator and name**

The requirement is for the ICAO location indicator allocated to the aerodrome and the name of aerodrome. An ICAO location indicator shall be an integral part of the referencing system applicable to all subsections in section AD 2.

****** AD 2.2 Aerodrome geographical and administrative data**

The requirement is for aerodrome geographical and administrative data, including:

1) aerodrome reference point (geographical coordinates in degrees, minutes and seconds) and its site;
2) direction and distance of aerodrome reference point from centre of the city or town which the aerodrome serves;
3) aerodrome elevation to the nearest metre or foot, reference temperature and mean low temperature;
4) where appropriate, geoid undulation at the aerodrome elevation position to the nearest metre or foot;
5) magnetic variation to the nearest degree, date of information and annual change;
6) name of aerodrome operator, address, telephone and telefax numbers, e-mail address, AFS address and, if available, website address;
7) types of traffic permitted to use the aerodrome (IFR/VFR); and
8) remarks.

개요

자료상태 : 현행

공항표준온도, 공항평균최저온도는 ICAO Doc. 10066(PANS-AIM, 항행업무절차-항공정보관리)를 근거로 요구되는 통계데이터로 각각 데이터의 의미는 아래와 같습니다.

* 공항표준온도(Reference Temperature, 단위:°C)
: 최근 5년간의 데이터에 대해서 연중 가장 더운 달(가장 더운 월 평균 기온이 있는달), 일일 최고 기온의 월 평균값을 5년 평균한 값
* 공항평균최저온도(Mean Low Temperature, 단위:°C)
: 최근 5년간의 데이터에 대해서 연중 가장 추운 달(가장 추운 월 평균 기온이 있는달), 일일 최저 기온의 월 평균값을 5년 평균한 값

주요요소

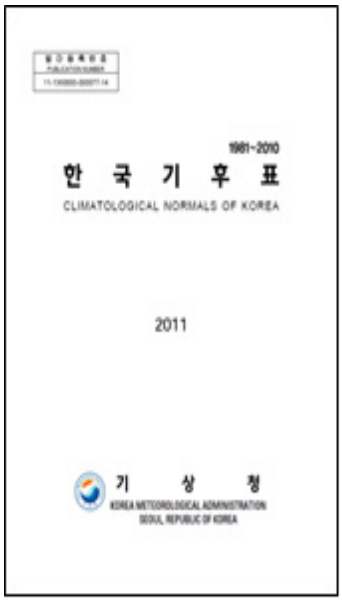
공항표준온도·공항평균최저온도		
요소명	영문요소명	단위
시각(년)	YEAR	KST
지점 ICAO 코드	STN_ID	-
활주로방향	STN_CD	-
지점명 한글	STN_KO	-
공항표준온도(최근5년간)	REF_TA	°C
공항평균최저온도(최근5년간)	MEAN_LOW_TA	°C
산출 기초자료 (가장 추운 달 일최저기온의 월평균값 및 가장 더운 달 일최고기온의 월평균값)		
요소명	영문요소명	단위
시각(년)	YEAR	KST
지점 ICAO 코드	STN_ID	-
활주로방향	STN_CD	-
지점명 한글	STN_KO	-
가장 더운 달	TA_MAX_MON	월(月)
일최고기온의 월평균	TA_MAX	°C
가장 추운 달	TA_MIN_MON	월(月)
일최저기온의 월평균	TA_MIN	°C

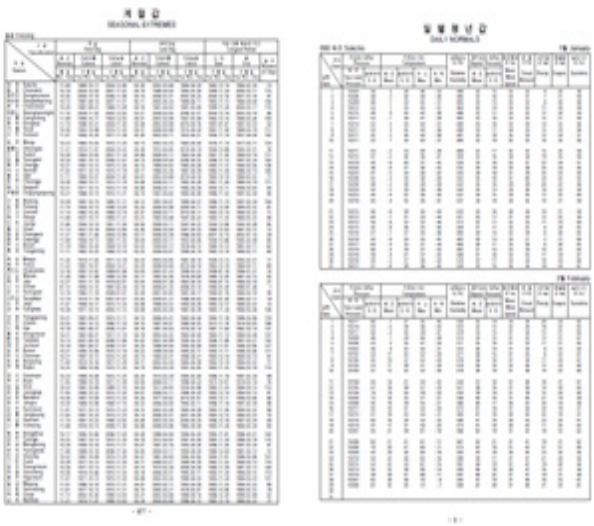
4.응용기상

- 체감온도
- 적산온도
- ICAO 데이터 카탈로그(공항표준온도, 공항평균최저온도)

제공범위	7개소 공항: 인천, 김포, 제주, 무안, 울산, 양양, 여수
제공기간	아래 기간의 자료를 활용하여 산출 인천: 2000년 9월 1일 자료부터 현재 김포: 1961년 1월 1일 자료부터 현재 제주: 1965년 11월 1일 자료부터 현재 무안: 2007년 11월 8일 자료부터 현재 울산: 1985년 1월 1일 자료부터 현재 양양: 2002년 4월 1일 자료부터 현재 여수: 1975년 1월 1일 자료부터 현재
생산·갱신 주기	연 1회
제공형식	txt
제공경로	기상청 API허브(https://apihub.kma.go.kr) → 항공기상 → 항공기상관측(METAR) → ICAO 데이터 카탈로그
	txt 포맷은 오픈API 서버를 통해 자료를 제공

한국기후표





개요

자료상태 : 현행

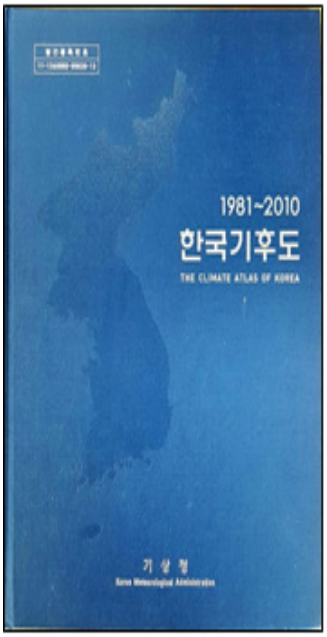
국내 관측지점의 기후평년값 등을 표로 정리한 간행물입니다. 발간판별 기준기간·수록지점·통계요소를 확인하고 필요한 표와 수치를 이용합니다.

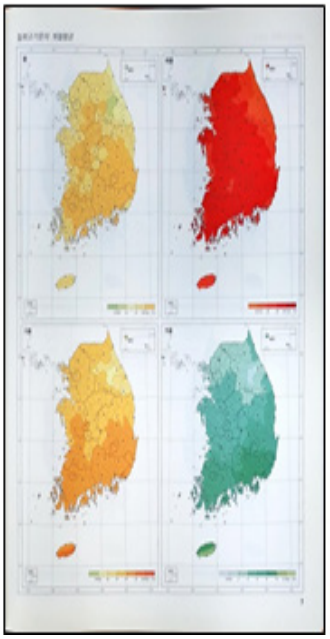
주요요소	기온, 기압, 증기압, 습도, 강수량, 증발량, 바람, 구름, 일조, 일사 등에 대한 일, 순, 월, 계절, 연별 평년값(30년간) 산출
제공기간	1977년, 1982년, 1991년, 2001년, 2011년, 2021년 발간
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 한국기후표
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

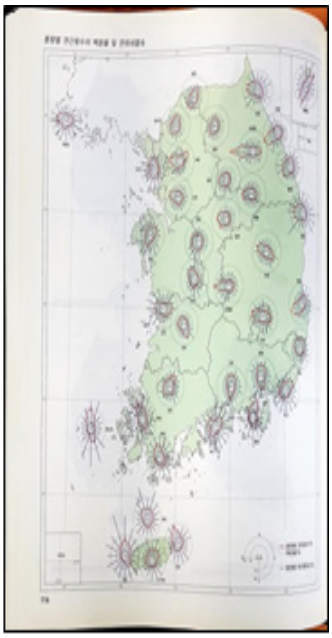
5.간행물

• 한국기후표
• 한국기후도
• 북한기상20년보
• 북한기상30년보
• 장마백서
• 태풍백서
• 항공기후정보
• 지구대기감시보고서

한국기후도







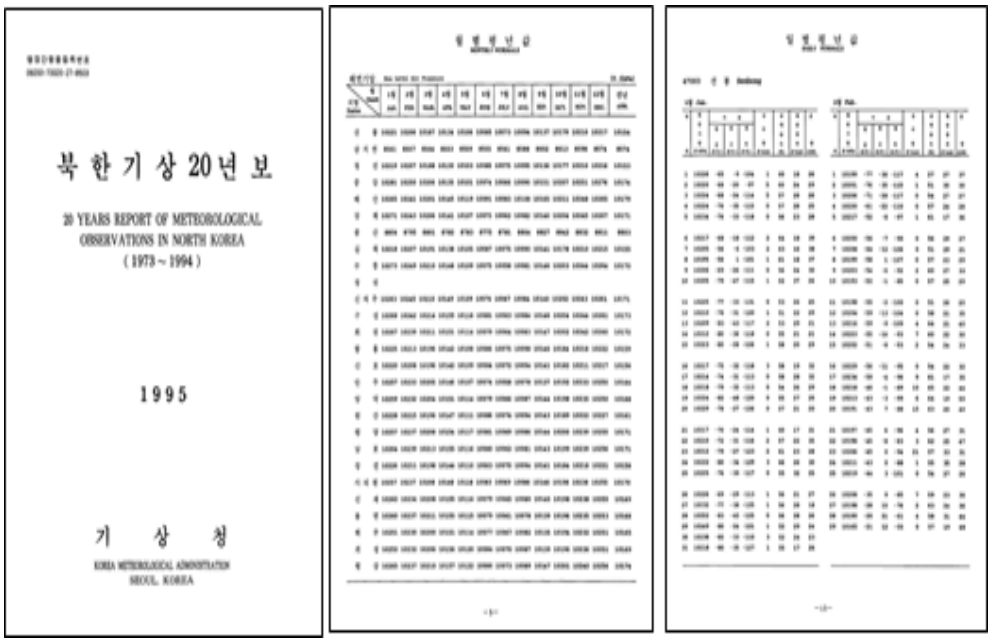
개요

자료상태 : 현행

기후평년값 등으로 우리나라의 지역별 기후 특성을 지도와 도표로 표현한 간행물입니다. 지도별 기준기간·요소·단위와 공간 표현방법을 확인하여 이용합니다.

주요요소	기온, 강수량, 적설, 해면기압, 습도, 증기압, 증발량, 바람, 일조, 냉·난방도일, 일사 등
제공범위	219지점
제공기간	• 2011년: 1981년 ~ 2010년 • 2021년: 1991년 ~ 2020년
생산·갱신주기	연 자료(10년 주기(요소별 30년 평년값))
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 한국기후도
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

북한기상20년보



개요

자료상태 : 현행

북한기상20년보에서 세계기상기구(WMO)의 기상통신망(GTS)을 통해 입수된 북한 27개 지점의 기압, 기온, 강수량, 습도, 풍속, 운량에 대한 22년(1973년~1994년)간의 평균값을 일별, 월별로 산출한 자료를 확인할 수 있습니다.

주요요소	기압, 기온, 강수량, 습도, 풍속, 운량의 22년(1973년~1994년)간 통계
제공범위	27지점
제공기간	1995년 발간
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 북한
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

5.간행물

- 한국기후표
- 한국기후도
- 북한기상20년보
- 북한기상30년보
- 장마백서
- 태풍백서
- 항공기후정보
- 지구대기감시보고서

북한기상30년보



개요

자료상태 : 현행

북한기상30년보에서 세계기상기구(WMO)의 기상통신망(GTS)을 통해 입수된 북한 27개 지점의 기압, 기온, 강수량, 습도, 풍속, 운량에 대한 30년(1991년~2020년) 평년값을 일, 순, 월, 계절, 년에 대해 산출한 자료를 확인할 수 있습니다.

주요요소	기압, 기온, 강수량, 습도, 풍속, 운량의 30년(1991년~2020년)간 통계
제공범위	27지점
제공기간	2011년(1981년 ~2010년), 2021년(1991년~2020년) 발간
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 지상 → 북한
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

장마백서



개요

자료상태 : 현행

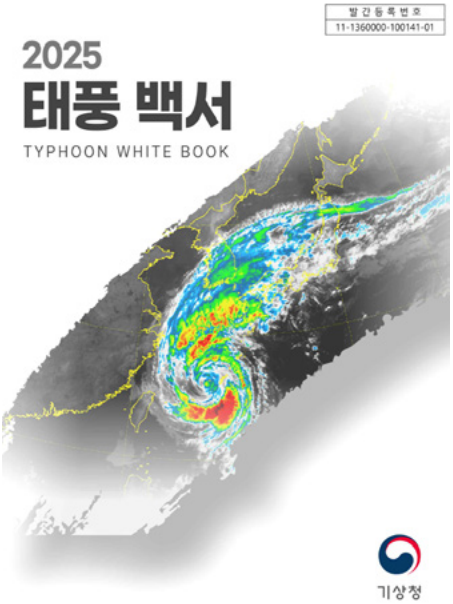
우리나라 장마의 특성과 분석 결과를 정리한 백서입니다. 발간판별 수록기간과 장마의 분석기준을 확인하여 기후 특성 및 과거 사례를 검토하는 데 이용합니다.

주요요소	강수량, 장마시종일, 장마기간 등
제공범위	장마시종일/지속기간, 강수량 등을 지역별(중부, 남부, 제주) 통계처리
제공기간	1995년, 2011년, 2022년 발간
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 백서 → 장마백서(1995년, 2011년, 2022년)
	기상청 기후정보포털(https://www.climate.go.kr) → 교육·자료 → 발간물 → 기타(2011년, 2022년 자료)
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

5.간행물

- 한국기후표
- 한국기후도
- 북한기상20년보
- 북한기상30년보
- 장마백서
- 태풍백서
- 항공기후정보
- 지구대기감시보고서

태풍백서



개요

자료상태 : 현행

태풍 발생 개수, 영향태풍 개수, 영향태풍 기간의 우리나라 강수량, 풍속, 기압 등의 극값 순위 등의 태풍 통계를 제공합니다.

주요요소	태풍영향 기간의 강수량, 풍속, 기압 등
제공범위	1951~2024년의 태풍 발생 개수와 영향태풍 개수 및 강수량, 풍속 등의 극값 통계
제공기간	2025년 발간
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 간행물 → 백서 → 태풍백서
	기상청 누리집(https://kma.go.kr) → 알림·자료 → 기상간행물 → 태풍백서
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

항공기후정보

5.간행물

• 한국기후표
• 한국기후도
• 북한기상20년보
• 북한기상30년보
• 장마백서
• 태풍백서
• 항공기후정보
• 지구대기감시보고서



개요

자료상태 : 현행

항공기후정보는 민간공항 7개소에 대한 최근 5년의 관측결과를 기반으로 작성한 기후정보간행물이다. 세계기상기구(WMO) 기술규정 '국제항공항행을 위한 기상 업무(WMO-No. 49)'의 ' Part III. Aeronautical Climatology'에 명시된 공항기후표(Table A~P) 및 공항기후개요(Model A~E) 양식을 참조하여 작성됩니다.

주요요소	5년 통계를 기반으로 작성한 공항기후표(Table A~P) 및 공항기후개요(Model A~E) [공항기후표] A: 계급별, 월별 시정 발생일수 B: 계급별, 월별 RVR 일수 C: 계급별, 월별 운량 5/8 이상인 최저운고고도 발생일수 D: 월별 풍향별 발생률 E: 월별 풍향별 평균풍속 F: 일별 해면기압 평균값 G: 일별 기온 평균값 H: 일별 최고기온 I: 일별 최저기온 J: 일별 풍속 평균값 K: 일별 운량 평균값 L: 일별 상대습도 평균값 M: 일별 강수량 합계값 N: 계급별, 월별 기상현상 발생일수 O: 월별 평균 및 극값 P: 월별 기상현상발생일수 [공항기후개요] A: 시각별,계급별,운고, RVR, VIS 빈도 B: 시각별, 계급별 시정 발생빈도 C: 시각별, 계급별 운고(ft) 발생빈도 D: 시각별, 계급별 풍향(30도간격), 풍속빈도(5KT 간격) E: 시각별 5°C 계급별 기온빈도
제공범위	7개소 공항: 인천, 김포, 제주, 무안, 울산, 양양, 여수
제공기간	2011년 발간 ~ 현재
생산·갱신주기	연 1회
제공형식	PDF, CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 간행물 → 항공- 항공기상청
	홈페이지(amo.kma.go.kr)→ 항공날씨정보 → 항공기상통계 → 항공기후정보
	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.

5.간행물

- 한국기후표
- 한국기후도
- 북한기상20년보
- 북한기상30년보
- 장마백서
- 태풍백서
- 항공기후정보
- 지구대기감시보고서

지구대기감시보고서



개요

자료상태 : 현행

1989년 시작된 WMO 지구대기감시(GAW) 프로그램의 일환으로, 우리나라 지구대기감시소에서 생산된 관측자료를 체계적으로 분석·정리한 보고서이다. 우리나라 지구대기감시소에서 관측한 온실가스, 반응가스, 에어로졸, 성층권오존▮자외선, 대기복사, 총대기침적 6종의 관측 결과와 분석 내용을 수록한다. 매년 4월 전년도 관측자료를 기반으로 발간한다.

주요요소	온실가스, 반응가스, 에어로졸, 성층권오존/자외선, 대기복사, 총대기침적
제공범위	• 온실가스 (안면도, 제주고산, 울릉도, 독도): 이산화탄소, 메탄, 아산화질소, 염화불화탄소류, 육불화황 • 반응가스(안면도, 제주고산, 울릉도): 일산화탄소, 오존, 이산화황, 질소산화물 • 에어로졸(안면도, 제주고산, 울릉도): PM10 질량농도, 크기별 수농도, 미세입자 크기별 수농도, 응결핵 수농도, 광산란계수, 광흡수계수, 광학깊이, 연직분포, 이온성분, 원소성분 • 성층권 오존/자외선(안면도, 제주고산, 울릉도, 포항): 오존전량, 오존 연직분포, 자외선A, 자외선B • 대기복사(안면도, 제주고산): 직달일사, 태양상향복사, 태양하향복사(전천일사), 산란일사, 지구상향복사, 지구하향복사, 순복사 • 총대기침적(안면도, 제주고산, 울릉도): 산성도, 전기전도도, 강수 이온성분
제공기간	(시작) 2001. ~현재
생산·갱신주기	일, 월, 연 자료
제공형식	PDF
제공경로	기후정보포털(https://www.climate.go.kr) → 기후변화감시 → 지구대기감시자료 → 통계자료
이용방법	간행물 종류와 발간 연월·판을 선택하여 파일을 내려받습니다. 수록기간·지점·통계기준·단위를 확인하고 표·그림의 출처와 쪽수를 기록하여 인용합니다. 최신 실황 또는 확정 전 자료와 구분합니다.
간행물	매년 4월 지구대기감시보고서 정기 발간

12

역사기록

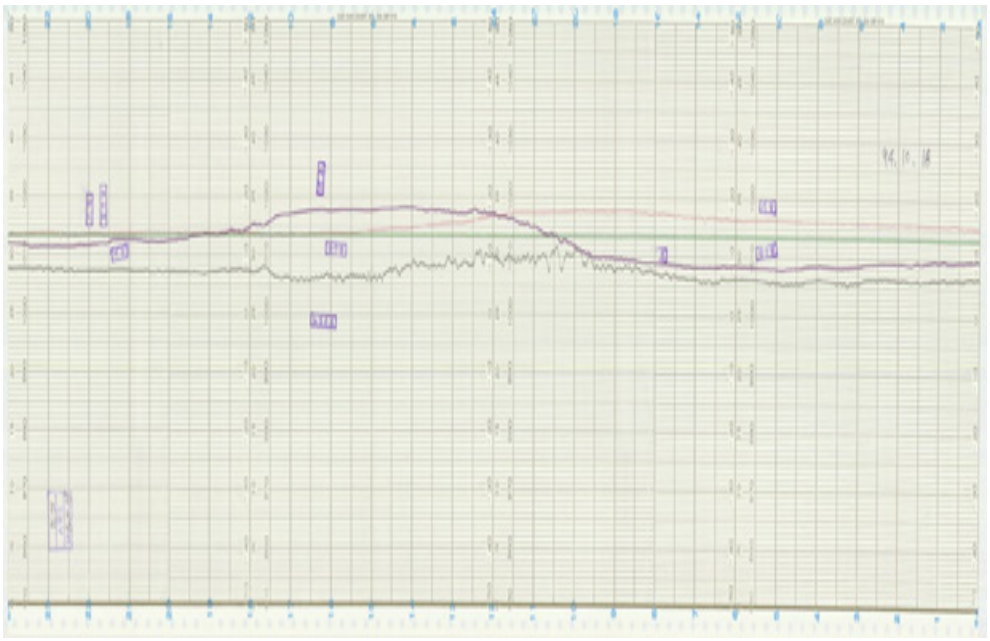
Historical Records



1.자기기록지

• 자기기록지
• 자기기록지설명자료

자기기록지

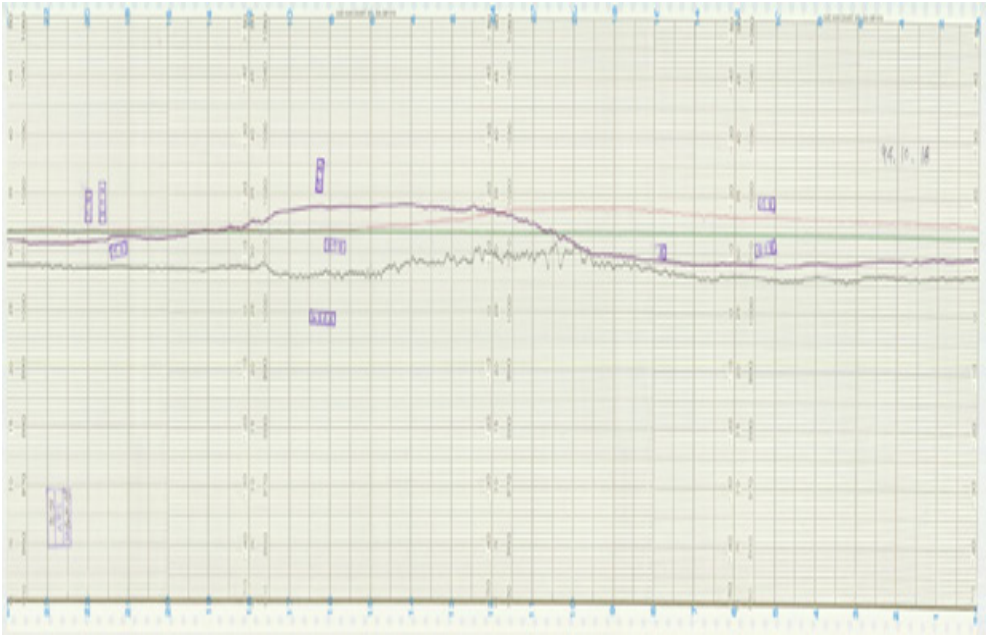


개요

기상요소의 시간 변화를 자기기록계(自記記錄計)가 종이에 연속 기록한 관측자료입니다. '자기'는 자동으로 기록한다는 뜻이며 지구자기장 관측과는 다릅니다. 기록지의 시간축·눈금·관측값 보정 내용을 함께 확인합니다.

주요요소	강우강도, 강우, 기압, 기온습도, 기온, 미압, 습도, 우량일조, 일사, 일조, 증발산, 증발산위, 지하수, 풍력, 풍속, 풍압, 풍향복엽
제공범위	요소별 지점 다름
제공기간	1944년 ~ 1999년 ※ 요소별 제공기간 다름
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 자기기록지
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 기록지의 눈금과 보정값을 적용합니다.

자기기록지설명자료



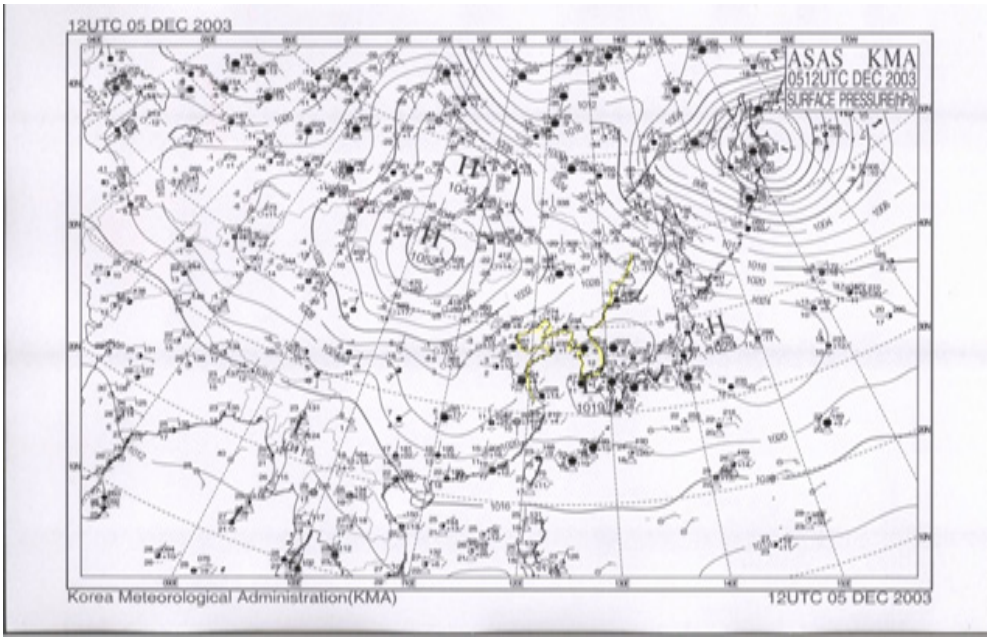
개요

연속적인 기상관측 실황을 보기 위하여 설치된 자기기록계는 관측 요소에 의해 결정되는 내부의 구성에 따라 기록펜이 상하운동을 하여 원통형 시계에 감긴 기록지가 기록되는 원리로 작동됩니다. 일반적으로 자기기록지의 횡축은 시각을, 종축은 관측 기록 값을 표시하며 실제 관측 값과의 비교 값도 기록하였습니다.

주요요소	강우강도, 강우, 기압, 기온습도, 기온, 미압, 습도, 우량일조, 일사, 일조, 증발산, 증발산위, 지하수, 풍력, 풍속, 풍압, 풍향복엽
제공범위	요소별 지점 다름
제공기간	1944년 ~ 1999년 ※ 요소별 제공기간 다름
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 자기기록지
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 원본 기록지와 해설을 함께 확인합니다.

2.종이일기도 종이일기도

• 종이일기도

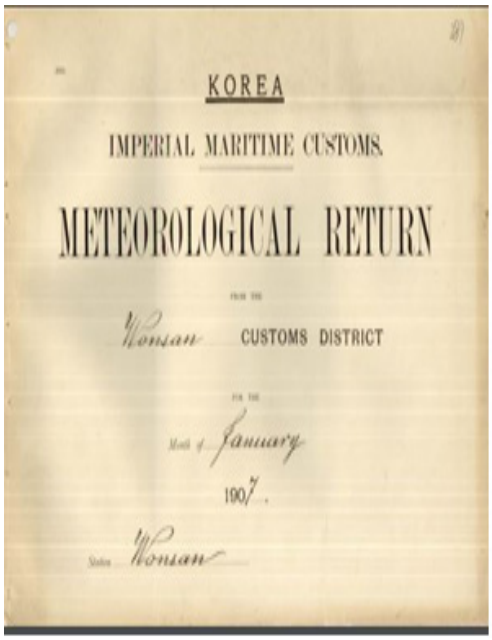


개요

종이일기도는 기상관측소에서 측정한 기압, 기온, 풍향, 풍속 등 기상요소를 그림으로 표시한 지도로서 고기압, 저기압, 바람의 방향 · 세기, 구름의 양 등의 기상정보가 있습니다. 이 자료를 이용하여 기압배치와 전선 등을 분석하여 현재의 날씨 상태를 알 수 있으며, 기상예보관이 미래의 날씨를 예측하는 기상예보의 기초자료로 활용됩니다.

주요요소	지상일기도	아시아(ASAS), 극동아시아(ASFE), 북반구(AUXN)
	고층일기도	아시아 925 · 850 · 700 · 500 · 300 · 200hPa, 북반구500hPa
제공범위	요소별 지점 다름	
제공기간	1944년 ~ 1999년 ※ 요소별 제공기간 다름	
제공형식	PDF	
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 종이일기도	
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 기록지의 눈금과 보정값을 적용합니다.	

해관기상관측자료(19~20세기)


KOREA

IMPERIAL MARITIME CUSTOMS.

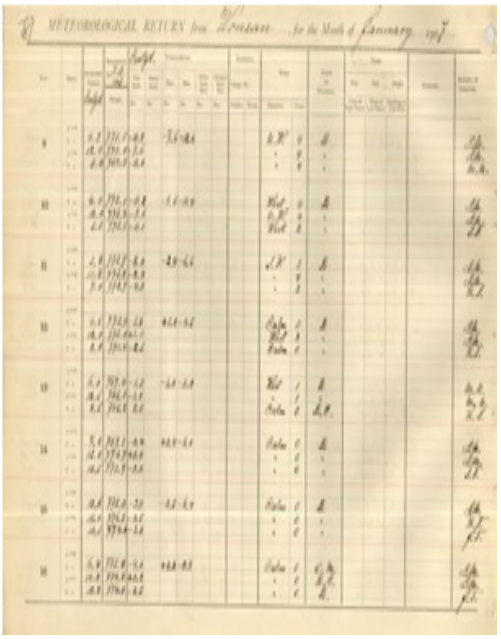
METEOROLOGICAL RETURN

Wonsan CUSTOMS DISTRICT

Month of January

1907.

Station Wonsan



개요

해관기상관측자료는 1883년(고종20년)부터 제물포, 부산에 있는 해관에서 정규적으로 관측한 기온, 기압, 바람 등의 연안 기상관측자료의 원부입니다. 이는 우리나라 근대기상 관측이 1904년 부산, 목포, 인천의 기상관측소에서 시작됐다고 알려진 것의 이전 자료입니다.

※ 해관(海關): 관세를 징수하는 것이 주된 임무로 수로 안전을 위한 업무와 연안기상관측 업무를 수행하던 기관

※ 부산 해관의 최초 기록 : 1883년 8월 12일

주요요소	시간, 부착온도, 기압, 온도, 강수량, 바람, 구름, 전운량, 하늘상태, 기사란
제공범위	인천, 원산, 부산
제공기간	1883년 ~ 1907년 ※ 지점별 제공기간 다름
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 역사자료 → 해관기상관측자료
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

3.역사자료

• 해관기상관측자료 (19~20세기)
• 측우기자료(조선시대)
• 조선왕조실록 기상기록
• 각사등록(조선시대)
• 역사지진 기록

측우기자료(조선시대)



개요

종관기상의 세계 최초 기록의 역사는 “1440년 조선에서 강우량 사용(Rain gauge used in Korea)”했다고 기록하고 있습니다(Bluestein,1992). 이 관측기기에 의해 관측된 강수량기록이 유네스코에 등재된 세계문화유산 문헌들에 남겨졌습니다. 측우기로 관측한 강우데이터는 승정원일기(1770~1894), 일성록(1785~1910)의 문헌을 참고하여 당시 1척(尺 : 약 200mm), 1촌(寸: 약 20mm), 1분(分 : 2mm)의 단위를 현재 국제표준 단위(mm)로 환산한 강수량입니다.

주요요소	강수량
제공범위	역사문헌에 기록이 존재하는 한반도 내 일부 지점
제공기간	1777~1907년
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 조선시대관측→ 측우기 강수량
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 전통 길이 단위와 현대 강수량의 환산 기준을 명시합니다.

조선왕조실록 기상기록



개요

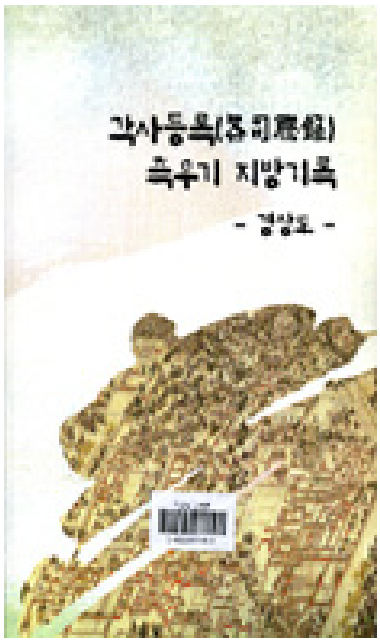
조선왕조실록은 국보 제151호이자, 유네스코가 정한 세계기록유산으로 1997년 등재되었습니다. 이를 통해 한국 역사에 대한 절대 다수의 지식뿐 아니라, 기상기후 현상을 포함하고 있어 조선 ~ 근대 기상기후 연구에 의미 있는 자료입니다. 또한 측우기 발명과 필요성에 대한 기록도 포함되어 있으며, 나라를 통치하면서 발생한 기상이벤트의 정성적 기록뿐만 아니라, 비가 온 뒤의 수심을 기록한 정량적 기록도 포함되어 있습니다.

주요요소	기상, 기후, 지변, 재해, 천문, 재이상서 ※ 재이상서: 기상현상에 대한 당시의 인식과 그에 따른 제사 및 기타 대책 등을 담음
제공기간	1392년 ~ 1495년 ※ 실록별로 다름
생산·갱신 주기	실록: 태조, 정종, 태종, 세종, 문종, 단종, 세조, 예종, 성종
제공형식	엑셀(xls)
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 역사자료 → 조선왕조실록
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

3.역사자료

• 해관기상관측자료 (19~20세기)
• 측우기자료(조선시대)
• 조선왕조실록 기상기록
• 각사등록(조선시대)
• 역사지진 기록

각사등록(조선시대)



개요

각사등록에는 조선8도의 감영과 유수부에서 관측한 측우기 및 우택 우량 자료가 담겨져 있어 조선시대 전국적 기상특성을 확인할 수 있는 귀중한 자료입니다. 전국적으로 도, 부, 군, 현의 강우량과 농사, 기상재해 기록이 비가 올 때마다 월별과 날짜별로 기록되어 있습니다.

※ 각사등록: 조선시대 각 관아에서 해당 지방의 정치, 사상, 사회, 경제 등을 직접 기록한 자료 (280책으로 구성)

주요요소	강우량, 우택
제공기간	• 경상도: 1863년 ~ 1900년 • 전라도: 1829년 ~ 1897년 • 충청도: 1835년 ~ 1900년 • 평안도: 1830년 ~ 1884년(책자만 있음)
제공형식	엑셀(xls)
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 역사자료 → 각사등록
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

역사지진 기록



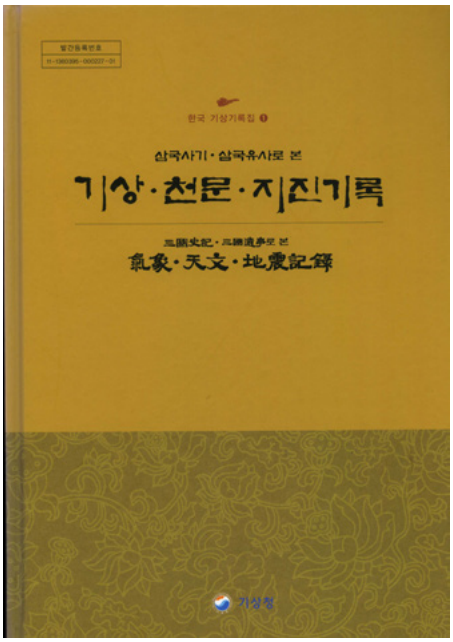
개요

역사지진에 대한 연구는 한반도에서 발생 가능한 지진과 그에 따른 지진피해를 추정하는데 중요한 역할을 합니다. <<삼국사기>>, <<고려사절요>>, <<고려사>>, <<조선왕조실록>>, <<중보문헌비고>>, <<승정원일기>>, <<일성록>>, <<풍운기>>, <<천변초출등록>>과 개인문집, 지방지를 참고하고, 기존의 연구자료를 검토하여 피해상황이나 지진현상을 목록화하였으며, 각 지진의 감진지역을 상세하게 구획화하고 진앙과 진도를 결정하였습니다.

주요요소	지진발생 일자(양력/음력), 진앙(위도/경도), 진도, 기사(역사문헌 서술, 출처)
제공지점	역사문헌에 기록이 존재하는 한반도 일대
제공기간	2년 ~ 1904년
자료형태	지도 및 목록
제공형식	csv
제공경로	NECIS(국가지진종합정보시스템) 홈페이지(http://necis.kma.go.kr) → 지진자료 → 역사지진
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 역사 진도·진앙 추정을 현대 관측 규모로 간주하지 않습니다.

4.기상기록집 기상천문지진기록

- 기상천문지진기록
- 한반도 역사지진기록
- 관상감에 기록한 17세기 밤하늘
- 기상인이 말하는 중앙관상대
- 농사와 측우기 기록

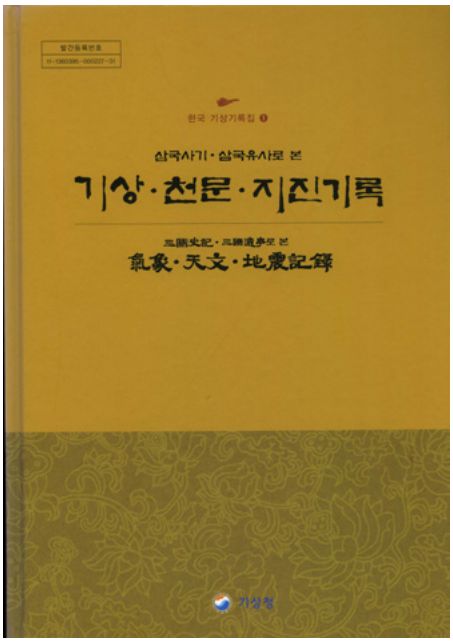


개요

본 책자는 삼국사기와 삼국 유사에 나오는 기상, 천문, 지진 및 기타 기상관련기사를 선별한 것입니다. 삼국사기의 기록을 기상, 천문, 지진으로 구분하니 424건, 218건, 97건이 조사되었으며 삼국유사는 기상기록 8건, 천문기록 5건, 지진기록이 2건으로 조사되었습니다.

주요요소	≪삼국사기≫	기상 · 천문 · 지진기록(본기, 잡지, 열전)
	≪삼국유사≫	기상 · 천문 · 지진기록
제공형식	PDF	
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 기상기록집	
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.	

한반도 역사지진기록



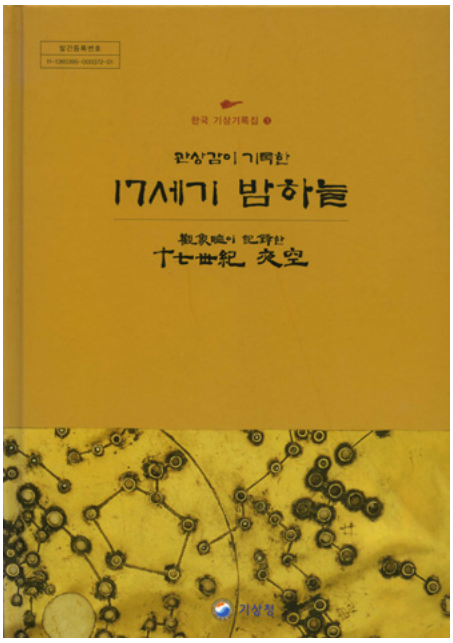
개요

역사문헌에 실린 지진 현상의 서술과 출처를 정리하고 진앙·진도를 추정한 기록집입니다. 현대 지진계로 관측한 규모·위치와는 자료 성격과 불확실성이 다릅니다.

주요요소	삼국시대, 고려시대, 조선시대의 한반도 역사지진
제공기간	2년 ~ 1904년
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 기상기록집
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다. 추정 진앙·진도의 불확실성을 함께 기술합니다.

4.기상기록집 관상감에 기록한 17세기 밤하늘

- 기상천문지진기록
- 한반도 역사지진기록
- 관상감에 기록한 17세기 밤하늘
- 기상인이 말하는 중앙관상대
- 농사와 측우기 기록

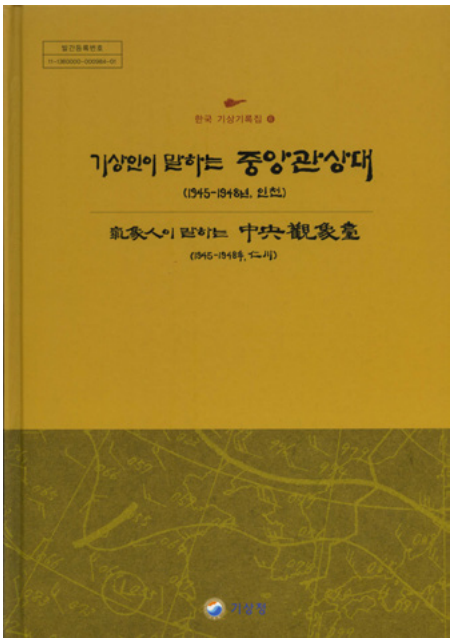


개요

조선시대 관상감은 하늘과 별과 구름 등을 관찰하고 시간, 달력, 택일 등을 담당하는 기관이었습니다.
성변등록은 밤하늘의 혜성과 별자리를 그림과 글로 기록한 성변측후단자들의 모음집입니다.
관상감이 기록한 17세기 밤하늘은 성변등록을 기반으로 제작되었습니다. 특히 1661년, 1664년, 1665년과 1668년 약 100일간의 기록에 주석을 보태어 누구나 쉽게 이해할 수 있도록 구성하였습니다.

주요요소	1661년, 1664년 ~ 1665년, 1668년≪성변등록≫, 조선시대 관상감 위치, 천상열차분야지도 등 ※ 성변등록: 관상감의 관측일지 모음집
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 기상기록집
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

기상인이 말하는 중앙관상대



개요

한국 기상기록집 기상인이 말하는 중앙관상대는 광복 후 1945년부터 1948년까지 기상청의 전신인 중앙관상대의 역사를 구술 사료를 토대로 정리하여 그 시기의 부족한 자료를 보강한 책입니다.
이 기록집에는 4분의 기상인들의 구술을 통하여 광복 직후 3년간 중앙관상대의 모습을 다양하게 보여주고 있습니다.

주요요소	총 4인의 기상인(변찬홍, 서정은, 김병탁, 김성식)이 말하는 1945년 ~ 1948년까지의 중앙관상대 역사
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 기상기록집
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

4.기상기록집 농사와 측우기 기록

- 기상천문지진기록
- 한반도 역사지진기록
- 관상감에 기록한 17세기 밤하늘
- 기상인이 말하는 중앙관상대
- 농사와 측우기 기록



개요

본 책은 1783년과 1784년 경기지역의 농사 상황과 각 고을 수령이 관찰사에게 보고한 비에 대한 상세한 정보를 담고 있습니다. 더욱이 당시 지방에서 널리 쓰이던 우택의 기록까지 고스란히 모은 방대한 정보입니다. 이 기록집은 국사편찬위원회에서 간행한 각사등록 중 경기감영 부분에서 농사와 기상과 관련된 기록을 발췌해 번역한 것입니다.

주요요소	1783년 ~ 1784년(정조7, 8년)의 농사와 측우기 기록, 강우량 관측단위와 관측시각 등
제공형식	PDF
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 역사기후 → 기상기록집
이용방법	자료 종류·관측관서 또는 기록명·기간을 선택해 원문과 해설자료를 조회합니다. 당시 날짜·시각·단위·관측방법과 기록의 누락·추정 여부를 확인합니다. 번역·환산 수치는 원문의 출처와 구분하여 기록합니다.

13

메타데이터

Metadata



1.지점정보

관측지점정보

• 관측지점정보
• 기상예보구역코드
• 기상특보구역코드

지점정보 | 관측장비 | 센서/관측요소

▪ 지점정보

지점유형	지상		
관리기관	기상청	운영기관	기상청 강원지방기상청 관측과
장비명	AOS		
구주소	강원도 고성군 토성면 봉포리 111-3 속초기상대	새주소	강원도 고성군 도성면 봉포5길 9
표준지점번호	90	기관지점번호	90
시설명(한글)	속초	시설명(영문)	Sokcho
관측개시일	1968-01-01	관측주기(분)	1
좌표(WGS84)	위도 : 38.25085 경도 : 128.56472		
GPS측정좌표계		GPS측정지점	표석
GPS측정일시	2005-08-23	해발고도(m)	17
설치목적	기상관측		
운영방법	자동		

개요

관측지점의 번호·명칭·위치·관측기간과 변경 이력을 제공하는 기준정보입니다. 관측자료를 결합할 때 관측 당시의 지점 위치와 유효기간을 함께 확인합니다.

주요요소	지점번호, 관측 시작·종료일, 위·경도, 관측장소 해발고도
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 메타데이터 → 지점정보
	기상청 API허브(https://apihub.kma.go.kr) → 지상·해양·고층·레이더 등의 지점정보
이용방법	관측 종류·지점번호·기준일을 선택해 지점정보와 이력을 조회합니다. 자료와 결합할 때 유효기간에 맞는 지점 위치를 적용하고 지점번호의 문자열 형식을 유지합니다.

기상예보구역코드



오픈 API 정보

데이터 오류·개선 의견이 있으신가요? 데이터 개선요청 > 오류신고 및 문의 >

기본 정보

OpenAPI 명	기상청_특보구역정보 조회서비스		
분류체계	환경 - 대기	제공기관	기상청
관리부서명	국가기후데이터센터	관리부서 전화번호	042-481-7480

상세 및 제공정보 — 오픈API 에러코드 안내

설명	기상청에서 발표하는 육상 및 해상의 기상특보구역코드 정보입니다. 한파, 건조, 안개, 폭염, 지진해일, 폭풍해일, 호우, 대설, 태풍, 풍랑, 강풍 12개 기상현상에 대해 중대한 재해발생이 예상될 때 해당 지역에 대하여 기상특보의 발표 기준에 따라 주의보 및 경보로 구분하여 178개 시 · 군 단위와 44개 해역으로 세분화하여 발표합니다. 특보발표기준은 날씨누리(https://www.weather.go.kr/w/community/knowledge/standard.do)에서 확인가능합니다.		
API 유형	LINK	데이터 포맷	TEXT
공간범위	-	시간범위	-
키워드	특보구역,특보정보,기상특보,호우,대설	활용신청 수	1005
등록일	2024-02-08	수정일	2025-09-17

개요

육상·해상 예보구역의 코드·명칭과 유효기간 등을 제공하는 기준정보입니다. 단기·중기 등 예보서비스에 맞는 구역코드를 사용하며, 단기예보의 격자 좌표와 구분합니다.

주요요소	예보구역코드, 종료시각, 시작시각, 예보구역명(한글, 영문), 지역특성, 위도(deg.), 경도(deg.), 해발고도(m), 상위구역코드, 특보발표관서, 3시간예보관서, 단기예보관서, 주간예보관서, 순번 정보
제공범위	육상광역, 육상국지, 해상광역, 해상국지
제공기간	변경 시 업데이트
제공형식	XML
제공경로	공공데이터포털(https://www.data.go.kr) → 기상청 특보구역정보 조회서비스' 검색 → 오픈 API 활용신청
이용방법	구역 종류·기준시각을 선택해 코드와 명칭·유효기간을 조회합니다. 예보 또는 특보 발표 당시 유효한 코드를 적용하고, 코드의 앞자리 0을 보존합니다.

1.지점정보

- 관측지점정보
- 기상예보구역코드
- 기상특보구역코드

기상특보구역코드

오픈 API 정보

데이터 오류·개선 의견이 있으신가요? 데이터 개선요청 > 오류신고 및 문의 >

기본 정보

OpenAPI 명	기상청_특보구역정보 조회서비스		
분류체계	환경 - 대기	제공기관	기상청
관리부서명	국가기후데이터센터	관리부서 전화번호	042-481-7480

상세 및 제공정보

오픈API 에러코드 안내

설명	기상청에서 발표하는 육상 및 해상의 기상특보구역코드 정보입니다. 한파, 건조, 안개, 폭염, 지진해일, 폭풍해일, 호우, 대설, 태풍, 풍랑, 강풍 12개 기상현상에 대해 중대한 재해발생이 예상될 때 해당 지역에 대하여 기상특보의 발표 기준에 따라 주의보 및 경보로 구분하여 178개 시 · 군 단위와 44개 해역으로 세분화하여 발표합니다. 특보발표기준은 날씨누리(https://www.weather.go.kr/w/community/knowledge/standard.do)에서 확인가능합니다.		
API 유형	LINK	데이터 포맷	TEXT
공간범위	-	시간범위	-
키워드	특보구역,특보정보,기상특보,호우,대설	활용신청 수	1005
등록일	2024-02-08	수정일	2025-09-17

개요

기상특보가 적용되는 구역의 코드·명칭과 유효기간 등을 제공하는 기준정보입니다. 육상·해상 구역과 변경 이력을 확인하고, 특보 발표 당시의 구역코드를 적용합니다.

주요요소	특보구역(코드), 시작시각, 종료시각, 특보구역 한글명, 특보구역 영문명, 구역특성, 상위특보구역, 행정동 코드 등
제공범위	육상(시/군 단위), 해상(광역/국지/특정관리해역) 특보구역 코드
제공기간	변경 시 업데이트
제공형식	XML
제공경로	공공데이터포털(https://www.data.go.kr) → '기상청 특보구역정보 조회서비스' 검색 → 오픈 API 활용신청
이용방법	구역 종류·기준시각을 선택해 코드와 명칭·유효기간을 조회합니다. 예보 또는 특보 발표 당시 유효한 코드를 적용하고, 코드의 앞자리 0을 보존합니다.

데이터카탈로그



개요

기상청이 제공하는 데이터의 목록과 설명정보를 모아 안내하는 자료입니다. 데이터별 관측·생산내용, 제공기간·형식·경로 등을 확인하는 데 이용합니다. 웹 카탈로그와 이를 정리한 책자가 있으며, 관측값 자체를 담은 데이터 파일과는 구분합니다.

주요요소	데이터명·분류, 자료 설명, 주요요소, 제공범위·기간·형식·경로 및 이용방법 등 설명정보
제공범위	기상청 제공 데이터의 안내 항목(관측·예보·수치모델·기후·통계·역사자료·메타데이터 등)
제공기간	웹 안내의 갱신일 또는 발간본의 기준시점에 따름. 개별 데이터의 보유·조회기간은 해당 항목에서 확인
제공형식	HTML(웹 안내), 책자·전자문서(발간본); 연결된 데이터의 형식은 항목별 확인
제공경로	기상자료개방포털(https://data.kma.go.kr) → 카탈로그 https://data.kma.go.kr/catalog/schem/selectCatalogList.do 실제 데이터의 다운로드·API는 각 항목에 안내된 서비스에서 이용합니다.
이용방법	필요한 데이터명을 검색하고 설명·제공기간·형식·경로를 확인합니다. 실제 값은 연결된 제공 서비스에서 조회·다운로드하거나 API 활용신청 후 요청합니다. 신청형 자료는 보유부서의 제공 가능 범위와 승인 절차를 확인합니다.

기후통계 메타데이터



개요

기후통계요소의 정의·산출방법·통계기간·단위·표시 자릿수 등을 설명하는 메타데이터입니다. 통계자료를 해석하거나 다시 계산할 때 해당 요소의 기준을 확인하는 데 이용합니다.

주요요소	지상(강수, 현상, 기압, 기온, 바람, 습도, 수증기압, 신적설, 안개, 운량, 이슬점온도, 일사, 일조, 일조시간, 적설, 증기압, 증발량, 지면온도, 지중온도, 최저초상온도) AWS(강수량, 기온, 바람) 고층(기압, 기온, 바람, 이슬점온도, 지오포텐셜고도) 농업(강수량, 기온, 바람, 복사, 상대습도, 일조시간, 조도, 증발량, 지면온도, 지중온도, 지하수위, 초상온도, 토양수분) 북한(강수량, 기압, 기온, 바람, 상대습도, 운량, 이슬점온도, 현지일기) 부이(기압, 기온, 바람, 습도, 수온, 파고, 파주기) 등표(기압, 기온, 바람, 상대습도, 수온, 수위, 조위, 파고, 파주기) 파고부이(수온, 파고, 파주기)
제공범위	지상, AWS, 고층, 농업, 북한, 부이, 등표, 파고부이
제공기간	-
제공형식	CSV
제공경로	기상자료개방포털(https://data.kma.go.kr) → 데이터 → 메타데이터 → 통계정보
이용방법	통계요소명을 찾아 정의·산출방법·통계기간·단위·표시 자릿수를 확인합니다. 직접 집계할 때에는 해당 기준과 결측 처리 방법을 적용하고, 계산에 사용한 기준을 기록합니다.

기상기후데이터 카탈로그

데이터 활용사례 50

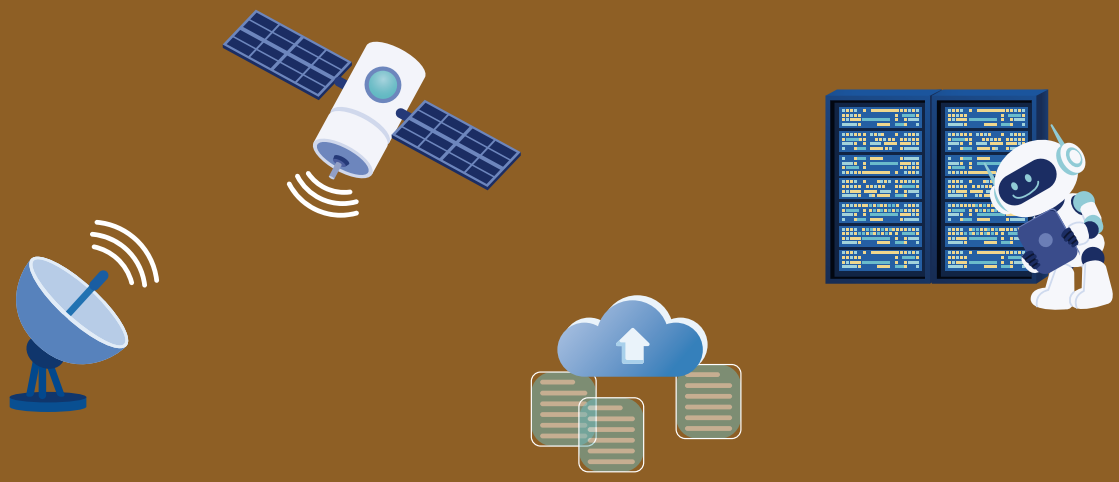
50 Data Use Cases



데이터 활용사례 50

기상관측

Meteorological Observation





이용 안내

데이터 활용사례 50은 기상기후데이터를 조회하고 분석하는 방법을 Python 예제로 소개합니다. 관심 있는 사례를 선택해 다음 사항을 확인한 후 활용해 주세요.

• 실행환경 준비

예제는 Python 3.9 이상을 기준으로 작성했습니다. 실행 전에 각 사례에서 사용하는 라이브러리를 설치해 주세요.

• 인증키 발급 및 활용신청

기상청 API허브(apihub.kma.go.kr)에서 인증키를 발급받고, 이용할 서비스의 활용신청을 완료해 주세요. 발급받은 인증키는 코드에 직접 입력하지 않고, KMA_API_KEY 환경변수에 등록해 사용합니다.

• 조회 날짜와 시각 확인

본문에 제시한 날짜와 시각은 예시입니다. 원하는 기간으로 변경하되, 해당 서비스에서 조회할 수 있는 기간을 확인해 주세요. 최근 자료만 제공하는 예보는 실행 시점에 조회 가능한 발표시각을 사용해야 합니다. 자료에 따라 한국표준시(KST) 또는 세계협정시(UTC)를 사용하므로 시간기준도 함께 확인해 주세요.

• 분석 결과 참고

표와 그림은 본문에 제시한 기간과 조건으로 분석한 예시입니다. 조회 기간·지점·조건이 다르거나 자료가 갱신·정정되면 결과가 달라질 수 있습니다. 과거 자료로 작성한 표와 그림은 해당 사례의 설명을 함께 참고해 주세요.

• 입력자료 준비

CSV·NetCDF 파일이나 별도로 가공한 자료를 사용하는 예제는 필요한 입력파일을 먼저 준비해 주세요. 파일의 변수명, 단위, 결측값, 좌표와 시간정보도 확인해야 합니다. 입력자료가 별도로 필요한 함수는 함수 정의만 실행해도 파일이 자동으로 만들어지는 것은 아닙니다.

• 지도 작성

지도 작성에 사용하는 Cartopy는 처음 실행할 때 지도 경계자료를 내려받을 수 있습니다. 인터넷 접속이 제한된 환경에서는 필요한 경계자료를 미리 준비해 주세요.

지상

1-①. 종관기상관측(ASOS)

기온·강수 특성과 냉난방도일 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS) > 1.4 일자료(기간 조회)

❶ 라이브러리 설치

API 조회에는 requests, 자료 처리에는 pandas·numpy, 그래프 작성에는 matplotlib, 지도 표출에는 cartopy를 사용합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 일자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
api_url = 'https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php'
COLS = ['TM', 'STN', 'WS_AVG', 'WR_DAY', 'WD_MAX', 'WS_MAX', 'WS_MAX_TM',
    'WD_INS', 'WS_INS', 'WS_INS_TM', 'TA_AVG', 'TA_MAX', 'TA_MAX_TM',
    'TA_MIN', 'TA_MIN_TM', 'TD_AVG', 'TS_AVG', 'TG_MIN', 'HM_AVG',
    'HM_MIN', 'HM_MIN_TM', 'PV_AVG', 'EV_S', 'EV_L', 'FG_DUR',
    'PA_AVG', 'PS_AVG', 'PS_MAX', 'PS_MAX_TM', 'PS_MIN', 'PS_MIN_TM',
    'CA_TOT', 'SS_DAY', 'SS_DUR', 'SS_CMB', 'SI_DAY', 'SI_60M_MAX',
    'SI_60M_MAX_TM', 'RN_DAY', 'RN_D99', 'RN_DUR', 'RN_60M_MAX']
def get_daily(tm1, tm2, stn=0):
  res = requests.get(
    api_url,
    params={
      "tm1": tm1,
      "tm2": tm2,
      "stn": stn,
      "help": 1,
      "authKey": API_KEY,
    },
    timeout=60,
  )
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.strip().startswith("#"))
  ]
  d = pd.DataFrame([r[:42] for r in rows if len(r) >= 50], columns=COLS)
  for c in COLS[2:]:
    d[c] = pd.to_numeric(d[c], errors="coerce")
    d[c] = (
      d[c].mask(d[c] <= -99.0)
      if c.startswith(("TA_", "TD_", "TS_", "TG_"))
```

```
        else d[c].mask(d[c] < 0)
    )
  d["STN"] = pd.to_numeric(d["STN"], errors="raise").astype(int)
  d["날짜"] = pd.to_datetime(d["TM"], format="%Y%m%d")
  return d
m = get_daily('20250801', '20250831')
print(len(m), m['STN'].nunique())
3007 97
```

표 1-1-1. 요소별 유효자료 비율 (2025년 8월)

요소	칸 이름	유효자료 수	유효자료 비율(%)
일 평균풍속	WS_AVG	2991	99.5
최대순간풍속	WS_INS	2996	99.6
일 평균기온	TA_AVG	3005	99.9
최고기온	TA_MAX	3007	100.0
최저기온	TA_MIN	3006	100.0
일 평균습도	HM_AVG	3005	99.9
일 평균 전운량	CA_TOT	3007	100.0
일조합	SS_DAY	2995	99.6
일사합	SI_DAY	2006	66.7
일 강수량	RN_DAY	1323	44.0
1시간 최다강수량	RN_60M_MAX	1319	43.9
최심 적설	SD_MAX	0	0.0

❸ 여름철 지역별 기온·강수량 조회

지점 좌표를 연결하고 월평균·합계를 지도에 표시합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
def get_station_coordinates(inf="SFC"):
  url = "https://apihub.kma.go.kr/api/typ01/url/stn_inf.php"
  response = requests.get(
    url,
    params={"inf": inf, "stn": 0, "help": 0, "authKey": API_KEY},
    timeout=60,
  )
  response.raise_for_status()
  response.encoding = "euc-kr"
  rows = []
  for line in response.text.splitlines():
    if not line.strip() or line.lstrip().startswith("#"):
      continue
    fields = line.split()
    if len(fields) < 11:
      raise ValueError("지점정보 행은 11개 열 이상")
    rows.append(
      {
        "STN": int(fields[0]),
        "지점": int(fields[0]),
        "경도": float(fields[1]),
```

지상

```
            "위도": float(fields[2]),
            "고도(m)": float(fields[4]),
            "지점명": fields[10],
        }
    )
  return pd.DataFrame(
    rows, columns=["STN", "지점", "경도", "위도", "고도(m)", "지점명"]
  )
stn = get_station_coordinates()
mm = (m.groupby('STN')
   .agg(평균기온=('TA_AVG', 'mean'),
        최고기온=('TA_MAX', 'max'),
        강수합=('RN_DAY', lambda s: s.sum(min_count=1)))
   .reset_index().merge(stn, on='STN'))     # stn : 지점정보 좌표표
fig = plt.figure(figsize=(6.0, 6.2))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([125.4, 130.1, 33.0, 38.7], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
          linewidth=0.6, edgecolor='#37474f')
sc = ax.scatter(mm['경도'], mm['위도'], c=mm['평균기온'], s=52, cmap='RdYlBu_r',
           edgecolors='#37474f', linewidths=0.4, transform=PC)
fig.colorbar(sc, ax=ax, label='8월 평균기온 (°C)')
plt.show()
```

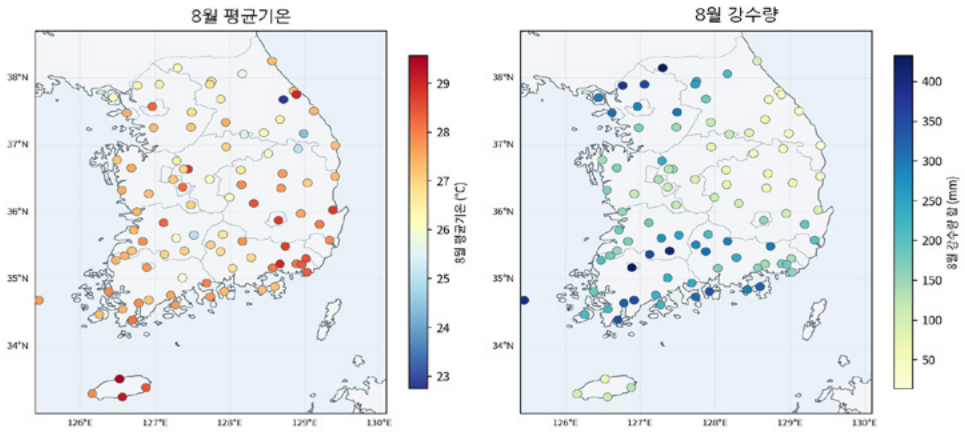


그림 1-1-1. 8월 평균기온과 강수량

표 1-1-2. 8월 기온 상위·하위 5곳

구분	지점	고도(m)	8월 평균기온(℃)	최고기온(℃)	강수합(mm)
더운 곳	제주	21	29.6	36.3	72
더운 곳	서귀포	52	29.2	34.0	104
더운 곳	강릉	27	29.0	38.0	41
더운 곳	북창원	51	29.0	37.0	147
더운 곳	대구	54	28.8	37.3	158
시원한 곳	대관령	772	22.7	32.0	52
시원한 곳	태백	714	24.4	34.9	43
시원한 곳	장수	407	25.0	33.9	251
시원한 곳	봉화	325	25.0	35.6	54
시원한 곳	제천	265	25.4	35.2	108

❹ 폭염일수와 일최저기온 참고지표 비교

일최고기온 33°C 이상일수와 일최저기온 25°C 이상일수를 집계합니다. 일최저기온 기준의 집계는 공식 열대야일수와 구분합니다.

```
# 한 해를 달마다 나눠 받아 이어 붙인다
from calendar import monthrange
g = pd.concat([get_daily(f'2025{mo:02d}01', f'2025{mo:02d}{monthrange(2025, mo)[1]:02d}') for mo
in range(1, 13)], ignore_index=True)
y = g.groupby('STN').apply(lambda x: pd.Series({
  '폭염일수': int((x['TA_MAX'] >= 33).sum()),    # 최고기온 33 °C 이상
  '열대야일수': int((x['TA_MIN'] >= 25).sum()), # 최저기온 25 °C 이상
  '강수일수': int((x['RN_DAY'] >= 0.1).sum()),
})).reset_index().merge(stn, on='STN')
print(y[['폭염일수', '열대야일수', '강수일수']].median())
```

```
폭염일수     30.0
열대야일수   11.0
강수일수    110.0
```

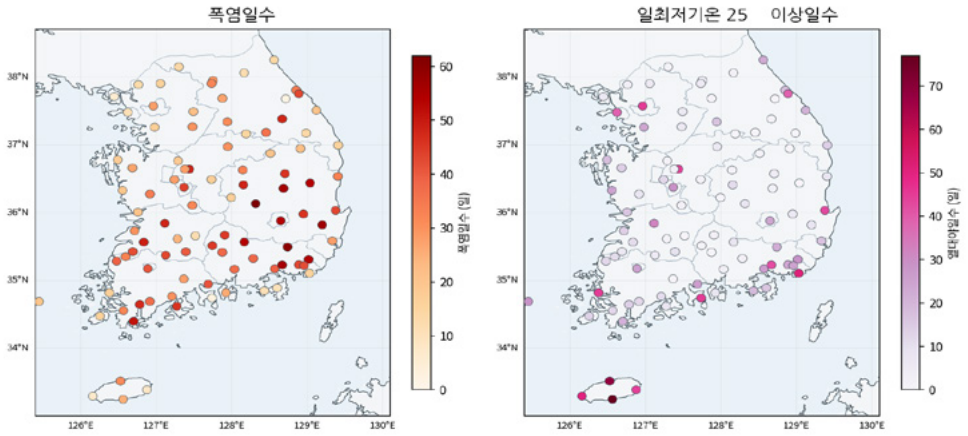


그림 1-1-2. 2025년 폭염일수와 열대야일수

표 1-1-3. 폭염·열대야 상위 지점

구분	지점	폭염일수	열대야일수	강수일수	강수량합(mm)
폭염 많은 곳	구미	62	7	106	1046
폭염 많은 곳	밀양	60	22	93	1314
폭염 많은 곳	경주시	57	8	97	1033
열대야 많은 곳	고산	7	50	119	930
열대야 많은 곳	서울	28	45	117	1582
열대야 많은 곳	목포	17	45	112	1247

지상

❺ 30년 냉난방도일 조회

기준온도와의 일별 차이를 연간 누적합니다. 난방은 18°C, 냉방은 24°C 기준입니다.

```
STNS = {108: '서울', 105: '강릉', 119: '수원', 133: '대전',
    143: '대구', 156: '광주', 159: '부산', 184: '제주'}
def degree_days(stn):
  parts = [
    get_daily(a, b, stn)
    for a, b in [
      ("19960101", "20051231"),
      ("20060101", "20151231"),
      ("20160101", "20251231"),
    ]
  ]
  x = pd.concat(parts).dropna(subset=["TA_AVG"])
  x["연도"] = x["날짜"].dt.year
  return (
    x.groupby("연도")
    .apply(
      lambda t: pd.Series(
        {
          "난방도일": (18 - t["TA_AVG"]).clip(lower=0).sum(),
          "냉방도일": (t["TA_AVG"] - 24).clip(lower=0).sum(),
          "연평균기온": t["TA_AVG"].mean(),
        }
      )
    )
    .reset_index()
  )
dd = pd.concat([degree_days(s).assign(지점=nm) for s, nm in STNS.items()])
print(np.polyfit(dd['연도'], dd['난방도일'], 1)[0] * 10,
   np.polyfit(dd['연도'], dd['냉방도일'], 1)[0] * 10)
-71 +48
```

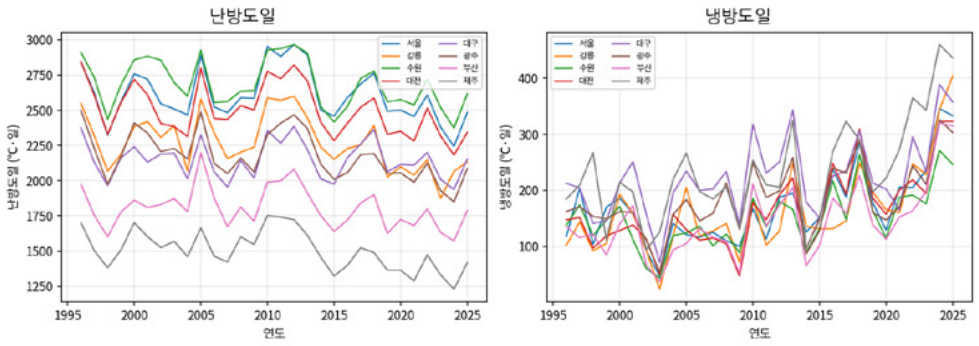


그림 1-1-3. 지점별 난방도일과 냉방도일 (1996~2025)

표 1-1-4. 지점별 도일 평균과 10년당 변화

지점	난방도일 평균	난방 10년당	냉방도일 평균	냉방 10년당	연평균 기온(°C)	기온 10년당(°C)
서울	2610	-54	169	+49	13.1	+0.38
강릉	2269	-96	163	+57	13.8	+0.61
수원	2687	-75	151	+34	12.8	+0.37
대전	2502	-91	165	+54	13.4	+0.52
대구	2150	-22	223	+40	14.6	+0.24
광주	2177	-99	185	+39	14.4	+0.49
부산	1807	-49	145	+43	15.2	+0.38
제주	1505	-83	234	+64	16.5	+0.53

1-②. 종관기상관측(ASOS)

기온·기압·시정의 일변화 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS) > 1.5 요소별 조회

❶ 요소별 시간자료 조회

```
https://apihub.kma.go.kr/api/typ01/url/kma_sfctm5.php?tm1=202609010000&tm2=202609012300&stn=0&obs=TA&authKey=...
# YYMMDDHHMI  STN      LON      LAT     HT    VAL
#     KST   ID     (deg)     (deg)    (m)
202609010000,90,128.56473000,38.25085000,17.53,22.5,=
202609010000,93,127.75443000,37.94738000,95.78,22.7,=
#  obs 를 바꾸면 TA·TD·HM·PA·PS·TS·VS·CA_TOT 가 같은 틀로 온다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
SFC_URL = "https://apihub.kma.go.kr/api/typ01/url/kma_sfctm5.php"
VARS = ["TA", "TD", "HM", "PA", "PS", "VS", "WS"]
VAR_KO = {"TA":("기온","°C"),"TD":("이슬점","°C"),"HM":("습도","%"),"PA":("현지기압","hPa"),"PS":("해면기압","hPa"),"VS":("시정","10m"),"WS":("풍속","m/s")}
def get_sfc(tm1, tm2, obs, stn=0):
    """선택한 요소의 지점별 시간자료를 조회합니다. 응답에는 경도·위도·노장 고도가 포함됩니다."""

    r = requests.get(
        SFC_URL,
        params={
            "tm1": tm1,
            "tm2": tm2,
            "stn": stn,
            "obs": obs,
            "authKey": API_KEY,
        },
        timeout=180,
    )
    if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
    r.encoding = "euc-kr"
    return r.text
def parse_sfc(text):
    """시각·지점·경도·위도·고도·관측값의 6개 필드를 읽습니다. 줄 끝의 ,= 표시는 제거합니다."""
    rows = []
    for ln in text.splitlines():
        s = ln.strip().rstrip("=").rstrip(",")
        if not s or s.startswith("#"):
```

지상

```
            continue
        p = s.split(",")
        if len(p) < 6:
            continue
        rows.append(
            {
                "시각": datetime.strptime(p[0], "%Y%m%d%H%M"),
                "지점": int(p[1]),
                "경도": float(p[2]),
                "위도": float(p[3]),
                "고도(m)": float(p[4]),
                "값": float(p[5]),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None if rows else ["시각", "지점", "경도", "위도", "고도(m)", "값"]
        ),
    )
def walk_vars(tm1, tm2, vars_, pause=0.2):
    """좌표를 먼저 기록하고 모든 요소를 같은 지점 순서로 재정렬한다."""
    data = {v: parse_sfc(get_sfc(tm1, tm2, v)) for v in vars_}
    valid = [d for d in data.values() if not d.empty]
    if not valid:
        raise ValueError("조회조건·메타데이터를 명시하세요: tm1, tm2, vars_")
    stations = pd.concat(valid).drop_duplicates("지점").sort_values("지점")
    ids = stations["지점"].tolist()
    out = [
        "@STN",
        " ".join(
            (
                f"{int(row['지점'])}:{row['경도']}:{row['위도']}:{row['고도(m)']}"
                for _, row in stations.iterrows()
            )
        ),
    ]
    for v, d in data.items():
        out.append(f"@V {v}")
        if d.empty:
            continue
        for tm, g in d.groupby("시각"):
            if g["지점"].duplicated().any():
                raise ValueError("지점·시각 중복 확인")
            values = g.set_index("지점")["값"].reindex(ids)
            out.append(
                tm.strftime("%Y%m%d%H%M")
                + " "
                + " ".join(("x" if pd.isna(x) else f"{x:g}" for x in values))
            )
        time.sleep(pause)
    return "\n".join(out)
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
    """지점 표 한 행과 요소별「시각 한 행」로 놀러 둔 판을 읽습니다."""
    lines = Path(path).read_text("utf-8").splitlines()
    stn = []
    for tok in lines[1].split():
        s, lo, la, ht = tok.split(":")
        stn.append(
            {
                "지점": int(s),
                "경도": float(lo),
                "위도": float(la),
                "고도(m)": float(ht),
            }
        )
    S = pd.DataFrame(
        stn, columns=None if stn else ["지점", "경도", "위도", "고도(m)"]
    )
    F, cur, i = ({}, None, 2)
    while i < len(lines):
```

```
    ln = lines[i]
    if ln.startswith("@V"):
      cur = ln.split()[1]
      F[cur] = {}
      i += 1
      continue
    p = ln.split()
    if p and cur:
      F[cur][datetime.strptime(p[0], "%Y%m%d%H%M")] = np.array(
        [np.nan if x == "x" else float(x) for x in p[1:]]
      )
    i += 1
  return (S, F)
```

❷ 전국 지점의 하루 자료 조회

좌표와 고도가 포함되어 지도와 시계열로 표출합니다.

```
def frame(S, F, var):
  """한 요소를 「시각 × 지점」 판으로 깐다."""
  tms = sorted(F[var])
  return pd.DataFrame([F[var][t] for t in tms], index=tms, columns=S["지점"])
def var_table(S, F):
  rows = []
  for v in VARS:
    a = np.concatenate([F[v][t] for t in sorted(F[v])])
    g = a[np.isfinite(a)]
    rows.append(
      {
        "부호": v,
        "뜻": VAR_KO[v][0],
        "단위": VAR_KO[v][1],
        "값 있는 칸": len(g),
        "결측(%)": round(100 * float(np.isnan(a).mean()), 1),
        "최소": round(float(g.min()), 1),
        "중앙값": round(float(np.median(g)), 1),
        "최대": round(float(g.max()), 1),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "부호",
        "뜻",
        "단위",
        "값 있는 칸",
        "결측(%)",
        "최소",
        "중앙값",
        "최대",
      ]
    ),
  )
```

지상

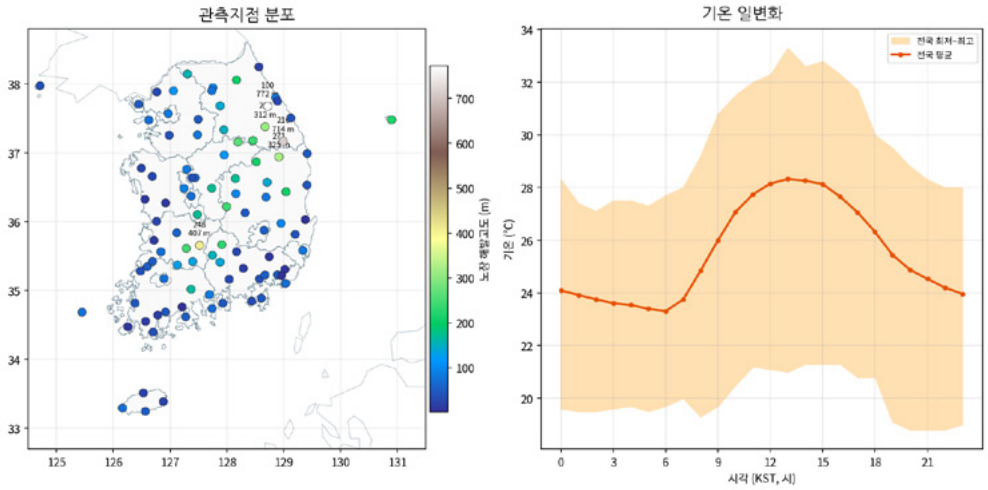


그림 1-2-1. 전국 지점의 하루 자료 조회

부호	뜻	단위	유효 격자 수	결측(%)	최소	중앙값	최대
TA	기온	℃	2328	0	18.8	25	33.3
TD	이슬점온도	℃	2328	0	17.4	23.1	26.4
HM	상대습도	%	2328	0	53	92	100
PA	현지기압	hPa	2328	0	924.6	1004	1012.6
PS	해면기압	hPa	2328	0	1007.9	1011	1013.4
TS	지면온도	℃	2328	0	20.5	25.9	52.1
VS	시정	m	2314	0.6	17	3683.5	5000
CA_TOT	전운량	1/10	2328	0	0	8	10

❸ 노장 고도를 활용한 기온감률 분석

관측값과 고도를 이용해 100 m 상승할 때의 기온 변화를 계산합니다.

```
def lapse_frame(S, F):
    """시각별로 노장 고도와 기온의 회귀계수를 계산합니다. 고도 100 m당 기온 변화와 상관계수를 비교합니다."""

    ta = frame(S, F, "TA")
    ht = S["고도(m)"].values
    rows = []
    for t, row in ta.iterrows():
        y = row.values
        m = np.isfinite(y)
        a, b = np.polyfit(ht[m], y[m], 1)
        rows.append(
            {
                "시각": t,
                "시": t.hour,
                "감률(℃/100m)": round(float(a * 100), 2),
                "상관": round(float(np.corrcoef(ht[m], y[m])[0, 1]), 2),
                "해면 기온(℃)": round(float(b), 1),
                "지점 수": int(m.sum()),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["시각", "시", "감률(℃/100m)", "상관", "해면 기온(℃)", "지점 수"]
        ),
    )
```

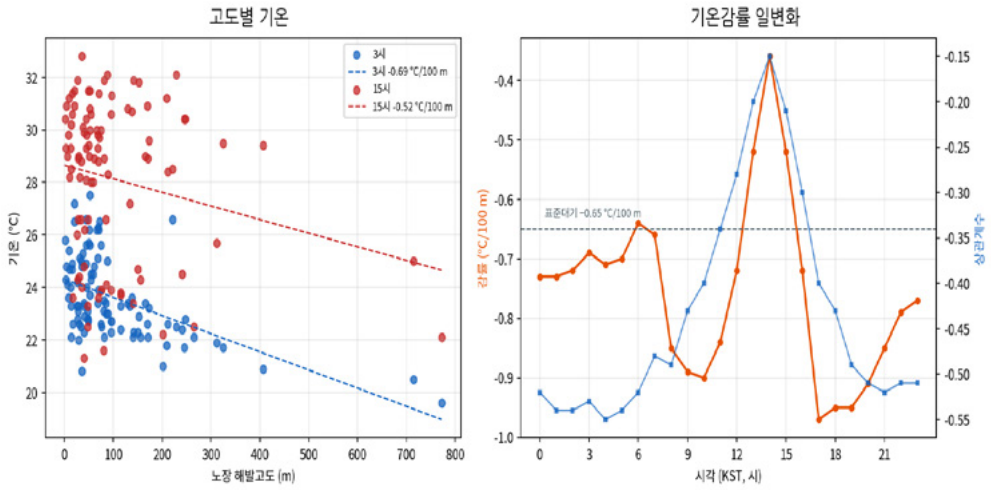


그림 1-2-2. 노장 고도를 활용한 기온감률 분석

시	감률(℃/100m)	상관	해면 기온(℃)	지점 수
0	-0.73	-0.52	24.8	97
1	-0.73	-0.54	24.7	97
2	-0.72	-0.54	24.5	97
21	-0.85	-0.52	25.4	97
22	-0.79	-0.51	25	97
23	-0.77	-0.51	24.7	97

❹ 현지기압과 해면기압을 활용한 고도 확인

고도와 비교해 자료의 일관성을 확인합니다.

```
def height_frame(S, F):
  """현지기압과 해면기압의 차이를 노장 고도와 비교합니다. 이 예의 고도 환산값은 회귀에 따른 참고값입니다."""


  pa = frame(S, F, "PA").mean()
  ps = frame(S, F, "PS").mean()
  d = pd.DataFrame(
    {
      "지점": S["지점"].values,
      "고도(m)": S["고도(m)"].values,
      "현지기압": pa.values,
      "해면기압": ps.values,
    }
  )
  d["기압차(hPa)"] = (d["해면기압"] - d["현지기압"]).round(2)
  m = d["기압차(hPa)"].notna() & d["고도(m)"].notna()
  a, b = np.polyfit(d.loc[m, "기압차(hPa)"], d.loc[m, "고도(m)"], 1)
  d["되돌린 고도(m)"] = (a * d["기압차(hPa)"] + b).round(1)
  d["차이(m)"] = (d["되돌린 고도(m)"] - d["고도(m)"]).round(1)
  return (
    d,
    a,
    b,
    float(np.corrcoef(d.loc[m, "기압차(hPa)"], d.loc[m, "고도(m)"])[0, 1]),
  )
```

지상

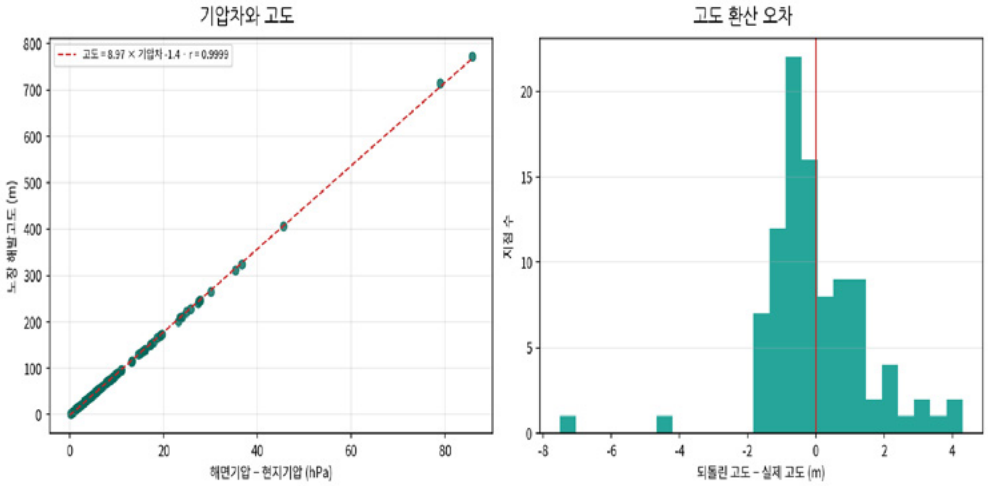


그림 1-2-3. 현지기압과 해면기압을 활용한 고도 확인

지점	고도(m)	현지기압	해면기압	기압차 (hPa)	되돌린 고도(m)	차이(m)
100	772.43	925.69	1011.46	85.77	767.8	-4.6
216	714.45	932.33	1011.32	78.99	707	-7.5
248	406.87	965.72	1011.31	45.59	407.4	0.5
272	211.32	987.45	1011.43	23.99	213.7	2.4
276	208.65	987.19	1010.74	23.55	209.8	1.2
211	202	988.28	1011.44	23.16	206.3	4.3

❺ 지면온도와 기온의 일변화 비교

```
def ground_frame(S, F):
  ta = frame(S, F, "TA")
  ts = frame(S, F, "TS")
  d = ts - ta
  return pd.DataFrame(
    {
      "시": [t.hour for t in d.index],
      "평균차(°C)": d.mean(axis=1).round(2).values,
      "최대차(°C)": d.max(axis=1).round(1).values,
      "최소차(°C)": d.min(axis=1).round(1).values,
      "기온평균(°C)": ta.mean(axis=1).round(1).values,
      "지면평균(°C)": ts.mean(axis=1).round(1).values,
    }
  )
```

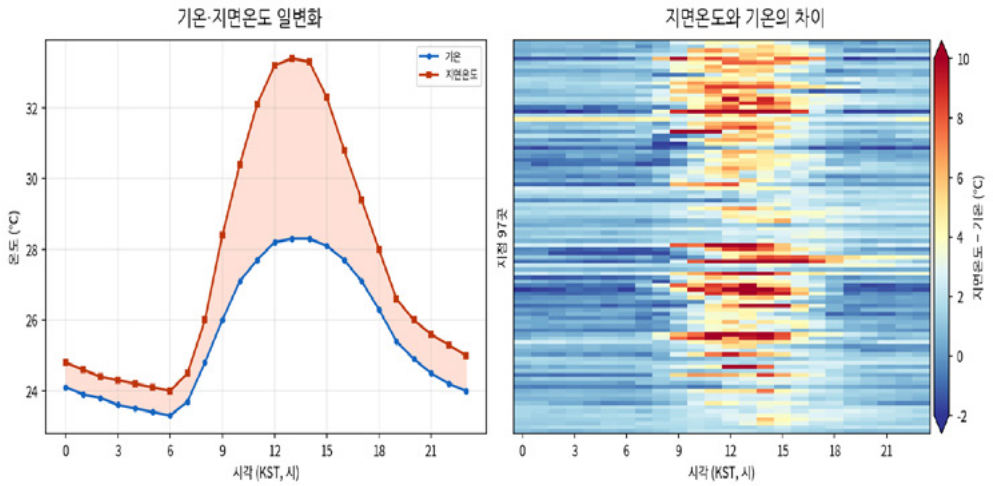


그림 1-2-4. 지면온도와 기온의 일변화 비교

시	평균차(°C)	최대차(°C)	최소차(°C)	기온평균(°C)	지면평균(°C)
0	0.66	4.5	-2	24.1	24.8
1	0.66	4.6	-1.8	23.9	24.6
2	0.64	4.6	-1.6	23.8	24.4
21	1.05	4.2	-2	24.5	25.6
22	1.08	3.8	-1.6	24.2	25.3
23	1.05	4.1	-1.7	24	25

❻ 기온·이슬점 차이와 시정의 관계

```
def moist_frame(S, F):
    """기온과 이슬점온도의 차이, 습도 및 시정을 같은 지점·시각에서 비교합니다. 안개 판단에는 관측 일기와 다른 기상조건도 확인합니다."""

    ta = frame(S, F, "TA")
    td = frame(S, F, "TD")
    hm = frame(S, F, "HM")
    vs = frame(S, F, "VS")
    ca = frame(S, F, "CA_TOT")
    d = pd.DataFrame(
        {
            "T-Td": (ta - td).values.ravel(),
            "습도": hm.values.ravel(),
            "시정": vs.values.ravel(),
            "전운량": ca.values.ravel(),
            "시": np.repeat([t.hour for t in ta.index], ta.shape[1]),
        }
    )
    return d.dropna(subset=["T-Td", "시정"])
def moist_table(d):
    d = d.copy()
    d["구간"] = pd.cut(
        d["T-Td"],
        [-0.1, 0.5, 1, 2, 4, 8, 40],
        labels=["0~0.5", "0.5~1", "1~2", "2~4", "4~8", "8+"],
    )
    t = (
        d.groupby("구간", observed=True)
        .agg(
            관수=("시정", "size"),
            습도중앙=("습도", "median"),
            시정중앙=("시정", "median"),
            시정하위10=("시정", lambda v: np.percentile(v, 10)),
```

지상

```
        안개비율=("시정", lambda v: 100 * float((v < 1000).mean())),
    )
    .round(1)
)
return t.reset_index()
```

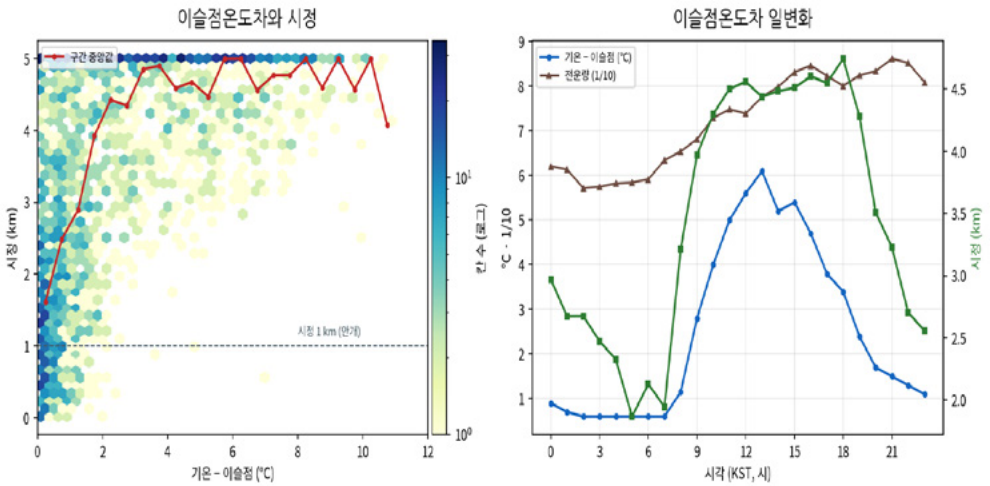


그림 1-2-5. 기온·이슬점 차이와 시정의 관계

구간	칸수	습도중앙	시정중앙	시정하위10	안개비율
0~0.5	542	100	1614.5	430.5	28.8
0.5~1	353	96	2486	668.2	17.3
1~2	468	92	3313.5	1154.2	8.1
2~4	387	84	4652	2852.6	0.8
4~8	490	72	4699.5	3316	0.4
8+	74	60	5000	4129.5	0

2.
방재기상관측(AWS)

집중호우와 강풍의 지역 분포 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 방재기상관측(AWS)

❶ AWS 분자료 조회와 집계

15분·60분·12시간·일 누적 강수량을 조회합니다.

```
https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min?tm1=202608170300&tm2=202608170309&stn=0&disp=0&help=0&authKey=...
# YYMMDDHHMI STN  WD1  WS1  WDS  WSS  WD10 WS10  TA  RE RN-15m RN-60m RN-12H RN-DAY …
202608170300 294 254.6  6.0 245. 10.3 247.5  5.5 24.1 1.0  38.6  124.5  402.1  215.6 …
#  RN-15m·RN-60m·RN-DAY 가 이미 누적되어 오므로 따로 더할 필요가 없다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
MIN_URL = "https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-aws2_min"
COL = {"TM":0, "STN":1, "WSS":5, "RN15":10, "RN60":11, "RNDAY":13} # 응답 help=1 헤더 확인
def get_min(tm1, tm2, stn=0, retry=3):
  """AWS 매분자료. 전체지점(stn=0)은 한 요청에서 최대 10분만 조회한다.
  더 긴 전국 자료는 10분 이하 구간으로 나누어 반복 호출한 뒤 결과를 합친다."""
  if stn == 0 and pd.to_datetime(tm2, format="%Y%m%d%H%M") - pd.to_datetime(
    tm1, format="%Y%m%d%H%M"
  ) > pd.Timedelta(minutes=9):
    raise ValueError(
      "전체지점(stn=0)은 10분 이하 구간으로 나누어 호출하세요."
    )
  params = {
    "tm1": tm1,
    "tm2": tm2,
    "stn": stn,
    "disp": 0,
    "help": 0,
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(MIN_URL, params=params, timeout=180)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      r.encoding = "euc-kr"
      return r.text
    except Exception as exc:
      if i == retry - 1:
```

지상

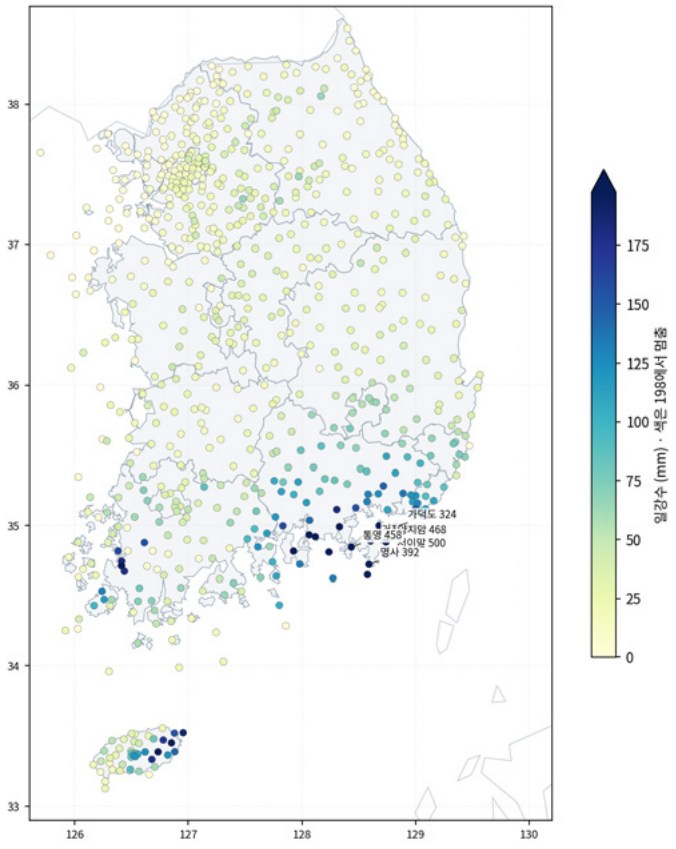
```
            raise
        log(f"{tm1} 재시도 {i + 1}: {type(exc).__name__}")
        time.sleep(2 * (i + 1))

def get_min_split(tm1, tm2, stn=0, pause=0.3):
    start = pd.to_datetime(tm1, format="%Y%m%d%H%M")
    end = pd.to_datetime(tm2, format="%Y%m%d%H%M")
    if start > end:
        raise ValueError("시작시각이 종료시각보다 늦음")
    agg = {}
    while start <= end:
        stop = min(start + pd.Timedelta(minutes=9), end)
        agg = fold_minutes(
            get_min(
                start.strftime("%Y%m%d%H%M"), stop.strftime("%Y%m%d%H%M"), stn
            ),
            agg,
        )
        start = stop + pd.Timedelta(minutes=1)
        time.sleep(pause)
    return (
        pd.DataFrame.from_dict(agg, orient="index")
        .rename_axis("지점")
        .reset_index()
    )
def fold_minutes(text, agg=None):
    """조회 구간 최댓값. 전부 결측인 요소는 NaN으로 유지한다."""
    agg = {} if agg is None else agg
    for line in text.splitlines():
        if not line.strip() or line.lstrip().startswith("#"):
            continue
        f = line.split()
        if len(f) <= max(COL.values()):
            raise ValueError("AWS 응답 행은 COL 정의 열 수 이상")
        stn = int(f[COL["STN"]])
        a = agg.setdefault(
            stn,
            {
                "일강수": np.nan,
                "최대60분": np.nan,
                "최대15분": np.nan,
                "최대순간풍속": np.nan,
                "시각": "",
            },
        )
        for target, field in [
            ("일강수", "RNDAY"),
            ("최대60분", "RN60"),
            ("최대15분", "RN15"),
            ("최대순간풍속", "WSS"),
        ]:
            value = float(f[COL[field]])
            if not np.isfinite(value) or value < 0:
                continue
            if not np.isfinite(a[target]) or value > a[target]:
                a[target] = value
                if target == "최대60분":
                    a["시각"] = f[COL["TM"]]
    return agg
```

❷ 관측지점별 일강수량 분포

같은 시·군 내 관측지점의 강수량 차이를 비교합니다.

```
def stn_frame(sec, coords):
  cols = ["지점", "일강수", "최대60분", "최대15분", "최대순간풍속", "시각"]
  d = pd.DataFrame([l.split(",") for l in sec["STN"]], columns=cols)
  d = d.astype(
    {
      "지점": int,
      "일강수": float,
      "최대60분": float,
      "최대15분": float,
      "최대순간풍속": float,
    }
  )
  return d.merge(coords, on="지점", how="left").dropna(subset=["경도"])
def load_stn(path):
  """지점정보에서 AWS 좌표와 이름만 가져옵니다."""
  sec, cur, rows = ({}, None, [])
  for line in Path(path).read_text("utf-8").splitlines():
    if line.startswith("#"):
      continue
    if line.startswith("@"):
      cur = line[1:].split(None, 1)[0]
      continue
    if cur == "AWS" and line:
      f = line.split(",")
      rows.append(
        {
          "지점": int(f[0]),
          "경도": float(f[1]),
          "위도": float(f[2]),
          "지점명": f[4],
        }
      )
  return pd.DataFrame(rows)
```



지상

그림 2-1. 관측지점별 일강수량 분포

지점	지점명	일강수	최대60분	최대15분	최대순간풍속	시각
294	거제	654.3	124.5	38.6	10.3	0232
909	서이말	499.5	100.5	34.5	16.2	0121
313	양지암	468.5	85.5	27	10.3	0425
162	통영	457.7	97.4	33	14.1	0611
947	명사	392	77.5	25	13.6	1218

❸ 지점별 1시간 최대 강수량

지점별 RN-60m의 하루 최댓값으로 집중호우 분포를 비교합니다.

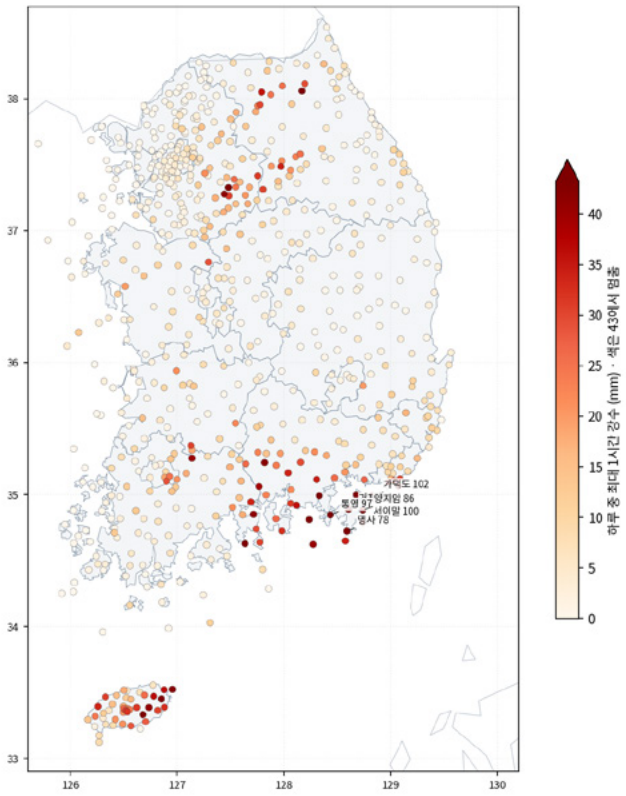


그림 2-2. 지점별 1시간 최대 강수량

지점	지점명	최대60분	시각	최대15분	일강수
294	거제	124.5	0232	38.6	654.3
921	가덕도	101.5	0359	31.5	324.5
909	서이말	100.5	0121	34.5	499.5
162	통영	97.4	0611	33	457.7
313	양지암	85.5	0425	27	468.5
947	명사	77.5	1218	25	392
930	사량도	71.5	0704	26.5	260.5
892	성산수산	67.5	0414	24.5	298

❹ 시간에 따른 강수 집중지역 변화

```
def min_frame(sec):
  cols = ["시각", "강수지점수", "전국최대60분", "지점", "전국최대순간풍속"]
  d = pd.DataFrame([l.split(",") for l in sec["MIN"]], columns=cols)
  d = d.astype(
    {
      "강수지점수": int,
      "전국최대60분": float,
      "지점": int,
      "전국최대순간풍속": float,
    }
  )
  d["분"] = d["시각"].str[:2].astype(int) * 60 + d["시각"].str[2:].astype(int)
  return d
```

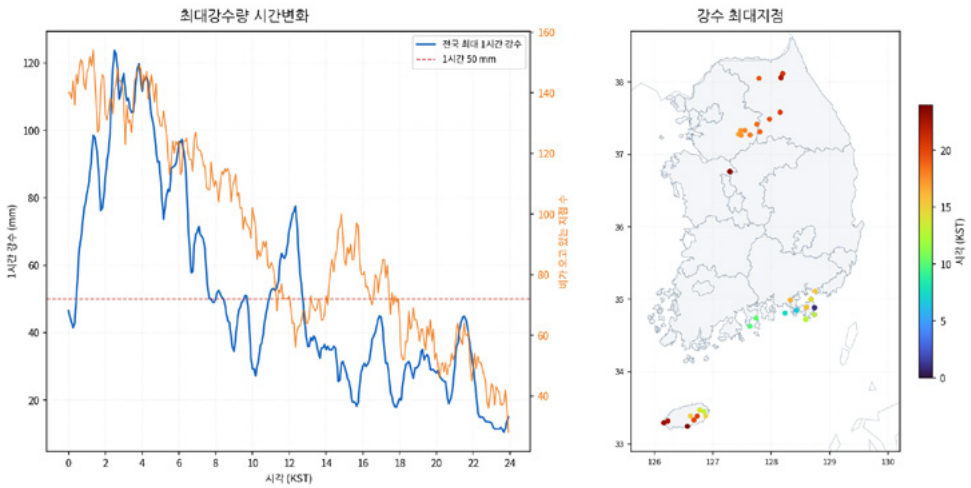


그림 2-3. 시간에 따른 강수 집중지역 변화

❺ 한 지점의 분 단위 강수 변화

강수량의 증가 시점과 지속시간을 확인합니다.

```
def rep_frame(sec):
  rows = []
  for line in sec["REP"]:
    stn, r60, rday = line.split("|")
    a, b = (r60.split(), rday.split())
    for i, (x, y) in enumerate(zip(a, b)):
      rows.append(
        {
          "지점": int(stn),
          "분": i * 5,
          "60분강수": float(x) if x else np.nan,
          "일누적": float(y) if y else np.nan,
        }
      )
  return pd.DataFrame(
    rows, columns=None if rows else ["지점", "분", "60분강수", "일누적"]
  )
```

지상

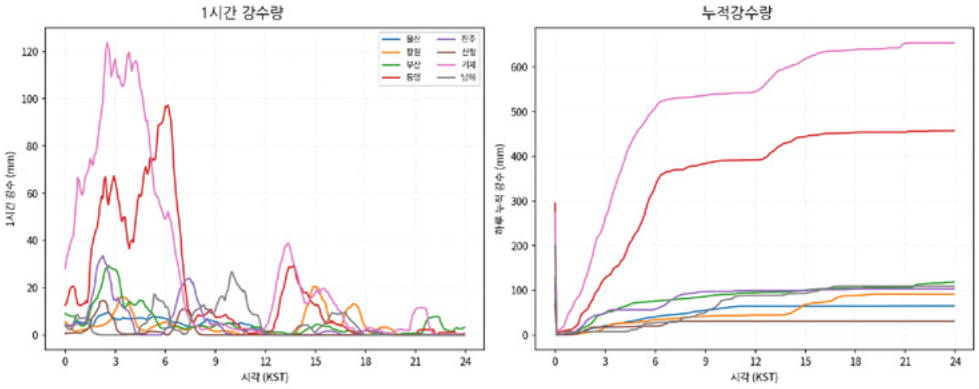


그림 2-4. 한 지점의 분 단위 강수 변화

❻ 반경 15 km 이내 지점과의 강수량 비교

```
def rain_gap(d, r=15.0):
  """반경 내 인접 관측지점의 일강수량을 비교합니다. 관측지점별 강수량과 인접 지점 평균의 차이를 계산합니다."""

  rows = []
  lon, lat, v = (d["경도"].values, d["위도"].values, d["일강수"].values)
  for i in range(len(d)):
    dx = (lon - lon[i]) * 111.32 * np.cos(np.radians(lat[i]))
    dy = (lat - lat[i]) * 110.57
    near = (np.hypot(dx, dy) <= r) & (np.arange(len(d)) != i)
    if near.sum() >= 2:
      rows.append(
        {
          "지점": d["지점"].values[i],
          "지점명": d["지점명"].values[i],
          "자기값": v[i],
          "이웃수": int(near.sum()),
          "이웃평균": round(float(v[near].mean()), 1),
          "차이": round(float(v[i] - v[near].mean()), 1),
        }
      )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["지점", "지점명", "자기값", "이웃수", "이웃평균", "차이"]
    ),
  )
```

▸ 실행 결과

지점명	자기값	이웃수	이웃평균	차이
거제	654.3	2	355.2	299
성산수산	298	8	136.1	161.9
시종	156.5	2	4	152.5
하이	289.9	2	206	83.9
한남	178	9	97.4	80.6
제주가시리	206.5	9	134.2	72.3
목포	161.5	5	89.9	71.6
솔라시도	196.3	4	131.2	65.2

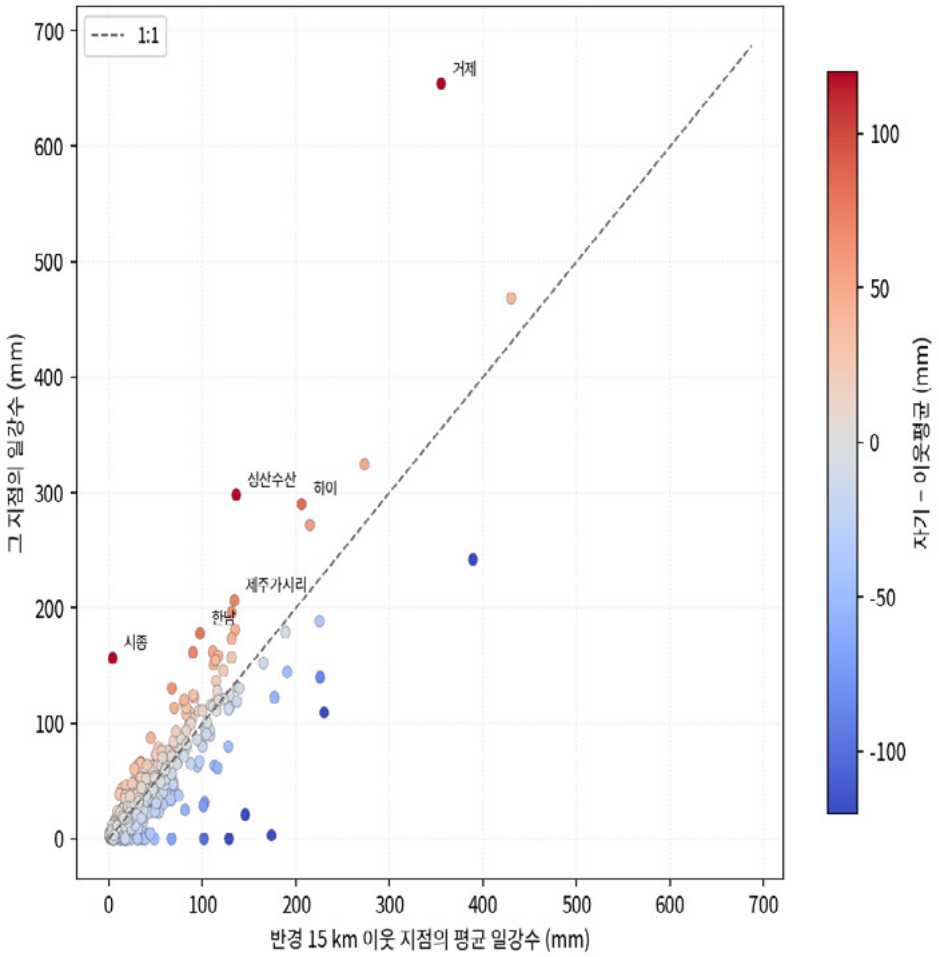


그림 2-5. 반경 15 km 이내 지점과의 강수량 비교

❼ 최대순간풍속의 지역 분포

```
def warn_table(d):
  """분석 기준값을 초과한 관측지점 수를 집계합니다. 집계 결과는 공식 기상특보의 발표 여부를 의미하지 않습니다."""
  out = {
    "관측 지점": f"{len(d)}곳",
    "일강수 100 mm 이상": f"{int((d['일강수'] >= 100).sum())}곳",
    "일강수 200 mm 이상": f"{int((d['일강수'] >= 200).sum())}곳",
    "1 시간 30 mm 이상": f"{int((d['최대60분'] >= 30).sum())}곳",
    "1 시간 50 mm 이상": f"{int((d['최대60분'] >= 50).sum())}곳",
    "1 시간 70 mm 이상": f"{int((d['최대60분'] >= 70).sum())}곳",
    "최대순간풍속 14 m/s 이상": f"{int((d['최대순간풍속'] >= 14).sum())}곳",
  }
  return out
```

지상

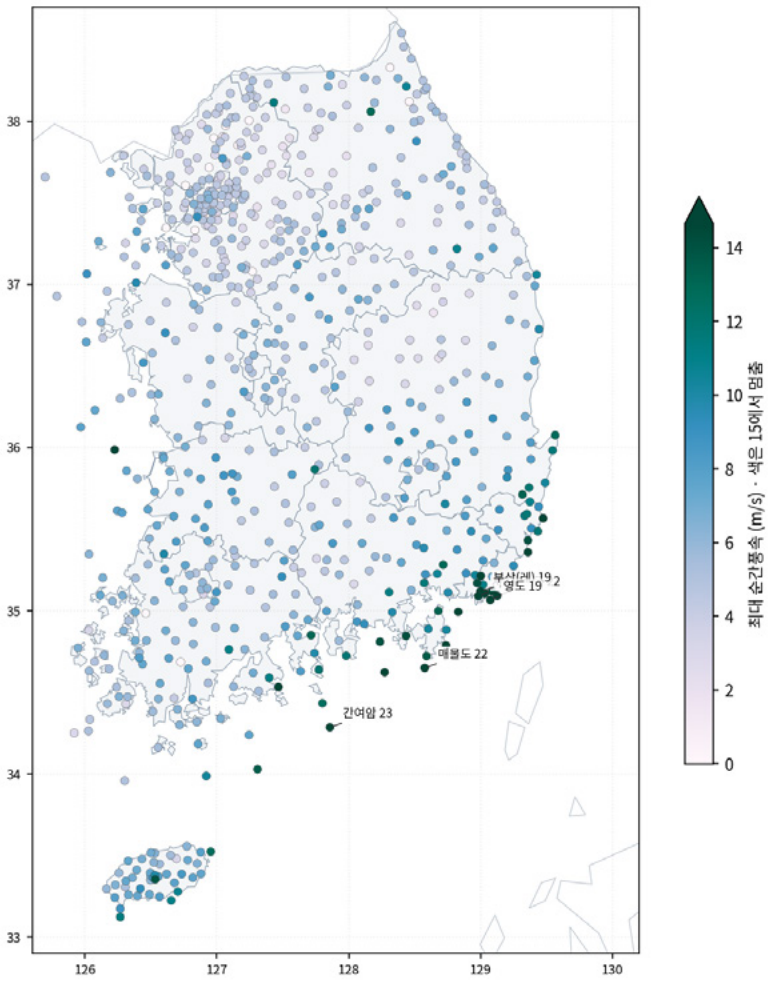


그림 2-6. 최대순간풍속의 지역 분포

3. 계절관측

지역별 개화 시기 비교

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 계절관측

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 계절관측 자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/sfc_ssn.php'
params = {'stn': 0, 'tm1': '20050101', 'tm2': '20251231',
        'ssn': 205, 'help': 0, 'authKey': API_KEY}
res = requests.get(api_url, params=params, timeout=60)
res.encoding = 'euc-kr'
rows = [ln.strip().rstrip(',').split(',') for ln in res.text.splitlines()
     if ln.strip() and not ln.startswith('#')]
d = pd.DataFrame([[c.strip() for c in r[:5]] for r in rows],
            columns=['연도', '지점', '관측일', 'SSN_ID', 'SSN_MD'])
d = d.astype({'연도': int, '지점': int, 'SSN_ID': int, 'SSN_MD': int})
d['관측일'] = pd.to_datetime(d['관측일'])
print(d.head(3))
   연도  지점       관측일 SSN_ID SSN_MD
0 2005  90 2005-03-25   205   201
1 2005  90 2005-04-11   205   202
2 2005  95 2005-04-10   205   201
```

SSN_ID는 요소, SSN_MD는 현상 코드입니다. ssn=205(벚나무)의 현상은 201 발아·202 개화·203 만발이며, 코드표는 API허브의 SSN_ID.pdf·SSN_MD.pdf를 참고합니다.

지상

SSN_MD	현상	건수
201	발아	894
202	개화	893
203	만발	275

표 3-1. ssn=205(벚나무)에서 관측된 현상별 건수

❸ 개화일 지도 표출

지점 좌표를 연결해 개화 시기를 지도에 표시합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
d['일수'] = pd.to_datetime('2000-' + d['관측일'].dt.strftime('%m-%d'), format='%Y-%m-%d').dt.dayofyear
# 윤년 영향 제거
def get_station_coordinates(inf="SFC"):
  url = "https://apihub.kma.go.kr/api/typ01/url/stn_inf.php"
  response = requests.get(
    url,
    params={"inf": inf, "stn": 0, "help": 0, "authKey": API_KEY},
    timeout=60,
  )
  if response.status_code != 200:
    raise RuntimeError(f"지점정보 HTTP {response.status_code}")
  response.encoding = "euc-kr"
  rows = []
  for line in response.text.splitlines():
    if not line.strip() or line.lstrip().startswith("#"):
      continue
    fields = line.split()
    if len(fields) < 11:
      raise ValueError("지점정보 열 구조 확인")
    rows.append(
      {
        "STN": int(fields[0]),
        "지점": int(fields[0]),
        "경도": float(fields[1]),
        "위도": float(fields[2]),
        "고도(m)": float(fields[4]),
        "지점명": fields[10],
      }
    )
  return pd.DataFrame(
    rows, columns=["STN", "지점", "경도", "위도", "고도(m)", "지점명"]
  )
stn = get_station_coordinates()
bloom = d[d['SSN_MD'] == 202].merge(stn, on='지점')  # stn : 지점 좌표표
m = (bloom.groupby('지점')
   .agg(일수=('일수', 'mean'), 위도=('위도', 'first'),
      경도=('경도', 'first'), 해=('연도', 'size'))
   .reset_index())
m = m[m['해'] >= 5]                    # 5해 이상 관측한 지점만
fig = plt.figure(figsize=(6.2, 6.4))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([125.4, 130.1, 33.0, 38.7], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
         linewidth=0.6, edgecolor='#37474f')
vmax = np.percentile(m['일수'], 95)   # 한 곳이 색을 다 먹지 않도록 상위에서 멈춘다
sc = ax.scatter(m['경도'], m['위도'], c=m['일수'], s=46, cmap='RdYlBu_r',
          vmin=m['일수'].min(), vmax=vmax,
          edgecolors='#37474f', linewidths=0.4, transform=PC, zorder=5)
fig.colorbar(sc, ax=ax, extend='max', label='평균 개화일 (1월 1일부터의 일수)')
plt.show()
```

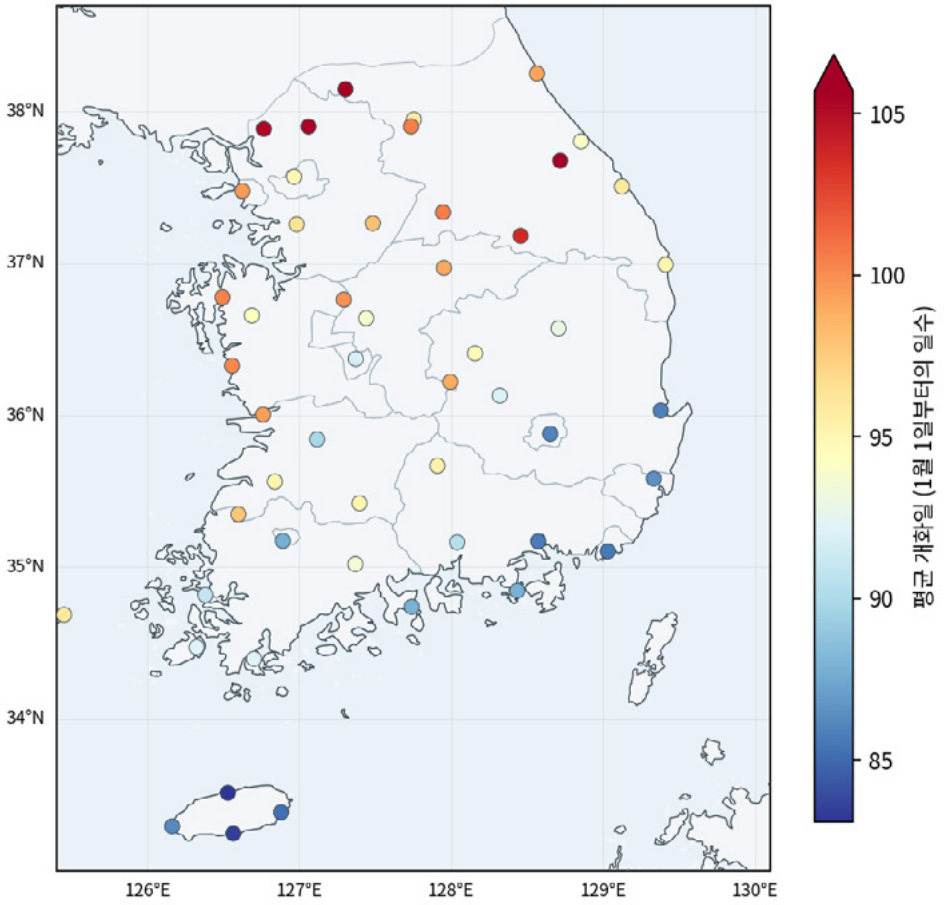


그림 3-1. 지점별 벚나무 평균 개화일

구분	지점번호	위도	평균 개화일	관측 해
이른 곳	184	33.51	03월 23일	21
이른 곳	189	33.25	03월 23일	18
이른 곳	188	33.39	03월 25일	11
이른 곳	159	35.1	03월 25일	21
이른 곳	155	35.17	03월 25일	21
늦은 곳	100	37.68	05월 04일	11
늦은 곳	102	37.97	04월 18일	21
늦은 곳	95	38.15	04월 15일	11
늦은 곳	98	37.9	04월 14일	11
늦은 곳	99	37.89	04월 14일	11

표 3-2. 개화가 이른 5곳과 늦은 5곳

지상

❹ 위도와 개화일

```
x, y = m['위도'].values, m['일수'].values
slope, intercept = np.polyfit(x, y, 1)
print(f'위도 1도당 {slope:+.1f}일, 상관 {np.corrcoef(x, y)[0, 1]:.2f}')
위도 1도당 +3.8일, 상관 0.70
```

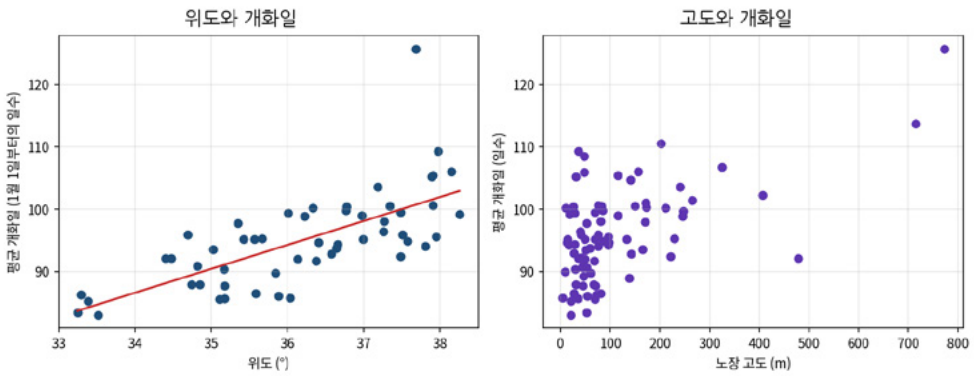


그림 3-2. 위도와 평균 개화일

❺ 연도별 개화일 변화

전 기간을 꾸준히 관측한 지점만 남겨 연도별 개화일을 비교합니다.

```
span = 2025 - 2005 + 1
keep = bloom.groupby('지점')['연도'].nunique().pipe(lambda s: s[s == span]).index
yr = (bloom[bloom['지점'].isin(keep)]
    .groupby('연도')['일수'].mean().reset_index())
ys, yi = np.polyfit(yr['연도'], yr['일수'], 1)
print(f'{len(keep)}개 지점 · 10년당 {ys * 10:+.1f}일')
20개 지점 · 10년당 -4.2일
```

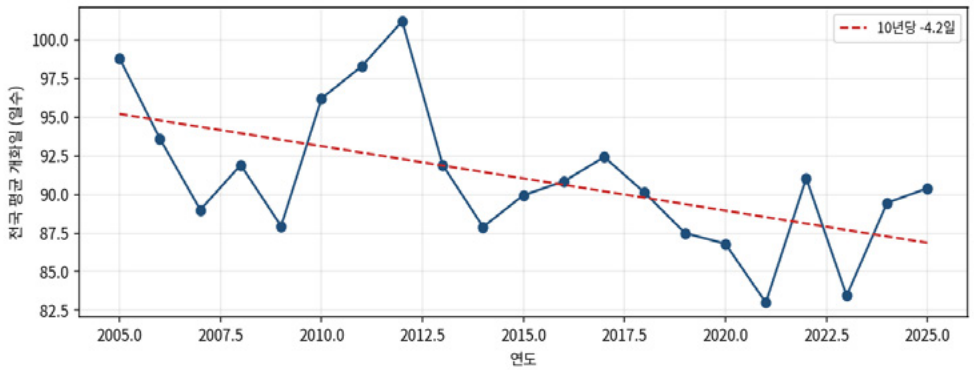


그림 3-3. 전 기간 관측한 지점의 연도별 평균 개화일

연도	지점 수	평균 개화일	일수
2005	20	04월 08일	98.8
2010	20	04월 06일	96.2
2015	20	03월 30일	90
2020	20	03월 26일	86.8
2021	20	03월 24일	83
2022	20	04월 01일	91
2023	20	03월 24일	83.4
2024	20	03월 29일	89.4
2025	20	03월 31일	90.4

표 3-3. 연도별 평균 개화일 (5년 간격과 최근 6년)

지상

4. 황사관측(PM10)

관측망별 농도 변화 비교

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 황사관측(PM10)

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 황사 시간통계 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
import re
api_url = 'https://apihub.kma.go.kr/api/typ01/url/dst_pm10_hr.php'
def get_pm10(tm):
  res = requests.get(
    api_url, params={"tm": tm, "help": 1, "authKey": API_KEY}, timeout=60
  )
  if not res.ok:
    res.raise_for_status()
  rows = []
  for ln in res.text.splitlines():
    if not ln.strip() or ln.lstrip().startswith("#"):
      continue
    c = re.sub("\\(\\s+(\\d+)\\s*\\)", "(\\1)", ln).split()
    if len(c) < 6:
      raise ValueError("PM10 응답 행은 6개 열 이상")
    match = re.fullmatch("(-?\\d+(?:\\.\\d+)?)\\((\\d+)\\)", c[3])
    if match is None:
      raise ValueError("AVG(CNT) 형식: 값(건수)")
    avg, cnt = (float(match[1]), int(match[2]))
    rows.append(
      [
        c[0],
```

```
            c[1],
            int(c[2]),
            avg if cnt > 0 and avg >= 0 else np.nan,
            cnt,
            float(c[4]),
            float(c[5]),
          ]
        )
    d = pd.DataFrame(
      rows, columns=["TM", "ORG", "STN", "AVG", "CNT", "MIN", "MAX"]
    )
    for col in ["MIN", "MAX"]:
      d[col] = d[col].where((d[col] >= 0) & (d["CNT"] > 0))
    return d
d = get_pm10('202503200000')
print(d.groupby('ORG')['STN'].nunique())
ORG
cma   11
kcc    1
kma   29
moe   663
```

ORG	기관	지점 수	자료 행
kma	기상청	29	3607
moe	환경부	663	82471
cma	중국기상국	11	1370
kcc	기타	1	125

표 4-1. 기관별 지점 수와 조회된 자료 행

3시간 간격으로 데이터를 불러옵니다.

```
times = pd.date_range('2025-03-08', '2025-03-23 21:00', freq='3h')
frames = []
for t in times:
  frames.append(get_pm10(t.strftime('%Y%m%d%H%M')))
  time.sleep(0.3)
g = pd.concat(frames)
print(len(times), g['ORG'].value_counts().to_dict())
125 {'moe': 82471, 'kma': 3607, 'cma': 1370, 'kcc': 125}
```

❸ 16일간 PM10 변화

기관별 유효 관측지점의 단순평균·최댓값을 비교합니다. 관측망 차이로 국가 대표평균과 같지 않습니다.

```
import matplotlib.pyplot as plt
k = g[g['ORG'] == 'kma'].groupby('TM')['AVG'].agg(['mean', 'max'])
c = g[g['ORG'] == 'cma'].groupby('TM')['AVG'].agg(['mean', 'max'])
m = g[g['ORG'] == 'moe'].groupby('TM')['AVG'].agg(['mean', 'max'])
fig, axes = plt.subplots(2, 1, figsize=(11.0, 5.8), sharex=True)
axes[0].fill_between(c.index, 0, c['mean'], color='#FFCC80')
axes[0].plot(c.index, c['max'], color='#E65100')
axes[1].fill_between(k.index, 0, k['mean'], color='#90CAF9')
axes[1].plot(k.index, k['max'], color='#0D47A1')
axes[1].plot(m.index, m['mean'], color='#2E7D32')
plt.show()
```

지상

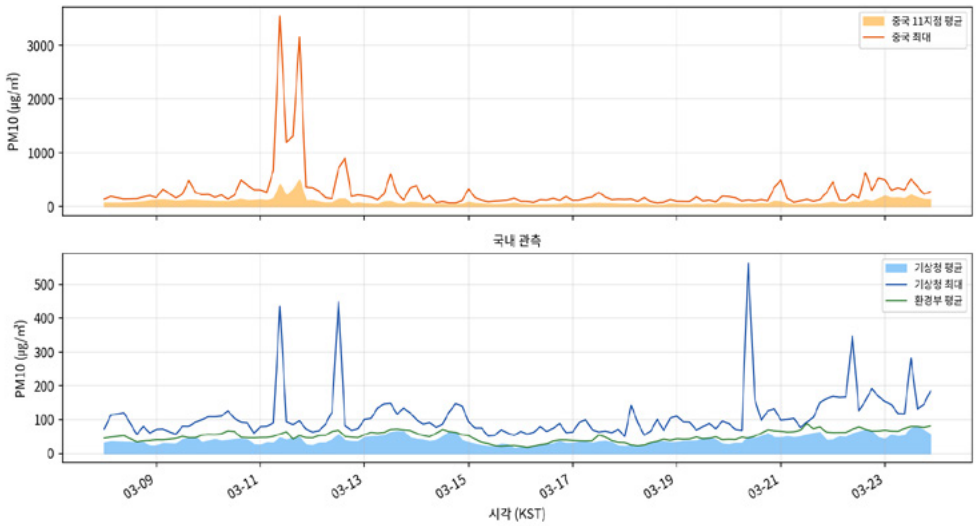


그림 4-1. 2025-03-08 00시~2025-03-23 21시 PM10 시간 흐름

지점번호	지점명	평균(μg/m³)	최대(μg/m³)	시각 수
268	진도군	63	562	125
232	천안	48	448	125
136	안동	33	281	125
152	울산	47	192	125
146	전주	72	185	125
93	북춘천	31	160	125
116	116	46	148	125
108	서울	50	129	125

표 4-2. 기간 최대가 컸던 기상청 지점 8곳

❹ 중국·국내 PM10의 시차 비교

```
def lagcorr(a, b, L):
    return np.corrcoef(a[L:] if L else a, b[: len(b) - L] if L else b)[0, 1]
s = k[['mean']].join(c[['mean']], lsuffix='_국내', rsuffix='_중국').dropna()
rr = [lagcorr(s['mean_국내'].values, s['mean_중국'].values, L)
      for L in range(17)]
print([round(v, 2) for v in rr[:6]])
[0.23, 0.09, 0.07, ...]
```

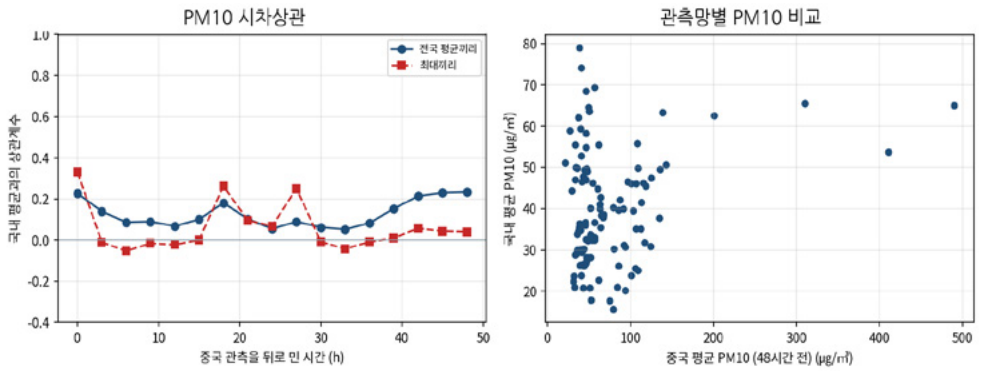


그림 4-2. 중국 관측 시간에 따른 상관

시간(h)	평균끼리 상관	최대끼리 상관
0	0.23	0.33
6	0.09	-0.05
12	0.07	-0.02
18	0.18	0.26
24	0.05	0.07
30	0.06	-0.01
36	0.08	-0.01
42	0.21	0.06
48	0.23	0.04

표 4-3. 시차별 상관계수

WMO 지점번호	블록	평균(μg/m³)	최대(μg/m³)
53336	53	124	3543
54497	54	129	3154
53276	53	83	1302
53068	53	70	1193
54135	54	114	897
54157	54	105	629

표 4-4. 중국기상국 지점의 기간 평균과 최대

❺ 국내 관측망의 PM10 비교

기상청과 환경부 관측망의 PM10 농도를 비교합니다. 관측지점의 수와 위치가 다르므로 평균값의 차이를 관측망 차이와 함께 해석합니다.

```
both = k[['mean']].join(m[['mean']], lsuffix='_kma', rsuffix='_moe').dropna()
print(both.corr().iloc[0, 1].round(2), len(both))
0.85 125
```

지상

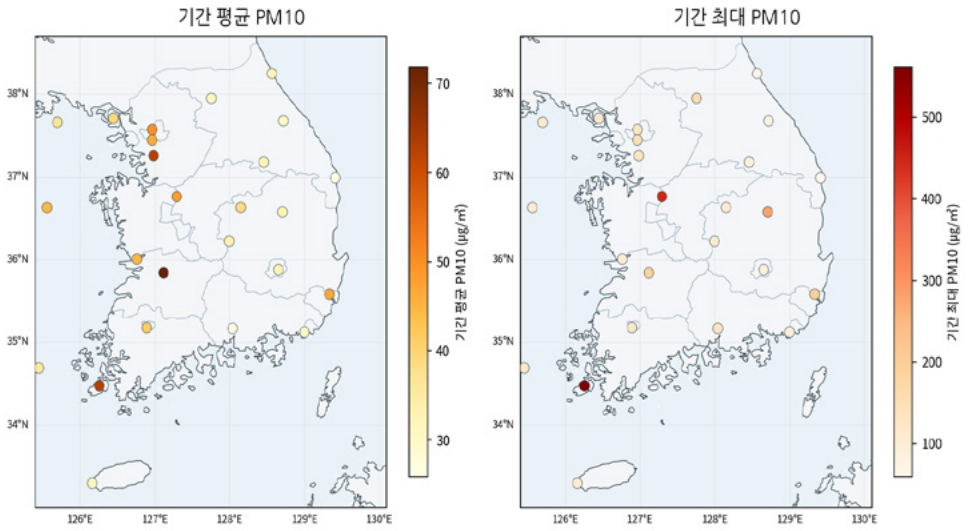


그림 4-3. 기상청 황사관측 지점의 기간 평균과 기간 최대

5.
낙뢰관측자료

발생 위치와 시간대 분석

데이터 활용 경로(API)

기상청 API허브 > 레이더 > 낙뢰관측

❶ 낙뢰 자료 조회와 격자 집계

```
https://apihub.kma.go.kr/api/typ01/url/lgt_pnt.php?tm=202607091500&authKey=...
#      시각        경도     위도  전류(kA)
20260709143022  128.817  36.254   -5.9
20260709143347  122.263  40.352   53.6
# 전류 부호가 극성이다 — 음수면 음극, 양수면 양극
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LGT_URL = "https://apihub.kma.go.kr/api/typ01/url/lgt_pnt.php"
```

KO : load_regions 반환 객체. EPSG:4326 경계 GeoJSON에서 생성
필수 입력: KO. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.
조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def load_regions(path, name_field="name"):
  """EPSG:4326 시도 경계 GeoJSON. name_field는 실제 속성 열 이름."""
  import json
  from shapely.geometry import shape, Point


  payload = json.loads(Path(path).read_text("utf-8"))
  features = payload.get("features", [])
  polygons = []
  for feature in features:
    properties = feature.get("properties", {})
    if name_field not in properties:
      raise ValueError("GeoJSON 시도명 속성 열 확인")
    geometry = shape(feature["geometry"])
    if not geometry.is_valid:
      raise ValueError("유효하지 않은 행정경계")

    polygons.append((properties[name_field], geometry))

  def assign(lon, lat):
    answer = []
    for x, y in zip(lon, lat):
```

지상

```
        if not np.isfinite(x) or not np.isfinite(y):
            answer.append(None)
            continue
        point = Point(float(x), float(y))
        matches = [name for name, geom in polygons if geom.covers(point)]

        answer.append(matches[0] if len(matches) == 1 else None)

    return answer

from types import SimpleNamespace

return SimpleNamespace(assign_province=assign)
```

시·도별 집계에는 EPSG:4326 좌표계의 행정경계 GeoJSON과 shapely가 필요합니다. load_regions()의 name_field에 시·도명 속성을 지정하고 반환 객체를 KO에 전달합니다. 경계가 중복되거나 행정구역이 확인되지 않는 위치는 미분류로 처리합니다.

```
def get_lgt(tm, retry=3):
    """1시간 동안 발생한 낙뢰를 조회합니다. 각 행에는 발생시각·경도·위도·전류(kA)가 포함됩니다."""

    for i in range(retry):
        try:
            r = requests.get(
                LGT_URL, params={"tm": tm, "authKey": API_KEY}, timeout=60
            )
            if not r.ok:
                raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
            r.encoding = "euc-kr"
            return r.text
        except Exception as exc:
            if i == retry - 1:
                raise
            log(f"{tm} 재시도 {i + 1}: {type(exc).__name__}")
            time.sleep(2 * (i + 1))
def parse_lgt(text):
    rows = []
    for line in text.splitlines():
        if not line.strip() or line.lstrip().startswith("#"):
            continue
        f = line.split()
        if len(f) < 4:
            raise ValueError("낙뢰 응답 필드 확인")
        rows.append(
            {
                "시각": f[0],
                "경도": float(f[1]),
                "위도": float(f[2]),
                "전류(kA)": float(f[3]),
            }
        )
    return pd.DataFrame(rows, columns=["시각", "경도", "위도", "전류(kA)"])


def bin_strikes(d, agg=None, res=0.1):
    """0.1도 격자별로 음극·양극 낙뢰 발생 횟수를 집계합니다."""

    agg = {} if agg is None else agg
    for _, r in d.iterrows():
        key = (int(np.floor(r["경도"] / res)), int(np.floor(r["위도"] / res)))
        a = agg.setdefault(key, [0, 0])
        a[0 if r["전류(kA)"] < 0 else 1] += 1
    return agg
```

❷ 격자별 낙뢰 발생 횟수 분포

0.1도 격자별 횟수를 로그 눈금으로 표시하면 수 회와 수백 회 발생한 지역을 함께 구분합니다.

```
def grid_frame(sec, lon0=None, res=0.1):
  if lon0 is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: lon0")
  "줄 하나가 위도 한 띠. 경도 순서로 [음극 3자, 양극 3자] 36진수로 적어 두었다."
  rows = []
  for line in sec["GRID"]:
    la, buf = line.split("|")
    lat = int(la) * res
    n = len(buf) // 6
    for i in range(n):
      neg = int(buf[i * 6 : i * 6 + 3], 36)
      pos = int(buf[i * 6 + 3 : i * 6 + 6], 36)
      if neg or pos:
        rows.append(
          {
            "경도": round(lon0 + i * res + res / 2, 3),
            "위도": round(lat + res / 2, 3),
            "음극": neg,
            "양극": pos,
          }
        )
  d = pd.DataFrame(
    rows, columns=None if rows else ["경도", "위도", "음극", "양극"]
  )
  d["합계"] = d["음극"] + d["양극"]
  d["양극비율(%)"] = (d["양극"] / d["합계"] * 100).round(1)
  return d
```

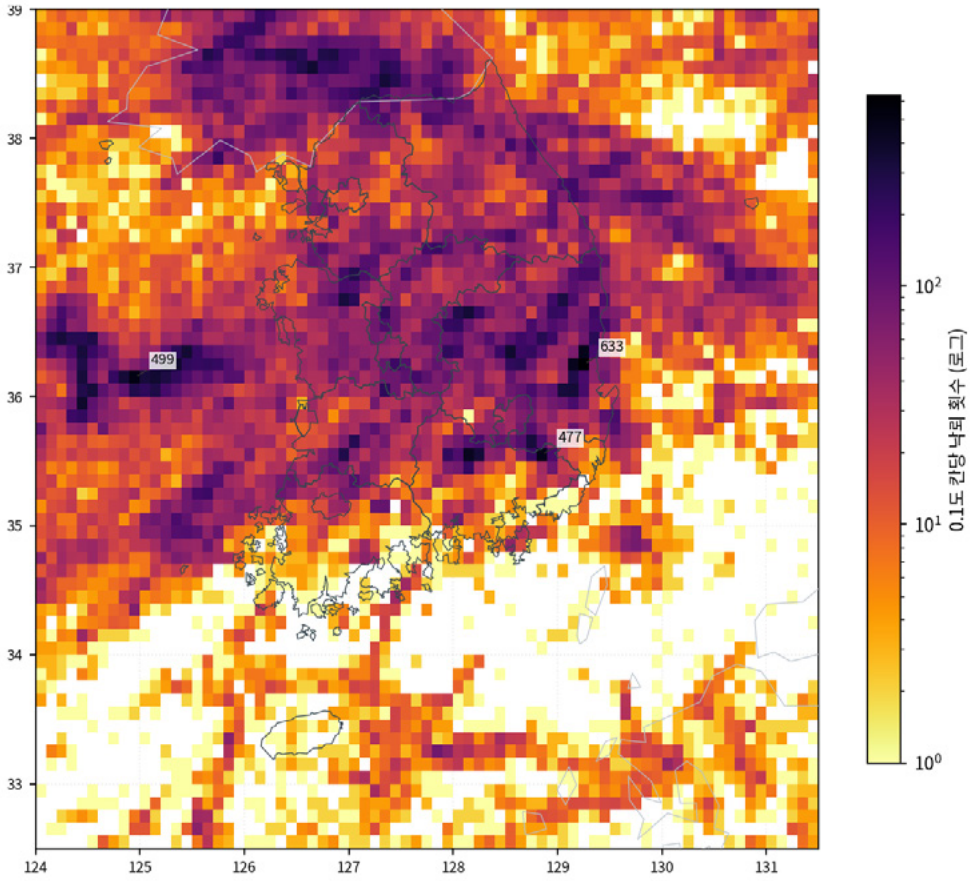


그림 5-1. 격자별 낙뢰 발생 횟수 분포

지상

❸ 시·도별 낙뢰 발생 횟수 비교

격자 중심점이 속한 시·도를 기준으로 낙뢰 횟수를 집계합니다. 면적에 따른 영향을 살피기 위해 총횟수와 격자당 평균을 함께 비교합니다.

```
def province_table(g, *, KO):
    """격자 가운데 점을 시·도에 넣는다. 바다 위 칸은 어디에도 들어가지 않는다."""
    if KO is None:
        return pd.DataFrame()
    d = g.copy()
    d["시도"] = KO.assign_province(d["경도"].values, d["위도"].values)
    d = d.dropna(subset=["시도"])
    t = (
        d.groupby("시도")
        .agg(낙뢰=("합계", "sum"), 칸=("합계", "size"), 양극=("양극", "sum"))
        .reset_index()
    )
    t["칸당"] = (t["낙뢰"] / t["칸"]).round(1)
    t["양극비율(%)"] = (t["양극"] / t["낙뢰"] * 100).round(1)
    return t.sort_values("낙뢰", ascending=False)
```

시도	낙뢰	칸	칸당	양극비율(%)
경상북도	18643	189	98.6	21.5
강원도	8692	170	51.1	25.9
충청북도	5020	76	66.1	30.6
울산광역시	105	10	10.5	20
부산광역시	35	5	7	11.4
제주특별자치도	21	12	1.8	19

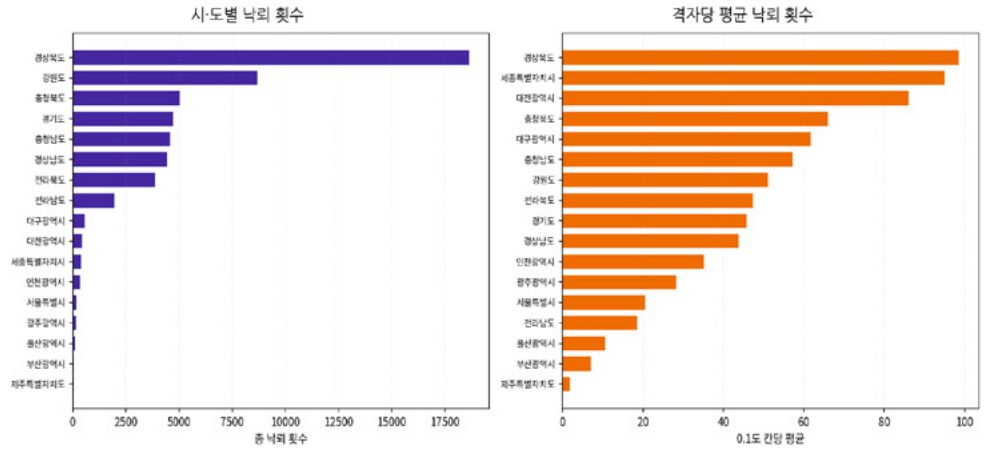


그림 5-2. 시·도별 낙뢰 발생 횟수 비교

❹ 날짜·시간대별 낙뢰 발생 횟수

날짜·시간대별로 낙뢰가 집중된 시기를 집계해 야외 작업·행사 일정 검토에 참고합니다.

```
def day_frame(sec):
  rows = [t.split(",") for t in sec["DAY"][0].split()]
  d = pd.DataFrame(rows, columns=["날짜", "횟수"])
  d["날짜"] = pd.to_datetime(d["날짜"], format="%Y%m%d")
  d["횟수"] = d["횟수"].astype(int)
  return d.sort_values("날짜").reset_index(drop=True)
def hour_frame(sec):
  rows = [t.split(",") for t in sec["HOUR"][0].split()]
  d = pd.DataFrame(rows, columns=["시각", "횟수"]).astype(int)
  return d.sort_values("시각").reset_index(drop=True)
def concentration(day):
  """며칠이 전체의 얼마를 만들었는지. 방재 계획은 평균이 아니라 이 쏠림을 본다."""
  s = day.sort_values("횟수", ascending=False)
  tot = s["횟수"].sum()
  out = {}
  for k in (1, 3, 5, 10):
    out[f"상위 {k}일 몫(%)"] = round(s.head(k)["횟수"].sum() / tot * 100, 1)
  out["낙뢰가 있던 날"] = f"{int((day['횟수'] > 0).sum())}일 / {len(day)}일"
  out["하루 최대"] = (
    f"{s.iloc[0]['날짜']:%Y-%m-%d} {int(s.iloc[0]['횟수']):,}회"
  )
  return out
```

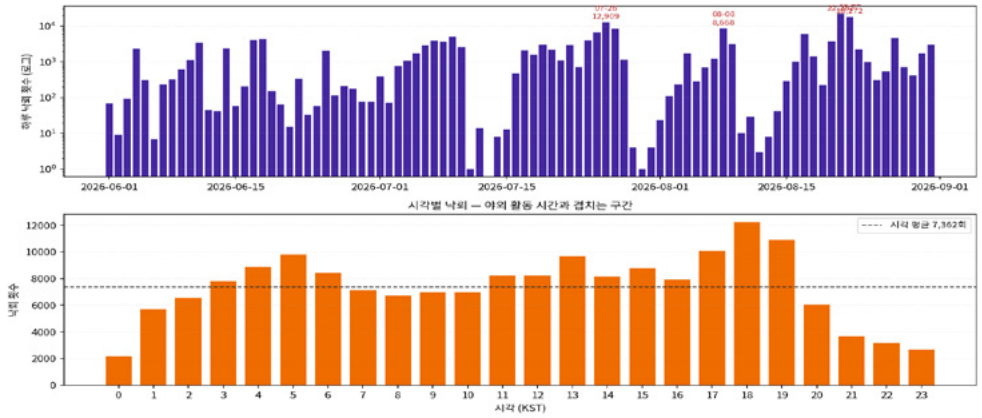


그림 5-3. 날짜·시간대별 낙뢰 발생 횟수

지상

❺ 낙뢰 극성과 전류 세기 비교

전류의 부호는 극성, 절댓값은 세기를 나타냅니다. 큰 전류값까지 분포하므로 시설 피해를 살필 때는 발생 횟수와 세기를 함께 확인합니다.

```
def st_frame(sec):
  """극성과 전류 세기를 5 kA 폭으로 묶어 둔 것. N 은 음극, P 는 양극."""
  rows = []
  for t in sec["ST"][0].split():
    k, v = t.split(",")
    rows.append(
      {
        "극성": "음극" if k[0] == "N" else "양극",
        "하한(kA)": int(k[1:]) * 5,
        "횟수": int(v),
      }
    )
  return pd.DataFrame(
    rows, columns=None if rows else ["극성", "하한(kA)", "횟수"]
  ).sort_values(["극성", "하한(kA)"])
```

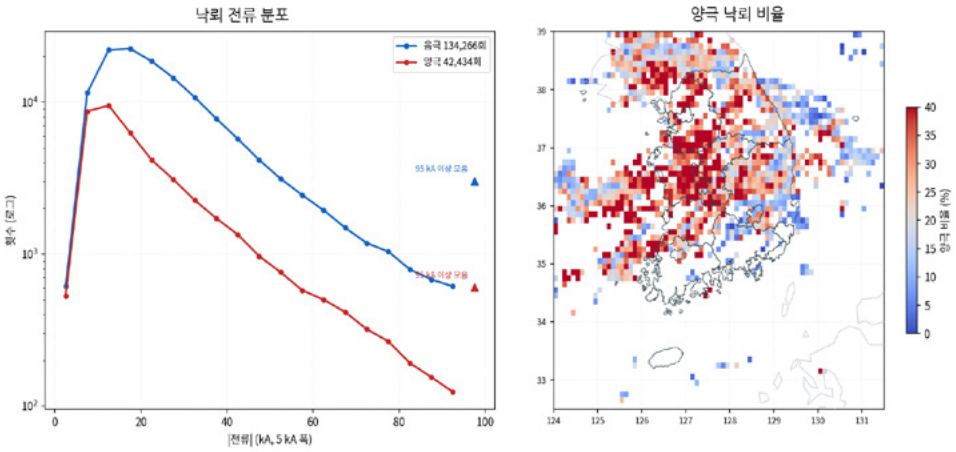


그림 5-4. 낙뢰 극성과 전류 세기 비교

6. 해양기상부이

해역별 파고와 바람 비교

데이터 활용 경로(API)

기상청 API허브 > 해양관측 > 해양기상부이·파고부이관측

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 해상 관측자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)

import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/sea_obs.php'
COLS = ['TP', 'TM', '지점', '지점명', '경도', '위도', '유의파고', '풍향',
    '풍속', '돌풍', '수온', '기온', '해면기압', '습도']
def get_sea(tm):
  res = requests.get(
    api_url,
    params={"tm": tm, "stn": 0, "help": 0, "authKey": API_KEY},
    timeout=30,
  )
  res.encoding = "euc-kr"
  rows = []
  for ln in res.text.splitlines():
    s = ln.strip()
    if not s or s.startswith("#"):
      continue
    c = [x.strip() for x in s.rstrip("=").rstrip(",").split(",")]
    if len(c) >= 14:
      rows.append(c[:14])
  d = pd.DataFrame(rows, columns=COLS)
  for c in COLS[4:]:
    d[c] = pd.to_numeric(d[c], errors="coerce").replace(-99.0, np.nan)
  return d
```

해양

```
d = get_sea('202608150000')
print(d.groupby('TP').size())
TP
B  42
C  61
D   8
F  17
J   1
L   8
N  55
```

TP	관측종류	지점 수	유의파고 있는 지점
B	BUOY(해양기상부이)	42	38
C	파고BUOY	61	59
D	표류BUOY	8	8
F	연안방재	17	0
J	기상1호	1	0
L	등표	8	0
N	조위관측소	55	2

표 6-1. 한 시각에 들어온 관측종류별 지점 수

❸ 관측 지점 배치와 파고 분포

위경도로 관측지점을 표시하고 유의파고를 색으로 구분해 해역별로 비교합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
snap = d[d['TP'] == 'B'].dropna(subset=['유의파고'])
fig = plt.figure(figsize=(7.0, 7.0))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([123.3, 133.2, 30.8, 39.3], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
          linewidth=0.6, edgecolor='#37474f')
sc = ax.scatter(snap['경도'], snap['위도'], c=snap['유의파고'], s=60,
           cmap='YlGnBu', vmin=0, edgecolors='#37474f',
           linewidths=0.5, transform=PC, zorder=5)
for _, r in snap.nlargest(10, '유의파고').iterrows():  # 겹치지 않게 10곳만
   ax.annotate(r['지점명'], (r['경도'], r['위도']), transform=PC,
           textcoords='offset points', xytext=(6, 4), fontsize=8)
fig.colorbar(sc, ax=ax, label='유의파고 (m)')
plt.show()
```

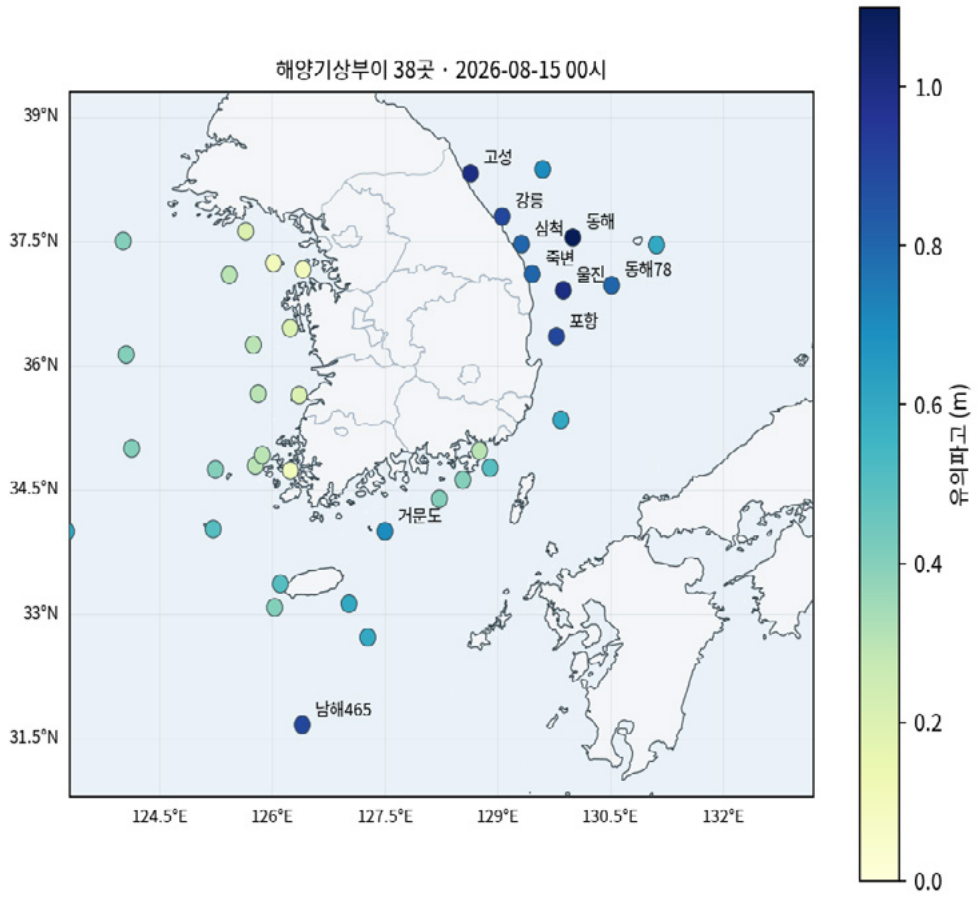


그림 6-1. 해양기상부이 지점 배치와 유의파고

지점번호	지점명	경도	위도	유의파고(m)
22105	동해	130	37.544	1.1
22190	울진	129.874	36.907	1
22310	고성	128.639	38.317	1
22301	남해465	126.4	31.667	0.9
22106	포항	129.783	36.35	0.9
22520	강릉	129.061	37.798	0.9
22302	동해78	130.517	36.967	0.8
22311	삼척	129.318	37.464	0.8
22523	죽변	129.459	37.103	0.8
22305	동해57	129.6	38.367	0.7

표 6-2. 이 시각 유의파고가 큰 10곳

❹ 하루 동안의 유의파고

시각별 자료를 모으고 지점별 시계열과 분포를 작성합니다.

```
frames = []
for h in range(24):
  tm = f'20260815{h:02d}00'
  frames.append(get_sea(tm))        # 요청 사이에 간격을 둔다
```

해양

```
g = pd.concat(frames)
g = g[g['TP'] == 'B']
g['시각'] = pd.to_datetime(g['TM'], format='%Y%m%d%H%M')
piv = g.pivot_table(index='시각', columns='지점명', values='유의파고')
piv = piv[piv.mean().sort_values(ascending=False).index]
print(piv.iloc[:, :4].head(3).round(2))
지점명          남해244  남해465  서귀포  동해
시각
2026-08-15 00:00    1.00  0.96  0.91  0.82
```

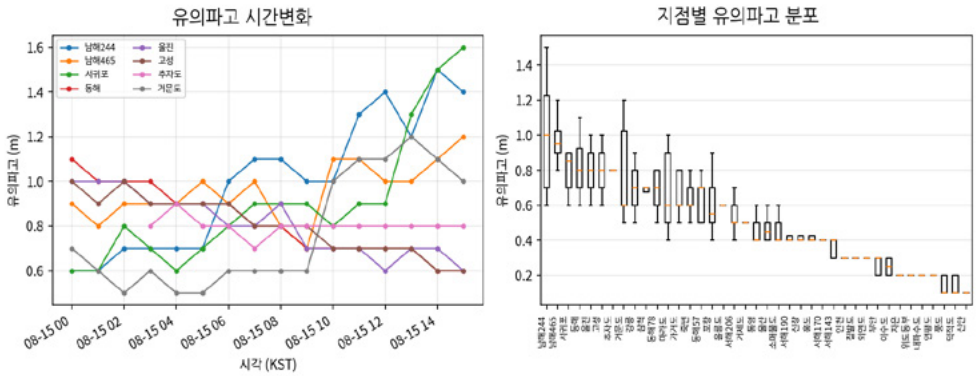


그림 6-2. 해양기상부이 유의파고의 하루 흐름과 지점별 분포

지점명	관측 시각 수	평균(m)	최대(m)	최소(m)
남해244	16	1	1.5	0.6
남해465	16	0.96	1.2	0.7
서귀포	16	0.91	1.6	0.6
삼척	16	0.69	0.8	0.6
동해78	16	0.68	0.8	0.5
마라도	16	0.68	1	0.4

표 6-3. 평균 파고가 큰 12곳 (하루)

❺ 파고와 바람

```
w = g.dropna(subset=['유의파고', '풍속'])
print(w[['유의파고', '풍속', '돌풍', '수온', '기온', '해면기압']]
    .corr()['유의파고'].round(3))
풍속    0.554
돌풍    0.579
수온    0.161
기온    0.064
해면기압   -0.119
습도    0.330
```

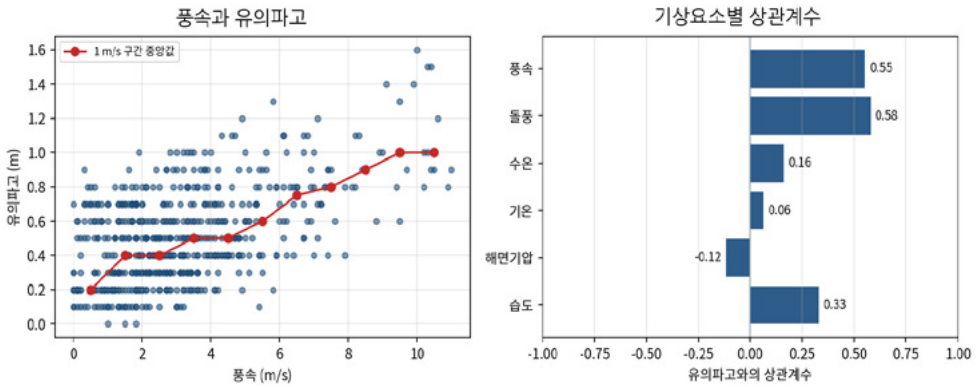


그림 6-3. 풍속과 유의파고, 요소별 상관

풍속 구간(m/s)	유의파고 중앙값(m)
0~1	0.2
1~2	0.4
2~3	0.4
3~4	0.5
4~5	0.5
5~6	0.6
6~7	0.75
7~8	0.8
8~9	0.9
9~10	1
10~11	1

표 6-4. 풍속 구간별 유의파고 중앙값

❻ 요소별 유효자료 비율

요소별 유효값 비율로 관측종류에 따른 자료 제공 범위를 확인합니다.

```
ELEM = ['유의파고', '풍향', '풍속', '돌풍', '수온', '기온', '해면기압', '습도']
print((g[ELEM].notna().sum() / len(g) * 100).round(1))
```

요소	유효자료 수	유효자료 비율(%)
유의파고	621	90.7
풍향	621	90.7
풍속	621	90.7
돌풍	621	90.7
수온	637	93
기온	637	93
해면기압	637	93
습도	573	83.6

표 6-5. 요소별 유효자료 비율

해양

7. 파고부이

지점별 파고 변화 비교

데이터 활용 경로(API)

기상청 API허브 > 해양관측 > 해양기상부이·파고부이관측

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 해상 관측자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/sea_obs.php'
COLS = ['TP', 'TM', '지점', '지점명', '경도', '위도', '유의파고', '풍향',
        '풍속', '돌풍', '수온', '기온', '해면기압', '습도']
def get_sea(tm):
    res = requests.get(
        api_url,
        params={"tm": tm, "stn": 0, "help": 0, "authKey": API_KEY},
        timeout=30,
    )
    res.encoding = "euc-kr"
    rows = []
    for ln in res.text.splitlines():
        s = ln.strip()
        if not s or s.startswith("#"):
            continue
        c = [x.strip() for x in s.rstrip("=").rstrip(",").split(",")]
        if len(c) >= 14:
            rows.append(c[:14])
    d = pd.DataFrame(rows, columns=COLS)
    for c in COLS[4:]:
        d[c] = pd.to_numeric(d[c], errors="coerce").replace(-99.0, np.nan)
    return d
d = get_sea('202608150000')
print(d.groupby('TP').size())
```

```
TP
B   42
C   61
D    8
F   17
J    1
L    8
N   55
```

TP	관측종류	지점 수	유의파고 있는 지점
B	BUOY(해양기상부이)	42	38
C	파고BUOY	61	59
D	표류BUOY	8	8
F	연안방재	17	0
J	기상1호	1	0
L	등표	8	0
N	조위관측소	55	2

표 7-1. 한 시각에 들어온 관측종류별 지점 수

❸ 관측 지점 배치와 파고 분포

위경도로 관측지점을 표시하고 유의파고를 색으로 구분해 해역별로 비교합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
snap = d[d['TP'] == 'C'].dropna(subset=['유의파고'])
fig = plt.figure(figsize=(7.0, 7.0))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([123.3, 133.2, 30.8, 39.3], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
          linewidth=0.6, edgecolor='#37474f')
sc = ax.scatter(snap['경도'], snap['위도'], c=snap['유의파고'], s=60,
           cmap='YlGnBu', vmin=0, edgecolors='#37474f',
           linewidths=0.5, transform=PC, zorder=5)
for _, r in snap.nlargest(10, '유의파고').iterrows():  # 겹치지 않게 10곳만
   ax.annotate(r['지점명'], (r['경도'], r['위도']), transform=PC,
          textcoords='offset points', xytext=(6, 4), fontsize=8)
fig.colorbar(sc, ax=ax, label='유의파고 (m)')
plt.show()
```

해양

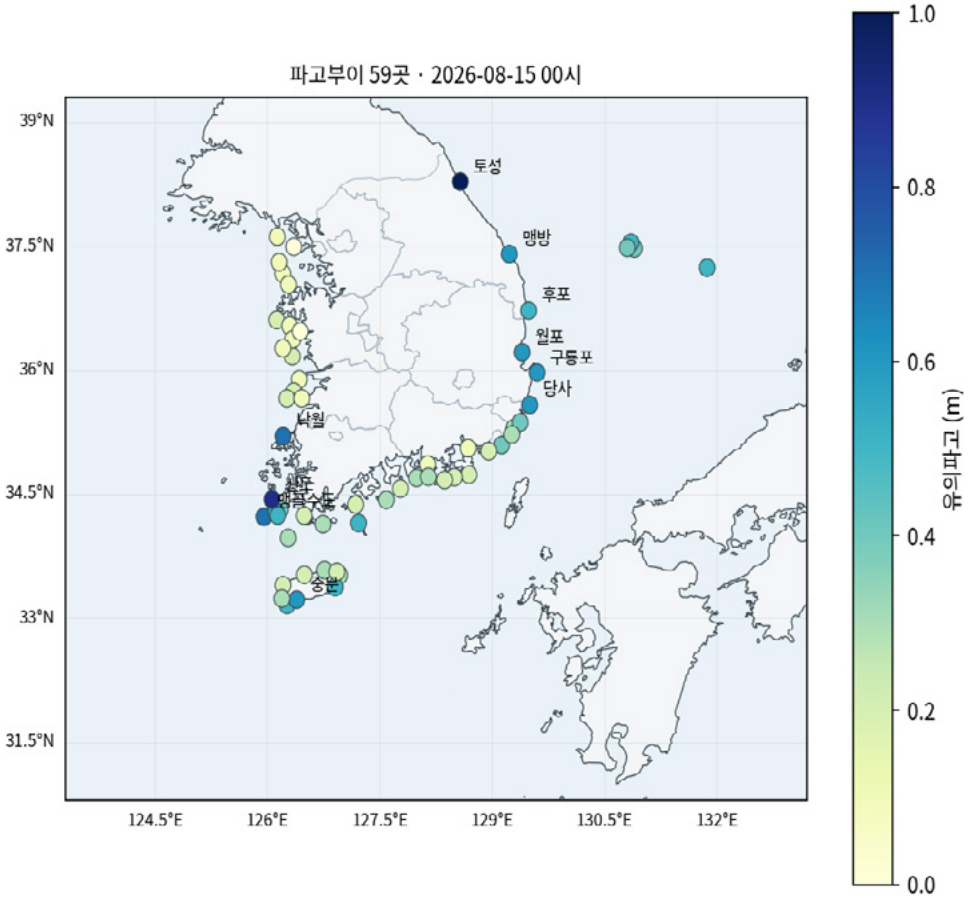


그림 7-1. 파고부이 지점 배치와 유의파고

지점번호	지점명	경도	위도	유의파고(m)
22471	토성	128.576	38.277	1
22449	진도	126.062	34.436	0.9
22481	맹골수도	125.955	34.227	0.7
22494	낙월	126.21	35.2	0.7
22479	맹방	129.229	37.401	0.6
22458	중문	126.393	33.225	0.6
22518	당사	129.503	35.578	0.6
22524	구룡포	129.6	35.969	0.6
22490	월포	129.401	36.216	0.6
22465	후포	129.488	36.719	0.5

표 7-2. 이 시각 유의파고가 큰 10곳

❹ 하루 동안의 유의파고

시각별 자료를 모으고 지점별 시계열과 분포를 작성합니다.

```
frames = []
for h in range(24):
    tm = f'20260815{h:02d}00'
    frames.append(get_sea(tm))        # 요청 사이에 간격을 둔다
g = pd.concat(frames)
```

```
g = g[g['TP'] == 'C']
g['시각'] = pd.to_datetime(g['TM'], format='%Y%m%d%H%M')
piv = g.pivot_table(index='시각', columns='지점명', values='유의파고')
piv = piv[piv.mean().sort_values(ascending=False).index]
print(piv.iloc[:, :4].head(3).round(2))
지점명          신산  중문  가파도  토성
시각
2026-08-15 00:00   0.80  0.72  0.69  0.65
```

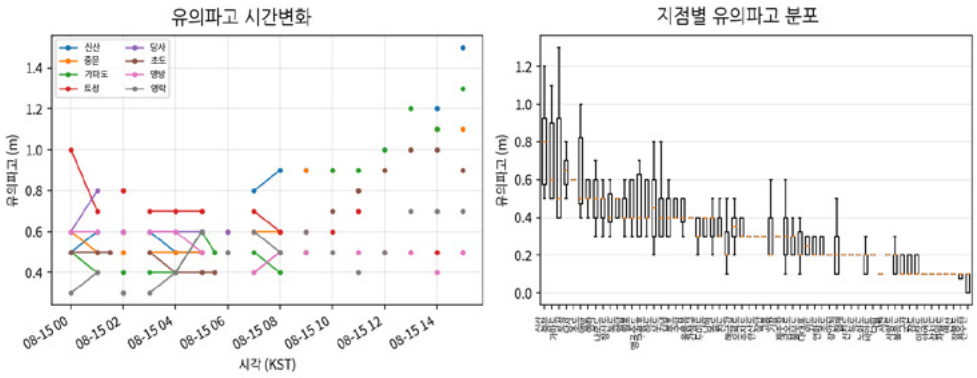


그림 7-2. 파고부이 유의파고의 하루 흐름과 지점별 분포

지점명	관측 시각 수	평균(m)	최대(m)	최소(m)
신산	16	0.8	1.5	0.5
중문	16	0.72	1.1	0.5
가파도	16	0.69	1.3	0.4
청산도	16	0.48	0.8	0.3
독도	16	0.47	0.5	0.4
혈암	16	0.46	0.7	0.3

표 7-3. 평균 파고가 큰 12곳 (하루)

❺ 해역별 파고와 수온

좌표로 서해·남해·동해를 갈라 비교합니다.

```
SEAS = [('서해', 124.0, 126.6, 33.5, 38.5),
    ('남해', 125.8, 129.6, 32.0, 34.9),
    ('동해', 128.4, 132.5, 35.0, 39.0)]
def sea_of(lon, lat):
  for nm, lo1, lo2, la1, la2 in SEAS:
    if lo1 <= lon < lo2 and la1 <= lat < la2:
      return nm
  return "기타"
g['해역'] = [sea_of(a, b) for a, b in zip(g['경도'], g['위도'])]
print(g.groupby('해역')[['유의파고', '수온']].median().round(2))
해역  유의파고  수온
서해    0.20  25.1
남해    0.40  26.8
동해    0.40  26.5
```

해양

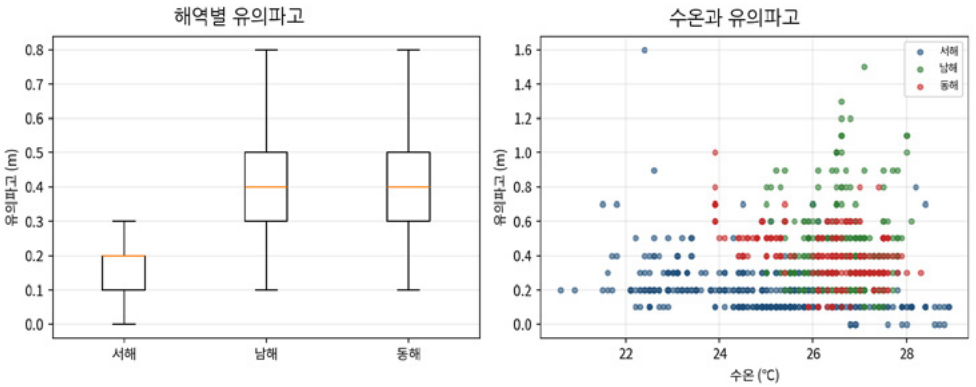


그림 7-3. 해역별 유의파고와 수온·파고 관계

해역	지점 수	파고 중앙값(m)	파고 최대(m)	수온 중앙값(℃)
서해	24	0.2	1.6	25.1
남해	19	0.4	1.5	26.8
동해	16	0.4	1	26.5

표 7-4. 해역별 파고와 수온

❻ 요소별 유효자료 비율

요소별 유효값 비율로 관측종류에 따른 자료 제공 범위를 확인합니다.

```
ELEM = ['유의파고', '풍향', '풍속', '돌풍', '수온', '기온', '해면기압', '습도']
print((g[ELEM].notna().sum() / len(g) * 100).round(1))
```

요소	유효자료 수	유효자료 비율(%)
유의파고	933	95.8
풍향	0	0
풍속	0	0
돌풍	0	0
수온	933	95.8
기온	0	0
해면기압	0	0
습도	0	0

표 7-5. 요소별 유효자료 비율

8-①. 레윈존데

고도별 기온·습도·바람 분석

데이터 활용 경로(API)

기상청 API허브 > 고층관측 > 레윈존데

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib  # 터미널에서 실행
```

❷ 연직 관측자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/upp_temp.php'
COLS = ['TM', 'STN', 'PA', 'GH', 'TA', 'TD', 'WD', 'WS', 'FLAG']
def get_upp(tm):
  res = requests.get(
    api_url,
    params={"tm": tm, "stn": 0, "pa": 0, "help": 1, "authKey": API_KEY},
    timeout=60,
  )
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.strip().startswith("#"))
  ]
  d = pd.DataFrame([r[:9] for r in rows if len(r) >= 9], columns=COLS)
  for c in ["STN", "PA", "GH", "TA", "TD", "WD", "WS"]:
    d[c] = pd.to_numeric(d[c], errors="coerce")
    d[c] = d[c].mask(d[c] <= -999)
  d["시각"] = pd.to_datetime(d["TM"], format="%Y%m%d%H%M", utc=True)
  return d
t = pd.concat([get_upp(tm) for tm in ['202509190000', '202509191200',
                    '202509200000', '202509201200']])
print(len(t), t['STN'].nunique(), t.groupby(['시각', 'STN']).size().describe()[['min', 'max']])
3088 8
min   38.0
max   144.0
```

고층

지점번호	가까운 지상 지점	경도	위도	층 수(4회 합)	최고 고도(km)
47102	백령도	124.712	37.974	198	31.0
47104	북강릉	128.855	37.805	399	31.2
47138	포항	129.380	36.033	450	31.2
47155	창원	128.573	35.170	410	31.2
47169	흑산도	125.451	34.687	477	31.2
47186	서귀포	126.678	33.331	331	31.2
47230	인천	126.103	37.264	408	31.2
47269	영광군	126.031	35.347	415	31.2

표 8-1-1. 이 창에 자료가 들어온 고층관측 지점

❸ 표준등압면과 특이점

1000·925·850·700·500 hPa 처럼 정해진 표준등압면과, 기온이나 바람이 꺾이는 곳을 따로 적은 특이점입니다.

```
STD = [1000, 925, 850, 700, 500, 400, 300, 250, 200, 150,
    100, 70, 50, 30, 20, 10]
t['표준면'] = t['PA'].isin(STD)
print(t.groupby('표준면')[['GH', 'WS']].apply(lambda g: g.notna().mean() * 100).round(1))
       GH   WS
표준면
False   0.0 57.3
True  100.0 98.3
```

구분	층 수	비율(%)	지오포텐셜고도 채움(%)	바람 채움(%)
표준등압면	465	15.1	100.0	98.3
특이점	2623	84.9	0.0	57.3

표 8-1-2. 표준등압면과 특이점의 층 수와 유효자료 비율

❹ 기온·습도·바람의 연직분포

기압을 역방향 로그축으로 두어 연직분포를 그립니다. 주눈금을 지정하고 잔눈금은 숨깁니다.

```
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
p = t[(t['STN'] == 47104) & (t['시각'] == '2025-09-20')].dropna(subset=['TA'])
p = p.sort_values('PA', ascending=False)
fig, axes = plt.subplots(1, 3, figsize=(11.4, 5.4), sharey=True)
axes[0].plot(p['TA'], p['PA'], color='#C62828', label='기온')
axes[0].plot(p.dropna(subset=['TD'])['TD'],
        p.dropna(subset=['TD'])['PA'], color='#1F4E79', label='이슬점')
axes[1].plot(p.dropna(subset=['WS'])['WS'],
        p.dropna(subset=['WS'])['PA'], color='#2E7D32')
axes[2].plot(p.dropna(subset=['GH'])['GH'] / 1000,
        p.dropna(subset=['GH'])['PA'], color='#5E35B1')
for ax in axes:
  ax.set_yscale('log')
  ax.set_ylim(1020, 10)
  ax.set_yticks([1000, 850, 700, 500, 300, 200, 100, 50, 20, 10])
  ax.yaxis.set_major_formatter(mticker.ScalarFormatter())
  ax.yaxis.set_minor_locator(mticker.NullLocator())  # 로그축 잔눈금 끄기
  ax.grid(alpha=0.3)
plt.show()
```

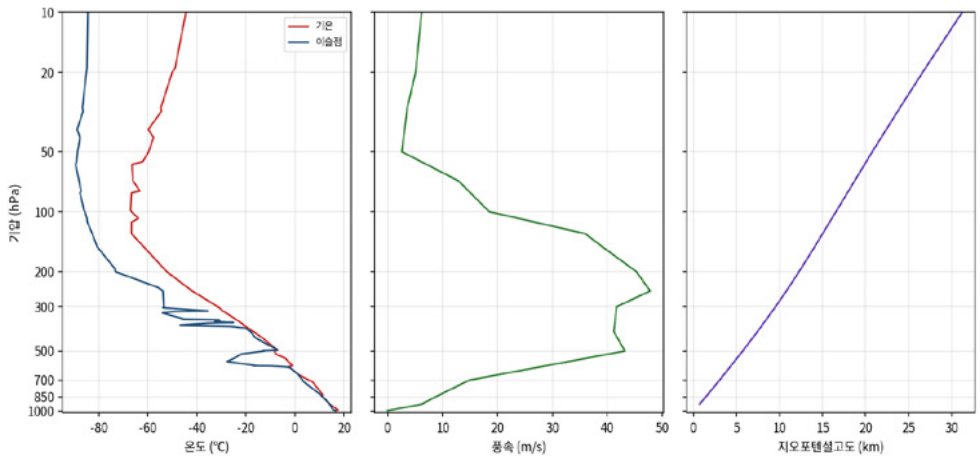


그림 8-1-1. 47104(북강릉) 2025-09-20 00 UTC 연직 프로파일 61층

❺ 상층 최저기온층 비교

성층권 위쪽을 잘라 내고 기온이 가장 낮은 층을 고르면 각 관측의 권계면을 자료 안에서 찾습니다.

```
def tropopause(g):
  g = g.dropna(subset=["TA", "PA"])
  g = g[g["PA"] >= 50]
  return g.loc[g["TA"].idxmin()] if len(g) >= 5 else None
rows = [dict(시각=tm, STN=s, 기압=r['PA'], 고도=r['GH'], 기온=r['TA'])
    for (tm, s), g in t.groupby(['시각', 'STN'])
    if (r := tropopause(g)) is not None]
tp = pd.DataFrame(rows)
print(tp[['기압', '기온']].median().round(1))
기압   99.0
기온   -72.2
```

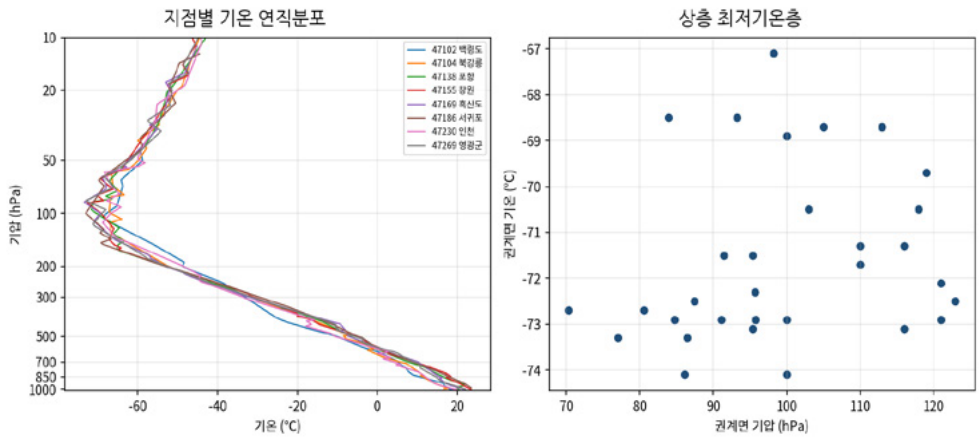


그림 8-1-2. 같은 시각 여덟 지점의 기온 프로파일과 권계면 분포

고층

지점번호	지점	권계면 기압 중앙값(hPa)	권계면 고도 중앙값(km)	권계면 기온 중앙값(℃)
47102	백령도	100	16.5	-70.9
47104	북강릉	96	-	-68.5
47138	포항	103	-	-71.6
47155	창원	92	-	-72.5
47169	흑산도	91	-	-73.0
47186	서귀포	93	16.7	-73.1
47230	인천	108	16.6	-69.7
47269	영광군	101	-	-72.5

표 8-1-3. 지점별 권계면

❻ 안정도 지수와 하늘 상태

한 줄에 CAPE·CIN·K지수·SSI·TPW 등 30여 개 지수가 있고, 끝에 그 시각의 구름 부호가 붙어 있어 지수가 실제 하늘과 맞는지 자료 안에서 확인할 수 있습니다.

```
idx_url = 'https://apihub.kma.go.kr/api/typ01/url/upp_idx.php'
res = requests.get(idx_url, params={'tm1': '202506010000', 'tm2': '202509210000',
                   'stn': 0, 'help': 1,
                   'authKey': API_KEY}, timeout=60)
# 끝 칸 CLOUD : SKC 맑음 · FEW · SCT · BKN · OVC 온흐림
print(x.groupby('CLOUD')[['KI', 'TPW', 'CAPE']].median().round(0))
       KI  TPW  CAPE
CLOUD
SKC    10   23    30
SCT    20   35    61
BKN    23   43    62
OVC    31   52    43
```

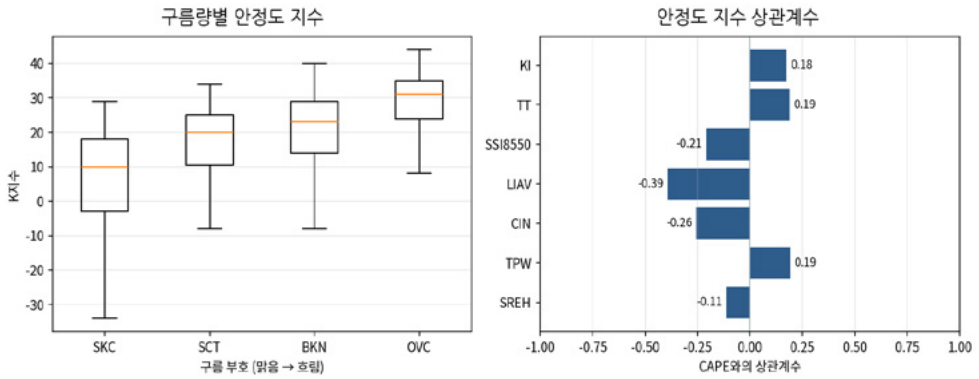


그림 8-1-3. 에너지대별 플럭스 스펙트럼

구름 부호	건수	K지수 중앙값	TPW 중앙값(mm)	CAPE 중앙값(J/kg)
SKC	193	10	23	30
SCT	131	20	35	61
BKN	922	23	43	62
OVC	1429	31	52	43

표 8-1-4. 시간·에너지대별 플럭스 변화

지수가 하늘과 같은 방향을 가리킨다는 것을 확인할 수 있습니다.

지수	CAPE와의 상관
KI	0.18
TT	0.19
SSI8550	-0.21
LIAV	-0.39
CIN	-0.26
TPW	0.19
SREH	-0.11

표 8-1-5. CAPE와 다른 지수의 상관

고층

8-②. 레윈존데

대기 불안정도 비교

데이터 활용 경로(API)

기상청 API허브 > 고층관측 > 레윈존데

❶ 안정도 지수 조회

기간을 지정해 기온·습도·CAPE·K-index·상승지수 등을 조회합니다.

```
https://apihub.kma.go.kr/api/typ01/url/upp_idx.php?tm1=2026060100&tm2=2026090300&stn=0&
authKey=...
# YYMMDDHHMI STN   TA   HM ··· KI ··· CAPE  CIN  TPW  TT LIAV ··· CLOUD
202609010000,47102, 22.4, 87.0, ···, 17, ···,  5.50,224.50,39.80, 43, -0.20, ···, OVC,=
202609010000,47104, 22.2, 97.0, ···, 29, ···,  1.10,  0.10,53.70, 39,  0.90, ···, OVC,=
#  -999 는 그 지수를 낼 수 없었던 자리 · 줄 끝의 ,= 는 줄 끝 표시
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
IDX_URL = "https://apihub.kma.go.kr/api/typ01/url/upp_idx.php"
COLS = ['TM','STN','TA','HM','WD','WS','FL','EQT850','TP','LCL','CCL','LFC','HEL','MW','SSI8550','SSI9250','SSI
9270','KI','TCKN','CVTT','MAXT','MINT','SREH','CAPE','CIN','TPW','TT','LIAV','LISF','LI92','UPRESS','UT','MPRE
SS','MT','LPRESS','LT','CLOUD']
WMO_BASE = 47000
CLOUD_KO = {'CLR':'맑음','FEW':'조금','SCT':'흩어짐','BKN':'많음','OVC':'흐림'}
KO_NAME = {c:c for c in COLS[2:-1]}  # 공식 요소 설명표와 함께 사용
def get_idx(tm1, tm2, stn=0):
    """레윈존데 관측의 안정도 지수를 조회합니다. tm1·tm2로 조회기간을 지정합니다."""

    r = requests.get(
        IDX_URL,
        params={"tm1": tm1, "tm2": tm2, "stn": stn, "authKey": API_KEY},
        timeout=300,
    )
    if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
    r.encoding = "euc-kr"
    return r.text
def parse_idx(text):
    """지점·시각별 안정도 지수와 구름 코드를 읽습니다. 줄 끝의 ,= 표시는 제거합니다."""

    rows = []
    for line in text.splitlines():
        s = line.strip()
        if not s or s.startswith("#"):
            continue
```

```
    p = [x.strip() for x in s.rstrip("=").rstrip(",").split(",")]
    if len(p) < len(COLS):
      continue
    rows.append(p[: len(COLS)])
  return pd.DataFrame(rows, columns=COLS)
def load_names(path):
  """WMO 번호(47000 + 종관지점번호)로 지점 이름과 좌표를 연결합니다."""
  out, sec = {}, None
  for line in Path(path).read_text("utf-8").splitlines():
    if line.startswith("#"):
      continue
    if line.startswith("@"):
      sec = line[1:]
      continue
    if sec in ("SFC", "UPP"):
      p = line.split(",")
      out[WMO_BASE + int(p[0])] = (p[4], float(p[1]), float(p[2]))
  return out
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path, stn_path):
  """받아 둔 석 달치를 읽습니다. -999 는 그 지수를 낼 수 없었던 자리다."""
  d = parse_idx(Path(path).read_text("utf-8"))
  for c in COLS[2:-1]:
    d[c] = pd.to_numeric(d[c], errors="coerce")
  d = d.replace(-999.0, np.nan)
  d["STN"] = d["STN"].astype(int)
  d["시각"] = pd.to_datetime(d["TM"].astype(str), format="%Y%m%d%H%M", utc=True)
  d["날짜"] = d["시각"].dt.date
  d["월"] = d["시각"].dt.month
  names = load_names(stn_path)
  d["지점명"] = d["STN"].map(lambda s: names.get(s, (str(s),))[0])
  d["경도"] = d["STN"].map(lambda s: names.get(s, (0, np.nan))[1])
  d["위도"] = d["STN"].map(lambda s: names.get(s, (0, np.nan, np.nan))[2])
  d["구름"] = d["CLOUD"].map(lambda c: CLOUD_KO.get(c, c))
  return d
```

❷ 한 번의 요청으로 안정도 지수 조회

고도(LCL·CCL·LFC·권계면), 불안정 지수(CAPE·K·SSI·TT·상승지수), 총가강수량(TPW)이 함께 제공됩니다.

```
def index_table(d):
  """지수별 유효자료 수와 값의 분포를 집계합니다. 결측 여부만으로 미산출 원인을 판단하지 않습니다."""
  rows = []
  for c, nm in KO_NAME.items():
    v = d[c]
    rows.append(
      {
        "부호": c,
        "뜻": nm,
        "값 있는 비율(%)": round(100 * float(v.notna().mean()), 1),
        "최소": round(float(v.min()), 1),
        "중앙값": round(float(v.median()), 1),
        "최대": round(float(v.max()), 1),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["부호", "뜻", "값 있는 비율(%)", "최소", "중앙값", "최대"]
    ),
  )
```

고층

▸ 실행 결과

부호	뜻	값 있는 비율(%)	최소	중앙값	최대
TA	기온(℃)	100	13.4	24.4	38.4
HM	습도(%)	100	-99	84	100
FL	빙결고도(gpm)	100	2555	4747	6299
TPW	총가강수량 (mm)	100	9.9	45.2	80.1
TT	TT-index	100	5	41	57
LIAV	상승지수 1000-500	100	-8.9	0.6	18.1

❸ 구름량에 따른 안정도 지수 비교

범주별 지수 분포를 비교해 구름 상태와의 관계를 확인합니다.

```
def cloud_table(d, cols=("KINDEX", "TPW", "LIAV", "SSI8550", "CAPE")):
    """실제 응답의 CLOUD 코드별 중앙값. 코드 의미는 공식 표로 확인한다."""
    order = sorted(d["CLOUD"].dropna().unique())
    g = d[d["CLOUD"].isin(order)]
    t = g.groupby("CLOUD")[list(cols)].median().reindex(order).round(1)
    t.insert(0, "건수", g.groupby("CLOUD").size().reindex(order))
    return t.reset_index()
```

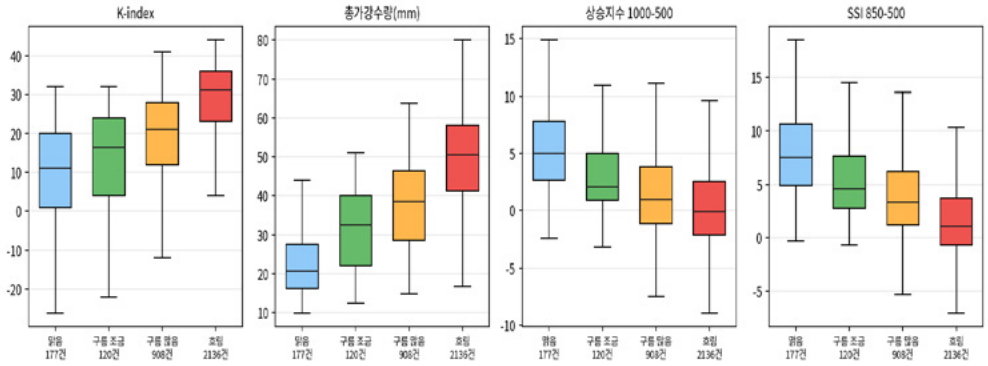


그림 8-2-1. 구름량에 따른 안정도 지수 비교

구름	건수	KINDEX	TPW	LIAV	SSI8550	CAPE
SKC (맑음)	177	11	20.7	5	7.5	27.9
SCT (구름 조금)	120	16.5	32.6	2.1	4.6	50.8
BKN (구름 많음)	908	21	38.5	1	3.3	28.9
OVC (흐림)	2136	31	50.6	-0.1	1.1	25.6

❹ 안정도 지수 간 상관 비교

지수 간 상관을 비교하면 정보가 중복되는 지수와 서로 다른 특성을 반영하는 지수를 구분합니다.

```
def corr_frame(d, cols=None):
  if cols is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: cols")
  return d[list(cols)].corr()
```

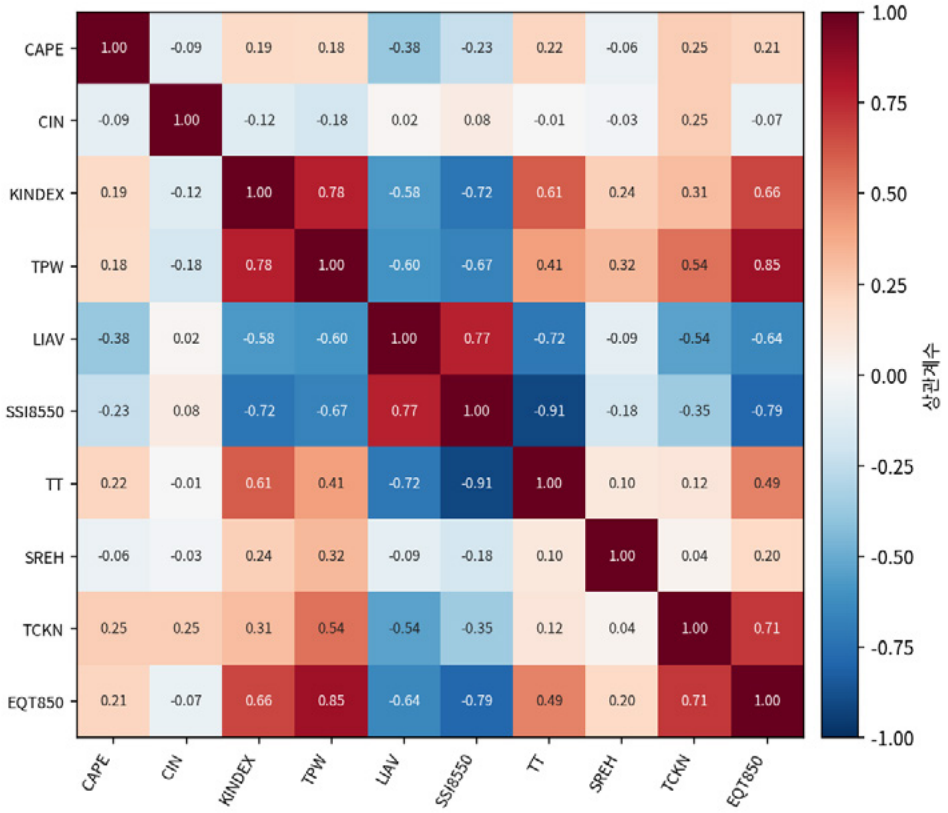


그림 8-2-2. 안정도 지수 간 상관 비교

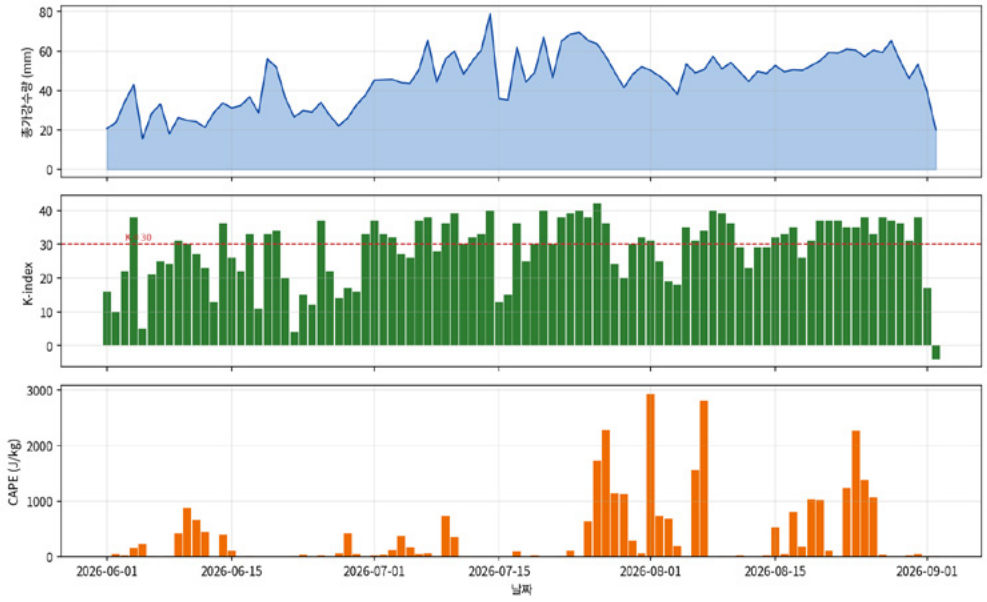
지수	CAPE	CIN	KINDEX	TPW	LIAV	SSI8550	TT	SREH	TCKN	EQT850
CAPE	1	-0.09	0.19	0.18	-0.38	-0.23	0.22	-0.06	0.25	0.21
CIN	-0.09	1	-0.12	-0.18	0.02	0.08	-0.01	-0.03	0.25	-0.07
KINDEX	0.19	-0.12	1	0.78	-0.58	-0.72	0.61	0.24	0.31	0.66
TPW	0.18	-0.18	0.78	1	-0.6	-0.67	0.41	0.32	0.54	0.85
LIAV	-0.38	0.02	-0.58	-0.6	1	0.77	-0.72	-0.09	-0.54	-0.64
SSI8550	-0.23	0.08	-0.72	-0.67	0.77	1	-0.91	-0.18	-0.35	-0.79
TT	0.22	-0.01	0.61	0.41	-0.72	-0.91	1	0.1	0.12	0.49
SREH	-0.06	-0.03	0.24	0.32	-0.09	-0.18	0.1	1	0.04	0.2
TCKN	0.25	0.25	0.31	0.54	-0.54	-0.35	0.12	0.04	1	0.71
EQT850	0.21	-0.07	0.66	0.85	-0.64	-0.79	0.49	0.2	0.71	1

고층

❺ 일최댓값으로 본 여름철 지수 변화

하루 네 번의 관측에서 날짜별 최댓값을 추출해 여름철 변화를 분석합니다.

```
def daily_frame(d, stn=47102):
  g = d[d["STN"] == stn].sort_values("시각")
  t = (
    g.groupby("날짜")
    .agg(
      TPW=("TPW", "max"),
      K=("KINDEX", "max"),
      CAPE=("CAPE", "max"),
      LI=("LIAV", "min"),
      건수=("TPW", "size"),
    )
    .reset_index()
  )
  t["날짜"] = pd.to_datetime(t["날짜"])
  return t
def month_table(d):
  g = (
    d.groupby("월")
    .agg(
      건수=("TPW", "size"),
      TPW중앙=("TPW", "median"),
      K중앙=("KINDEX", "median"),
      CAPE중앙=("CAPE", "median"),
      LI중앙=("LIAV", "median"),
      CAPE10000이상=("CAPE", lambda v: int((v >= 1000).sum())),
    )
    .round(1)
  )
  return g.reset_index()
```



월	건수	TPW중앙	K중앙	CAPE중앙	LI중앙	CAPE 10000이상
6	1070	29.2	17	16	4	7
7	1115	51	30	28	-0.9	153
8	1104	50.3	29	46.8	-0.6	209
9	62	47.7	28	40.4	-1.6	17

그림 8-2-3. 일최댓값으로 본 여름철 지수 변화

❻ CAPE와 CIN의 결합 분석

CAPE는 대류에 사용될 수 있는 에너지, CIN은 대류를 억제하는 에너지입니다. 두 값을 함께 로그 축에 표시하고 네 구간으로 구분해 대류 발생 조건을 비교합니다.

```
def quad_table(d, cape_thr=1000, cin_thr=50):
  """CAPE와 CIN의 분석 기준값에 따라 자료를 네 구간으로 분류합니다. CAPE는 대류가용잠재에너지, CIN은 대류억제에너지입니다."""

  g = d.dropna(subset=["CAPE", "CIN"])
  rows = []
  for cl, cname in [
    (True, f"CAPE ≥ {cape_thr}"),
    (False, f"CAPE < {cape_thr}"),
  ]:
    for il, iname in [
      (True, f"CIN ≥ {cin_thr} (뚜껑 있음)"),
      (False, f"CIN < {cin_thr} (뚜껑 얇음)"),
    ]:
      m = ((g["CAPE"] >= cape_thr) == cl) & ((g["CIN"] >= cin_thr) == il)
      rows.append(
        {
          "CAPE": cname,
          "CIN": iname,
          "건수": int(m.sum()),
          "비율(%)": round(100 * float(m.mean()), 1),
          "K 중앙값": (
            round(float(g[m]["KINDEX"].median()), 1)
            if m.any()
            else np.nan
          ),
          "흐림 비율(%)": (
            round(100 * float((g[m]["CLOUD"] == "OVC").mean()), 1)
            if m.any()
            else np.nan
          ),
        }
      )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["CAPE", "CIN", "건수", "비율(%)", "K 중앙값", "흐림 비율(%)"]
    ),
  )
```

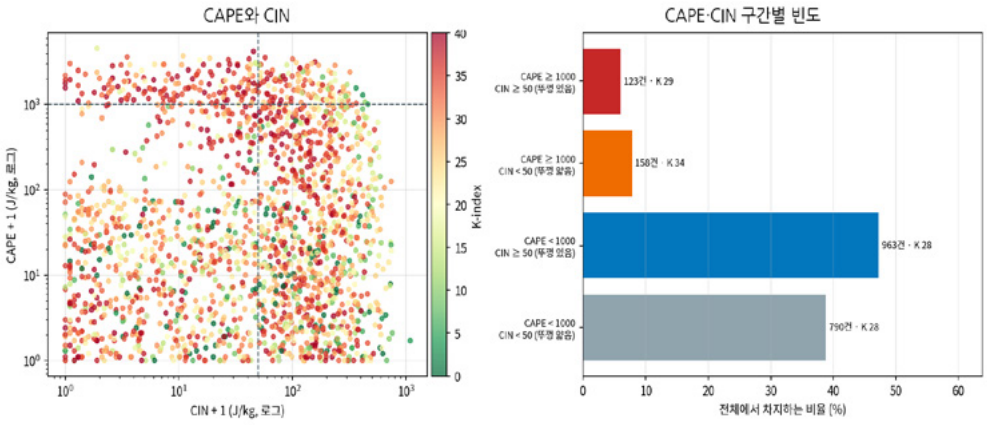


그림 8-2-4. CAPE와 CIN의 결합 분석

고층

CAPE	CIN	건수	비율(%)	K 중앙값	흐림 비율(%)
CAPE ≥ 1000	CIN ≥ 50	123	6	29	41.5
CAPE ≥ 1000	CIN < 50	158	7.8	34	77.8
CAPE < 1000	CIN ≥ 50	963	47.3	28	54.1
CAPE < 1000	CIN < 50	790	38.8	28	79.5

❼ 지점별 불안정 빈도 비교

지점별 불안정 관측 빈도를 지도와 위도별 그래프로 비교해 남북 차이를 살펴봅니다.

```
def stn_frame(d):
    """지점별 안정도 지수를 집계하고 지점정보를 연결합니다. 지도에는 좌표가 확인된 지점만 표시합니다."""

    g = d.copy()
    t = (
        g.groupby(["STN", "지점명"])
        .agg(
            건수=("TPW", "size"),
            TPW중앙=("TPW", "median"),
            K중앙=("KINDEX", "median"),
            CAPE중앙=("CAPE", "median"),
            CAPE최대=("CAPE", "max"),
            불안정비율=("KINDEX", lambda v: 100 * float((v >= 30).mean())),
        )
        .round(1)
        .reset_index()
    )
    pos = g.drop_duplicates("STN").set_index("STN")[["경도", "위도"]]
    t = t.join(pos, on="STN")
    return t.sort_values("위도")
```

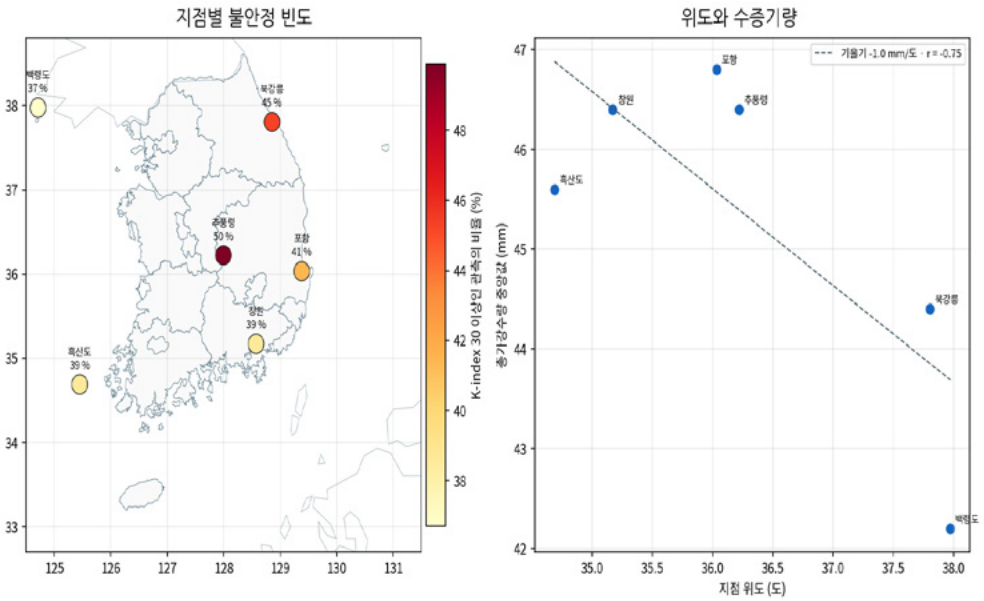


그림 8-2-5. 지점별 불안정 빈도 비교

지점명	STN	경도	위도	건수	TPW 중앙	K중앙	CAPE 중앙	CAPE 최대	불안정 비율
흑산도	47169	125.45	34.69	376	45.6	25	59.7	6283.7	38.6
창원	47155	128.57	35.17	376	46.4	27	30.2	4464.3	38.6
포항	47138	129.38	36.03	372	46.8	27	28.5	2933.3	41.4
추풍령	47135	127.99	36.22	375	46.4	29	20.1	3940.8	49.9
북강릉	47104	128.86	37.8	375	44.4	29	38	4199.5	45.3
백령도	47102	124.71	37.97	376	42.2	25	20.3	2933.3	36.7
47186	47186			375	45.3	27	33.4	3659.5	38.4
47230	47230			360	43.4	25	15.2	2495.4	35.6
47269	47269			366	46.3	26	25.6	3480.5	40.2

고층

9. 연직바람관측

시간·고도별 바람 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 고층관측 > 연직바람관측

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib  # 터미널에서 실행
```

❷ 연직 바람 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/kma_wpf.php'
COLS = ['TM', 'STN', 'HT', 'WD', 'WS', 'U', 'V', 'W', 'QC']
def get_wpf(tm):
  res = requests.get(
    api_url,
    params={"tm": tm, "stn": 0, "help": 1, "authKey": API_KEY},
    timeout=60,
  )
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.strip().startswith("#"))
  ]
  d = pd.DataFrame([r[:9] for r in rows if len(r) >= 9], columns=COLS)
  for c in COLS[1:]:
    d[c] = pd.to_numeric(d[c], errors="coerce")
    d[c] = d[c].mask(d[c] <= -999)
  d["시각"] = pd.to_datetime(d["TM"], format="%Y%m%d%H%M", utc=True)
  return d
snap = get_wpf('202509200000')
print(len(snap), snap['STN'].nunique())
1563 22
```

❸ 지점별 관측 높이 비교

```
g = (snap.groupby('STN')
   .agg(층수=('HT', 'size'), 최저=('HT', 'min'), 최고=('HT', 'max'))
   .reset_index())
g['간격'] = snap.groupby('STN')['HT'].apply(lambda s: pd.Series(np.diff(np.sort(s.dropna().unique())))).
median()).reindex(g['STN']).to_numpy()
print(g.sort_values('최고', ascending=False).head(4))
47884  164   50  8200  50
47269  160   50  8000  50
47152  87    50  5150  59
47135  99   100  5000  50
```

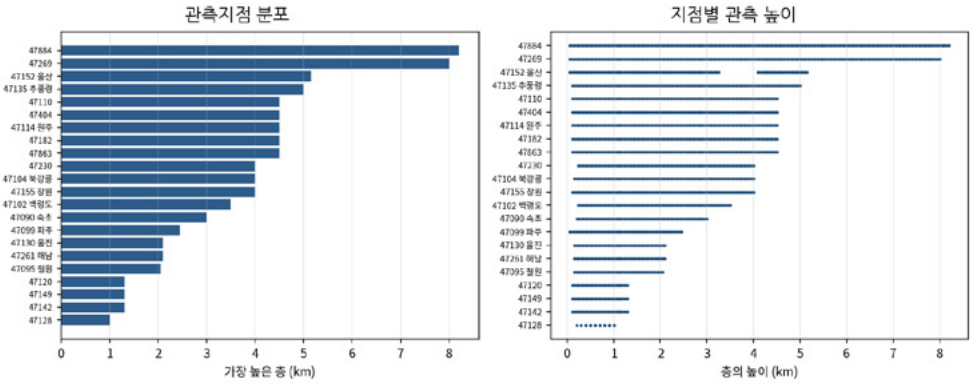


그림 9-1. 22개 지점이 닿는 높이와 층이 놓인 자리

지점번호	대응 지상 지점	층 수	가장 낮은 층(m)	가장 높은 층(m)	층 간격(m)
47884	-	164	50	8200	50
47269	-	160	50	8000	50
47152	울산	87	50	5150	59
47230	-	76	250	4000	50
47104	북강릉	78	150	4000	50
47155	창원	79	100	4000	50

표 9-1. 높이가 높은 12개 지점 (한 시각 기준)

❹ 시간·고도별 바람 분포

시간·고도별 풍속을 색으로 표시하고 U·V 바람깃으로 방향을 함께 나타냅니다.

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
sel = pd.concat([get_wpf(f'20250920{h:02d}00') for h in range(24)])
p = sel[sel['STN'] == 47269].dropna(subset=['WS'])
piv = p.pivot_table(index='HT', columns='시각', values='WS').sort_index()
fig, ax = plt.subplots(figsize=(11.2, 3.2))
X = mdates.date2num(piv.columns.to_pydatetime())
m = ax.pcolormesh(X, piv.index / 1000, np.ma.masked_invalid(piv.values),
          cmap='viridis', shading='nearest', vmin=0, vmax=30)
q = p[p['HT'] % 500 == 0]              # 바람깃은 500 m 마다만
u = q.pivot_table(index='HT', columns='시각', values='U')
v = q.pivot_table(index='HT', columns='시각', values='V').reindex(index=u.index, columns=u.columns)
XX, YY = np.meshgrid(mdates.date2num(u.columns.to_pydatetime()),
```

고층

```
            u.index / 1000)
ax.barbs(XX[:, ::2], YY[:, ::2], u.values[:, ::2], v.values[:, ::2],
      length=4.6, linewidth=0.5, color='#eeeeee')
fig.colorbar(m, ax=ax, label='풍속 (m/s)')
plt.show()
```

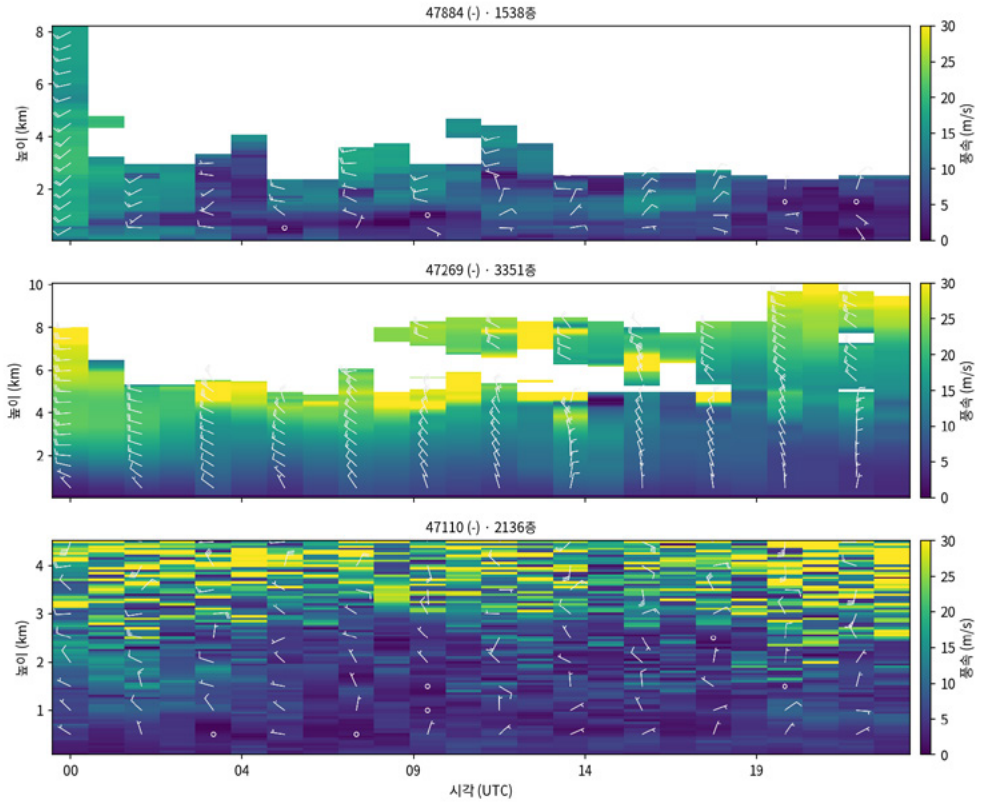


그림 9-2. 2025-09-20 하루의 시간 × 고도 풍속 단면 (세 지점)

❺ 고도별 유효자료 비율과 자료 품질

고도 1 km 구간별 유효값 비율로 관측 가능 범위를 비교합니다.

```
sel['구간'] = (sel['HT'] // 1000) * 1000
fill = 100 * sel.groupby('구간')['WS'].count() / sel.groupby('구간').size()
print(fill.round(1))
   0  100.0
1000  100.0
2000   95.7
3000   88.5
4000   90.4
5000   59.1
6000   75.8
7000   97.2
8000  100.0
9000  100.0
```

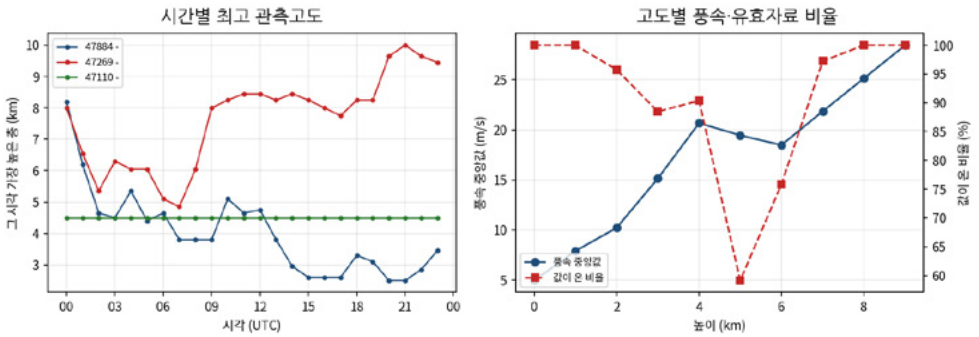


그림 9-3. 하루 동안 닿은 높이와 높이별 풍속·유효자료 비율

높이 구간(km)	층 수	값이 온 비율(%)	풍속 중앙값(m/s)	풍속 최대(m/s)
0~1	1344	100.0	5.0	18.4
1~2	1440	100.0	7.9	21.4
2~3	1387	95.7	10.2	40.2
3~4	1215	88.5	15.1	67.5
4~5	881	90.4	20.7	72.8
5~6	460	59.1	19.4	56.1
6~7	327	75.8	18.5	40.5
7~8	326	97.2	21.9	59.1
8~9	148	100.0	25.1	49.4
9~10	57	100.0	28.4	35.7

표 9-2. 높이 1 km 구간별 층 수와 유효자료 비율

QC	층 수	비율(%)	풍속이 온 비율 (%)	풍속 중앙값(m/s)	풍속 최대 (m/s)	평균 높이 (km)
1	6488	85.5	95.1	10.2	59.1	2.9
3	856	11.3	100.0	15.8	72.8	3.5
4	241	3.2	0.0	-	-	5.8

표 9-3. QC 값별 층 수와 풍속

항공

10. 공항기상관측(AMOS)

시정·운고·바람 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 항공기상 > 공항기상관측(AMOS)

❶ 공항 관측자료 조회

활주로 가시거리(RVR)·최저 운고와 2분·10분 평균풍속·순간 최대·최소값을 비교합니다.

```
https://apihub.kma.go.kr/api/typ01/url/amos.php?tm=202608311000&stn=113&disp=0&authKey=...
# TM L_VIS R_VIS L_RVR R_RVR CH_MIN TA TD HM PS PA RN ... WD10 WS10 ...
 113 202608311000 10000 -99999 2000 -99999 6600 246 244 99 10090 10082 42 ...
# stn 을 빼면 그 시각의 모든 공항이 한 번에 온다
# 기온·이슬점 0.1 ℃ · 기압 0.1 hPa · 풍속 0.1 m/s 단위의 정수
# -99999 는 값 없음
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
AMOS_URL = "https://apihub.kma.go.kr/api/typ01/url/amos.php"
def get_amos(tm, stn=None, retry=3):
    """지정한 시각의 공항 관측자료를 조회합니다. stn을 생략하면 전체 공항을 조회하며, 시각을 바꾸어 시계열을 수집합니다."""

    params = {"tm": tm, "disp": 0, "authKey": API_KEY}
    if stn:
        params["stn"] = stn
    for i in range(retry):
        try:
            r = requests.get(AMOS_URL, params=params, timeout=60)
            if not r.ok:
                raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
            r.encoding = "euc-kr"
            return r.text
        except Exception as exc:
            if i == retry - 1:
                raise
            log(f"{tm} 재시도 {i + 1}: {type(exc).__name__}")
            time.sleep(2 * (i + 1))
def parse_amos(text):
    """27열 실응답과 25열 명세를 구분하고 변환 전에 결측을 처리한다."""
    prefix = [
        "S",
        "TM",
        "L_VIS",
        "R_VIS",
        "L_RVR",
        "R_RVR",
        "CH_MIN",
```

```
    "TA",
    "TD",
    "HM",
    "PS",
    "PA",
    "RN",
  ]
  wind = [
    "WD02",
    "WD02_MAX",
    "WD02_MIN",
    "WS02",
    "WS02_MAX",
    "WS02_MIN",
    "WD10",
    "WD10_MAX",
    "WD10_MIN",
    "WS10",
    "WS10_MAX",
    "WS10_MIN",
  ]
  records = []
  for line in text.splitlines():
    if not line.strip() or line.lstrip().startswith("#"):
      continue
    values = line.split()
    if len(values) == 27:
      values = values[:13] + values[15:]
    elif len(values) != 25:
      raise ValueError("AMOS 응답 행은 25개 또는 27개 열")
    records.append(values)
  cols = prefix + wind
  d = pd.DataFrame(records, columns=cols)
  for c in cols:
    if c != "TM":
      d[c] = pd.to_numeric(d[c], errors="coerce").replace(-99999, np.nan)
  for c in [
    "TA",
    "TD",
    "PS",
    "PA",
    "RN",
    "WS02",
    "WS02_MAX",
    "WS02_MIN",
    "WS10",
    "WS10_MAX",
    "WS10_MIN",
  ]:
    d[c] = d[c] / 10
  d["시각"] = pd.to_datetime(d["TM"], format="%Y%m%d%H%M").dt.tz_localize(
    "Asia/Seoul"
  )
  return d.rename(
    columns={
      "S": "지점",
      "L_VIS": "시정(m)",
      "L_RVR": "RVR(m)",
      "CH_MIN": "운고(m)",
      "TA": "기온(℃)",
      "TD": "이슬점(℃)",
      "HM": "습도(%)",
      "PS": "해면기압(hPa)",
      "PA": "현지기압(hPa)",
      "RN": "강수(mm)",
      "WS02": "WS02(m/s)",
      "WS10": "WS10(m/s)",
      "WS10_MAX": "10분최대(m/s)",
      "WS10_MIN": "10분최소(m/s)",
    }
  )
```

항공

❷ 공항 기상요소의 하루 변화

하루 변화를 확인합니다. 시정·운고, 기온·이슬점, 평균·순간 최대·최소 풍속을 같은 시간축에 표시해 공항의 기상 상태를 비교합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def collect_amos_day(day, station=113, step_minutes=10):
    """KST 하루의 지정 간격 자료를 모아 CSV로 저장한다."""
    if not 1 <= step_minutes <= 60:
        raise ValueError("간격은 1~60분")
    start = datetime.strptime(day, "%Y%m%d").replace(tzinfo=KST)
    frames = []
    for minutes in range(0, 1440, step_minutes):
        tm = (start + timedelta(minutes=minutes)).strftime("%Y%m%d%H%M")
        frame = parse_amos(get_amos(tm, station))
        if not frame.empty:
            frames.append(frame)
        time.sleep(0.2)
    if not frames:
        raise ValueError("조회조건·메타데이터를 명시하세요: day, station")
    d = pd.concat(frames, ignore_index=True)
    d.to_csv("amos_day.csv", index=False, encoding="utf-8-sig")
    return d
# 실행 예: collect_amos_day("20260831", station=113, step_minutes=10)
# 전체 분자료가 필요하면 step_minutes=1로 지정한다.
def read_pack(path):
    """parse_amos 결과를 저장한 CSV를 읽습니다. 임의 압축 형식은 사용하지 않는다."""
    d = pd.read_csv(path, encoding="utf-8-sig")
    required = {"지점", "시각", "시정(m)", "운고(m)", "WS02(m/s)", "WS10(m/s)"}
    if required - set(d.columns):
        raise ValueError(f"AMOS CSV 필수 열: {sorted(required)}")
    d["시각"] = pd.to_datetime(d["시각"], utc=True).dt.tz_convert("Asia/Seoul")
    d["운고(ft)"] = d["운고(m)"] / 0.3048
    d["시정한계"] = d["시정(m)"] >= 10000
    d["돌풍비"] = d["10분최대(m/s)"] / d["WS10(m/s)"].replace(0, np.nan)
    return d
```

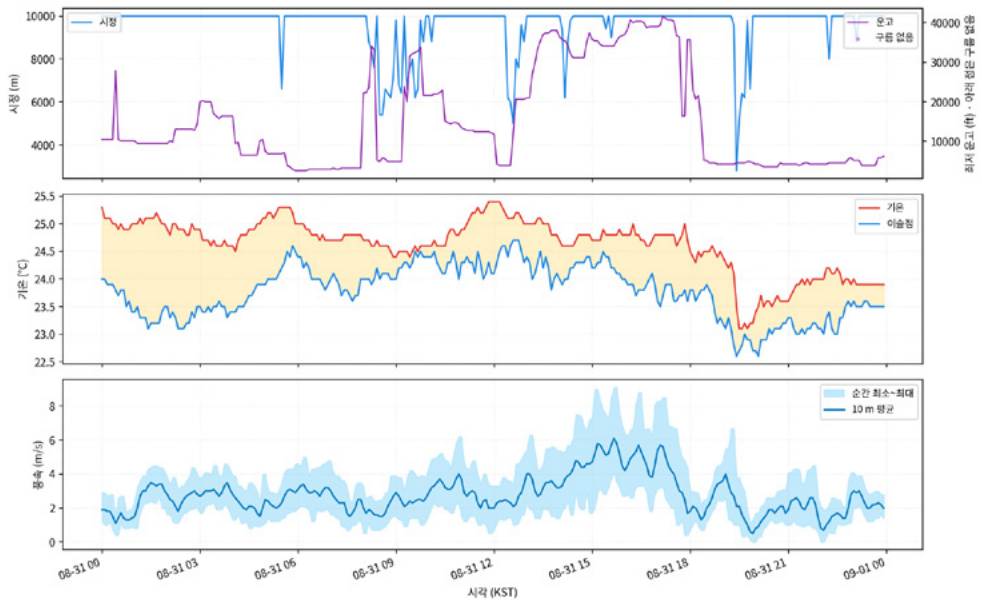


그림 10-1. 공항 기상요소의 하루 변화

❸ 시정·운고에 따른 비행기상 범주

시정(m)과 최저 운고(ft) 중 더 나쁜 조건을 기준으로 VFR·MVFR·IFR·LIFR을 분류하여 기상 조건을 확인합니다.

분류는 미국 AWC의 시정·운고 비교 기준(시정 statute mile, 운고 ft)을 참고한 분석 지표입니다. 결측은 '미분류'이며 실제 운항 판단에는 해당 공항·항공기의 운항 기준을 적용합니다. 출처: https://aviationweather.gov/impactboard/help.html

```
CAT = [("VFR", 5 * 1609.344, 3000, "green"), ("MVFR", 3 * 1609.344, 1000, "blue"), ("IFR", 1609.344, 500,
"red"), ("LIFR", 0, 0, "magenta"), ("미분류", np.nan, np.nan, "gray")]
def category(vis, ceil_ft):
  """시정(m)·운고(ft)의 AWC 비교 범주. 결측은 미분류로 둔다."""
  if pd.isna(vis) or pd.isna(ceil_ft):
    return "미분류"
  miles = float(vis) / 1609.344
  if miles < 1 or ceil_ft < 500:
    return "LIFR"
  if miles < 3 or ceil_ft < 1000:
    return "IFR"
  if miles <= 5 or ceil_ft <= 3000:
    return "MVFR"
  return "VFR"
def cat_frame(d):
  """모든 칸에 범주를 매긴다. 이것이 곧 화면의 색이다."""
  d = d.copy()
  d["범주"] = [category(v, c) for v, c in zip(d["시정(m)"], d["운고(ft)"])]
  return d
def cat_table(d):
  """공항마다 하루의 몇 %가 어느 범주였는지."""
  c = cat_frame(d)
  t = pd.crosstab(c["지점"], c["범주"], normalize="index") * 100
  t = t.reindex(columns=[n for n, *_ in CAT if n in t]).round(1)
  t["칸수"] = c.groupby("지점").size()
  return t.reset_index()
```

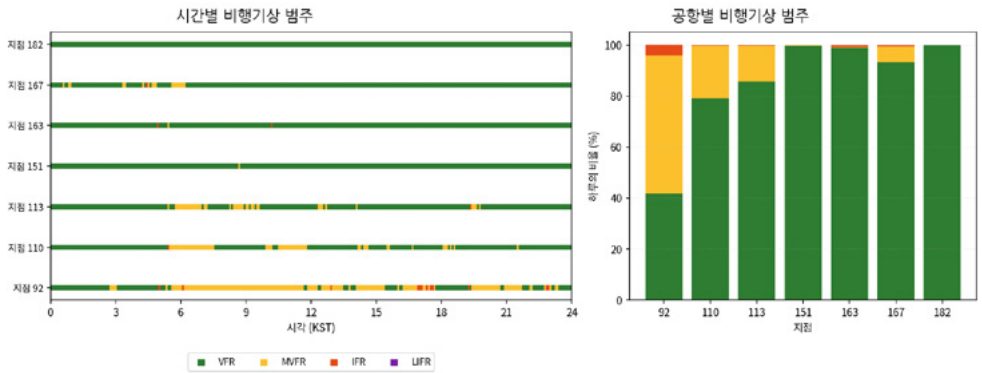


그림 10-2. 시정·운고에 따른 비행기상 범주

지점	VFR	MVFR	IFR	칸수
92	41.7	54.2	4.2	288
110	79.2	20.5	0.3	288
113	85.8	13.9	0.3	288
151	99.7	0.3	0	288
163	99	0.3	0.7	288
167	93.4	6.2	0.3	288
182	100	0	0	288

항공

❹ 10분최대풍속과 평균풍속의 비율

같은 시각의 10분최대풍속을 평균풍속으로 나누어 돌풍비를 계산합니다.

```
def gust_table(d):
    """돌풍은「10분최대 ÷ 10분 평균」으로 잰다. 값이 클수록 바람이 거칠다."""
    t = (
        d.dropna(subset=["돌풍비"])
        .groupby("지점")
        .agg(
            평균풍속=("WS10(m/s)", "mean"),
            최대순간=("10분최대(m/s)", "max"),
            돌풍비중앙=("돌풍비", "median"),
            돌풍비최대=("돌풍비", "max"),
            칸수=("돌풍비", "size"),
        )
        .round(2)
        .reset_index()
    )
    return t.sort_values("최대순간", ascending=False)
```

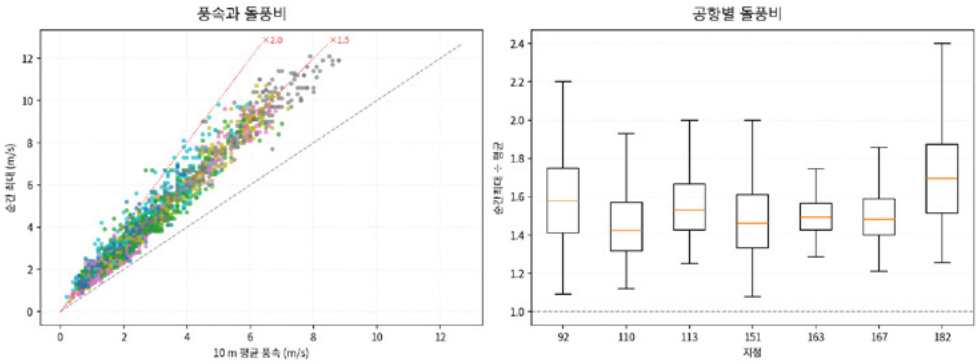


그림 10-3. 10분최대풍속과 평균풍속의 비율

❺ 평균기간별 바람 비교와 기압을 활용한 고도 추정

```
def wind_pair(d):
    """같은 관측점의 2분·10분 평균풍속을 비교한다. 관측높이 비교가 아니다."""
    s = d.dropna(subset=["WS02(m/s)", "WS10(m/s)"])
    s = s[(s["WS02(m/s)"] > 0) & (s["WS10(m/s)"] > 0)]
    if len(s) < 2 or s["WS02(m/s)"].nunique() < 2:
        raise ValueError("회귀에 필요한 서로 다른 풍속 표본 부족")
    a, b = np.polyfit(s["WS02(m/s)"], s["WS10(m/s)"], 1)
    r = np.corrcoef(s["WS02(m/s)"], s["WS10(m/s)"])[0, 1]
    return (s, a, b, r)
def elevation(d):
    """현지기압과 해면기압의 차이에 8.3 m/hPa를 적용한 고도 근사값입니다. 기온·습도·환산 방식에 영향을 받으므로 실제 지점
고도를 대신하지 않습니다."""


    t = (
        d.groupby("지점")
        .agg(해면=("해면기압(hPa)", "median"), 현지=("현지기압(hPa)", "median"))
        .reset_index()
    )
    t["기압차(hPa)"] = (t["해면"] - t["현지"]).round(2)
    t["추정 고도(m)"] = (t["기압차(hPa)"] * 8.3).round(0)
    return t.sort_values("추정 고도(m)")
```

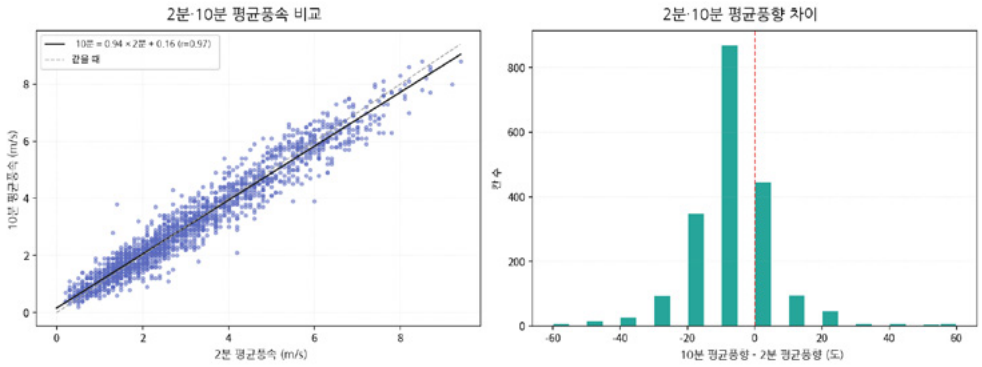


그림 10-4. 평균기간별 바람 비교와 기압을 활용한 고도 추정

지점	해면	현지	기압차(hPa)	추정 고도(m)
113	1008.7	1007.9	0.8	7
167	1009	1008.1	0.9	7
151	1009.4	1008.45	0.95	8
163	1007.6	1006.5	1.1	9
110	1008.8	1006.8	2	17
182	1008	1005.3	2.7	22
92	1009.8	1002.2	7.6	63

❻ 기온·이슬점 차이와 시정 저하 분석

```
def spread(d):
  """기온과 이슬점의 차이. 0 에 가까워지면 안개가 생길 자리다."""
  s = d.copy()
  s["기온-이슬점(℃)"] = s["기온(℃)"] - s["이슬점(℃)"]
  return s
def fog_hours(d, gap=1.0, vis=5000):
  """이슬점차가 좁고 시정이 나빴던 칸을 공항마다 집계합니다."""
  s = spread(d)
  t = (
    s.groupby("지점")
    .agg(
      칸수=("시정(m)", "size"),
      이슬점차좁음=("기온-이슬점(℃)", lambda x: int((x <= gap).sum())),
      시정나쁨=("시정(m)", lambda x: int((x <= vis).sum())),
      최저시정=("시정(m)", "min"),
      최저운고=("운고(ft)", "min"),
    )
    .reset_index()
  )
  t["둘 다(칸)"] = [
    int(((g["기온-이슬점(℃)"] <= gap) & (g["시정(m)"] <= vis)).sum())
    for _, g in s.groupby("지점")
  ]
  return t
```

항공

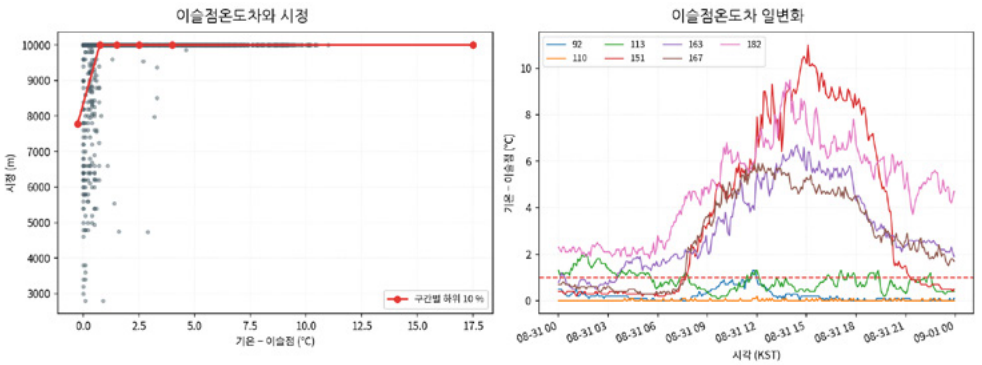


그림 10-5. 기온·이슬점 차이와 시정 저하 분석

지점	칸수	이슬점차 좁음	시정나쁨	최저시정	최저운고	둘 다(칸)
92	288	285	14	2800	1312.34	14
110	288	288	1	3800	2296.59	1
113	288	193	2	2800	2624.67	2
151	288	126	0	7977	9514.44	0
163	288	40	2	4732	4265.09	0
167	288	92	1	4937	2296.59	1
182	288	0	0	10000	4921.26	0

11. 공항기상라이다(TDWL)

공항 주변 바람 분포 분석

데이터 활용 경로(API)

기상청 API허브 > 항공기상 > LIDAR 관측자료

❶ 라이다 자료 조회와 격자 해석

고도를 달리 조회해 상공 6층을 비교합니다.

```
https://apihub.kma.go.kr/api/typ01/url/lidar.php?tm=202609011200&stn=cju&var=hwind&altitude=300&authKey=...
<product datatype="V" datetime="2026-09-01T03:04Z" name="HWIND" type="hwind">
 <data date="2026-09-01" time="03:00:00">
  <cell refid="ok">0 0 0 1 1 0 0 0 0 1 1 1 ···</cell>
  <cell refid="lon">126.431693 126.447894 126.464095 ···</cell>
  <cell refid="lat">33.627117 33.627121 33.627123 ···</cell>
  <cell refid="hordir">0 0 0 83.19 59.05 0 0 ···</cell>
  <cell refid="horspeed">0 0 0 9.66 3.23 0 0 ···</cell>
# ok 가 1 인 셀만 풍향·풍속이 값이다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LID_URL = "https://apihub.kma.go.kr/api/typ01/url/lidar.php"
STN = "cju"
ALTS = [300, 600, 900, 1200, 1500, 1800]  # ft
CROSS_LIMIT = 15 * 0.5144444444 # 15 knot 분석용 문턱, 운항 제한 기준 아님
```

CJU : 공항 기준점 (경도,위도). 공항 메타데이터 확인 후 입력

필수 입력: CJU. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.

조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def trim(d):
  """품질처리된 유효 바람을 사용. 거리만으로 강풍을 삭제하지 않는다."""
  return d.dropna(subset=["풍향", "풍속"])
def get_lidar(tm, alt, stn=None):
  if stn is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: stn")
  "한 시각·한 고도의 수평바람장. 격자 셀마다 경도·위도·풍향·풍속과\n      그 셀을 믿을 수 있는지(ok)가 함께 온다."
  r = requests.get(
    LID_URL,
    params={
      "tm": tm,
      "stn": stn,
```

항공

```
        "var": "hwind",
        "altitude": alt,
        "authKey": API_KEY,
      },
      timeout=60,
    )
    if not r.ok:
      raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
    return r.text

def parse_lidar(xml_text):
    """XML의 refid로 변수별 배열을 읽고 같은 격자 위치의 값을 연결합니다."""

    root = ET.fromstring(xml_text)
    got = {}
    for c in root.iter("cell"):
      got[c.get("refid")] = np.array(c.text.split(), dtype=float)
    d = root.find(".//data")
    tm = f"{d.get('date')}T{d.get('time')}" if d is not None else None
    f = pd.DataFrame(
      {
        "경도": got["lon"],
        "위도": got["lat"],
        "ok": got["ok"].astype(int),
        "풍향": got["hordir"],
        "풍속": got["horspeed"],
      }
    )
    f.loc[f["ok"] == 0, ["풍향", "풍속"]] = np.nan
    return (tm, f)
def walk_lidar(tms=None, alts=None, pause=0.2):
    if tms is None or alts is None:
      raise ValueError("조회조건·메타데이터를 명시하세요: tms, alts")
    "시각 × 고도를 돌아 하루치를 모은다. 격자 자리는 늘 같으므로 한 번만 적어 둔다."
    out, grid = ([], False)
    for tm in tms:
      for al in alts:
        utc, f = parse_lidar(get_lidar(tm, al, STN))
        if not grid:
          out.append(f"@GRID n={len(f)}")
          out.append(" ".join((f"{v:.6f}" for v in f["경도"])))
          out.append(" ".join((f"{v:.6f}" for v in f["위도"])))
          grid = True
        g = f[f["ok"] == 1]
        out.append(f"@REC tm={tm} alt={al} utc={utc} n={len(f)} ok={len(g)}")
        out.append(
          " ".join(
            (
              f"{i}:{r['풍향']:.1f}:{r['풍속']:.1f}"
              for i, r in g.iterrows()
            )
          )
        )
        time.sleep(pause)
    return "\n".join(out)
```

❷ 관측 격자와 유효자료 분포

공항을 중심으로 관측 범위와 유효자료가 집중된 위치를 확인합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
    """모아 둔 묶음을 읽습니다. 값이 있는 셀만 「번호:풍향:풍속」으로 적혀 있다."""
    lines = Path(path).read_text("utf-8").splitlines()
    lon = np.array(lines[1].split(), dtype=float)
    lat = np.array(lines[2].split(), dtype=float)
    recs, i = ({}, 3)
    while i < len(lines):
      if not lines[i].startswith("@REC"):
        i += 1
        continue
      h = dict((kv.split("=") for kv in lines[i].split()[1:]))
      v = np.full((len(lon), 2), np.nan)
```

```
    for tok in lines[i + 1].split() if i + 1 < len(lines) else []:
      k, dr, sp = tok.split(":")
      v[int(k)] = (float(dr), float(sp))
    recs[h["tm"], int(h["alt"])] = v
    i += 2
  return (lon, lat, recs)
def frame(lon, lat, recs, tm, alt, *, CJU):
  """격자자료을 표로 만든다. 풍향·풍속을 동서·남북 성분으로 바꿔 두면 그리기가 쉽다."""
  v = recs[tm, alt]
  f = pd.DataFrame({"경도": lon, "위도": lat, "풍향": v[:, 0], "풍속": v[:, 1]})
  f["U"] = -f["풍속"] * np.sin(np.radians(f["풍향"]))
  f["V"] = -f["풍속"] * np.cos(np.radians(f["풍향"]))
  f["거리(km)"] = km(f["경도"], f["위도"], *CJU)
  return f
def km(lo, la, lo0, la0):
  return np.hypot(
    (np.asarray(lo) - lo0) * 111 * np.cos(np.radians(la0)),
    (np.asarray(la) - la0) * 111,
  )
def rows_of(lon):
  """경도가 초기값으로 돌아가는 위치를 기준으로 격자 행을 구분하고 행별 셀 수를 확인합니다."""

  drop = np.where(np.diff(lon) < 0)[0]
  return np.diff(np.r_[-1, drop, len(lon) - 1])
def grid_shape(lon, lat, *, CJU):
  """격자가 어디를 덮는지. 이웃한 셀까지의 거리로 한 칸의 크기도 잰다."""
  d = km(lon, lat, *CJU)
  rw = rows_of(lon)
  near = []
  for i in range(0, len(lon), 7):
    dd = km(lon, lat, lon[i], lat[i])
    dd[i] = np.inf
    near.append(dd.min())
  return pd.DataFrame(
    {
      "항목": [
        "격자 셀 수",
        "줄 수",
        "줄마다 셀 수",
        "경도 범위",
        "위도 범위",
        "공항에서 가장 가까운 셀",
        "공항에서 가장 먼 셀",
        "이웃 셀까지 거리(중앙값)",
        "덮는 넓이(추정)",
      ],
      "값": [
        len(lon),
        len(rw),
        f"{rw.min()} ~ {rw.max()} (중앙값 {int(np.median(rw))})",
        f"{lon.min():.4f} ~ {lon.max():.4f}",
        f"{lat.min():.4f} ~ {lat.max():.4f}",
        f"{d.min():.2f} km",
        f"{d.max():.1f} km",
        f"{np.median(near):.2f} km",
        f"{len(lon) * np.median(near) ** 2:.0f} km²",
      ],
    }
  )
def far_outliers(lon, lat, recs, thr=20.0, *, CJU):
  """거리 구간별로 강한 바람의 분포를 비교합니다. 관측오류 여부는 품질정보와 주변 관측을 대조해 확인합니다."""

  sp, ds = ([], [])
  for key in recs:
    f = frame(lon, lat, recs, *key, CJU=CJU).dropna(subset=["풍속"])
    sp += list(f["풍속"])
    ds += list(f["거리(km)"])
  sp, ds = (np.array(sp), np.array(ds))
  hi = sp > thr
  return pd.DataFrame(
    {
      "구분": [f"{thr:.0f} m/s 넘는 값", f"{thr:.0f} m/s 이하"],
```

항공

```
        "셀 수": [int(hi.sum()), int((~hi).sum())],
        "비율(%)": [
          round(100 * float(hi.mean()), 2),
          round(100 * float((~hi).mean()), 2),
        ],
        "공항에서 거리 중앙값(km)": [
          round(float(np.median(ds[hi])), 1),
          round(float(np.median(ds[~hi])), 1),
        ],
        "90 백분위(km)": [
          round(float(np.percentile(ds[hi], 90)), 1),
          round(float(np.percentile(ds[~hi], 90)), 1),
        ],
      }
    )
```

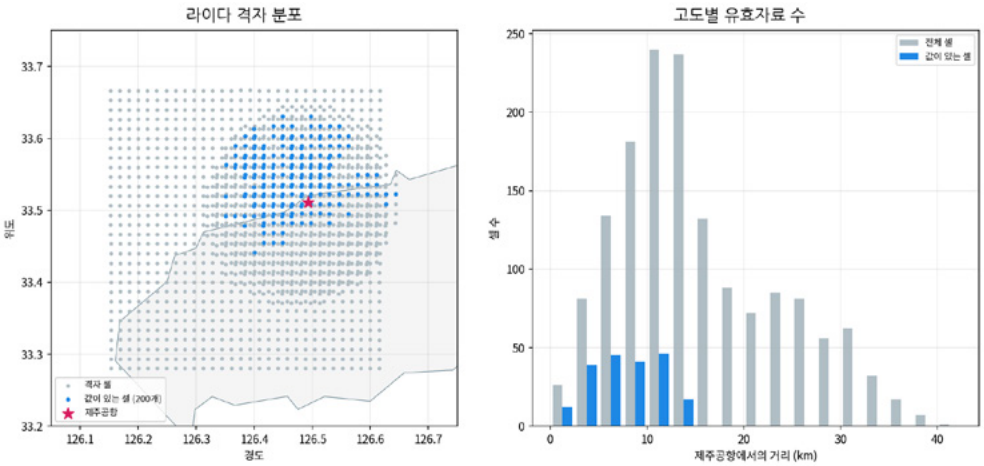


그림 11-1. 관측 격자와 유효자료 분포

항목	값
격자 셀 수	1532
줄 수	70
줄마다 셀 수	6 ~ 30 (중앙값 20)
경도 범위	126.1523 ~ 126.6432
위도 범위	33.2800 ~ 33.6666
공항에서 가장 가까운 셀	0.48 km
공항에서 가장 먼 셀	40.6 km
이웃 셀까지 거리(중앙값)	0.67 km
덮는 넓이(추정)	691 km²

구분	셀 수	비율(%)	공항에서 거리 중앙값(km)	90 백분위(km)
20 m/s 넘는 값	124	0.52	21.9	27.9
20 m/s 이하	23703	99.48	8.7	13.9

❸ 바람 벡터로 본 수평바람장

풍향·풍속을 동서·남북 성분으로 바꾸어 셀별 화살표로 표시합니다.

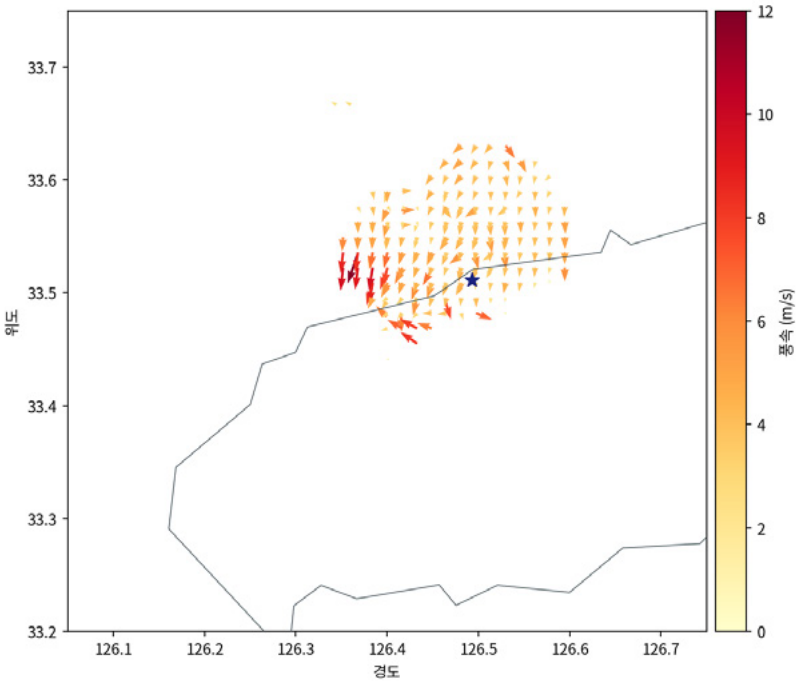


그림 11-2. 바람 벡터로 본 수평바람장

❹ 여섯 고도의 바람 비교

같은 위치의 다른 고도 자료를 조회합니다.

```
def mean_wind(lon, lat, recs, *, CJU):
  """판마다 평균 바람. 풍향은 그냥 평균하면 안 되므로 성분으로 평균한 뒤 되변환합니다."""
  rows = []
  for (tm, al), _ in recs.items():
    f = trim(frame(lon, lat, recs, tm, al, CJU=CJU))
    if f.empty:
      continue
    u, v = (f["U"].mean(), f["V"].mean())
    rows.append(
      {
        "시각": tm,
        "시": int(tm[8:10]),
        "고도(ft)": al,
        "값 있는 셀": len(f),
        "유효율(%)": round(100 * len(f) / len(lon), 1),
        "평균풍속(m/s)": round(float(f["풍속"].mean()), 2),
        "최대풍속(m/s)": round(float(f["풍속"].max()), 1),
        "벡터평균(m/s)": round(float(np.hypot(u, v)), 2),
        "평균풍향(도)": round(
          float((np.degrees(np.arctan2(-u, -v)) + 360) % 360), 0
        ),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
```

항공

```
        else [
          "시각",
          "시",
          "고도(ft)",
          "값 있는 셀",
          "유효율(%)",
          "평균풍속(m/s)",
          "최대풍속(m/s)",
          "벡터평균(m/s)",
          "평균풍향(도)",
        ]
      ),
    ).sort_values(["고도(ft)", "시"])
```

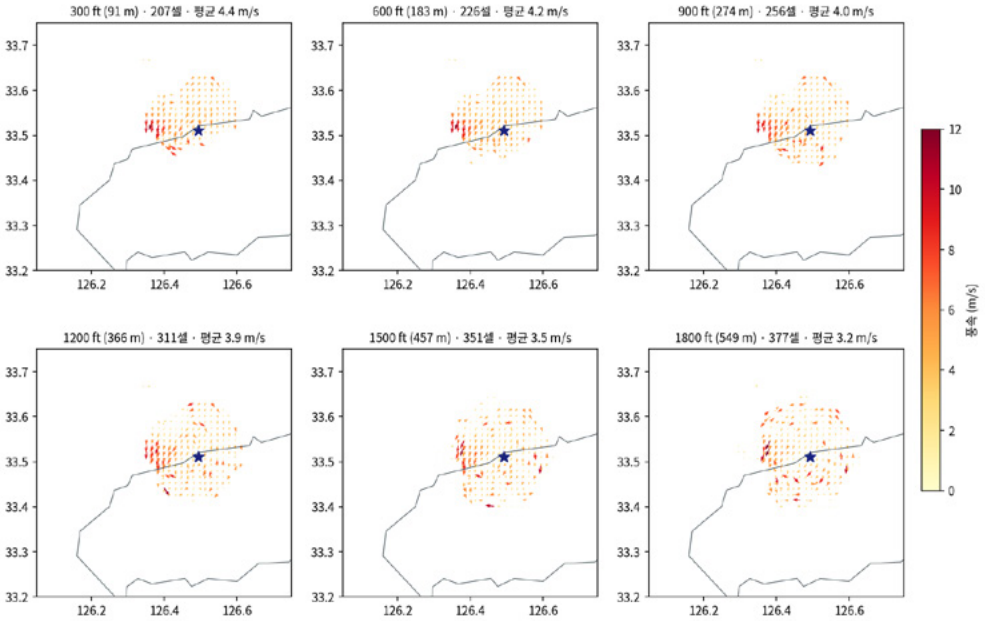


그림 11-3. 6개 고도 바람 비교

고도 (ft)	값 있는 셀	유효율 (%)	평균풍속 (m/s)	최대풍속 (m/s)	벡터평균 (m/s)	평균풍향 (도)
300	205	13.4	4.38	11.8	3.82	14
600	224	14.6	4.25	11.3	3.83	12
900	254	16.6	4.04	9.5	3.6	14
1200	309	20.2	3.89	11.1	3.23	14
1500	351	22.9	3.47	11.7	2.58	17
1800	376	24.5	3.25	12.2	2.15	21

❺ 시간·고도별 풍속과 자료 수

각 시각·고도의 유효 셀 수와 평균 풍속을 두 개의 분포도로 표출합니다.

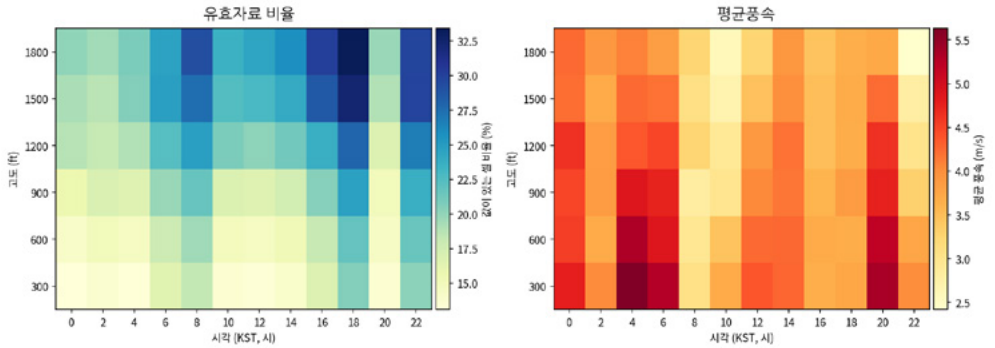


그림 11-4. 시간·고도별 풍속과 자료 수

시	300	600	900	1200	1500	1800
0	4.8	4.5	4.5	4.6	4.2	4.2
2	4	3.7	3.9	3.8	3.7	3.9
4	5.6	5.3	4.9	4.4	4.2	4.1
18	3.7	3.7	3.9	3.7	3.7	3.7
20	5.4	5.2	4.8	4.6	4.2	3.7
22	4	3.8	3.3	3	2.8	2.4

❻ 활주로 방향을 기준으로 한 측풍 분석

시간·고도별 측풍을 표시해 측풍이 강해지는 위치와 시기를 확인합니다.

```
def runway_frame(lon, lat, recs, heading=None, *, CJU):
  if heading is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: heading")
  "활주로 방향에 대고 바람을 나눈다. 활주로 축에 나란한 것이 정풍,\n     가로지르는 것이 측풍이고, 이·착륙에서 문제가 되는 것은
측풍이다."
  rows = []
  for tm, al in recs:
    f = trim(frame(lon, lat, recs, tm, al, CJU=CJU))
    if f.empty:
      continue
    ang = np.radians(f["풍향"] - heading)
    head = f["풍속"] * np.cos(ang)
    cross = f["풍속"] * np.sin(ang)
    rows.append(
      {
        "시각": tm,
        "시": int(tm[8:10]),
        "고도(ft)": al,
        "정풍 평균(m/s)": round(float(head.mean()), 2),
        "측풍 평균(m/s)": round(float(cross.abs().mean()), 2),
        "측풍 최대(m/s)": round(float(cross.abs().max()), 1),
        "측풍 15kt 넘는 셀(%)": round(
          100 * float((cross.abs() > CROSS_LIMIT).mean()), 1
        ),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
```

항공

```
        else [
          "시각",
          "시",
          "고도(ft)",
          "정풍 평균(m/s)",
          "측풍 평균(m/s)",
          "측풍 최대(m/s)",
          "측풍 15kt 넘는 셀(%)",
        ]
      ),
    ).sort_values(["고도(ft)", "시"])
```

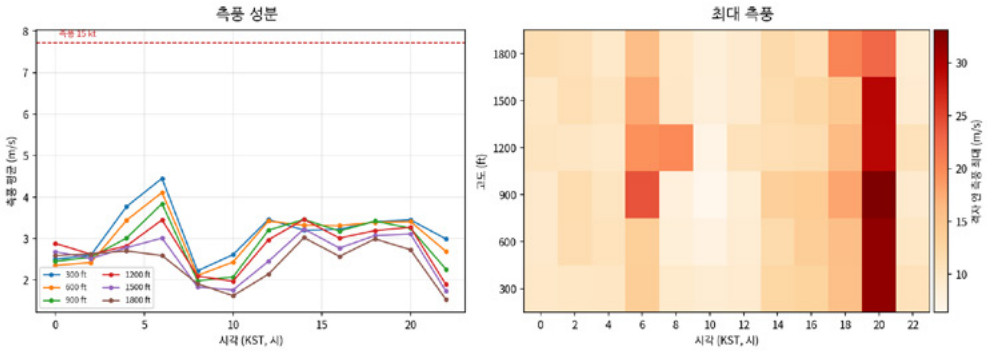


그림 11-5. 활주로 방향을 기준으로 한 측풍 분석

시	300	600	900	1200	1500	1800
0	2.5	2.35	2.45	2.88	2.68	2.59
2	2.6	2.42	2.55	2.62	2.52	2.63
4	3.77	3.44	3.01	2.82	2.78	2.7
18	3.41	3.39	3.42	3.19	3.07	2.99
20	3.45	3.41	3.25	3.27	3.11	2.73
22	2.99	2.69	2.26	1.89	1.73	1.53

❼ 평균바람 대비 공간 변동

공간적 변동성과 풍속의 관계를 분석합니다.

```
def spread_frame(lon, lat, recs, *, CJU):
    """같은 시각·고도의 격자별 바람을 평균값과 비교하여 공간 변동을 계산합니다."""

    rows = []
    for tm, al in recs:
        f = trim(frame(lon, lat, recs, tm, al, CJU=CJU))
        if len(f) < 20:
            continue
        u0, v0 = (f["U"].mean(), f["V"].mean())
        dev = np.hypot(f["U"] - u0, f["V"] - v0)
        rows.append(
            {
                "시각": tm,
                "시": int(tm[8:10]),
                "고도(ft)": al,
                "평균 풍속(m/s)": round(float(f["풍속"].mean()), 2),
                "벡터 평균(m/s)": round(float(np.hypot(u0, v0)), 2),
                "고른 정도": round(float(np.hypot(u0, v0) / f["풍속"].mean()), 2),
                "흩어짐 평균(m/s)": round(float(dev.mean()), 2),
```

```
        "흩어짐 최대(m/s)": round(float(dev.max()), 1),
        "가장 다른 셀 거리(km)": round(
          float(f.loc[dev.idxmax(), "거리(km)"]), 1
        ),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "시각",
        "시",
        "고도(ft)",
        "평균 풍속(m/s)",
        "벡터 평균(m/s)",
        "고른 정도",
        "흩어짐 평균(m/s)",
        "흩어짐 최대(m/s)",
        "가장 다른 셀 거리(km)",
      ]
    ),
  ).sort_values(["고도(ft)", "시"])
```

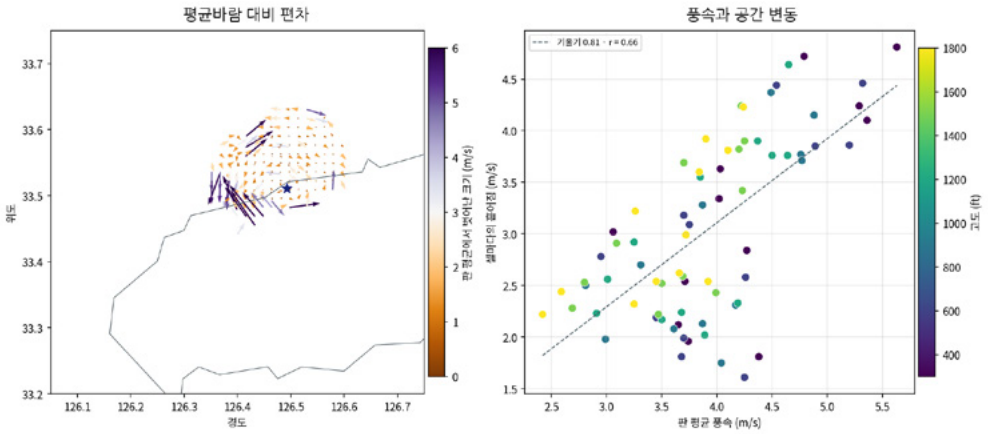


그림 11-6. 평균 바람을 기준으로 한 공간적 변동

고도(ft)	판수	평균풍속	벡터평균	고른정도	흩어짐	흩어짐최대
300	12	4.33	2.62	0.61	3.26	33.3
600	12	4.14	2.66	0.65	2.99	33.5
900	12	3.96	2.45	0.62	2.89	34.8
1200	12	3.87	2.07	0.55	3.01	36.2
1500	12	3.65	1.59	0.44	3.05	31.4
1800	12	3.53	1.36	0.39	3.04	27.7

데이터 활용사례 50

기상위성

Weather satellite



정지궤도
기상위성

12-①.
천리안위성 2A호(GK2A)

적외영상과 안개 분포 비교

데이터 활용 경로(API)

기상청 API허브 > 위성 > 천리안 2A호

❶ 위성 원본·산출물 파일 읽기

도법 정보로 격자와 위경도를 연결합니다.

```
https://apihub.kma.go.kr/api/typ05/api/GK2A/LE1B/IR105/KO/data?date=202608310000&authK
ey=...
https://apihub.kma.go.kr/api/typ05/api/GK2A/LE2/FOG/KO/data?date=202608310000&authKey=...
#  응답은 HDF5(netCDF) 파일 하나 — 변수 image_pixel_values 는 uint16 900×900
#  gk2a_imager_projection 변수의 속성이 도법을 알려 준다
#    lambert_conformal_conic · 표준위도 30·60 · 원점 38N 126E
#    화소 2 km · 왼쪽 위 (-899000, 899000) m
#  FOG 는 1 맑음 · 2 중상층운 · 3 판단 못 함 · 4 안개 가능 · 5 안개 · 6 눈 · 7 사막
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LE1B_URL = "https://apihub.kma.go.kr/api/typ05/api/GK2A/LE1B/{ch}/{area}/data"
LE2_URL = "https://apihub.kma.go.kr/api/typ05/api/GK2A/LE2/{prod}/{area}/data"
FOG_KO = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

범주와 상태는 공식 코드표로 확인합니다. 코드표에 없는 값은 원래 코드로 표시합니다.
채널·산출물의 실제 2차원 배열을 np.save("ir.npy", array), np.save("fog.npy", array)로 저장합니다. 두 자료의 시간·투영·해상도·범위를 일치시킨 뒤 비교합니다. 속성의 결측 및 채널 보정계수를 적용하며 화솟값 크기를 온도로 단정하지 않습니다.
PROJ : 도법 dict: ul_e·ul_n(m), pixel(m), sp1·sp2·olat·olon(도). 배열 원점/행 방향도 일치시킬 것
필수 입력: PROJ. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.
조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_le1b(ch, area, tm, retry=3):
    """적외·가시 채널 원본. HDF5(netCDF) 파일 하나가 통째로 반환됩니다."""
    url = LE1B_URL.format(ch=ch, area=area)
    for i in range(retry):
        try:
            r = requests.get(
                url, params={"date": tm, "authKey": API_KEY}, timeout=300
            )
            if not r.ok:
                raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
            return r.content
        except Exception as exc:
            if i == retry - 1:
                raise
            log(f"{ch}/{area} 재시도 {i + 1}: {type(exc).__name__}")
            time.sleep(3 * (i + 1))
def get_le2(prod, area, tm, retry=3):
    """기상산출물. 값이 아니라「뜻」이 붙은 범주가 들어 있는 것이 많다."""
    url = LE2_URL.format(prod=prod, area=area)
    for i in range(retry):
        try:
            r = requests.get(
                url, params={"date": tm, "authKey": API_KEY}, timeout=300
            )
            if not r.ok:
                raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
            return r.content
        except Exception as exc:
            if i == retry - 1:
                raise
            log(f"{prod}/{area} 재시도 {i + 1}: {type(exc).__name__}")
            time.sleep(3 * (i + 1))
```

❷ 도법 정보를 활용한 위경도 변환

```
def lcc_xy(nx, ny, *, PROJ):
    """격자 번호를 지도 좌표(m)로. 왼쪽 위 모서리와 화소 크기만 있으면 된다."""
    e = PROJ["ul_e"] + (nx + 0.5) * PROJ["pixel"]
    n = PROJ["ul_n"] - (ny + 0.5) * PROJ["pixel"]
    return (e, n)
def lcc_lonlat(e, n, *, PROJ):
    """파일의 도법 인자를 적용하여 람베르트 정각원추 좌표를 위경도로 변환합니다."""

    d = math.pi / 180.0
    re = 6371.00877 * 1000.0
    sp1, sp2 = (PROJ["sp1"] * d, PROJ["sp2"] * d)
    olat, olon = (PROJ["olat"] * d, PROJ["olon"] * d)
    sn = math.log(math.cos(sp1) / math.cos(sp2)) / math.log(
        math.tan(math.pi * 0.25 + sp2 * 0.5)
        / math.tan(math.pi * 0.25 + sp1 * 0.5)
    )
    sf = math.tan(math.pi * 0.25 + sp1 * 0.5) ** sn * math.cos(sp1) / sn
    ro = re * sf / math.tan(math.pi * 0.25 + olat * 0.5) ** sn
    x = np.asarray(e)
    y = ro - np.asarray(n)
    ra = np.hypot(x, y)
    if sn < 0:
        ra = -ra
    lat = 2.0 * np.arctan((re * sf / ra) ** (1.0 / sn)) - math.pi * 0.5
    lon = np.arctan2(x, y) / sn + olon
    return (lon / d, lat / d)
def grid_lonlat(n=None, step=None, *, PROJ):
    if n is None or step is None:
        raise ValueError("조회조건·메타데이터를 명시하세요: n, step")
    "솎아 낸 격자의 위경도 판을 만든다."
    ix = np.arange(n) * step
    iy = np.arange(n) * step
    X, Y = np.meshgrid(ix, iy)
    e, nn = lcc_xy(X, Y, PROJ=PROJ)
    return lcc_lonlat(e, nn, PROJ=PROJ)
```

정지궤도 기상위성

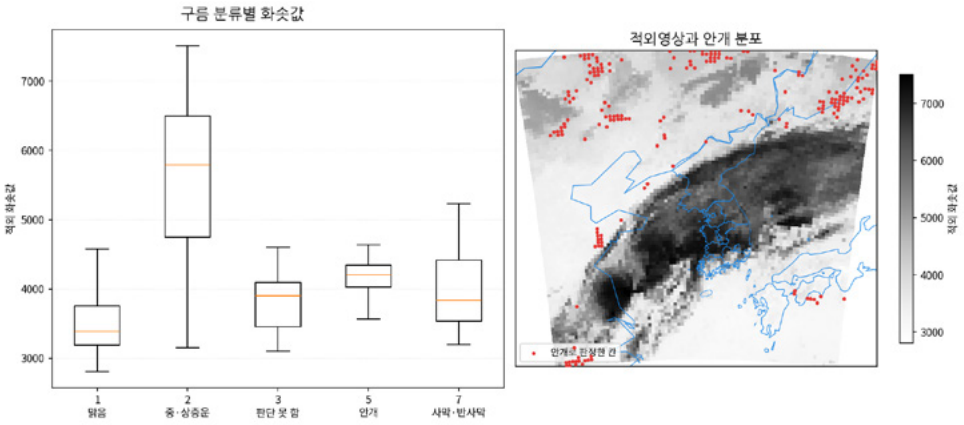


그림 12-1-1. 도법 정보를 활용한 위경도 변환

격자점	경도	위도
왼쪽 위 (0,0)	113.978	45.712
오른쪽 위 (899,0)	138.049	45.709
왼쪽 아래 (0,899)	116.744	29.324
오른쪽 아래 (899,899)	135.276	29.322
한가운데 (450,450)	126.023	37.982

❸ 적외 화솟값과 구름 분류의 관계

중·상층운으로 분류한 셀과 맑음 셀을 비교합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_ir(path):
    """2차원 NumPy 파일 읽기. 원시 채널의 물리온도 변환은 보정표 적용 후 수행."""
    a = np.load(path, allow_pickle=False)
    if a.ndim != 2:
        raise ValueError("2차원 원시 채널 배열을 저장")
    return a.astype(float)



def ir_direction(ir, fog):
    """구름 분류별 적외 화솟값의 분포를 비교합니다. 온도 해석에는 해당 채널의 보정계수와 단위 확인이 필요합니다."""

    hi = float(np.median(ir[fog == 2])) if (fog == 2).any() else np.nan
    lo = float(np.median(ir[fog == 1])) if (fog == 1).any() else np.nan
    return (hi, lo, "범주별 원시값 차이(물리적 온도는 채널 보정표로 산출)")
def ir_table(ir):
    """화솟값이 어떻게 퍼져 있는지. 방향은 ir_direction 이 자료에서 정한다."""
    q = np.percentile(ir, [1, 5, 25, 50, 75, 95, 99])
    return pd.DataFrame(
        {
            "백분위": ["1 %", "5 %", "25 %", "50 %", "75 %", "95 %", "99 %"],
            "화솟값": q.round(0).astype(int),
        }
    )
def cold_mask(ir, q=90):
    """원시 화솟값의 상위 백분위 구간을 선택합니다. 보정 전 화솟값의 대소를 실제 온도의 대소로 해석하지 않습니다."""

    thr = np.percentile(ir, q)
    return (ir >= thr, float(thr))
```

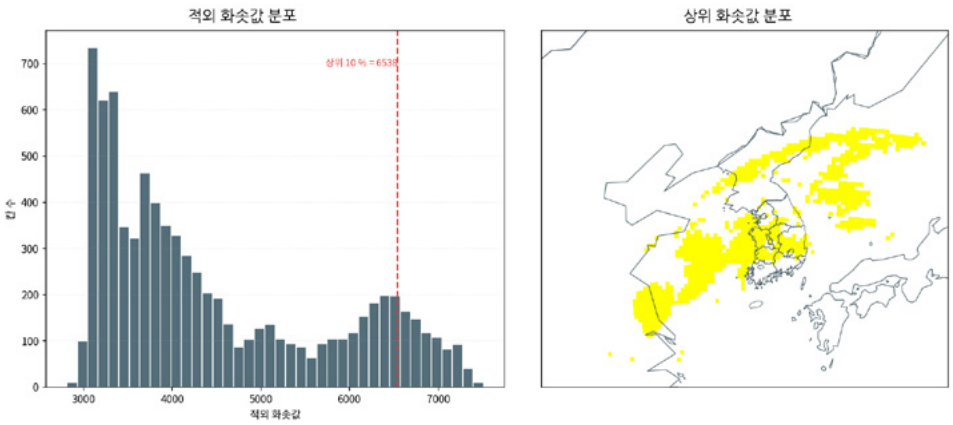


그림 12-1-2. 적외 화솟값과 구름 분류의 관계

백분위	화솟값
1%	3038
5%	3092
25%	3382
50%	4004
75%	5457
95%	6855
99%	7245

❹ 안개 산출물의 범주별 분포

```
def read_rle(path):
    """범주 2차원 NumPy 파일. 자체 RLE 입력 대신 배열 모양을 파일에 보존."""
    a = np.load(path, allow_pickle=False)
    if a.ndim != 2:
        raise ValueError("2차원 범주 배열을 저장")
    return a
def fog_table(fog):
    """범주마다 몇 칸인지. 값이 아니라 뜻이므로 세는 것이 곧 요약이다."""
    v, c = np.unique(fog, return_counts=True)
    t = pd.DataFrame(
        {"코드": v, "뜻": [FOG_KO.get(int(x), "?") for x in v], "칸 수": c}
    )
    t["비율(%)"] = (100 * t["칸 수"] / t["칸 수"].sum()).round(1)
    return t
```

정지궤도 기상위성

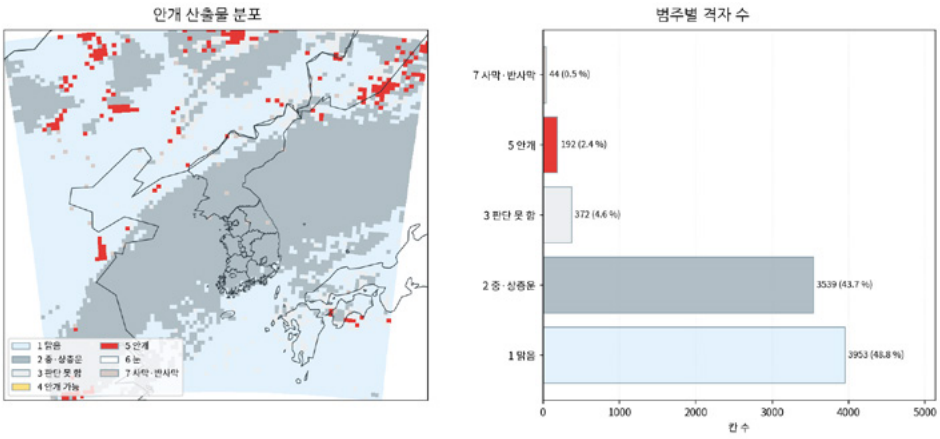


그림 12-1-3. 안개 산출물의 범주별 분포

코드	뜻	칸 수	비율(%)
1	맑음	3953	48.8
2	중·상층운	3539	43.7
3	판단 못 함	372	4.6
5	안개	192	2.4
7	사막·반사막	44	0.5

❺ 같은 격자의 원본·산출물 비교

```
def ir_by_fog(ir, fog):
    """범주마다 적외 화솟값이 어느 구간에 놓이는지."""
    rows = []
    for v in sorted(set(fog.ravel())):
        m = ir[fog == v]
        rows.append(
            {
                "코드": int(v),
                "뜻": FOG_KO.get(int(v), "?"),
                "칸 수": int(m.size),
                "중앙값": int(np.median(m)),
                "25 %": int(np.percentile(m, 25)),
                "75 %": int(np.percentile(m, 75)),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None if rows else ["코드", "뜻", "칸 수", "중앙값", "25 %", "75 %"]
        ),
    )
```

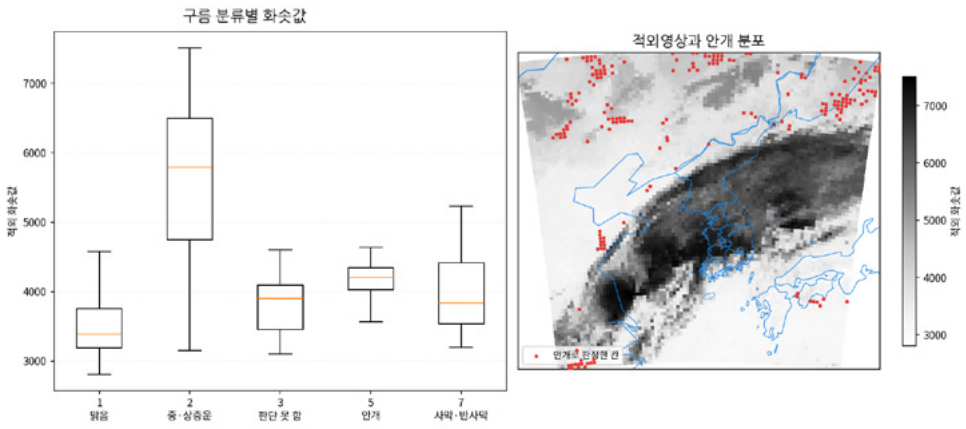


그림 12-1-4. 같은 격자의 원본·산출물 비교

코드	뜻	칸 수	중앙값	25%	75%
1	맑음	3953	3392	3193	3755
2	중·상층운	3539	5792	4748	6494
3	판단 못 함	372	3903	3451	4093
5	안개	192	4201	4024	4346
7	사막·반사막	44	3833	3539	4420

정지궤도
기상위성

12-②.
천리안위성 2A호(GK2A)

산불위험도와 화점 분포 분석

데이터 활용 경로(API)
기상청 API허브 > 위성 > 천리안 2A호

❶ 산불 산출물 조회와 격자 변환

```
https://apihub.kma.go.kr/api/typ01/cgi-bin/sat/nph-sat_ana_txt
 ?tm=202609010900&obs=fr&disp=A&authKey=...
#  격자정보(NY, NX) = 900, 900
-9999,-9999,-9999,  62,  71,  80,  85, …
#  obs=fr 산불위험도(0~99) · obs=ff 산불탐지(0/1)
#  -9999 는 산출 대상이 아닌 자리 · 값은 아래에서 위로 쌓여 온다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
ANA_URL = "https://apihub.kma.go.kr/api/typ01/cgi-bin/sat/nph-sat_ana_txt"
```

KO : load_regions 반환 객체. EPSG:4326 경계 GeoJSON에서 생성
MISS : 공식 결측값 또는 처리경계
NX : 열 수
NY : 행 수
PROJ : 도법 dict: ul_e·ul_n(m), pixel(m), sp1·sp2·olat·olon(도). 배열 원점/행 방향도 일치시킬 것
RISK : 산불위험도 분석용 (하한,상한,표시명) 구간 목록. 공식 위험등급과 구분
필수 입력: KO, MISS, NX, NY, PROJ, RISK. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def save_fire_pack(fr_tm, ff_times, *, nx, ny, fire_codes, path="fire_pack.npz"):
    """공식 fr/ff 응답을 원래 격자값으로 저장. 탐지코드는 공식 코드표로 지정."""
    if not fire_codes:
        raise ValueError("fire_codes에 공식 화점 탐지 코드를 지정하세요")
    fr = parse_ana(get_ana(fr_tm, "fr"), nx=nx, ny=ny)
    arrays = {
        "fr": fr,
        "fr_tm": np.array(str(fr_tm)),
        "fire_codes": np.asarray(fire_codes, dtype=float),
    }
    times = list(ff_times)
    arrays["ff_times"] = np.asarray(times, dtype=str)
    for i, tm in enumerate(times):
        arrays[f"ff_{i}"] = parse_ana(get_ana(tm, "ff"), nx=nx, ny=ny)
        time.sleep(0.2)
    np.savez_compressed(path, **arrays)
    return path
```

입력 파일은 save_fire_pack으로 생성한 fire_pack.npz입니다. fr_tm·ff_times는 실제 산출시각(KST), fire_codes는 해당 산출물의 공식 탐지 코드로 지정합니다. read_pack에는 NX·NY·MISS와 분석용 RISK 구간 목록을 전달합니다. 과거 임의 @FR/@FF 텍스트 묶음은 사용하지 않습니다.

```
def rle_decode(text, max_cells=10000000):
    values = []
    for token in text.split():
        match = re.fullmatch("(-?\\d+(?:\\.\\d+)?)x(\\d+)", token)
        if match is None:
            raise ValueError("RLE token 형식")
        value, count = (float(match[1]), int(match[2]))
        if count <= 0 or len(values) + count > max_cells:
            raise ValueError("RLE 길이")
        values.extend([value] * count)
    return np.asarray(values, dtype=float)
def load_regions(path, name_field="name"):
    """EPSG:4326 시도 경계 GeoJSON. name_field는 실제 속성 열 이름."""
    import json
    from shapely.geometry import shape, Point


    payload = json.loads(Path(path).read_text("utf-8"))
    features = payload.get("features", [])
    polygons = []
    for feature in features:
        properties = feature.get("properties", {})
        if name_field not in properties:
            raise ValueError("GeoJSON 시도명 속성 열 확인")
        geometry = shape(feature["geometry"])
        if not geometry.is_valid:
            raise ValueError("유효하지 않은 행정경계")

        polygons.append((properties[name_field], geometry))

    def assign(lon, lat):
        answer = []
        for x, y in zip(lon, lat):
            if not np.isfinite(x) or not np.isfinite(y):
                answer.append(None)
                continue
            point = Point(float(x), float(y))
            matches = [name for name, geom in polygons if geom.covers(point)]

            answer.append(matches[0] if len(matches) == 1 else None)

        return answer

    from types import SimpleNamespace

    return SimpleNamespace(assign_province=assign)
```

정지궤도 기상위성

시·도별 집계에는 EPSG:4326 좌표계의 행정경계 GeoJSON과 shapely가 필요합니다. load_regions()의 name_field에 시·도명 속성을 지정하고 반환 객체를 KO에 전달합니다. 경계가 중복되거나 행정구역이 확인되지 않는 위치는 미분류로 처리합니다.

```
def get_ana(tm, obs):
  """산불 산출물. obs=fr 은 위험도(하루 한 번, 09시 KST), obs=ff 는 탐지(10분 간격)이고 둘 다 900 × 900 격자로 반환됩니
다."""

  r = requests.get(
    ANA_URL,
    params={"tm": tm, "obs": obs, "disp": "A", "authKey": API_KEY},
    timeout=300,
  )
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  r.encoding = "euc-kr"
  return r.text
def parse_ana(text, nx=None, ny=None):
  if nx is None or ny is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: nx, ny")
  "주석을 걷어 내고 숫자만 남긴다. 아래에서 위로, 왼쪽에서 오른쪽 순서다."
  body = " ".join(
    (
      ln
      for ln in text.splitlines()
      if ln.strip() and (not ln.startswith("#"))
    )
  )
  v = np.array([float(x) for x in body.replace(",", " ").split()])
  if v.size != nx * ny:
    raise ValueError("응답 값 개수와 격자 크기가 다릅니다")
  return v.reshape(ny, nx)
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path, *, NX, NY, RISK, MISS):
  """save_fire_pack의 NPZ를 위험도·요약·시각별 화점 표로 변환한다."""
  summ, ff = [], []
  with np.load(path, allow_pickle=False) as saved:
    fr = saved["fr"].astype(float)
    if fr.shape != (NY, NX):
      raise ValueError("NX·NY가 저장 격자와 다릅니다")
    fr[fr <= MISS + 1] = np.nan
    values = fr[np.isfinite(fr)]
    row = {
      "날짜": pd.to_datetime(
        str(saved["fr_tm"].item()), format="%Y%m%d%H%M"
      ).tz_localize("Asia/Seoul"),
      "값 있는 칸": int(values.size),
      "평균": float(values.mean()) if values.size else np.nan,
      "최대": float(values.max()) if values.size else np.nan,
    }
    for low, high, label in RISK:
      row[label] = int(((values >= low) & (values < high)).sum())
    summ.append(row)
    codes = saved["fire_codes"]
    for i, tm in enumerate(saved["ff_times"]):
      a = saved[f"ff_{i}"]
      if a.shape != fr.shape:
        raise ValueError("화점 격자 크기 불일치")
      idx = np.flatnonzero(np.isin(a.ravel(), codes)).tolist()
      ff.append(
        {
          "시각": pd.to_datetime(
            str(tm), format="%Y%m%d%H%M"
          ).tz_localize("Asia/Seoul"),
          "화점 수": len(idx),
          "격자번호": idx,
        }
      )
  return (
    fr,
    pd.DataFrame(summ),
    pd.DataFrame(ff, columns=["시각", "화점 수", "격자번호"]),
  )
```

❷ 산불위험도 격자의 위경도 변환

```
def lcc_xy(nx, ny, *, PROJ):
    """격자 번호를 지도 좌표(m)로. 왼쪽 위 모서리와 화소 크기만 있으면 된다."""
    e = PROJ["ul_e"] + (np.asarray(nx) + 0.5) * PROJ["pixel"]
    n = PROJ["ul_n"] - (np.asarray(ny) + 0.5) * PROJ["pixel"]
    return (e, n)
def lcc_lonlat(e, n, *, PROJ):
    """람베르트 정각원추 도법을 적용하여 위경도를 산출합니다."""
    d = math.pi / 180.0
    re = 6371.00877 * 1000.0
    sp1, sp2 = (PROJ["sp1"] * d, PROJ["sp2"] * d)
    olat, olon = (PROJ["olat"] * d, PROJ["olon"] * d)
    sn = math.log(math.cos(sp1) / math.cos(sp2)) / math.log(
        math.tan(math.pi * 0.25 + sp2 * 0.5)
        / math.tan(math.pi * 0.25 + sp1 * 0.5)
    )
    sf = math.tan(math.pi * 0.25 + sp1 * 0.5) ** sn * math.cos(sp1) / sn
    ro = re * sf / math.tan(math.pi * 0.25 + olat * 0.5) ** sn
    x = np.asarray(e)
    y = ro - np.asarray(n)
    ra = np.hypot(x, y)
    lat = 2.0 * np.arctan((re * sf / ra) ** (1.0 / sn)) - math.pi * 0.5
    lon = np.arctan2(x, y) / sn + olon
    return (lon / d, lat / d)
def axes(step=3, *, NX, NY, PROJ):
    """전국 한 장에 담기 위해 몇 칸마다 하나씩 골라 격자의 위경도를 만든다."""
    ix = np.arange(0, NX, step)
    iy = np.arange(0, NY, step)
    X, Y = np.meshgrid(ix, iy)
    e, n = lcc_xy(X, Y, PROJ=PROJ)
    return lcc_lonlat(e, n, PROJ=PROJ)
def idx_lonlat(idx, *, NX, PROJ):
    """탐지 화점의 격자 번호를 위경도로. 번호를 폭으로 나누면 행과 열이 나반환됩니다."""
    j, i = (np.asarray(idx) // NX, np.asarray(idx) % NX)
    e, n = lcc_xy(i, j, PROJ=PROJ)
    return lcc_lonlat(e, n, PROJ=PROJ)
def corner_table(*, NX, NY, PROJ):
    rows = []
    for nm, x, y in [
        ("(0, 0) 왼쪽 위", 0, 0),
        ("(899, 0) 오른쪽 위", NX - 1, 0),
        ("(0, 899) 왼쪽 아래", 0, NY - 1),
        ("(899, 899) 오른쪽 아래", NX - 1, NY - 1),
        ("가운데 (450, 450)", NX // 2, NY // 2),
    ]:
        e, n = lcc_xy(x, y, PROJ=PROJ)
        lo, la = lcc_lonlat(e, n, PROJ=PROJ)
        rows.append(
            {
                "격자점": nm,
                "경도": round(float(lo), 4),
                "위도": round(float(la), 4),
            }
        )
    return pd.DataFrame(
        rows, columns=None if rows else ["격자점", "경도", "위도"]
    )
def risk_grid(fr, step=3, *, MISS):
    """결측 기준값 이하의 격자를 제외하고 지정 간격으로 추출합니다. 배열의 방향은 공식 격자정보로 확인합니다."""

    a = fr.copy()
    a[a <= MISS + 1] = np.nan
    return a[::step, ::step]
```

정지궤도 기상위성

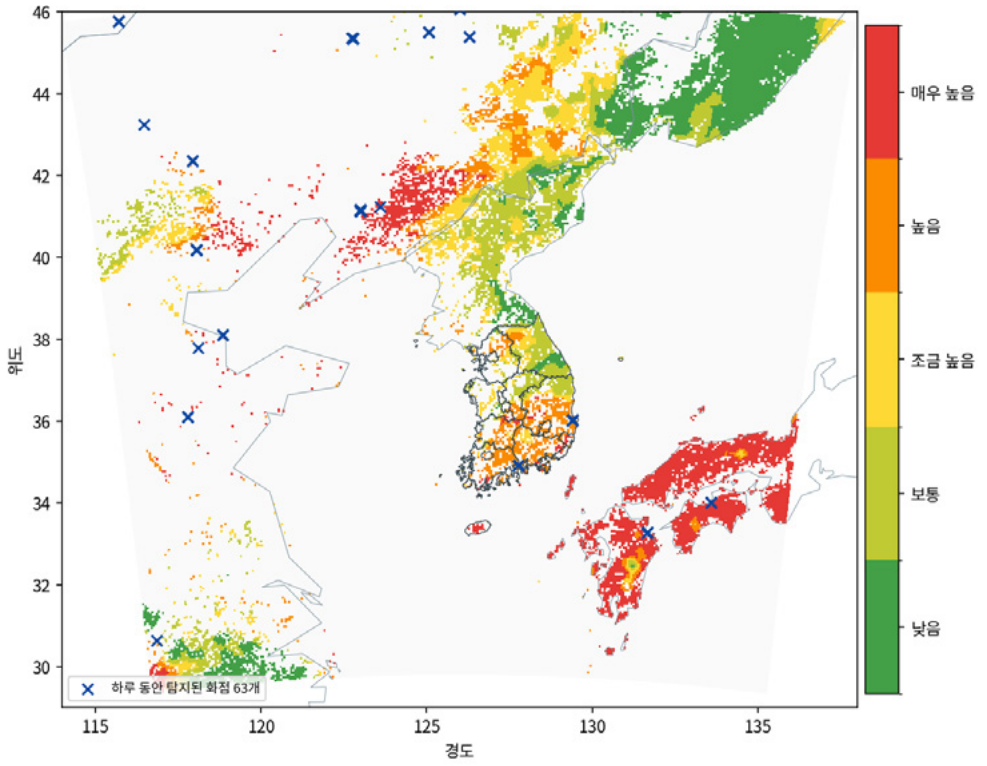


그림 12-2-1. 산불위험도 격자의 위경도 변환

격자점	경도	위도
(0, 0) 왼쪽 위	113.978	45.7117
(899, 0) 오른쪽 위	138.048	45.7089
(0, 899) 왼쪽 아래	116.744	29.3243
(899, 899) 오른쪽 아래	135.276	29.3223
가운데 (450, 450)	126.023	37.9815

등급	칸 수	비율(%)	넓이(km²)
낮음 (0~20)	29508	25	118032
보통 (20~40)	25404	21.5	101616
조금 높음 (40~60)	20216	17.1	80864
높음 (60~80)	14914	12.6	59656
매우 높음 (80~100)	28184	23.8	112736

❸ 시·도 및 위도별 산불위험도

```
def province_frame(g, lon, lat, *, KO):
  if not KO:
    return pd.DataFrame()
  prov = np.asarray(
    KO.assign_province(lon.ravel(), lat.ravel()), dtype=object
  ).reshape(lon.shape)
  rows = []
  for p in sorted({v for v in prov.ravel() if isinstance(v, str)}):
    m = (prov == p) & np.isfinite(g)
    if m.sum() < 5:
      continue
    v = g[m]
    rows.append(
      {
        "시·도": p,
        "칸 수": int(m.sum()),
        "평균": round(float(v.mean()), 1),
        "중앙값": round(float(np.median(v)), 1),
        "80 이상 비율(%)": round(100 * float((v >= 80).mean()), 1),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["시·도", "칸 수", "평균", "중앙값", "80 이상 비율(%)"]
    ),
  ).sort_values("평균", ascending=False)
```

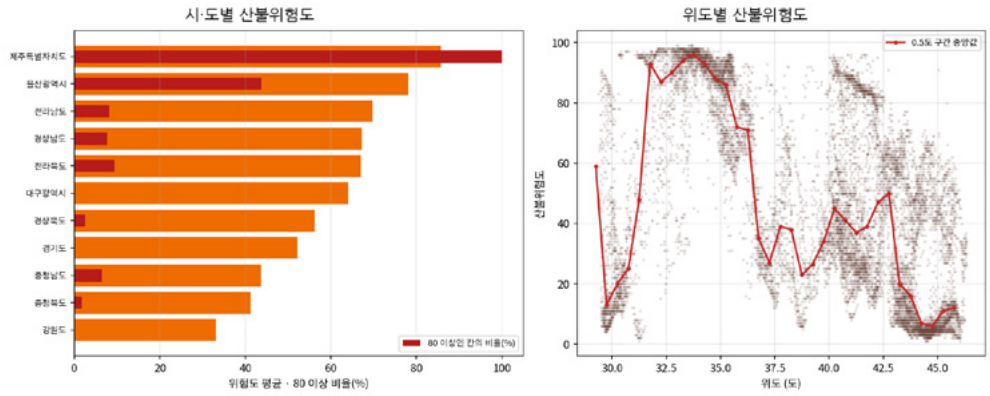


그림 12-2-2. 시·도 및 위도별 산불위험도

시·도	칸 수	평균	중앙값	80 이상 비율(%)
제주특별자치도	19	85.7	86	100
울산광역시	16	78.1	77	43.8
전라남도	136	69.7	69.5	8.1
경상남도	155	67.2	66	7.7
전라북도	97	67	69	9.3
대구광역시	12	64	64	0
경상북도	314	56.2	65	2.5
경기도	89	52.1	48	0
충청남도	62	43.6	44	6.5
충청북도	116	41.3	31	1.7
강원도	375	33.2	29	0

정지궤도 기상위성

❹ 일별 산불위험도 변화

전체 평균과 등급별 비율을 비교합니다.

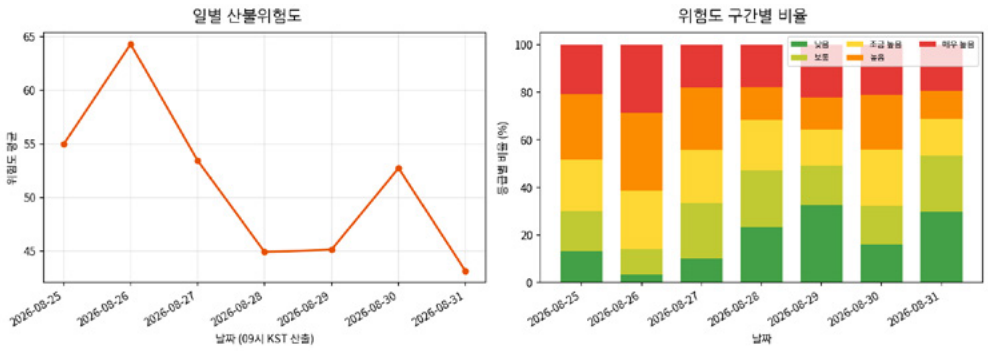


그림 12-2-3. 일별 산불위험도 변화

날짜	유효 격자 수	평균	최대	낮음	보통	조금 높음	높음	매우 높음
08월 25일	118226	54.97	99	15678	19887	25644	32491	24526
08월 26일	118226	64.3	99	3988	12545	29013	38867	33813
08월 27일	118226	53.45	98	11776	27807	26247	31006	21390
08월 28일	118226	44.92	99	27653	28241	25028	16194	21110
08월 29일	118226	45.14	99	38370	19942	17772	15881	26261
08월 30일	118226	52.74	98	19000	19146	28111	27186	24783
08월 31일	118226	43.13	99	35252	27834	18206	14007	22927

❺ 시간별 화점 탐지 위치와 횟수

좌표로 변환하고, 위치별 탐지 횟수를 표시해 반복 탐지된 화점을 확인합니다.

```
def fire_points(ff, *, NX, PROJ):
    """시각별 탐지 격자번호를 위경도로 변환합니다. 반복 탐지는 연속 연소시간과 구분합니다."""

    rows = []
    for _, r in ff.iterrows():
        if not r["격자번호"]:
            continue
        lo, la = idx_lonlat(r["격자번호"], NX=NX, PROJ=PROJ)
        for k, (a, b) in enumerate(zip(np.atleast_1d(lo), np.atleast_1d(la))):
            rows.append(
                {
                    "시각": r["시각"],
                    "시": r["시각"].hour,
                    "격자번호": r["격자번호"][k],
                    "경도": round(float(a), 4),
                    "위도": round(float(b), 4),
                }
            )
    return pd.DataFrame(
        rows, columns=None if rows else ["시각", "시", "격자번호", "경도", "위도"]
    )
def repeat_table(fp):
    t = (
        fp.groupby(["격자번호", "경도", "위도"])
        .agg(
            잡힌횟수=("시각", "size"),
```

```
        처음=("시각", "min"),
        마지막=("시각", "max"),
    )
    .reset_index()
    .sort_values("잡힌횟수", ascending=False)
)
t["첫·마지막 탐지 시각차(h)"] = (
    t["마지막"] - t["처음"]
).dt.total_seconds() / 3600
return t
```

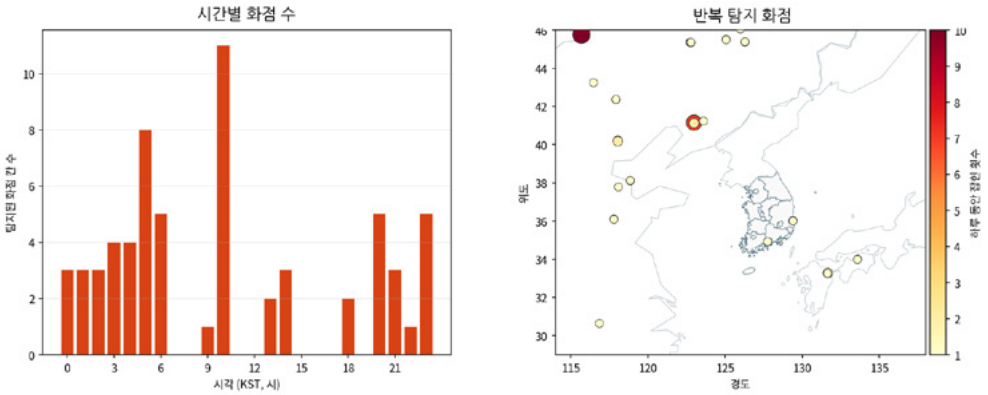


그림 12-2-4. 시간별 화점 탐지 위치와 횟수

격자번호	경도	위도	잡힌횟수	처음	마지막	이어진 시간(h)
5464	115.692	45.7667	10	1	23	23
248727	122.99	41.1618	7	0	23	24
249627	122.991	41.1433	7	0	23	24
45326	122.738	45.3628	1	10	10	1
47260	126.292	45.3718	1	10	10	1
46360	126.292	45.3904	1	10	10	1

정지궤도 기상위성

❻ 탐지 화점의 산불위험도 비교

```
def match_frame(fr, rt, *, MISS, NX):
    """탐지 화점 위치의 산불위험도를 추출하여 전체 격자의 분포와 비교합니다."""

    a = fr.copy()
    a[a <= MISS + 1] = np.nan
    rows = []
    for _, r in rt.iterrows():
        j, i = (int(r["격자번호"]) // NX, int(r["격자번호"]) % NX)
        rows.append(
            {
                "격자번호": int(r["격자번호"]),
                "경도": r["경도"],
                "위도": r["위도"],
                "잡힌횟수": int(r["잡힌횟수"]),
                "그 칸의 위험도": (
                    float(a[j, i]) if np.isfinite(a[j, i]) else np.nan
                ),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["격자번호", "경도", "위도", "잡힌횟수", "그 칸의 위험도"]
        ),
    )
```

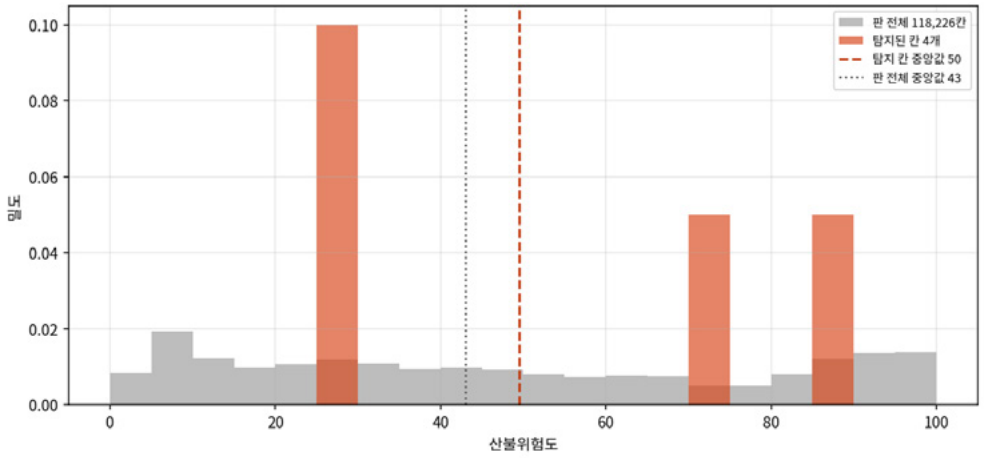


그림 12-2-5. 탐지 화점의 산불위험도 비교

격자번호	경도	위도	잡힌횟수	그 칸의 위험도
5464	115.692	45.7667	10	
248727	122.99	41.1618	7	
249627	122.991	41.1433	7	
45326	122.738	45.3628	1	
47260	126.292	45.3718	1	
46360	126.292	45.3904	1	

13.
천리안위성 2A호(GK2A)

우주 입자와 자기장 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 위성 > 천리안 2A호

❶ 우주기상 파일 조회와 읽기

```
# 1) 파일 목록 조회 → 응답 예시
https://apihub.kma.go.kr/api/typ05/api/GK2A/LV1/PD-E-1M/NA/dataList?sDate=202609010000&eDa
te=202609012359&authKey=...
{"list":[{"item":"202609010000"}]}
# 2) 목록의 파일 시각으로 다운로드 → 응답 예시
https://apihub.kma.go.kr/api/typ05/api/GK2A/LV1/PD-E-1M/NA/data?date=202609010000&authK
ey=...
https://apihub.kma.go.kr/api/typ05/api/GK2A/LV1/MG-1M/NA/data?date=202609010000&authK
ey=...
# HTTP 200, Content-Type: application/x-netcdf (NetCDF4/HDF5), 하루 한 개 파일
#  Time_Tag  (1440, 1)  2000-01-01 12:00 UTC 로부터의 초
#  E1 … E10  (1440, 1)  채널별 플럭스, 개 cm^-2 sr^-1 s^-1 keV^-1
#  E1_QEF …          품질 깃발 (0 정상)
#  Hp, He, Hn (1440, 1) 자기장 세 성분 (nT)
#  CM       (1440, 1) 위성체 내부대전량 (pA/cm^2)
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LV1_URL = "https://apihub.kma.go.kr/api/typ05/api/GK2A/LV1"
import h5py
DATE = "202609010000" # 재현 예시 시각; 다른 날짜는 list_lv1()로 목록부터 확인
MISS = -99999.0
EPOCH = datetime(2000, 1, 1, 12, tzinfo=timezone.utc)
LV1_URL = "https://apihub.kma.go.kr/api/typ05/api/GK2A/LV1"
FILES = {"PD-E-1M":"electron.nc", "PD-P-1M":"proton.nc", "MG-1M":"magnetic.nc", "CM-
1M":"charging.nc"}
CH = ["E" + str(i) for i in range(1, 11)]
E_BAND = {"E1":(100,150),"E2":(150,225),"E3":(225,325),"E4":(325,450),"E5":(450,700),"E6":(700,1350),"
E7":(1350,1800),"E8":(1800,2600),"E9":(2600,3800),"E10":(2000,3800)}  # 전자 파일 실제 속성값
def get_lv1(kind, date):
 """date는 list_lv1()로 받은 파일 시각(YYYYMMDDHHMM). NetCDF 바이트를 반환한다."""
 response = requests.get(
  f"{LV1_URL}/{kind}/NA/data",
  params={"date": date, "authKey": API_KEY},
  timeout=300,
 )
 response.raise_for_status()
 content = response.content
 # NetCDF4(HDF5) 또는 NetCDF3 시그니처로 파일 형식 확인
 assert content.startswith((b"\x89HDF\r\n\x1a\n", b"CDF")), "NetCDF 형식"
 return content
```

정지궤도 기상위성

```
def list_lv1(kind, s, e):
 """sDate~eDate 사이 파일 시각 목록을 정렬해 반환한다. 응답: {"list":[{"item":...}]}"""
 response = requests.get(
  f"{LV1_URL}/{kind}/NA/dataList",
  params={"sDate": s, "eDate": e, "authKey": API_KEY},
  timeout=120,
 )
 response.raise_for_status()
 payload = response.json()
 return sorted({str(x["item"]) for x in payload.get("list", [])})

def read_h5(path):
  """차원 척도만 제외하고 실제 자기장 MAG_x·MAG_y를 보존한다."""
  import h5py


  def decode(value):
    return (
      value.decode("utf-8", "replace")
      if isinstance(value, bytes)
      else value

    )

  with h5py.File(path, "r") as f:
    meta = {k: decode(v) for k, v in f.attrs.items()}
    missing = float(meta["Definition of data missing"])
    d = {}
    for k, obj in f.items():
      if not isinstance(obj, h5py.Dataset):
        continue
      if decode(obj.attrs.get("CLASS", "")) == "DIMENSION_SCALE":
        continue
      values = np.asarray(obj, dtype=float).ravel()
      d[k] = np.where(values == missing, np.nan, values)
  epoch = datetime(2000, 1, 1, 12, tzinfo=timezone.utc)
  d["시각"] = np.array(
    [
      epoch + timedelta(seconds=float(s)) if np.isfinite(s) else pd.NaT
      for s in d["Time_Tag"]
    ],
    dtype=object,
  )
  for k in list(d):
    q = d.get(k + "_QEF")
    if q is not None:
      d[k] = np.where(q == 0, d[k], np.nan)
  return (d, meta)
def read_all(base):
  out = {}
  for k, fn in FILES.items():
    out[k] = read_h5(Path(base, fn))
  return out
```

❷ 일별 파일의 1분 자료 구조

채널의 에너지 경계와 플래그 설명은 파일 속성에 포함됩니다.

```
def frame(d, bands):
  channels = [c for c in bands if c in d]
  t = pd.DataFrame({c: d[c] for c in channels})
  t.insert(0, "시각", d["시각"])
  t["시(UTC)"] = [
    x.hour + x.minute / 60 if not pd.isna(x) else np.nan for x in d["시각"]
  ]
  return t
def meta_table(metas):
  rows = []
  for k, (_, m) in metas.items():
    rows.append(
      {
        "자료": k,
        "센서": m.get("Sensor", ""),
        "설명": m.get("Data description", ""),
        "단위": m.get("Units", ""),
        "파일명": m.get("File_Name", ""),
        "결측 표시": m.get("Definition of data missing", ""),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["자료", "센서", "설명", "단위", "파일명", "결측 표시"]
    ),
  )
def band_table(bands, d, title):
  rows = []
  for c, (lo, hi) in bands.items():
    if c not in d:
      continue
    v = d[c][np.isfinite(d[c])]
    q = d.get(c + "_QEF")
    rows.append(
      {
        "채널": c,
        "에너지대(keV)": f"{lo}–{hi}",
        "가운데(keV)": int(np.sqrt(lo * hi)),
        "최소": v.min() if v.size else np.nan,
        "중앙값": np.median(v) if v.size else np.nan,
        "최대": v.max() if v.size else np.nan,
        "품질 깃발 0 아닌 줄": (
          int(np.count_nonzero(np.isfinite(q) & (q != 0)))
          if q is not None
          else np.nan
        ),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "채널",
        "에너지대(keV)",
        "가운데(keV)",
        "최소",
        "중앙값",
        "최대",
        "품질 깃발 0 아닌 줄",
      ]
    ),
  )
```

정지궤도 기상위성

▸ 실행 결과

자료	센서	설명	단위	파일명	결측 표시
PD-E-1M	Particle Detector	1-minute averaging electron flux data	Particles cm-2 sr-1 s-1 keV-1	gk2a_ksem_pd_e_1m_le1_20260901.nc	-99999
PD-P-1M	Particle Detector	1-minute averaging proton flux data	Particles cm-2 sr-1 s-1 keV-1	gk2a_ksem_pd_p_1m_le1_20260901.nc	-99999
MG-1M	Magnetometer	1-minute averaging magnetic field data	nT	gk2a_ksem_mg_auto_1m_le1_20260901.nc	-99999
CM-1M	Charging Monitor	1-minute averaging current density data	pA cm-2	gk2a_ksem_cm_1m_le1_20260901.nc	-99999

채널	에너지대 (keV)	가운데 (keV)	최소	중앙값	최대	품질 깃발 0 아닌 줄
E1	100–150	122	1.89E+04	2.77E+04	6.42E+04	0
E2	150–225	183	1.74E+03	3.53E+03	5.87E+03	0
E3	225–325	270	1.67E+03	3.40E+03	4.63E+03	0
E4	325–450	382	651	1.32E+03	1.81E+03	0
E5	450–700	561	393	832	1.18E+03	0
E6	700–1350	972	73.8	169	230	0
E7	1350–1800	1558	9.03	23.9	35.9	0
E8	1800–2600	2163	1.62	3.47	5.9	0
E9	2600–3800	3143	0.348	1.22	3.25	0

❸ 에너지대별 플럭스 스펙트럼

```
def spectrum(d, bands):
    """채널별 대표 에너지와 플럭스의 일 중앙값으로 스펙트럼을 작성합니다. 로그 좌표의 기울기는 해당 구간의 변화 특성을 나타냅니다."""

    x = np.array([np.sqrt(bands[c][0] * bands[c][1]) for c in CH])
    y = np.array([np.nanmedian(d[c]) for c in CH])
    m = np.isfinite(y) & (y > 0)
    a, b = np.polyfit(np.log10(x[m]), np.log10(y[m]), 1)
    return (x, y, a, b)
```

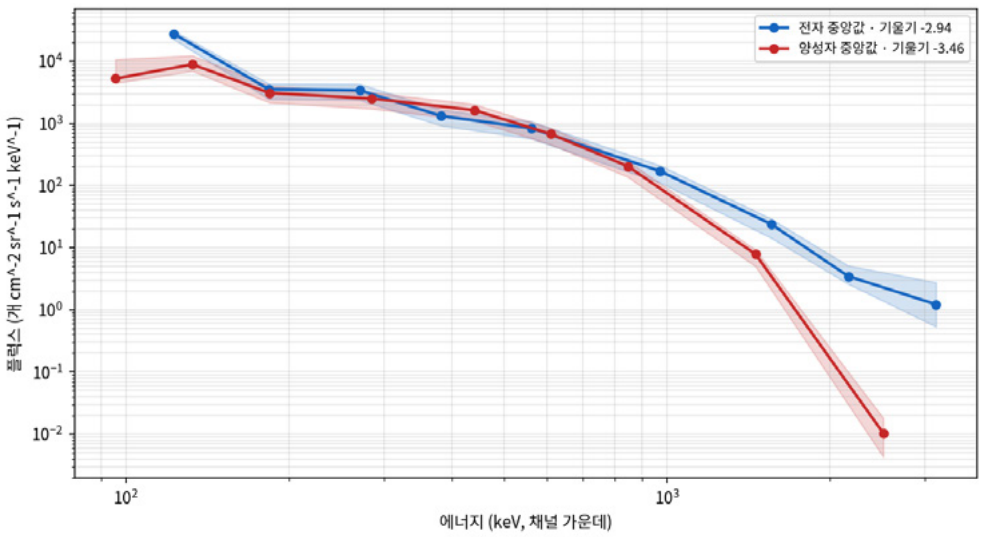


그림 13-1. 하늘 상태별 K지수와 CAPE의 상관 구조

채널	에너지대 (keV)	가운데 (keV)	최소	중앙값	최대	품질 깃발 0 아닌 줄
E1	77–119	95	3.73E+03	5.22E+03	1.13E+04	0
E2	119–148	132	6.17E+03	8.84E+03	1.29E+04	0
E3	148–229	184	1.61E+03	3.09E+03	4.30E+03	0
E4	229–354	284	1.26E+03	2.50E+03	3.33E+03	411
E5	354–548	440	785	1.63E+03	2.17E+03	1440
E6	548–681	610	318	670	898	1440
E7	681–1052	846	94.1	204	273	1440
E8	1052–2021	1458	3.25	7.87	11.1	1440
E9	2021–3123	2512	0	0.0104	0.0315	616

정지궤도 기상위성

❹ 시간·에너지대별 플럭스 변화

시간과 에너지를 두 축으로 표시해 시간대별로 강해지는 에너지 구간을 확인합니다.

```
def diurnal(d, ch="E1"):
  t = pd.DataFrame({"시각": d["시각"], "값": d[ch]})
  t["시"] = [x.hour for x in t["시각"]]
  return t.groupby("시")["값"].median()
def daily_table(ks):
  rows = []
  for kind, nm in [("PD-E-1M", "전자"), ("PD-P-1M", "양성자")]:
    d = ks[kind][0]
    for c in ("E1", "E3", "E5", "E7", "E9"):
      g = diurnal(d, c)
      rows.append(
        {
          "입자": nm,
          "채널": c,
          "가장 센 시(UTC)": int(g.idxmax()),
          "가장 약한 시(UTC)": int(g.idxmin()),
          "최대/최소": round(float(g.max() / max(g.min(), 1e-09)), 1),
          "하루 중앙값": f"{np.nanmedian(d[c]):.3g}",
        }
      )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "입자",
        "채널",
        "가장 센 시(UTC)",
        "가장 약한 시(UTC)",
        "최대/최소",
        "하루 중앙값",
      ]
    ),
  )
```

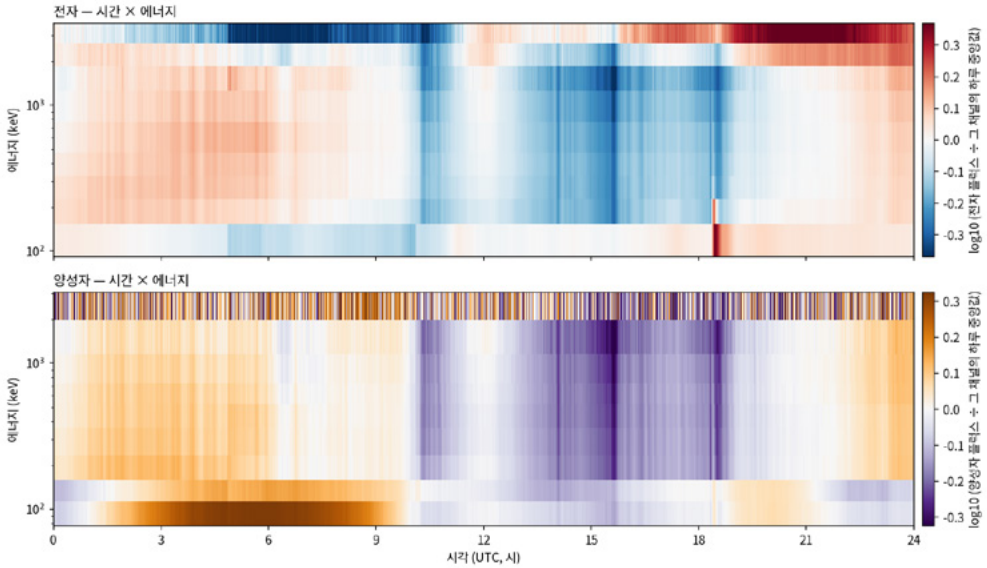


그림 13-2. 하늘 상태별 K지수와 CAPE의 상관 구조

입자	채널	가장 센 시(UTC)	가장 약한 시(UTC)	최대/최소	하루 중앙값	품질 깃발 0 아닌 줄
전자	E1	18	5	1.6	2.77E+04	0
전자	E3	3	15	2.1	3.40E+03	0
전자	E5	5	15	2.2	832	0
전자	E7	23	15	2.6	23.9	411
전자	E9	21	6	6.3	1.22	1440
양성자	E1	5	15	2.7	5.22E+03	1440
양성자	E3	3	15	2.1	3.09E+03	1440
양성자	E5	23	15	2.1	1.63E+03	1440
양성자	E7	23	15	2.2	204	616
양성자	E9	8	18	2.4	0.0104	0.0104

❺ 채널별 전자·양성자 플럭스 비교

채널별 비율로 상대적 크기를 비교하고, 로그 변환 후 상관으로 시간 변화의 유사성을 비교합니다.

```
def pair_frame(ks):
    e = ks["PD-E-1M"][0]
    p = ks["PD-P-1M"][0]
    rows = []
    for c in CH:
        ev, pv = (e[c], p[c])
        m = np.isfinite(ev) & np.isfinite(pv) & (pv > 0)
        rows.append(
            {
                "채널": c,
                "전자 중앙값": float(np.nanmedian(ev)),
                "양성자 중앙값": float(np.nanmedian(pv)),
                "전자/양성자": round(
                    float(np.nanmedian(ev) / max(np.nanmedian(pv), 1e-12)), 2
                ),
                "상관": round(
                    float(np.corrcoef(np.log10(ev[m]), np.log10(pv[m]))[0, 1]), 2
                ),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["채널", "전자 중앙값", "양성자 중앙값", "전자/양성자", "상관"]
        ),
    )
```

정지궤도 기상위성

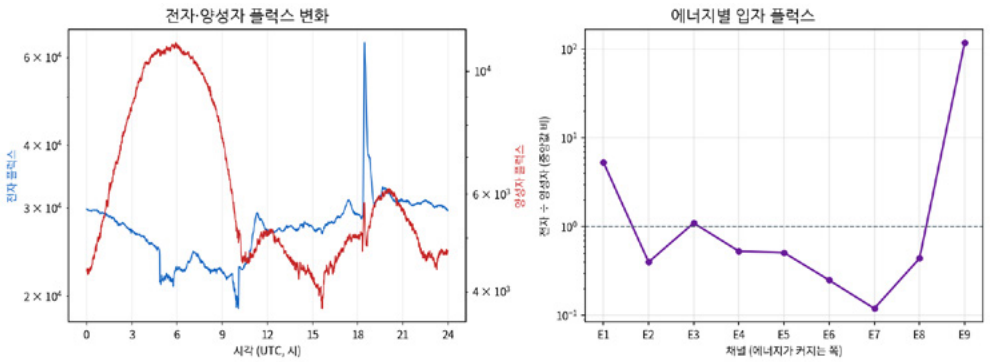


그림 13-3. 채널별 전자·양성자 플럭스 비교

채널	전자중앙값	양성자중앙값	전자/양성자	상관
E1	2.77E+04	5.22E+03	5.3	-0.52
E2	3.53E+03	8.84E+03	0.4	0.51
E3	3.40E+03	3.09E+03	1.1	0.99
E4	1.32E+03	2.50E+03	0.53	0.99
E5	832	1.63E+03	0.51	0.99
E6	169	670	0.25	0.99
E7	23.9	204	0.12	0.98
E8	3.47	7.87	0.44	0.73
E9	1.22	0.0104	117.97	-0.22

❻ 자기장 성분과 위성체 내부대전량 비교

총자기장을 계산해 하루 변화를 살펴보고, 같은 시각의 내부대전량과 고에너지 전자를 비교합니다.

```
def mag_frame(ks):
  m = ks["MG-1M"][0]
  t = pd.DataFrame(
    {"시각": m["시각"], "Hp": m["Hp"], "He": m["He"], "Hn": m["Hn"]}
  )
  t["총자기장"] = np.sqrt(t["Hp"] ** 2 + t["He"] ** 2 + t["Hn"] ** 2)
  t["시"] = [x.hour + x.minute / 60 for x in t["시각"]]
  return t
def mag_table(mf):
  rows = []
  for c in ("Hp", "He", "Hn", "총자기장"):
    v = mf[c].dropna()
    rows.append(
      {
        "성분": c,
        "최소(nT)": round(float(v.min()), 1),
        "중앙값(nT)": round(float(v.median()), 1),
        "최대(nT)": round(float(v.max()), 1),
        "하루 폭(nT)": round(float(v.max() - v.min()), 1),
        "가장 큰 시(UTC)": round(float(mf.loc[mf[c].idxmax(), "시"]), 1),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "성분",
        "최소(nT)",
        "중앙값(nT)",
        "최대(nT)",
        "하루 폭(nT)",
        "가장 큰 시(UTC)",
      ]
    ),
  )
def charge_table(ks):
  c = ks["CM-1M"][0]
  e = ks["PD-E-1M"][0]
  rows = []
  for ch in ("E5", "E6", "E7", "E8", "E9"):
    v, w = (c["CM"], e[ch])
    m = np.isfinite(v) & np.isfinite(w) & (w > 0)
    rows.append(
      {
        "전자 채널": ch,
        "에너지대(keV)": f"{E_BAND[ch][0]}–{E_BAND[ch][1]}",
        "대전량과의 상관": round(
          float(np.corrcoef(v[m], np.log10(w[m]))[0, 1]), 2
        ),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None if rows else ["전자 채널", "에너지대(keV)", "대전량과의 상관"]
    ),
  )
```

정지궤도 기상위성

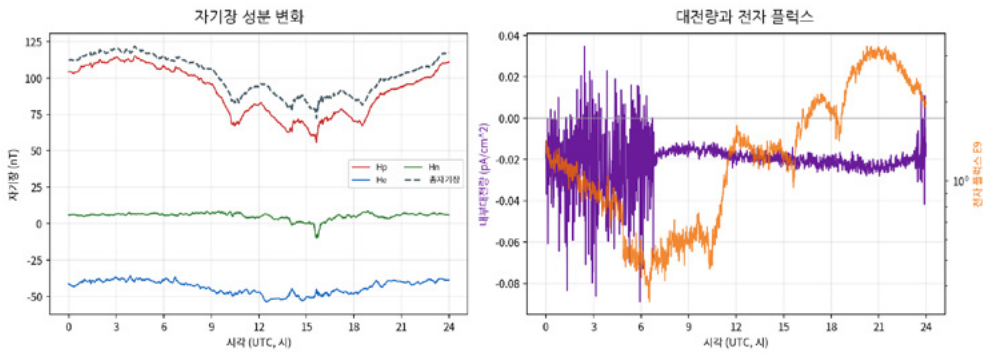


그림 13-4. 자기장 성분과 위성체 내부대전량 비교

성분	최소(nT)	중앙값(nT)	최대(nT)	하루 폭(nT)	가장 큰 시 UTC)
Hp	56	94.3	115	59	4.2
He	-53.7	-42.3	-35.9	17.8	3.9
Hn	-9.8	6	8.7	18.5	18.9
총자기장	72.4	103.7	121.8	49.4	4.1
E9	1.22	0.0104	117.97		-0.22

전자 채널	에너지대(keV)	대전량과의 상관
E5	450–700	-0.09
E6	700–1350	-0.08
E7	1350–1800	-0.07
E8	1800–2600	-0.1
E9	2600–3800	-0.08

데이터 활용사례 50

레이더

Radar



사이트

14.
레이더 원시자료(UF)
관측 범위와 빔고도 확인

데이터 활용 경로(API)
기상청 API허브 > 레이더 > 레이더 원시자료

❶ 파일 목록 조회

```
GET /api/typ01/url/rdr_file_list.php?rdr=UF&qcd=0&tm=20260910&authKey=...
# 실호출 응답(2026-11-08 확인) (일부)
RDR_DJK_202609100000.uf,  966622,=
RDR_DJK_202609100001.uf,  996616,=
RDR_DJK_202609100002.uf,  1005253,=
...
# 하루 전체: 4,133개 파일, 4.89GB (3개 지점 합계)
```

필드	값	의미
rdr	UF	자료 종류(원시자료)
qcd	0	QC 여부(0=NoQC, 원본 그대로)
tm	20260910	조회 대상일(년월일)
파일명 패턴	RDR_{지점코드}_{년월일시분}.uf	지점코드는 문서에 나열된 관측소 코드(KWK 등)와 별개로, 유관기관 포함 실제 운영 지점코드 사용
filesize	약 0.9 ~ 1.2MB (DJK 기준)	스윕 1건(약 1분 간격) 당 파일 크기, 지점·관측모드별 상이

❷ 지점별 파일 수와 용량

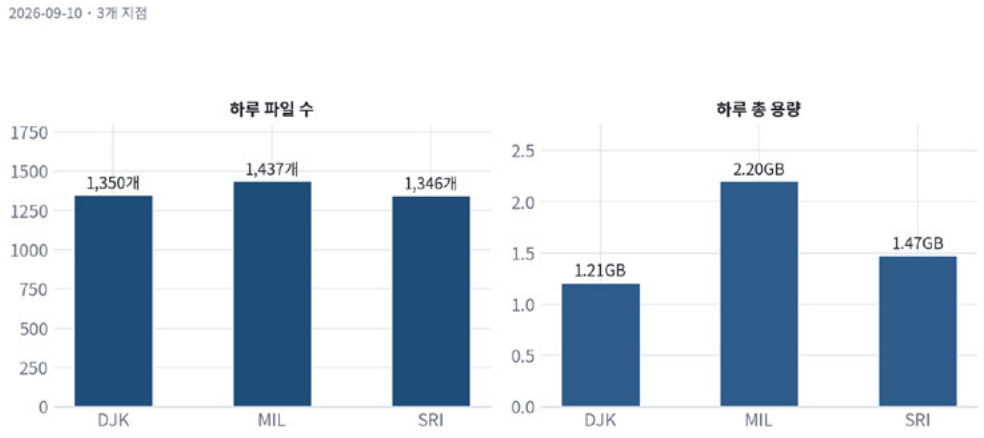


그림 14-1. 지점별 파일 수와 용량

❸ 수집 주기와 저장 용량 산정

지점	하루 파일 수	평균 생성 간격	하루 용량	30일 환산(개략)
DJK	1,350개	약 1.1분	1.21GB	약 36GB
MIL	1,437개	약 1.0분	2.20GB	약 66GB
SRI	1,346개	약 1.1분	1.47GB	약 44GB

※ 계절·강수 상황에 따라 실제 용량은 달라질 수 있습니다.

❹ 파일 헤더의 관측 구조 확인

```
GET /api/typ01/cgi-bin/url/nph-rdr_uf_inf?tm=202609100005&stn=DJK&qcd=0&help=1&authKey=...
# 실호출 응답(2026-11-08 확인) (헤더 일부, EUC-KR)
filename : /DATA/RDR/RAW/202609/10/RDR_DJK_202609100005.uf
filesize : 955309
time : 2026.09.10.09.05.07 (KST)
nvolumes : 45
latitude : 37.15.29 longitude : 126.05.54 height : 116
#Vn nsw sw evel nray nbin gate bin1 width nyq_vel
 0 3 0 1.5000 360 600 150 75 1.266 0.000
 0 3 1 2.1406 360 600 150 75 1.266 0.000
 0 3 2 15.0000 360 600 150 75 1.266 0.000
```

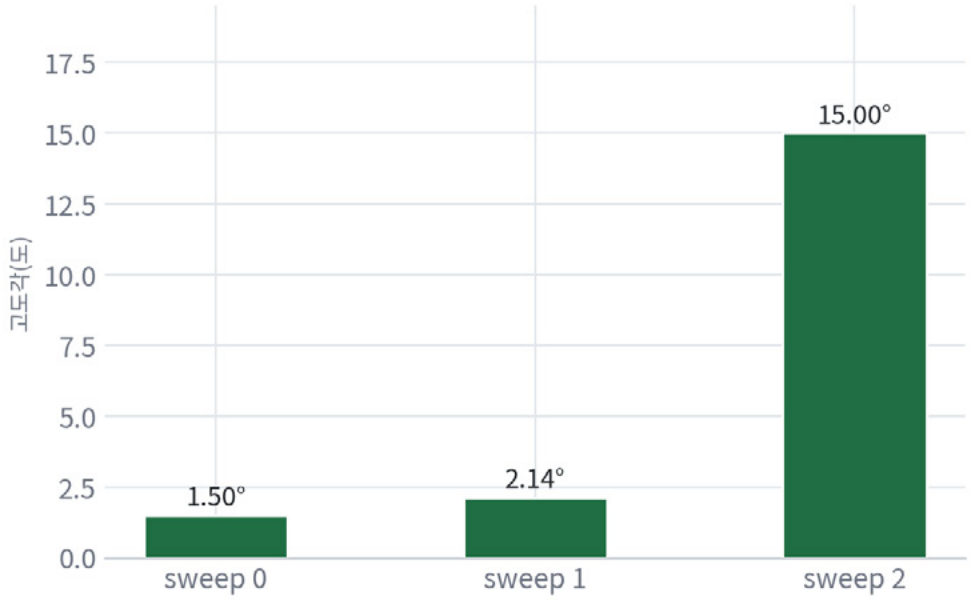


그림 14-2. 파일 헤더의 관측 구조 확인

사이트

❺ 파일 목록 조회 후 필요한 자료 수집

```
RADAR_STATIONS = ["DJK", "MIL", "SRI"]  # 유관기관 포함 실제 운영 지점코드
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LIST_URL = "https://apihub.kma.go.kr/api/typ01/url/rdr_file_list.php"
DOWNLOAD_URL = "https://apihub.kma.go.kr/api/typ01/url/rdr_file_down.php"
```

속도차 분석은 nyq_vel>0 자료를 선택한 뒤 품질검사·접힘 보정이 끝난 속도장을 입력합니다. 좌표는 각 게이트의 실제 km 좌표입니다. 출력은 속도변화 후보이며 메소사이클론·회오리 판정이나 경보에 바로 사용하지 않습니다.

```
def list_then_fetch(
  day, target_hours, auth_key=None, gap_sec=5, directory="radar_uf"
  ):
  """하루 목록 한 번 조회→중복 제거→지점·시간 선별→바이트 저장."""
  key = API_KEY if auth_key is None else auth_key
  response = requests.get(
    LIST_URL,
    params={"rdr": "UF", "qcd": 0, "tm": day, "authKey": key},
    timeout=60,
  )
  response.raise_for_status()
  response.encoding = "euc-kr"
  target = Path(directory)
  target.mkdir(parents=True, exist_ok=True)
  saved, seen = ([], set())
  for match in re.finditer("RDR_([A-Za-z0-9]+)_(\\d{12})\\.uf", response.text):
    station, tm = (match.group(1), match.group(2))
    filename = match.group(0)
    if (
      filename in seen
      or station not in RADAR_STATIONS
      or int(tm[8:10]) not in target_hours
    ):
      continue
    seen.add(filename)
    r = requests.get(
      DOWNLOAD_URL,
      params={
        "rdr": "UF",
        "stn": station,
        "qcd": 0,
        "tm": tm,
        "authKey": key,
      },
      timeout=120,
    )
     r.raise_for_status()
    # UF 바이너리만 저장 (빈 응답·HTML 안내문은 건너뜀)
    if not r.content or "text/html" in r.headers.get("Content-Type", "").lower():
     continue
    path = target / filename
```

```
    path.write_bytes(r.content)
    saved.append(str(path))
    time.sleep(gap_sec)
  return saved
```

❻ 고도각과 거리에 따른 빔고도 비교

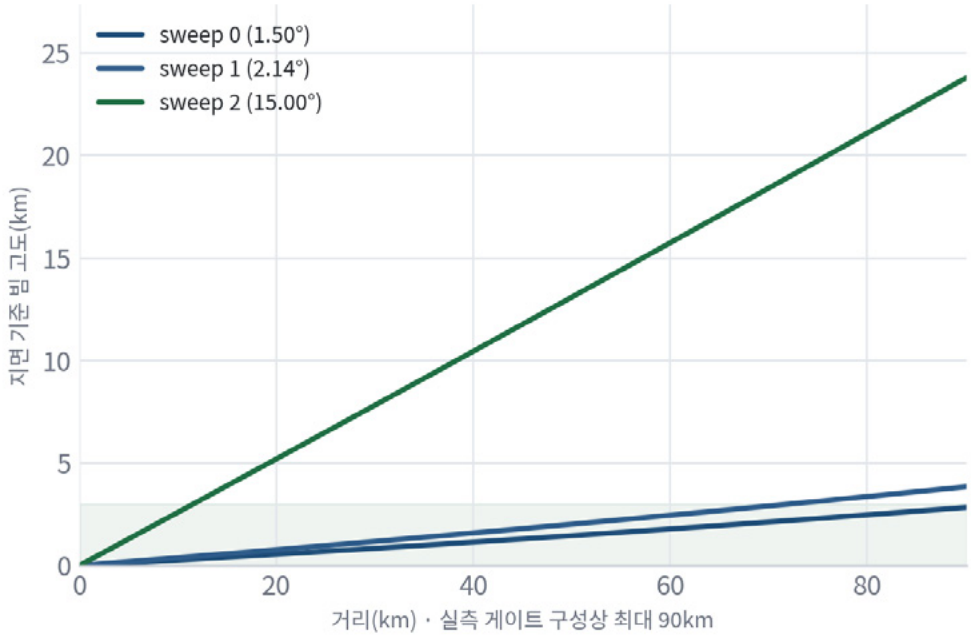


그림 14-3. 고도각과 거리에 따른 빔고도 비교

```
def select_precip_sweep(
  sweeps, range_m, radar_height_m=0, max_beam_height_km=3.0
  ):
  """고도각 열(evel)을 가진 스윕 표에서 해당 거리의 최저 빔고도 선택."""
  if range_m < 0:
    raise ValueError("거리 음수")
  radius = 6371000.0 * 4 / 3
  d = sweeps.copy().sort_values("evel")
  angle = np.radians(pd.to_numeric(d["evel"], errors="raise"))
  d["빔고도(m)"] = (
    np.sqrt(radius**2 + range_m**2 + 2 * radius * range_m * np.sin(angle))
    - radius
    + radar_height_m
  )
  usable = d[d["빔고도(m)"] <= max_beam_height_km * 1000]
  return usable.iloc[0] if not usable.empty else None
```

사이트

❼ 도플러 속도 자료의 선별과 해석

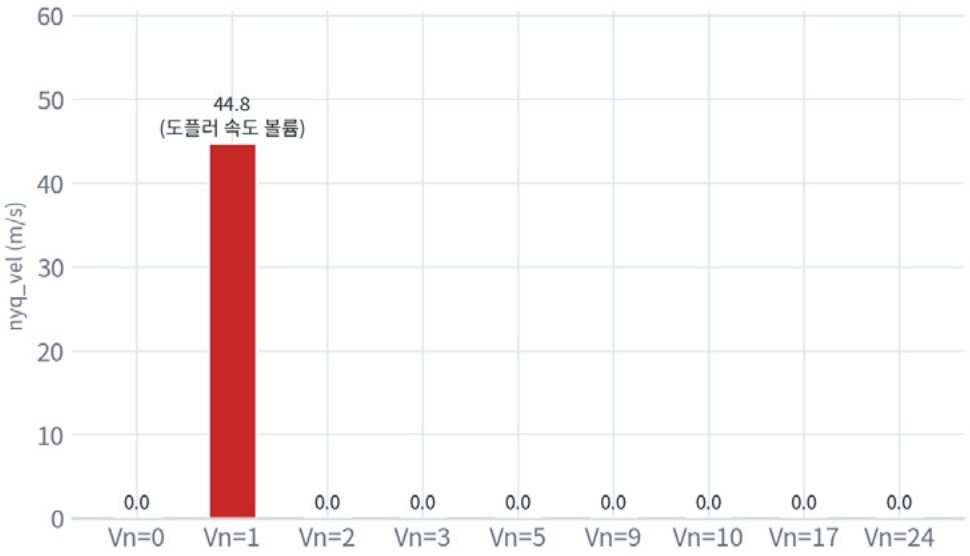


그림 14-4. 도플러 속도 자료의 선별과 해석

```
def select_doppler_volumes(volume_list):
  """헤더 목록(help=1) 조회 결과에서 nyq_vel > 0인 볼륨만 선별 — 다운로드 대상 축소"""
  return [v for v in volume_list if v["nyq_vel"] > 0]
def detect_shear_couplet(
  vel_field, x_km, y_km, velocity_dealiased, shear_thresh_mps_per_km=15.0
  ):
  """실제 인접 게이트 거리를 사용하는 국지 속도차 후보. 회전 판정이 아니다."""
  if not velocity_dealiased:
    raise ValueError("속도 접힘 보정·품질검사를 먼저 수행")
  velocity = np.asarray(vel_field, float)
  x, y = (np.asarray(x_km, float), np.asarray(y_km, float))
  if (
    velocity.ndim != 2
    or x.shape != velocity.shape
    or y.shape != velocity.shape
  ):
    raise ValueError("속도·좌표의 동일 2차원 모양 필요")
  candidate = np.zeros(velocity.shape, dtype=bool)
  for axis in [0, 1]:
    distance = np.hypot(np.diff(x, axis=axis), np.diff(y, axis=axis))
    delta = np.abs(np.diff(velocity, axis=axis))
    gradient = np.full(delta.shape, np.nan)
    np.divide(
      delta,
      distance,
      out=gradient,
      where=(distance > 0) & np.isfinite(delta),
    )
    high = gradient >= shear_thresh_mps_per_km
    if axis == 0:
      candidate[:-1, :] |= high
      candidate[1:, :] |= high
    else:
      candidate[:, :-1] |= high
      candidate[:, 1:] |= high
  return candidate
```

15.
레이더 합성장(HSR)
반사도·에코고도·강수량 비교

데이터 활용 경로(API)
기상청 API허브 > 레이더 > 레이더 강수량(HSR)

❶ 합성장 정보와 격자값 조회
한 시각의 반사도·에코고도·1시간 누적강수를 비교합니다.

```
https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-rdr_cmp_inf?tm=202609010600&cmp=HSR&qc
d=MSK&authKey=...
 nx = 2305 · ny = 2881 · dxy = 500 · map_code = 1
 num_stn = 17 · (0) stn_cd = KWK, tm = …, tm_in = …
https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-rdr_cmp1_api
 ?tm=202609010600&cmp=HSR&qcd=MSK&obs=ECHO&map=HB&disp=A&authKey=...
 2305, 2881,=
-30000,-30000,-30000,-30000,-30000,-30000,-30000,…
# 값은 dBZ × 100 · -30000 관측영역 밖 · -25000 에코 없음
# obs=EBH 로 바꾸면 그 값이 「에코가 온 높이(m)」가 된다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
INF_URL = "https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-rdr_cmp_inf"
VAL_URL = "https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-rdr_cmp1_api"
STN_KO = {} # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

DXY : 원본 격자간격(m)
MAP_CODE : 지도코드별 olat·olon(도), ix0·iy0(해당 파서의 0 또는 1 시작 번호 확인) dict
MISS_BELOW : 요소별 결측 처리경계 dict. 실제 의미 있는 음수값 제외 금지
NX : 열 수
NY : 행 수
RE_KM : 해당 투영 명세의 지구반경(km)
SLAT1 : 제1 표준위도(도)
SLAT2 : 제2 표준위도(도)
UNIT : 요소별 (단위 변환배율,단위 문자열) dict. 변환은 한 번만 적용
ZR_A : Z=aR^b의 a. 분석용 경험계수이며 공식 강수 산출식으로 단정 금지
ZR_B : Z=aR^b의 b. 분석용 경험계수

사이트

dxy_m : 헤더의 원본 격자간격(m)
필수 입력: DXY, MAP_CODE, MISS_BELOW, NX, NY, RE_KM, SLAT1, SLAT2, UNIT, ZR_A, ZR_B, dxy_m. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_inf(tm=None, cmp="HSR", qcd="MSK"):
  if tm is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: tm")
  "합성판의 정보. 격자 크기·간격·지도코드와 합성에 들어간 레이더 목록이\n    사람이 읽는 글로 온다 — 값을 받기 전에 이것부터 본다."
  r = requests.get(
    INF_URL,
    params={"tm": tm, "cmp": cmp, "qcd": qcd, "authKey": API_KEY},
    timeout=120,
  )
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  r.encoding = "euc-kr"
  return r.text

def get_values(tm=None, cmp="HSR", qcd="MSK", obs="ECHO", acc=None, map_="HB"):
  if tm is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: tm")
  "값 판. disp=A 면 쉼표로 이어진 글로 오고, 첫 줄이 격자 수다.\n    한 판이 2305 × 2881 이라 40 MB 를 넘으므로 일반 요청으로 받으므로 메모리 사용량을 확인한다."
  p = {
    "tm": tm,
    "cmp": cmp,
    "qcd": qcd,
    "obs": obs,
    "map": map_,
    "disp": "A",
    "authKey": API_KEY,
  }
  if acc:
    p["acc"] = acc
  r = requests.get(VAL_URL, params=p, timeout=600)
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  return r.text

def parse_values(text, nx=None, ny=None):
  if nx is None or ny is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: nx, ny")
  "첫 「=」 앞이 격자 수이고 그 뒤가 값이다.\n    왼쪽 아래에서 오른쪽 위로 가로 우선으로 이어져 있다."
  head, body = text.split("=", 1)
  n = [int(v) for v in re.findall("\\d+", head)]
  v = np.fromstring(body.replace("=", " ").replace("\n", " "), sep=",")
  return v[: n[0] * n[1]].reshape(n[1], n[0])
```

❷ 격자 정보와 참여 레이더 확인

```
def inf_table(inf):
  """정보 글에서 「이름 = 값」 꼴만 뽑아 표로 만든다."""
  rows = []
  for ln in inf.splitlines():
    m = re.match("\\s*(\\w+)\\s*=\\s*([^,]+)", ln)
    if m and (not ln.strip().startswith("(")):
      rows.append({"항목": m[1], "값": m[2].strip()})
  return pd.DataFrame(rows, columns=None if rows else ["항목", "값"])
def stn_table(inf):
  """합성에 들어간 레이더와 그 자료가 도착한 시각. 합성이 언제 완성되는지가 나반환됩니다."""
  rows = []
  for m in re.finditer(
    "\\(\\s*(\\d+)\\)\\s*stn_cd\\s*=\\s*(\\w+),\\s*tm\\s*=\\s*([\\d.:]+),\\s*tm_in\\s*=\\s*([\\d.:]+)",
    inf,
  ):
    t0 = datetime.strptime(m[3], "%Y.%m.%d.%H:%M:%S")
    t1 = datetime.strptime(m[4], "%Y.%m.%d.%H:%M:%S")
    rows.append(
      {
        "번호": int(m[1]),
        "레이더": m[2],
        "이름": STN_KO.get(m[2], m[2]),
        "관측시각": t0.strftime("%H:%M:%S"),
        "도착시각": t1.strftime("%H:%M:%S"),
        "걸린 시간(초)": int((t1 - t0).total_seconds()),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "번호",
        "레이더",
        "이름",
        "관측시각",
        "도착시각",
        "걸린 시간(초)",
      ]
    ),
  )
def lcc(dxy_m=None, map_code=1, *, MAP_CODE, RE_KM, SLAT1, SLAT2):
  if dxy_m is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: dxy_m")
  "람베르트 정각원추 도법 상수. 격자 간격은 정보 글의 dxy 를 그대로 쓴다."
  m = MAP_CODE[map_code]
  d = math.pi / 180.0
  grid = dxy_m / 1000.0
  sn = math.tan(math.pi * 0.25 + SLAT2 * d * 0.5) / math.tan(
    math.pi * 0.25 + SLAT1 * d * 0.5
  )
  sn = math.log(math.cos(SLAT1 * d) / math.cos(SLAT2 * d)) / math.log(sn)
  sf = math.tan(math.pi * 0.25 + SLAT1 * d * 0.5)
  sf = sf**sn * math.cos(SLAT1 * d) / sn
  ro = math.tan(math.pi * 0.25 + m["olat"] * d * 0.5)
  ro = RE_KM / grid * sf / ro**sn
  return (d, grid, sn, sf, ro, m)
def xy_to_lonlat(x, y, dxy_m=None, map_code=1, *, MAP_CODE, RE_KM, SLAT1, SLAT2):
  if dxy_m is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: dxy_m")
  "격자 번호를 위경도로. 기준격자점이 원점 위경도에 떨어지는지로 확인한다."
  d, grid, sn, sf, ro, m = lcc(
    dxy_m, map_code, MAP_CODE=MAP_CODE, RE_KM=RE_KM, SLAT1=SLAT1, SLAT2=SLAT2
  )
  X = np.asarray(x, dtype=float) - m["ix0"]
  Yr = ro - (np.asarray(y, dtype=float) - m["iy0"])
  ra = np.hypot(X, Yr)
  if sn < 0:
```

사이트

```
    ra = -ra
  alat = 2.0 * np.arctan((RE_KM / grid * sf / ra) ** (1.0 / sn)) - math.pi * 0.5
  alon = np.arctan2(X, Yr) / sn + m["olon"] * d
  return (alon / d, alat / d)
def corner_table(*, MAP_CODE, NX, NY, RE_KM, SLAT1, SLAT2, dxy_m):
  ix0, iy0 = (MAP_CODE[1]["ix0"], MAP_CODE[1]["iy0"])
  pts = [
    ("(1, 1) 왼쪽 아래", 0, 0),
    (f"({NX}, 1) 오른쪽 아래", NX - 1, 0),
    (f"(1, {NY}) 왼쪽 위", 0, NY - 1),
    (f"({NX}, {NY}) 오른쪽 위", NX - 1, NY - 1),
    (f"기준격자점 ({ix0}, {iy0})", ix0, iy0),
  ]
  rows = []
  for nm, x, y in pts:
    lo, la = xy_to_lonlat(
      x,
      y,
      dxy_m=dxy_m,
      MAP_CODE=MAP_CODE,
      RE_KM=RE_KM,
      SLAT1=SLAT1,
      SLAT2=SLAT2,
    )
    rows.append(
      {
        "격자점": nm,
        "경도": round(float(lo), 4),
        "위도": round(float(la), 4),
      }
    )
  return pd.DataFrame(
    rows, columns=None if rows else ["격자점", "경도", "위도"]
  )
```

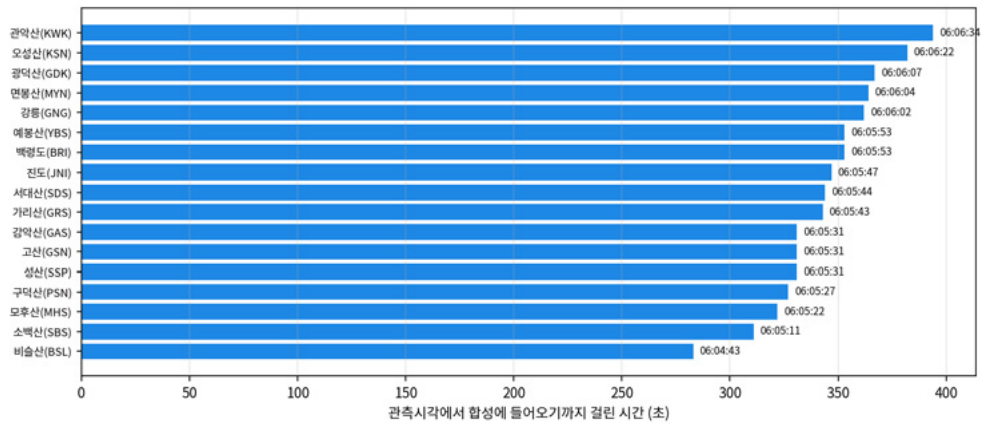


그림 15-1. 격자 정보와 참여 레이더 확인

항목	값
file	/DATA/RDR/CMP/202609/01/RDR_CMP_HSR_EXT_202609010600.bin.gz
version	1
ptype	5
tm	2026.09.01.06:00:00
tm_in	2026.09.01.06:06:52
num_stn	17
map_code	1
map_etc	0
nx	2305
ny	2881
nz	1
dxy	500
dz	0
z_min	0

격자점	경도	위도
DJK	1,350개	약 36GB
(1, 1) 왼쪽 아래	120.162	30.1419
(2305, 1) 오른쪽 아래	132.159	30.1211
(1, 2881) 왼쪽 위	118.821	43.3273
(2305, 2881) 오른쪽 위	133.574	43.3007
기준격자점 (1121, 1681)	126	38

레이더	이름	관측시각	도착시각	걸린 시간(초)
KWK	관악산	6:00:00	6:06:34	394
GDK	광덕산	6:00:00	6:06:07	367
GNG	강릉	6:00:00	6:06:02	362
MHS	모후산	6:00:00	6:05:22	322
BSL	비슬산	6:00:00	6:04:43	283
YBS	예봉산	6:00:00	6:05:53	353

사이트

❸ 격자값을 활용한 반사도 지도

```
def rle_decode(s):
  """횟수*값 형식의 압축 배열을 복원합니다. o는 관측영역 밖, n은 영역 내 무에코 값입니다."""

  out = []
  for tok in s.split():
    c, _, t = tok.partition("*")
    if not t:
      c, t = (1, c)
    v = np.nan if t == "o" else -np.inf if t == "n" else float(t) / 10.0
    out += [v] * int(c)
  return np.array(out, dtype=float)
def to_value(name, a, *, MISS_BELOW, UNIT):
  """자료 종류별 결측 코드를 처리하고 물리값과 단위로 변환합니다."""

  v = np.asarray(a, dtype=float).copy()
  v[np.isneginf(v)] = np.nan
  v[v <= MISS_BELOW[name]] = np.nan
  return v * UNIT[name][0]
def masks(name, a, *, MISS_BELOW):
  """세 가지를 가른다 — 관측영역 밖, 영역 안이지만 값 없음, 값 있음."""
  out = np.isnan(a)
  null = np.isneginf(a) | ~out & (a <= MISS_BELOW[name])
  return (out, null, ~(out | null))
def axes_of(
  nx, ny, step=None, x0=0, y0=0, *, MAP_CODE, RE_KM, SLAT1, SLAT2, dxy_m
  ):
  if step is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: step")
  X = x0 + np.arange(nx) * step
  Y = y0 + np.arange(ny) * step
  XX, YY = np.meshgrid(X, Y)
  return xy_to_lonlat(
    XX,
    YY,
    dxy_m=dxy_m,
    MAP_CODE=MAP_CODE,
    RE_KM=RE_KM,
    SLAT1=SLAT1,
    SLAT2=SLAT2,
  )
def value_table(fields, *, MISS_BELOW, UNIT):
  rows = []
  for nm, a in fields.items():
    o, n, v = masks(nm, a, MISS_BELOW=MISS_BELOW)
    g = to_value(nm, a, MISS_BELOW=MISS_BELOW, UNIT=UNIT)
    rows.append(
      {
        "판": nm,
        "단위": UNIT[nm][1],
        "칸 수": a.size,
        "관측영역 밖(%)": round(100 * float(o.mean()), 1),
        "값 없음(%)": round(100 * float(n.mean()), 1),
        "값 있음(%)": round(100 * float(v.mean()), 1),
        "최소": round(float(np.nanmin(g)), 1),
        "중앙값": round(float(np.nanmedian(g)), 1),
        "최대": round(float(np.nanmax(g)), 1),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "판",
        "단위",
        "칸 수",
        "관측영역 밖(%)",
        "값 없음(%)",
```

```
            "값 있음(%)",
            "최소",
            "중앙값",
            "최대",
          ]
        ),
    )
```

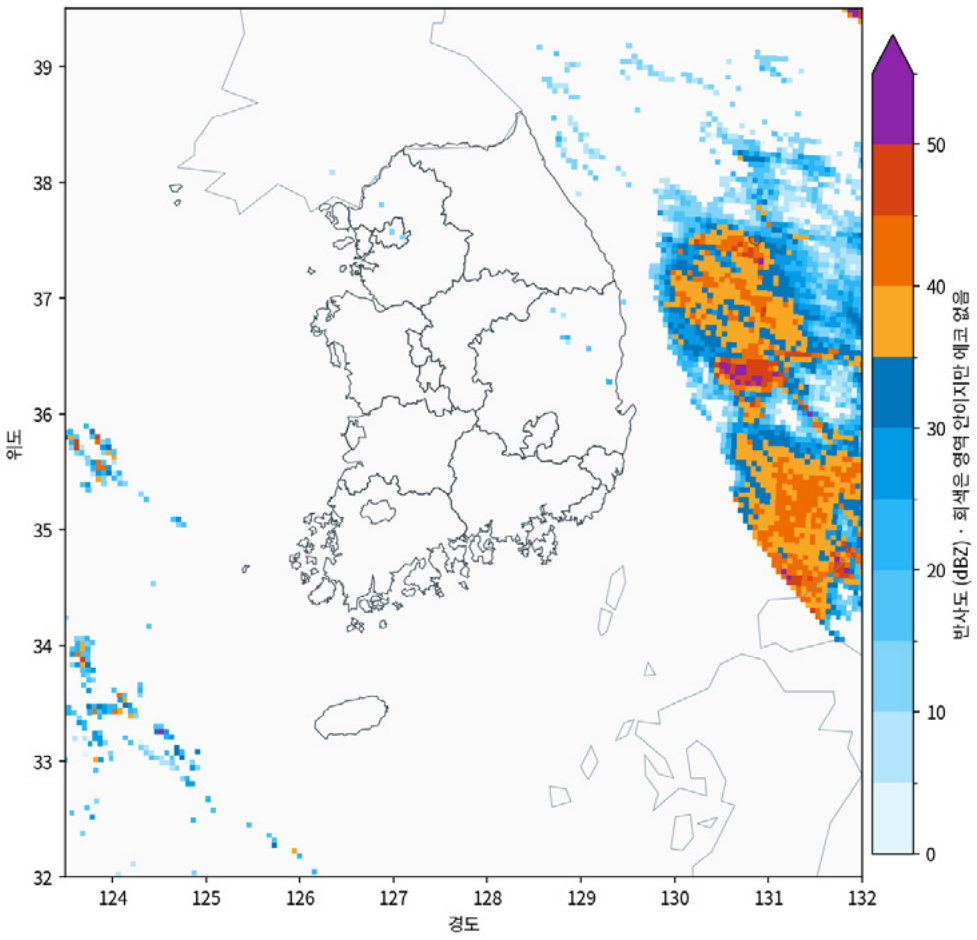


그림 15-2. 격자값을 활용한 반사도 지도

판	단위	칸 수	관측영역 밖(%)	값 없음 (%)	값 있음 (%)	최소	중앙값	최대
ECHO	dBZ	66759	59.6	33	7.4	-5.5	25	54.7
EBH	m	66759	0	59.6	40.4	0	1630	11440
PCP60	값(원값÷100)	66759	59.6	32.5	8	0	24.2	51.2

사이트

❹ 반사도·에코고도·누적강수 비교

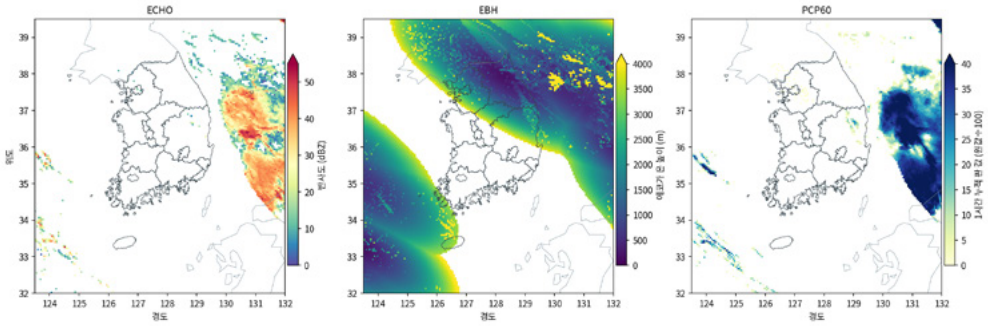


그림 15-3. 반사도·에코고도·누적강수 비교

구간	칸수	에코높이중앙	누적판값중앙	누적판값최대
~10	436	1630	15.5	38.6
10월 20일	1305	1490	18.6	47.2
20-25	487	1110	22.4	48.2
25-30	579	940	26.9	48.6
30-35	660	1015	31.3	47.4
35-40	731	1330	36.6	48.4
40+	480	1715	39.9	49.6

❺ 반사도·에코고도·누적강수의 상관 분석

```
def pair_frame(fields, *, MISS_BELOW, UNIT):
  """같은 격자의 반사도·에코고도·누적강수량을 연결하여 비교합니다."""

  e = to_value("ECHO", fields["ECHO"], MISS_BELOW=MISS_BELOW, UNIT=UNIT)
  h = to_value("EBH", fields["EBH"], MISS_BELOW=MISS_BELOW, UNIT=UNIT)
  p = to_value("PCP60", fields["PCP60"], MISS_BELOW=MISS_BELOW, UNIT=UNIT)
  m = np.isfinite(e)
  d = pd.DataFrame({"dBZ": e[m], "EBH": h[m], "PCP": p[m]})
  d["구간"] = pd.cut(
    d["dBZ"],
    [0, 10, 20, 25, 30, 35, 40, 60],
    labels=["~10", "10-20", "20-25", "25-30", "30-35", "35-40", "40+"],
  )
  t = (
    d.groupby("구간", observed=True)
    .agg(
      칸수=("dBZ", "size"),
      에코높이중앙=("EBH", "median"),
      누적판값중앙=("PCP", "median"),
      누적판값최대=("PCP", "max"),
    )
    .round(1)
  )
  return (d, t.reset_index())
def zr_rain(dbz, *, ZR_A, ZR_B):
  return (10.0 ** (np.asarray(dbz, dtype=float) / 10.0) / ZR_A) ** (1.0 / ZR_B)
```

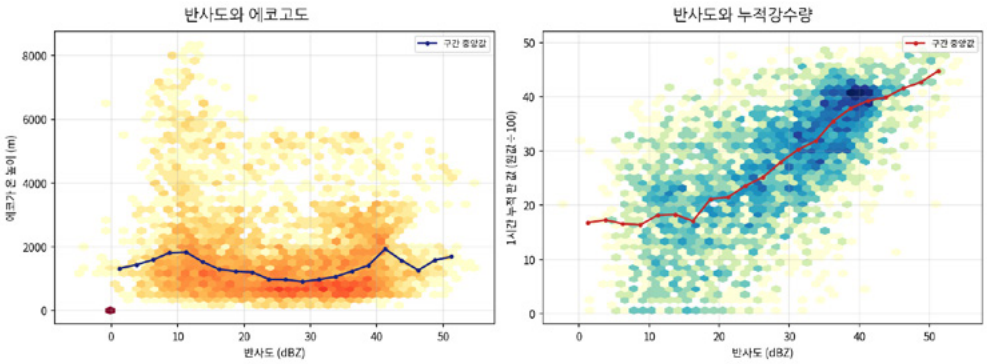


그림 15-4. 반사도·에코고도·누적강수의 상관 분석

❻ 500 m 해상도로 본 강수 영역의 단면

```
def win_lonlat(meta, *, MAP_CODE, RE_KM, SLAT1, SLAT2, dxy_m):
  X = meta["x0"] + np.arange(meta["nx"])
  Y = meta["y0"] + np.arange(meta["ny"])
  XX, YY = np.meshgrid(X, Y)
  return xy_to_lonlat(
    XX,
    YY,
    dxy_m=dxy_m,
    MAP_CODE=MAP_CODE,
    RE_KM=RE_KM,
    SLAT1=SLAT1,
    SLAT2=SLAT2,
  )
def cell_table(win, meta, *, DXY, MISS_BELOW, UNIT, ZR_A, ZR_B):
  """관심영역의 반사도 구간별 격자 수를 집계합니다. 500 m 격자의 면적은 격자 수×0.25 km²입니다."""
  a = to_value("ECHO", win, MISS_BELOW=MISS_BELOW, UNIT=UNIT)
  rows = []
  for lo, hi, nm in [
    (0, 15, "약한 에코"),
    (15, 25, "약한 비"),
    (25, 35, "보통 비"),
    (35, 45, "강한 비"),
    (45, 60, "매우 강한 비"),
  ]:
    m = (a >= lo) & (a < hi)
    rows.append(
      {
        "세기": f"{nm} ({lo}~{hi} dBZ)",
        "칸 수": int(m.sum()),
        "넓이(km²)": round(int(m.sum()) * (DXY / 1000) ** 2, 1),
        "강수강도(mm/h)": f"{zr_rain(lo, ZR_A=ZR_A, ZR_B=ZR_B):.1f} ~ {zr_rain(hi, ZR_A=ZR_A, ZR_B=ZR_
B):.0f}",
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None if rows else ["세기", "칸 수", "넓이(km²)", "강수강도(mm/h)"]
    ),
  )
```

사이트

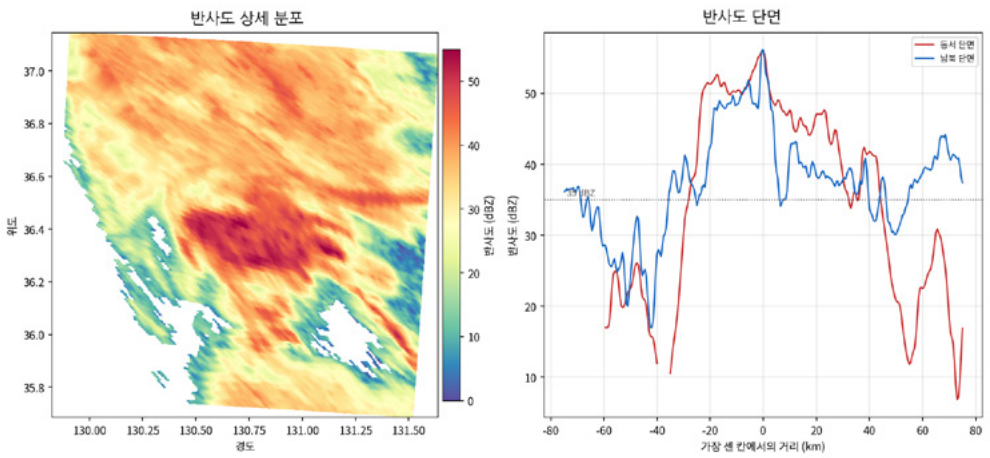


그림 15-5. 500 m 해상도로 본 강수 영역의 단면

세기	칸 수	넓이(km²)	강수강도(mm/h)
약한 에코 (0~15 dBZ)	8590	2147.5	0.0 ~ 0
약한 비 (15~25 dBZ)	9856	2464	0.3 ~ 1
보통 비 (25~35 dBZ)	24042	6010.5	1.3 ~ 6
강한 비 (35~45 dBZ)	27602	6900.5	5.6 ~ 24
매우 강한 비 (45~60 dBZ)	5177	1294.2	23.7 ~ 205

16.
레이더 강수량

격자별 강수강도 분석

데이터 활용 경로(API)

기상청 API허브 > 레이더 > 레이더 강수량

❶ 레이더 이진 파일 읽기

```
https://apihub.kma.go.kr/api/typ04/url/rdr_cmp_file.php?tm=202608310600&cmp=HSR&data=bin&
authKey=...
# gzip 을 풀면 1024바이트 머리글 + short int 배열
# RDR_CMP_HEAD(64) : version · ptype · 관측시각 · 생성시각 ·
#              레이더 수 · map_code · nx · ny · dxy · 블록 구성
# RDR_CMP_STN_LIST(20 × 48) : 합성에 들어간 레이더 코드
# 값은 dBZ × 100 · -25000 관측 없음 · -30000 반경 밖
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
RDR_URL = "https://apihub.kma.go.kr/api/typ04/url/rdr_cmp_file.php"
HEAD_BYTES = 64
STN_BYTES = 20
STN_COUNT = 48
DATA_CODE = {}
PTYPE = {}
```

DBZ_LEVELS : (하한dBZ,상한dBZ,표시명) 목록. 분석용 구간임을 명시
MAP_CODE : 지도코드별 olat·olon(도), ix0·iy0(해당 파서의 0 또는 1 시작 번호 확인) dict
RE_KM : 해당 투영 명세의 지구반경(km)
SLAT1 : 제1 표준위도(도)
SLAT2 : 제2 표준위도(도)
VAL_INSIDE_NULL : 해당 이진 산출물의 내부 결측 경계
a : Z=aR^b의 경험계수 a
b : Z=aR^b의 경험계수 b
필수 입력: DBZ_LEVELS, MAP_CODE, RE_KM, SLAT1, SLAT2, VAL_INSIDE_NULL, a, b. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

합성

```
def _tm(buf, offset):
  if len(buf) < offset + 7:
    raise ValueError("레이더 시간 헤더: offset부터 7바이트")
  year, month, day, hour, minute, second = struct.unpack_from(
    "<HBBBBB", buf, offset
  )
  return datetime(year, month, day, hour, minute, second, tzinfo=KST)
```

헤더 구조는 해당 합성자료의 포맷 명세와 일치하는 경우에 사용합니다. version·ptype·자료블록 코드가 다른 포맷은 같은 파서로 처리하지 않습니다. 실제 적용 전 help/헤더에서 포맷과 바이트 순서를 대조합니다.

```
def get_cmp(tm, cmp="HSR", retry=3):
  """지정한 시각의 gzip 압축 합성자료를 조회합니다. 이 예의 포맷은 앞 1024바이트 헤더와 뒤의 short int 자료블록으로 구성됩니다."""

  params = {"tm": tm, "cmp": cmp, "data": "bin", "authKey": API_KEY}
  for i in range(retry):
    try:
      r = requests.get(RDR_URL, params=params, timeout=300)
      r.raise_for_status()
      return (
        gzip.decompress(r.content)
        if r.content[:2] == b"\x1f\x8b"
        else r.content
      )
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{tm} 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(3 * (i + 1))
def read_header(buf):
  """RDR_CMP_HEAD 64바이트의 격자 크기·간격·지도코드를 읽습니다. 좌표 변환에는 해당 자료의 도법 명세도 확인합니다."""


  o = 0
  version = struct.unpack_from("<B", buf, o)[0]
  o += 1
  ptype = struct.unpack_from("<h", buf, o)[0]
  o += 2
  tm = _tm(buf, o)
  o += 7
  tm_in = _tm(buf, o)
  o += 7
  num_stn = buf[o]
  o += 1
  map_code = buf[o]
  o += 1
  map_etc = buf[o]
  o += 1
  nx, ny, nz = struct.unpack_from("<hhh", buf, o)
  o += 6
  dxy, dz, z_min = struct.unpack_from("<hhh", buf, o)
  o += 6
  num_data = buf[o]
  o += 1
  codes = list(struct.unpack_from("<16b", buf, o))
  return {
    "version": version,
    "ptype": ptype,
    "산출물": PTYPE.get(ptype, str(ptype)),
    "관측시각": tm,
    "생성시각": tm_in,
    "합성 레이더 수": num_stn,
    "map_code": map_code,
    "nx": nx,
```

```
    "ny": ny,
    "nz": nz,
    "dxy(m)": dxy,
    "자료블록 수": num_data,
    "블록 구성": [DATA_CODE.get(c, str(c)) for c in codes[:num_data]],
  }
```

❷ 헤더 64바이트의 격자 정보

```
def read_stations(buf, num_stn):
  """머리글 뒤 20바이트 × 48. 어느 레이더가 이 합성에 들어갔는지 나반환됩니다."""
  rows = []
  for i in range(min(num_stn, STN_COUNT)):
    b = HEAD_BYTES + i * STN_BYTES
    code = buf[b : b + 6].decode("ascii", "ignore").strip().strip("\x00")
    if not code:
      continue
    rows.append({"번호": i + 1, "레이더 코드": code})
  return pd.DataFrame(rows, columns=None if rows else ["번호", "레이더 코드"])
def lcc(dxy_m, map_code=1, *, MAP_CODE, RE_KM, SLAT1, SLAT2):
  """람베르트 정각원추 도법 상수. 격자 간격은 머리글의 dxy 를 그대로 사용합니다."""
  m = MAP_CODE[map_code]
  d = math.pi / 180.0
  grid = dxy_m / 1000.0
  sn = math.tan(math.pi * 0.25 + SLAT2 * d * 0.5) / math.tan(
    math.pi * 0.25 + SLAT1 * d * 0.5
  )
  sn = math.log(math.cos(SLAT1 * d) / math.cos(SLAT2 * d)) / math.log(sn)
  sf = math.tan(math.pi * 0.25 + SLAT1 * d * 0.5)
  sf = sf**sn * math.cos(SLAT1 * d) / sn
  ro = math.tan(math.pi * 0.25 + m["olat"] * d * 0.5)
  ro = RE_KM / grid * sf / ro**sn
  return (d, grid, sn, sf, ro, m)
def xy_to_lonlat(x, y, dxy_m=500, map_code=1, *, MAP_CODE, RE_KM, SLAT1, SLAT2):
  """격자 번호를 위경도로. 기준격자점이 원점 위경도에 떨어지는지 먼저 확인한다."""
  d, grid, sn, sf, ro, m = lcc(
    dxy_m, map_code, MAP_CODE=MAP_CODE, RE_KM=RE_KM, SLAT1=SLAT1, SLAT2=SLAT2
  )
  X = np.asarray(x, dtype=float) - m["ix0"]
  Yr = ro - (np.asarray(y, dtype=float) - m["iy0"])
  ra = np.hypot(X, Yr)
  if sn < 0:
    ra = -ra
  alat = 2.0 * np.arctan((RE_KM / grid * sf / ra) ** (1.0 / sn)) - math.pi * 0.5
  alon = np.arctan2(X, Yr) / sn + m["olon"] * d
  return (alon / d, alat / d)
def corner_table(h, *, MAP_CODE, RE_KM, SLAT1, SLAT2):
  """네 귀퉁이와 기준격자점을 위경도로 바꿔 격자가 어디를 덮는지 확인한다."""
  nx, ny, dxy = (h["nx"], h["ny"], h["dxy(m)"])
  ix0 = MAP_CODE[h["map_code"]]["ix0"]
  iy0 = MAP_CODE[h["map_code"]]["iy0"]
  pts = [
    ("(1, 1) 왼쪽 아래", 0, 0),
    (f"({nx}, 1) 오른쪽 아래", nx - 1, 0),
    (f"(1, {ny}) 왼쪽 위", 0, ny - 1),
    (f"({nx}, {ny}) 오른쪽 위", nx - 1, ny - 1),
    (f"기준격자점 ({ix0}, {iy0})", ix0, iy0),
  ]
  rows = []
  for name, x, y in pts:
    lon, lat = xy_to_lonlat(
      x,
      y,
      dxy,
      h["map_code"],
      MAP_CODE=MAP_CODE,
```

합성

```
        RE_KM=RE_KM,
        SLAT1=SLAT1,
        SLAT2=SLAT2,
    )
    rows.append(
        {
            "격자점": name,
            "경도": round(float(lon), 4),
            "위도": round(float(lat), 4),
        }
    )
return pd.DataFrame(
    rows, columns=None if rows else ["격자점", "경도", "위도"]
)
```

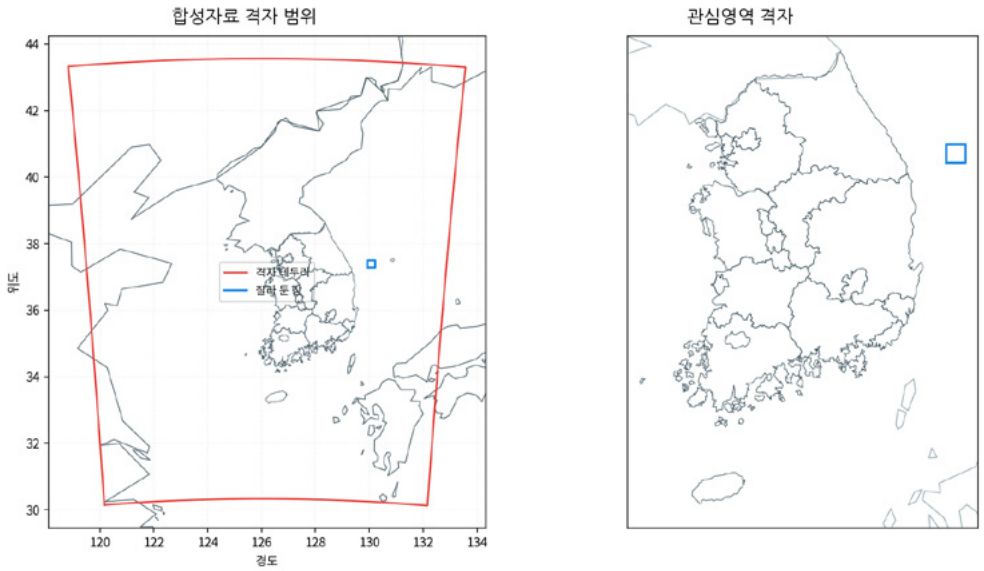


그림 16-1. 헤더 64바이트의 격자 정보

격자점	경도	위도
(1, 1) 왼쪽 아래	120.162	30.1419
(2305, 1) 오른쪽 아래	132.159	30.1211
(1, 2881) 왼쪽 위	118.821	43.3273
(2305, 2881) 오른쪽 위	133.574	43.3007
기준격자점 (1121, 1681)	126	38

번호	레이더 코드
1	KWK
2	GDK
3	GNG
4	KSN
5	JNI
6	MYN
7	PSN
8	GSN
9	SSP
10	BRI

❸ 관측영역 밖·무에코 값 구분

```
def read_window(path, n=None):
  if n is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: n")
  "잘라 둔 창을 읽는다. 값은 short int 이고 dBZ × 100 이다."
  raw = Path(path).read_bytes()
  v = np.frombuffer(raw, dtype="<i2").astype(float)
  return v[: n * n].reshape(n, n)
def to_dbz(win, *, VAL_INSIDE_NULL):
  """센티널을 걸러 내고 100 으로 나눈다."""
  d = win.copy()
  d[d <= VAL_INSIDE_NULL] = np.nan
  return d / 100.0
def value_table(probe):
  """값이 무엇으로 이루어져 있는지. 센티널이 절반을 넘는다."""
  s = probe["센티널"]
  total = probe["격자"]["값수"]
  echo = probe["에코_셀"]
  rows = [
    {"값": "-30000", "뜻": "관측 반경 밖", "칸 수": s["-30000"]},
    {"값": "-25000", "뜻": "반경 안이지만 관측 없음", "칸 수": s["-25000"]},
    {"값": "그 밖", "뜻": "실제 반사도 (dBZ × 100)", "칸 수": echo},
  ]
  t = pd.DataFrame(rows)
  t["비율(%)"] = (100 * t["칸 수"] / total).round(1)
  return t
def dbz_class(dbz, *, DBZ_LEVELS, a, b):
  """dBZ 구간마다 몇 칸인지. 구간이 곧 화면의 색이다."""
  rows = []
  v = dbz[~np.isnan(dbz)]
  for lo, hi, name in DBZ_LEVELS:
    m = (v >= lo) & (v < hi)
    rows.append(
      {
        "dBZ 구간": f"{lo}~{hi}",
        "뜻": name,
        "칸 수": int(m.sum()),
        "강수강도(mm/h)": f"{zr_rain(lo, a=a, b=b):.1f} ~ {zr_rain(hi, a=a, b=b):.1f}",
      }
    )
  t = pd.DataFrame(
    rows,
    columns=None if rows else ["dBZ 구간", "뜻", "칸 수", "강수강도(mm/h)"],
  )
  t["비율(%)"] = (100 * t["칸 수"] / max(len(v), 1)).round(1)
  return t
```

합성

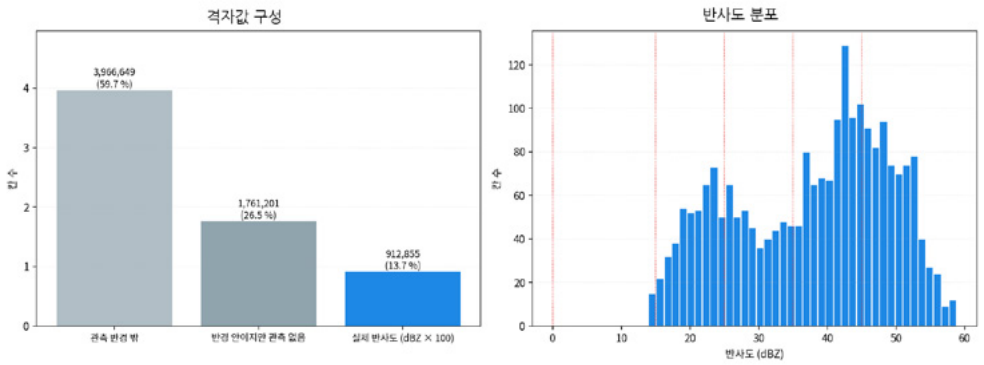


그림 16-2. 관측영역 밖·무에코 값 구분

값	뜻	칸 수	비율(%)
-30000	관측 반경 밖	3966649	59.7
-25000	반경 안이지만 관측 없음	1761201	26.5
그 밖	실제 반사도 (dBZ × 100)	912855	13.7

❹ 관심영역의 반사도 지도 작성

```
def win_lonlat(
    x0=None,
    y0=None,
    n=None,
    dxy=500,
    map_code=1,
    *,
    MAP_CODE,
    RE_KM,
    SLAT1,
    SLAT2
    ):
    if x0 is None or y0 is None or n is None:
        raise ValueError("조회조건·메타데이터를 명시하세요: x0, y0, n")
    "창의 격자 번호를 위경도 판으로 바꾼다."
    xs = np.arange(x0, x0 + n)
    ys = np.arange(y0, y0 + n)
    X, Y = np.meshgrid(xs, ys)
    return xy_to_lonlat(
        X,
        Y,
        dxy,
        map_code,
        MAP_CODE=MAP_CODE,
        RE_KM=RE_KM,
        SLAT1=SLAT1,
        SLAT2=SLAT2,
    )
```

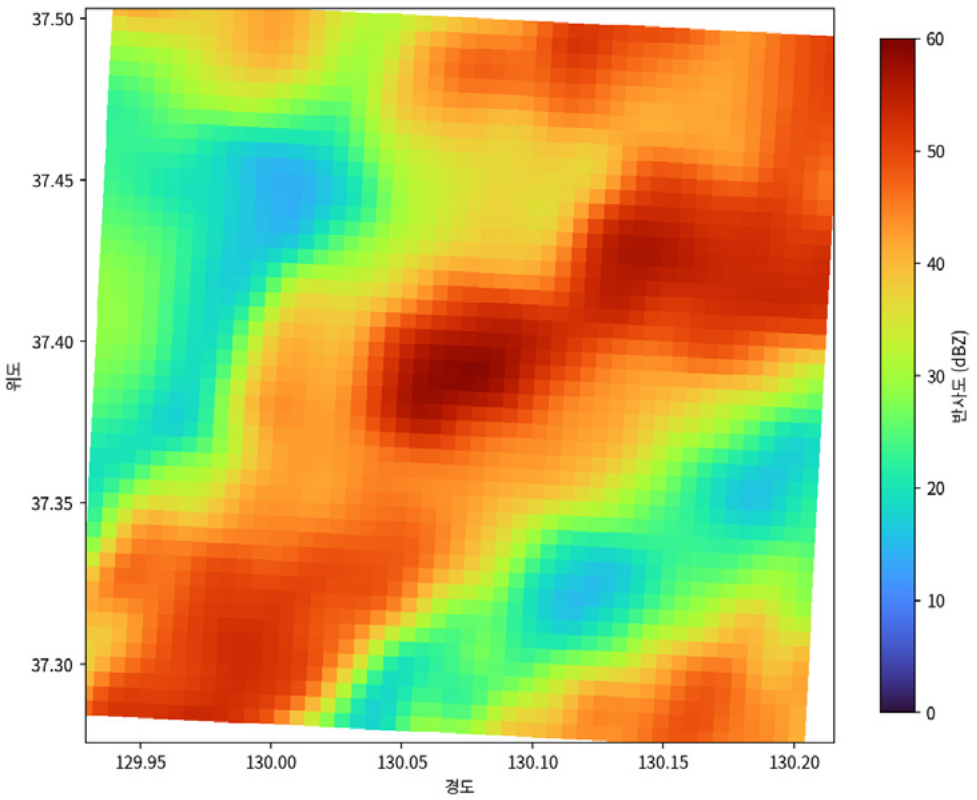


그림 16-3. 관심영역의 반사도 지도 작성

❺ 반사도의 강수강도 변환

```
def zr_rain(dbz, a=None, b=None):
  if a is None or b is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: a, b")
  "반사도를 강수강도로. Z = a R^b 를 뒤집으면 R = (10^(dBZ/10) / a)^(1/b).\n      계수는 강수형태에 따라 달라지므로 무엇을 썼는지 함께 적어야 한다."
  z = 10.0 ** (np.asarray(dbz) / 10.0)
  return (z / a) ** (1.0 / b)
```

합성

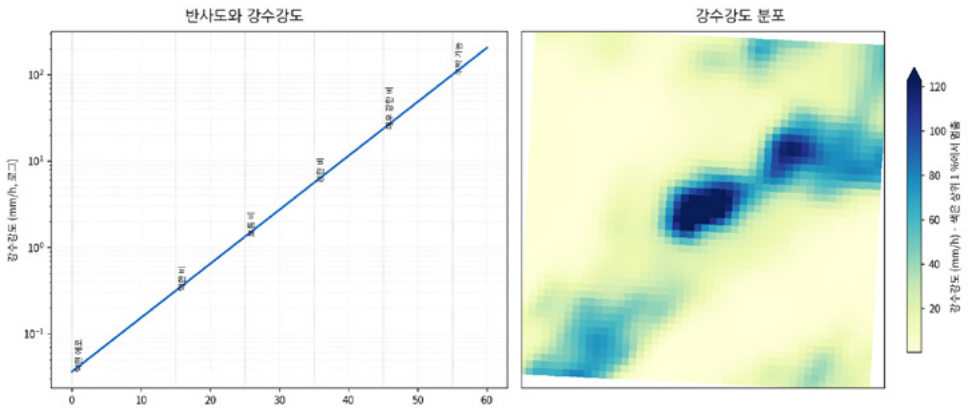


그림 16-4. 반사도의 강수강도 변환

dBZ 구간	뜻	칸 수	강수강도(mm/h)	비율(%)
0~15	약한 에코	15	0.0 ~ 0.3	0.7
15~25	약한 비	430	0.3 ~ 1.3	18.7
25~35	보통 비	422	1.3 ~ 5.6	18.3
35~45	강한 비	734	5.6 ~ 23.7	31.9
45~55	매우 강한 비	649	23.7 ~ 99.9	28.2
55~100	우박 가능	54	99.9 ~ 64842.0	2.3

❻ 최대 반사도 지점의 동서·남북 단면

최대 반사도 셀을 지나는 동서·남북 단면으로 에코의 폭과 기울기를 확인합니다.

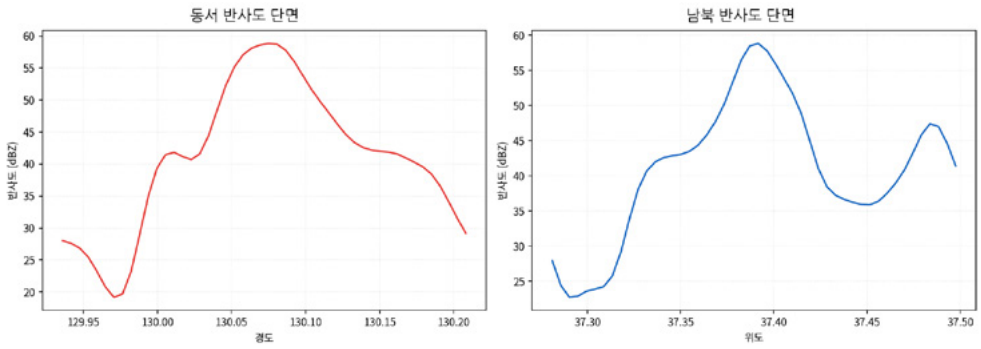


그림 16-5. 최대 반사도 지점의 동서·남북 단면

데이터 활용사례 50

세계기상전문(GTS)

Global Telecommunication System



17.
세계기상전문 지상관측(GTS SYNOP)

지역별 관측값 비교

데이터 활용 경로(API)

기상청 API허브 > 세계기상 > GTS 관측

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 지상 전문 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/gts_syn1.php'
COLS = ['TM', 'STN', 'IW', 'IR', 'IX', 'CH', 'VV', 'CA', 'WD', 'WS',
    'TA', 'TD', 'HM', 'PA', 'PS', 'PT', 'PR', 'RN', 'RH', 'WC', 'WP',
    'CD', 'CL', 'CM', 'CT']
def get_syn(tm):
  res = requests.get(
    api_url,
    params={"tm": tm, "stn": 0, "help": 1, "authKey": API_KEY},
    timeout=60,
  )
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.startswith("#"))
  ]
  d = pd.DataFrame([r[:25] for r in rows if len(r) >= 25], columns=COLS)
  for c in COLS[1:]:
    d[c] = pd.to_numeric(d[c], errors="coerce")
  d[["CH", "VV", "CA", "WD", "WC", "WP"]] = d[
    ["CH", "VV", "CA", "WD", "WC", "WP"]
  ].replace(-9, np.nan)
  d[["TA", "TD", "HM", "PA", "PS", "RN"]] = d[
    ["TA", "TD", "HM", "PA", "PS", "RN"]
  ].replace(-99.0, np.nan)
  return d
d = get_syn('202609200000')
print(len(d), d['STN'].nunique())
5034 5034
```

```
TMS = [f'20260920{h:02d}00' for h in range(0, 24, 3)]
g = pd.concat([get_syn(tm) for tm in TMS])    # 요청 사이에 간격을 둔다
print(g.groupby('TM').size())
00:00  5034
03:00  4614
06:00  5081
09:00  4654
12:00  5393
15:00  4546
18:00  4868
21:00  4232
```

관측시각(UTC)	지점 수	기온이 온 지점	기온(%)
0:00	5034	4999	99.3
3:00	4614	4567	99
6:00	5081	5036	99.1
9:00	4654	4615	99.2
12:00	5393	5345	99.1
15:00	4546	4510	99.2
18:00	4868	4830	99.2
21:00	4232	4194	99.1

표 17-1. 3시간 간격 여덟 시각의 지점 수 (2026-09-20 UTC)

❸ 지점번호별 지역 구분

```
d['블록'] = (d['STN'] // 1000).astype(int)
REG = [('유럽', 1, 19), ('러시아·중앙아시아', 20, 39), ('아시아', 40, 49),
    ('중국', 50, 59), ('아프리카', 60, 68), ('북아메리카', 70, 79),
    ('남아메리카', 80, 88), ('남극', 89, 89), ('오세아니아·태평양', 90, 99)]
def region(b):
  for nm, a, z in REG:
    if a <= b <= z:
      return nm
  return "기타"
d['지역'] = [region(b) for b in d['블록']]
print(d['지역'].value_counts())
유럽        647
러시아·중앙아시아  1207
아시아       1021
아프리카       401
북아메리카      978
남아메리카      170
남극          41
오세아니아·태평양   569
```

관측

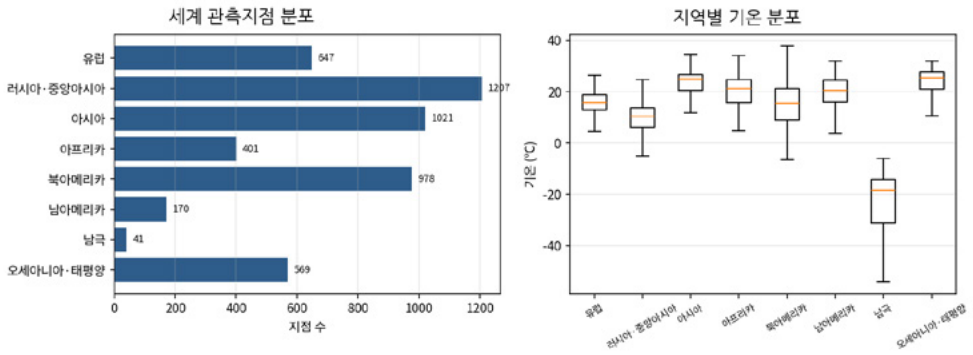


그림 17-1. 지역별 지점 수와 기온 분포

지역	블록번호	지점 수	기온 중앙값(℃)	유인관측(%)
유럽	01~19	647	15.9	34
러시아·중앙아시아	20~39	1207	10.4	98
아시아	40~49	1021	24.9	75
중국	50~59	0	-	-
아프리카	60~68	401	21.2	48
북아메리카	70~79	978	15.5	28
남아메리카	80~88	170	20.4	100
남극	89	41	-18.4	46
오세아니아·태평양	90~99	569	25.4	30

표 17-2. 블록번호로 나눈 지역별 지점 수와 대푯값

❹ 운량·현재일기 코드 집계

```
ELEM = ['CH', 'VV', 'CA', 'WD', 'WS', 'TA', 'TD', 'HM', 'PA', 'PS', 'WC']
print((d[ELEM].notna().sum() / len(d) * 100).round(1))
print(d['CA'].value_counts().sort_index())
CH  64.4
VV  67.4
CA  63.8
WD  96.9
WS  98.1
TA  99.3
TD  98.5
HM  93.2
PA  93.3
PS  89.1
WC  24.4
```

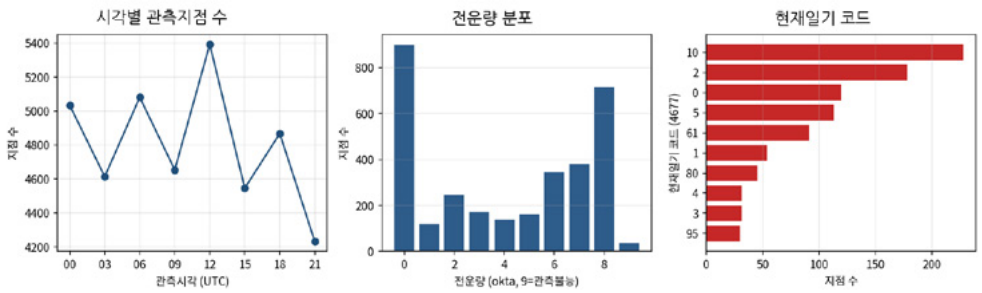


그림 17-2. 시각별 지점 수, 전운량 분포, 현재일기 상위 코드

요소	칸 이름	값이 온 지점	유효자료 비율(%)
최저운고	CH	3243	64.4
시정	VV	3391	67.4
전운량	CA	3214	63.8
풍향	WD	4877	96.9
풍속	WS	4937	98.1
기온	TA	4999	99.3
이슬점온도	TD	4961	98.5
상대습도	HM	4693	93.2
현지기압	PA	4697	93.3
해면기압	PS	4484	89.1
현재일기	WC	1226	24.4

표 17-3. 요소별 유효자료 비율 (00 UTC 기준)

코드 범위	뜻	지점 수	비율(%)
00~19	강수 없음 (구름 변화·연무·박무 등)	807	65.8
20~29	관측 전 1시간의 현상	97	7.9
30~39	모래·먼지폭풍, 낮은 날림눈	1	0.1
40~49	안개	60	4.9
50~59	이슬비	28	2.3
60~69	비	127	10.4
70~79	눈	8	0.7
80~89	소나기	60	4.9
90~99	뇌전	38	3.1

표 17-4. 현재일기 코드 범위별 지점 수

관측

❺ 기온·이슬점·상대습도 비교

```
def es(t):
  return 6.112 * np.exp(17.62 * t / (243.12 + t))
w = g.dropna(subset=['TA', 'TD', 'HM'])
계산 = 100 * es(w['TD']) / es(w['TA'])
print((계산 - w['HM']).abs().describe().round(3))
count   36399.000
mean        0.019
50%         0.000
max       584.900
```

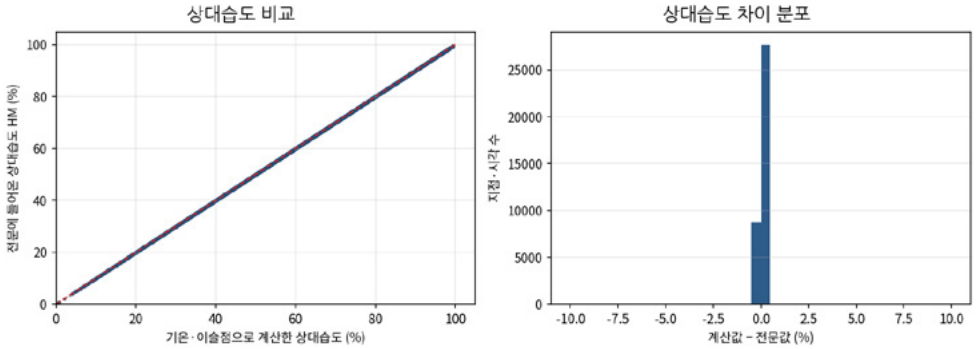


그림 17-3. 계산한 상대습도와 전문에 들어온 상대습도

구간	줄 수	비율(%)
차이 1 % 이내	36398	100
1~3 %	0	0
3~5 %	0	0
5 % 초과	1	0

표 17-5. 계산값과 전문값의 차이

❻ 해상 전문의 좌표와 관측값

선박은 고정된 지점이 아니므로 좌표를 자료 안에서 받아야 합니다.

```
api_ship = 'https://apihub.kma.go.kr/api/typ01/url/gts_ship.php'
res = requests.get(api_ship, params={'tm': '202609200000', 'help': 1,
                   'authKey': API_KEY}, timeout=60)
rows = [ln.split() for ln in res.text.splitlines()
    if ln.strip() and not ln.startswith('#')]
s = pd.DataFrame([r[:35] for r in rows if len(r) >= 35],
         columns=['TM', 'STN', 'LAT', 'LON'] + [f'c{i}' for i in range(31)])
print(len(s), s['STN'].nunique(), s['TM'].min(), s['TM'].max())
7282 1251 202609192100 202609200300
```

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
sa = s.dropna(subset=['TA'])
fig = plt.figure(figsize=(11.2, 5.4))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([-180, 180, -80, 85], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#f2f7fb')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#e3e7ea')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
          linewidth=0.6, edgecolor='#37474f')
sc = ax.scatter(sa['LON'], sa['LAT'], c=sa['TA'], s=7, cmap='RdYlBu_r',
          vmin=-10, vmax=32, transform=PC, linewidths=0)
fig.colorbar(sc, ax=ax, extend='both', label='기온 (℃)')
plt.show()
```

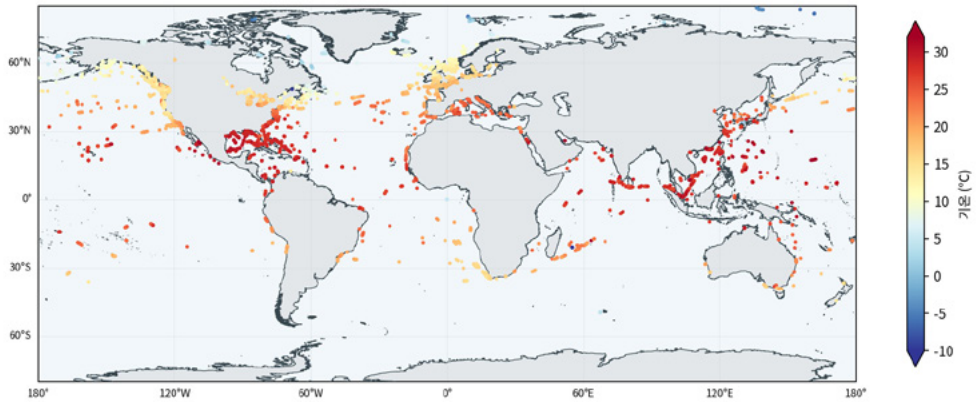


그림 17-4. 해상 전문 5,739건의 위치와 기온

위도대	전문 수	기온 중앙값(℃)	수온 중앙값(℃)	풍속 중앙값(m/s)
남위 60도 이남	0	-	-	-
남위 60~30도	148	16.7	20	11
남위 30~0도	326	22	23.7	7.1
북위 0~30도	1449	28.4	29.2	5
북위 30~60도	4707	16.3	17.6	6
북위 60도 이북	652	10.1	9.3	9.8

표 17-6. 위도대별 해상 전문

관측

18.
공항 정시기상관측보고(METAR)

공항별 시정과 구름 상태 비교

데이터 활용 경로(API)
기상청 API허브 > 항공기상 > 항공기상관측(METAR)

❶ METAR 조회와 XML 해석

태그·단위·공항 좌표·표고를 활용해 지도와 현황 화면을 작성합니다.

```
https://apihub.kma.go.kr/api/typ02/openApi/AmmIwxxmService/getMetar?icao=RKPC&authKey=...
<aixm:designator>RKPC</aixm:designator>
<aixm:name>JEJU INTERNATIONAL AIRPORT</aixm:name>
<gml:pos>33.3 126.29</gml:pos>
<aixm:elevation uom="M">36.0</aixm:elevation>
<iwxxm:airTemperature uom="Cel">27</iwxxm:airTemperature>
<iwxxm:dewpointTemperature uom="Cel">23</...>
<iwxxm:qnh uom="hPa">1011</iwxxm:qnh>
<iwxxm:meanWindDirection uom="deg">210</...>
<iwxxm:meanWindSpeed uom="[kn_i]">004</...>
<iwxxm:prevailingVisibility uom="m">10000</...>
<iwxxm:amount xlink:href=".../FEW"/><iwxxm:base uom="[ft_i]">2500</...>
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
MET_URL = "https://apihub.kma.go.kr/api/typ02/openApi/AmmIwxxmService/getMetar"
KT = 0.5144444444 # knot → m/s
CEIL_FROM = {"BKN", "OVC"}  # 부분·전운량 층에서 운고 선택
```

KO : load_regions 반환 객체. EPSG:4326 경계 GeoJSON에서 생성
필수 입력: KO. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.
조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def load_regions(path, name_field="name"):
    """EPSG:4326 시도 경계 GeoJSON. name_field는 실제 속성 열 이름."""
    import json
    from shapely.geometry import shape, Point


    payload = json.loads(Path(path).read_text("utf-8"))
    features = payload.get("features", [])
```

```
    polygons = []
    for feature in features:
      properties = feature.get("properties", {})
      if name_field not in properties:
        raise ValueError("GeoJSON 시도명 속성 열 확인")
      geometry = shape(feature["geometry"])
      if not geometry.is_valid:
        raise ValueError("유효하지 않은 행정경계")

      polygons.append((properties[name_field], geometry))

    def assign(lon, lat):
      answer = []
      for x, y in zip(lon, lat):
        if not np.isfinite(x) or not np.isfinite(y):
          answer.append(None)
          continue
        point = Point(float(x), float(y))
        matches = [name for name, geom in polygons if geom.covers(point)]

        answer.append(matches[0] if len(matches) == 1 else None)

      return answer

    from types import SimpleNamespace

    return SimpleNamespace(assign_province=assign)
```

시·도별 집계에는 EPSG:4326 좌표계의 행정경계 GeoJSON과 shapely가 필요합니다.
load_regions()의 name_field에 시·도명 속성을 지정하고 반환 객체를 KO에 전달합니다.
경계가 중복되거나 행정구역이 확인되지 않는 위치는 미분류로 처리합니다.

```
def get_metar(icao):
  """공항 하나의 가장 최근 전문. 인자 이름이 icao 라야 받아 준다."""
  r = requests.get(
    MET_URL, params={"icao": icao, "authKey": API_KEY}, timeout=60
  )
  r.raise_for_status()
  return r.text
def walk_metar(icaos=None, pause=0.1):
  if icaos is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: icaos")
  "공항을 하나씩 돌아 전국 한 판을 만든다. 한 번에 여러 공항을 주는 인자가 \n    없으므로 ICAO 부호를 차례로 넣는다."
  out = []
  for a in icaos:
    out.append(f"@ICAO {a}")
    out.append(get_metar(a))
    time.sleep(pause)
  return "\n@@SEP@@\n".join(out)

def parse_metar(xml_text, icao):
  root = ET.fromstring(xml_text)
  code = root.findtext("./header/resultCode")
  if code not in (None, "00", "03"):
    raise RuntimeError(f"resultCode {code}")
  body = next(
    (
      e
      for e in root.iter()
      if e.tag.rsplit("}", 1)[-1] in ("METAR", "SPECI")
    ),
    None,
  )
  rec = {"ICAO": icao, "전문": None}
  if body is None:
    return rec


  def node(parent, tag):
    return (
      next(
        (e for e in parent.iter() if e.tag.rsplit("}", 1)[-1] == tag),
        None,
```

관측

```
        )
        if parent is not None
        else None

    )

def text(parent, tag):

    e = node(parent, tag)
    return (e.text or "").strip() if e is not None else ""

def number(parent, tag, target=None):
    e = node(parent, tag)
    if e is None or e.text is None:
        return np.nan
    try:
        value = float(e.text)
    except ValueError:
        return np.nan
    unit = e.get("uom", "").lower()
    if target == "kt" and unit in ("m/s", "m.s-1"):

        value /= 0.5144444444
    if target == "ft" and unit in ("m", "metre", "meter"):
        value /= 0.3048

    if target == "m" and unit in ("[ft_i]", "ft"):
        value *= 0.3048
    return value

rec["전문"] = body.tag.rsplit("}", 1)[-1]
rec["공항"] = text(body, "name")
rec["자동관측"] = body.get("automatedStation") == "true"
observed = node(body, "observationTime")
tm = text(observed, "timePosition")
rec["관측시각(UTC)"] = pd.to_datetime(tm, utc=True, errors="coerce")
rec["관측시각(KST)"] = (
    rec["관측시각(UTC)"].tz_convert("Asia/Seoul")
    if not pd.isna(rec["관측시각(UTC)"])
    else pd.NaT
)
pos = text(body, "pos").split()
rec["위도값"], rec["경도값"] = (
    map(float, pos) if len(pos) == 2 else (np.nan, np.nan)
)
rec["표고(m)"] = number(body, "elevation", "m")
obs = node(body, "MeteorologicalAerodromeObservation")
rec["CAVOK"] = (
    obs.get("cloudAndVisibilityOK") == "true" if obs is not None else False
)
for col, tag, target in [
    ("기온(C)", "airTemperature", None),
    ("이슬점(C)", "dewpointTemperature", None),
    ("QNH(hPa)", "qnh", None),
    ("풍향(도)", "meanWindDirection", None),
    ("풍속(kt)", "meanWindSpeed", "kt"),
    ("돌풍(kt)", "windGustSpeed", "kt"),
    ("시정(m)", "prevailingVisibility", "m"),
]:
    rec[col] = number(obs, tag, target)
href = "{http://www.w3.org/1999/xlink}href"
weather = (
    [
        e.get(href, "").rsplit("/", 1)[-1]
        for e in obs.iter()
        if e.tag.rsplit("}", 1)[-1] == "presentWeather"
    ]
    if obs is not None
    else []
)
rec["현재날씨"] = " ".join(weather)
rec["날씨풀이"] = rec["현재날씨"]
layers = []
for layer in obs.iter() if obs is not None else []:
    if layer.tag.rsplit("}", 1)[-1] != "CloudLayer":
        continue
    amount = node(layer, "amount")
    if amount is not None:
        layers.append(
            (
                amount.get(href, "").rsplit("/", 1)[-1],
```

```
            number(layer, "base", "ft"),
          )
        )
    rec["구름층"] = layers
    rec["구름표기"] = " ".join(
      (
        f"{a}{int(round(b / 100)):03d}" if np.isfinite(b) else a
        for a, b in layers
      )
    )
    return rec
```

❷ XML 전문을 공항별 표로 변환

```
def split_pack(path):
    """받아 둔 응답 묶음을 공항별 XML 로 자른다."""
    txt = Path(path).read_text("utf-8")
    cur, out = (None, {})
    for blk in txt.split("@@SEP@@"):
        b = blk.strip()
        if b.startswith("@@METAR@@"):
            b = b[len("@@METAR@@") :].strip()
        if b.startswith("@ICAO"):
            cur = b.split()[1]
        elif b.startswith("<?xml") and cur:
            out[cur] = b
    return out
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
    """공항별 XML 을 한 행씩으로 펴서 표로 만든다."""
    rows = [parse_metar(x, k) for k, x in split_pack(path).items()]
    d = pd.DataFrame(rows)
    d["풍속(m/s)"] = (d["풍속(kt)"] * KT).round(1)
    d["T-Td(C)"] = d["기온(C)"] - d["이슬점(C)"]
    return d
```

▸ 실행 결과

ICAO	공항	전문	기온 (C)	이슬점 (C)	QNH (hPa)	풍향 (도)	풍속 (m/s)	시정 (m)	CAVOK	구름 표기	날씨 풀이
RKSI	INCHEON	METAR	24	22	1012	50	0.5		TRUE		
RKSS	GIMPO	METAR	22	22	1012	0	0	8000	FALSE		박무
RKPC	JEJU	METAR	27	23	1011	210	2.1	10000	FALSE	FEW2500 SCT7000 BKN20000	
RKPK	GIMHAE	METAR	27	25	1011	50	1.5		TRUE		
RKPU	ULSAN	METAR	26	24	1011	360	3.6		TRUE		
RKPS	SACHEON	METAR	25	25	1011	20	1		TRUE		
RKTU	CHEONGJU	METAR	22	21	1012	90	1	4800	FALSE	BKN15000 BKN25000	박무
RKTN	DAEGU	SPECI	25	24	1012	30	1	10000	FALSE	BKN3000	약한 비
RKTH	POHANG	METAR	26	25	1012	40	3.1	6000	FALSE	FEW800 BKN1000 BKN2000	박무
RKJJ	GWANGJU	METAR	25	25	1011	0	0		TRUE		
RKJB	MUAN	METAR	25	22	1011	340	2.1		TRUE		
RKJY	YEOSU	METAR	26	24	1011	300	1.5		TRUE		
RKJK	GUNSAN	METAR	2	-2	1023	270	1.5		TRUE		
RKNY	YANGYANG	METAR	21	21	1013	320	1.5		TRUE		
RKNW	WONJU	METAR	22	21	1012	30	1	6000	FALSE	SCT500 SCT1000 BKN20000	박무

관측

❸ 좌표 해석 방식 비교

```
def as_deg_min(v):
  """소수 자리를「분」으로 읽습니다. 37.28 → 37도 28분 → 37.4667도."""
  d = int(v)
  m = round((v - d) * 100)
  return np.nan if m >= 60 else d + m / 60.0
def coord_frame(d, *, KO):
  """좌표를 도분과 십진도로 각각 해석한 결과를 비교합니다. 공항 기준점과 공식 좌표 형식으로 올바른 해석을 확인합니다."""

  t = d.dropna(subset=["위도값"]).copy()
  t["소수위도"] = t["위도값"]
  t["소수경도"] = t["경도값"]
  t["도분위도"] = t["위도값"].map(as_deg_min)
  t["도분경도"] = t["경도값"].map(as_deg_min)
  if KO:
    t["소수 시·도"] = KO.assign_province(
      t["소수경도"].values, t["소수위도"].values
    )
    t["도분 시·도"] = KO.assign_province(
      t["도분경도"].values, t["도분위도"].values
    )
  t["분자리"] = (t["위도값"] % 1 * 100).round().astype(int)
  t["분자리(경도)"] = (t["경도값"] % 1 * 100).round().astype(int)
  t["도분으로 읽힘"] = t["도분위도"].notna() & t["도분경도"].notna()
  t["위도"] = np.where(t["도분으로 읽힘"], t["도분위도"], t["소수위도"])
  t["경도"] = np.where(t["도분으로 읽힘"], t["도분경도"], t["소수경도"])
  return t
```

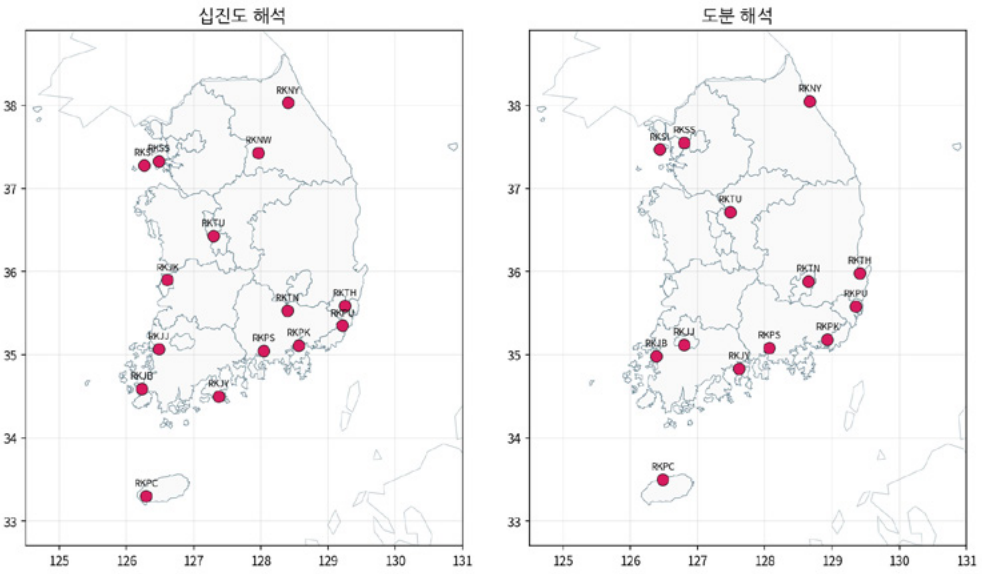


그림 18-1. 좌표 해석 방식 비교

ICAO	위도값	경도값	소수 시·도	도분위도	도분경도	도분 시·도
RKSI	37.28	126.26		37.4667	126.433	인천광역시
RKSS	37.33	126.48		37.55	126.8	서울특별시
RKPC	33.3	126.29	제주특별자치도	33.5	126.483	제주특별자치도
RKPK	35.11	128.56	경상남도	35.1833	128.933	부산광역시
RKPU	35.35	129.21	부산광역시	35.5833	129.35	울산광역시
RKPS	35.05	128.04		35.0833	128.067	경상남도
RKTU	36.43	127.29	세종특별자치시	36.7167	127.483	충청북도
RKTN	35.53	128.39	경상남도	35.8833	128.65	대구광역시
RKTH	35.59	129.25	울산광역시	35.9833	129.417	경상북도
RKJJ	35.07	126.48	전라남도	35.1167	126.8	광주광역시
RKJB	34.59	126.23		34.9833	126.383	전라남도
RKJY	34.5	127.37	전라남도	34.8333	127.617	전라남도
RKJK	35.9	126.61				
RKNY	38.03	128.4	강원도	38.05	128.667	강원도
RKNW	37.43	127.96	강원도	37.7167		

❹ 시정·운고에 따른 비행 범주

공항별 범주를 색으로 표시해 기상 조건을 비교합니다.

```
def ceiling_ft(rec_layers, cavok):
    """BKN·OVC 구름층 중 가장 낮은 층의 운고를 구합니다. CAVOK은 시정·구름·일기 조건을 만족하는 표기로, 모든 구름이 없다는 뜻은 아닙니다."""

    if cavok or not rec_layers:
        return np.inf
    base = [b for a, b in rec_layers if a in CEIL_FROM and np.isfinite(b)]
    return min(base) if base else np.inf
def vis_m(v, cavok):
    """CAVOK 이면 시정은 10 km 이상이라는 뜻이므로 그 값을 넣는다."""
    if cavok:
        return 10000.0
    return v
```

분류는 미국 AWC의 시정·운고 비교 기준(시정 statute mile, 운고 ft)을 참고한 분석 지표입니다. 결측은 '미분류'이며 실제 운항 판단에는 해당 공항·항공기의 운항 기준을 적용합니다. 출처: https://aviationweather.gov/impactboard/help.html

```
def flight_rule(vis, ceil):
    """시정(m)·운고(ft)의 AWC 비교 범주. 결측은 미분류로 둔다."""
    if pd.isna(vis) or pd.isna(ceil):
        return "미분류"
    miles = float(vis) / 1609.344
    if miles < 1 or ceil < 500:
        return "LIFR"
    if miles < 3 or ceil < 1000:
        return "IFR"
    if miles <= 5 or ceil <= 3000:
        return "MVFR"
    return "VFR"
def rule_frame(d, c):
```

관측

```
rows = []
for _, r in d.dropna(subset=["위도값"]).iterrows():
    cav = bool(r["CAVOK"])
    v = vis_m(r["시정(m)"], cav)
    ce = ceiling_ft(r["구름층"], cav)
    rows.append(
        {
            "ICAO": r["ICAO"],
            "공항": r["공항"],
            "CAVOK": cav,
            "시정(m)": v,
            "운고(ft)": None if not np.isfinite(ce) else int(ce),
            "구름": r["구름표기"] or "-",
            "날씨": r["날씨풀이"] or "-",
            "등급": flight_rule(v, ce),
        }
    )
t = pd.DataFrame(
    rows,
    columns=(
        None
        if rows
        else [
            "ICAO",
            "공항",
            "CAVOK",
            "시정(m)",
            "운고(ft)",
            "구름",
            "날씨",
            "등급",
        ]
    ),
).merge(c[["ICAO", "경도", "위도"]], on="ICAO")
return t
```

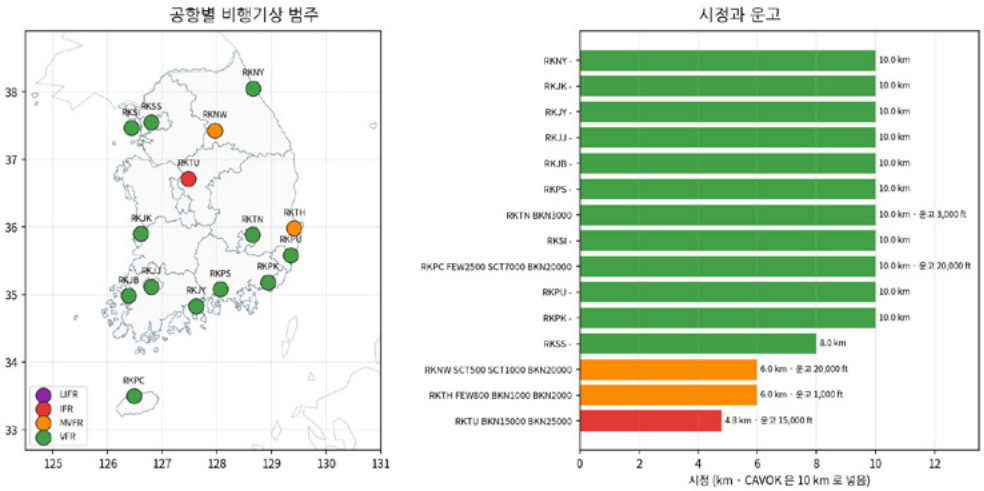


그림 18-2. 시정·운고에 따른 비행 범주

ICAO	공항	CAVOK	시정 (m)	운고 (ft)	구름	날씨	등급
RKTU	CHEONGJU	FALSE	4800	15000	BKN15000 BKN25000	박무	IFR
RKTH	POHANG	FALSE	6000	1000	FEW800 BKN1000 BKN2000	박무	MVFR
RKNW	WONJU	FALSE	6000	20000	SCT500 SCT1000 BKN20000	박무	MVFR
RKSS	GIMPO	FALSE	8000		-	박무	VFR
RKPK	GIMHAE	TRUE	10000		-	-	VFR
RKPU	ULSAN	TRUE	10000		-	-	VFR
RKPC	JEJU	FALSE	10000	20000	FEW2500 SCT7000 BKN20000	-	VFR
RKSI	INCHEON	TRUE	10000		-	-	VFR
RKTN	DAEGU	FALSE	10000	3000	BKN3000	약한 비	VFR
RKPS	SACHEON	TRUE	10000		-	-	VFR
RKJB	MUAN	TRUE	10000		-	-	VFR
RKJJ	GWANGJU	TRUE	10000		-	-	VFR
RKJY	YEOSU	TRUE	10000		-	-	VFR
RKJK	GUNSAN	TRUE	10000		-	-	VFR
RKNY	YANGYANG	TRUE	10000		-	-	VFR

❺ 기온·이슬점 차와 시정 비교

기온·이슬점 차는 공기가 포화에 가까운 정도를 나타냅니다.

```
def spread_frame(d, t):
    """기온과 이슬점온도의 차이를 박무·안개 보고 및 시정과 비교합니다."""

    g = (
        d.dropna(subset=["기온(C)"])
        .drop(columns=["시정(m)"])
        .merge(t[["ICAO", "시정(m)", "등급"]], on="ICAO")
    )
    g["습함"] = g["현재날씨"].str.contains("BR|FG", na=False)
    return g[
        [
            "ICAO",
            "공항",
            "기온(C)",
            "이슬점(C)",
            "T-Td(C)",
            "QNH(hPa)",
            "풍속(m/s)",
            "시정(m)",
            "현재날씨",
            "습함",
            "등급",
        ]
    ]
```

관측

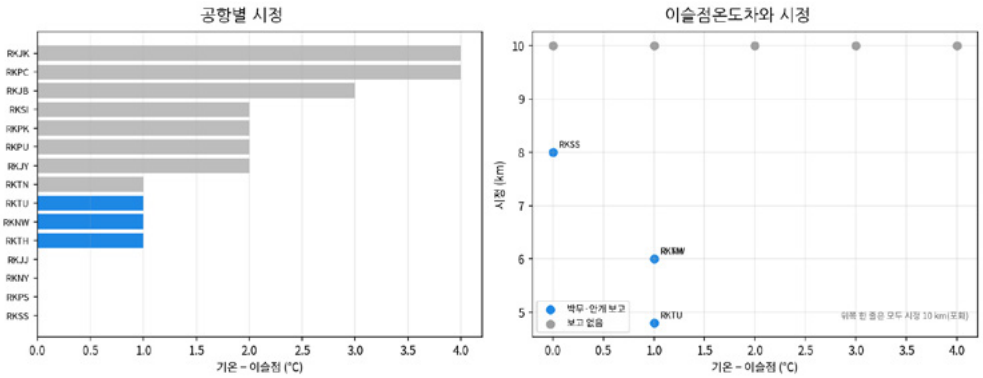


그림 18-3. 기온·이슬점 차와 시정 비교

ICAO	공항	기온 (C)	이슬점 (C)	T-Td (C)	QNH (hPa)	풍속 (m/s)	시정 (m)	현재 날씨	습함	등급
RKSS	GIMPO	22	22	0	1012	0	8000	BR	TRUE	VFR
RKPS	SACHEON	25	25	0	1011	1	10000		FALSE	VFR
RKNY	YANGYANG	21	21	0	1013	1.5	10000		FALSE	VFR
RKJJ	GWANGJU	25	25	0	1011	0	10000		FALSE	VFR
RKTH	POHANG	26	25	1	1012	3.1	6000	BR	TRUE	MVFR
RKNW	WONJU	22	21	1	1012	1	6000	BR	TRUE	MVFR
RKTU	CHEONGJU	22	21	1	1012	1	4800	BR	TRUE	IFR
RKTN	DAEGU	25	24	1	1012	1	10000	-RA	FALSE	VFR
RKJY	YEOSU	26	24	2	1011	1.5	10000		FALSE	VFR
RKPU	ULSAN	26	24	2	1011	3.6	10000		FALSE	VFR
RKPK	GIMHAE	27	25	2	1011	1.5	10000		FALSE	VFR
RKSI	INCHEON	24	22	2	1012	0.5	10000		FALSE	VFR
RKJB	MUAN	25	22	3	1011	2.1	10000		FALSE	VFR
RKPC	JEJU	27	23	4	1011	2.1	10000		FALSE	VFR
RKJK	GUNSAN	2	-2	4	1023	1.5	10000		FALSE	VFR

❻ 관측 경과시간과 결측 표시

```
def age_frame(d, now=None):
    """공항별 관측시각과 조회시각의 차이를 분 단위로 계산합니다."""

    ok = d.dropna(subset=["관측시각(UTC)"]).copy()
    now = now or ok["관측시각(UTC)"].max()
    ok["나이(분)"] = (
        ((now - ok["관측시각(UTC)"]).dt.total_seconds() / 60).round().astype(int)
    )
    ok["관측(KST)"] = ok["관측시각(KST)"].dt.strftime("%m-%d %H:%M")
    miss = d[d["관측시각(UTC)"].isna()] if "관측시각(UTC)" in d else d.iloc[:0]
    return (
        ok[
            ["ICAO", "공항", "전문", "자동관측", "관측(KST)", "나이(분)"]
        ].sort_values("나이(분)"),
        miss,
    )
```

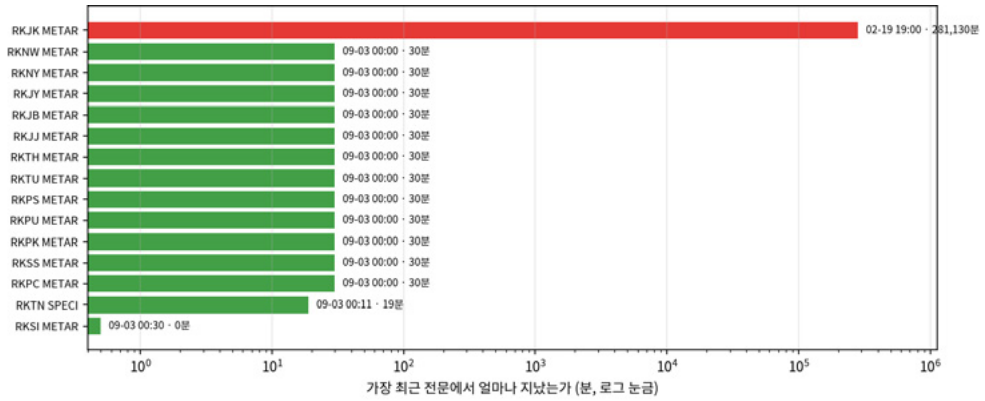


그림 18-4. 관측 경과시간과 결측 표시

ICAO	공항	전문	자동관측	관측(KST)	나이(분)
RKSI	INCHEON	METAR	FALSE	2026-09-03 0:30	0
RKTN	DAEGU	SPECI	TRUE	2026-09-03 0:11	19
RKPC	JEJU	METAR	FALSE	2026-09-03 0:00	30
RKSS	GIMPO	METAR	FALSE	2026-09-03 0:00	30
RKPK	GIMHAE	METAR	TRUE	2026-09-03 0:00	30
RKPU	ULSAN	METAR	TRUE	2026-09-03 0:00	30
RKPS	SACHEON	METAR	TRUE	2026-09-03 0:00	30
RKTU	CHEONGJU	METAR	TRUE	2026-09-03 0:00	30
RKTH	POHANG	METAR	TRUE	2026-09-03 0:00	30
RKJJ	GWANGJU	METAR	TRUE	2026-09-03 0:00	30
RKJB	MUAN	METAR	TRUE	2026-09-03 0:00	30
RKJY	YEOSU	METAR	TRUE	2026-09-03 0:00	30
RKNY	YANGYANG	METAR	TRUE	2026-09-03 0:00	30
RKNW	WONJU	METAR	TRUE	2026-09-03 0:00	30
RKJK	GUNSAN	METAR	TRUE	2026-02-19 19:00	281130

데이터 활용사례 50

기상예보

Weather Forecast



날씨예보

19. 초단기실황

지역별 현재 날씨 비교

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 단기예보

❶ 초단기실황 조회

기온·1시간강수·습도·강수형태·풍향·풍속으로 위치별 현황을 표시합니다. 실행 시 최근 유효 발표시각을 자동 선택합니다.

```
https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getUltraSrtNcst
  ?base_date={최근날짜}&base_time={최근정시}&nx=60&ny=127&dataType=JSON&authKey=...
{"category":"T1H","obsrValue":"28.5","nx":60,"ny":127}
{"category":"REH","obsrValue":"62", …}
{"category":"PTY","obsrValue":"0", …}
#  category 가 행으로 오므로 열로 뒤집어 써야 한다
#  PTY : 0 없음 · 1 비 · 2 비/눈 · 3 눈 · 5 빗방울 · 6 빗방울/눈날림 · 7 눈날림
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
NCST_URL = "https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getUltraSrtNcst"
PTY = {0:"없음",1:"비",2:"비/눈",3:"눈",4:"소나기",5:"빗방울",6:"빗방울/눈날림",7:"눈날림"}
SKY = {1:"맑음",3:"구름많음",4:"흐림"}
CATS = ["T1H", "REH", "WSD", "RN1", "PTY", "VEC", "UUU", "VVV"]
```

위치 표는 지점정보 API에서 받은 지점·지점명·경도·위도 열을 사용합니다.
d=grid_frame(stations)로 수집하고 d.to_csv("ncst.csv",index=False,encoding="utf-8-sig")로 저장합니다. API 제공기간 밖의 24시간 과거자료는 실황 API에서 무리하게 조회하지 않고 관측자료 API로 분석합니다.

```
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 인증·권한·접속정책 확인")
  js = r.json()
  response = js.get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code == "03":
    return ([], 0)
  if code != "00":
    raise RuntimeError(f"API 오류코드 {code}: 제공기간·요청조건 확인")
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):
    items = [items]
  if not isinstance(items, list):
    raise ValueError("API items 구조 확인 필요")
  return (items, int(body.get("totalCount", len(items))))
def dfs_xy(lon, lat):
  """위경도를 단기예보 격자번호(nx, ny)로 변환합니다. 람베르트 정각원추 도법, 표준위도 30N·60N, 기준점 126E·38N 및 5
km 격자를 적용합니다."""


  RE, GRID = (6371.00877, 5.0)
  SLAT1, SLAT2, OLON, OLAT, XO, YO = (30.0, 60.0, 126.0, 38.0, 43, 136)
  D = math.pi / 180.0
  re, slat1, slat2 = (RE / GRID, SLAT1 * D, SLAT2 * D)
  olon, olat = (OLON * D, OLAT * D)
  sn = math.tan(math.pi * 0.25 + slat2 * 0.5) / math.tan(
    math.pi * 0.25 + slat1 * 0.5
  )
  sn = math.log(math.cos(slat1) / math.cos(slat2)) / math.log(sn)
  sf = math.tan(math.pi * 0.25 + slat1 * 0.5)
  sf = sf**sn * math.cos(slat1) / sn
  ro = math.tan(math.pi * 0.25 + olat * 0.5)
  ro = re * sf / ro**sn
  ra = math.tan(math.pi * 0.25 + lat * D * 0.5)
  ra = re * sf / ra**sn
  theta = lon * D - olon
  theta = (theta + math.pi) % (2 * math.pi) - math.pi
  theta *= sn
  return (
    int(ra * math.sin(theta) + XO + 0.5),
    int(ro - ra * math.cos(theta) + YO + 0.5),
  )
def get_ncst(nx, ny, base_date=None, base_time=None, retry=3):
  """KST 정시 후보; 자료 없음일 때만 이전 정시를 확인한다."""
  if (base_date is None) != (base_time is None):
    raise ValueError("날짜·시간을 함께 지정")
  now = pd.Timestamp.now(tz="Asia/Seoul").floor("h")
  candidates = (
    [(base_date, base_time)]
    if base_date
    else [
      (
        (now - pd.Timedelta(hours=h)).strftime("%Y%m%d"),
        (now - pd.Timedelta(hours=h)).strftime("%H00"),
      )
      for h in range(6)
    ]
  )
  for date, hour in candidates:
    items, _ = request_items(
      NCST_URL,
      {
        "base_date": date,
        "base_time": hour,
        "nx": nx,
        "ny": ny,
        "dataType": "JSON",
        "numOfRows": 100,
        "pageNo": 1,
        "authKey": API_KEY,
      },
    )
    if items:
      return items
  raise ValueError("조회조건·메타데이터를 명시하세요: base_date, base_time")

def parse_ncst(items):
```

날씨예보

```
"""category 가 행이 아니라 열이 되도록 뒤집는다."""
m = {it["category"]: it["obsrValue"] for it in items}
return {c: float(m[c]) if c in m else np.nan for c in CATS}
```

❷ 위경도를 예보 격자로 변환

이 API는 위경도 대신 격자 번호를 받습니다. 람베르트 도법으로 좌표를 변환해 요청에 사용합니다.

```
def grid_frame(stations, base_date=None, base_time=None):
    """지점표(지점·지점명·경도·위도)의 좌표별 API 실황을 모은다."""
    rows = []
    for _, station in stations.iterrows():
        nx, ny = dfs_xy(float(station["경도"]), float(station["위도"]))
        items = get_ncst(nx, ny, base_date=base_date, base_time=base_time)
        if not items:
            continue
        row = dict(station)
        row.update(parse_ncst(items))
        row.update(nx=nx, ny=ny)
        row["시각"] = pd.Timestamp(
            datetime.strptime(
                items[0]["baseDate"] + items[0]["baseTime"], "%Y%m%d%H%M"
            ),
            tz="Asia/Seoul",
        )
        rows.append(row)
        time.sleep(0.2)
    d = pd.DataFrame(rows)
    if not d.empty:
        d["강수형태"] = d["PTY"].map(PTY)
    return d
```

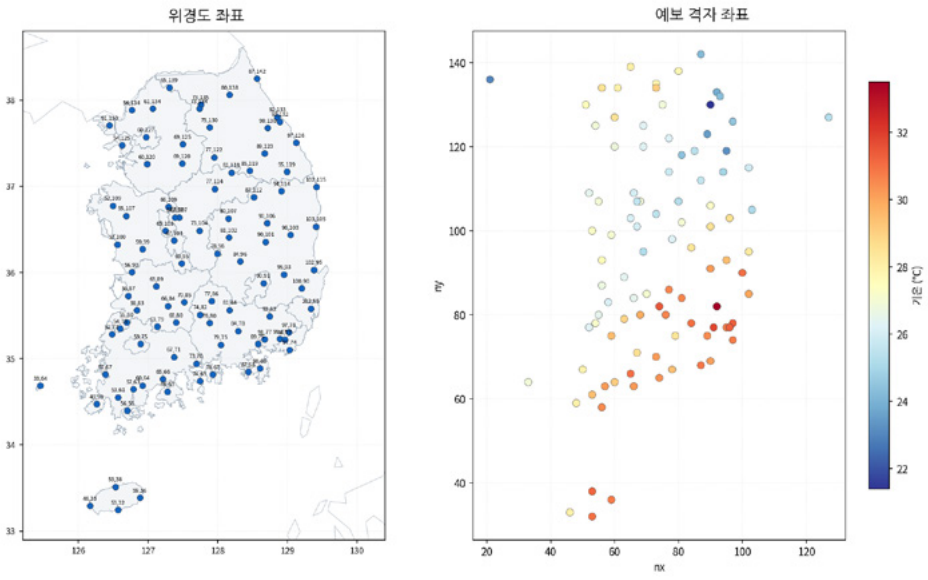


그림 19-1. 위경도를 예보 격자로 변환

❸ 같은 시각의 요소별 전국 분포

지점마다 여섯 요소를 함께 받으므로 같은 응답으로 요소별 지도를 만들고, 화면에 필요한 항목을 선택할 수 있습니다.

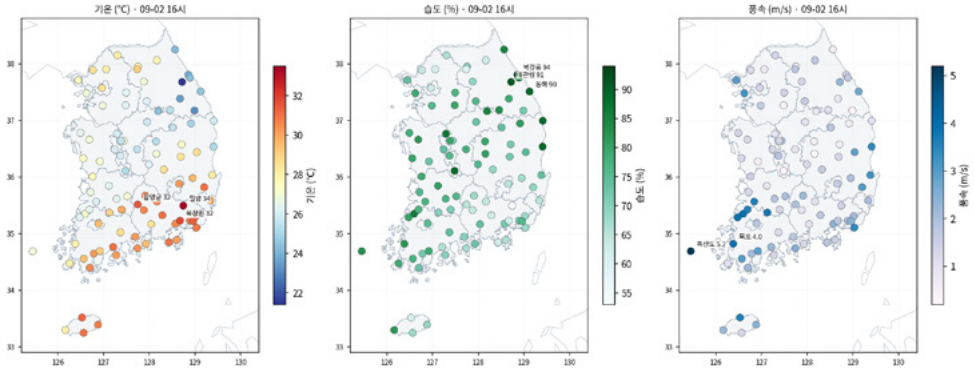


그림 19-2. 같은 시각의 요소별 전국 분포

지점명	nx	ny	T1H	REH	WSD	RN1	강수형태
밀양	92	82	33.5	53	1.4	0	없음
북창원	91	77	31.8	62	1.7	0	없음
함양군	74	82	31.5	65	1.9	0	없음
성산	59	36	30.8	69	2.4	0	없음
부산	97	74	30.8	68	3.4	0	없음
김해시	95	77	30.7	73	2	0	없음

❹ 격자 실황과 지점 관측 비교

```
def compare(g, o):
    """격자 실황과 지점 관측을 같은 지점에서 비교합니다."""
    m = g.merge(o, on="지점")
    m["기온차"] = (m["T1H"] - m["관측기온"]).round(1)
    m["습도차"] = (m["REH"] - m["관측습도"]).round(1)
    m["풍속차"] = (m["WSD"] - m["관측풍속"]).round(1)
    return m
```

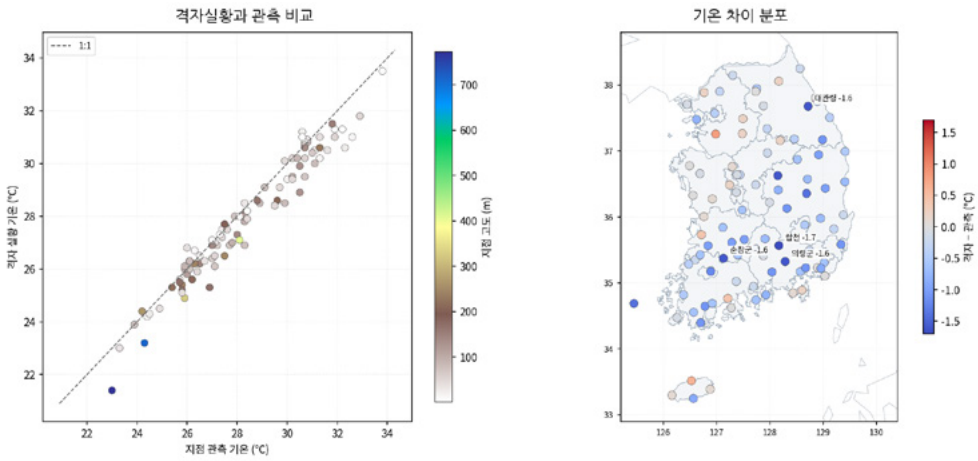


그림 19-3. 격자 실황과 지점 관측 비교

날씨예보

❺ 시간별 실황으로 하루 변화 분석

```
def h24_frame(path):
    """grid_frame 결과를 여러 시각 수집한 CSV. 조회 가능 기간 안에서 생성한다."""
    d = pd.read_csv(path, encoding="utf-8-sig")
    if "시각" not in d:
        raise ValueError("grid_frame 출력 CSV 필요")
    d["시각"] = pd.to_datetime(d["시각"], utc=True).dt.tz_convert("Asia/Seoul")
    return d
```

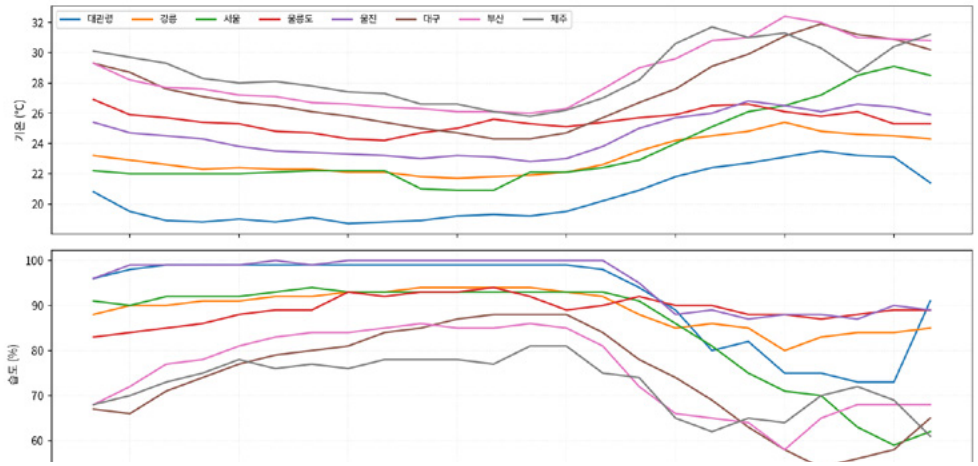


그림 19-4. 시간별 실황으로 하루 변화 시계열

❻ 위치별 날씨 카드 구성

```
def card(g, stn):
    """화면 하나에 올릴 값만 골라 카드로 만든다."""
    r = g[g["지점"] == stn].iloc[0]
    return {
        "지점": f"{r['지점명']} ({int(r['지점'])})",
        "격자": f"nx={int(r['nx'])}, ny={int(r['ny'])}",
        "기온": f"{r['T1H']:.1f} ℃",
        "습도": f"{r['REH']:.0f} %",
        "바람": f"{r['WSD']:.1f} m/s · {r['VEC']:.0f}°",
        "강수": f"{r['강수형태']} · 1시간 {r['RN1']:.1f} mm",
    }
```

20-①.
초단기예보
시간별 강수 변화 분석

데이터 활용 경로(API)
기상청 API허브 > 예특보 > 단기예보

❶ 초단기예보 조회
기온·1시간강수·하늘상태·강수형태·습도·풍속·낙뢰로 강수 알림을 구성합니다. 실행 시 최근 유효 HH30 발표시각을 자동 선택합니다.

```
https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getUltraSrtFcst
  ?base_date={최근날짜}&base_time={최근 HH30}&nx=60&ny=127&dataType=JSON&numOfRows=200
&authKey=...
{"category":"T1H","fcstDate":"20260902","fcstTime":"1700","fcstValue":"28","nx":60,"ny":127}
{"category":"RN1", … ,"fcstValue":"강수없음"}
{"category":"SKY", … ,"fcstValue":"4"}
#  RN1 은 숫자가 아니라 문구로 온다 — '강수없음' '1mm 미만' '4.0mm' '30.0~50.0mm'
#  SKY : 1 맑음 · 3 구름많음 · 4 흐림 / PTY : 0 없음 · 1 비 · 3 눈
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
FCST_URL = "https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getUltraSrtFcst"
PTY = {0:"없음",1:"비",2:"비/눈",3:"눈",4:"소나기",5:"빗방울",6:"빗방울/눈날림",7:"눈날림"}
SKY = {1:"맑음",3:"구름많음",4:"흐림"}
CATS = ["T1H", "RN1", "SKY", "PTY", "REH", "WSD", "LGT", "VEC", "UUU", "VVV"]
def forecast_coordinates(nx, ny):
    """단기예보 5km 격자 중심의 (경도, 위도). 공식 격자 도법을 적용한다."""
    re = 6371.00877 / 5.0
    rad = math.pi / 180.0
    s1, s2 = 30.0 * rad, 60.0 * rad
    sn = math.log(math.cos(s1) / math.cos(s2)) / math.log(
        math.tan(math.pi / 4 + s2 / 2) / math.tan(math.pi / 4 + s1 / 2)
    )
    sf = math.tan(math.pi / 4 + s1 / 2) ** sn * math.cos(s1) / sn
    ro = re * sf / math.tan(math.pi / 4 + 38.0 * rad / 2) ** sn
    x = np.asarray(nx, dtype=float) - 43.0
    y = ro - (np.asarray(ny, dtype=float) - 136.0)
    rho = np.hypot(x, y)
    lat = 2 * np.arctan((re * sf / rho) ** (1 / sn)) - math.pi / 2
    lon = np.arctan2(x, y) / sn + 126.0 * rad
    return lon / rad, lat / rad
```

날씨예보

실행 순서: d = save_forecast_csv(60, 127); d = read_pack("forecast.csv"). 분석표의 지점은 관측소 번호가 아니라 "nx,ny" 격자 식별자이며, 경도·위도는 격자 중심 좌표입니다. 다른 격자는 nx·ny를 바꿔 조회하고 pd.concat으로 결합합니다.

```
def save_forecast_csv(nx=60, ny=127, path="forecast.csv"):
  d = parse_fcst(get_fcst(nx, ny))
  d.to_csv(path, index=False, encoding="utf-8-sig")
  return d
```

입력 파일은 save_forecast_csv()로 생성합니다. 한 격자 예시(60,127)에서 확인한 뒤 여러 위치로 확장합니다. 전국 도표는 해당 도표의 위치·기간과 동일한 조건이 필요합니다.

```
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 인증·권한·접속정책 확인")
  js = r.json()
  response = js.get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code == "03":
    return ([], 0)
  if code != "00":
    raise RuntimeError(f"API 오류코드 {code}: 제공기간·요청조건 확인")
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):
    items = [items]
  if not isinstance(items, list):
    raise ValueError("API items 구조 확인 필요")
  return (items, int(body.get("totalCount", len(items))))
def get_fcst(nx, ny, base_date=None, base_time=None, retry=3):
  if (base_date is None) != (base_time is None):
    raise ValueError("날짜·시간을 함께 지정")
  now = pd.Timestamp.now(tz="Asia/Seoul")
  candidate = now.floor("h") + pd.Timedelta(minutes=30)
  if candidate > now:
    candidate -= pd.Timedelta(hours=1)
  candidates = (
    [(base_date, base_time)]
    if base_date
    else [
      (
        (candidate - pd.Timedelta(hours=h)).strftime("%Y%m%d"),
        (candidate - pd.Timedelta(hours=h)).strftime("%H%M"),
      )
      for h in range(6)
    ]
  )
  for date, hour in candidates:
    items, _ = request_items(
      FCST_URL,
      {
        "base_date": date,
        "base_time": hour,
        "nx": nx,
        "ny": ny,
        "dataType": "JSON",
        "numOfRows": 1000,
        "pageNo": 1,
        "authKey": API_KEY,
      },
    )
    if items:
      return items
  raise ValueError("조회조건·메타데이터를 명시하세요: base_date, base_time")

def parse_fcst(items):
  """요소 행을 예보시각별 열로 변환하고 발표시각을 보존한다."""
  if not items:
    return pd.DataFrame(
      columns=[
        "예보시각",
```

```
        "발표시각",
        "지점",
        "지점명",
        "경도",
        "위도",
        "선행(h)",
        "nx",
        "ny",
        "T1H",
        "RN1",
        "TMP",
        "PCP",
        "SKY",
        "PTY",
        "POP",
        "WSD",
        "하늘",
        "강수형태",
      ]
    )
raw = pd.DataFrame(items)
required = {
    "baseDate",
    "baseTime",
    "fcstDate",
    "fcstTime",
    "category",
    "fcstValue",
    "nx",
    "ny",
}
if required - set(raw.columns):
    raise ValueError(f"예보 응답 필수 열: {sorted(required)}")
keys = ["baseDate", "baseTime", "fcstDate", "fcstTime", "nx", "ny"]
if raw.duplicated(keys + ["category"]).any():
    raise ValueError("중복 예보 요소")
d = raw.pivot(
    index=keys, columns="category", values="fcstValue"
).reset_index()
d["예보시각"] = pd.to_datetime(
    d["fcstDate"] + d["fcstTime"], format="%Y%m%d%H%M"
).dt.tz_localize("Asia/Seoul")
d["발표시각"] = pd.to_datetime(
    d["baseDate"] + d["baseTime"], format="%Y%m%d%H%M"
).dt.tz_localize("Asia/Seoul")
for column in d.columns.intersection(
    [
        "T1H",
        "SKY",
        "PTY",
        "REH",
        "WSD",
        "VEC",
        "UUU",
        "VVV",
        "LGT",
        "POP",
        "WAV",
        "TMN",
        "TMX",
    ]
):
    d[column] = pd.to_numeric(d[column], errors="coerce")
if "RN1" in d:
    d["RN1원문"] = d["RN1"]
    d["RN1"] = d["RN1"].map(rn1_mm)
d["하늘"] = d["SKY"].map(SKY) if "SKY" in d else ""
d["강수형태"] = d["PTY"].map(PTY) if "PTY" in d else ""
d["선행(h)"] = (d["예보시각"] - d["발표시각"]).dt.total_seconds() / 3600
d["지점"] = d["nx"].astype(str) + "," + d["ny"].astype(str)
d["지점명"] = "격자(" + d["지점"] + ")"
d["경도"], d["위도"] = forecast_coordinates(d["nx"], d["ny"])
return d
```

날씨예보

❷ 여섯 시간 예보 시계열

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
    """parse 함수의 CSV 출력 읽기. 연도와 발표시각을 파일에 보존한다."""
    d = pd.read_csv(path, encoding="utf-8-sig")
    for c in ["예보시각", "발표시각"]:
        if c not in d:
            raise ValueError("parse 함수로 생성한 CSV를 입력")
        d[c] = pd.to_datetime(d[c], utc=True).dt.tz_convert("Asia/Seoul")
    d["선행(h)"] = (d["예보시각"] - d["발표시각"]).dt.total_seconds() / 3600
    return d
```

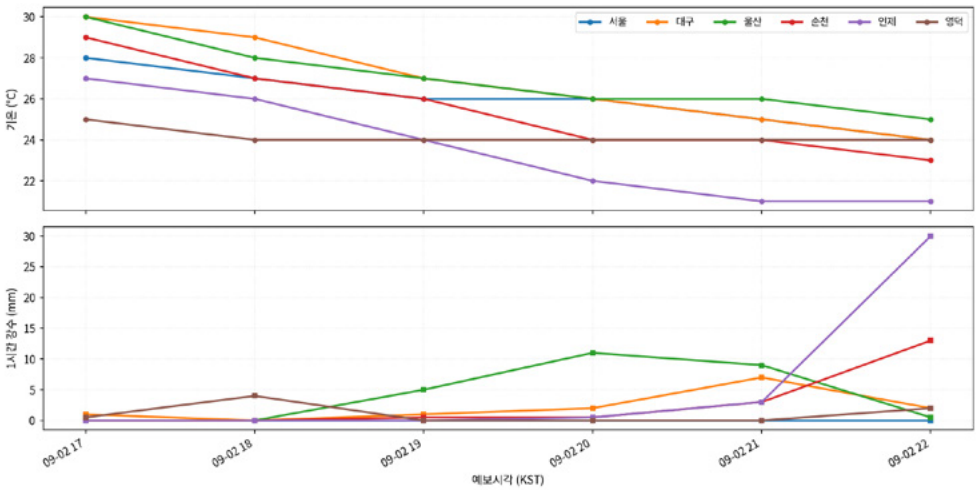


그림 20-1-1. 6시간 예보 시계열

❸ 강수량 문구의 수치 변환

일관된 규칙으로 변환합니다.

```
def rn1_mm(v):
    """RN1의 강수없음·수치·구간 문구를 시각화용 수치로 변환합니다. 구간 하한은 참고값이며 안전 판단에 사용하지 않습니다. 원문도 보존합니다."""

    if v is None or not str(v).strip():
        return np.nan
    s = str(v).strip()
    if "강수없음" in s or s in ("0", "0.0"):
        return 0.0
    if "미만" in s:
        n = re.findall("[\\d.]+", s)
        return float(n[0]) / 2 if n else 0.5
    n = re.findall("[\\d.]+", s)
    if not n:
        return np.nan
    return float(n[0])
def rn1_codes(d):
    """RN1 로 실제로 오는 문구를 세어 본다. 이것이 곧 파싱 규칙표가 된다."""
    t = (
        d["RN1원문"]
        .value_counts()
        .rename_axis("RN1 원문")
        .reset_index(name="칸 수")
    )
    t["숫자로"] = t["RN1 원문"].map(rn1_mm)
    return t
```

RN1 원문	칸 수	숫자로
강수없음	528	0
1mm 미만	28	0.5
2.0mm	5	2
13.0mm	1	13
30.0~50.0mm	1	30
19.0mm	1	19

❹ 시각별 강수 분포 비교

수치로 변환한 강수량으로 시각별 지도를 만듭니다.

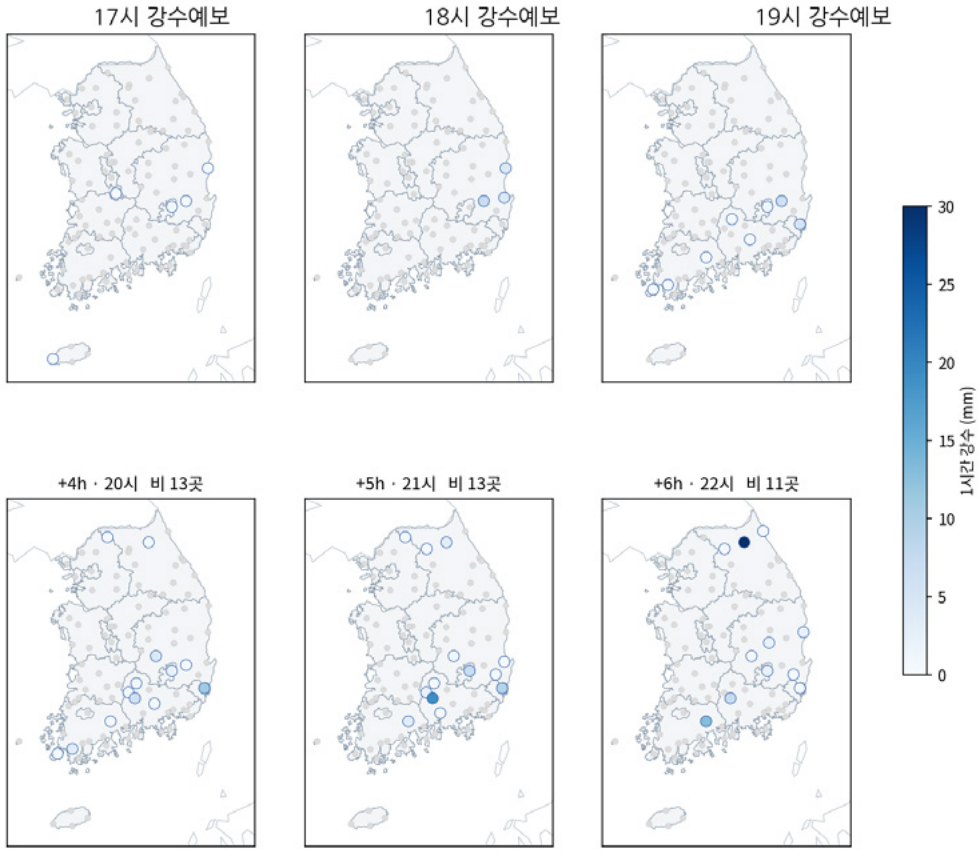


그림 20-1-2. 시각별 강수 분포 비교

날씨예보

❺ 첫 강수 예보 시각 산출

```
def onset(d):
  """격자별로 첫 강수 예보시각과 발표시각으로부터의 선행시간을 계산합니다."""

  rows = []
  for (stn, nm, lon, lat), g in d.groupby(["지점", "지점명", "경도", "위도"]):
    g = g.sort_values("예보시각")
    wet = g[(g["PTY"] > 0) | (g["RN1"] > 0)]
    rows.append(
      {
        "지점": stn,
        "지점명": nm,
        "경도": lon,
        "위도": lat,
        "비 오는 칸": int(len(wet)),
        "첫 강수(h)": float(wet["선행(h)"].min()) if len(wet) else np.nan,
        "최대 RN1": float(g["RN1"].max()),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "지점",
        "지점명",
        "경도",
        "위도",
        "비 오는 칸",
        "첫 강수(h)",
        "최대 RN1",
      ]
    ),
  )
```

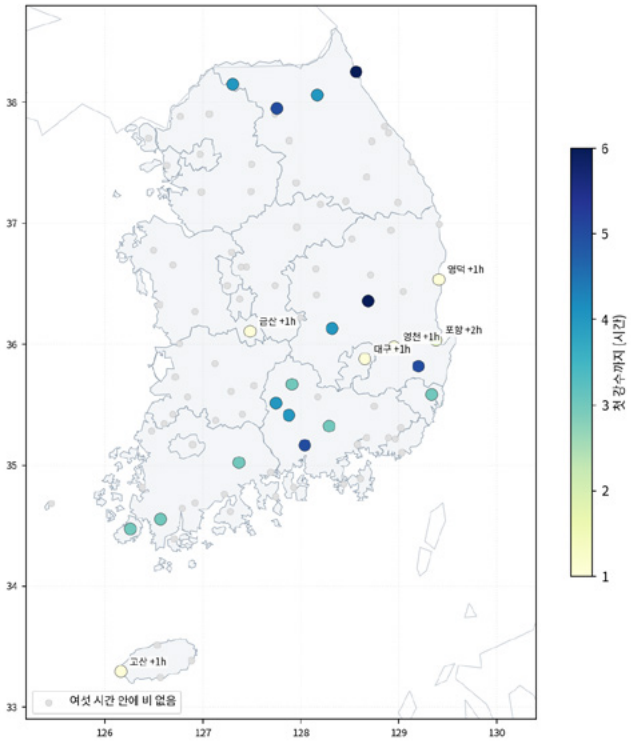


그림 20-1-3. 첫 강수 예보 시각 산출

지점명	첫 강수(h)	비 오는 칸	최대 RN1
대구	1	5	7
고산	1	1	0.5
금산	1	1	0.5
함양군	4	2	2
철원	4	2	0.5
인제	4	3	30

❻ 하늘상태·강수형태별 아이콘

```
def sky_cross(d):
    """하늘상태와 강수형태의 조합을 집계합니다. 아이콘은 이 조합으로 정해진다."""
    t = pd.crosstab(d["하늘"], d["강수형태"])
    return t.reset_index()
```

하늘	소나기	없음
구름많음	16	177
맑음	3	59
흐림	35	292

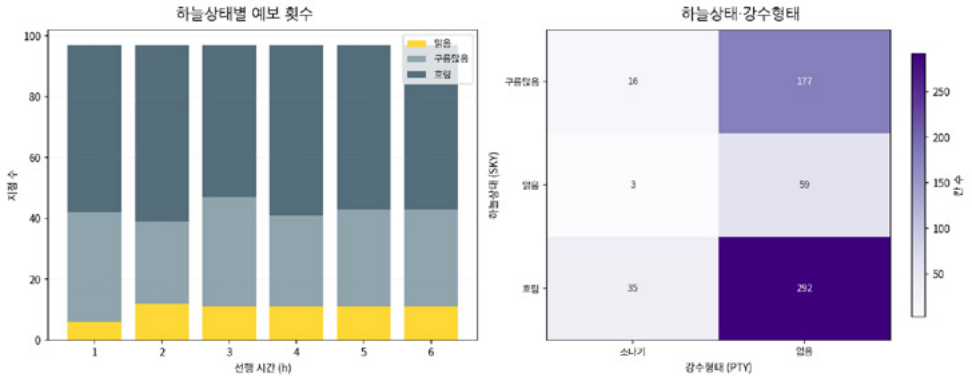


그림 20-1-4. 하늘상태·강수형태별 아이콘

날씨예보

20-②. 초단기예보

기온·강수·바람의 공간 분포 비교

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 단기예보

❶ 초단기예보 격자자료 조회

쉼표 구분값에 위경도를 연결해 전국 지도를 작성합니다. 초단기예보 격자자료는 10분 간격 발표시각을 사용합니다.

```
https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-dfs_vsrt_grd
 ?tmfc={최근 10분 발표시각}&vars=T1H&authKey=...
-99.00, -99.00, -99.00, -99.00, -99.00, -99.00, …
…, 24.00, 24.00, 23.00, 23.00, 22.00, -99.00, …
# 값이 37,697개 = 149 × 253 · -99 는 예보 구역 밖
# vars 를 바꾸면 T1H·RN1·SKY·PTY·REH·UUU·VVV 가 같은 판으로 온다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
GRD_URL = "https://apihub.kma.go.kr/api/typ01/cgi-bin/url/nph-dfs_vsrt_grd"
```

CITIES : 도시 표본 목록. 좌표를 쓰는 함수는 (이름,경도,위도), 격자는 (이름,nx,ny)
DE : 각도→라디안 배율(pi/180)
GRID : 단기예보 격자간격(km)
KO : load_regions 반환 객체. EPSG:4326 경계 GeoJSON에서 생성
MISS : 공식 결측값 또는 처리경계
NX : 열 수
NY : 행 수
OLAT : 기준위도(도)
OLON : 기준경도(도)
PTY_KO : 공식 강수형태 코드별 표시명 dict
RE : 단기예보 도법의 지구반경(km)
SKY_KO : 공식 하늘상태 코드별 표시명 dict
SLAT1 : 제1 표준위도(도)
SLAT2 : 제2 표준위도(도)

VARS : 입력에 있는 요소명 목록
VAR_KO : 요소명별 (표시명,단위) dict
XO : 기준점 x 번호
YO : 기준점 y 번호
nx : 열 수 또는 API의 x 격자번호(함수 정의에 따름)
ny : 행 수 또는 y 격자번호
필수 입력: CITIES, DE, GRID, KO, MISS, NX, NY, OLAT, OLON, PTY_KO, RE, SKY_KO, SLAT1, SLAT2, VARS, VAR_KO, XO, YO, nx, ny. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.
가공파일 생성 예: text = walk_grid(vars_=["T1H"], MISS=-99, nx=149, ny=253, CITIES=[]); Path("grid.txt").write_text(text, encoding="utf-8"); fields, sr = read_pack("grid.txt", CITIES=[], NX=149, NY=253). 통계에는 요소명이 보존되며 series_table은 기온(T1H)만 요약합니다. 동일 시각의 최고–최저는 공간기온차입니다.

```
def rle_decode(text, max_cells=10000000):
    """rle_encode의 count*value / value / x(결측) 형식을 복원한다."""
    values = []
    for token in text.split():
        parts = token.split("*")
        if len(parts) == 2:
            count = int(parts[0])
            word = parts[1]
        elif len(parts) == 1:
            count = 1
            word = parts[0]
        else:
            raise ValueError("RLE token 형식")
        if count <= 0 or len(values) + count > max_cells:
            raise ValueError("RLE 길이")
        value = np.nan if word == "x" else float(word)
        values.extend([value] * count)
    return np.asarray(values, dtype=float)
def load_regions(path, name_field="name"):
    """EPSG:4326 시도 경계 GeoJSON. name_field는 실제 속성 열 이름."""
    import json
    from shapely.geometry import shape, Point


    payload = json.loads(Path(path).read_text("utf-8"))
    features = payload.get("features", [])
    polygons = []
    for feature in features:
        properties = feature.get("properties", {})
        if name_field not in properties:
            raise ValueError("GeoJSON 시도명 속성 열 확인")
        geometry = shape(feature["geometry"])
        if not geometry.is_valid:
            raise ValueError("유효하지 않은 행정경계")

        polygons.append((properties[name_field], geometry))

    def assign(lon, lat):
        answer = []
        for x, y in zip(lon, lat):
            if not np.isfinite(x) or not np.isfinite(y):
                answer.append(None)
                continue
            point = Point(float(x), float(y))
            matches = [name for name, geom in polygons if geom.covers(point)]

            answer.append(matches[0] if len(matches) == 1 else None)

        return answer

    from types import SimpleNamespace

    return SimpleNamespace(assign_province=assign)
```

날씨예보

시·도별 집계에는 EPSG:4326 좌표계의 행정경계 GeoJSON과 shapely가 필요합니다. load_regions()의 name_field에 시·도명 속성을 지정하고 반환 객체를 KO에 전달합니다. 경계가 중복되거나 행정구역이 확인되지 않는 위치는 미분류로 처리합니다.

```
def get_grid(tmfc, var):
  """한 시각·한 요소의 전국 격자자료를 조회합니다. 149×253 격자의 값이 쉼표로 구분되어 반환됩니다."""

  r = requests.get(
    GRD_URL,
    params={"tmfc": tmfc, "vars": var, "authKey": API_KEY},
    timeout=180,
  )
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  return r.text
def parse_grid(text, nx=None, ny=None, *, MISS):
  if nx is None or ny is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: nx, ny")
  "쉼표로 끊어 숫자만 남기고 격자 모양으로 접는다.\n    예보 구역 밖은 -99 로 채워져 있으므로 그것을 결측으로 돌린다."
  v = np.array(
    [float(x) for x in text.replace("\n", " ").split(",") if x.strip()],
    dtype=float,
  )
  v[v <= MISS] = np.nan
  if v.size != nx * ny:
    raise ValueError("응답 값 개수와 격자 크기가 다릅니다")
  return v.reshape(ny, nx)

def walk_grid(tmfc=None, vars_=None, pause=0.2, *, MISS, nx, ny, CITIES=()):
  """각 요소 격자와 통계 블록을 함께 생성. CITIES는 (이름,x,y) 목록이다."""
  if not vars_:
    raise ValueError("조회 요소 vars_를 지정하세요")
  if tmfc is None:
    now = pd.Timestamp.now(tz="Asia/Seoul") - pd.Timedelta(minutes=20)
    tmfc = now.floor("10min").strftime("%Y%m%d%H%M")
  out = []
  for variable in vars_:
    a = parse_grid(get_grid(tmfc, variable), nx=nx, ny=ny, MISS=MISS)
    valid = a[np.isfinite(a)]
    cells = []
    for name, x, y in CITIES:
      if not (1 <= x <= nx and 1 <= y <= ny):
        raise ValueError("CITIES 격자 번호 범위 오류")
      cells.append(str(float(a[int(y) - 1, int(x) - 1])))
    mean, minimum, maximum = (
      (float(valid.mean()), float(valid.min()), float(valid.max()))
      if valid.size
      else (np.nan, np.nan, np.nan)
    )
    out += [
      f"@F {variable} tm={tmfc} n={a.size}",
      rle_encode(a.ravel()),
      f"@S {variable} tm={tmfc} n={valid.size} mean={mean} min={minimum} max={maximum} cells={','.
join(cells)}",
    ]
    time.sleep(pause)
  return "\n".join(out)
def rle_encode(a):
  """같은 값이 길게 이어지므로 「횟수*값」으로 눌러 둔다. x 는 예보 구역 밖."""
  out, prev, cnt = ([], None, 0)
  for v in a:
    t = "x" if not np.isfinite(v) else f"{float(v):.17g}"
    if t == prev:
      cnt += 1
    else:
      if prev is not None:
        out.append(f"{cnt}*{prev}" if cnt > 1 else prev)
      prev, cnt = (t, 1)
  out.append(f"{cnt}*{prev}" if cnt > 1 else prev)
  return " ".join(out)
```

❷ 격자 좌표의 위경도 역변환

응답에는 값만 있으므로 격자 정의를 이용해 위치를 계산합니다. 5 km 람베르트 격자를 역변환하고 유효값이 있는 격자를 표시해 예보 영역을 확인합니다.

```
def xy_to_lonlat(x, y, *, DE, GRID, OLAT, OLON, RE, SLAT1, SLAT2, XO, YO):
    """단기예보 격자 번호를 위경도로. 격자 5 km · 표준위도 30·60 ·
    원점 126 E 38 N · 기준격자 (43, 136) 은 이 격자의 정의값이다."""
    re = RE / GRID
    s1, s2 = (SLAT1 * DE, SLAT2 * DE)
    sn = math.log(math.cos(s1) / math.cos(s2)) / math.log(
        math.tan(math.pi * 0.25 + s2 * 0.5) / math.tan(math.pi * 0.25 + s1 * 0.5)
    )
    sf = math.tan(math.pi * 0.25 + s1 * 0.5) ** sn * math.cos(s1) / sn
    ro = re * sf / math.tan(math.pi * 0.25 + OLAT * DE * 0.5) ** sn
    xn = np.asarray(x, float) - XO
    yn = ro - (np.asarray(y, float) - YO)
    ra = np.hypot(xn, yn)
    alat = 2.0 * np.arctan((re * sf / ra) ** (1.0 / sn)) - math.pi * 0.5
    alon = np.arctan2(xn, yn) / sn + OLON * DE
    return (alon / DE, alat / DE)
def axes(*, DE, GRID, NX, NY, OLAT, OLON, RE, SLAT1, SLAT2, XO, YO):
    X, Y = np.meshgrid(np.arange(1, NX + 1), np.arange(1, NY + 1))
    return xy_to_lonlat(
        X,
        Y,
        DE=DE,
        GRID=GRID,
        OLAT=OLAT,
        OLON=OLON,
        RE=RE,
        SLAT1=SLAT1,
        SLAT2=SLAT2,
        XO=XO,
        YO=YO,
    )
def grid_table(fields, lon, lat, *, GRID, KO, NX, NY):
    ok = np.isfinite(fields["T1H"])
    rows = [
        {"항목": "격자", "값": f"{NX} × {NY} = {NX * NY:,}칸 · {GRID:.0f} km"},
        {
            "항목": "덮는 범위",
            "값": f"경도 {lon.min():.2f} ~ {lon.max():.2f} · 위도 {lat.min():.2f} ~ {lat.max():.2f}",
        },
        {
            "항목": "값이 있는 칸",
            "값": f"{int(ok.sum()):,}칸 ({100 * float(ok.mean()):.1f} %) · {int(ok.sum()) * GRID ** 2:,.0f} km²",
        },
        {
            "항목": "값이 있는 범위",
            "값": f"경도 {lon[ok].min():.2f} ~ {lon[ok].max():.2f} · 위도 {lat[ok].min():.2f} ~ {lat[ok].max():.2f}",
        },
    ]
    if KO:
        p = np.asarray(KO.assign_province(lon[ok], lat[ok]), dtype=object)
        land = np.array([isinstance(v, str) for v in p])
        rows.append(
            {
                "항목": "시·도 안에 드는 칸",
                "값": f"{int(land.sum()):,}칸 ({100 * float(land.mean()):.0f} %) — 나머지는 바다와 경계 밖",
            }
        )
    return pd.DataFrame(rows)

def city_table(
    fields,
    lon,
    lat,
    *,
    CITIES,
    DE,
    GRID,
    OLAT,
    OLON,
    RE,
    SKY_KO,
    SLAT1,
    SLAT2,
    VAR_KO,
```

날씨예보

```
    XO,
    YO
    ):
    """도시별 격자번호를 위경도로 변환하여 기준 좌표와의 차이를 확인합니다."""
    rows = []
    for nm, x, y in CITIES:
        lo, la = xy_to_lonlat(
            [x],
            [y],
            DE=DE,
            GRID=GRID,
            OLAT=OLAT,
            OLON=OLON,
            RE=RE,
            SLAT1=SLAT1,
            SLAT2=SLAT2,
            XO=XO,
            YO=YO,
        )
        rec = {
            "도시": nm,
            "격자X": x,
            "격자Y": y,
            "경도": round(float(lo[0]), 3),
            "위도": round(float(la[0]), 3),
        }
        for v in ("T1H", "REH", "SKY", "PTY", "RN1"):
            rec[VAR_KO[v][0]] = fields[v][y - 1, x - 1]
        rec["하늘"] = (
            SKY_KO.get(int(rec["하늘상태"]), rec["하늘상태"])
            if np.isfinite(rec["하늘상태"])
            else "-"
        )
        rows.append(rec)
    t = pd.DataFrame(rows)
    return t.drop(columns=["하늘상태"])
```

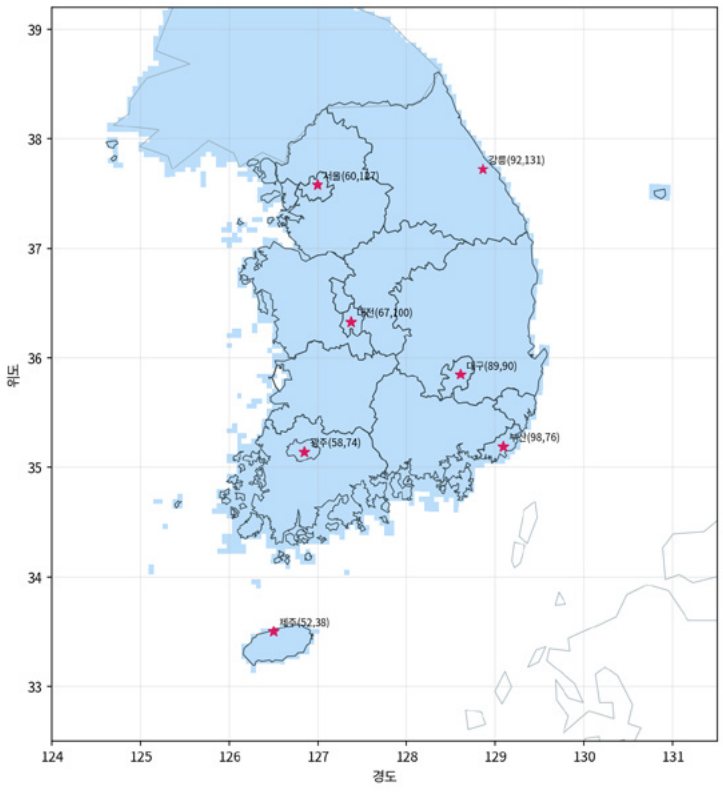


그림 20-2-1. 격자 좌표의 위경도 역변환

항목	값
격자	149 × 253 = 37,697칸 · 5 km
덮는 범위	경도 123.31 ~ 132.77 · 위도 31.65 ~ 43.43
값이 있는 칸	9,295칸 (24.7 %) · 232,375 km
값이 있는 범위	경도 124.31 ~ 131.91 · 위도 32.09 ~ 42.99
시·도 안에 드는 칸	3,805칸 (41 %) — 나머지는 바다와 경계 밖

도시	격자X	격자Y	경도	위도	기온	습도	강수형태	1시간 강수량	하늘
서울	60	127	126.989	37.58	23	90	0	0	맑음
대전	67	100	127.37	36.332	23	95	0	0	구름많음
대구	89	90	128.606	35.847	24	90	4	0	흐림
광주	58	74	126.841	35.144	25	95	0	0	흐림
부산	98	76	129.085	35.191	26	80	0	0	구름많음
강릉	92	131	128.859	37.729	22	95	0	0	맑음
제주	52	38	126.492	33.501	25	85	0	0	구름많음

❸ 기상요소별 공간 분포

여섯 요소를 나란히 표시해 전국 예보를 비교합니다.

```
def field_table(fields, *, VARS, VAR_KO):
  rows = []
  for v in VARS:
    a = fields[v]
    g = a[np.isfinite(a)]
    rows.append(
      {
        "부호": v,
        "뜻": VAR_KO[v][0],
        "단위": VAR_KO[v][1],
        "최소": round(float(g.min()), 1),
        "중앙값": round(float(np.median(g)), 1),
        "최대": round(float(g.max()), 1),
        "값 종류 수": int(len(np.unique(g))),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["부호", "뜻", "단위", "최소", "중앙값", "최대", "값 종류 수"]
    ),
  )
```

날씨예보

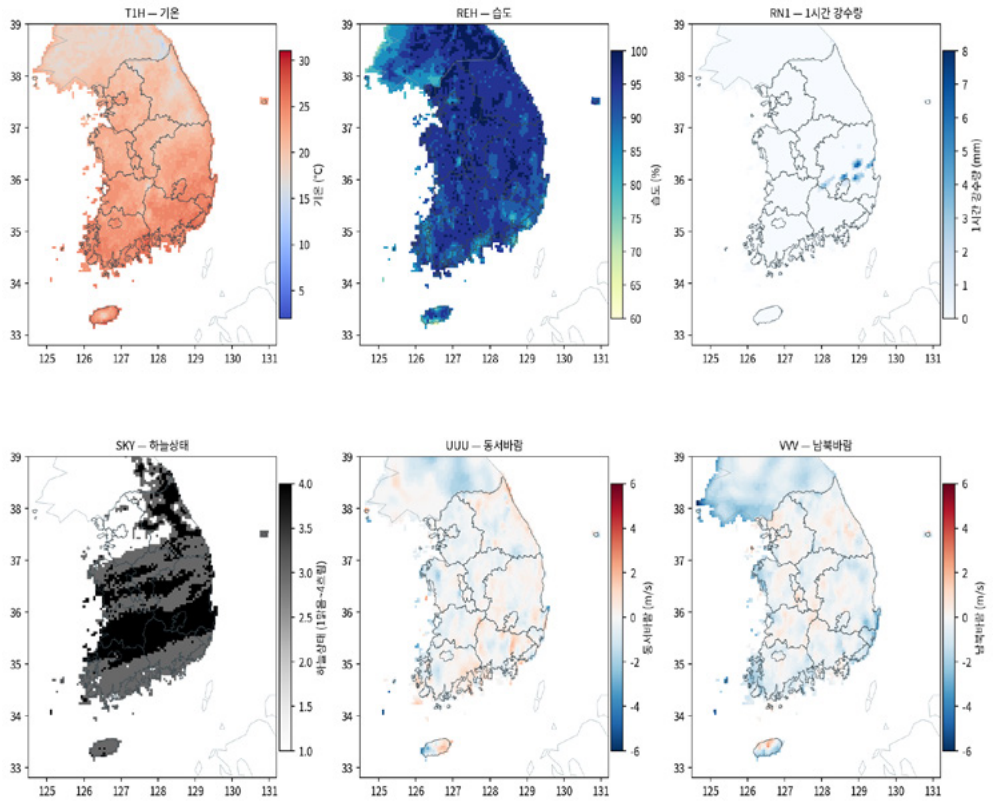


그림 20-2-2. vars별 여섯 요소 분포

부호	뜻	단위	최소	중앙값	최대	값 종류 수
T1H	기온	℃	2	20	31	28
RN1	1시간 강수량	mm	0	0	7	8
SKY	하늘상태	1맑음~4흐림	1	3	4	3
PTY	강수형태	코드	0	0	5	4
REH	습도	%	60	95	100	9
UUU	동서바람	m/s	-5.5	-0.1	3.1	70
VVV	남북바람	m/s	-6.2	-0.4	2.7	75

❹ 바람 성분으로 풍향·풍속 지도 구성

풍향별 격자 수를 집계해 전국 바람 분포도 비교합니다.

```
def wind(fields):
  """동서·남북 바람 성분을 풍속과 풍향으로 변환합니다."""

  u, v = (fields["UUU"], fields["VVV"])
  sp = np.hypot(u, v)
  wd = (np.degrees(np.arctan2(-u, -v)) + 360) % 360
  return (u, v, sp, wd)
def wind_table(fields, lon, lat):
  _, _, sp, wd = wind(fields)
  ok = np.isfinite(sp)
  d = pd.DataFrame({"풍속": sp[ok], "풍향": wd[ok], "위도": lat[ok]})
  d["방위"] = pd.cut(
    (d["풍향"] + 22.5) % 360,
    np.arange(0, 361, 45),
    labels=["북", "북동", "동", "남동", "남", "남서", "서", "북서"],
    right=False,
  )
  t = (
    d.groupby("방위", observed=True)
    .agg(칸수=("풍속", "size"), 평균풍속=("풍속", "mean"))
    .round(2)
  )
  t["비율(%)"] = (100 * t["칸수"] / t["칸수"].sum()).round(1)
  return t.reset_index()
```

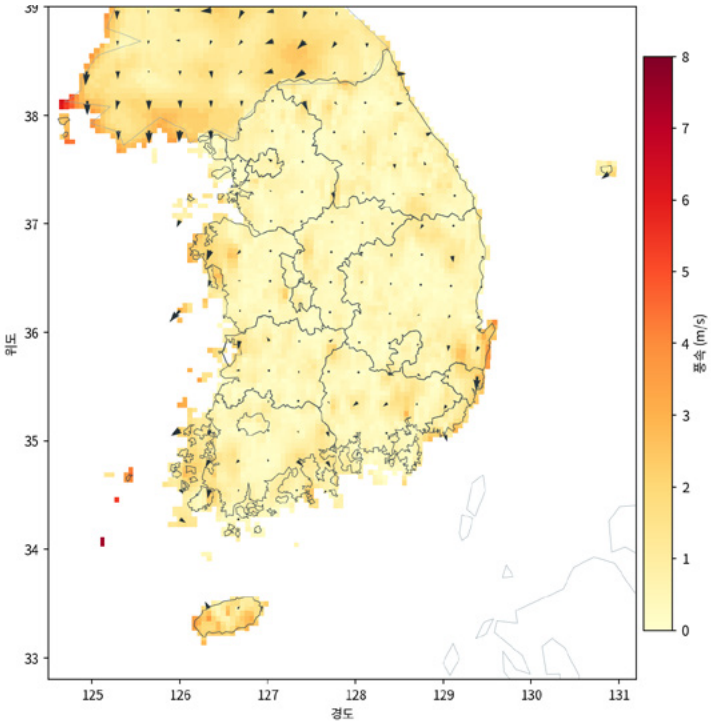


그림 20-2-3. 바람 성분으로 풍향·풍속 지도 구성

날씨예보

방위	칸수	평균풍속	비율(%)
북	2905	1.34	31.3
북동	1600	1.09	17.2
동	1596	0.89	17.2
남동	409	0.48	4.4
남	509	0.31	5.5
남서	325	0.53	3.5
서	836	0.93	9
북서	1115	1.09	12

❺ 하늘상태·강수형태 교차 집계

두 코드를 교차 집계해 구름이 많지만 강수는 없는 격자 수 등을 확인합니다.

```
def state_table(fields, *, PTY_KO, SKY_KO):
    sky, pty = (fields["SKY"], fields["PTY"])
    ok = np.isfinite(sky)
    d = pd.DataFrame({"하늘": sky[ok].astype(int), "강수": pty[ok].astype(int)})
    t = pd.crosstab(d["하늘"].map(SKY_KO), d["강수"].map(PTY_KO))
    t["합계"] = t.sum(axis=1)
    t["비율(%)"] = (100 * t["합계"] / t["합계"].sum()).round(1)
    return t.reset_index()
```

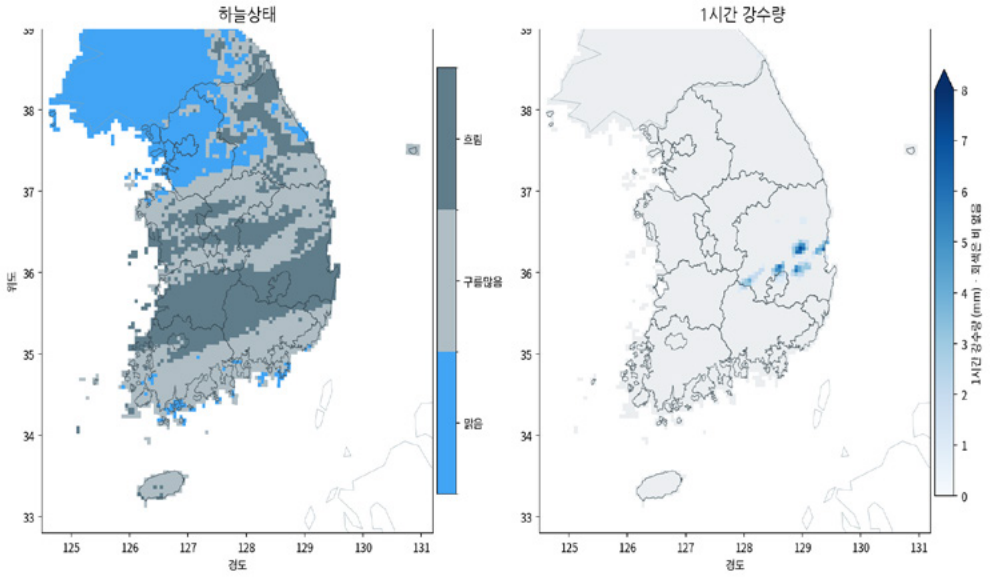


그림 20-2-4. 하늘상태·강수형태 교차 집계

하늘	비	빗방울	소나기	없음	합계	비율(%)
구름많음	5	7	49	2635	2696	29
맑음	1	1	0	4145	4147	44.6
흐림	69	8	193	2182	2452	26.4

❻ 시간별 기온 변화와 도시별 비교

전국 평균·최저·최고를 집계하고 도시별 격자를 추출해 전국과 도시의 기온 시계열을 만듭니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path, *, CITIES, NX, NY):
    """눌러 둔 판들과 시각별 요약을 읽습니다."""
    lines = Path(path).read_text("utf-8").splitlines()
    fields, series, i = ({}, [], 0)
    while i < len(lines):
        ln = lines[i]
        if ln.startswith("@F"):
            nm = ln.split()[1]
            fields[nm] = rle_decode(lines[i + 1]).reshape(NY, NX)
            i += 2
            continue
        if ln.startswith("@S"):
            h = dict((kv.split("=") for kv in ln.split()[2:] if "=" in kv))
            series.append(
                {
                    "요소": ln.split()[1],
                    "시각": datetime.strptime(h["tm"], "%Y%m%d%H%M").replace(
                        tzinfo=KST
                    ),
                    "칸수": int(h["n"]),
                    "평균": float(h["mean"]),
                    "최저": float(h["min"]),
                    "최고": float(h["max"]),
                    **{
                        nm: float(v)
                        for (nm, *_), v in zip(
                            CITIES,
                            (
                                h.get("cells", "").split(",")
                                if h.get("cells")
                                else []
                            ),
                        )
                    },
                }
            )
        i += 1
    cols = ["요소", "시각", "칸수", "평균", "최저", "최고"] + [
        c[0] for c in CITIES
    ]
    result = pd.DataFrame(series, columns=cols)
    result["시각"] = pd.to_datetime(result["시각"])
    return fields, result.sort_values("시각")
def series_table(sr, *, CITIES):
    t = sr[sr["요소"].eq("T1H")].copy() if "요소" in sr else sr.copy()
    t["시각"] = t["시각"].dt.strftime("%m-%d %H시")
    cols = ["시각", "평균", "최저", "최고"] + [n for n, *_ in CITIES]
    t["공간기온차"] = (t["최고"] - t["최저"]).round(1)
    return t[cols + ["공간기온차"]].round(2)
```

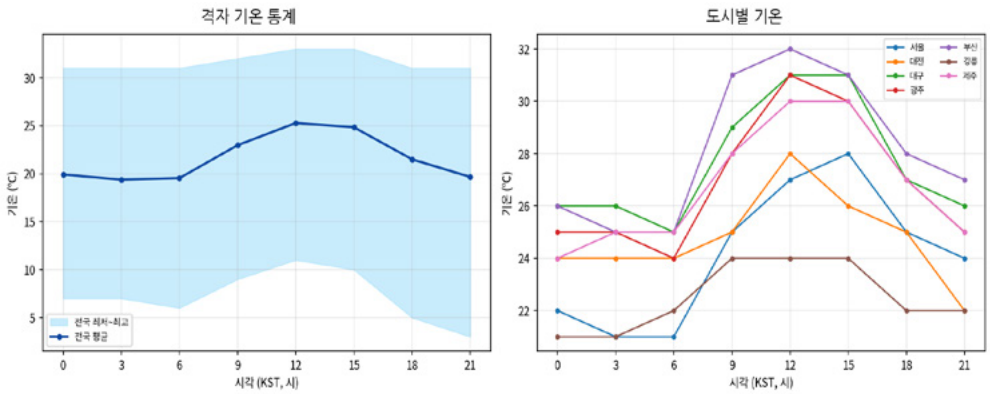


그림 20-2-5. 시간별 기온 변화와 도시별 비교

날씨예보

시각	평균	최저	최고	서울	대전	대구	광주	부산	강릉	제주	일교차
09-02 00시	19.92	7	31	22	24	26	25	26	21	24	24
09-02 03시	19.37	7	31	21	24	26	25	25	21	25	24
09-02 06시	19.54	6	31	21	24	25	24	25	22	25	25
09-02 09시	22.97	9	32	25	25	29	28	31	24	28	23
09-02 12시	25.28	11	33	27	28	31	31	32	24	30	22
09-02 15시	24.84	10	33	28	26	31	30	31	24	30	23
09-02 18시	21.5	5	31	25	25	27	27	28	22	27	26
09-02 21시	19.66	3	31	24	22	26	25	27	22	25	28

21.
단기예보

야외활동 시간대별 날씨 비교

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 단기예보

❶ 단기예보 조회

```
https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getVilageFcst
 ?base_date=20260902&base_time=1400&nx=60&ny=127&dataType=JSON&numOfRows=1200&a
uthKey=...
{"category":"TMP","fcstDate":"20260903","fcstTime":"1500","fcstValue":"27","nx":60,"ny":127}
{"category":"POP", … ,"fcstValue":"30"}
{"category":"PCP", … ,"fcstValue":"강수없음"}
{"category":"TMX","fcstDate":"20260903","fcstTime":"1500","fcstValue":"27.0"}
#  PCP 는 숫자가 아니라 문구 — '강수없음' '1mm 미만' '4.0mm'
#  TMN·TMX 는 하루에 한 칸(06시·15시)에만 실려 온다
#  SKY : 1 맑음 · 3 구름많음 · 4 흐림 / PTY : 0 없음 · 1 비 · 4 소나기
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
FCST_URL = "https://apihub.kma.go.kr/api/typ02/openApi/VilageFcstInfoService_2.0/getVilageFcst"
PTY = {0:"없음",1:"비",2:"비/눈",3:"눈",4:"소나기",5:"빗방울",6:"빗방울/눈날림",7:"눈날림"}
SKY = {1:"맑음",3:"구름많음",4:"흐림"}
CATS = ["TMP", "UUU", "VVV", "VEC", "WSD", "SKY", "PTY", "POP", "WAV", "PCP", "REH", "SNO", "TMN",
"TMX"]
def forecast_coordinates(nx, ny):
    """단기예보 5km 격자 중심의 (경도, 위도). 공식 격자 도법을 적용한다."""
    re = 6371.00877 / 5.0
    rad = math.pi / 180.0
    s1, s2 = 30.0 * rad, 60.0 * rad
    sn = math.log(math.cos(s1) / math.cos(s2)) / math.log(
        math.tan(math.pi / 4 + s2 / 2) / math.tan(math.pi / 4 + s1 / 2)
    )
    sf = math.tan(math.pi / 4 + s1 / 2) ** sn * math.cos(s1) / sn
    ro = re * sf / math.tan(math.pi / 4 + 38.0 * rad / 2) ** sn
    x = np.asarray(nx, dtype=float) - 43.0
    y = ro - (np.asarray(ny, dtype=float) - 136.0)
    rho = np.hypot(x, y)
    lat = 2 * np.arctan((re * sf / rho) ** (1 / sn)) - math.pi / 2
    lon = np.arctan2(x, y) / sn + 126.0 * rad
    return lon / rad, lat / rad
```

날씨예보

실행 순서: d = save_forecast_csv(60, 127); d = read_pack("forecast.csv"). 분석표의 지점은 관측소 번호가 아니라 "nx,ny" 격자 식별자이며, 경도·위도는 격자 중심 좌표입니다. 다른 격자는 nx·ny를 바꿔 조회하고 pd.concat으로 결합합니다.
시간대 추천 실행 예: best_slots(d, "60,127"). 점수와 강수확률 기준값은 활용 예시용 기준이며 기상청 공식 판정값이 아닙니다.

```
def save_forecast_csv(nx=60, ny=127, path="forecast.csv"):
  d = parse_vilage(get_vilage(nx, ny))
  d.to_csv(path, index=False, encoding="utf-8-sig")
  return d
```

입력 파일은 save_forecast_csv()로 생성합니다. 한 격자 예시(60,127)에서 확인한 뒤 여러 위치로 확장합니다. 전국 도표는 해당 도표의 위치·기간과 동일한 조건이 필요합니다.

```
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 인증·권한·접속정책 확인")
  js = r.json()
  response = js.get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code == "03":
    return ([], 0)
  if code != "00":
    raise RuntimeError(f"API 오류코드 {code}: 제공기간·요청조건 확인")
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):
    items = [items]
  if not isinstance(items, list):
    raise ValueError("API items 구조 확인 필요")
  return (items, int(body.get("totalCount", len(items))))
def get_vilage(nx, ny, base_date=None, base_time=None, retry=3):
  if (base_date is None) != (base_time is None):
    raise ValueError("날짜·시간을 함께 지정")
  now = pd.Timestamp.now(tz="Asia/Seoul")
  times = [
    now.normalize() + pd.Timedelta(hours=h)
    for h in [2, 5, 8, 11, 14, 17, 20, 23]
  ]
  times += [
    now.normalize() - pd.Timedelta(days=1) + pd.Timedelta(hours=h)
    for h in [2, 5, 8, 11, 14, 17, 20, 23]
  ]
  candidates = (
    [(base_date, base_time)]
    if base_date
    else [
      (t.strftime("%Y%m%d"), t.strftime("%H%M"))
      for t in sorted(times, reverse=True)
      if t <= now
    ]
  )
  for date, hour in candidates:
    all_items, page = ([], 1)
    while True:
      items, total = request_items(
        FCST_URL,
        {
          "base_date": date,
          "base_time": hour,
          "nx": nx,
          "ny": ny,
          "dataType": "JSON",
          "numOfRows": 1000,
          "pageNo": page,
          "authKey": API_KEY,
        },
      )
      if not items:
```

```
            if all_items:
                return all_items  # 마지막 쪽까지 수신
            break
        all_items.extend(items)
        if len(all_items) >= total:
            return all_items
        page += 1
        if page > 100:  # 안전장치: 최대 100쪽
            return all_items
        time.sleep(0.3)
    raise ValueError("조회조건·메타데이터를 명시하세요: base_date, base_time")
def parse_vilage(items):
    """요소 행을 예보시각별 열로 변환하고 발표시각을 보존한다."""
    if not items:
        return pd.DataFrame(
            columns=[
                "예보시각",
                "발표시각",
                "지점",
                "지점명",
                "경도",
                "위도",
                "선행(h)",
                "nx",
                "ny",
                "T1H",
                "RN1",
                "TMP",
                "PCP",
                "SKY",
                "PTY",
                "POP",
                "WSD",
                "하늘",
                "강수형태",
            ]
        )
    raw = pd.DataFrame(items)
    required = {
        "baseDate",
        "baseTime",
        "fcstDate",
        "fcstTime",
        "category",
        "fcstValue",
        "nx",
        "ny",
    }
    if required - set(raw.columns):
        raise ValueError(f"예보 응답 필수 열: {sorted(required)}")
    keys = ["baseDate", "baseTime", "fcstDate", "fcstTime", "nx", "ny"]
    if raw.duplicated(keys + ["category"]).any():
        raise ValueError("중복 예보 요소")
    d = raw.pivot(
        index=keys, columns="category", values="fcstValue"
    ).reset_index()
    d["예보시각"] = pd.to_datetime(
        d["fcstDate"] + d["fcstTime"], format="%Y%m%d%H%M"
    ).dt.tz_localize("Asia/Seoul")
    d["발표시각"] = pd.to_datetime(
        d["baseDate"] + d["baseTime"], format="%Y%m%d%H%M"
    ).dt.tz_localize("Asia/Seoul")
    for column in d.columns.intersection(
        [
            "TMP",
            "SKY",
            "PTY",
            "REH",
            "WSD",
            "VEC",
            "UUU",
            "VVV",
            "LGT",
            "POP",
            "WAV",
            "TMN",
            "TMX",
        ]
    ):
        d[column] = pd.to_numeric(d[column], errors="coerce")
    if "PCP" in d:
```

날씨예보

```
    d["PCP원문"] = d["PCP"]
    d["PCP"] = d["PCP"].map(pcp_mm)
  d["하늘"] = d["SKY"].map(SKY) if "SKY" in d else ""
  d["강수형태"] = d["PTY"].map(PTY) if "PTY" in d else ""
  d["선행(h)"] = (d["예보시각"] - d["발표시각"]).dt.total_seconds() / 3600
  d["지점"] = d["nx"].astype(str) + "," + d["ny"].astype(str)
  d["지점명"] = "격자(" + d["지점"] + ")"
  d["경도"], d["위도"] = forecast_coordinates(d["nx"], d["ny"])
  return d
```

❷ 시간별 기온·강수확률 비교

시각을 행, 요소를 열로 정리해 기온과 강수확률을 그립니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
    """parse 함수의 CSV 출력 읽기. 연도와 발표시각을 파일에 보존한다."""
    d = pd.read_csv(path, encoding="utf-8-sig")
    for c in ["예보시각", "발표시각"]:
        if c not in d:
            raise ValueError("parse 함수로 생성한 CSV를 입력")
        d[c] = pd.to_datetime(d[c], utc=True).dt.tz_convert("Asia/Seoul")
    d["선행(h)"] = (d["예보시각"] - d["발표시각"]).dt.total_seconds() / 3600
    return d
```

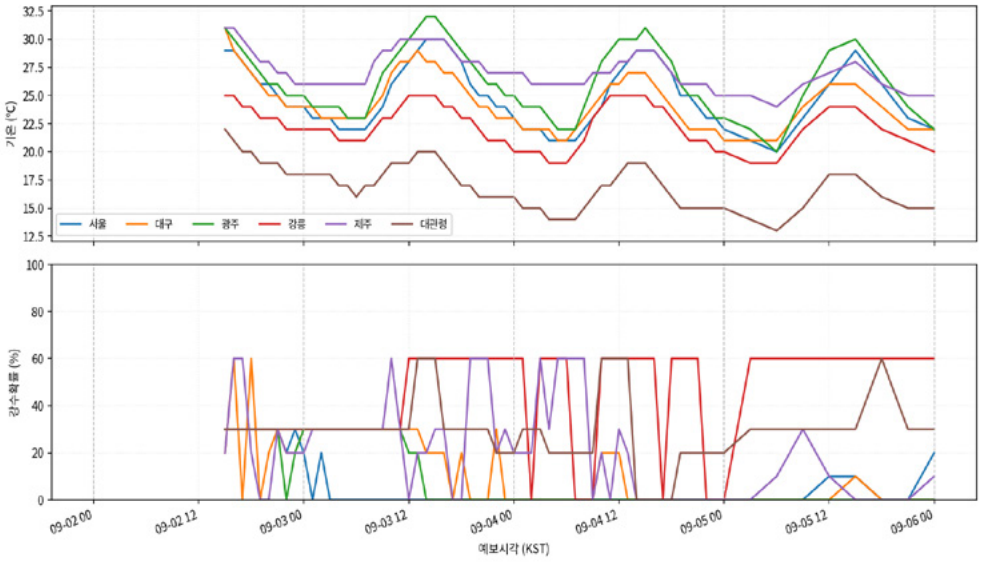


그림 21-1. 시간별 기온·강수확률 비교

❸ 일 최저·최고기온 비교

지점·날짜별로 모아 일별 예보 카드에 표시합니다.

```
def minmax_frame(d):
  """실제 TMN·TMX 예보의 날짜별 카드. 미제공 값을 임의 보간하지 않는다."""
  if d.empty:
    return pd.DataFrame(
      columns=["지점", "지점명", "nx", "ny", "날짜", "최저", "최고"]
    )
  g = d.copy()
  g["날짜"] = g["예보시각"].dt.date
  identity = [c for c in ["지점", "지점명", "nx", "ny", "날짜"] if c in g]
  values = [c for c in ["TMN", "TMX"] if c in g]
  if not values:
    return pd.DataFrame(columns=identity + ["최저", "최고"])
  return (
    g.groupby(identity)[values]
    .first()
    .reset_index()
    .rename(columns={"TMN": "최저", "TMX": "최고"})
  )
```

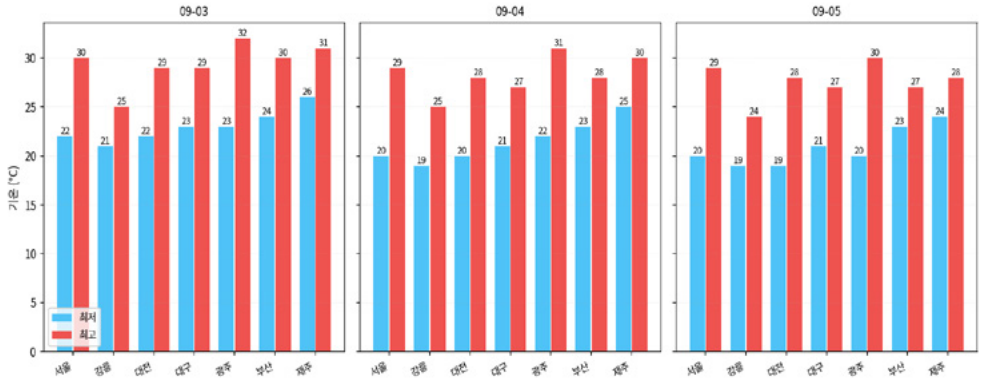


그림 21-2. 일 최저·최고 기온 카드

지점명	날짜	최저	최고	일교차
영월	09월 05일	17	29	12
원주	09월 05일	17	29	12
영월	09월 04일	18	29	11
홍천	09월 04일	17	28	11
정선군	09월 03일	18	29	11
이천	09월 05일	17	28	11

날씨예보

❹ 시각별 강수확률 지도

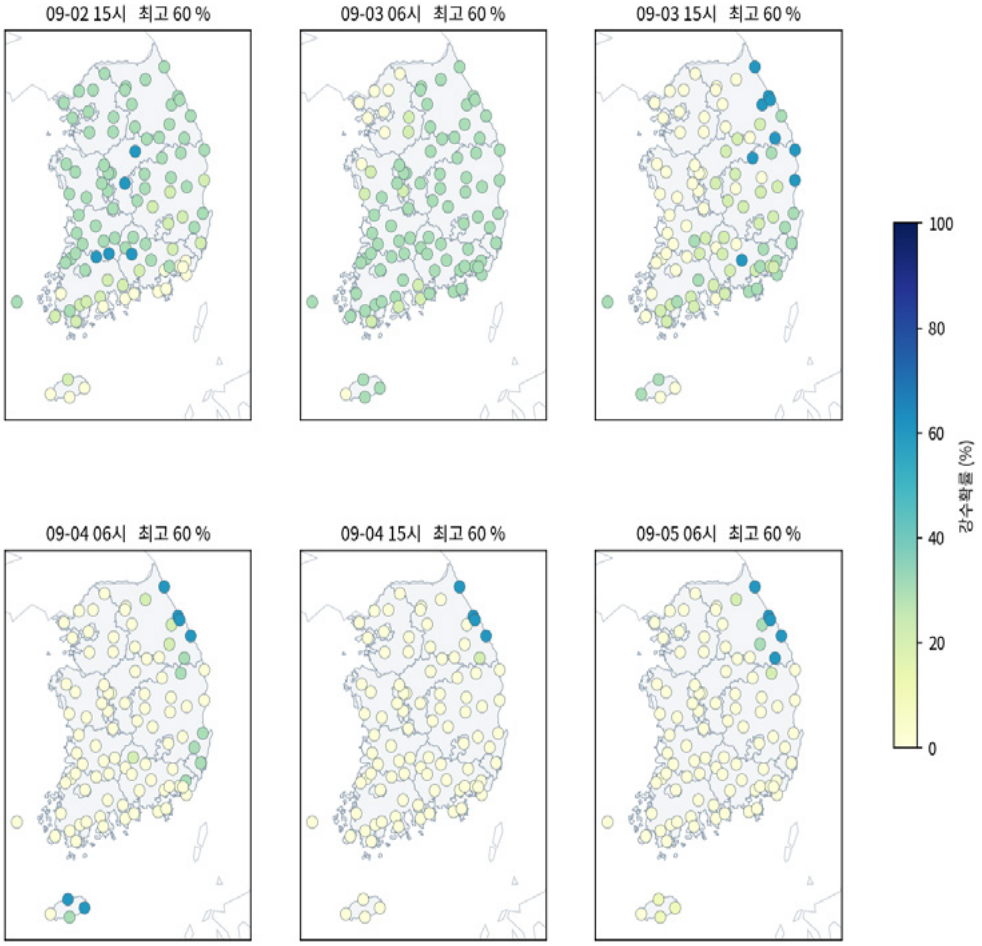


그림 21-3. 시각별 강수확률 지도

지점명	날짜	최고POP	강수합	비오는칸	최대풍속
동해	09월 03일	70	5	10	4.7
속초	09월 06일	60	1	1	1
거창	09월 02일	60	14	1	0.5
남원	09월 02일	60	9	1	0.4
남원	09월 03일	60	18	1	2.7
북강릉	09월 04일	60	11.5	20	3.6

❺ 강수확률·강수량 기반 우산 알림

확률별 강수량 예보 여부를 확인하고 우산 알림 기준을 정합니다.

```
def pcp_mm(v):
    """PCP의 강수없음·수치·구간 문구를 시각화용 수치로 변환합니다. 구간은 하한값을 사용하므로 실제 강수량으로 단정하지 않습니다."""

    if v is None:
        return np.nan
    s = str(v).strip()
    if s in ("", "-", "강수없음", "0", "0.0"):
        return 0.0
    if s.startswith("<"):
        n = re.findall("[\\d.]+", s)
        return float(n[0]) / 2 if n else 0.5
    n = re.findall("[\\d.]+", s)
    if not n:
        return np.nan
    return float(n[0])
def pop_codes(d):
    """PCP 로 실제 오는 문구와 그때의 강수확률을 나란히 집계합니다."""
    t = (
        d.groupby("PCP원문")
        .agg(
            칸수=("POP", "size"),
            평균POP=("POP", "mean"),
            최소POP=("POP", "min"),
            최대POP=("POP", "max"),
        )
        .reset_index()
        .sort_values("칸수", ascending=False)
    )
    t["숫자로(mm)"] = t["PCP원문"].map(pcp_mm)
    t["평균POP"] = t["평균POP"].round(1)
    return t.rename(columns={"PCP원문": "PCP 원문"})
def pop_hit(d, steps=(10, 20, 30, 40, 50, 60, 70, 80)):
    """강수확률 기준값별로 같은 예보의 강수량이 양수인 칸을 집계합니다. 관측과 비교한 예보 적중률이 아니라 예보 요소 간 관계를 요약한다."""

    rows = []
    for p in steps:
        s = d[d["POP"] >= p]
        rows.append(
            {
                "POP 문턱(%)": p,
                "해당 칸": len(s),
                "강수량이 붙은 칸": int((s["PCP"] > 0).sum()),
                "비율(%)": (
                    round(100 * (s["PCP"] > 0).mean(), 1) if len(s) else np.nan
                ),
                "평균 강수(mm)": (
                    round(float(s["PCP"].mean()), 2) if len(s) else np.nan
                ),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "POP 문턱(%)",
                "해당 칸",
                "강수량이 붙은 칸",
                "비율(%)",
                "평균 강수(mm)",
            ]
        ),
    )
```

날씨예보

PCP 원문	칸수	평균POP	최소POP	최대POP	숫자로(mm)
-	6140	10.2	0	30	0
<1	127	60.1	60	70	0.5
1	37	60	60	60	1
4	5	60	60	60	4
8	5	60	60	60	8
9	5	60	60	60	9

POP 문턱(%)	해당 칸	강수량이 붙은 칸	비율(%)	평균 강수(mm)
10	2613	262	10	0.26
20	2544	262	10.3	0.27
30	1914	262	13.7	0.36
40	262	262	100	2.62
50	262	262	100	2.62
60	262	262	100	2.62
70	1	1	100	0.5
80	0	0		

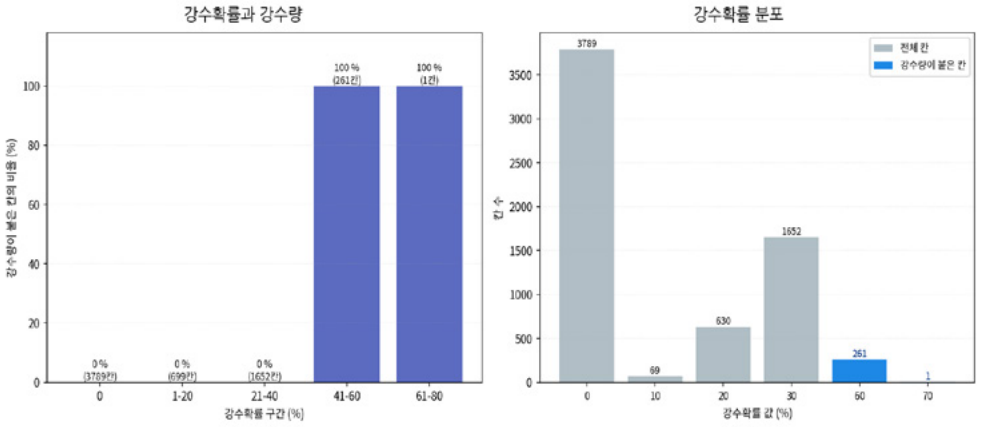


그림 21-4. 강수확률·강수량 기반 우산 알림

❻ 야외활동 참고점수 비교

강수확률·하늘상태·기온·풍속을 점수로 환산해 날짜 × 시각 표에 표시합니다. 점수가 높은 시간대를 야외 일정 후보로 활용합니다. 뒤쪽 이틀은 세 시간 간격이므로 예보가 없는 시각은 회색으로 구분합니다.

```
def outdoor_score(d):
    """강수확률·하늘상태·기온·풍속에 분석용 가중치를 적용해 야외활동 참고점수를 계산합니다."""

    s = 100.0 - d["POP"] * 0.8
    s -= d["SKY"].map({1: 0.0, 3: 5.0, 4: 12.0})
    s -= (d["TMP"] - 22).abs() * 1.5
    s -= (d["WSD"] - 4).clip(lower=0) * 4
    s -= (d["PCP"] > 0) * 20
    return s.clip(lower=0, upper=100)
def best_slots(d, stn, top=10):
    """한 지점에서 점수가 높은 시간대를 추출합니다."""
    if d.empty:
        return pd.DataFrame(
            columns=["예보시각", "TMP", "POP", "하늘", "WSD", "점수"]
        )
    g = d[d["지점"] == stn].copy()
    g["점수"] = outdoor_score(g)
    g = g.sort_values("점수", ascending=False).head(top)
    return g[["예보시각", "TMP", "POP", "하늘", "WSD", "점수"]].assign(
        예보시각=lambda x: x["예보시각"].dt.strftime("%m-%d %H시"),
        점수=lambda x: x["점수"].round(1),
    )
```

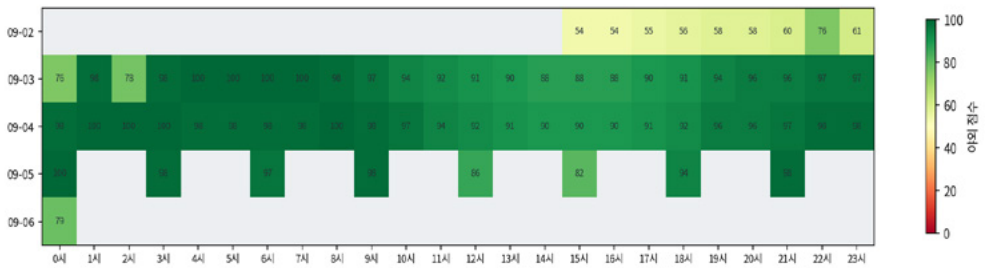


그림 21-5. 기상 점수로 야외 일정 비교

예보시각	TMP	POP	하늘	WSD	점수
09-03 06시	22	0	맑음	0.6	100
09-03 05시	22	0	맑음	0.6	100
09-05 00시	22	0	맑음	0.4	100
09-04 03시	22	0	맑음	0.6	100
09-03 07시	22	0	맑음	0.5	100
09-03 04시	22	0	맑음	0.6	100
09-04 08시	22	0	맑음	1.6	100
09-04 01시	22	0	맑음	0.8	100
09-04 02시	22	0	맑음	0.6	100
09-04 22시	23	0	맑음	0.9	98.5

날씨예보

22. 중기예보

날짜별 기온과 강수확률 비교

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 중기예보

❶ 중기예보 조회

※ 중기예보는 최근 24시간 발표자료만 제공합니다. 실행 시 가장 최근의 06시 또는 18시 발표시각을 tmFc에 입력합니다.

최저·최고 기온과 변동 범위를 비교하고, 예보구역코드(FCT_ID)로 좌표를 연결합니다.

```
https://apihub.kma.go.kr/api/typ02/openApi/MidFcstInfoService/
getMidTa?regId=11B10101&tmFc={최근 06시 또는 18시}&dataType=JSON&authKey=...
{"regId":"11B10101",
 "taMin4":21,"taMin4Low":1,"taMin4High":1,
 "taMax4":28,"taMax4Low":1,"taMax4High":1, … "taMax10":28}
…/getMidLandFcst?regId=11B00000&tmFc={최근 06시 또는 18시} …
{"rnSt4Am":10,"rnSt4Pm":10, … "rnSt10":20,
 "wf4Am":"맑음","wf4Pm":"맑음", … "wf10":"구름많음"}
#  +4~+7일은 오전·오후가 나뉘고 +8~+10일은 하루 한 칸이다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
MID = "https://apihub.kma.go.kr/api/typ02/openApi/MidFcstInfoService"
```

DAYS : 응답 항목에 실제 있는 며칠 뒤 목록
LAND_NAME : 공식 육상 예보구역 코드별 표시명 dict
SKY_SCORE : 일정 비교에 사용자가 정한 하늘상태 점수 dict. 공식 위험등급 아님
TMFC : 입력자료의 실제 생산시각 datetime. KIM은 UTC, 중기예보는 KST
필수 입력: DAYS, LAND_NAME, SKY_SCORE, TMFC. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 인증·권한·접속정책 확인")
  js = r.json()
  response = js.get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code == "03":
    return ([], 0)
  if code != "00":
    raise RuntimeError(f"API 오류코드 {code}: 제공기간·요청조건 확인")
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):
    items = [items]
  if not isinstance(items, list):
    raise ValueError("API items 구조 확인 필요")
  return (items, int(body.get("totalCount", len(items))))
def get_mid(kind, reg_id, tm_fc=None, retry=3):
  if kind not in ["getMidTa", "getMidLandFcst", "getMidSeaFcst"]:
    raise ValueError("서비스 종류 확인")
  now = pd.Timestamp.now(tz="Asia/Seoul")
  if tm_fc is None:
    candidates = [now.normalize() + pd.Timedelta(hours=h) for h in [6, 18]]
    candidates += [
      now.normalize() - pd.Timedelta(days=1) + pd.Timedelta(hours=h)
      for h in [6, 18]
    ]
    candidates = [
      t
      for t in sorted(candidates, reverse=True)
      if pd.Timedelta(0) <= now - t < pd.Timedelta(hours=24)
    ]
  else:
    candidates = [tm_fc]
  for t in candidates:
    tm = t.strftime("%Y%m%d%H%M") if hasattr(t, "strftime") else str(t)
    items, _ = request_items(
      f"{MID}/{kind}",
      {
        "regId": reg_id,
        "tmFc": tm,
        "dataType": "JSON",
        "numOfRows": 100,
        "pageNo": 1,
        "authKey": API_KEY,
      },
    )
    if items:
      return items[0]
  raise ValueError("조회조건·메타데이터를 명시하세요: tm_fc")

def parse_ta(item, tm_fc, *, DAYS):
  """taMin4·taMax4···· 를 날짜로 편다. 뒤의 숫자가 며칠 뒤인지를 뜻한다."""
  rows = []
  for d in DAYS:
    rows.append(
      {
        "구역": item["regId"],
        "며칠뒤": d,
        "날짜": tm_fc.date() + timedelta(days=d),
        "최저기온": item[f"taMin{d}"],
        "최고기온": item[f"taMax{d}"],
        "최저Low": item.get(f"taMin{d}Low"),
        "최저High": item.get(f"taMin{d}High"),
        "최고Low": item.get(f"taMax{d}Low"),
        "최고High": item.get(f"taMax{d}High"),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "구역",
        "며칠뒤",
        "날짜",
        "최저기온",
        "최고기온",
```

날씨예보

```
            "최저Low",
            "최저High",
            "최고Low",
            "최고High",
          ]
        ),
    )
def parse_land(item, tm_fc, *, DAYS):
    """+4~+7일은 오전·오후가 나뉘고 +8~+10일은 하루 한 칸이다."""
    rows = []
    for d in DAYS:
        if d <= 7:
            pair = [("오전", f"{d}Am"), ("오후", f"{d}Pm")]
        else:
            pair = [("종일", f"{d}")]
        for when, key in pair:
            rows.append(
                {
                    "구역": item["regId"],
                    "며칠뒤": d,
                    "때": when,
                    "날짜": tm_fc.date() + timedelta(days=d),
                    "강수확률": item[f"rnSt{key}"],
                    "하늘": item[f"wf{key}"],
                }
            )
    return pd.DataFrame(
        rows,
        columns=(
            None if rows else ["구역", "며칠뒤", "때", "날짜", "강수확률", "하늘"]
        ),
    )
def land_of(fct_id):
    """지점정보의 예보구역코드(FCT_ID) 앞자리로 중기 육상예보 구역을 고른다."""
    if fct_id.startswith("11B"):
        return "11B00000"
    if fct_id.startswith("11G"):
        return "11G00000"
    return fct_id[:4] + "0000"
```

❷ 날짜별 최저·최고 기온 비교

```
def stn_frame(sec, *, DAYS, TMFC):
    rows = []
    for line in sec["STN"]:
        head, mn, mx = line.split("|")
        stn, lon, lat, nm, fct, land = head.split(",")
        for d, a, b in zip(DAYS, mn.split(), mx.split()):
            rows.append(
                {
                    "지점": int(stn),
                    "경도": float(lon),
                    "위도": float(lat),
                    "지점명": nm,
                    "예보구역": fct,
                    "육상구역": land,
                    "며칠뒤": d,
                    "날짜": TMFC.date() + timedelta(days=d),
                    "최저기온": float(a),
                    "최고기온": float(b),
                }
            )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "지점",
                "경도",
                "위도",
                "지점명",
                "예보구역",
                "육상구역",
```

```
            "며칠뒤",
            "날짜",
            "최저기온",
            "최고기온",
        ]
    ),
)
```

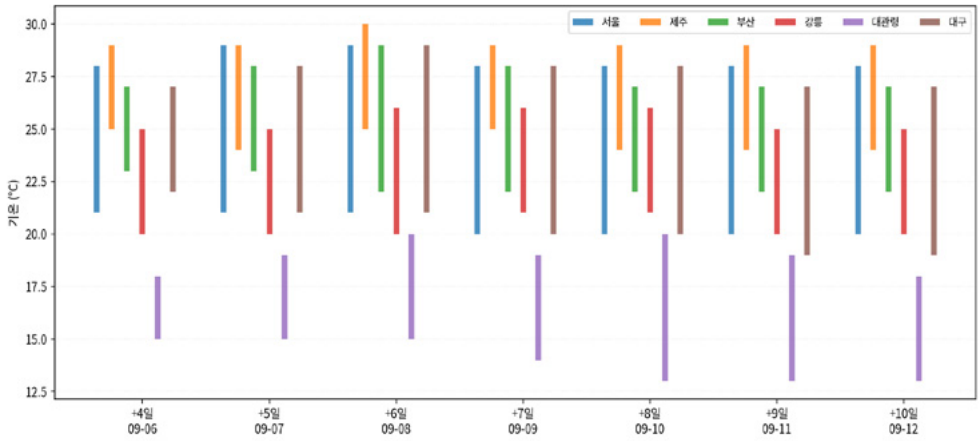


그림 22-1. 날짜별 최저·최고 기온 비교

❸ 날짜별 지역 기온 분포

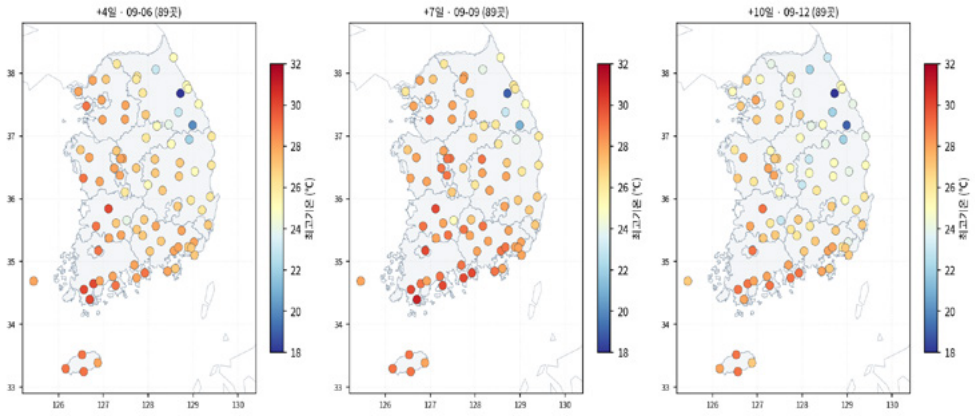


그림 22-2. 날짜별 지역 기온 분포

날씨예보

❹ 구역·날짜별 강수확률 비교

```
def land_frame(sec, *, DAYS, LAND_NAME, TMFC):
    rows = []
    for line in sec["LAND"]:
        rg, rn, wf = line.split("|")
        rn, wf = (rn.split(), wf.split(","))
        k = 0
        for d in DAYS:
            for when in ("오전", "오후"):
                rows.append(
                    {
                        "육상구역": rg,
                        "구역명": LAND_NAME.get(rg, rg),
                        "며칠뒤": d,
                        "때": when,
                        "날짜": TMFC.date() + timedelta(days=d),
                        "강수확률": float(rn[k]),
                        "하늘": wf[k],
                    }
                )
                k += 1
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "육상구역",
                "구역명",
                "며칠뒤",
                "때",
                "날짜",
                "강수확률",
                "하늘",
            ]
        ),
    )
```

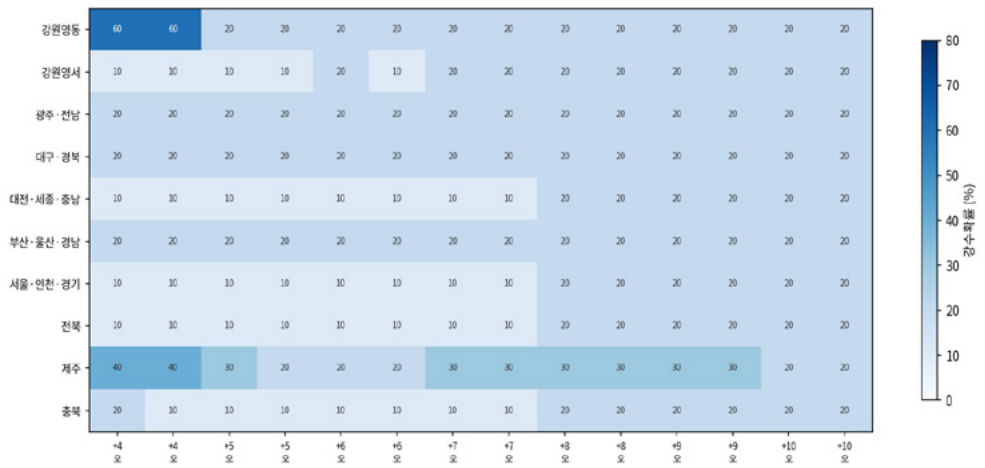


그림 22-3. 구역·날짜별 강수확률 비교

❺ 기온예보 범위 확인

```
def unc_frame(sec, *, DAYS):
  """기온 예보에 함께 오는 아래·위 폭. 며칠 뒤인지에 따라 넓어진다."""
  rows = []
  for line in sec["UNC"]:
    stn, buf = line.split("|")
    for d, tok in zip(DAYS, buf.split()):
      a, b, c, e = [float(x) for x in tok.split("/")]
      rows.append(
        {"지점": int(stn), "며칠뒤": d, "최저폭": a + b, "최고폭": c + e}
      )
  d = pd.DataFrame(
    rows, columns=None if rows else ["지점", "며칠뒤", "최저폭", "최고폭"]
  )
  return d[(d["최저폭"] > 0) | (d["최고폭"] > 0)]
def spread_table(u):
  t = u.groupby("며칠뒤")[["최저폭", "최고폭"]].mean().round(2).reset_index()
  t["지점 수"] = u.groupby("며칠뒤").size().values
  return t
```

며칠뒤	최저폭	최고폭	지점 수
4	1.82	1.64	39
5	2.13	1.9	39
6	2.36	1.9	39
7	2.77	2.05	39
8	1.44	1.31	39
9	1.56	1.59	39
10	1.62	1.62	39

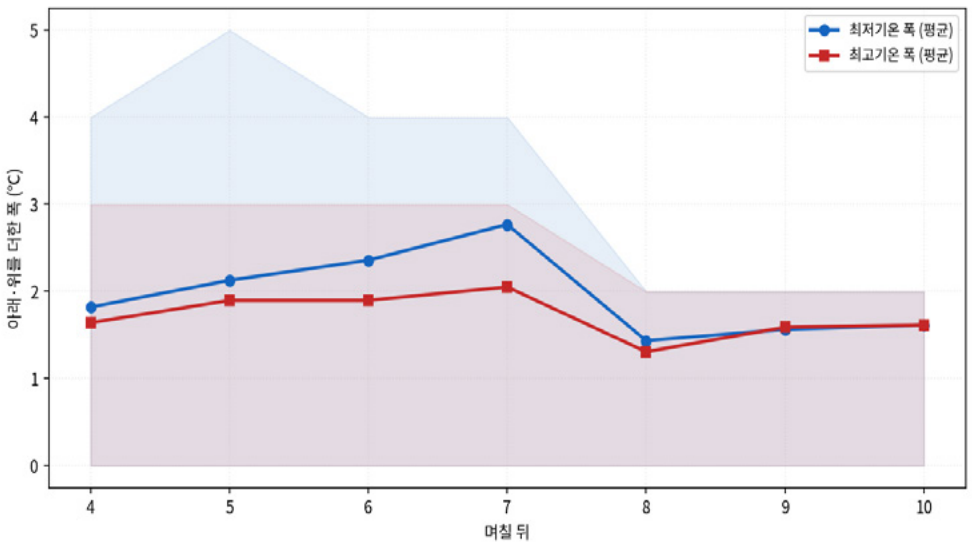


그림 22-4. 기온 예보의 변동 범위 확인

날씨예보

❻ 기온·강수·하늘상태 점수 비교

행사 날짜가 정해지면 지역별 기상 조건을 점수로 비교합니다. 기온이 26℃를 넘거나 16℃에 못 미치는 정도, 강수확률, 하늘상태를 합산합니다.

```
def plan_table(s, l, day, *, SKY_SCORE):
    """날짜·지역별 기온·강수확률·하늘상태를 분석용 점수로 비교합니다. 점수가 낮을수록 이 예의 조건에 적합하며 공식 기상판정은 아닙니다."""

    a = s[s["며칠뒤"] == day].copy()
    b = (
        l[l["며칠뒤"] == day]
        .groupby("육상구역")
        .agg(
            강수확률=("강수확률", "max"),
            하늘=("하늘", lambda x: max(x, key=lambda v: SKY_SCORE.get(v, 1))),
        )
        .reset_index()
    )
    m = a.merge(b, on="육상구역", how="left")
    m["하늘점수"] = m["하늘"].map(lambda v: SKY_SCORE.get(v, 1))
    m["더위벌점"] = (m["최고기온"] - 26).clip(lower=0)
    m["추위벌점"] = (16 - m["최저기온"]).clip(lower=0)
    m["점수"] = (
        m["강수확률"] / 10 + m["하늘점수"] * 2 + m["더위벌점"] + m["추위벌점"]
    ).round(1)
    return m.sort_values("점수")[
        [
            "지점명",
            "날짜",
            "최저기온",
            "최고기온",
            "강수확률",
            "하늘",
            "점수",
            "경도",
            "위도",
        ]
    ]
```

지점명	날짜	최저기온	최고기온	강수확률	하늘	점수
추풍령	2026-09-09	18	26	10	맑음	1
장수	2026-09-09	16	25	10	맑음	1
제천	2026-09-09	18	26	10	맑음	1
파주	2026-09-09	18	28	10	맑음	3
인천	2026-09-09	20	28	10	맑음	3
천안	2026-09-09	18	28	10	맑음	3

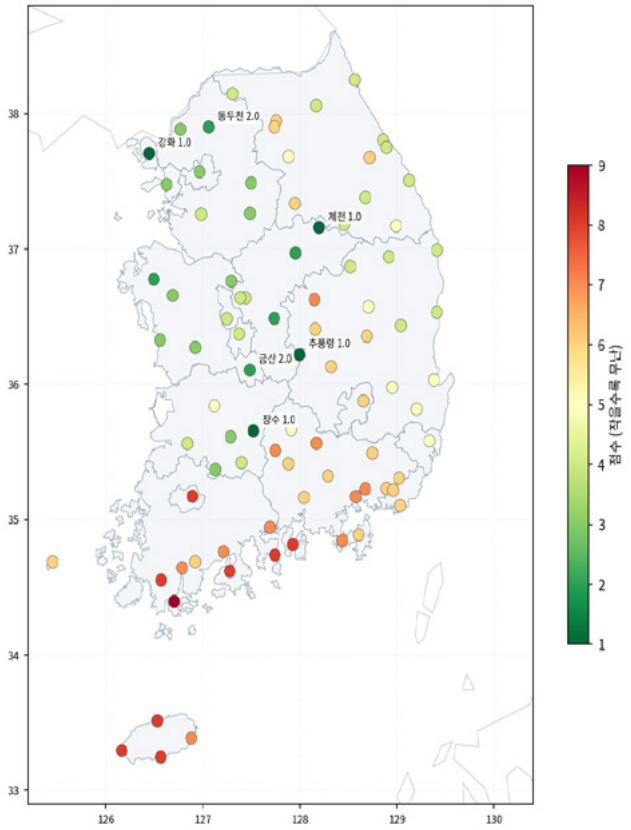


그림 22-5. 기온·강수·하늘상태 점수 비교

날씨예보

23. 영향예보

지역·분야별 위험수준 비교

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 영향예보

❶ 영향예보 조회와 집계

0~4 등급을 기준일·구역·분야·예보시점별로 집계해 위험 분포와 변화를 표시합니다.

```
https://apihub.kma.go.kr/api/typ01/url/ifs_fct_pstt.php?tmef1=20260801&tmef2=20260831&ifpar=hw&ifarea=0&ef_sn=3&authKey=...
#    TM_FC,   TM_EF, STN,   REG_ID, IFPAR, IFAREA, ILVL, EF_SN
202608091130, 20260810, 105,  L1020800,   HW,    1,   3,   1
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
STN_NAME = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
LVL = {1:"관심",2:"주의",3:"경고",4:"위험"}
def parse_ifs(text):
    """한 행이 (기준일 × 특보구역 × 영향분야) 하나의 위험수준이다."""
    rows = []
    for line in text.splitlines():
        s = line.strip()
        if not s or s.startswith("#"):
            continue
        p = [x.strip() for x in s.rstrip("=").rstrip(",").split(",")]
        if len(p) < 8:
            continue
        rows.append(
            {
                "발표시각": p[0],
                "기준일": p[1],
                "관서": int(p[2]),
                "특보구역": p[3],
                "요소": p[4],
                "영향분야": int(p[5]),
                "위험수준": int(p[6]),
                "예보일수": int(p[7]),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "발표시각",
                "기준일",
                "관서",
```

```
        "특보구역",
        "요소",
        "영향분야",
        "위험수준",
        "예보일수",
      ]
    ),
  )
```

ILVL	뜻	IFPAR	뜻
0	영향없음	HW	폭염
1	관심	CW	한파
2	주의	EF_SN 1	하루 전(내일) 예보
3	경고	EF_SN 2	이틀 전(모레) 예보
4	위험		

❷ 한 달간 위험수준 변화

```
def day_table(d):
  """기준일마다 위험수준별 구역 수. 폭염이 오르내리는 모양이 그대로 나반환됩니다."""
  p = (
    d[d["sn"] == 1]
    .pivot_table(index="ef", columns="lvl", values="건수", aggfunc="sum")
    .fillna(0)
    .astype(int)
  )
  p.columns = [LVL[c] for c in p.columns]
  p["합계"] = p.sum(axis=1)
  return p.reset_index().rename(columns={"ef": "기준일"})
```

기준일	관심	주의	경고	위험	합계
20260801	8	115	107	5	235
20260802	7	84	123	21	235
20260803	7	49	157	22	235
20260810	3	82	118	0	203
20260811	36	145	8	0	189
20260812	98	61	0	0	159

날씨예보

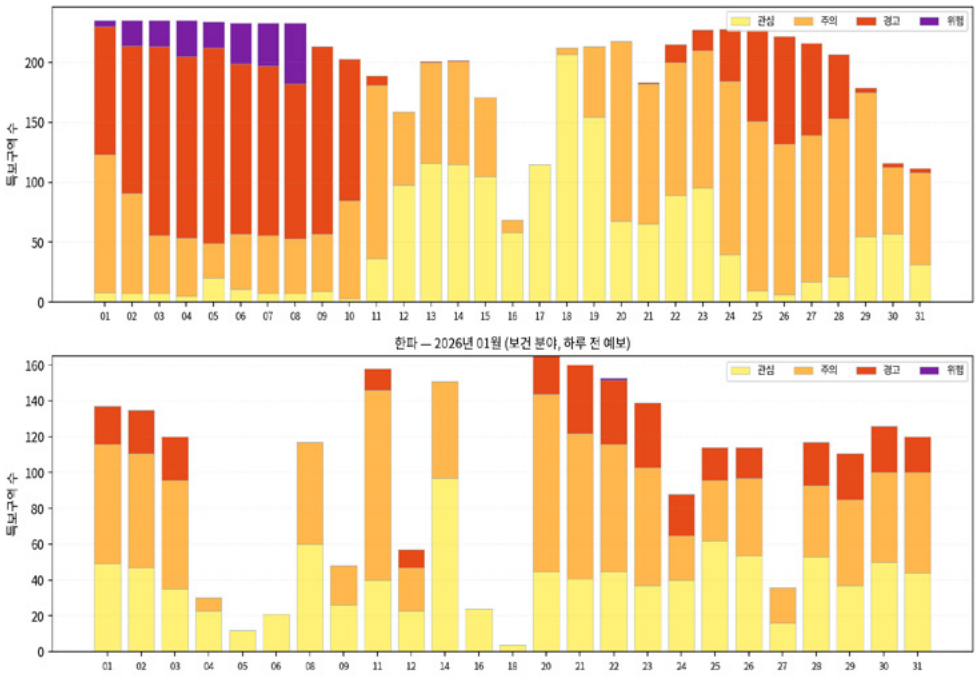


그림 23-1. 한 달간 위험수준 변화

❸ 분야별 위험수준 구성

```
def area_table(d, names):
  p = (
    d.pivot_table(index="area", columns="lvl", values="건수", aggfunc="sum")
    .fillna(0)
    .astype(int)
  )
  p.columns = [LVL[c] for c in p.columns]
  p.index = [names.get(i, str(i)) for i in p.index]
  p["합계"] = p.sum(axis=1)
  p["경고 이상 비율(%)"] = (
    (p.get("경고", 0) + p.get("위험", 0)) / p["합계"] * 100
  ).round(1)
  return p.reset_index().rename(columns={"index": "영향분야"})
```

영향분야	관심	주의	경고	위험	합계	경고 이상 비율(%)
보건 일반인	1642	2525	1782	221	6170	32.5
보건 취약인	605	3563	2138	469	6775	38.5
산업	1642	2525	1782	221	6170	32.5
축산업	324	3664	3153	5	7146	44.2
농업	1220	2278	2028	1253	6779	48.4
수산양식	404	466	330	917	2117	58.9
기타	819	4969	1374	9	7171	19.3

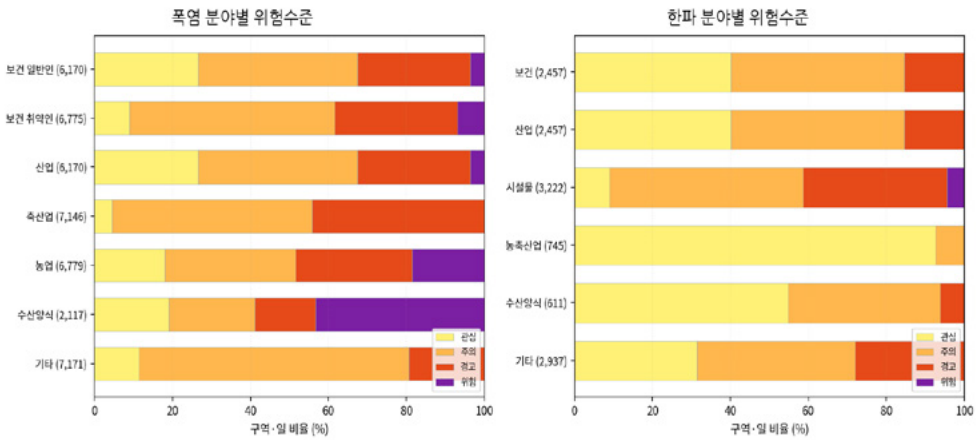


그림 23-2. 분야별 위험수준 구성

❹ 발표관서별 위험수준 분포

```
def stn_table(d):
    p = (
        d.pivot_table(index="stn", columns="lvl", values="건수", aggfunc="sum")
        .fillna(0)
        .astype(int)
    )
    p.columns = [LVL[c] for c in p.columns]
    p["합계"] = p.sum(axis=1)
    p["경고 이상"] = p.get("경고", 0) + p.get("위험", 0)
    p["경고 이상 비율(%)"] = (p["경고 이상"] / p["합계"] * 100).round(1)
    p.insert(0, "관서", [STN_NAME.get(i, str(i)) for i in p.index])
    return p.reset_index()
```

관서	관심	주의	경고	위험	합계	경고 이상	경고 이상 비율(%)
강릉	187	199	110	7	503	117	23.3
수도권	492	420	323	103	1338	426	31.8
청주	105	129	80	0	314	80	25.5
대전	167	225	166	1	559	167	29.9
대구	197	388	265	24	874	289	33.1
전주	107	207	158	8	480	166	34.6
광주	162	467	342	29	1000	371	37.1
부산	145	360	283	49	837	332	39.7
제주	80	130	55	0	265	55	20.8

날씨예보

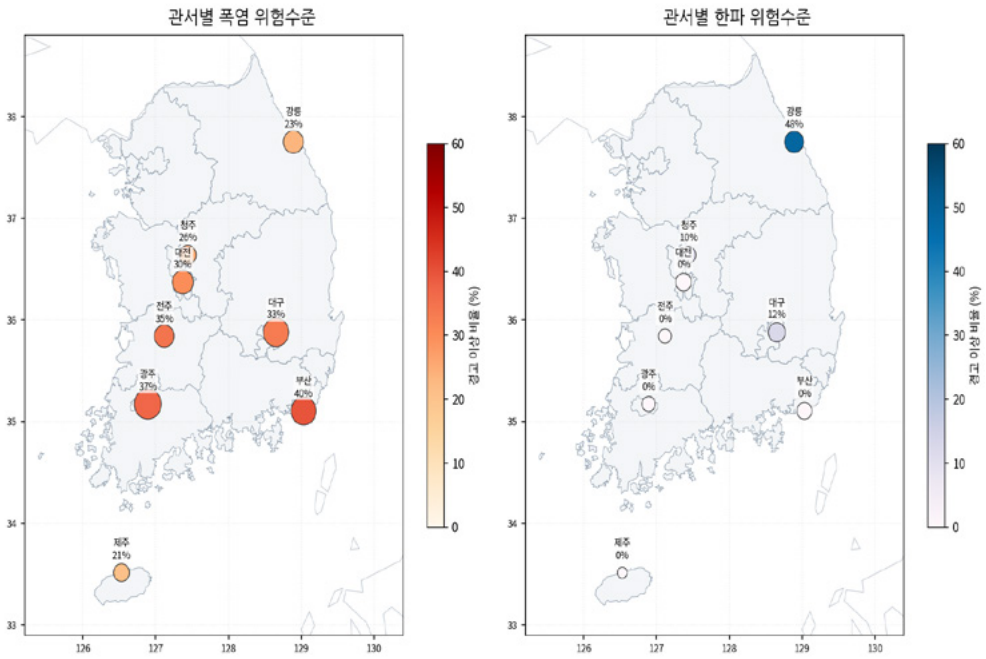


그림 23-3. 발표관서별 위험수준 분포

❺ 하루 전·이틀 전 예보 비교

```
def lead_table(t):
    """같은 기준일을 하루 전에 본 것과 이틀 전에 본 것을 비교합니다."""
    p = (
        t.pivot_table(index="lvl", columns="sn", values="건수", aggfunc="sum")
        .fillna(0)
        .astype(int)
    )
    p.columns = [f"{c}일 전 예보" for c in p.columns]
    p.index = [LVL[i] for i in p.index]
    p["차이(1일 전 − 2일 전)"] = p.iloc[:, 0] - p.iloc[:, 1]
    return p.reset_index().rename(columns={"index": "위험수준"})
```

위험수준	1일 전 예보	2일 전 예보	차이(1일 전 − 2일 전)
영향없음	1008	1327	-319
관심	1642	2518	-876
주의	2525	2225	300
경고	1782	1068	714
위험	221	31	190

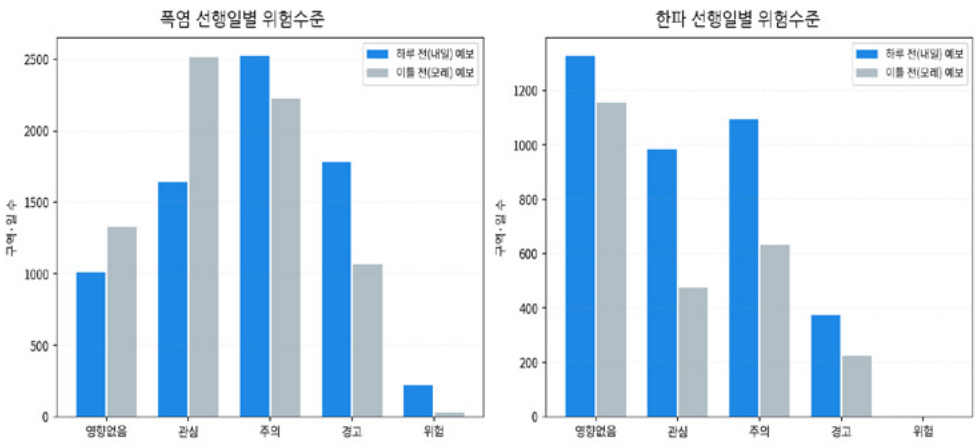


그림 23-4. 하루 전·이틀 전 예보 비교

기상특·정보

24. 기상특보

지역별 발표 현황 확인

데이터 활용 경로(API)

기상청 API허브 > 예특보 > 기상특보 / 예·특보 구역정보

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 특보 자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/wrn_met_data.php'
COLS = ['TM_FC', 'TM_EF', 'TM_IN', 'STN', 'REG_ID', 'WRN', 'LVL', 'CMD',
        'GRD', 'CNT', 'RPT']
def get_wrn(tmfc1, tmfc2):
    res = requests.get(
        api_url,
        params={
            "tmfc1": tmfc1,
            "tmfc2": tmfc2,
            "disp": 0,
            "help": 1,
            "authKey": API_KEY,
        },
        timeout=60,
    )
    rows = []
    for ln in res.text.splitlines():
        s = ln.strip()
        if not s or s.startswith("#"):
            continue
        c = [x.strip() for x in s.rstrip("=").rstrip(",").split(",")]
        if len(c) >= 11:
            rows.append(c[:11])
    d = pd.DataFrame(rows, columns=COLS)
    d[["LVL", "CMD"]] = d[["LVL", "CMD"]].astype(int)
    for c in ["TM_FC", "TM_EF", "TM_IN"]:
        d[c] = pd.to_datetime(d[c], format="%Y%m%d%H%M")
    return d
```

```
d = pd.concat([get_wrn(f'{a}0000', f'{b}0000') for a, b in
          [('20250701', '20250711'), ('20250711', '20250721'),
           ('20250721', '20250801')]])
print(len(d), d['REG_ID'].nunique(), d['WRN'].unique())
2621 217 ['R' 'H' 'V' 'W' 'O']
```

WRN	특보 종류	명령 수	예비(LVL=1)	주의보	경보
R	호우	1531	384	939	208
H	폭염	793	0	476	317
V	풍랑	180	66	111	3
W	강풍	97	50	46	1
O	해일	20	0	20	0

표 24-1. 특보 종류별 명령 수와 수준 구성

❸ 특보 명령의 발효 구간 변환

구역·특보종류별로 CMD를 시간순 재생해 발효 구간을 구합니다.

```
OPEN, CLOSE = {1, 2, 5, 6}, {3, 4, 7}    # 발표·대치·연장·변경 / 해제 계열
eff = d[d['LVL'] >= 2].sort_values(['REG_ID', 'WRN', 'TM_EF', 'TM_FC'])
segs, cur = [], {}
for _, r in eff.iterrows():
    key = (r['REG_ID'], r['WRN'])
    if key in cur and (r['CMD'] in OPEN or r['CMD'] in CLOSE):
        s = cur.pop(key)
        if r['TM_EF'] > s['시작']:
            segs.append({**s, '끝': r['TM_EF']})
    if r['CMD'] in OPEN:
        cur[key] = {'REG_ID': r['REG_ID'], 'WRN': r['WRN'],
                    'LVL': r['LVL'], '시작': r['TM_EF']}
for s in cur.values():                  # 창 끝까지 열려 있는 구간
    segs.append({**s, '끝': d['TM_EF'].max()})
sg = pd.DataFrame(segs)
sg['시간'] = (sg['끝'] - sg['시작']).dt.total_seconds() / 3600
print(len(sg), sg['시간'].median().round(1))
1468 19.8
```

CMD	명령	건수	역할
1	발표	1124	구간 열기
3	해제	717	구간 닫기
5	연장	13	구간 열기
6	변경	767	구간 열기

표 24-2. 명령 코드별 건수와 역할

기상특·정보

❹ 특보 발효 구역 수와 지속시간

시간별로 유효한 구간을 집계해 발효 중인 구역 수의 시계열을 작성합니다.

```
hours = pd.date_range(sg['시작'].min().floor('h'),
                      sg['끝'].max().ceil('h'), freq='h')
def count_open(g):
  c = np.zeros(len(hours), dtype=int)
  for _, r in g.iterrows():
    c += ((hours >= r["시작"]) & (hours < r["끝"])).astype(int)
  return c
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(11.0, 3.2))
bot = np.zeros(len(hours))
for k in sg['WRN'].value_counts().index:
  c = count_open(sg[sg['WRN'] == k])
  ax.fill_between(hours, bot, bot + c, label=k, lw=0)
  bot += c
ax.legend(fontsize=8, ncol=4)
plt.show()
```

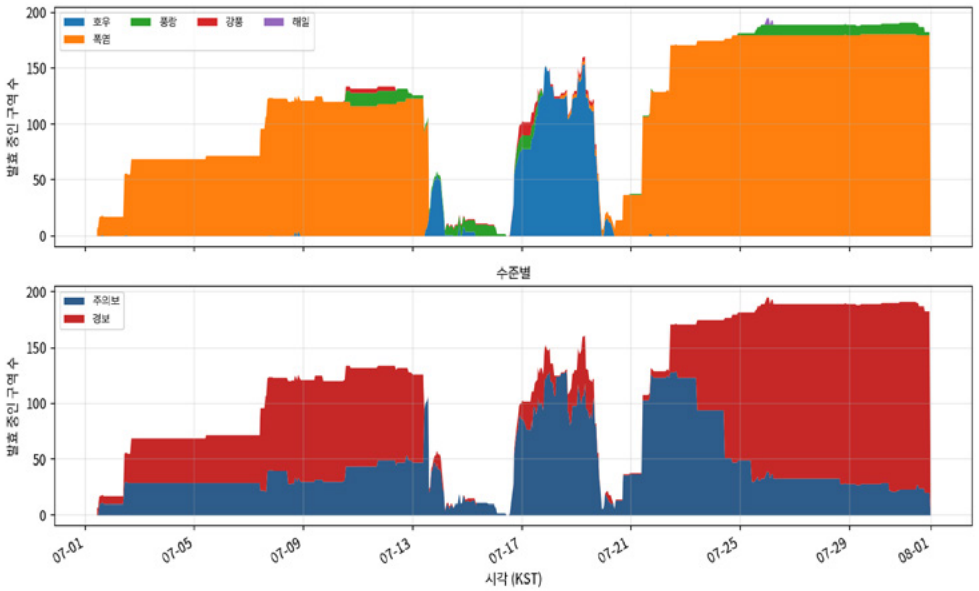


그림 24-1. 2025년 7월 발효 중인 특보구역 수 (종류별·수준별)

특보 종류	구간 수	구역 수	지속 중앙값 (h)	지속 최대 (h)	구역·시간 합(h)
호우	759	176	7.3	75	9259
폭염	608	181	120	325	71094
풍랑	66	33	43	90	2808
강풍	25	18	19.8	46	560
해일	10	10	3	4	29

표 24-3. 특보 종류별 구간 수와 지속시간

❺ 구역별 특보 발효 시간

```
reg_url = 'https://apihub.kma.go.kr/api/typ01/url/wrn_reg.php'
aws_url = 'https://apihub.kma.go.kr/api/typ01/url/wrn_reg_aws2.php'
reg = requests.get(reg_url, params={'help': 1, 'authKey': API_KEY})
aws = requests.get(aws_url, params={'help': 1, 'authKey': API_KEY})
# reg : REG_ID 시작 종료 구역구분 상위구역 이름
# aws : 지점번호, 지점명, 사용구분, 경도, 위도, 고도, 예보구역, 특보구역, 구역명, 관서
land = sg[~sg['REG_ID'].str.startswith('S')]    # L 육상 / S 해상
byreg = land.groupby('REG_ID')['시간'].sum()
a['시간'] = a['REG_ID'].map(byreg).fillna(0)
print(len(reg), len(a))
534 711
```

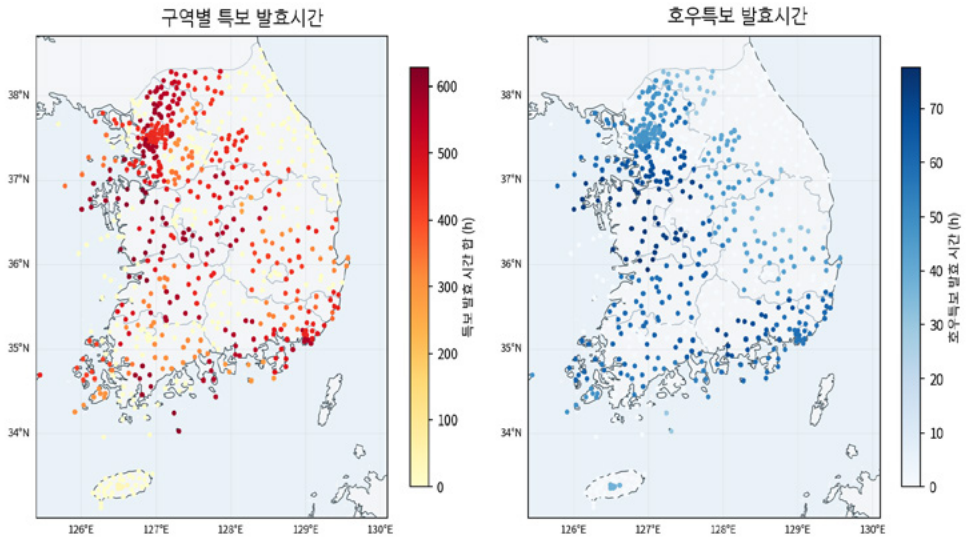


그림 24-2. 구역별 특보 발효 시간 (AWS 지점 위치로 표시)

구분	구역 수	보기
육상	398	전국 · 경기도 · 서울
해상	136	전해상 · 동해전해상 · 동해남부전해상

표 24-4. 특보구역 코드의 두 갈래

특보구역코드	구역명	구간 수	발효 시간 합(h)
L1051600	완도	8	650
L1031200	당진	8	629
L1031600	홍성	8	629
L1031400	보령	10	624
L1031000	예산	7	613
L1060400	김제	7	612
L1030800	부여	7	611
L1031800	세종	7	611
L1052600	거문도.초도	6	609
L1030400	아산	6	609

표 24-5. 7월 발효 시간이 길었던 육상 구역 10곳

태풍예고

25. 태풍정보

이동 경로와 접근 시간 분석

데이터 활용 경로(API)

기상청 API허브 > 태풍 > 태풍정보

❶ 태풍 목록과 경로 조회

중심 위경도·최대풍속·중심기압·등급으로 이동 경로·강도·최근접 시각을 분석합니다.

```
https://apihub.kma.go.kr/api/typ01/url/typ_lst.php?yy=2026&authKey=...
 2026  6  2  2 202605270000 202606031200 장미 JANGMI …
https://apihub.kma.go.kr/api/typ01/url/typ_now.php?yy=2026&typ=6&authKey=...
 FT YY TYP SEQ TMD TYP_TM(UTC)  LAT  LON DIR SP  PS  WS …
  0 2026  6  1  0 202605270000 8.7 137.3  N 18 1002  18 …
#  응답은 EUC-KR 고정폭 텍스트다
#  등급 : TD 열대저압부 · TS 열대폭풍 · STS 강한열대폭풍 · TY 태풍 · L 온대저기압
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
LST_URL = "https://apihub.kma.go.kr/api/typ01/url/typ_lst.php"
NOW_URL = "https://apihub.kma.go.kr/api/typ01/url/typ_now.php"
GRADE = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
SEOUL = (126.9780, 37.5665)  # 서울 기준점 예시
GRADE_ORDER = []  # 미등록 등급은 원래 코드로 집계
def get_list(year, retry=3):
  """그 해에 이름이 붙은 태풍 목록. 번호·이름·발생/소멸 시각이 반환됩니다."""
  params = {"yy": year, "authKey": API_KEY}
  for i in range(retry):
    try:
      r = requests.get(LST_URL, params=params, timeout=60)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      r.encoding = "euc-kr"
      return r.text
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{year} 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(2 * (i + 1))
def get_track(year, seq, retry=3):
  """태풍별 중심 위경도·최대풍속·중심기압·등급을 조회합니다. 실제 시각 간격은 응답의 발표·예측시각으로 확인합니다."""

  params = {"yy": year, "typ": seq, "authKey": API_KEY}
  for i in range(retry):
    try:
      r = requests.get(NOW_URL, params=params, timeout=60)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      r.encoding = "euc-kr"
```

```
        return r.text
      except Exception as exc:
        if i == retry - 1:
          raise
        log(f"{year}-{seq} 재시도 {i + 1}: {type(exc).__name__}")
        time.sleep(2 * (i + 1))
def km(lon1, lat1, lon2, lat2):
  """두 지점 사이 거리(km). 태풍이 얼마나 가까이 왔는지를 이 값으로 잰다."""
  r, rad = (6371.0, math.radians)
  a = (
    math.sin(rad(lat2 - lat1) / 2) ** 2
    + math.cos(rad(lat1))
    * math.cos(rad(lat2))
    * math.sin(rad(lon2 - lon1) / 2) ** 2
  )
  return 2 * r * math.asin(math.sqrt(a))
```

❷ 열 해의 월별 태풍 발생 분포

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
  """응답 묶음의 @SUMMARY 요약과 @NEAR 접근 경로 자료를 읽습니다."""

  txt = Path(path).read_text("utf-8")
  sec, rows, trk, cur = (None, [], [], None)
  for line in txt.splitlines():
    if not line.strip() or line.startswith("#"):
      continue
    if line.startswith("@"):
      sec = line.split()[0][1:]
      continue
    if sec == "SUMMARY":
      p = line.split()
      rows.append(
        {
          "번호": p[0],
          "이름": p[1],
          "연도": int(p[2]),
          "월": int(p[3]),
          "최대풍속(m/s)": float(p[4]),
          "최저기압(hPa)": float(p[5]),
          "서울최근접(km)": float(p[6]),
          "최근접시각": p[7],
          "관측점": int(p[8]),
        }
      )
    elif sec == "NEAR":
      if line.startswith(">"):
        cur = line[1:].split()
        continue
      for blk in line.split():
        tm, lon, lat, ws, ps, gr = blk.split(",")
        trk.append(
          {
            "번호": cur[0],
            "이름": cur[1],
            "연도": int(cur[0][:2]) + 2000,
            "시각": datetime.strptime(
              f"{2000 + int(cur[0][:2])}{tm}", "%Y%m%d%H"
            ),
            "경도": float(lon),
            "위도": float(lat),
            "최대풍속(m/s)": float(ws),
            "중심기압(hPa)": float(ps),
            "등급": gr,
          }
        )
  s = pd.DataFrame(rows)
  t = pd.DataFrame(trk)
  t = t[t["최대풍속(m/s)"] > 0].copy()
  t["서울거리(km)"] = [km(a, b, *SEOUL) for a, b in zip(t["경도"], t["위도"])]
  t["등급명"] = t["등급"].map(GRADE).fillna(t["등급"])
```

태풍예고

```
    return (s, t)
def year_month(s):
    """연도 × 월 발생 수. 태풍철이 언제인지가 이 표 하나로 정해진다."""
    return pd.crosstab(s["연도"], s["월"])
```

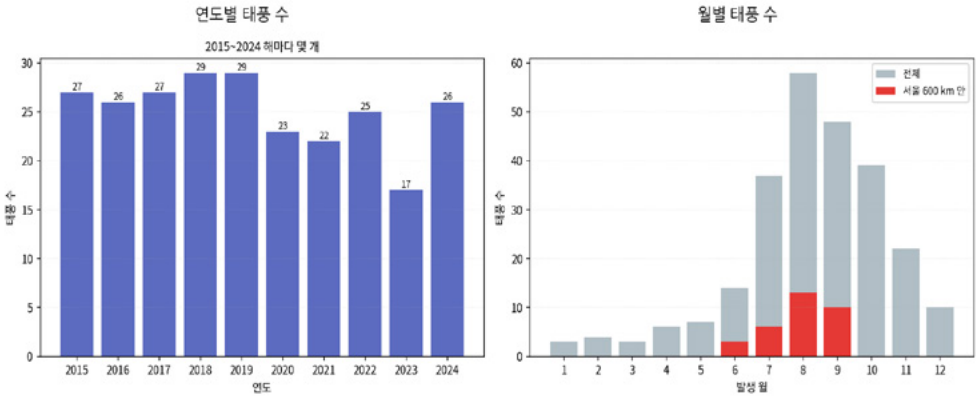


그림 25-1. 열 해의 월별 태풍 발생 분포

연도	1	2	3	4	5	6	7	8	9	10	11	12
2015	1	1	2	1	2	2	4	3	5	4	1	1
2016	0	0	0	0	0	0	4	8	6	5	2	1
2017	0	0	0	1	0	1	9	5	3	4	2	2
2018	1	1	1	0	0	4	5	9	4	1	3	0
2019	1	1	0	0	0	1	4	6	5	4	6	1
2020	0	0	0	0	1	1	1	6	4	7	2	1
2021	0	1	0	1	1	2	3	4	4	4	1	1
2022	0	0	0	2	0	2	2	5	7	5	1	1
2023	0	0	0	1	1	1	3	6	2	2	0	1
2024	0	0	0	0	2	0	2	6	8	3	4	1

❸ 최근접 거리로 접근 태풍 선별

각 경로점과 기준점 사이의 거리를 계산하고 태풍별 최솟값을 구합니다.

```
def near_frame(s, limit=None):
    if limit is None:
        raise ValueError("조회조건·메타데이터를 명시하세요: limit")
    "서울까지 정해진 거리 안쪽까지 들어온 태풍만 남긴다."
    n = s[s["서울최근접(km)"] <= limit].copy()
    n["최근접"] = n["최근접시각"].map(lambda v: f"{v[:2]}-{v[2:4]} {v[4:6]}시")
    return n.sort_values("서울최근접(km)")
```

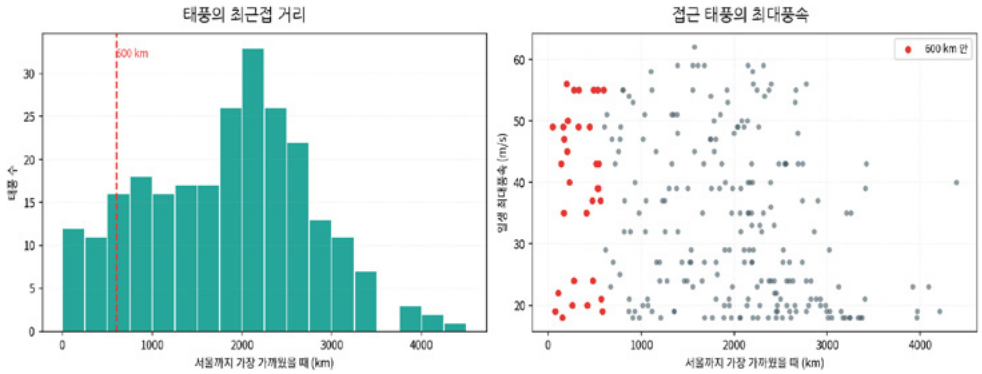


그림 25-2. 최근접 거리로 접근 태풍 선별

번호	이름	연도	월	서울최근접 (km)	최대풍속 (m/s)	최저기압 (hPa)	최근접
2306	KHANUN	2023	7	49	49	930	08-10 15시
2409	JONGDARI	2024	8	77	19	996	08-21 00시
1905	DANAS	2019	7	109	22	985	07-20 21시
2112	OMAIS	2021	8	278	24	990	08-23 18시
1825	KONG-REY	2018	9	279	55	915	10-06 03시
1618	CHABA	2016	9	327	49	935	10-05 03시

태풍예고

❹ 접근 태풍의 경로·풍속 지도

선별한 태풍의 경로를 겹쳐 그리고 각 경로점을 풍속별로 표시합니다. 주요 접근 경로와 강도가 높은 구간을 함께 비교합니다.

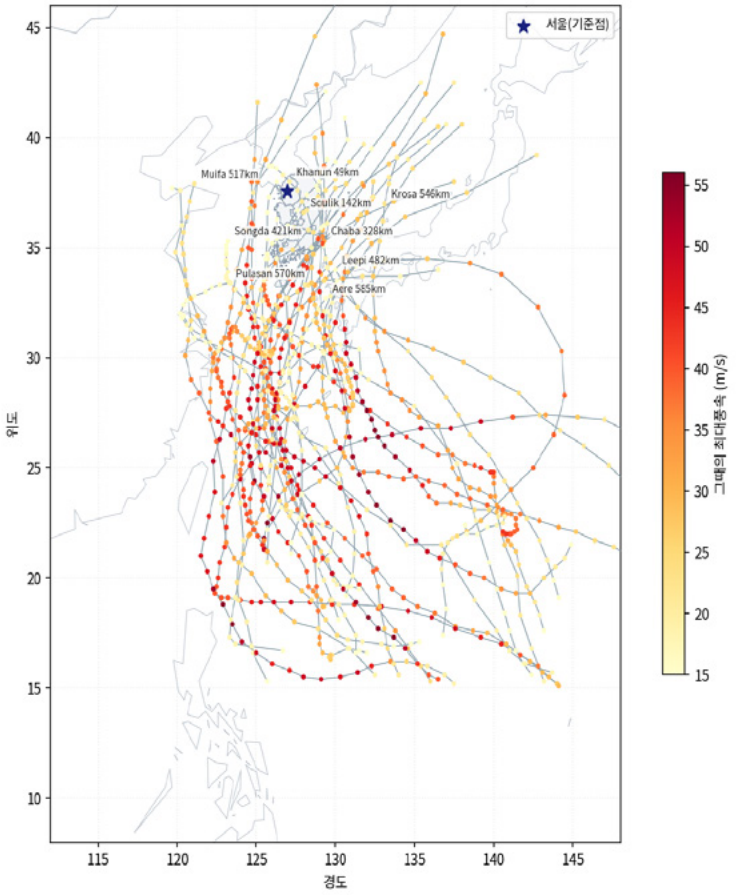


그림 25-3. 접근 태풍의 경로·풍속 지도

❺ 최대풍속과 중심기압의 관계

```
def ws_ps_fit(t):
  """최대풍속과 중심기압의 관계. 한쪽만 있어도 다른 쪽을 짐작할 수 있다."""
  a, b = np.polyfit(t["최대풍속(m/s)"], t["중심기압(hPa)"], 1)
  r = np.corrcoef(t["최대풍속(m/s)"], t["중심기압(hPa)"])[0, 1]
  return (a, b, r)
def grade_table(t):
  """등급마다 풍속·기압이 어느 구간에 놓이는지 집계합니다."""
  g = (
    t.groupby("등급")
    .agg(
      칸수=("최대풍속(m/s)", "size"),
      풍속최소=("최대풍속(m/s)", "min"),
      풍속최대=("최대풍속(m/s)", "max"),
      기압최소=("중심기압(hPa)", "min"),
      기압최대=("중심기압(hPa)", "max"),
    )
    .reindex([x for x in sorted(set(t["등급"]))])
    .reset_index()
  )
  g["등급명"] = g["등급"].map(GRADE).fillna(g["등급"].astype(str))
  return g[
    ["등급", "등급명", "칸수", "풍속최소", "풍속최대", "기압최소", "기압최대"]
  ]
```

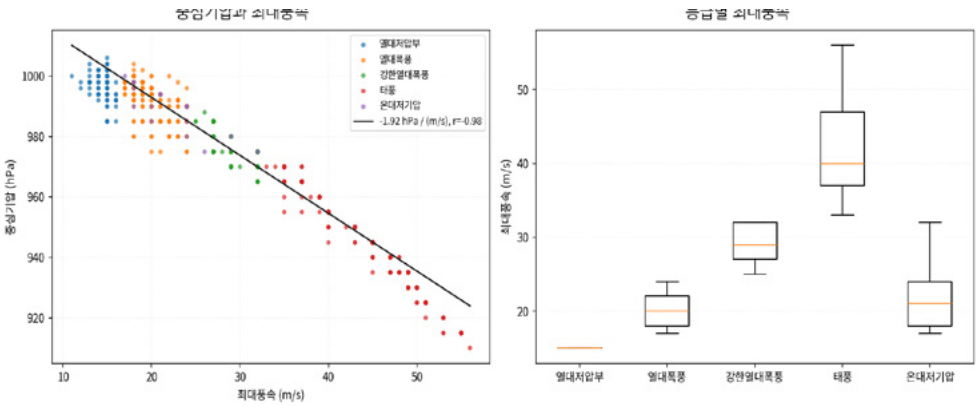


그림 25-4. 최대풍속과 중심기압의 관계

등급	등급명	칸수	풍속최소	풍속최대	기압최소	기압최대
TD	열대저압부	119	11	16	985	1006
TS	열대폭풍	298	17	24	975	1004
STS	강한열대폭풍	218	25	32	965	988
TY	태풍	435	33	56	910	970
L	온대저기압	17	17	32	975	1000

태풍예고

❻ 최근접까지의 시간과 이동 속도

기준점까지의 거리를 시간순으로 정리해 최근접까지 남은 시간을 구합니다. 인접 경로점의 거리와 시간 차이로 이동 속도도 계산합니다.

```
def approach(t, limit=None):
    if limit is None:
        raise ValueError("조회조건·메타데이터를 명시하세요: limit")
    "경로를 시간순으로 훑어 서울에서 정해진 거리 안으로 들어온 첫 시각과\n      가장 가까웠던 시각을 찾는다. 둘의 차이가 곧 준비
할 수 있는 시간이다."
    rows = []
    for (tid, nm), g in t.groupby(["번호", "이름"]):
        g = g.sort_values("시각")
        inside = g[g["서울거리(km)"] <= limit]
        if not len(inside):
            continue
        near = g.loc[g["서울거리(km)"].idxmin()]
        first = inside.iloc[0]
        lead = (near["시각"] - first["시각"]).total_seconds() / 3600
        rows.append(
            {
                "번호": tid,
                "이름": nm,
                f"{limit}km 진입": first["시각"].strftime("%y-%m-%d %H시"),
                "최근접 시각": near["시각"].strftime("%y-%m-%d %H시"),
                "남는 시간(h)": round(lead, 1),
                "최근접(km)": round(near["서울거리(km)"]),
                "그때 등급": near["등급명"],
                "그때 풍속(m/s)": near["최대풍속(m/s)"],
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "번호",
                "이름",
                "최근접 시각",
                "남는 시간(h)",
                "최근접(km)",
                "그때 등급",
                "그때 풍속(m/s)",
            ]
        ),
    ).sort_values("최근접(km)")
def speed_frame(t):
    """이웃한 두 점의 거리와 시간차로 이동 속도를 산출합니다."""
    rows = []
    for (tid, nm), g in t.groupby(["번호", "이름"]):
        g = g.sort_values("시각").reset_index(drop=True)
        for i in range(1, len(g)):
            dt = (g.loc[i, "시각"] - g.loc[i - 1, "시각"]).total_seconds() / 3600
            if dt <= 0:
                continue
            d = km(
                g.loc[i - 1, "경도"],
                g.loc[i - 1, "위도"],
                g.loc[i, "경도"],
                g.loc[i, "위도"],
            )
            rows.append(
                {
                    "번호": tid,
                    "이름": nm,
                    "시각": g.loc[i, "시각"],
                    "위도": g.loc[i, "위도"],
                    "속도(km/h)": d / dt,
                    "서울거리(km)": g.loc[i, "서울거리(km)"],
                }
            )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["번호", "이름", "시각", "위도", "속도(km/h)", "서울거리(km)"]
        ),
    )
```

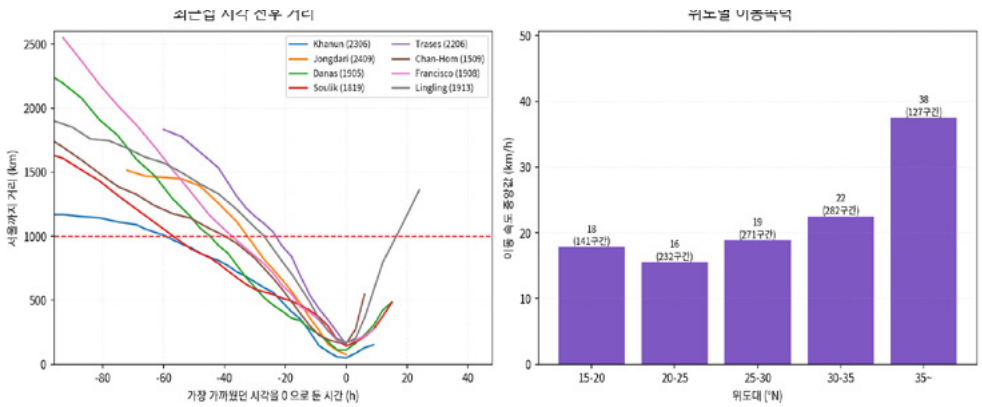


그림 25-5. 최근접까지의 시간과 이동 속도

번호	이름	1000km 진입	최근접 시각	남는 시간(h)	최근접 (km)	그때 등급	그때 풍속(m/s)
2306	KHANUN	23-08-08 06시	23-08-10 15시	57	49	열대폭풍	19
2409	JONGDARI	24-08-19 18시	24-08-21 00시	30	75	열대저압부	13
1905	DANAS	19-07-19 03시	19-07-20 21시	42	111	열대저압부	14
2112	OMAIS	21-08-23 00시	21-08-23 18시	18	279	열대폭풍	18
1825	KONG-REY	18-10-05 06시	18-10-06 03시	21	280	강한열대폭풍	29
1618	CHABA	16-10-04 06시	16-10-05 03시	21	328	강한열대폭풍	32

태풍예고

26. 태풍 베스트트랙

과거 태풍 경로와 강도 비교

데이터 활용 경로(API)

기상청 API허브 > 태풍 > 태풍 베스트트랙

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 베스트트랙 자료 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다. 보유기간 표기와 실제 제공연도가 다를 수 있어 연도별 빈 응답은 제외합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
api_url = 'https://apihub.kma.go.kr/api/typ01/url/typ_besttrack.php'
COLS = ['GRADE', 'TCID', 'YEAR', 'MONTH', 'DAY', 'HOUR', 'LON', 'LAT',
    'MAX_WS', 'PS', 'GALE_LONG', 'GALE_SHORT', 'GALE_DIR',
    'STORM_LONG', 'STORM_SHORT', 'STORM_DIR', 'NAME']
def get_track(year):
  res = requests.get(
    api_url, params={"year": year, "help": 0, "authKey": API_KEY}, timeout=60
  )
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.startswith("#"))
  ]
  return pd.DataFrame([r[:17] for r in rows if len(r) >= 17], columns=COLS)
frames = []
START_YEAR = int(input("조회 시작연도(공식 보유기간 확인): "))
for y in range(START_YEAR, pd.Timestamp.now(tz="Asia/Seoul").year + 1):
  t = get_track(y)
  if not t.empty:
    frames.append(t)
if not frames: raise ValueError("조회조건·메타데이터를 명시하세요: START_YEAR")
d = pd.concat(frames, ignore_index=True)
for c in ['LON', 'LAT', 'MAX_WS', 'PS', 'GALE_LONG', 'GALE_SHORT',
      'STORM_LONG', 'STORM_SHORT']:
  d[c] = pd.to_numeric(d[c], errors='coerce')
d['MAX_WS'] = d['MAX_WS'].replace(-9, np.nan)      # 결측 표기
d[['GALE_LONG', 'GALE_SHORT', 'STORM_LONG', 'STORM_SHORT']] = \
  d[['GALE_LONG', 'GALE_SHORT', 'STORM_LONG', 'STORM_SHORT']].replace(-999, np.nan)
```

```
print(len(d), d['TCID'].nunique())
6877 278
```

GRADE	등급	기록 수	최대풍속 중앙값(m/s)	중심기압 중앙값(hPa)
TD	열대저압부	1268	15	1002
TS	열대폭풍	2408	20	994
STS	강한열대폭풍	1176	29	980
TY	태풍	1933	40	955
L	온대저기압	92	21	990

표 26-1. 등급별 기록 수와 대표값

❸ 태풍 경로 지도

태풍호수별로 좌표를 시간순 연결하고 중심기압을 색으로 표시합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
d['시각'] = pd.to_datetime(dict(year=d['YEAR'].astype(int),
                    month=d['MONTH'].astype(int),
                    day=d['DAY'].astype(int),
                    hour=d['HOUR'].astype(int)))
fig = plt.figure(figsize=(7.6, 6.6))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([105, 165, 0, 52], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#e3e7ea')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
          linewidth=0.6, edgecolor='#37474f')
for tid, g in d.groupby('TCID'):          # 태풍마다 선 하나
  g = g.sort_values('시각')
  ax.plot(g['LON'], g['LAT'], lw=0.7, color='#90a4ae',
        alpha=0.55, transform=PC)
sc = ax.scatter(d['LON'], d['LAT'], c=d['PS'], s=4, cmap='YlOrRd_r',
           vmin=900, vmax=1010, transform=PC, linewidths=0)
fig.colorbar(sc, ax=ax, extend='min', label='중심기압 (hPa)')
plt.show()
```

태풍예고

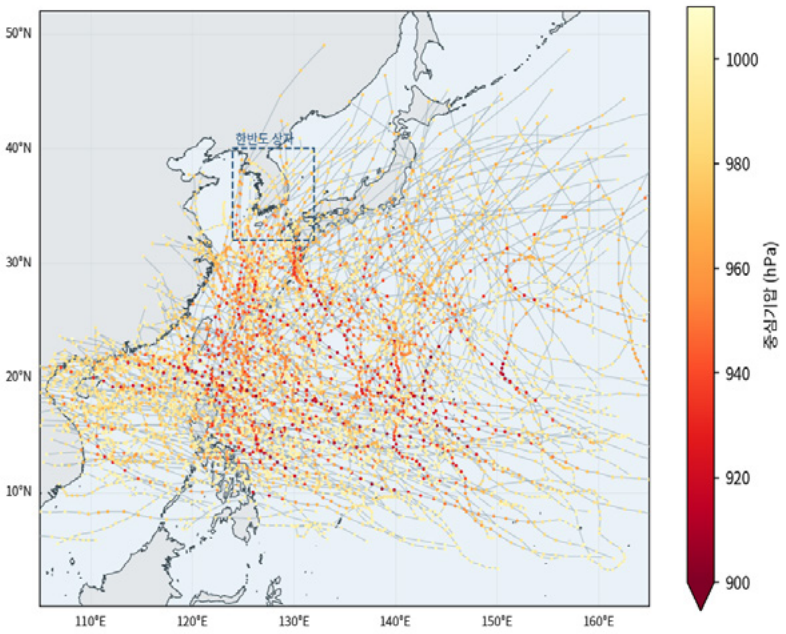


그림 26-1. API에서 조회된 기간의 태풍 경로와 중심기압

TCID	이름	연도	최저기압(hPa)	최대풍속(m/s)	수명(일)
1614	MERANTI	2016	890	62	5.5
2518	RAGASA	2025	895	61	7.8
1513	SOUDELOR	2015	900	59	10.8
1601	NEPARTAK	2016	900	59	6.8
1622	HAIMA	2016	900	59	7.8
1826	YUTU	2018	900	59	11.8
2102	SURIGAE	2021	900	59	11.8
2315	BOLAVEN	2023	900	59	7.8
1504	MAYSAK	2015	905	58	9.2
2302	MAWAR	2023	905	58	14.2

표 26-2. 중심기압이 가장 낮았던 태풍 10개

❹ 연도·월별 태풍 발생 빈도

태풍호수로 연도별 발생 수와 TY 도달 수를 집계하고, 첫 기록의 월로 발생 시기를 비교합니다.

```
tc = (d.groupby('TCID')
    .agg(이름=('NAME', 'first'), 연도=('YEAR', 'first'),
         최저기압=('PS', 'min'), 최대풍속=('MAX_WS', 'max'),
         시작=('시각', 'min'), 끝=('시각', 'max'))
    .reset_index())
tc['태풍여부'] = tc['TCID'].isin(d.loc[d['GRADE'] == 'TY', 'TCID'].unique())
yr = tc.groupby('연도').agg(발생=('TCID', 'nunique'),
                          태풍급=('태풍여부', 'sum')).reset_index()
print(yr.head(3))
```

```
   연도  발생  태풍급
0  2015  27   20
1  2016  26   13
2  2017  27   11
```

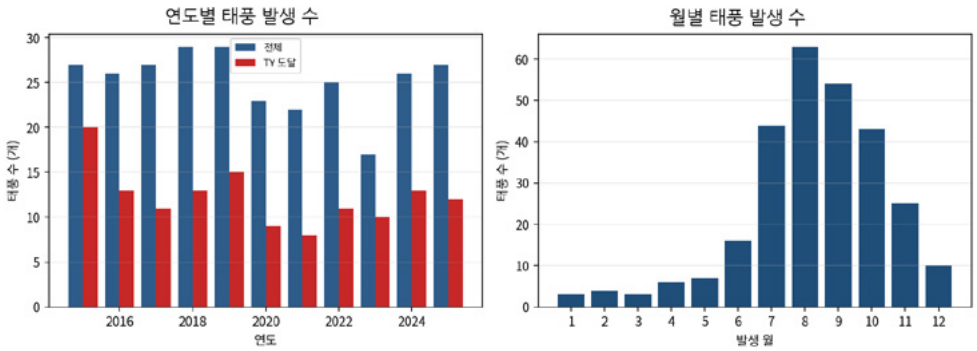


그림 26-2. 연도별 발생 수와 월별 발생 분포

연도	태풍 수	TY 도달	최저기압(hPa)
2015	27	20	900
2016	26	13	890
2017	27	11	925
2018	29	13	900
2019	29	15	910
2020	23	9	910
2021	22	8	900
2022	25	11	915
2023	17	10	900
2024	26	13	915
2025	27	12	895

표 26-3. 연도별 태풍 수와 그 해 최저 중심기압

태풍예고

❺ 중심기압과 최대풍속

같은 기록의 중심기압과 최대풍속으로 상관을 분석합니다.

```
w = d.dropna(subset=['PS', 'MAX_WS'])
print(np.corrcoef(w['PS'], w['MAX_WS'])[0, 1].round(3))
b = (w['PS'] // 10) * 10              # 10 hPa 구간
print(w.groupby(b)['MAX_WS'].median().tail(4))
-0.980
PS
970.0  32.0
980.0  27.0
990.0  20.0
1000.0  15.0
```

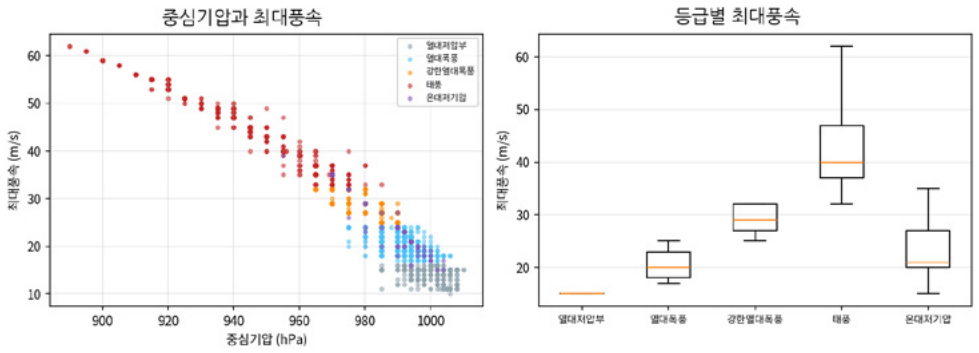


그림 26-3. 중심기압과 최대풍속, 등급별 최대풍속 분포

중심기압 구간(hPa)	최대풍속 중앙값(m/s)	기록 수
900~909	59	29
910~919	55	96
920~929	51	133
930~939	49	189
940~949	45	266
950~959	40	470
960~969	37	462
970~979	32	691
980~989	27	955
990~999	20	2103
1000~1009	15	1448

표 26-4. 중심기압 10 hPa 구간별 최대풍속 중앙값

❻ 강풍·폭풍 반경

```
rr = d.dropna(subset=['GALE_LONG', 'MAX_WS'])
print(np.corrcoef(rr['MAX_WS'], rr['GALE_LONG'])[0, 1].round(3))
ratio = (rr['GALE_SHORT'] / rr['GALE_LONG']).clip(0, 1)
print(ratio.median().round(2))
0.610
0.70
```

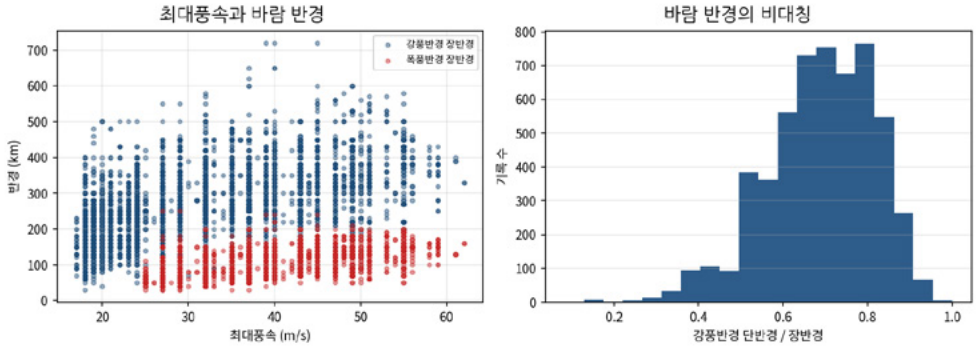


그림 26-4. 최대풍속과 바람반경, 단반경/장반경 비

등급	강풍반경 기록	강풍 장반경 중앙값(km)	폭풍 장반경 중앙값(km)
열대저압부	0	-	-
열대폭풍	2406	200	50
강한열대폭풍	1176	270	70
태풍	1933	320	110
온대저기압	1	150	-

표 26-5. 등급별 바람반경 중앙값

항공예보

27. 공항예보 TAF

공항별 바람과 비행기상 변화 비교

데이터 활용 경로(API)

기상청 API허브 > 항공기상 > 공항예·특보

❶ 공항예보 조회와 시간대 정리

https://apihub.kma.go.kr/api/typ02/openApi/AmmIwxxmService/getTaf?icao=RKSI&numOfRows=1&dataType=XML&authKey=...
<iwxxm:baseForecast> … 유효기간 전체의 기준 상태
<iwxxm:changeForecast changeIndicator="BECOMING"> … 이후로 바뀜
<iwxxm:changeForecast changeIndicator="TEMPORARY_FLUCTUATIONS"> … 그 구간만 잠깐

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
KOR = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
def parse_taf(text):
    root = ET.fromstring(text)
    code = root.findtext("./header/resultCode")
    if code not in (None, "00", "03"):
        raise RuntimeError(f"resultCode {code}")
    out = []
    for taf in root.iter():
        if taf.tag.rsplit("}", 1)[-1] != "TAF":
            continue
        ns = {
            "iwxxm": taf.tag.split("}")[0][1:],
            "aixm": "http://www.aixm.aero/schema/5.1.1",
            "gml": "http://www.opengis.net/gml/3.2",
        }
        icao = taf.findtext(".//aixm:designator", namespaces=ns)
        segments = [("BASE", taf.find("iwxxm:baseForecast", ns))] + [
            ("CHG", e) for e in taf.findall("iwxxm:changeForecast", ns)
        ]
        for kind, element in segments:
            if element is None:
                continue
            forecast = element.find("iwxxm:MeteorologicalAerodromeForecast", ns)
            if forecast is None:
                continue
            wind = forecast.find(".//iwxxm:meanWindSpeed", ns)
            value = float(wind.text) if wind is not None and wind.text else np.nan
            if wind is not None and wind.get("uom", "").lower() in (
                "m/s",
                "m.s-1",
            ):
                value /= 0.5144444444
            out.append(
                {
                    "ICAO": icao,
```

```
            "구분": kind,
            "변화": forecast.get("changeIndicator") or "-",
            "시작": forecast.findtext(
              ".//gml:beginPosition", namespaces=ns
            ),
            "끝": forecast.findtext(".//gml:endPosition", namespaces=ns),
            "시정(m)": forecast.findtext(
              ".//iwxxm:prevailingVisibility", namespaces=ns
            ),
            "풍향(°)": forecast.findtext(
              ".//iwxxm:meanWindDirection", namespaces=ns
            ),
            "풍속(kt)": value,
          }
        )
    return pd.DataFrame(
      out,
      columns=[
        "ICAO",
        "구분",
        "변화",
        "시작",
        "끝",
        "시정(m)",
        "풍향(°)",
        "풍속(kt)",
      ],
    )
def ddmm(v):
    """공항 기준점의 도분(DDMM.mm) 좌표를 십진도로 변환합니다. 공식 좌표 형식을 우선 확인합니다."""

    if pd.isna(v):
        return np.nan
    d = int(v)
    m = round((v - d) * 100, 6)
    return v if m >= 60 else d + m / 60.0
```

ICAO	공항	위도	경도	발표시각(UTC)	유효기간
RKSI	인천	37.4667	126.433	2026-09-02T05:00	09-02T06:00~09-03T12:00
RKSS	김포	37.55	126.8	2026-09-02T05:00	09-02T06:00~09-03T12:00
RKPC	제주	33.5	126.483	2026-09-02T05:00	09-02T06:00~09-03T12:00
RKNW	원주	37.7167	127.96	2026-09-02T05:00	09-02T06:00~09-03T12:00
RKJY	여수	34.8333	127.617	2026-09-02T05:00	09-02T06:00~09-03T12:00
RKJK	군산	35.9	126.61	2026-09-02T00:00	09-02T00:00~09-03T06:00

표 27-1. 공항별 유효기간

항공예보

❷ 기준 상태와 변화 구간의 시간별 정리

기준 상태·변화 구간을 1시간 단위로 정리하고 시정·운고로 공항별 비행 범주를 비교합니다.

```
def cloud_base(s):
    """구름 표기에서 BKN·OVC 층의 최저 운고를 추출합니다."""
    if not isinstance(s, str) or s in ("-", "CAVOK"):
        return np.nan
    v = []
    for tok in s.split("+"):
        for amt in ("BKN", "OVC"):
            if tok.startswith(amt) and tok[3:].isdigit():
                v.append(int(tok[3:]))
    return min(v) if v else np.nan
```

분류는 미국 AWC의 시정·운고 비교 기준(시정 statute mile, 운고 ft)을 참고한 분석 지표입니다. 결측은 '미분류'이며 실제 운항 판단에는 해당 공항·항공기의 운항 기준을 적용합니다. 출처: https://aviationweather.gov/impactboard/help.html

```
CAT = [("VFR", 5 * 1609.344, 3000, "green"), ("MVFR", 3 * 1609.344, 1000, "blue"), ("IFR", 1609.344, 500,
"red"), ("LIFR", 0, 0, "magenta"), ("미분류", np.nan, np.nan, "gray")]
def category(vis, base):
    """시정(m)·운고(ft)의 AWC 비교 범주. 결측은 미분류로 둔다."""
    if pd.isna(vis) or pd.isna(base):
        return "미분류"
    miles = float(vis) / 1609.344
    if miles < 1 or base < 500:
        return "LIFR"
    if miles < 3 or base < 1000:
        return "IFR"
    if miles <= 5 or base <= 3000:
        return "MVFR"
    return "VFR"
def timeline(head, seg, step_h=1):
    """예보 구간을 시간별 상태로 변환합니다. BECOMING은 변화 이후의 상태, TEMPORARY_FLUCTUATIONS는 해당 구
간의 일시적 변화로 처리합니다."""


    out = []
    for _, h in head.iterrows():
        s = seg[seg["ICAO"] == h["ICAO"]].sort_values("시작")
        if not len(s):
            continue
        base = s.iloc[0]
        t0, t1 = (base["시작"], base["끝"])
        cur = {
            "시정(m)": base["시정(m)"],
            "풍향(°)": base["풍향(°)"],
            "풍속(kt)": base["풍속(kt)"],
            "구름": base["구름"],
        }
        ts = pd.date_range(t0, t1, freq=f"{step_h}h")
        perm = s[(s["구분"] == "CHG") & (s["변화"] == "BECOMING")]
        tempo = s[(s["구분"] == "CHG") & (s["변화"] != "BECOMING")]
        for t in ts:
            for _, c in perm.iterrows():
                if t >= c["끝"]:
                    for k in cur:
                        if not (pd.isna(c[k]) if k != "구름" else c[k] == "-"):
                            cur[k] = c[k]
            row = dict(cur)
            tp = tempo[(tempo["시작"] <= t) & (t < tempo["끝"])]
            row["일시적"] = len(tp) > 0
            if len(tp):
                q = tp.iloc[0]
                for k in ("시정(m)", "풍향(°)", "풍속(kt)", "구름"):
                    if not (pd.isna(q[k]) if k != "구름" else q[k] == "-"):
                        row[k] = q[k]
            row.update({"ICAO": h["ICAO"], "공항": h["공항"], "시각": t})
            row["운고(ft)"] = cloud_base(row["구름"])
            row["범주"] = category(row["시정(m)"], row["운고(ft)"])
            out.append(row)
    return pd.DataFrame(out)
```

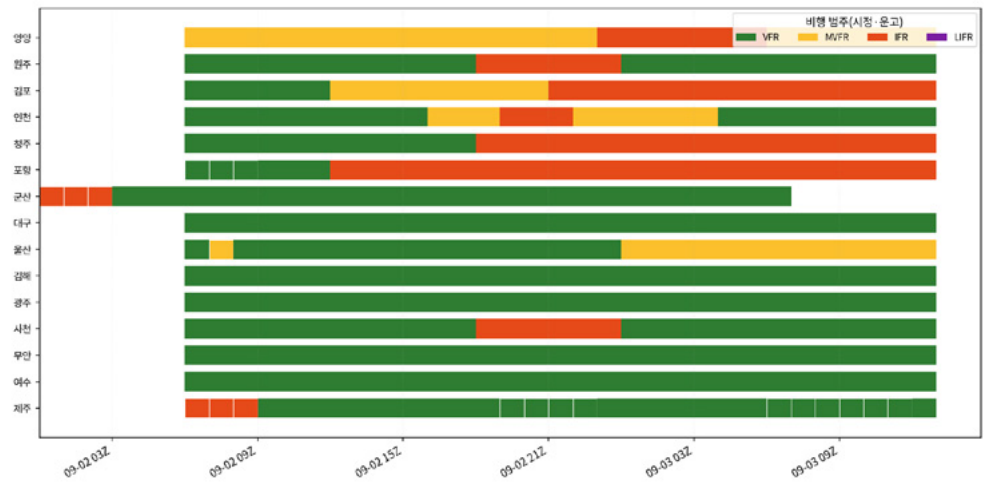


그림 27-1. 기준 상태와 변화 구간의 시간별 정리

❸ 시정 악화 시각 비교

시간별 시정을 공항별로 비교해 악화 시각과 정도를 확인합니다. 가로 점선으로 비행 범주 경계를 표시합니다.

```
def vis_table(tl):
  g = tl.groupby("공항").agg(
    최저시정=("시정(m)", "min"),
    최저운고=("운고(ft)", "min"),
    나쁜시각수=("범주", lambda x: int((x != "VFR").sum())),
    시각수=("범주", "size"),
  )
  g["나쁜 시간 비율(%)"] = (g["나쁜시각수"] / g["시각수"] * 100).round(1)
  return g.sort_values("최저시정").reset_index()
```

공항	최저시정	최저운고	나쁜시각수	시각수	나쁜 시간 비율(%)
김포	3000	2000	25	31	80.6
제주	3500	3500	3	31	9.7
인천	4000		12	31	38.7
무안	9999	3500	0	31	0
김해	9999	3000	0	31	0
여수	9999	3500	0	31	0

표 27-2. 공항별 최저시정·최저운고

항공예보

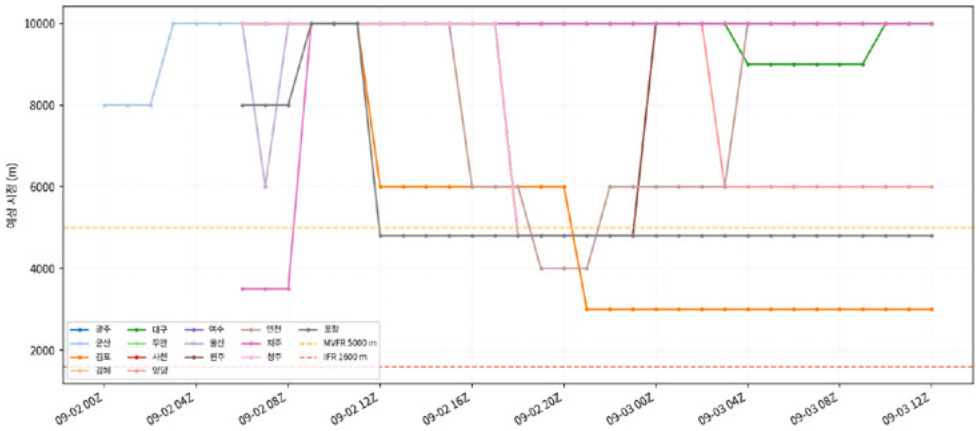


그림 27-2. 시정 악화 시각 비교

❹ 공항별 최저 시정 지도

응답의 공항 기준점 좌표를 이용해 지도에 표시합니다. 예보 기간의 최저 시정에 따라 색을 지정해 공항별 조건을 비교합니다.

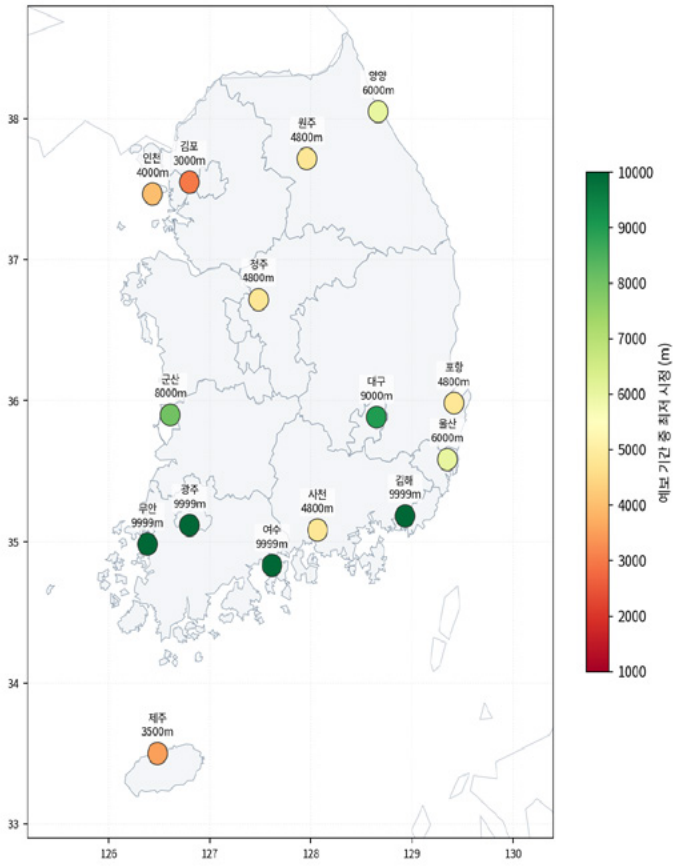


그림 27-3. 공항별 최저 시정 지도

❺ 급격한 풍향 변화 시각

시간별 풍향·풍속에서 한 시간 사이 풍향이 크게 바뀌는 구간을 찾습니다.

```
def wind_shift(tl, deg=90):
    """시간별 풍향 변화량을 계산하고 분석 기준값 이상인 시각을 추출합니다."""

    rows = []
    for nm, g in tl.groupby("공항"):
        g = g.sort_values("시각")
        d = g["풍향(°)"].values
        for i in range(1, len(d)):
            if np.isnan(d[i]) or np.isnan(d[i - 1]):
                continue
            diff = abs((d[i] - d[i - 1] + 180) % 360 - 180)
            if diff >= deg:
                rows.append(
                    {
                        "공항": nm,
                        "시각": g["시각"].iloc[i],
                        "이전 풍향": d[i - 1],
                        "이후 풍향": d[i],
                        "변화(°)": round(diff),
                        "풍속(kt)": g["풍속(kt)"].iloc[i],
                    }
                )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["공항", "시각", "이전 풍향", "이후 풍향", "변화(°)", "풍속(kt)"]
        ),
    )
```

공항	시각	이전 풍향	이후 풍향	변화(°)	풍속(kt)
군산	2026-09-02 3:00	210	330	120	10
김포	2026-09-02 12:00	280	60	140	5
김포	2026-09-02 21:00	60	320	100	5
원주	2026-09-02 15:00	240	30	150	6
원주	2026-09-03 3:00	30	240	150	6
인천	2026-09-02 16:00	290	70	140	5

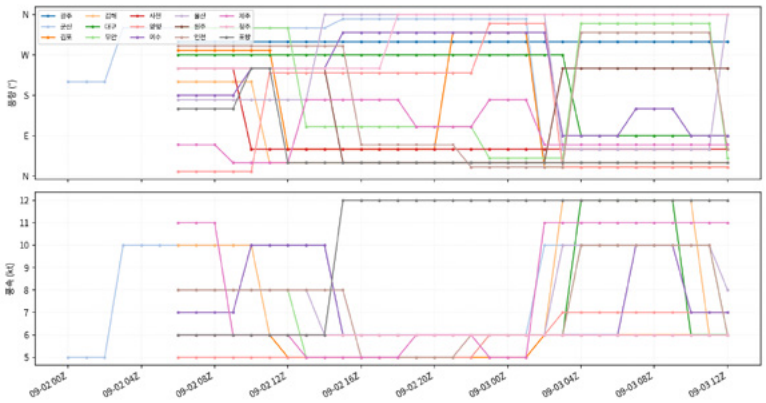


그림 27-4. 급격한 풍향 변화 시각

항공예보

❻ 변화 유형과 구간 길이

```
def load_taf(path):
    """받아 둔 TAF 를 공항 정보와 구간 표 둘로 나눠 읽습니다."""
    head, rows, cur = ([], [], None)
    for line in Path(path).read_text("utf-8").splitlines():
        s = line.strip()
        if not s or s.startswith("#"):
            continue
        if s.startswith(">"):
            p = s[1:].split()
            if len(p) < 2 or p[1] == "NODATA":
                cur = None
                continue
            cur = p[0]
            head.append(
                {
                    "ICAO": cur,
                    "공항": KOR.get(cur, cur),
                    "영문명": p[1].replace("_", " "),
                    "위도": ddmm(float(p[2])),
                    "경도": ddmm(float(p[3])),
                    "발표시각(UTC)": p[4].split("=")[1],
                    "유효기간": p[5].split("=")[1],
                }
            )
            continue
        if cur is None:
            continue
        p = s.split(",")
        if len(p) != 10:
            continue
        num = lambda x: np.nan if x == "-" else float(x)
        rows.append(
            {
                "ICAO": cur,
                "구분": p[0],
                "변화": p[1],
                "시작": pd.Timestamp("2026-" + p[2]),
                "끝": pd.Timestamp("2026-" + p[3]),
                "시정(m)": num(p[4]),
                "풍향(°)": num(p[5]),
                "풍속(kt)": num(p[6]),
                "돌풍(kt)": num(p[7]),
                "일기": p[8],
                "구름": p[9],
            }
        )
    return (pd.DataFrame(head), pd.DataFrame(rows))
```

구분	변화	구간 수	평균 길이(시간)
BASE	-	15	30
CHG	BECOMING	49	1.6
CHG	TEMPORARY_FLUCTUATIONS	6	3.3

데이터 활용사례 50

수치모델

Numerical model



단·중기모델

28. 전지구예보모델(KIM-GDAPS) 예측자료

예측시간별 기온·바람 비교

데이터 활용 경로(API)
기상청 API허브 > 수치모델 > 수치예보모델

❶ KIM 전구모델 격자자료 조회

※ group=KIMG, nwp=NE57은 KIM 전구모델입니다. 2026년 7월 이후 신규 활용은 표준화 NC API를 사용합니다.

```
https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_xy_txt2_std
 ?group=KIMG&nwp=NE57&data=U&name=t2m&level=0
 &sub=1489,1475,1584,1556&tmfc=2026090100&hf=24&map=S&disp=A&authKey=...
# 변수명 = t2m, unit = K, level = 0, i = 96, j = 82,
#  map = S (lon1 = 124.0, lat1 = 32.9, lon2 = 131.9, lat2 = 39.6,
#       x_min = 1489, y_min = 1475, x_max = 1584, y_max = 1556)
# j = 1
 3.00040e+02  3.00430e+02  3.00670e+02 ···
#  sub 는 모델 격자 번호 네 개 · 머리글이 그 창의 위경도를 알려 준다
#  hf 는 예보시간(h) · name 만 바꾸면 같은 창의 다른 변수가 온다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
NC_URL = "https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_xy_txt2_std"
```

HOURS : 입력파일에 실제 있는 선행시간(h) 목록
LAT0 : 입력자료의 위도하한
LAT1 : 위도상한
LON0 : 경도하한
LON1 : 경도상한
TMFC : 입력자료의 실제 생산시각 datetime. KIM은 UTC, 중기예보는 KST
grid_lat : 같은 모양의 격자위도 배열
grid_lon : 모델에서 제공한 실제 2차원 격자경도 배열
필수 입력: HOURS, LAT0, LAT1, LON0, LON1, TMFC, grid_lat, grid_lon. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def save_kim_grids(variables, hours, tmfc, sub, path="kim_grids.npz"):
  import json


  fields, headers = ({}, {})
  for variable in variables:
    for hour in hours:
      text = get_grid(variable, hour, sub=sub, tmfc=tmfc)
      a, head = parse_grid(text)
      key = f"{variable}_{hour}"
      fields[key] = a
      headers[key] = dict(head, tmfc=tmfc, hf=hour, variable=variable)
      time.sleep(0.2)
  np.savez_compressed(path, **fields, metadata=np.array(json.dumps(headers)))
  return (fields, headers)
def to_c(value):
  """원래 단위가 K로 확인된 기온에만 적용한다."""
  return np.asarray(value, float) - 273.15
def get_grid(name, hf, sub=None, tmfc=None, retry=3):
  if tmfc is None:
    raise ValueError("tmfc에 API 조회범위 내 모델 생산시각(UTC)을 지정하세요")
  if sub is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: sub")
  "모델 격자에서 창 하나를 잘라 받는다.\n    sub 는 모델 격자 번호 네 개(x_min,y_min,x_max,y_max)이고,\n    응답 머
리글이 그 창이 실제로 어느 위경도인지 되돌려 준다."
  params = {
    "group": "KIMG",
    "nwp": "NE57",
    "data": "U",
    "name": name,
    "map": "S",
    "sub": sub,
    "tmfc": tmfc,
    "hf": hf,
    "level": 0,
    "disp": "A",
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(NC_URL, params=params, timeout=120)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      r.encoding = "euc-kr"
      return r.text
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{name} +{hf}h 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(2 * (i + 1))


def parse_grid(text):
  """머리글에서 창의 크기와 네 귀퉁이 좌표를 읽고, 나머지를 2차원 배열로 만든다."""
  head = " ".join(
    (l.strip() for l in text.splitlines() if l.lstrip().startswith("#"))
  )
  i = int(re.search("i\\s*=\\s*(\\d+)", head).group(1))
  j = int(re.search("j\\s*=\\s*(\\d+)", head).group(1))
  lon1 = float(re.search("lon1 = ([\\d.]+)", head).group(1))
  lat1 = float(re.search("lat1 = ([\\d.]+)", head).group(1))
  lon2 = float(re.search("lon2 = ([\\d.]+)", head).group(1))
  lat2 = float(re.search("lat2 = ([\\d.]+)", head).group(1))
  vals = []
  for line in text.splitlines():
    if line.startswith("#") or not line.strip():
      continue
    vals += [float(v) for v in line.split()]
  return (
    np.array(vals[: i * j]).reshape(j, i),
    {"i": i, "j": j, "lon1": lon1, "lat1": lat1, "lon2": lon2, "lat2": lat2},
  )
```

단·중기모델

❷ 필요한 격자 영역만 조회

좌표로 위치를 확인하고 격자값을 지도에 표시합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
  """save_kim_grids가 저장한 2차원 배열을 원래 물리단위로 읽습니다."""
  with np.load(path, allow_pickle=False) as saved:
    return {
      name: saved[name].astype(float)
      for name in saved.files
      if name != "metadata"
    }


def axes(grid_lon, grid_lat):
  """모델에서 제공하는 실제 격자 위경도. 모서리 선형보간은 사용하지 않는다."""
  lon, lat = (np.asarray(grid_lon, float), np.asarray(grid_lat, float))
  if lon.ndim != 2 or lon.shape != lat.shape:
    raise ValueError("같은 모양의 2차원 위경도 배열 필요")
  return (lon, lat)
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
NC_URL = "https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_xy_txt2_std"
```

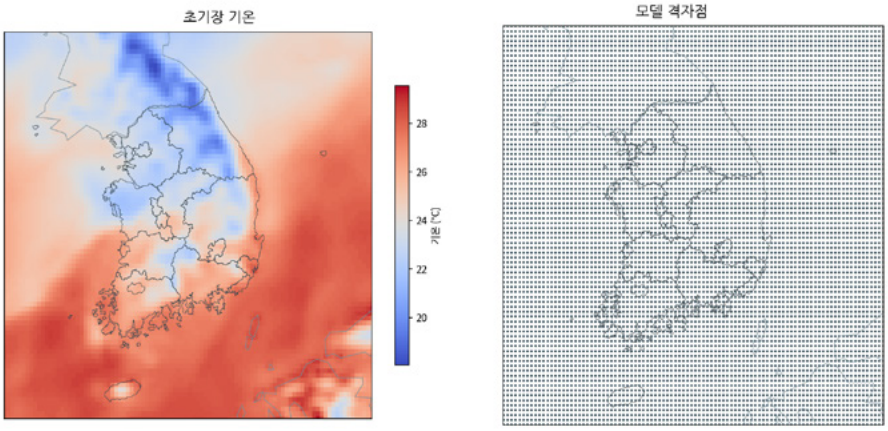


그림 28-1. 필요한 격자 영역만 조회

❸ 예측시간별 기온 분포 비교

```
def stat_frame(fields, *, HOURS, TMFC):
    """예보시간마다 창 전체의 통계를 산출합니다. 하루의 오르내림이 여기 있다."""
    rows = []
    for h in HOURS:
        g = to_c(fields[f"t2m_{h}"])
        rows.append(
            {
                "예보시간(h)": h,
                "시각(UTC)": (TMFC + timedelta(hours=h)).strftime("%m-%d %HZ"),
                "평균(℃)": round(float(g.mean()), 2),
                "최저(℃)": round(float(g.min()), 2),
                "최고(℃)": round(float(g.max()), 2),
                "표준편차": round(float(g.std()), 2),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "예보시간(h)",
                "시각(UTC)",
                "평균(℃)",
                "최저(℃)",
                "최고(℃)",
                "표준편차",
            ]
        ),
    )
```

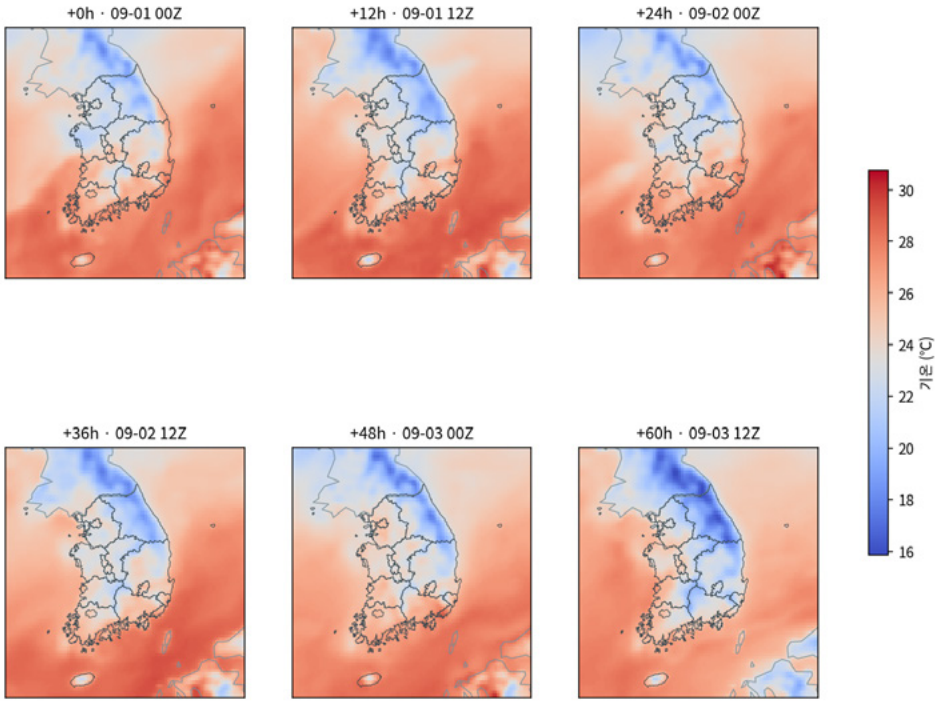


그림 28-2. 예측시간별 기온 분포 비교

단·중기모델

예보시간(h)	시각(UTC)	평균(▯)	최저(▯)	최고(▯)	표준편차
0	09-01 00Z	25.67	18.05	29.55	2.26
12	09-01 12Z	25.77	17.75	29.85	2.47
24	09-02 00Z	25.59	18.15	30.75	2.24
36	09-02 12Z	25.62	17.65	29.45	2.45
48	09-03 00Z	25.33	17.05	30.25	2.28
60	09-03 12Z	24.83	15.85	28.35	2.51
72	09-04 00Z	24.25	15.05	28.35	2.25

❹ 같은 격자의 기상 변수 비교

```
def wind_speed(fields, h=24):
    """동서·남북 성분을 합쳐 풍속으로."""
    u, v = (fields[f"u10m_{h}"], fields[f"v10m_{h}"])
    return np.hypot(u, v)
```

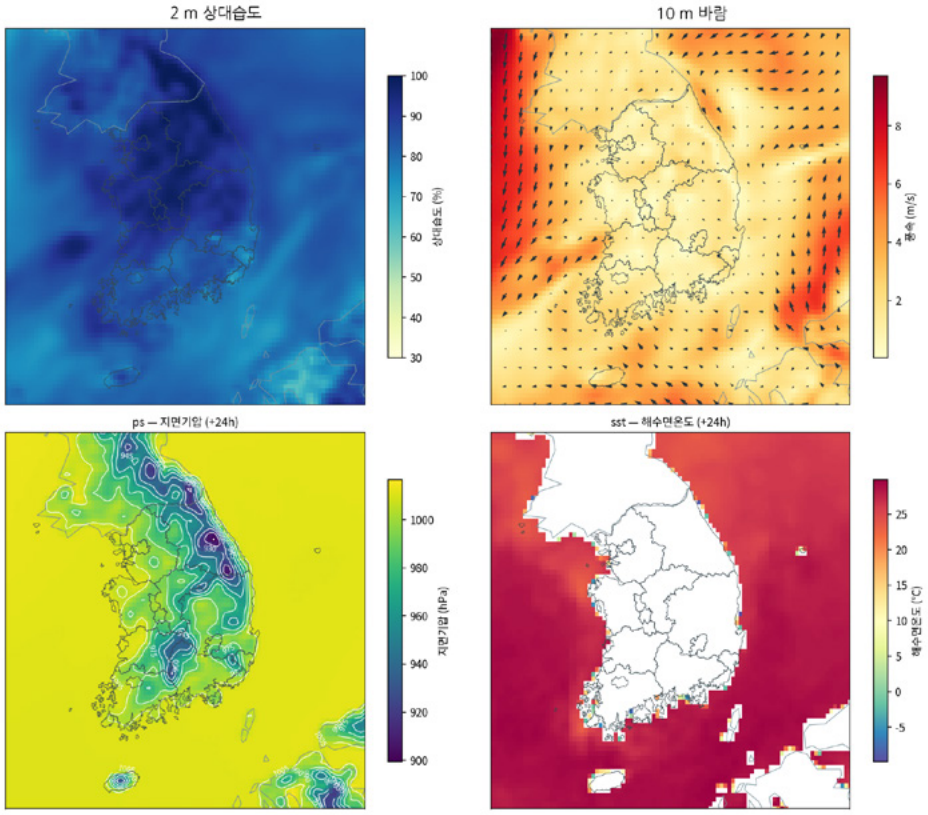


그림 28-3. 같은 격자의 기상 변수 비교

❺ 좌표별 보간과 기온 시계열

```
def at_point(fields, key, lon, lat, grid_lon, grid_lat):
  """실제 격자좌표의 최근접 값. 투영 좌표 없는 모서리 보간 금지."""
  x, y = axes(grid_lon, grid_lat)
  a = np.asarray(fields[key], float)
  if a.shape != x.shape:
    raise ValueError("값과 격자좌표 모양 불일치")
  distance = (((x - lon + 180) % 360 - 180) * np.cos(np.radians(lat))) ** 2 + (
    y - lat
  ) ** 2
  distance[~np.isfinite(x) | ~np.isfinite(y)] = np.nan
  if not np.isfinite(distance).any():
    raise ValueError("유효 격자좌표 없음")
  return float(a.ravel()[np.nanargmin(distance)])

def point_series(
  fields, lon, lat, name="t2m", *, HOURS, TMFC, grid_lat, grid_lon
  ):
  """예보시간마다 같은 점의 값을 뽑아 곡선으로 만든다."""
  rows = []
  for h in HOURS:
    k = f"{name}_{h}"
    if k not in fields:
      continue
    rows.append(
      {
        "예보시간(h)": h,
        "시각(UTC)": TMFC + timedelta(hours=h),
        "값": at_point(
          fields, k, lon, lat, grid_lat=grid_lat, grid_lon=grid_lon
        ),
      }
    )
  return pd.DataFrame(
    rows, columns=None if rows else ["예보시간(h)", "시각(UTC)", "값"]
  )
def city_table(fields, cities, *, HOURS, grid_lat, grid_lon):
  """도시마다 예보시간별 기온을 표로."""
  rows = []
  for nm, (lon, lat) in cities.items():
    r = {"도시": nm}
    for h in HOURS:
      r[f"+{h}h"] = round(
        to_c(at_point(fields, f"t2m_{h}", lon, lat, grid_lon, grid_lat)),
        1,
      )
    rows.append(r)
  return pd.DataFrame(rows)
```

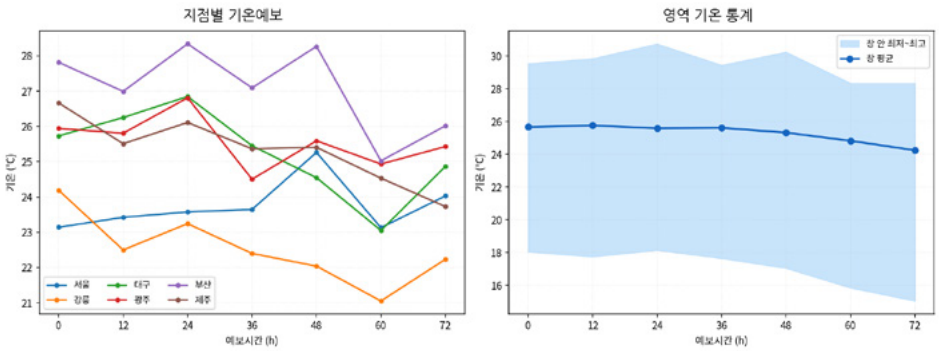


그림 28-4. 좌표별 보간과 기온 시계열

단·중기모델

도시	+0h	+12h	+24h	+36h	+48h	+60h	+72h
서울	23.1	23.4	23.6	23.6	25.3	23.1	24
강릉	24.2	22.5	23.2	22.4	22	21.1	22.2
대구	25.7	26.3	26.8	25.5	24.5	23.1	24.9
광주	25.9	25.8	26.8	24.5	25.6	24.9	25.4
부산	27.8	27	28.3	27.1	28.3	25	26
제주	26.7	25.5	26.1	25.4	25.4	24.5	23.7

❻ 초기장과 지상관측 비교

모델 +0h 초기장과 같은 시각의 지상관측을 지점별로 비교합니다.

```
def load_stn(path):
  """모델 값을 견줄 관측 지점 좌표."""
  out, sec = ({}, None)
  for line in Path(path).read_text("utf-8").splitlines():
    if line.startswith("#"):
      continue
    if line.startswith("@"):
      sec = line[1:]
      continue
    if sec == "SFC":
      p = line.split(",")
      out[47000 + int(p[0])] = (p[4], float(p[1]), float(p[2]), float(p[3]))
  return out

def compare_obs(fields, stn, obs, *, LAT0, LAT1, LON0, LON1, grid_lat, grid_lon):
  """모델 격자값과 같은 시각의 관측을 지점마다 비교합니다."""
  rows = []
  for s, ta in obs.items():
    if s not in stn:
      continue
    nm, lon, lat, ht = stn[s]
    if not (LON0 <= lon <= LON1 and LAT0 <= lat <= LAT1):
      continue
    m = to_c(at_point(fields, "t2m_0", lon, lat, grid_lon, grid_lat))
    rows.append(
      {
        "지점": s,
        "지점명": nm,
        "경도": lon,
        "위도": lat,
        "고도(m)": ht,
        "모델(℃)": round(m, 2),
        "관측(℃)": ta,
        "차이(℃)": round(m - ta, 2),
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "지점",
        "지점명",
        "경도",
        "위도",
        "고도(m)",
        "모델(℃)",
        "관측(℃)",
        "차이(℃)",
      ]
    ),
  )
```

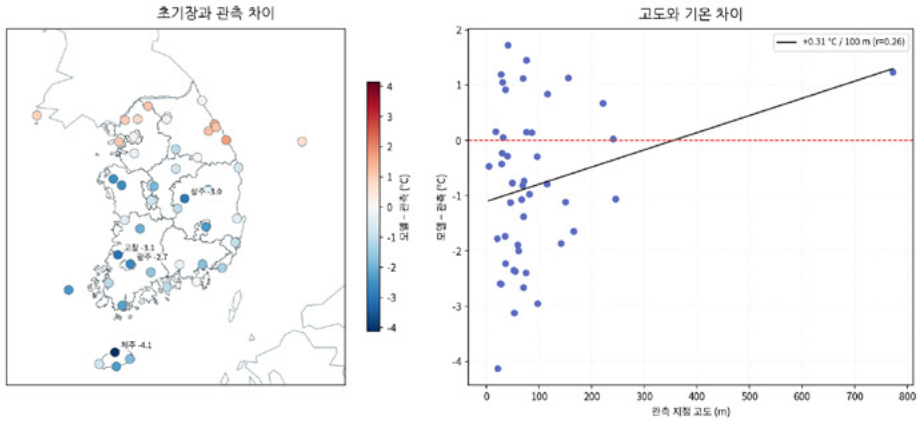


그림 28-5. 초기장과 지상관측 비교

지점명	고도(m)	모델(℃)	관측(℃)	차이(℃)
제주	20.8	26.67	30.8	-4.13
고창	52.4	25.28	28.4	-3.12
상주	96.6	24.25	27.2	-2.95
청주	58.7	23.61	25.5	-1.89
안동	141.3	23.64	25.5	-1.86
성산	20.3	27.62	29.4	-1.78

단·중기모델

29-①. 국지예보모델(KIM-LDAPS) 예측자료

경로별 기온 연직단면 분석

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 수치예보모델

❶ 표준 NC 임의 위경도 조회로 경로별 기압면 자료 구성

```
https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_pt_txt2_std
 ?group=KIML&nwp=L010&data=P&name=T
 &tmfc={UTC 분석시각}&hf=24&lon=126.0&lat=34.0
 &disp=A&help=1&authKey=...  # 경로 표본 위경도를 바꾸어 반복 조회


# 표준 NC 지점응답: TMFC, TMEF, VARN, LEVEL(hPa), VALUE, NAME(단위)
# 경로 표본 개수는 npts로 지정하며, 각 표본 위경도를 반복 조회한다.
# 표준 NC는 공식 변수명 name을 사용한다. GRIB 변수번호 계산식 사용하지 않음.
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
PT_URL = "https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_pt_txt2_std"
```

LINE : 경로명별 (시작경도,시작위도,끝경도,끝위도) dict
TMFC : 입력자료의 실제 생산시각 datetime. KIM은 UTC, 중기예보는 KST
필수 입력: LINE, TMFC. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_xz(name, hf, levels, lon1, lat1, lon2, lat2, tmfc, npts=32):
    """경로 지점별 표준화 NC 등압면 조회. tmfc는 UTC YYYYMMDDHH."""
    if npts < 2:
        raise ValueError("경로 표본은 2개 이상")
    if name not in ["T", "U", "V", "W", "RH", "q"]:
        raise ValueError("공식 변수표의 대소문자·모델별 제공 여부 확인")
    rows = []
    for index, (lon, lat) in enumerate(
        zip(np.linspace(lon1, lon2, npts), np.linspace(lat1, lat2, npts))
    ):
        params = {
            "group": "KIML",
            "nwp": "L010",
            "data": "P",
            "name": name,
            "tmfc": tmfc,
            "hf": hf,
            "lon": float(lon),
            "lat": float(lat),
            "disp": "A",
            "help": 1,
            "authKey": API_KEY,
```

```
    }
    response = requests.get(PT_URL, params=params, timeout=120)
    if not response.ok:
      response.raise_for_status()
    found = 0
    for line in response.text.splitlines():
      if not line.strip() or line.lstrip().startswith("#"):
        continue
      parts = line.split()
      if len(parts) < 6:
        raise ValueError("표준 NC 지점응답 행은 6개 열 이상")
      pressure, value = (float(parts[3]), float(parts[4]))
      if pressure not in levels:
        continue
      rows.append(
        {
          "층(hPa)": pressure,
          "번호": index,
          "변수": name,
          "값": value,
          "경도": float(lon),
          "위도": float(lat),
          "분석시각": parts[0],
          "예측시각": parts[1],
          "단위": parts[5],
        }
      )
      found += 1
    if found == 0:
      raise ValueError("조회조건·메타데이터를 명시하세요: 변수, 시각, levels")
    time.sleep(0.3)
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "층(hPa)",
        "번호",
        "변수",
        "값",
        "경도",
        "위도",
        "분석시각",
        "예측시각",
        "단위",
      ]
    ),
  ).to_csv(index=False)
def parse_xz(text):
  d = pd.read_csv(io.StringIO(text))
  required = {"층(hPa)", "번호", "변수", "값", "경도", "위도"}
  if not required.issubset(d.columns):
    raise ValueError("표준 NC 경로 조회의 CSV 결과를 입력")
  if d.empty:
    raise ValueError("표준 NC 경로 조회의 CSV 결과를 입력")
  return (d, d["번호"].nunique())
```

실행 예: UTC 분석시각을 공식 제공기간 내에서 지정하고 get_xz("T", 24, [1000,925,850, 700,500,300], 126.0,34.0,130.0,38.5, "2026090100") 결과를 CSV로 저장합니다. 조회시각·경로·표본 수에 따라 분석 결과가 달라집니다.

단·중기모델

❷ 두 지점 사이의 기압면 값 조회

기압면 하나를 선택해 거리축에 표시해 경로상의 변화를 비교합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
  """받아 둔 응답 묶음을 읽습니다. 블록 하나가 (변수, 예보시간 또는 선) 하나다."""
  out, cur, buf = ({}, None, [])
  lines = Path(path).read_text("utf-8").splitlines()
  for line in lines + [">END"]:
    if line.startswith(">"):
      if cur:
        out[cur] = parse_xz("\n".join(buf))[0]
      cur, buf = (line[1:], [])
      continue
    if line.startswith("#"):
      continue
    buf.append(line)
  out.pop("END", None)
  return out
def to_grid(d):
  """(층 × 점) 판으로 세운다. 층은 위로 갈수록 기압이 낮아지도록 정렬한다."""
  p = d.pivot_table(index="층(hPa)", columns="번호", values="값")
  return p.sort_index(ascending=False)
def dist_axis(key, n, *, LINE):
  """선 위 점의 번호를 실제 거리(km)로 바꾼다."""
  lon1, lat1, lon2, lat2 = LINE[key]
  dx = (lon2 - lon1) * 111 * math.cos(math.radians((lat1 + lat2) / 2))
  dy = (lat2 - lat1) * 111
  total = math.hypot(dx, dy)
  return (np.linspace(0, total, n), total)
```

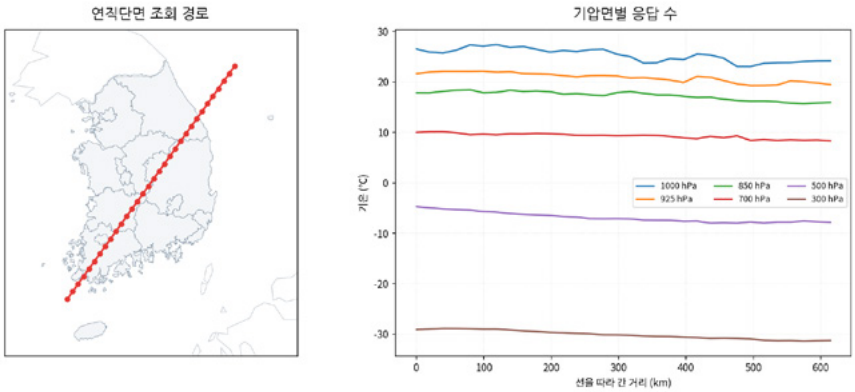


그림 29-1-1. 두 지점 사이의 기압면 값 조회

층(hPa)	평균(℃)	최저(℃)	최고(℃)	선 위 폭(점)
1000	25.31	23.04	27.37	4.33
925	20.88	19.26	22.12	2.86
850	17.27	15.68	18.46	2.78
700	9.27	8.31	10.12	1.81
500	-6.88	-7.99	-4.73	3.26
300	-30.12	-31.38	-28.88	2.5

❸ 기온·비습 연직 단면 구성

```
def q_to_gkg(v):
    """비습은 kg/kg 로 반환됩니다. 1,000 을 곱해야 사람이 읽는 g/kg 이 된다."""
    return v * 1000
```

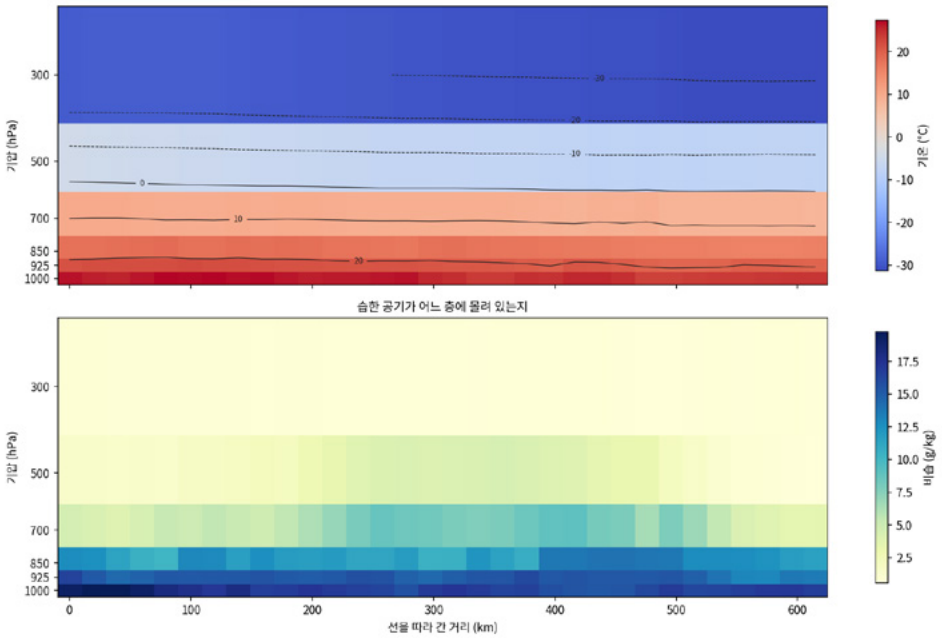


그림 29-1-2. 기온·비습 연직 단면 구성

층(hPa)	평균(g/kg)	최저(g/kg)	최고(g/kg)	선 위 폭(g/kg)
1000	17.03	15.28	19.74	4.46
925	14.97	12.96	16.48	3.52
850	12.18	10.23	14.03	3.8
700	6.35	3.53	8.98	5.45
500	2.59	0.59	4.41	3.82
300	0.78	0.59	0.94	0.35

단·중기모델

❹ 바람 성분의 연직 단면

```
def wind_grid(fields):
    """동서·남북 성분을 합쳐 층별 풍속 판을 만든다."""
    u, v = (to_grid(fields["U_24"]), to_grid(fields["V_24"]))
    return (np.hypot(u, v), u, v)
```

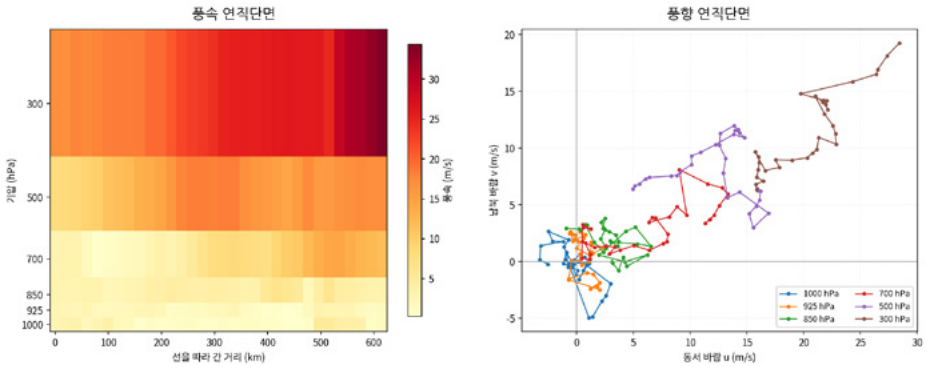


그림 29-1-3. 바람 성분의 연직 단면

층(hPa)	평균(m/s)	최저(m/s)	최고(m/s)	선 위 폭(m/s)
1000	1.87	0.3	5.06	4.76
925	1.87	0.3	3.19	2.89
850	3.85	2.04	6.67	4.63
700	7.03	0.55	14.59	14.04
500	14.84	8.09	18.39	10.3
300	23.58	17.07	34.35	17.28

❺ 예측시간별 연직 단면 비교

```
def hour_change(fields, *, TMFC):
    """예보시간마다 선 위 평균 기온을 층별로 산출합니다."""
    rows = []
    for h in (0, 12, 24, 36):
        k = f"T_{h}"
        if k not in fields:
            continue
        g = to_grid(fields[k]) - 273.15
        r = {
            "예보시간(h)": h,
            "시각(UTC)": (TMFC + timedelta(hours=h)).strftime("%m-%d %HZ"),
        }
        for lv in g.index:
            r[f"{lv:.0f} hPa"] = round(float(g.loc[lv].mean()), 2)
        rows.append(r)
    return pd.DataFrame(rows)
```

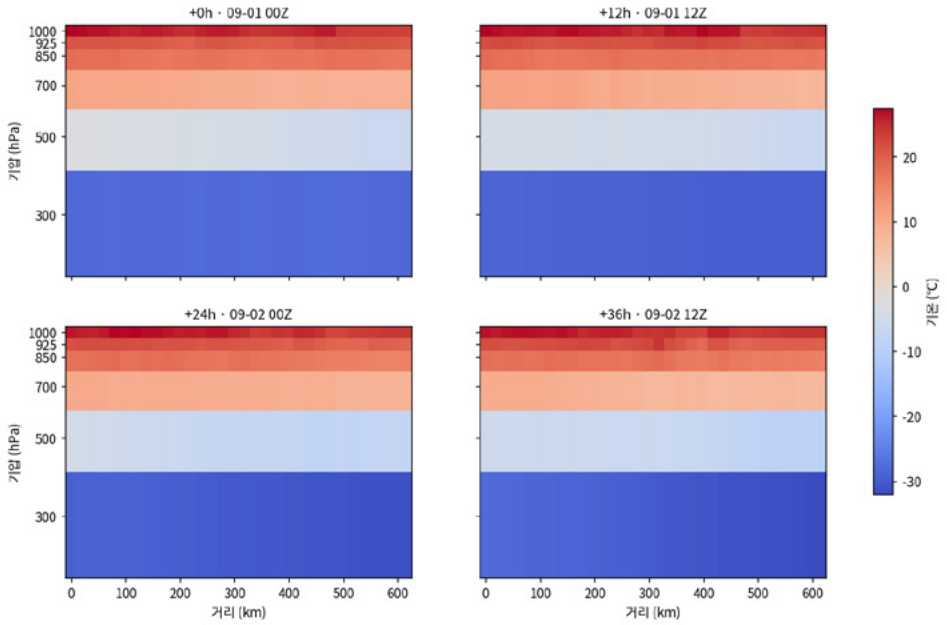


그림 29-1-4. 예측시간별 연직 단면 비교

예보시간 (h)	시각 (UTC)	1000 hPa	925 hPa	850 hPa	700 hPa	500 hPa	300 hPa
0	09-01 00Z	25.34	20.78	17.55	9.53	-4.45	-27.94
12	09-01 12Z	25.83	21.75	17.48	9.66	-5.03	-29.15
24	09-02 00Z	25.31	20.88	17.27	9.27	-6.87	-30.12
36	09-02 12Z	25.21	21.32	17.35	8.26	-6.95	-29.93

❻ 동서·남북 단면 비교

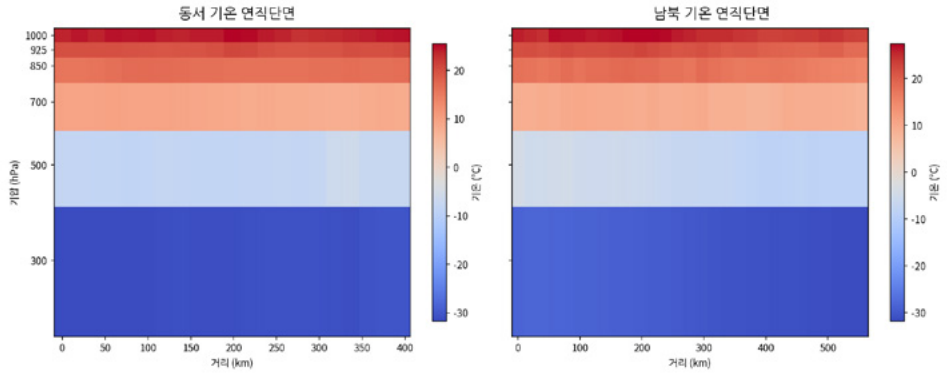


그림 29-1-5. 동서·남북 단면 비교

단·중기모델

29-②. 국지예보모델(KIM-LDAPS) 예측자료

지역별 예측요소 분포 분석

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 수치예보모델

❶ 국지모델 격자자료 조회

```
https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_xy_txt2_std
 ?group=KIML&nwp=L010&data=U&name=RAIN&map=S
 &sub=560,200,960,850&tmfc=2026090100&hf=6&disp=A&authKey=...
# 변수명 = RAIN, unit = mm, level = 0, i = 401, j = 651,
#  map = S (lon1 = 125.7, lat1 = 32.8, lon2 = 130.4, lat2 = 38.7,
#       x_min = 560, y_min = 200, x_max = 960, y_max = 850)
# j = 1
 2.00043e-02  2.00043e-02  4.00085e-02 ...
# j = 2
 2.00043e-02  4.00085e-02  4.00085e-02 ...
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
XY_URL = "https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_xy_txt2_std"
P0 = 101325.0  # Pa. 표준대기 근사이며 실제 지점고도 아님
```

CITIES : 도시 표본 목록. 좌표를 쓰는 함수는 (이름,경도,위도), 격자는 (이름,nx,ny)
HFS : 사용자가 선택한 예측 선행시간(h) 목록
KO : load_regions 반환 객체. EPSG:4326 경계 GeoJSON에서 생성
X0 : 조회창 시작 x 번호
X1 : 조회창 끝 x 번호
Y0 : 조회창 시작 y 번호
Y1 : 조회창 끝 y 번호
tmfc : 선택한 실제 모델 생산시각(UTC, YYYYMMDDHH)
필수 입력: CITIES, HFS, KO, X0, X1, Y0, Y1, tmfc. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def load_regions(path, name_field="name"):
  """EPSG:4326 시도 경계 GeoJSON. name_field는 실제 속성 열 이름."""
  import json
  from shapely.geometry import shape, Point


  payload = json.loads(Path(path).read_text("utf-8"))
  features = payload.get("features", [])
  polygons = []
  for feature in features:
    properties = feature.get("properties", {})
    if name_field not in properties:
      raise ValueError("GeoJSON 시도명 속성 열 확인")
    geometry = shape(feature["geometry"])
    if not geometry.is_valid:
      raise ValueError("유효하지 않은 행정경계")

    polygons.append((properties[name_field], geometry))

  def assign(lon, lat):
    answer = []
    for x, y in zip(lon, lat):
      if not np.isfinite(x) or not np.isfinite(y):
        answer.append(None)
        continue
      point = Point(float(x), float(y))
      matches = [name for name, geom in polygons if geom.covers(point)]

      answer.append(matches[0] if len(matches) == 1 else None)

    return answer

  from types import SimpleNamespace

  return SimpleNamespace(assign_province=assign)
```

시·도별 집계에는 EPSG:4326 좌표계의 행정경계 GeoJSON과 shapely가 필요합니다. load_regions()의 name_field에 시·도명 속성을 지정하고 반환 객체를 KO에 전달합니다. 경계가 중복되거나 행정구역이 확인되지 않는 위치는 미분류로 처리합니다.

```
def get_window(name, hf, sub, tmfc=None):
  if tmfc is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: tmfc")
  "격자 창을 잘라 받는다. sub 은「왼아래 x,y, 오른위 x,y」이고\n      머리글이 그 창의 두 귀퉁이 위경도를 함께 돌려준다."
  r = requests.get(
    XY_URL,
    params={
      "group": "KIML",
      "nwp": "L010",
      "data": "U",
      "name": name,
      "map": "S",
      "sub": sub,
      "tmfc": tmfc,
      "hf": hf,
      "disp": "A",
      "authKey": API_KEY,
    },
    timeout=300,
  )
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  return r.text

def corner_of(text):
  """머리글「lon1 = 125.7, lat1 = 32.8, lon2 = 130.4, lat2 = 38.7」."""
  m = re.search(
    "lon1 = ([\\d.]+), lat1 = ([\\d.]+), lon2 = ([\\d.]+), lat2 = ([\\d.]+)",
    text,
  )
  return tuple((float(v) for v in m.groups())) if m else ()
def parse_window(text):
  """행 번호(# j)를 기준으로 여러 텍스트 행에 걸친 격자값을 읽습니다."""

  rows, cur = ([], None)
```

단·중기모델

```
    for ln in text.splitlines():
        if ln.startswith("# j ="):
            if cur is not None:
                rows.append(cur)
            cur = []
            continue
        if ln.startswith("#") or cur is None:
            continue
        cur += [float(v) for v in ln.split()]
    if cur:
        rows.append(cur)
    return np.array(rows)
def probe_grid(xs, ys, pause=0.1, *, tmfc):
    """여러 위치의 단일 격자를 조회하여 격자번호와 위경도의 대응표를 작성합니다."""

    out = []
    for x in xs:
        for y in ys:
            c = corner_of(get_window("RAIN", 0, f"{x},{y},{x},{y}", tmfc=tmfc))
            if c:
                out.append((x, y, c[0], c[1]))
            time.sleep(pause)
    return pd.DataFrame(out, columns=["X", "Y", "경도", "위도"])
```

❷ 격자 번호와 위경도 관계 추정

```
def fit_xy(pr, order=2):
    """격자번호와 위경도 사이의 2차 다항 회귀식을 계산합니다. X·Y의 교차항을 포함합니다."""

    x = pr["X"].values.astype(float)
    y = pr["Y"].values.astype(float)
    A = design(x, y)
    cl = np.linalg.lstsq(A, pr["경도"].values, rcond=None)[0]
    ca = np.linalg.lstsq(A, pr["위도"].values, rcond=None)[0]
    return (cl, ca)
def design(x, y):
    return np.c_[np.ones_like(x), x, y, x*x, x*y, y*y]
def to_lonlat(cl, ca, X, Y):
    A = design(np.asarray(X, float).ravel(), np.asarray(Y, float).ravel())
    shape = np.shape(X)
    return ((A @ cl).reshape(shape), (A @ ca).reshape(shape))
def fit_table(pr, cl, ca):
    """조회 좌표와 회귀 좌표의 차이를 비교합니다. 출력 좌표의 반올림 정밀도를 고려하여 오차를 해석합니다."""

    lo, la = to_lonlat(cl, ca, pr["X"].values, pr["Y"].values)
    return pd.DataFrame(
        {
            "항목": [
                "표본 수",
                "경도 잔차 RMS",
                "위도 잔차 RMS",
                "경도 잔차 최대",
                "위도 잔차 최대",
                "0.1도 눈금 반올림 표준편차",
            ],
            "값": [
                len(pr),
                round(float((lo - pr["경도"]).std()), 4),
                round(float((la - pr["위도"]).std()), 4),
                round(float(np.abs(lo - pr["경도"]).max()), 4),
                round(float(np.abs(la - pr["위도"]).max()), 4),
                round(0.1 / np.sqrt(12), 4),
            ],
        }
    )
def spacing_table(cl, ca, *, X0, X1, Y0, Y1):
    """격자 한 칸이 실제로 몇 km 인지. 자리마다 달라지는지도 함께 본다."""
    rows = []
    for nm, (x, y) in [
        ("창 왼아래", (X0, Y0)),
        ("창 가운데", ((X0 + X1) // 2, (Y0 + Y1) // 2)),
        ("창 오른위", (X1, Y1)),
    ]:
        lo0, la0 = to_lonlat(cl, ca, [x], [y])
```

```
        lo1, la1 = to_lonlat(cl, ca, [x + 1], [y])
        lo2, la2 = to_lonlat(cl, ca, [x], [y + 1])
        dx = np.hypot(
            (lo1 - lo0) * 111 * np.cos(np.radians(la0)), (la1 - la0) * 111
        )[0]
        dy = np.hypot(
            (lo2 - lo0) * 111 * np.cos(np.radians(la0)), (la2 - la0) * 111
        )[0]
        rows.append(
            {
                "자리": nm,
                "격자X": x,
                "격자Y": y,
                "경도": round(float(lo0[0]), 3),
                "위도": round(float(la0[0]), 3),
                "X 한 칸(km)": round(float(dx), 3),
                "Y 한 칸(km)": round(float(dy), 3),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "자리",
                "격자X",
                "격자Y",
                "경도",
                "위도",
                "X 한 칸(km)",
                "Y 한 칸(km)",
            ]
        ),
    )
def axes_of(cl, ca, nx, ny, step=None, x0=None, y0=None):
    if step is None or x0 is None or y0 is None:
        raise ValueError("조회조건·메타데이터를 명시하세요: step, x0, y0")
    "창 안 격자점의 위경도 판을 만든다. 5칸마다 하나씩 골라 두었으므로\n    격자 번호도 5씩 건너뛴다."
    X = x0 + np.arange(nx) * step
    Y = y0 + np.arange(ny) * step
    XX, YY = np.meshgrid(X, Y)
    return to_lonlat(cl, ca, XX, YY)
```

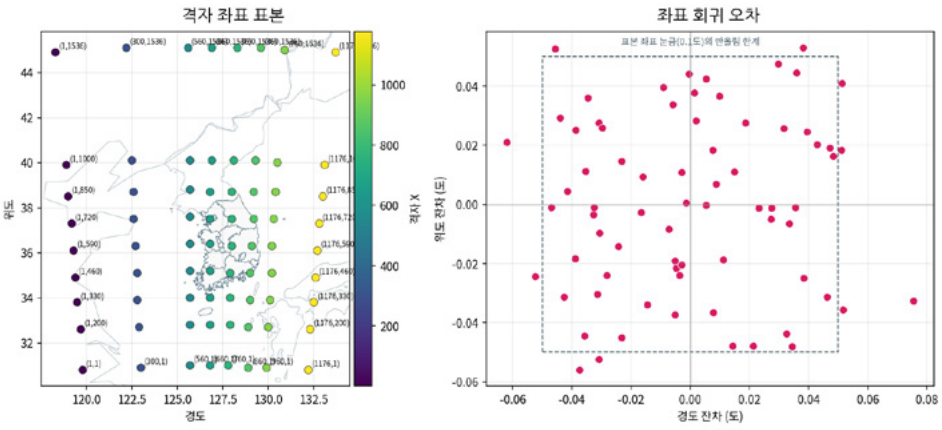


그림 29-2-1. 격자 번호와 위경도 관계 추정

단·중기모델

항목	값
표본 수	72
경도 잔차 RMS	0.032
위도 잔차 RMS	0.0298
경도 잔차 최대	0.0755
위도 잔차 최대	0.056
0.1도 눈금 반올림 표준편차	0.0289

자리	격자X	격자Y	경도	위도	X 한 칸(km)	Y 한 칸(km)
창 왼아래	560	200	125.705	32.8	1.006	1.007
창 가운데	760	525	127.947	35.741	1.021	1.015
창 오른위	960	850	130.414	38.652	1.03	1.024

❸ 추정 좌표로 강수 지도 구성

```
def rain_stats(fields, lon, lat, *, HFS):
  """예측시각마다 비가 얼마나 넓게, 얼마나 세게 오는지."""
  rows = []
  for hf in HFS:
    v = fields["RAIN", hf]
    rows.append(
      {
        "예측시간": f"+{hf}H",
        "최대(mm)": round(float(v.max()), 1),
        "평균(mm)": round(float(v.mean()), 2),
        "1 mm 넘는 칸(%)": round(100 * float((v > 1).mean()), 1),
        "10 mm 넘는 칸(%)": round(100 * float((v > 10).mean()), 1),
        "최대 자리(경도,위도)": f"{lon[np.unravel_index(v.argmax(), v.shape)]:.2f}, {lat[np.unravel_index(v.
argmax(), v.shape)]:.2f}",
      }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "예측시간",
        "최대(mm)",
        "평균(mm)",
        "1 mm 넘는 칸(%)",
        "10 mm 넘는 칸(%)",
        "최대 자리(경도,위도)",
      ]
    ),
  )
```

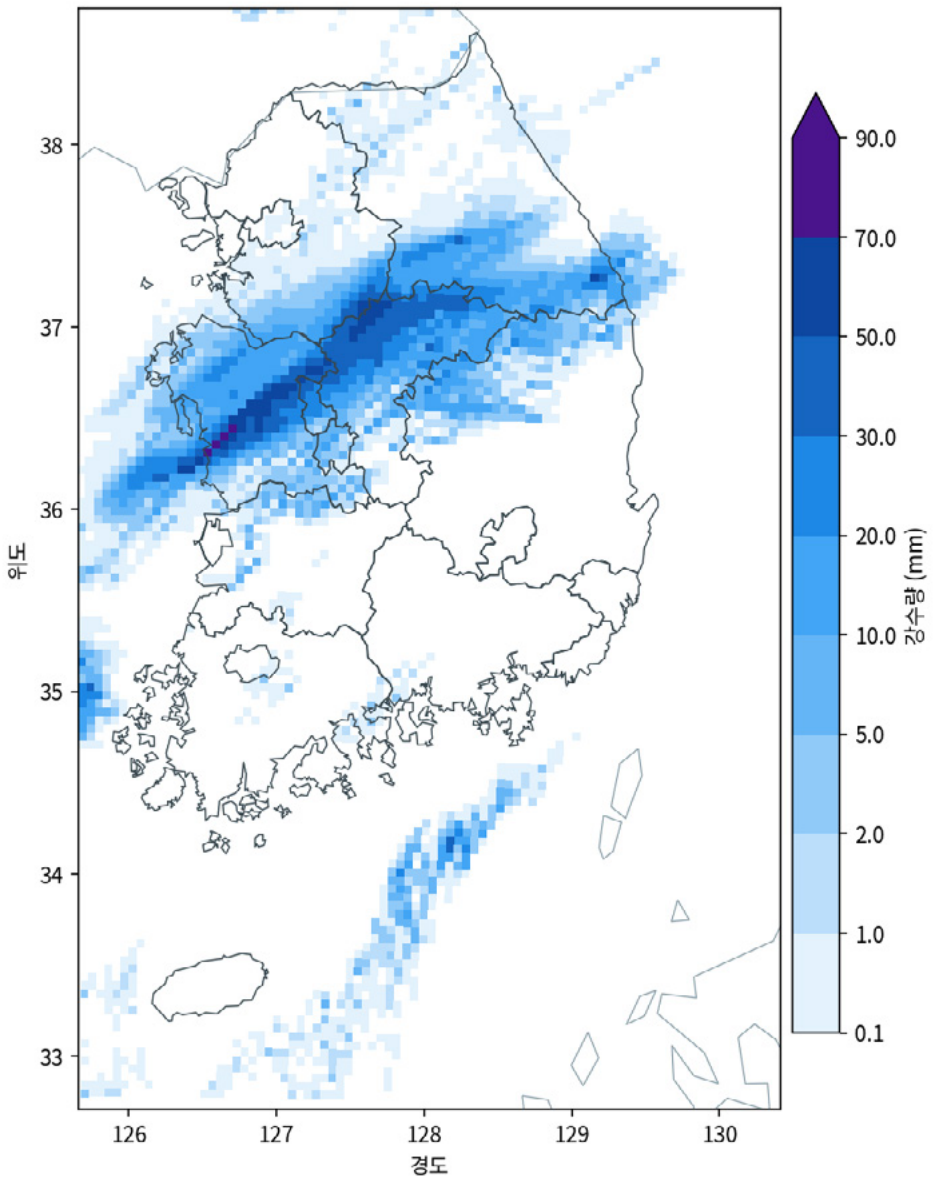


그림 29-2-2. 추정 좌표로 강수 지도 구성

단·중기모델

❹ 예측시간별 강수 구역 변화

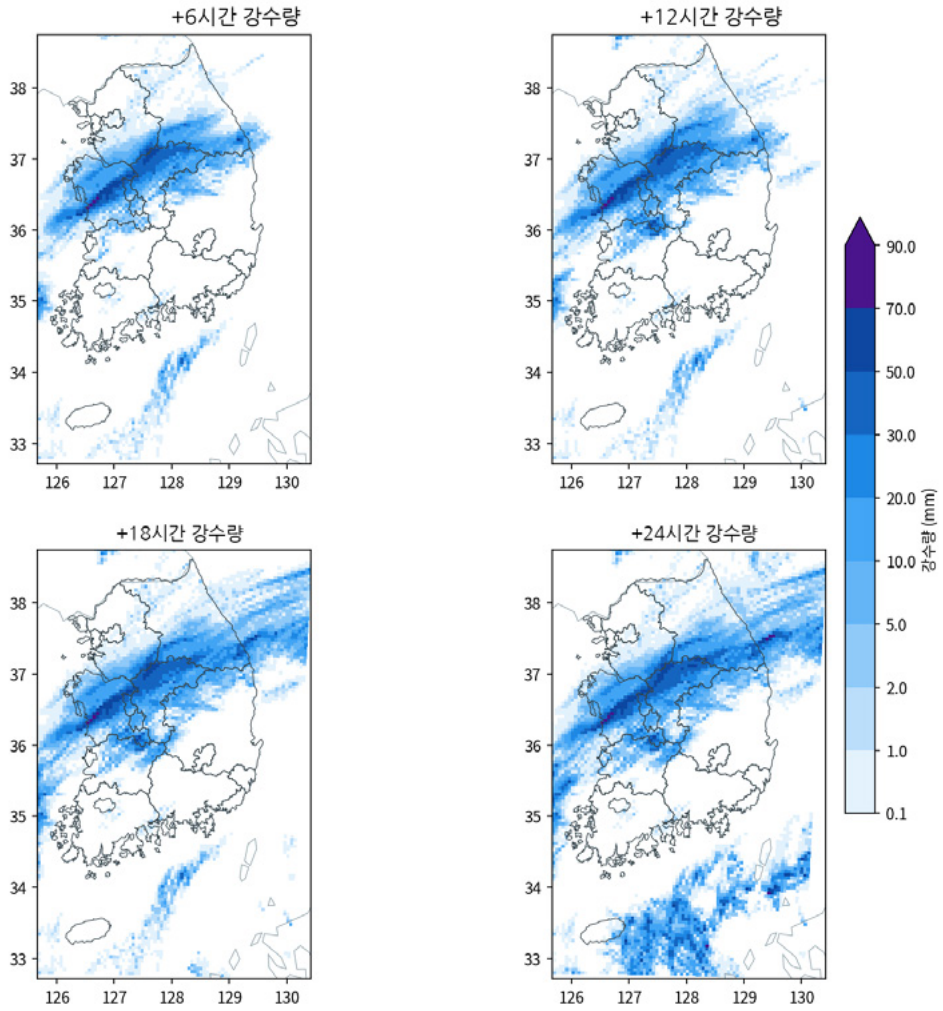


그림 29-2-3. 예측시간별 강수 구역 변화

예측시간	최대(mm)	평균(mm)	1 mm 넘는 칸(%)	10 mm 넘는 칸(%)	최대 자리(경도,위도)
+6H	88.2	1.89	15.3	6.2	126.59, 36.35
+12H	88.4	2.29	18	7.6	126.59, 36.35
+18H	88.4	2.89	23.9	9.9	126.59, 36.35
+24H	91.6	4.27	33	14.7	129.40, 37.49

❺ 지표기압으로 고도 추정

지표기압은 고도의 영향을 받습니다. 표준대기 식으로 고도를 근사하고 강수 분포와 비교합니다.

```
def terrain(psfc):
  """Pa 단위 지상기압의 표준대기 고도 근사. 실제 지형고도가 아니다."""
  p = np.asarray(psfc, float)
  valid = p[np.isfinite(p)]
  if valid.size and ((valid <= 10000).any() or (valid > 120000).any()):
    raise ValueError("기압 단위·결측 확인: Pa 입력 필요")
  return 44330.0 * (1.0 - (p / 101325.0) ** 0.1903)
def terrain_table(h, lon, lat):
  rows = []
  for lo, hi, nm in [
    (-50, 50, "바다·낮은 땅 (50 m 이하)"),
    (50, 200, "50 ~ 200 m"),
    (200, 500, "200 ~ 500 m"),
    (500, 900, "500 ~ 900 m"),
    (900, 9999, "900 m 이상"),
  ]:
    m = (h >= lo) & (h < hi)
    rows.append(
      {
        "높이대": nm,
        "칸 수": int(m.sum()),
        "비율(%)": round(100 * float(m.mean()), 1),
      }
    )
  j, i = np.unravel_index(h.argmax(), h.shape)
  rows.append(
    {
      "높이대": f"가장 높은 칸 {h.max():.0f} m ({lon[j, i]:.2f}, {lat[j, i]:.2f})",
      "칸 수": 1,
      "비율(%)": np.nan,
    }
  )
  return pd.DataFrame(
    rows, columns=None if rows else ["높이대", "칸 수", "비율(%)"]
  )
def high_provinces(h, lon, lat, thr=500, *, KO):
  """높은 칸이 어느 시·도에 몰려 있는지 — 산출기의 자리를 자료로 말하게 한다."""
  if not KO:
    return pd.DataFrame()
  prov = np.asarray(
    KO.assign_province(lon.ravel(), lat.ravel()), dtype=object
  ).reshape(lon.shape)
  m = h > thr
  s = pd.Series([p for p in prov[m] if isinstance(p, str)]).value_counts()
  return s.rename_axis("시·도").reset_index(name=f"{thr} m 넘는 칸")
```

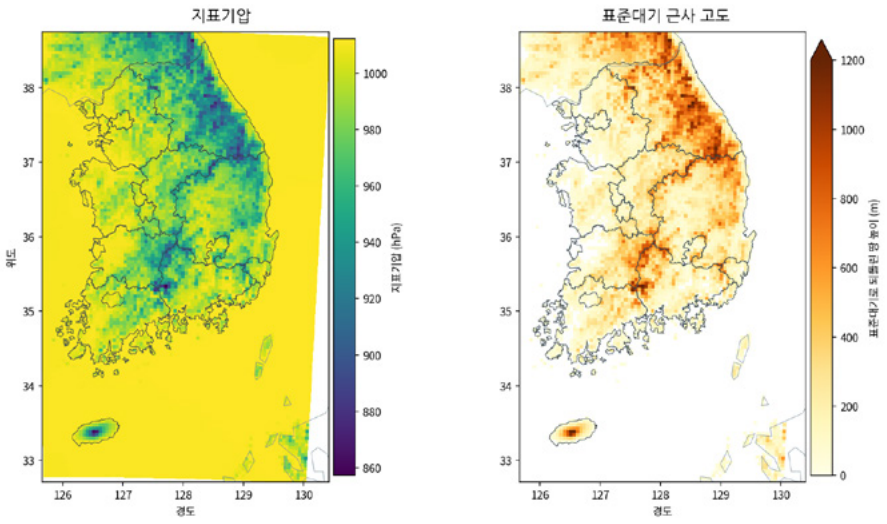


그림 29-2-4. 지표기압으로 고도 추정

단·중기모델

높이대	칸 수	비율(%)
바다·낮은 땅 (50 m 이하)	6618	62.4
50 ~ 200 m	1770	16.7
200 ~ 500 m	1558	14.7
500 ~ 900 m	575	5.4
900 m 이상	90	0.8
가장 높은 칸 1387 m (127.71, 35.33)	1	

시·도	500 m 넘는 칸
강원도	292
경상북도	99
경상남도	58
전라북도	33
충청북도	25
제주특별자치도	12
경기도	11
전라남도	10

❻ 시·도별 집계와 도시별 시계열

```
def province_frame(fields, lon, lat, psfc, *, HFS, KO):
    """추정 좌표와 행정경계를 연결하여 시·도별로 집계합니다. 행정구역이 확인되지 않은 격자는 제외합니다."""

    if not KO:
        return pd.DataFrame()
    prov = np.asarray(
        KO.assign_province(lon.ravel(), lat.ravel()), dtype=object
    ).reshape(lon.shape)
    h = terrain(psfc)
    rows = []
    for p in sorted({v for v in prov.ravel() if isinstance(v, str)}):
        m = prov == p
        rec = {
            "시·도": p,
            "칸 수": int(m.sum()),
            "평균 높이(m)": round(float(h[m].mean())),
        }
        for hf in HFS:
            v = fields["RAIN", hf]
            rec[f"+{hf}H 평균(mm)"] = round(float(v[m].mean()), 2)
        rec["가장 젖은 때"] = max(HFS, key=lambda k: fields["RAIN", k][m].mean())
        rows.append(rec)
    return pd.DataFrame(rows).sort_values("칸 수", ascending=False)


def city_series(fields, lon, lat, *, CITIES, HFS):
    """도시에 가장 가까운 격자 칸을 골라 예측시간별 값을 추출합니다."""
    rows = []
    for nm, lo, la in CITIES:
        d2 = (lon - lo) ** 2 + (lat - la) ** 2
        j, i = np.unravel_index(d2.argmin(), d2.shape)
        rec = {
            "도시": nm,
            "경도": round(float(lon[j, i]), 3),
            "위도": round(float(lat[j, i]), 3),
            "떨어짐(km)": round(
                float(
                    np.hypot(
                        (lon[j, i] - lo) * 111 * np.cos(np.radians(la)),
                        (lat[j, i] - la) * 111,
```

```
            )
          ),
          1,
        ),
      }
      for hf in HFS:
        rec[f"+{hf}H"] = round(float(fields["RAIN", hf][j, i]), 2)
      rows.append(rec)
    return pd.DataFrame(rows)
```

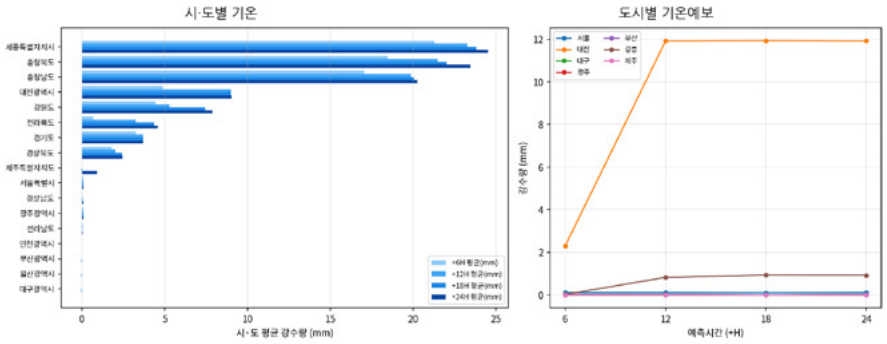


그림 29-2-5. 시·도별 집계와 도시별 시계열

시·도	칸 수	평균 높이(m)	+6H 평균(mm)	+12H 평균(mm)	+18H 평균(mm)	+24H 평균(mm)	가장 젖은 때
경상북도	722	292	1.82	2.03	2.42	2.44	24
강원도	636	497	4.45	5.29	7.45	7.89	24
전라남도	466	140	0.05	0.05	0.04	0.05	24
광주광역시	19	92	0.08	0.08	0.08	0.08	24
대전광역시	19	141	4.89	9.01	9	9.03	24
세종특별자치시	19	116	21.29	23.27	23.84	24.53	24

도시	경도	위도	떨어짐(km)	+6H	+12H	+18H	+24H
서울	126.953	37.59	2.6	0.1	0.1	0.09	0.1
대전	127.393	36.346	1.3	2.3	11.92	11.93	11.92
대구	128.633	35.863	1.4	0	0	-0.01	0
광주	126.864	35.163	1.3	0	0	-0.01	0
부산	129.053	35.164	3	0	0	-0.01	0
강릉	128.887	37.737	1.8	0.02	0.82	0.93	0.92
제주	126.518	33.526	2.1	0	0	-0.01	0

단·중기모델

29-③. 국지예보모델(KIM-LDAPS) 예측자료

격자 연직 프로파일

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 수치예보모델

❶ 지점별 기압면 자료 조회

변수는 쉼표로 연결하고 예측시간을 바꾸어 시간·고도별 분포를 작성합니다.

```
https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_pt_txt2_std
 ?group=KIML&nwp=L010&data=P&name=T,RH,U,V,W
 &tmfc=2026090100&hf=12&lat=37.57&lon=126.97&disp=A&authKey=...
#  X = 672, Y = 722, Lon = 127.0, Lat = 37.6
#    TMFC      TMEF    VARN   LEVEL      VALUS   NAME
2026090100 2026090112     7    1000  2.98030e+02  T(K)
2026090100 2026090112     7     975  2.96460e+02  T(K)
2026090100 2026090112     8    1000  9.44200e+01  RH(%)
2026090100 2026090112    11      50 -6.99997e-03  W(ms-1)
#  머리글이 모델이 고른 격자 번호와 그 격자의 위경도를 알려 준다
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
PT_URL = "https://apihub.kma.go.kr/api/typ06/cgi-bin/url/nph-kim_nc_pt_txt2_std"
G = 9.80665
R_DRY = 287.05  # 건조공기 기체상수 J/(kg K)
```

CITIES : 도시 표본 목록. 좌표를 쓰는 함수는 (이름,경도,위도), 격자는 (이름,nx,ny)
LEVELS : 실제 자료의 등압면(hPa) 순서. 보통 큰 기압→작은 기압
name : 해당 API 명세의 요소 이름
tmfc : 선택한 실제 모델 생산시각(UTC, YYYYMMDDHH)
필수 입력: CITIES, LEVELS, name, tmfc. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_point(lon, lat, hf, name=None, tmfc=None):
  if name is None or tmfc is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: name, tmfc")
  "위경도 하나를 주면 그 자리의 등압면 값이 층마다 한 줄씩 온다.\n      name 에 변수를 쉼표로 이어 주면 한 번에 여러 변수를 받는다."
  r = requests.get(
    PT_URL,
    params={
      "group": "KIML",
      "nwp": "L010",
      "data": "P",
      "name": name,
      "tmfc": tmfc,
      "hf": hf,
      "disp": "A",
      "lat": lat,
      "lon": lon,
      "authKey": API_KEY,
    },
    timeout=120,
  )
  if not r.ok:
    raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
  return r.text

def walk_points(cities=None, hfs=None, pause=0.2, *, name, tmfc):
  if cities is None or hfs is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: cities, hfs")
  "도시 × 예측시간을 돌아 한 묶음으로 모은다.\n      머리글에 모델이 실제로 고른 격자 번호와 그 격자의 위경도가 함께 온다."
  out = []
  for nm, lon, lat in cities:
    for hf in hfs:
      txt = get_point(lon, lat, hf, name=name, tmfc=tmfc)
      grid = grid_of(txt)
      out.append(f">{nm} {lon} {lat} hf={hf} " + " ".join(map(str, grid)))
      out += [
        ln.strip()
        for ln in txt.splitlines()
        if ln.strip() and (not ln.startswith("#"))
      ]
      time.sleep(pause)
  return "\n".join(out)

def grid_of(txt):
  """머리글「X = 672, Y = 722, Lon = 127.0, Lat = 37.6」를 꺼산출합니다."""
  import re


  m = re.search(
    "X\\s*=\\s*(\\d+),\\s*Y\\s*=\\s*(\\d+),\\s*Lon\\s*=\\s*([\\d.]+),\\s*Lat\\s*=\\s*([\\d.]+)",
    txt,
  )
  return (int(m[1]), int(m[2]), float(m[3]), float(m[4])) if m else ()
```

❷ 24개 기압면의 연직 프로파일

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
  """모아 둔 응답을 한 표로 편다. 한 행이「도시 × 예측시간 × 변수 × 기압면」이다."""
  rows, cur = ([], None)
  for ln in Path(path).read_text("utf-8").splitlines():
    ln = ln.rstrip()
    if ln.startswith(">"):
      p = ln[1:].split()
      cur = {
        "도시": p[0],
        "요청경도": float(p[1]),
        "요청위도": float(p[2]),
        "예측시간": int(p[3][3:]),
        "격자X": int(p[4]),
        "격자Y": int(p[5]),
        "격자경도": float(p[6]),
        "격자위도": float(p[7]),
```

단·중기모델

```
            }
            continue
        p = ln.split()
        if len(p) < 6 or cur is None:
            continue
        rows.append(
            dict(
                cur,
                분석시각=p[0],
                예측시각=p[1],
                변수번호=int(p[2]),
                기압면=int(p[3]),
                값=float(p[4]),
                이름=p[5],
            )
        )
    d = pd.DataFrame(rows)
    d["변수"] = d["이름"].str.split("(").str[0]
    d["예측시각"] = pd.to_datetime(d["예측시각"], format="%Y%m%d%H")
    return d
def profile(d, city, hf, *, LEVELS):
    """한 도시·한 시각의 연직 프로파일. 기온은 ‖ 로 바꿔 둔다."""
    g = d[(d["도시"] == city) & (d["예측시간"] == hf)]
    p = g.pivot_table(index="기압면", columns="변수", values="값").reindex(LEVELS)
    p["T(C)"] = p["T"] - 273.15
    p["풍속"] = np.hypot(p["U"], p["V"])
    p["풍향"] = (np.degrees(np.arctan2(-p["U"], -p["V"])) + 360) % 360
    return p
def grid_table(d):
    """조회 지점과 모델 격자 중심의 좌표 및 거리를 비교합니다."""
    t = d.drop_duplicates("도시")[
        ["도시", "요청경도", "요청위도", "격자X", "격자Y", "격자경도", "격자위도"]
    ].copy()
    t["어긋남(km)"] = np.hypot(
        (t["격자경도"] - t["요청경도"]) * 111 * np.cos(np.radians(t["요청위도"])),
        (t["격자위도"] - t["요청위도"]) * 111,
    ).round(2)
    return t
```

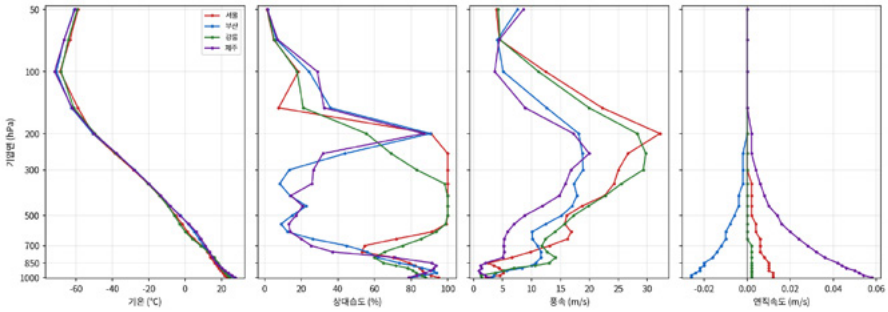


그림 29-3-1. 24개 기압면의 연직 프로파일

도시	요청경도	요청위도	격자X	격자Y	격자경도	격자위도	어긋남(km)
서울	126.97	37.57	672	722	127	37.6	4.25
부산	129.03	35.1	859	458	129	35.1	2.72
강릉	128.89	37.75	836	746	128.9	37.8	5.62
제주	126.53	33.51	637	280	126.5	33.5	2.99

❸ 시간·고도별 기온과 습도 변화

같은 지점을 예측시간별로 조회해 연결해 시간·고도별 분포를 만듭니다.

```
def wide(d, city, var, *, LEVELS):
    """한 도시·한 변수를 「기압면 × 예측시간」 판으로 깐다. 그림 한 장의 재료다."""
    g = d[(d["도시"] == city) & (d["변수"] == var)]
    piv = g.pivot_table(index="기압면", columns="예측시간", values="값")
    return piv.reindex(LEVELS)
def layer_change(d, city, lo=1000, hi=500, *, LEVELS):
    """두 기압면 사이의 변화를 예측시간별로 잰다 — 「언제 바뀌는가」의 표다."""
    t = wide(d, city, "T", LEVELS=LEVELS) - 273.15
    r = wide(d, city, "RH", LEVELS=LEVELS)
    out = pd.DataFrame(
        {
            "예측시간": t.columns,
            f"{lo}hPa 기온": t.loc[lo].values.round(1),
            f"{hi}hPa 기온": t.loc[hi].values.round(1),
            f"{lo}hPa 습도": r.loc[lo].values.round(0),
            f"{hi}hPa 습도": r.loc[hi].values.round(0),
        }
    )
    out["층후 기온차"] = (out[f"{lo}hPa 기온"] - out[f"{hi}hPa 기온"]).round(1)
    return out
```

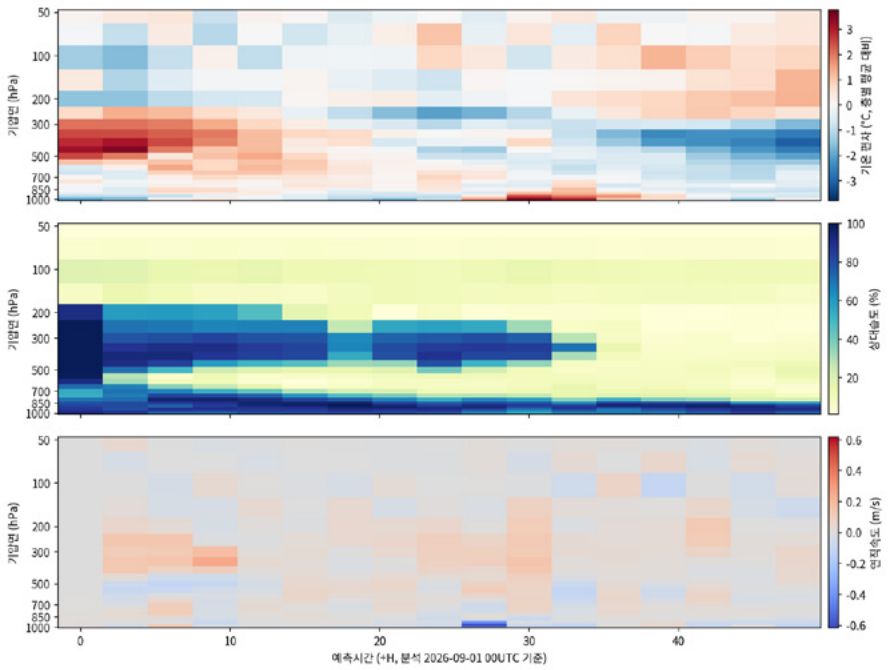


그림 29-3-2. 시간·고도별 기온과 습도 변화

예측시간	1000hPa 기온	500hPa 기온	1000hPa 습도	500hPa 습도	층후 기온차
0	23.1	-5.2	95	100	28.3
3	23.7	-5.8	85	71	29.5
6	25.5	-7.3	71	50	32.8
42	24.6	-9.1	74	10	33.7
45	23.7	-9.5	83	8	33.2
48	25.5	-9.8	73	15	35.3

단·중기모델

❹ U·V 성분으로 층별 바람 표시

동서·남북 바람 성분을 바람깃으로 표시하고 풍속을 배경색으로 나타냅니다.

```
def wind_frame(d, city, *, LEVELS):
    u = wide(d, city, "U", LEVELS=LEVELS)
    v = wide(d, city, "V", LEVELS=LEVELS)
    sp = np.hypot(u, v)
    return (u, v, sp)
def jet_table(d, *, CITIES, LEVELS):
    """층마다 가장 센 바람이 어디에 있었는지. 상층 제트의 높이와 세기가 나란환됩니다."""
    rows = []
    for nm, *_ in CITIES:
        _, _, sp = wind_frame(d, nm, LEVELS=LEVELS)
        for h in sp.columns:
            col = sp[h].dropna()
            up = col[col.index <= 400]
            low = col[col.index >= 850]
            rows.append(
                {
                    "도시": nm,
                    "예측시간": h,
                    "상층최대(m/s)": round(float(up.max()), 1),
                    "상층높이(hPa)": int(up.idxmax()),
                    "하층최대(m/s)": round(float(low.max()), 1),
                    "하층높이(hPa)": int(low.idxmax()),
                }
            )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "도시",
                "예측시간",
                "상층최대(m/s)",
                "상층높이(hPa)",
                "하층최대(m/s)",
                "하층높이(hPa)",
            ]
        ),
    )
```

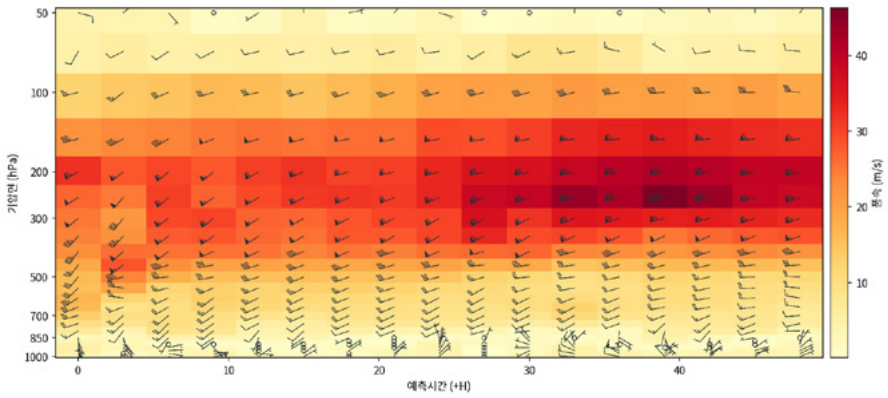


그림 29-3-3. U·V 성분으로 층별 바람 표시

도시	상층최대	상층평균	주된높이	하층최대
강릉	47.8	36.2	250	13.1
부산	34.9	26.5	200	10.7
서울	46.3	36	250	5.1
제주	30.5	24.2	200	5.8

❺ 층후와 기온감률 비교

```
def thickness(d, city, lo=1000, hi=500, *, LEVELS):
    """측고 공식 Z=(R/g)·T̄·ln(p_lo/p_hi)으로 두 기압면 사이의 층후를 계산합니다."""

    t = wide(d, city, "T", LEVELS=LEVELS)
    seg = t.loc[[l for l in t.index if hi <= l <= lo]]
    tbar = seg.mean()
    return R_DRY / G * tbar * np.log(lo / hi)
def lapse_frame(d, city, lo=1000, hi=850, *, LEVELS):
    """하층 기온감률. 층이 뒤집혀 있으면(위가 더 따뜻하면) 값이 음이 된다."""
    t = wide(d, city, "T", LEVELS=LEVELS) - 273.15
    dz = thickness(d, city, lo, hi, LEVELS=LEVELS)
    return ((t.loc[lo] - t.loc[hi]) / dz * 1000).rename("감률(℃/km)")
def stability_table(d, *, CITIES, LEVELS):
    rows = []
    for nm, *_ in CITIES:
        th = thickness(d, nm, LEVELS=LEVELS)
        lp = lapse_frame(d, nm, LEVELS=LEVELS)
        rows.append(
            {
                "도시": nm,
                "층후 최소(m)": round(float(th.min())),
                "층후 최대(m)": round(float(th.max())),
                "층후 폭(m)": round(float(th.max() - th.min())),
                "감률 최소(℃/km)": round(float(lp.min()), 1),
                "감률 최대(℃/km)": round(float(lp.max()), 1),
                "감률 중앙(℃/km)": round(float(lp.median()), 1),
                "가장 두꺼운 때(+H)": int(th.idxmax()),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "도시",
                "층후 최소(m)",
                "층후 최대(m)",
                "층후 폭(m)",
                "감률 최소(℃/km)",
                "감률 최대(℃/km)",
                "감률 중앙(℃/km)",
                "가장 두꺼운 때(+H)",
            ]
        ),
    )
```

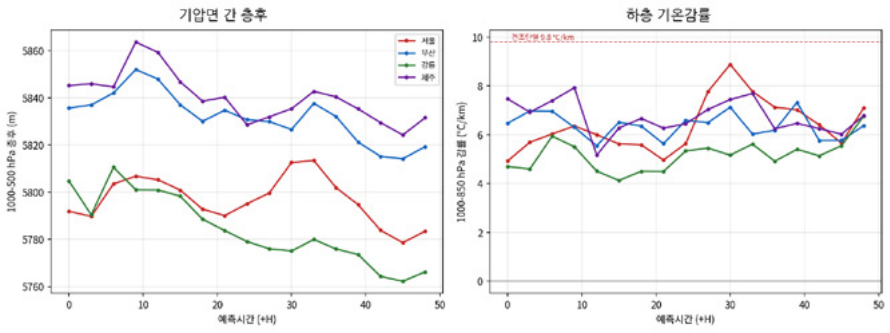


그림 29-3-4. 층후와 기온감률 비교

단·중기모델

도시	층후 최소(m)	층후 최대(m)	층후 폭(m)	감률 최소(℃/km)	감률 최대(℃/km)	감률 중앙(℃/km)	가장 두꺼운 때(+H)
서울	5779	5813	35	4.9	8.9	6	33
부산	5814	5852	38	5.6	7.3	6.4	9
강릉	5762	5811	48	4.1	6.8	5.2	6
제주	5824	5864	39	5.2	7.9	6.7	9

❻ 고습층으로 구름 가능 구간 추정

상대습도가 높은 층이 연속된 구간을 묶어 하단과 두께를 구합니다.

```
def cloud_layers(d, city, thr=80, *, LEVELS):
    """상대습도 기준값을 넘는 연속 기압층의 하단·상단과 두께를 구합니다. 실제 구름 관측과는 구분합니다."""

    r = wide(d, city, "RH", LEVELS=LEVELS)
    rows = []
    for h in r.columns:
        col = r[h]
        wet = [
            l for l in LEVELS if np.isfinite(col.get(l, np.nan)) and col[l] >= thr
        ]
        if not wet:
            continue
        blocks, cur = ([], [wet[0]])
        for a, b in zip(wet, wet[1:]):
            if LEVELS.index(b) - LEVELS.index(a) == 1:
                cur.append(b)
            else:
                blocks.append(cur)
                cur = [b]
        blocks.append(cur)
        for bl in blocks:
            rows.append(
                {
                    "도시": city,
                    "예측시간": h,
                    "밑면(hPa)": max(bl),
                    "윗면(hPa)": min(bl),
                    "층수": len(bl),
                    "최고습도(%)": round(float(col[bl].max()), 0),
                }
            )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "도시",
                "예측시간",
                "밑면(hPa)",
                "윗면(hPa)",
                "층수",
                "최고습도(%)",
            ]
        ),
    )
```

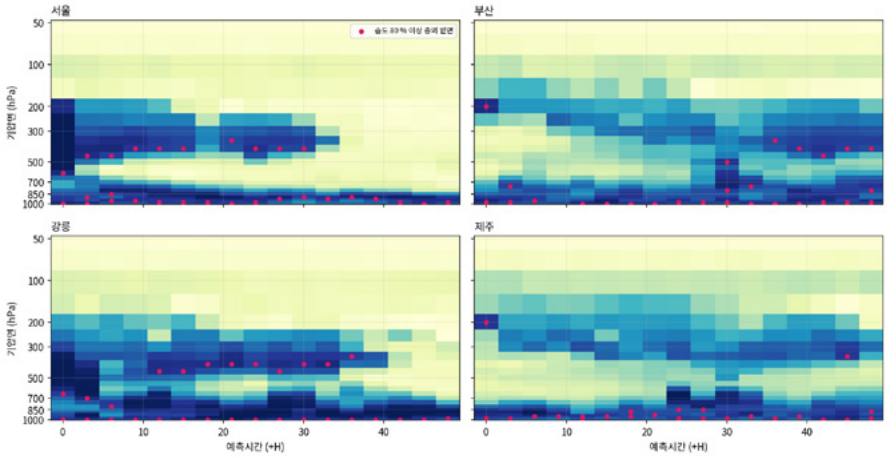


그림 29-3-5. 고습층으로 구름 가능 구간 추정

도시	덩어리수	평균층수	가장두꺼운층수	가장낮은밑면	가장높은윗면
강릉	29	5.9	11	1000	300
부산	27	3.9	9	1000	200
서울	29	4.1	9	1000	200
제주	24	3.8	10	1000	200

초단기모델

30. 초단기모델(KLAPS) 분석 자료

예측시간별 강수 분포 비교

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 초단기예측

❶ 이미지 요청과 응답

```
GET /api/typ03/cgi/dfs/nph-qpf_ana_img?eva=1&tm=202212221350&qpf=B&ef=60
  &map=HR&grid=2&legend=1&size=600&zoom_level=0&stn=108&authKey=...
# 예제 작성 당시 응답 예시 — Content-Type: image/png, 38,040 bytes / 실행 시 Content-Type을 반드시 확인
```

주요 요청 파라미터는 다음과 같습니다.

필드	값	의미
qpf	K / M / B	자료 소스: K(초단기모델·KLAPS), M(레이더외삽), B(융합예측)
tm	202212221350	기준시각(202212221350)
ef	0, 1, 2 ...	기준시각으로부터의 예측시간(분)
eva	0.1/1/5	범례 평가기준: 0.1mm/1mm/5mm
map, grid, size	HR, 2, 600	지도 코드·격자 크기(km)·이미지 픽셀 크기

❷ 수신 이미지의 강수 분포

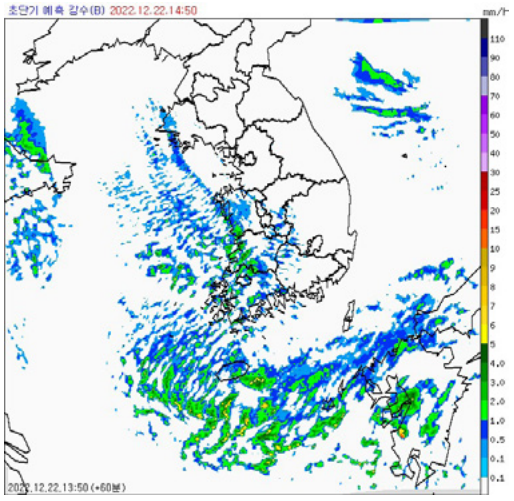


그림 30-1. 수신 이미지의 강수 분포

❸ K·B 모드의 응답 비교

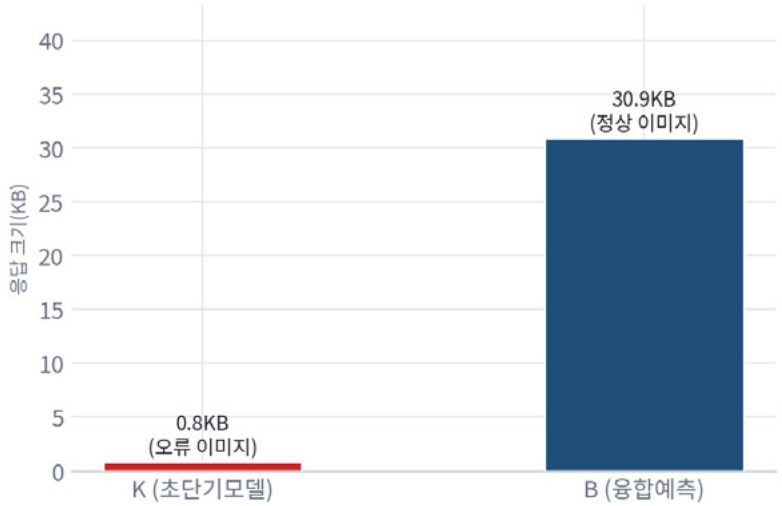


그림 30-2. K·B 모드의 응답 비교

❹ 예측시간(ef)별 응답 확인

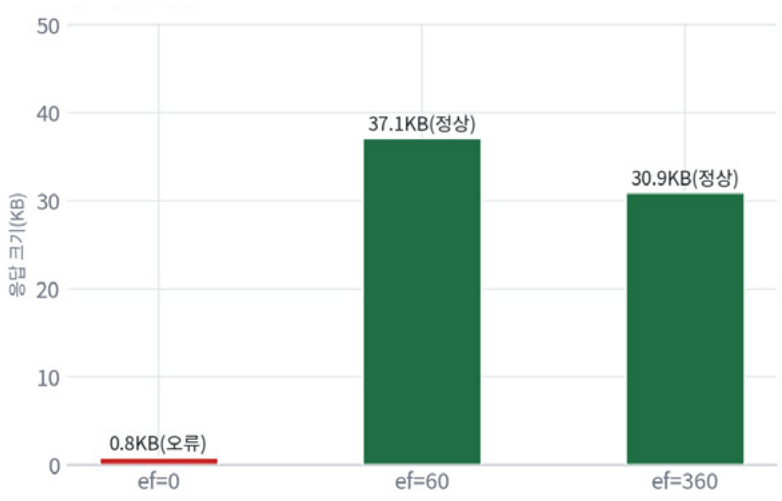


그림 30-3. 예측시간(ef)별 응답 확인

초단기모델

❺ 이미지 저장과 강수 영역 지표

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
```

```
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
IMG_URL = "https://apihub.kma.go.kr/api/typ03/cgi/dfs/nph-qpf_ana_img"
def fetch_and_score(tm, ef_list, auth_key, out_dir="qpf_images", gap_sec=0.3):
    """이미지 파일 저장; 지도 색상에서 강수면적을 추정하지 않는다."""
    folder = Path(out_dir)
    folder.mkdir(parents=True, exist_ok=True)
    saved = {}
    for ef in ef_list:
        r = requests.get(
            IMG_URL,
            params={"tm": tm, "qpf": "B", "ef": ef, "authKey": auth_key},
            timeout=60,
        )
        r.raise_for_status()
        if not r.headers.get("Content-Type", "").lower().startswith("image/"):
            raise ValueError("응답 형식: image/*")
        suffix = (
            ".png"
            if r.content.startswith(b"\x89PNG")
            else ".jpg" if r.content.startswith(b"\xff\xd8") else None
        )
        if suffix is None:
            raise ValueError("이미지 형식: PNG 또는 JPEG")
        path = folder / f"qpf_B_{tm}_{ef}{suffix}"
        path.write_bytes(r.content)
        saved[ef] = str(path)
        time.sleep(gap_sec)
    return saved
```

31.
지역파랑모델(RWW3) 예측자료

과거 해구별 파고와 바람 비교

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 수치예보모델

❶ 소해구별 파랑예측 조회

※ UM 기반 파고모델은 2026-03-31 이후 신규 생산·제공이 중단되었습니다. 이 사례는 종료 이전 과거자료를 조회·분석하는 용도로 사용합니다.

```
https://apihub.kma.go.kr/api/typ06/url/marine_small_zone.php
 ?tma_fc=2025012500&tma_ef=2025012609
 &Lzone=1,2,3,...&Szone=0&disp=0&help=0&authKey=...
# TMA_FC   TMA_EF  LZONE SZONE  LAT_LB  LON_LB ... WH_SIG WVPRD_MAX  WVDR   WS    WD
TW SWELL
 # 실행 시 지정한 과거시각의 소해구별 예측값이 반환됩니다.
 # 2026-03-31 이후 시각은 신규 생산자료가 아니므로 과거자료 조회에 사용하지 않습니다.
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)

import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
SZ_URL = "https://apihub.kma.go.kr/api/typ06/url/marine_small_zone.php"
SWELL_NAME = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

SEAS : 사용자가 지정한 분석 구역 (이름,경도하한,경도상한,위도하한,위도상한) 목록. 공식 해구와 다름

필수 입력: SEAS. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.

조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_small_zone(lzones, tma_ef, tma_fc=None):
  if tma_fc is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: tma_fc")
  "소해구 예측을 받는다. 대해구 번호는 한 번에 40개까지만 받으므로\n    번호 묶음을 40개씩 잘라 부르고, Szone=0 으로 그 안 9칸을 모두 받는다."
  r = requests.get(
    SZ_URL,
    params={
      "tma_fc": tma_fc,
      "tma_ef": tma_ef,
      "Lzone": ",".join(map(str, lzones)),
      "Szone": 0,
      "disp": 0,
      "help": 0,
```

파랑모델

```
        "authKey": API_KEY,
      },
      timeout=120,
    )
    if not r.ok:
      raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
    return r.text

def parse_zone(text):
    """주석을 제외하고 소해구별 자료를 읽습니다. 앞 12개 필드는 발표·발효시각과 모서리 좌표, 뒤 9개는 예측값입니다."""

    rows = []
    for ln in text.splitlines():
      ln = ln.strip()
      if not ln or ln.startswith("#"):
        continue
      c = ln.split()
      if len(c) < 21:
        continue
      rows.append(c)
    return rows
```

❷ 대해구·소해구 좌표 구조

```
def cell_size(base):
    """한 대해구 안 9칸의 위경도 간격에서 소해구 한 칸의 크기를 잰다."""
    g = base.groupby("대해구")
    dlat = g["위도"].apply(lambda s: np.diff(np.unique(s)).mean())
    dlon = g["경도"].apply(lambda s: np.diff(np.unique(s)).mean())
    return pd.DataFrame({"위도간격": dlat, "경도간격": dlon}).median()
def zone_table(base):
    """대해구 하나를 골라 그 안 소해구 9칸의 자리를 표로 만든다."""
    z = base["대해구"].iloc[0]
    d = base[base["대해구"] == z].copy()
    d["자리"] = [
      "위-왼",
      "위-가운데",
      "위-오른",
      "가운데-왼",
      "가운데-가운데",
      "가운데-오른",
      "아래-왼",
      "아래-가운데",
      "아래-오른",
    ][: len(d)]
    return d[["대해구", "소해구", "자리", "위도", "경도"]]
```

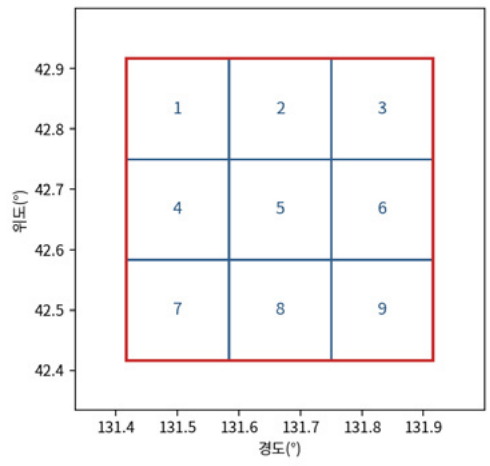


그림 31-1. 대해구·소해구 좌표 구조

대해구	소해구	자리	위도	경도
1	1	위-왼	42.833	131.5
1	2	위-가운데	42.833	131.667
1	3	위-오른	42.833	131.833
1	4	가운데-왼	42.667	131.5
1	5	가운데-가운데	42.667	131.667
1	6	가운데-오른	42.667	131.833
1	7	아래-왼	42.5	131.5
1	8	아래-가운데	42.5	131.667
1	9	아래-오른	42.5	131.833

❸ 유의파고·최대파주기 분포

```
def to_grid(base, values):
  """소해구 좌표를 위경도 격자판으로 되변환합니다."""
  lons = np.unique(base["경도"].values)
  lats = np.unique(base["위도"].values)
  gi = {v: i for i, v in enumerate(lons)}
  gj = {v: i for i, v in enumerate(lats)}
  g = np.full((len(lats), len(lons)), np.nan)
  for lo, la, v in zip(base["경도"], base["위도"], values):
    g[gj[la], gi[lo]] = v
  return (lons, lats, g)
def sea_of(lon, lat, *, SEAS):
  """해역 상자로 서해·남해·동해를 가른다."""
  out = np.full(len(lon), "기타", dtype=object)
  for name, lo1, lo2, la1, la2 in SEAS:
    m = (lon >= lo1) & (lon < lo2) & (lat >= la1) & (lat < la2)
    out[m & (out == "기타")] = name
  return out
def sea_table(f, *, SEAS):
  """해역마다 유의파고·풍속·수온을 요약한다."""
  d = f.copy()
  d["해역"] = sea_of(d["경도"].values, d["위도"].values, SEAS=SEAS)
  g = d[d["해역"] != "기타"].groupby("해역")
  t = pd.DataFrame(
    {
      "칸수": g.size(),
      "유의파고 중앙값(m)": g["유의파고"].median().round(2),
      "유의파고 최대(m)": g["유의파고"].max().round(2),
      "풍속 중앙값(m/s)": g["풍속"].median().round(1),
      "수온 중앙값(℃)": g["수온"].median().round(1),
    }
  )
  return t.reset_index()
```

파랑모델

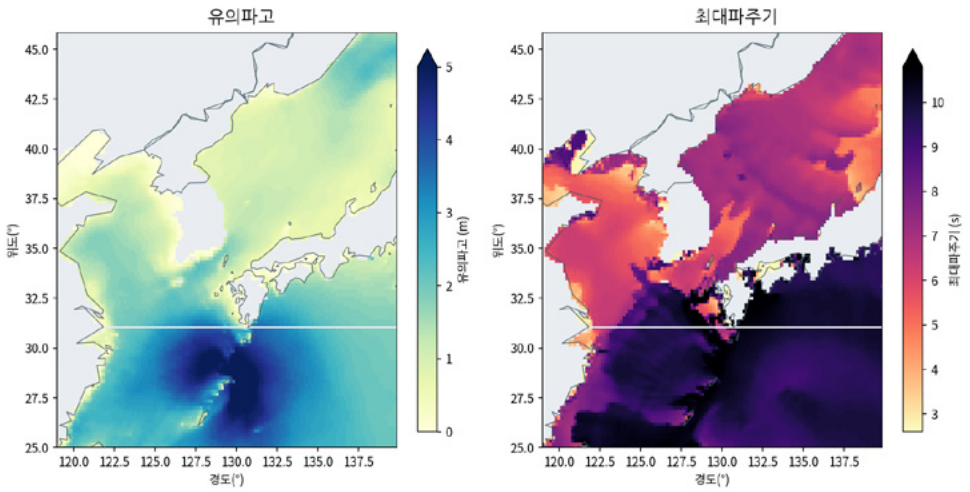


그림 31-2. 유의파고·최대파주기 분포

해역	칸수	유의파고 중앙값(m)	유의파고 최대(m)	풍속 중앙값(m/s)	수온 중앙값(℃)
남해	324	1.8	2.4	10.4	29.5
동해	612	1.1	1.9	7.2	27.4
서해	450	1.1	1.9	7.9	27.7

❹ 파고와 풍속·수온의 관계

```
def wind_wave_frame(f, step=1.0):
    """풍속 구간별 유의파고 중앙값을 비교합니다."""

    d = f.dropna(subset=["풍속", "유의파고"]).copy()
    d["풍속구간"] = d["풍속"] // step * step
    g = d.groupby("풍속구간")["유의파고"]
    t = pd.DataFrame(
        {
            "칸수": g.size(),
            "파고중앙값": g.median().round(2),
            "파고90%": g.quantile(0.9).round(2),
        }
    )
    return t[t["칸수"] >= 30].reset_index()
def period_corr(f):
    """유의파고와 최대파주기·풍속의 상관을 계산한다."""
    d = f[["유의파고", "최대파주기", "풍속", "수온"]].dropna()
    return d.corr()["유의파고"].drop("유의파고").round(3)
```

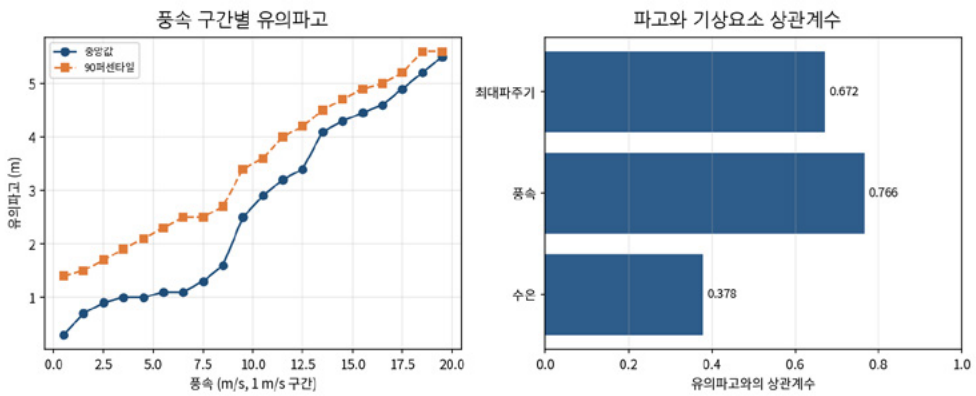


그림 31-3. 파고와 풍속·수온의 관계

풍속구간	칸수	파고중앙값	파고90%
0	68	0.3	1.4
1	252	0.7	1.5
2	474	0.9	1.7
11	478	3.2	4
12	351	3.4	4.2
13	189	4.1	4.5

❺ 시간별 파고 변화와 최고 시각

```
def to_hours(efs, tma_fc=None):
  if tma_fc is None:
    raise ValueError("조회조건·메타데이터를 명시하세요: tma_fc")
  "발효시각 문자열을 발표시각으로부터의 예측시간(h)으로 바꾼다."
  t0 = datetime.strptime(tma_fc, "%Y%m%d%H")
  return np.array(
    [
      (datetime.strptime(e, "%Y%m%d%H") - t0).total_seconds() / 3600.0
      for e in efs
    ]
  )
def lead_series(wh, hours):
  """예측시각마다 전 소해구의 파고 분포를 요약한다."""
  return pd.DataFrame(
    {
      "예측시간(h)": hours.astype(int),
      "중앙값(m)": np.nanmedian(wh, axis=1).round(2),
      "90퍼센타일(m)": np.nanpercentile(wh, 90, axis=1).round(2),
      "최대(m)": np.nanmax(wh, axis=1).round(2),
      "2m0상 칸수": (wh >= 2.0).sum(axis=1),
    }
  )
def peak_frame(wh, hours, base):
  """칸마다 파고가 가장 높아지는 시각을 찾는다."""
  ok = ~np.all(np.isnan(wh), axis=0)
  idx = np.nanargmax(np.where(np.isnan(wh), -1, wh), axis=0)
  d = base.copy()
  d["최고파고"] = np.nanmax(wh, axis=0)
  d["최고시각"] = hours[idx]
  d.loc[~ok, ["최고파고", "최고시각"]] = np.nan
  return d
```

파랑모델

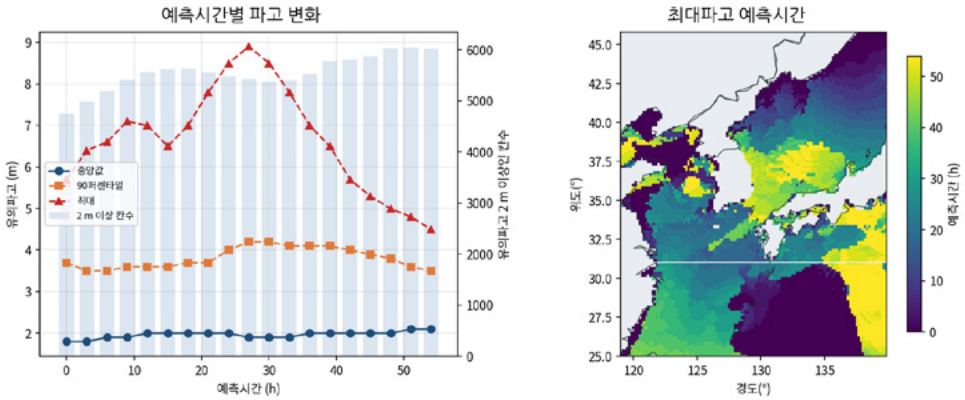


그림 31-4. 시간별 파고 변화와 최고 시각

예측시간(h)	중앙값(m)	90퍼센타일(m)	최대(m)	2m이상 칸수
0	1.8	3.7	5.7	4752
3	1.8	3.5	6.4	4988
6	1.9	3.5	6.6	5193
48	2	3.8	5	6043
51	2.1	3.6	4.8	6053
54	2.1	3.5	4.5	6028

❻ 너울 위험등급 분포

너울 위험예측은 1 관심·2 주의·3 경계·4 위험으로 구분됩니다.

```
def swell_table(f):
  d = f.dropna(subset=["너울위험"]).copy()
  d["등급"] = d["너울위험"].astype(int)
  g = (
    d.groupby("등급")
    .agg(
      칸수=("너울위험", "size"),
      유의파고중앙값=("유의파고", "median"),
      최대파주기중앙값=("최대파주기", "median"),
      풍속중앙값=("풍속", "median"),
    )
    .reset_index()
  )
  g["이름"] = g["등급"].map(lambda c: SWELL_NAME.get(c, str(c)))
  return g.round(2)
```

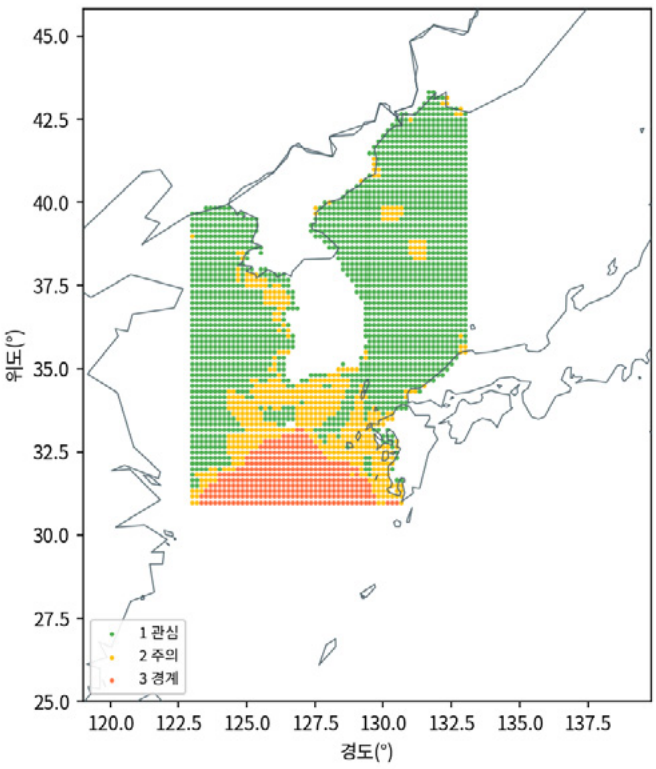


그림 31-5. 너울 위험등급 분포

등급	이름	칸수	유의파고 중앙값(m)	최대파주기 중앙값(s)	풍속 중앙값(m/s)
1	관심	2021	1	6.9	6.6
2	주의	523	1	6.8	7.5
3	경계	333	2.3	9.7	10.2

해구별예측

32.
해구별 예측정보(대해구·소해구)

과거 해구별 파고 변화 비교

데이터 활용 경로(API)

기상청 API허브 > 수치모델 > 수치예보모델

❶ 해구별 예측자료 읽기

**※ 이 API는 UM 기반 RWW3 해구자료의 과거자료 조회용으로 사용합니다.
2026-03-31 이후 신규 파고모델 생산은 중단되었습니다.**

```
https://apihub.kma.go.kr/api/typ06/url/marine_large_zone.php?tma_fc=2025012500&tma_ef=2025012600&Lzone=0&disp=0&authKey=...
# TMA_FC   TMA_EF  LZONE  LAT_LB  LON_LB  …  WH_SIG  WVPRD_MAX  WVDR  WS   WD
 # 실행 시 지정한 과거시각의 대해구별 예측값이 반환됩니다.
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
ZONE_NAME = {} # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
SWELL = {} # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

범주와 상태는 공식 코드표로 확인합니다. 코드표에 없는 값은 원래 코드로 표시합니다.

```
def parse_zone(text, small=False):
    """한 행이 해구 하나다. 좌하단·좌상단·우상단·우하단 네 모서리 다음에 유의파고·최대파주기·파향·풍속·풍향이 반환됩니다.
-999.0 은 자료 없음이다."""

    rows = []
    for line in text.splitlines():
        s = line.strip()
        if not s or s.startswith("#"):
            continue
        p = s.split()
        k = 4 if small else 3
        if len(p) < k + 13:
            continue
        v = [float(x) for x in p[k:]]
        d = {"발표시각": p[0], "발효시각": p[1], "대해구": int(p[2])}
        if small:
            d["소해구"] = int(p[3])
        d.update(
            {
                "좌하위도": v[0],
                "좌하경도": v[1],
                "유의파고(m)": v[8],
                "최대파주기(s)": v[9],
                "파향(°)": v[10],
```

```
            "풍속(m/s)": v[11],
            "풍향(°)": v[12],
          }
        )
        if small and len(v) >= 17:
          d.update(
            {
              "시정(m)": v[13],
              "강수(mm)": v[14],
              "수온(℃)": v[15],
              "너울": int(v[16]),
            }
          )
        rows.append(d)
    d = pd.DataFrame(rows)
    return d.replace(-999.0, np.nan)
```

❷ 해구별 파고 지도 작성

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
import os, math, re, time, io, gzip, struct, zipfile
def load_grid(path):
    """받아 둔 대해구 격자자료. 여섯 자리 36진수에 위치와 값이 눌러 담겨 있다."""
    txt = Path(path).read_text("utf-8").splitlines()
    grid, zones, sec = ("", "", None)
    ser, sz = ([], [])
    for line in txt:
        s = line.strip()
        if s.startswith("#"):
            continue
        if s.startswith("@"):
            sec = s.split()[0][1:]
            continue
        if not s:
            continue
        if sec == "LGRID":
            grid += (" " if grid else "") + s
        elif sec == "LZONE":
            zones += (" " if zones else "") + s
        elif sec == "SERIES":
            p = s.split()
            if len(p) == 6:
                ser.append(
                    {
                        "예측시간(h)": int(p[0]),
                        "대해구": int(p[1]),
                        "유의파고(m)": float(p[2]),
                        "최대파주기(s)": float(p[3]),
                        "풍속(m/s)": float(p[4]),
                        "풍향(°)": float(p[5]),
                    }
                )
        elif sec == "SZONE":
            p = s.split()
            if len(p) == 12:
                sz.append(
                    {
                        "대해구": int(p[0]),
                        "소해구": int(p[1]),
                        "위도": float(p[2]),
                        "경도": float(p[3]),
                        "유의파고(m)": float(p[4]),
                        "최대파주기(s)": float(p[5]),
                        "풍속(m/s)": float(p[6]),
                        "풍향(°)": float(p[7]),
                        "시정(m)": float(p[8]),
                        "강수(mm)": float(p[9]),
                        "수온(℃)": float(p[10]),
                        "너울": int(p[11]),
                    }
                )
    rows = []
    zl = [int(x) for x in zones.split()]
    for tok, z in zip(grid.split(), zl):
```

해구별예측

```
        if len(tok) != 12 or "-" in tok:
            continue
        b = [int(tok[i : i + 2], 36) for i in range(0, 12, 2)]
        rows.append(
            {
                "대해구": z,
                "위도": 25 + b[1] / 2,
                "경도": 119 + b[2] / 2,
                "유의파고(m)": b[3] / 10,
                "풍속(m/s)": float(b[4]),
                "풍향(°)": b[5] * 10.0,
            }
        )
    L = pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else ["대해구", "위도", "경도", "유의파고(m)", "풍속(m/s)", "풍향(°)"]
        ),
    )
    s = pd.DataFrame(sz).replace(-999.0, np.nan)
    return (L, pd.DataFrame(ser), s)
def grid_summary(L):
    return {
        "대해구 수": len(L),
        "덮는 범위": f"위도 {L['위도'].min():.1f}~{L['위도'].max() + 0.5:.1f}, 경도 {L['경도'].min():.1f}~{L['경도'].max() +
0.5:.1f}",
        "한 칸 크기": "0.5도 × 0.5도 (위도 35도에서 약 56 × 46 km)",
        "유의파고 범위": f"{L['유의파고(m)'].min():.1f} ~ {L['유의파고(m)'].max():.1f} m",
        "1.5 m 이상 칸": f"{int((L['유의파고(m)'] >= 1.5).sum())}칸 ({(L['유의파고(m)'] >= 1.5).mean() * 100:.0f} %)",
        "풍속 범위": f"{L['풍속(m/s)'].min():.0f} ~ {L['풍속(m/s)'].max():.0f} m/s",
    }
```

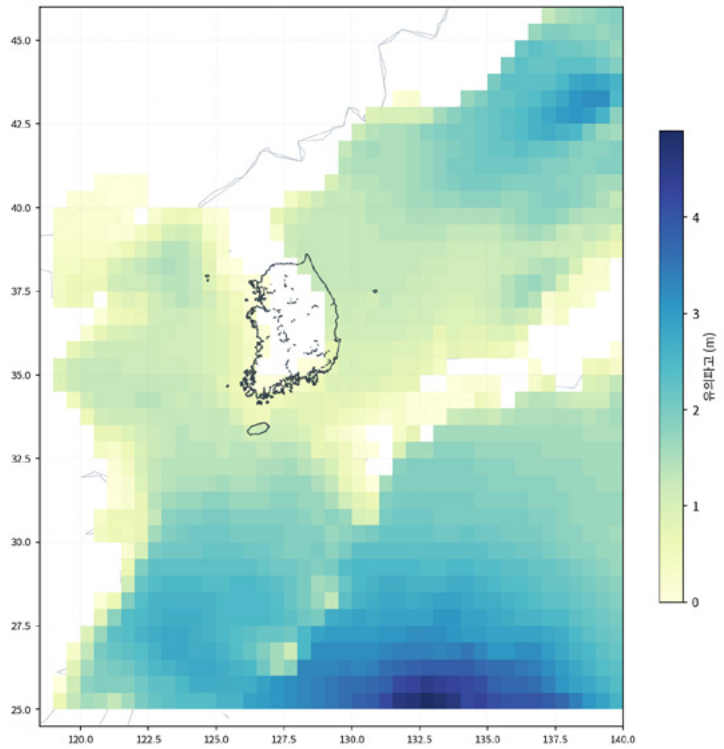


그림 32-1. 해구별 파고 지도 작성

❸ 파고 지도에 풍향·풍속 표시

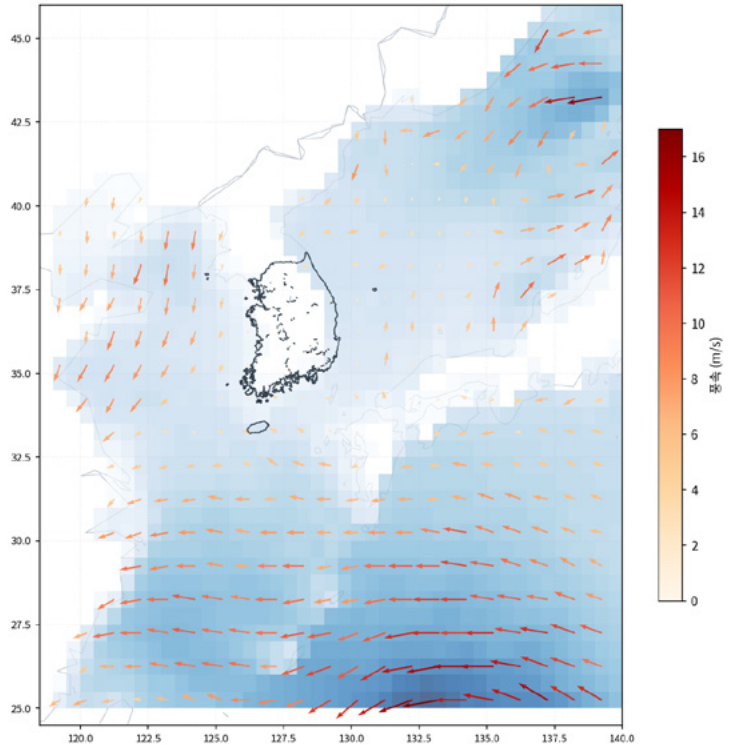


그림 32-2. 파고 지도에 풍향·풍속 표시

❹ 해구별 3시간 간격 파고 변화

```
def series_table(S):
    p = S.pivot_table(index="예측시간(h)", columns="대해구", values="유의파고(m)")
    p.columns = [f"{c} {ZONE_NAME.get(c, '')}".strip() for c in p.columns]
    return p.reset_index()
```

예측시간(h)	53 동해 북부	64 동해 중부(강릉 앞)	98 남해 동부(통영 앞)	173 서해 중부(태안 앞)	191 서해 남부	233 제주 남부
0	1.6	1.2	0.5	1	1.6	1
6	1.8	1.2	0.4	1	1.3	1.1
12	1.7	1.4	0.4	0.8	1	1.2
60	1.4	1.3	0.6	0.9	1.4	1.7
66	1.4	1.6	0.6	1	1.6	2.1
72	1.8	2	0.8	1	1.9	3

해구별예측

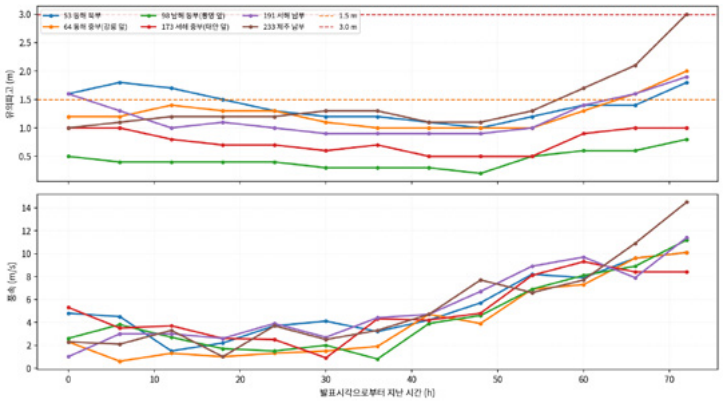


그림 32-3. 해구별 3시간 간격 파고 변화

❺ 소해구의 시정·강수량·수온·너울 비교

```
def szone_table(sz):
  g = sz.groupby("대해구").agg(
    소해구=("소해구", "count"),
    파고최소=("유의파고(m)", "min"),
    파고최대=("유의파고(m)", "max"),
    수온=("수온(℃)", "mean"),
    너울최대=("너울", "max"),
  )
  g["해구명"] = [ZONE_NAME.get(i, "") for i in g.index]
  g["너울등급"] = g["너울최대"].map(
    lambda x: SWELL.get(int(x), "-") if pd.notna(x) else "-"
  )
  return g.round(2).reset_index()
```

대해구	소해구	파고최소	파고최대	수온	너울최대	해구명	너울등급
53	9	1.3	1.4	26.17	1	동해 북부	관심
64	9	1.3	1.3	26.6	2	동해 중부(강릉 앞)	주의
98	9	0	0.6	29.28	2	남해 동부(통영 앞)	주의
173	9	0.6	0.8	27.88	1	서해 중부(태안 앞)	관심
191	9	1	1.1	29.78	1	서해 남부	관심
233	9	0.9	1.3	29.46	2	제주 남부	주의

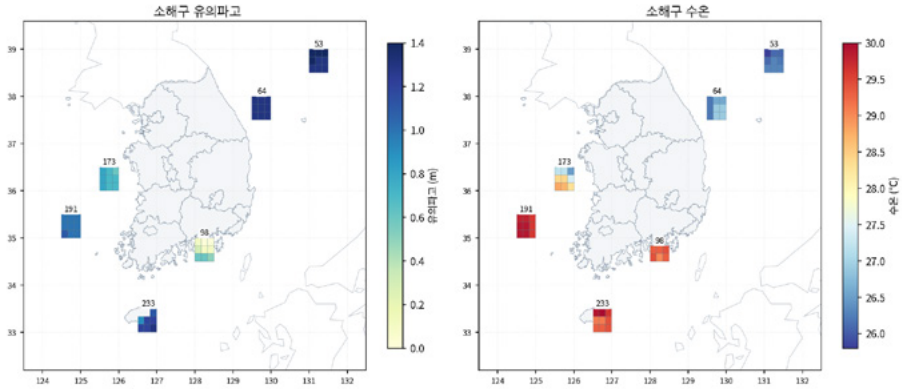
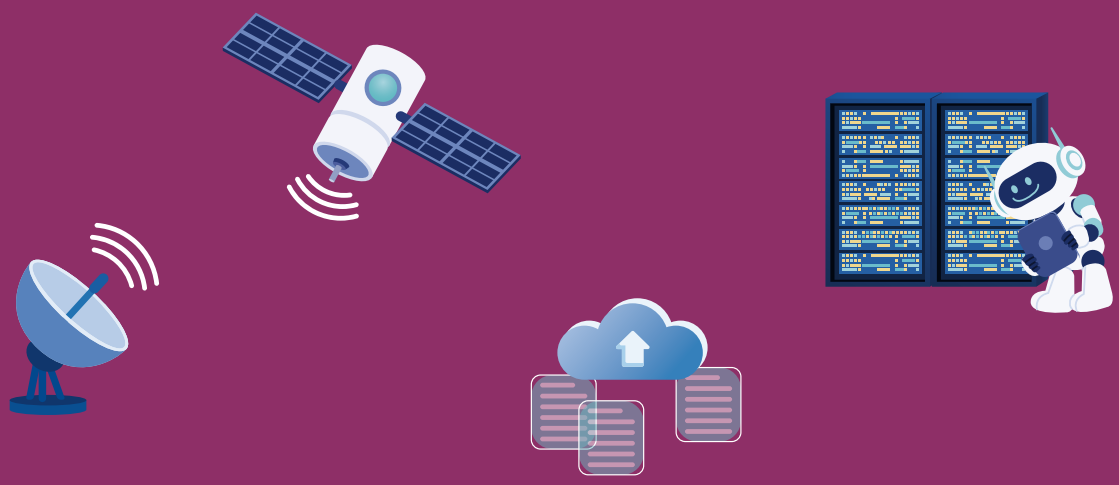


그림 32-4. 소해구의 시정·강수량·수온·너울 비교

데이터 활용사례 50

기후

Climate





지구대기 감시

33. 온실가스

감시소별 농도 변화 비교

데이터 활용 경로(API)

데이터 > 기후 > 기후변화감시 > 온실가스

❶ CSV 읽기와 자료 확인

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
KST = timezone(timedelta(hours=9))
log = print
def read_greenhouse(path, encoding="cp949"):
  d = pd.read_csv(path, encoding=encoding)
  if "일시" not in d:
    raise ValueError("일시 열이 있는 포털 CSV를 사용하세요")
  d["일시"] = pd.to_datetime(d["일시"], errors="coerce")
  return d.dropna(subset=["일시"])
def monthly_greenhouse(d, column):
  """공식 CSV의 물질명·단위 열을 지정. 감시소별 유효값 월평균."""
  if column not in d:
    raise ValueError("column에는 CSV에 있는 농도 열 이름을 지정하세요")
  x = d.copy()
  x[column] = pd.to_numeric(x[column], errors="coerce")
  x.loc[x[column] < 0, column] = np.nan
  x["월"] = x["일시"].dt.to_period("M").dt.to_timestamp()
  keys = [c for c in ["지점", "지점명", "월"] if c in x]
  return (
    x.groupby(keys, dropna=False)[column]
    .agg(월평균="mean", 유효자료수="count")
    .reset_index()
  )
```

실행 예: d = read_greenhouse(CSV_PATH); print(d.columns.tolist()); t = monthly_greenhouse(d, "평균 이산화탄소(CO2) 배경대기농도(ppm)"). 농도 열 이름과 단위는 다운로드한 CSV 헤더 그대로 지정합니다. 서로 다른 물질·감시소·시간 해상도를 합쳐 평균하지 않습니다.

```
CSV_PATH = Path(input('다운로드한 온실가스 CSV 경로: '))
df = pd.read_csv(CSV_PATH, encoding='cp949')
# 기존 문서의 파일 예시 내용 (앞 2행)
지점,지점명,일시,평균 이산화탄소(CO2) 배경대기농도(ppm),평균 메탄(CH4) 배경대기농도(ppb), ...
115,울릉도독도(감),2022-01-01,424.5,2014,335.873342560615,,,,
115,울릉도독도(감),2022-01-02,424.5,2014,335.87506749116,,,,
# 결측 현황 (df.isna().sum()) — CFC류는 안면도 1개소에서만 관측
CFC11  2191 / 2937 결측  CFC12  2057 / 2937 결측  CFC113  2251 / 2937 결측
CO2·CH4·N2O·SF6는 3개소 모두 관측, 결측 거의 없음
```

감시소	관측 시작	관측 종료	일별 자료 수	관측 물질
안면도(감)	2022-01-01	2024-09-05	979	CO2·CH4·N2O·CFC11·CFC12·CFC113·SF6 (7종)
울릉도독도(감)	2022-01-01	2024-09-05	979	CO2·CH4·N2O·SF6 (4종)
제주고산(감)	2022-01-01	2024-09-05	979	CO2·CH4·N2O·SF6 (4종)

❷ CO2 월평균의 계절 변화와 상승 추세

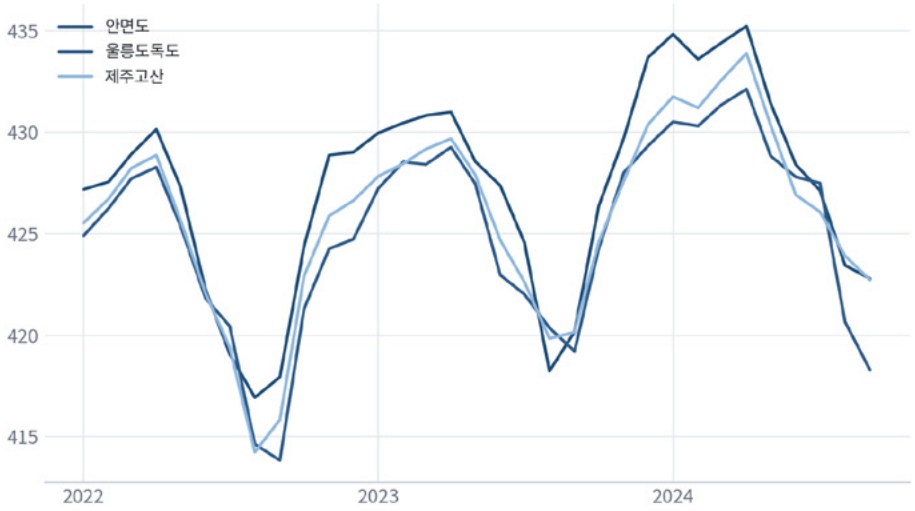


그림 33-1. CO2 월평균의 계절 변화와 상승 추세

❸ 7종 물질의 2022~2024년 농도 변화

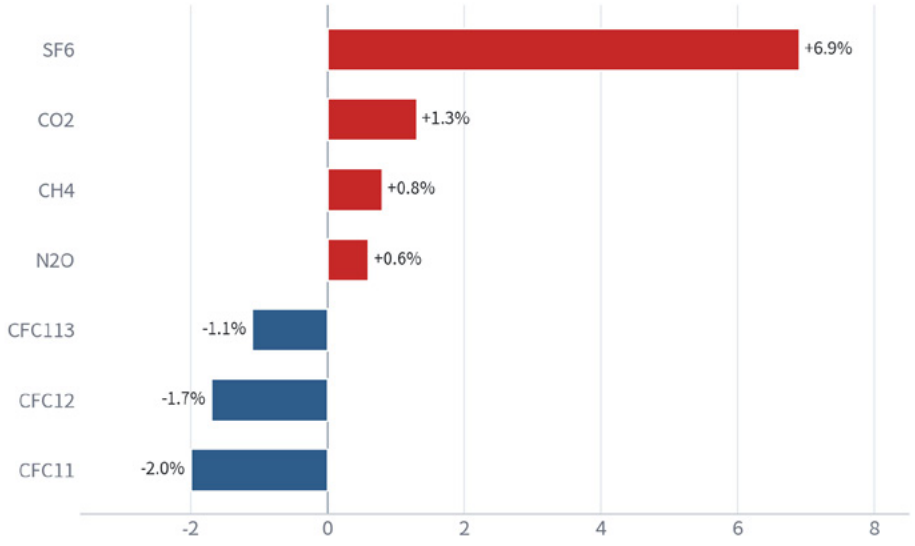


그림 33-2. 7종 물질의 2022~2024년 농도 변화

지구대기 감시

❹ 감시소별 CO2·N2O·CH4 농도 비교

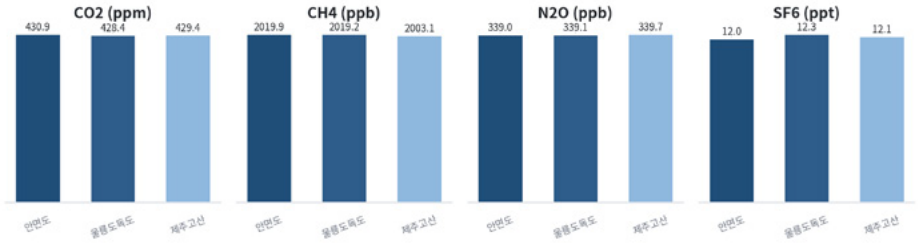


그림 33-3. 감시소별 CO2·N2O·CH4 농도 비교

❺ 안면도의 CFC 농도 감소 추세

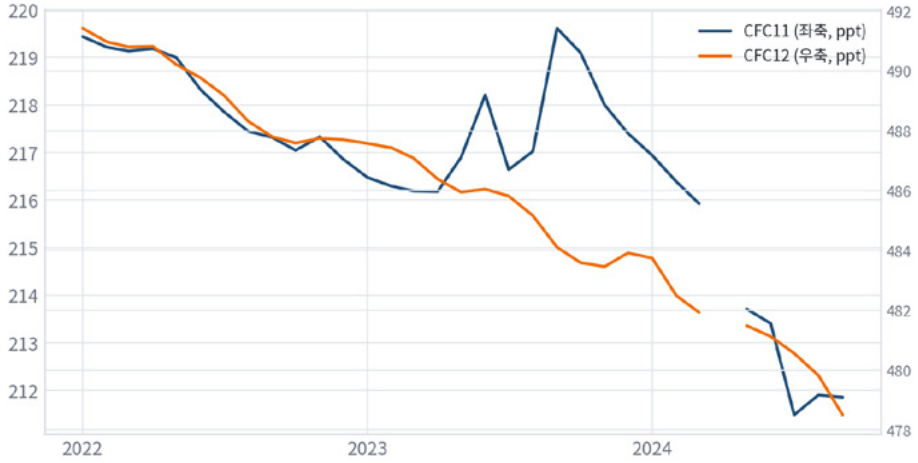


그림 33-4. 안면도의 CFC 농도 감소 추세

❻ 7종 물질의 연도별 평균과 변화율

물질	2022년	2023년	2024년	변화율
CO2 (ppm)	423.89	426.35	429.56	1.30%
CH4 (ppb)	1997.76	2004.65	2014.08	0.80%
N2O (ppb)	337.33	338.38	339.28	0.60%
CFC11 (ppt)	218.7	216.83	214.22	-2.00%
CFC12 (ppt)	490.06	486.39	481.49	-1.70%
CFC113 (ppt)	68.79	68.66	68.03	-1.10%

34.
자외선

시간대별 자외선 변화와 누적량 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 자외선관측

❶ 자외선 관측자료 조회

```
https://apihub.kma.go.kr/api/typ01/url/kma_sfctm_uv.php?tm=202608311200&stn=0&disp=0&auth
Key=...
# YYMMDDHHMI  STN   UVB    UVA    EUV  UV-B UV-A  TEMP1  TEMP2
#      KST   ID
202608311200   13  0.389  2.562  -999.0  5.4  0.5   32.2  32.3
202608311200  108  0.085  0.587  -999.0  1.2  0.1   24.5  24.4
202608311200  159  0.670  3.343  -999.0  9.4  1.5   24.9  24.8
#  UVB 는 10분 동안 쌓인 홍반 유효량(MED) · UV-B 는 그 순간의 지수
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
UV_URL = "https://apihub.kma.go.kr/api/typ01/url/kma_sfctm_uv.php"
STEP_MIN = 10
UVI_LEVEL = [(0,3,"낮음","green"),(3,6,"보통","yellow"),(6,8,"높음","orange"),(8,11,"매우높음","red"),(11,np.inf,"
위험","purple")]
```

DAY1 : 입력 관측일 문자열(YYYY-MM-DD)

DAY2 : 비교 관측일 문자열(YYYY-MM-DD)

필수 입력: DAY1, DAY2. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다. 조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_uv(tm, stn=0):
    """10분 간격의 자외선 관측자료를 조회합니다. stn=0은 전체 관측지점입니다."""

    r = requests.get(
        UV_URL,
        params={"tm": tm, "stn": stn, "disp": 0, "authKey": API_KEY},
        timeout=60,
    )
    if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
    r.encoding = "euc-kr"
    return r.text
def walk_uv(start, end, stn=0, pause=0.05):
    """시각별 API를 10분 간격으로 반복 호출해 지정기간의 자료를 수집합니다. 이틀의 정규 시각은 288개입니다."""

    t, rows = (start, [])
    while t <= end:
        rows.extend(parse_uv(get_uv(t.strftime("%Y%m%d%H%M"), stn)))
        t += timedelta(minutes=STEP_MIN)
        time.sleep(pause)
    return pd.DataFrame(rows)
```

지구대기 감시

```
def parse_uv(text):
    """공백으로 나뉜 고정폭 텍스트다. 한 행이 「한 시각 · 한 지점」이고 UVB·UVA 는 10분 동안 쌓인 양, UV-B·UV-A 는 그것을 지수로 바꾼 값이다."""

    rows = []
    for line in text.splitlines():
        if not line.strip() or line.startswith("#"):
            continue
        p = line.split()
        if len(p) < 9:
            continue
        rows.append(
            {
                "시각": datetime.strptime(p[0], "%Y%m%d%H%M"),
                "지점": int(p[1]),
                "UVB": float(p[2]),
                "UVA": float(p[3]),
                "EUV": float(p[4]),
                "UVB지수": float(p[5]),
                "UVA지수": float(p[6]),
                "내부온도1": float(p[7]),
                "내부온도2": float(p[8]),
            }
        )
    return rows
```

❷ 10분 간격 자외선 일변화 분석

지수가 0을 넘은 첫 시각과 마지막 시각의 차이로 관측상 자외선 검출 시간 범위를 구합니다.

```
def read_stn(path):
    """자외선 지점 목록. 경도·위도·고도가 들어 있어 지도에 바로 얹을 수 있다."""
    out = {}
    for line in Path(path).read_text("utf-8", errors="replace").splitlines():
        if line.startswith("#") or not line.strip():
            continue
        p = line.split()
        if len(p) < 6 or not p[0].isdigit():
            continue
        out[int(p[0])] = {
            "경도": float(p[1]),
            "위도": float(p[2]),
            "고도(m)": float(p[3]),
            "이름": p[5],
        }
    return out
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path, stn_path):
    """수집자료의 결측(-999)을 처리하고 지점정보를 연결합니다. 날짜 집계는 관측시각의 KST 기준을 적용합니다."""

    info = read_stn(stn_path)
    d = pd.DataFrame(parse_uv(Path(path).read_text("utf-8")))
    for c in ("UVB", "UVA", "EUV", "UVB지수", "UVA지수"):
        d.loc[d[c] <= -99, c] = np.nan
    d["이름"] = d["지점"].map(lambda s: info.get(s, {}).get("이름", str(s)))
    d["경도"] = d["지점"].map(lambda s: info.get(s, {}).get("경도", np.nan))
    d["위도"] = d["지점"].map(lambda s: info.get(s, {}).get("위도", np.nan))
    d["고도(m)"] = d["지점"].map(lambda s: info.get(s, {}).get("고도(m)", np.nan))
    d["일"] = d["시각"].dt.strftime("%Y-%m-%d")
    d["분"] = d["시각"].dt.hour * 60 + d["시각"].dt.minute
    return (d.sort_values(["지점", "시각"]).reset_index(drop=True), info)
def daylight(d):
    """자외선지수가 양수인 첫 관측시각과 마지막 관측시각을 구합니다. 천문학적 일출·일몰시각과는 구분합니다."""

    rows = []
    for (dy, s), g in d.groupby(["일", "지점"]):
        h = g[g["UVB지수"] > 0]
        if h.empty:
            continue
        rows.append(
            {
                "일": dy,
                "지점": s,
                "이름": g["이름"].iloc[0],
```

```
            "처음": h["시각"].min().strftime("%H:%M"),
            "마지막": h["시각"].max().strftime("%H:%M"),
            "지속(시간)": round(
              (h["시각"].max() - h["시각"].min()).total_seconds() / 3600, 1
            ),
            "최고지수": round(float(g["UVB지수"].max()), 1),
            "최고시각": g.loc[g["UVB지수"].idxmax(), "시각"].strftime(
              "%H:%M"
            ),
          }
        )
    return pd.DataFrame(
      rows,
      columns=(
        None
        if rows
        else [
          "일",
          "지점",
          "이름",
          "처음",
          "마지막",
          "지속(시간)",
          "최고지수",
          "최고시각",
        ]
      ),
    )
```

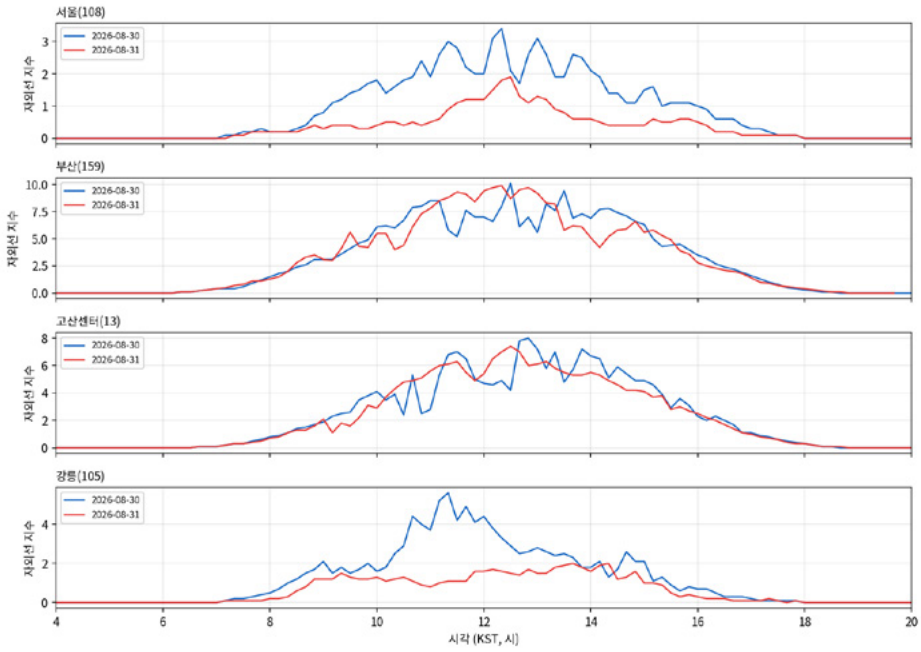


그림 34-1. 10분 간격 자외선 일변화 분석

이름	처음	마지막	지속(시간)	최고지수	최고시각
고산센터	6:40	18:40	12	7.4	12:30
강릉	7:10	17:50	10.7	2	13:40
서울	7:20	17:50	10.5	1.9	12:30
울산	6:10	18:50	12.7	5.5	11:50
광주	7:20	18:40	11.3	6	12:00
부산	6:20	18:40	12.3	9.9	12:20

지구대기 감시

❸ 지점별 하루 최고 자외선지수 비교

지점 목록의 경도·위도로 하루 최고 지수를 지도에 표시합니다.

```
def peak_frame(d):
    """지점마다 하루 최고 지수와 그 시각. 지도 한 장의 재료다."""
    rows = []
    for (dy, s), g in d.groupby(["일", "지점"]):
        if not np.isfinite(g["UVB지수"]).any():
            continue
        i = g["UVB지수"].idxmax()
        rows.append(
            {
                "일": dy,
                "지점": s,
                "이름": g["이름"].iloc[0],
                "경도": g["경도"].iloc[0],
                "위도": g["위도"].iloc[0],
                "고도(m)": g["고도(m)"].iloc[0],
                "최고지수": round(float(g.loc[i, "UVB지수"]), 1),
                "최고시각": g.loc[i, "시각"].strftime("%H:%M"),
                "누적MED": round(float(g["UVB"].sum()), 2),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "일",
                "지점",
                "이름",
                "경도",
                "위도",
                "고도(m)",
                "최고지수",
                "최고시각",
                "누적MED",
            ]
        ),
    )
def lat_fit(pk, dy):
    """위도가 낮을수록 높은지를 실제로 재 본다 — 기울기와 상관계수."""
    g = pk[pk["일"] == dy].dropna(subset=["위도", "최고지수"])
    a, b = np.polyfit(g["위도"], g["최고지수"], 1)
    r = float(np.corrcoef(g["위도"], g["최고지수"])[0, 1])
    return (a, b, r, len(g))
```

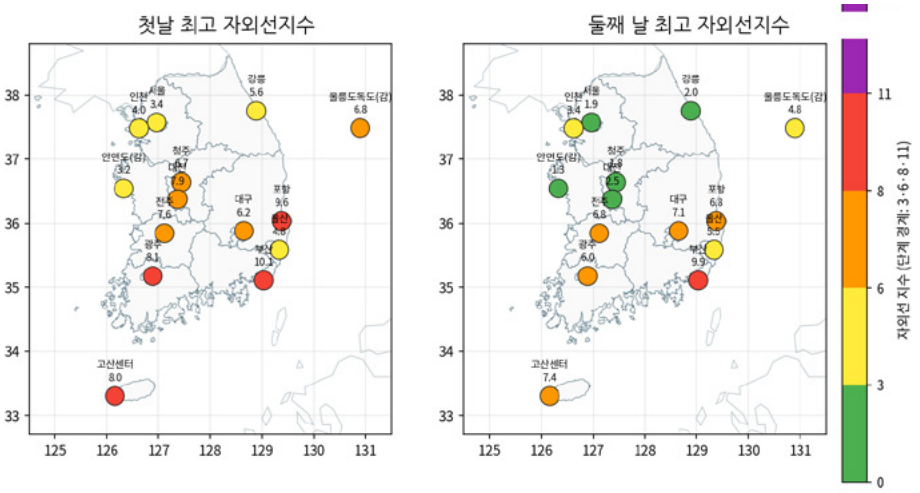


그림 34-2. 지점별 하루 최고 자외선지수 비교

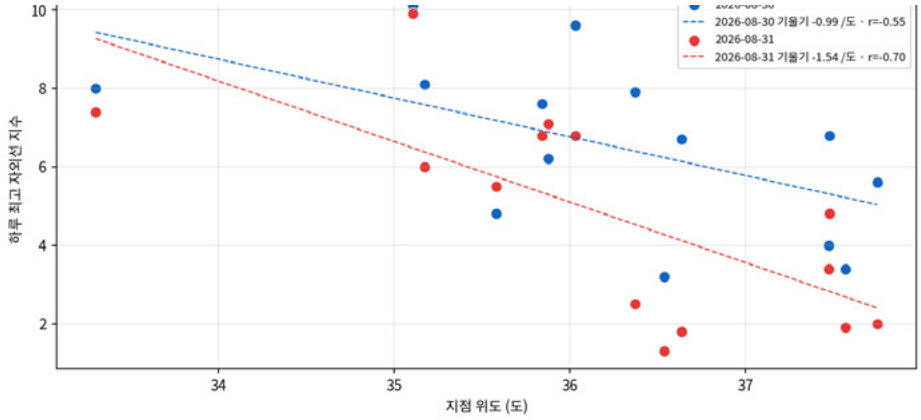


그림 34-3. 지점별 하루 최고 자외선지수 비교

이름	경도	위도	고도(m)	최고지수	최고시각	누적MED
부산	129.032	35.105	69.56	10.1	12:30	22.07
포항	129.38	36.033	2.28	9.6	12:30	16.23
광주	126.892	35.173	72.38	8.1	13:00	13.5
인천	126.625	37.478	68.99	4	12:40	8.15
서울	126.965	37.571	85.5	3.4	12:20	6.25
안면도(감)	126.33	36.54	47	3.2	9:50	6.2

❹ UVB 누적량과 1 MED 도달 시간

```
def cumulative(d, stn, dy):
  """10분 UVB 유효량(MED)을 누적합니다. MED는 최소홍반량이며 개인별 피부 특성에 따라 실제 반응이 다릅니다."""

  g = d[(d["지점"] == stn) & (d["일"] == dy)].sort_values("시각")
  return (g["시각"].values, g["UVB"].cumsum(skipna=False).values)
def med_table(d, levels=(1, 2, 3)):
  """지점·날짜별 누적 UVB가 1·2·3 MED에 도달하는 시각을 구합니다. 개인별 안전 노출시간을 의미하지 않습니다."""
  rows = []
  for (dy, s), g in d.groupby(["일", "지점"]):
    g = g.sort_values("시각")
    cu = g["UVB"].cumsum(skipna=False)
    rec = {
      "일": dy,
      "지점": s,
      "이름": g["이름"].iloc[0],
      "하루누적(MED)": round(float(cu.iloc[-1]), 2),
    }
    for lv in levels:
      hit = g["시각"][cu >= lv]
      rec[f"{lv} MED 도달"] = (
        hit.iloc[0].strftime("%H:%M") if len(hit) else "-"
      )
    start = g["시각"][g["UVB지수"] > 0]
    h1 = g["시각"][cu >= 1]
    rec["해뜬뒤(분)"] = (
      int((h1.iloc[0] - start.iloc[0]).total_seconds() / 60)
      if len(h1) and len(start)
      else np.nan
    )
    rows.append(rec)
  return pd.DataFrame(rows)
```

지구대기 감시

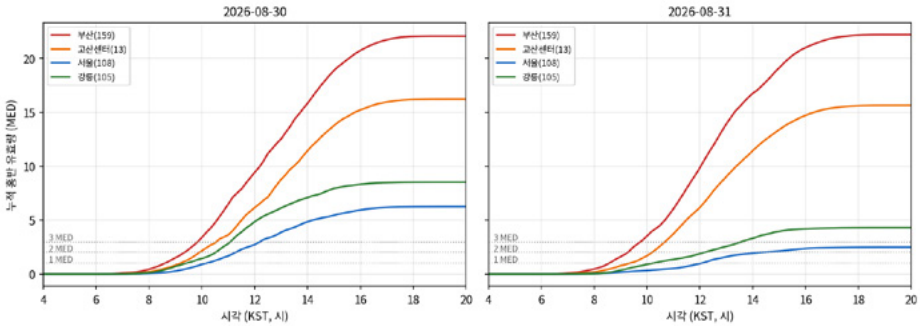


그림 34-4. UVB 누적량과 1 MED 도달 시간

이름	1 MED 도달	2 MED 도달	3 MED 도달	해뜬뒤(분)	하루누적(MED)
고산센터	9:20	10:00	10:40	160	16.22
강릉	9:30	10:40	11:10	140	8.54
서울	10:20	11:20	12:20	190	6.25
울산	9:40	10:20	10:50	210	8.85
광주	9:30	10:30	11:00	170	13.5
부산	8:40	9:30	10:00	140	22.07

❺ 자외선 위험단계와 지속시간 비교

```
def uvi_level(v):
  """지수를 단계 이름으로 바꾼다. 구간은 기상청 발표 기준과 같다."""
  if not np.isfinite(v):
    return None
  for lo, hi, name, _ in UVI_LEVEL:
    if lo <= v < hi:
      return name
  return UVI_LEVEL[-1][2]
def level_grid(d, dy):
  """시간·지점별 자외선 위험단계를 배열로 구성합니다. 지수가 0인 시각은 표시에서 제외합니다."""

  g = d[d["일"] == dy]
  piv = g.pivot_table(index="이름", columns="분", values="UVB지수")
  piv = piv.mask(piv <= 0)
  return piv.loc[piv.max(axis=1).sort_values(ascending=False).index]
def level_hours(d):
  """위험단계별 관측 횟수에 10분을 곱해 환산시간을 계산합니다. 결측 또는 불규칙 관측이 있으면 연속 지속시간과 다를 수 있습
니다."""
  rows = []
  for (dy, s), g in d.groupby(["일", "지점"]):
    v = g["UVB지수"].dropna()
    rec = {"일": dy, "이름": g["이름"].iloc[0]}
    for lo, hi, nm, _ in UVI_LEVEL:
      rec[nm] = round(
        float(((v >= lo) & (v < hi)).sum()) * STEP_MIN / 60, 1
      )
    rec["보통이상"] = round(float((v >= 3).sum()) * STEP_MIN / 60, 1)
    rows.append(rec)
  return pd.DataFrame(rows)
```

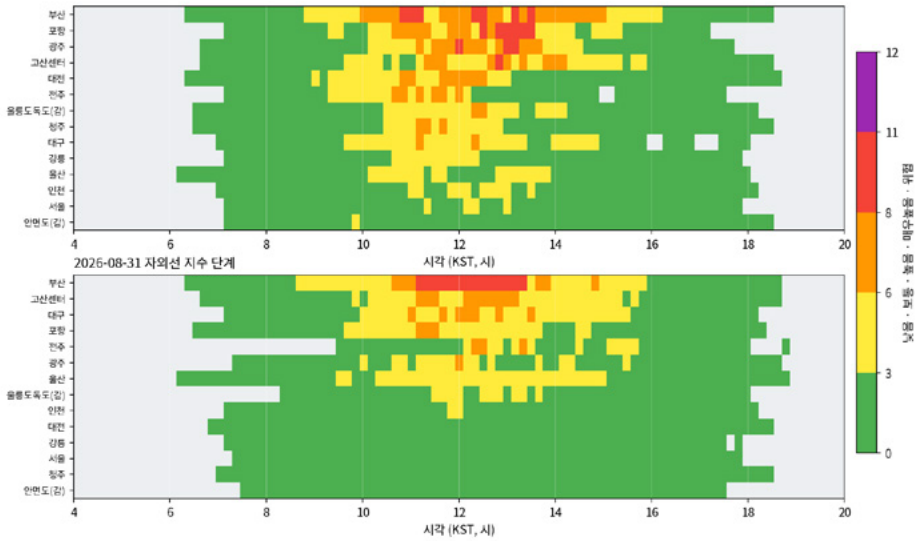


그림 34-5. 자외선 위험단계와 지속시간 비교

일	이름	낮음	보통	높음	매우높음	위험	보통이상
2026-08-30	부산	16.3	2.8	3.5	1.2	0	7.5
2026-08-30	고산센터	18.3	4	1.5	0.2	0	5.7
2026-08-30	포항	18.7	2.8	1.5	1	0	5.3
2026-08-30	인천	21.2	1.7	0	0	0	1.7
2026-08-30	서울	23.3	0.7	0	0	0	0.7
2026-08-30	안면도(감)	23.8	0.2	0	0	0	0.2

❻ 시각별 최대값 대비 자외선 비율 비교

여러 날의 시각별 최대 관측값을 상한선으로 두고 일별 값의 비율과 낮 시간 평균을 계산합니다.

```
def envelope(d):
  """같은 지점·시각의 기간 최댓값을 기준값으로 구성합니다. 기간 최댓값은 청천 조건이나 이론적 상한을 보장하지 않습니다."""

  piv = d.pivot_table(index=["지점", "분"], columns="일", values="UVB")
  piv["상한"] = piv.max(axis=1)
  return piv
def shade_frame(d, frac=0.3, *, DAY1, DAY2):
  """실제 관측값과 해당 기간의 같은 시각 최댓값의 비율을 계산합니다. 구름의 차폐율로 직접 해석하지 않습니다."""

  piv = envelope(d)
  names = d.drop_duplicates("지점").set_index("지점")["이름"]
  rows = []
  for s, g in piv.groupby(level=0):
    g = g[g["상한"] >= frac * g["상한"].max()]
    for dy in (DAY1, DAY2):
      rows.append(
        {
          "지점": s,
          "이름": names.get(s, str(s)),
          "일": dy,
          "가림비율": round(float((g[dy] / g["상한"]).mean()), 3),
          "낮칸수": len(g),
        }
```

지구대기 감시

```
        )
    return pd.DataFrame(
        rows,
        columns=None if rows else ["지점", "이름", "일", "가림비율", "낮칸수"],
    )
def roughness(d, frac=0.3, need=12):
    """일 최고치의 30%를 넘는 구간에서 2차 차분을 최고치로 나누어 시간 변동 지표를 계산합니다."""


    rows = []
    for (dy, s), g in d.groupby(["일", "지점"]):
        g = g.sort_values("시각")
        v = g["UVB"].values
        if not np.isfinite(v).any():
            continue
        mx = np.nanmax(v)
        y = v[np.nan_to_num(v) >= frac * mx]
        if len(y) < need or mx <= 0:
            continue
        rows.append(
            {
                "일": dy,
                "지점": s,
                "이름": g["이름"].iloc[0],
                "최고(MED/10분)": round(float(mx), 3),
                "거칠기(%)": round(
                    float(np.nanmedian(np.abs(np.diff(y, 2))) / mx * 100), 2
                ),
                "누적(MED)": round(float(np.nansum(v)), 2),
                "낮시간(시간)": round(len(y) * STEP_MIN / 60, 1),
            }
        )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "일",
                "지점",
                "이름",
                "최고(MED/10분)",
                "거칠기(%)",
                "누적(MED)",
                "낮시간(시간)",
            ]
        ),
    )
```

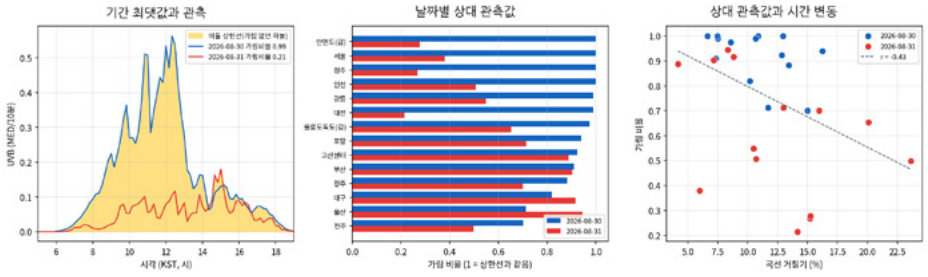


그림 34-6. 시각별 최대값 대비 자외선 비율 비교

이름	일	가림비율	거칠기(%)	최고(MED/10분)
전주	2026-08-30	0.701	15	0.54
울산	2026-08-30	0.712	11.71	0.346
대구	2026-08-30	0.82	10.18	0.442
부산	2026-08-31	0.903	7.17	0.704
대구	2026-08-31	0.916	8.82	0.51
울산	2026-08-31	0.945	8.33	0.396

기후변화 시나리오

35. SSP 시나리오

미래 해수면온도 변화 비교

데이터 활용 경로
기후변화 상황지도 > 적응대책 분석지원 > 데이터 다운로드 (파일 다운로드)

❶ NetCDF 읽기와 격자 확인

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
KST = timezone(timedelta(hours=9))
log = print
import netCDF4
def read_ssp(path, variable="SST"):
  """NetCDF 좌표·달력·단위·결측 속성으로 월별 시나리오 배열을 읽습니다."""
  import netCDF4

  with netCDF4.Dataset(path) as ds:
    if variable not in ds.variables:
      raise ValueError("변수명을 NetCDF 목록에서 확인하세요")
    v = ds.variables[variable]
    if set(v.dimensions) != {"time", "lat", "lon"}:
      raise ValueError("time/lat/lon 3차원 자료가 필요합니다")
    axis = [v.dimensions.index(k) for k in ["time", "lat", "lon"]]
    values = np.ma.filled(np.ma.asarray(v[:], dtype=float), np.nan).transpose(
      axis
    )
    unit = str(getattr(v, "units", ""))
    if unit.strip().lower() in {"k", "kelvin"}:
      values -= 273.15
      unit = "degC"
    timevar = ds.variables["time"]
    times = netCDF4.num2date(
      timevar[:],
      units=timevar.units,
      calendar=getattr(timevar, "calendar", "standard"),
    )
    dates = np.asarray([f"{t.year:04d}-{t.month:02d}" for t in times])
    return (
      values,
      dates,
      np.asarray(ds.variables["lat"][:]),
      np.asarray(ds.variables["lon"][:]),
      unit,
    )
def ssp_monthly(path, variable="SST"):
  a, dates, lat, lon, unit = read_ssp(path, variable)
  # 격자 단순평균 예시. 정확한 면적평균에는 격자 면적 가중치가 필요하다.
  valid = np.isfinite(a).sum(axis=(1, 2))
  mean = np.divide(
    np.nansum(a, axis=(1, 2)),
    valid,
    out=np.full(len(a), np.nan),
    where=valid > 0,
  )
  return pd.DataFrame(
    {"월": dates, "격자단순평균": mean, "유효격자수": valid, "단위": unit}
  )
```

실행 예: t = ssp_monthly(NC_PATH, "SST"). 변수명·차원명은 입력 파일에서 확인합니다. 자료의 달력과 결측 속성을 적용하며 켈빈 단위일 때만 섭씨로 변환합니다. 결과는 기후변화 시나리오의 격자 단순평균으로, 단기 예보나 면적 가중 평균이 아닙니다.

```
NC_PATH = Path(input('다운로드한 SSP NetCDF 경로: '))
ds = netCDF4.Dataset(NC_PATH)
# 기존 문서의 파일 예시 구조
VARS: SST(time, lat, lon), time, lat, lon
DIMS: time=240, lat=170, lon=140  (2021~2040 파일, 2041~2060 파일과 동일 격자)
lat 29.85~41.17 · lon 121.63~133.16 · 해상 격자 17,557개 / 전체 23,800개
missing_value = -9990.0  (육상·자료없음 지점)
```

시나리오	요소	공간해상도	기간	자료형식	해상 격자 수
SSP1-2.6	해수면온도(SST)	한반도(8km)	2021-01-31 ~ 2060-12-31	NetCDF(월자료)	17,557개

❷ 월별 해수면온도와 12개월 이동평균

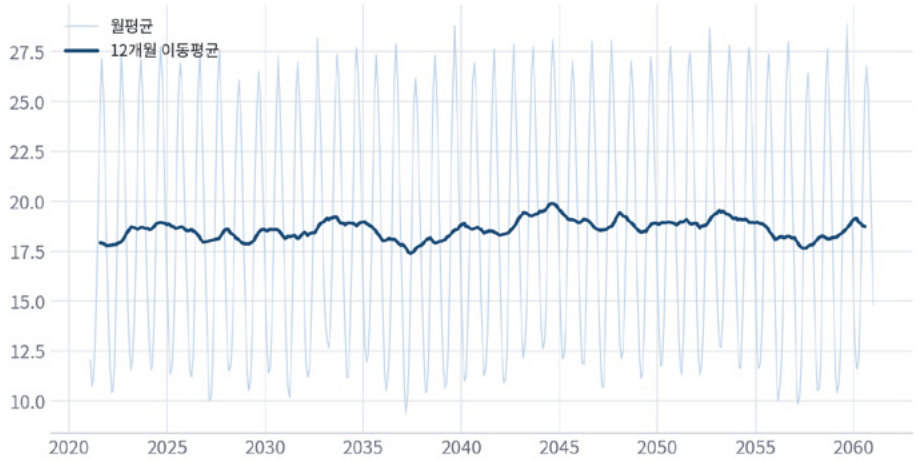


그림 35-1. 월별 해수면온도와 12개월 이동평균

❸ 10년 평균 해수면온도 비교

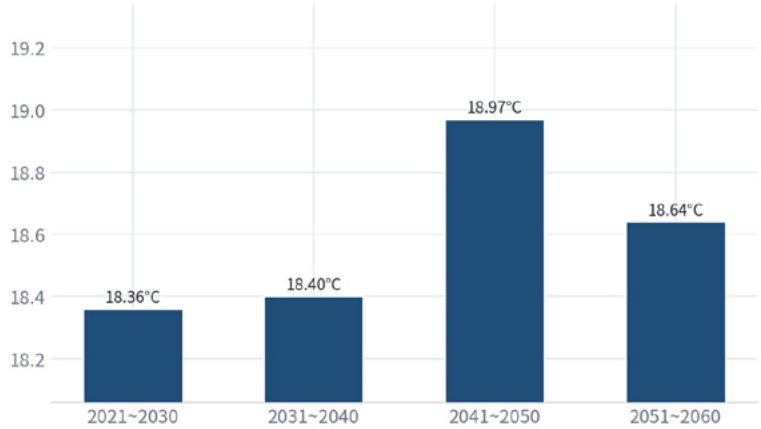


그림 35-2. 10년 평균 해수면온도 비교

기후변화 시나리오

❹ 격자별 해수면온도 상승폭 비교

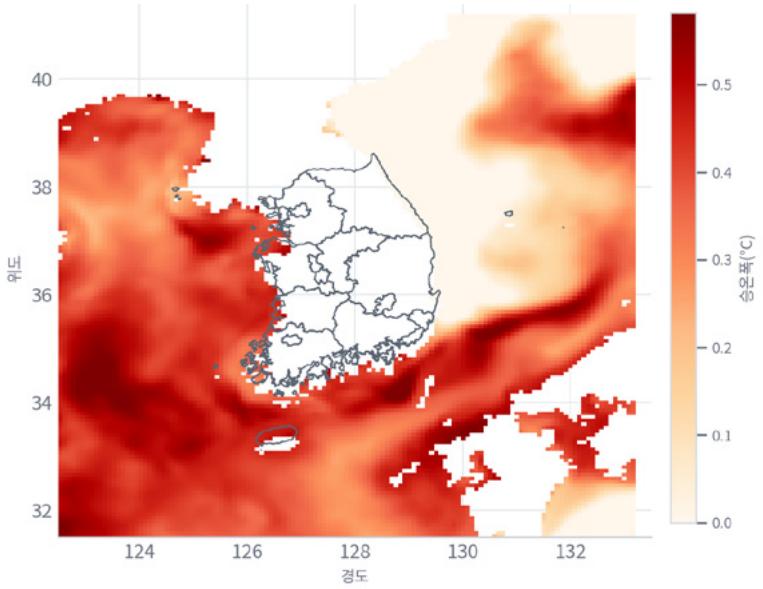


그림 35-3. 격자별 해수면온도 상승폭 비교

❺ 격자별 조회와 10년 평균 활용

기간	평균 SST	전기간(2021~2030) 대비
2021~2030	18.36°C	기준
2031~2040	18.40°C	+0.04°C
2041~2050	18.97°C	+0.61°C
2051~2060	18.64°C	+0.28°C

36.
표준강수지수

지역별 가뭄 정도 비교

데이터 활용 경로(API)

기상자료개방포털 > 기후 > 가뭄 > 표준강수지수 (파일 다운로드)

❶ CSV 읽기와 지수 기준

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
KST = timezone(timedelta(hours=9))
log = print
def read_spi(path, encoding="cp949"):
  d = pd.read_csv(path, encoding=encoding)
  if "일시" not in d:
    raise ValueError("일시 열이 있는 SPI CSV를 사용하세요")
  d["일시"] = pd.to_datetime(d["일시"], errors="coerce")
  for column in [c for c in d if re.fullmatch(r"SPI\d+", c)]:
    d[column] = pd.to_numeric(d[column], errors="coerce")
    d.loc[d[column] <= -99, column] = np.nan
  return d.dropna(subset=["일시"])
def spi_monitor(d, column="SPI6", date=None):
  """누적기간별 SPI를 섞지 않고 해당 날짜의 낮은 지수부터 표시."""
  if column not in d:
    raise ValueError("요청한 SPI 누적기간 열이 없습니다")
  when = pd.Timestamp(date) if date is not None else d["일시"].max()
  x = d[d["일시"].eq(when)].copy()
  x["-1 이하"] = x[column].le(-1) & x[column].notna()
  return x.sort_values(column, na_position="last")
```

실행 예: d = read_spi(CSV_PATH); t = spi_monitor(d, "SPI6"). SPI는 음수도 유효값입니다. 기본 조회일은 입력 파일의 마지막 날짜이며, 서로 다른 누적기간의 값을 하나의 지수처럼 합치지 않습니다.

```
CSV_PATH = Path(input('다운로드한 SPI CSV 경로: '))
df = pd.read_csv(CSV_PATH, encoding='cp949')
# 기존 문서의 파일 예시 내용 (일부)
지점,지점명,일시,SPI1,SPI2,SPI3,SPI4,SPI5,SPI6,SPI9,SPI12,SPI18,SPI24
159,부산,2026-09-13,.98,-.13,-.80,-.85,-1.29,-1.06,-1.19,-1.36,-1.41,-.42
143,대구,2026-09-13,-.18,-.72,-1.78,-1.60,-1.78,-1.60,-1.69,-1.67,-1.02,-.78
```

가뭄정보

SPI 값	단계	2026-09-13 기준 해당 지점 수
1.00 이상	습함	0
0.99 ~ -0.99	정상	43
-1.00 ~ -1.49	약한가뭄(관심)	18
-1.50 ~ -1.99	보통가뭄(주의)	4
-2.00 이하	심한가뭄(경계)	1

❷ SPI6 기준 전국 가뭄 단계 분포

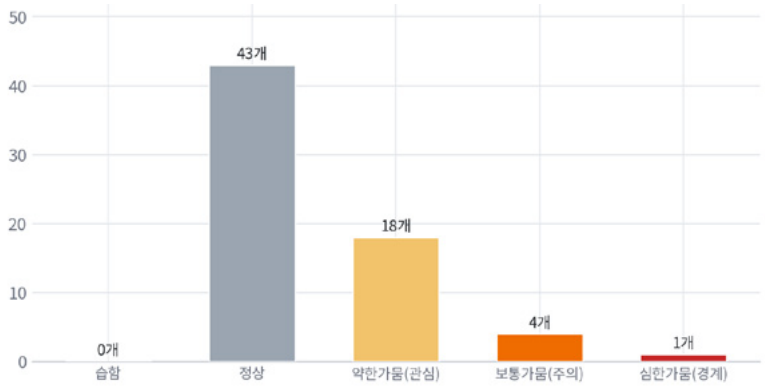


그림 36-1. SPI6 기준 전국 가뭄 단계 분포

❸ SPI6가 가장 낮은 10개 지점

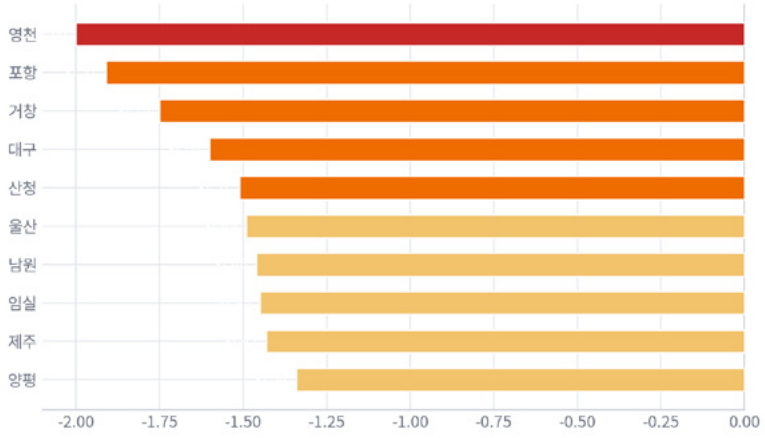


그림 36-2. SPI6가 가장 낮은 10개 지점

❹ 영천의 누적기간별 SPI 비교

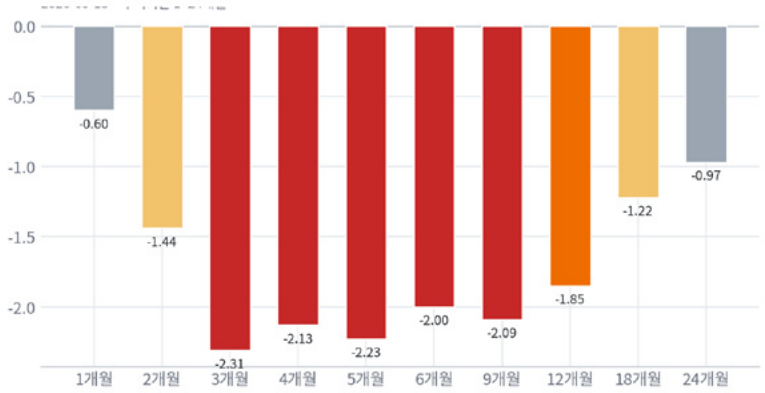


그림 36-3. 영천의 누적기간별 SPI 비교

❺ 지점별 SPI 임계값 모니터링

지점	SPI3	SPI6	SPI12	SPI24
영천	-2.31	-2	-1.85	-0.97
포항	-2.1	-1.91	-1.62	-1.16
거창	-2.1	-1.75	-1.51	-0.72
대구	-1.78	-1.6	-1.67	-0.78
산청	-1.81	-1.51	-1.59	-0.61
울산	-1.44	-1.49	-1.61	-0.66
남원	-1.59	-1.46	-1.44	0.13
임실	-1.28	-1.45	-1.37	0.28
제주	-1.95	-1.43	-1.15	-1.2
양평	-1.07	-1.34	-0.6	-0.42

기상기후데이터 카탈로그

데이터 활용사례 50

응용기상

Applied Meteorology



기상지수

37. 자외선지수

지역별 자외선 위험단계 비교

데이터 활용 경로(API)

기상청 API허브 > 융합기상 > 생활안전

❶ 자외선지수 예보 조회

자외선지수 예보 조회

```
https://apihub.kma.go.kr/api/typ02/openApi/LivingWthrIdxServiceV3/getUVIdxV3?areaNo=1111000000&time=2026090206&dataType=JSON&authKey=...
{"code":"A07_2","areaNo":"1111000000","date":"2026090206",
 "h0":"1","h3":"6","h6":"5","h9":"3","h12":"0", ···,
 "h72":"","h75":""}
#  h 뒤의 숫자가 발표시각으로부터 몇 시간 뒤인지 (3시간 간격 26칸)
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
UV_URL = "https://apihub.kma.go.kr/api/typ02/openApi/LivingWthrIdxServiceV3/getUVIdxV3"
def attach_locations(d, locations):
    """지역코드별 표시명과 대표좌표를 연결. 좌표는 API 값으로 추정하지 않는다."""
    required = {"지역코드", "지점명", "경도", "위도"}
    if required - set(locations):
        raise ValueError("지역코드·지점명·경도·위도 열 필요")
    m = locations[list(required)].copy()
    m["지역코드"] = m["지역코드"].astype(str)
    if m["지역코드"].duplicated().any():
        raise ValueError("지역코드별 좌표는 하나씩 지정")
    return d.drop(columns=["지점명", "경도", "위도"], errors="ignore").merge(
        m, on="지역코드", how="left", validate="many_to_one"
    )
```

실행 순서: d = parse_uv(get_uv(input("공식 지역코드: "), input("조회 가능한 발표시각 (KST, YYYYMMDDHH): "))); d.to_csv("uv.csv", index=False, encoding="utf-8-sig"); d = read_pack("uv.csv"); risk_table(d). 지역별 지수 집계는 좌표 없이 가능하며 지도·위도 분석에는 attach_locations(d, locations)로 공식 지역코드와 대표좌표 표를 연결합니다. 지점 열은 지역코드입니다.

```
def save_uv_csv(area_no, tm, path="uv_forecast.csv"):
    d = parse_uv(get_uv(area_no, tm))
    d.to_csv(path, index=False, encoding="utf-8-sig")
    return d
```

```
def get_uv(area_no, tm, retry=3):
  """자외선지수 예보. 제공된 h항목의 범위를 응답에서 확인한다.
  h 뒤의 숫자가 발표시각으로부터 몇 시간 뒤인지를 뜻한다."""
  params = {
    "areaNo": area_no,
    "time": tm,
    "dataType": "JSON",
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(UV_URL, params=params, timeout=60)
      r.raise_for_status()
      return r.json()
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{area_no} 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(2 * (i + 1))
def parse_uv(js):
  response = js.get("response", {})
  code = str(response.get("header", {}).get("resultCode", ""))
  if code not in ("00", "03"):
    raise RuntimeError(f"resultCode {code}")
  items = (response.get("body", {}).get("items") or {}).get("item", [])
  if isinstance(items, dict):
    items = [items]
  rows = []
  for item in items:
    base = pd.Timestamp(
      datetime.strptime(item["date"], "%Y%m%d%H"), tz="Asia/Seoul"
    )
    for key in sorted(
      (k for k in item if re.fullmatch("h\\d+", k)),
      key=lambda k: int(k[1:]),
    ):
      hours = int(key[1:])
      value = pd.to_numeric(item[key], errors="coerce")
      rows.append(
        {
          "지역코드": item["areaNo"],
          "발표시각": base,
          "예보시각": base + timedelta(hours=hours),
          "선행(h)": hours,
          "자외선지수": value,
        }
      )
  d = pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else ["지역코드", "발표시각", "예보시각", "선행(h)", "자외선지수"]
    ),
  )
  d["지역코드"] = d["지역코드"].astype(str)
  d["지점"] = d["지역코드"]
  d["지점명"] = d["지역코드"]
  d["경도"], d["위도"] = np.nan, np.nan
  return d


def area_of(stn_line, region_codes):
  """공식 지역코드표에 있는 코드만 사용. 법정동코드를 임의 절단하지 않는다."""
  return next(
    (token for token in stn_line.split() if token in set(region_codes)), None
  )
```

❷ 사흘간 자외선지수 예보 변화

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_pack(path):
  """parse_uv 출력 CSV를 읽습니다. 발표시각과 지역코드를 보존한다."""
```

기상지수

```
    d = pd.read_csv(path, encoding="utf-8-sig", dtype={"지역코드": str})
    for c in ["예보시각", "발표시각"]:
        d[c] = pd.to_datetime(d[c], utc=True).dt.tz_convert("Asia/Seoul")
    return d
def peak_hours(d):
    """지점·날짜별 최고 자외선지수의 예보 시간대를 집계합니다."""

    if d.empty:
        return pd.DataFrame(columns=["시각", "횟수"])
    g = d.dropna(subset=["자외선지수"]).copy()
    g["날짜"] = g["예보시각"].dt.date
    idx = g.groupby(["지점", "날짜"])["자외선지수"].idxmax()
    top = g.loc[idx]
    return (
        top["예보시각"]
        .dt.hour.value_counts()
        .sort_index()
        .rename_axis("시각")
        .reset_index(name="횟수")
    )
```

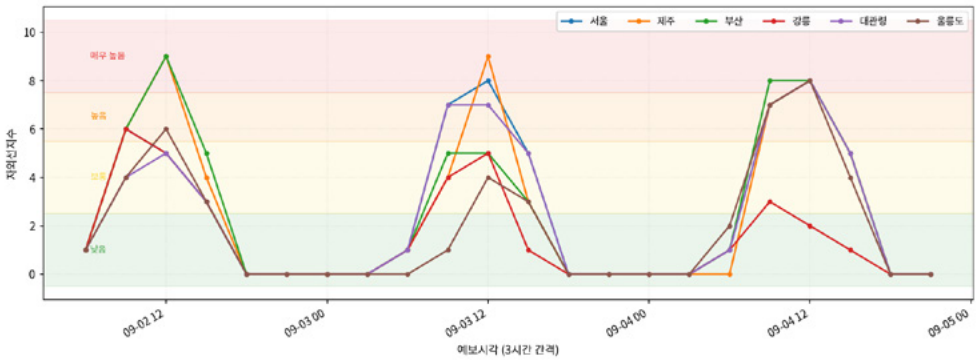


그림 37-1. 사흘간 자외선지수 예보 변화

시각	횟수
9	42
12	249

❸ 날짜별 낮 12시 자외선지수 분포

지역별 좌표에 같은 예보시각의 값을 표시합니다. 사흘간 낮 12시 지도를 비교하면 높은 지수가 예보된 지역의 변화를 볼 수 있습니다.

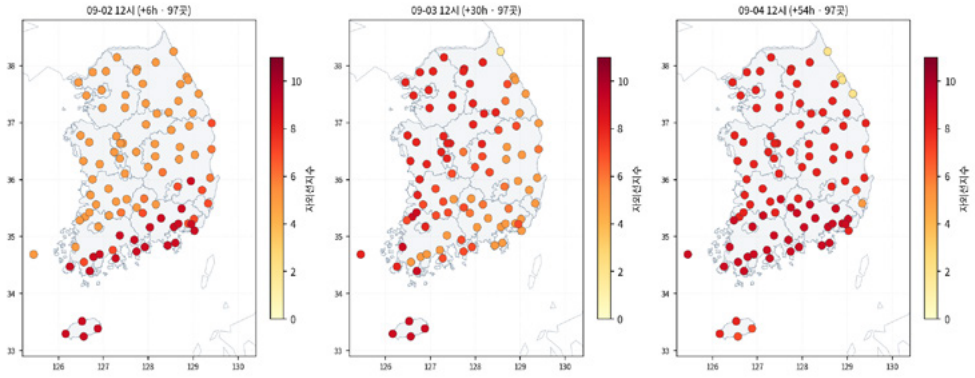


그림 37-2. 날짜별 낮 12시 자외선지수 분포

❹ 자외선지수 6 이상 예보 횟수와 환산지표

지수 6 이상 예보값의 개수와 3시간구간 환산지표를 비교합니다. 환산값은 실제 연속 위험시간이 아닙니다.

```
def risk_table(d, thr_hi=6, thr_vhi=8):
    """지역별로 자외선지수 높음 이상인 예보 횟수를 집계합니다. 3시간 환산값은 실제 연속 위험시간과 구분합니다."""

    g = d.dropna(subset=["자외선지수"]).groupby(
        ["지점", "지점명", "경도", "위도"], dropna=False
    )
    t = g["자외선지수"].agg(최대="max", 평균="mean").reset_index()
    t["평균"] = t["평균"].round(2)
    t["높음이상(칸)"] = (
        g["자외선지수"].apply(lambda s: int((s >= thr_hi).sum())).values
    )
    t["매우높음(칸)"] = (
        g["자외선지수"].apply(lambda s: int((s >= thr_vhi).sum())).values
    )
    t["3시간구간 환산시간(h)"] = t["높음이상(칸)"] * 3
    return t.sort_values("최대", ascending=False)
```

지점명	최대	평균	높음이상(칸)	매우높음(칸)	3시간구간 환산시간(h)
창원	9	2.5	4	2	12
부산	9	2.59	4	3	12
광주	9	2.41	3	2	9
영천	9	2.41	3	2	9
거제	9	2.5	4	2	12
북부산	9	2.77	5	3	15

기상지수

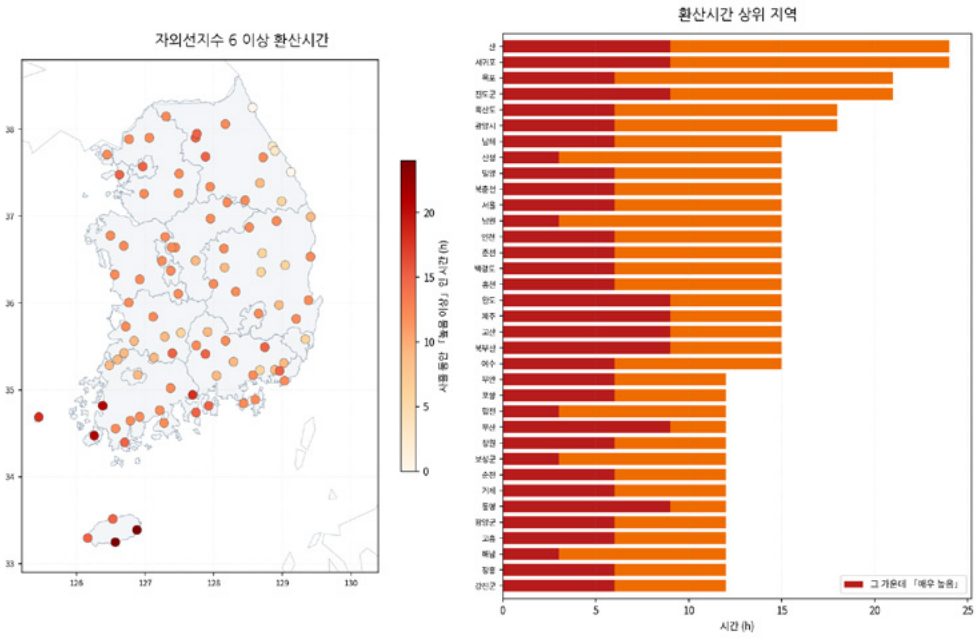


그림 37-3. 자외선지수 6 이상 예보 횟수와 환산지표

❺ 시간대별 지역 편차와 위도 관계

```
def slot_table(d):
    """시각(칸)마다 전국 분포를 요약한다. 하루 안의 모양이 한 표로 나반환됩니다."""
    if d.empty:
        return pd.DataFrame(columns=["시각", "지점수", "평균", "최대", "최소"])
    g = d.dropna(subset=["자외선지수"]).groupby("예보시각")["자외선지수"]
    t = g.agg(지점수="size", 평균="mean", 최대="max", 최소="min").reset_index()
    t["평균"] = t["평균"].round(2)
    t["시각"] = t["예보시각"].dt.strftime("%m-%d %H시")
    return t[["시각", "지점수", "평균", "최대", "최소"]]
def lat_fit(t):
    """좌표와 최대지수가 유효한 서로 다른 위도의 지역만 회귀한다."""
    g = (
        t[["위도", "최대"]]
        .apply(pd.to_numeric, errors="coerce")
        .replace([np.inf, -np.inf], np.nan)
        .dropna()
    )
    if len(g) < 2 or g["위도"].nunique() < 2:
        return {
            "지점 수": len(g),
            "읽는 법": "위도 회귀에는 좌표가 있는 서로 다른 위도의 두 지역 이상이 필요합니다.",
        }
    a, b = np.polyfit(g["위도"], g["최대"], 1)
    r = g["위도"].corr(g["최대"])
    return {
        "지점 수": len(g),
        "위도 1도당": f"{a:+.2f}",
        "상관 r": f"{r:.2f}",
        "최대 지수 범위": f"{g['최대'].min():.0f} ~ {g['최대'].max():.0f}",
        "읽는 법": "조회기간·지역에 한정한 회귀이며 인과관계나 모든 계절의 경향을 뜻하지 않습니다.",
    }
```

시각	지점수	평균	최대	최소
09-02 06시	97	1	1	1
09-02 09시	97	4.76	6	4
09-02 12시	97	6.2	9	5
09-04 15시	97	4.82	6	1
09-04 18시	97	0	0	0
09-04 21시	97	0	0	0

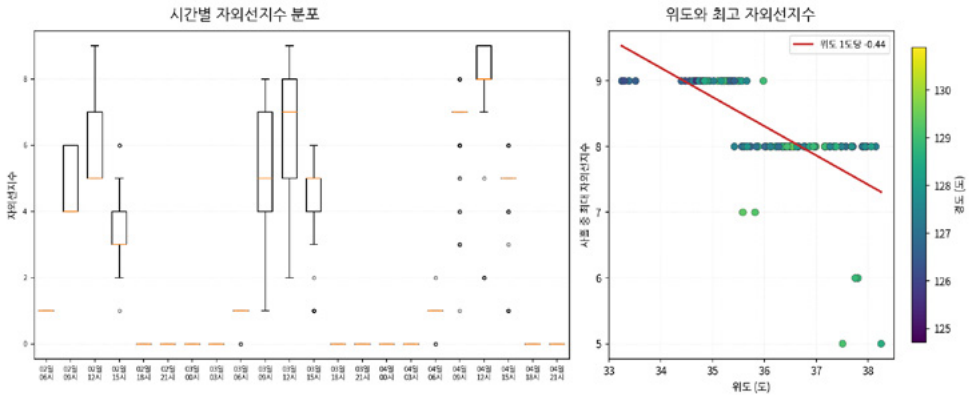


그림 37-4. 시간대별 지역 편차와 위도 관계

기상지수

38.
꽃가루농도위험지수

지역·종류별 위험단계 비교

데이터 활용 경로(API)

기상청 API허브 > 융합기상 > 생활안전

❶ 꽃가루 위험지수 조회

※ 기상자료개방포털 안내 기준 제공기간은 참나무·소나무 3~6월, 잡초류 8~10월입니다. 일부 기존 API 오류메시지에는 3~6월로 표시될 수 있으므로 이용 시 최신 안내를 우선 적용합니다.

행정구역별 제공 지점을 행정구역코드로 지정합니다. 위험단계는 0(낮음)·1(보통)·2(높음)·3(매우 높음)이며, 요소별 제공기간은 다릅니다.

```
# 1) 잡초류 (제공기간 8~10월) → 응답 예시 (첫 지점)
https://apihub.kma.go.kr/api/typ02/openApi/HealthWthrIdxServiceV2/getWeedsPollenRiskndxV2?numOfRows=4000&pageNo=1&dataType=JSON&areaNo=&time=2026091506&authKey=...
{"response":{"header":{"resultCode":"00","resultMsg":"NORMAL_SERVICE"},"body":{"items":{"item":[
{"code":"D08","areaNo":"1100000000","date":"2026091506","today":"2","tomorrow":"2","dayaftertomorrow":"2"}
]},"totalCount":3851}}}
# 2) 참나무·소나무 (제공기간 4~6월) 같은 시각 호출 → 제공기간 안내 응답 예시
https://apihub.kma.go.kr/api/typ02/openApi/HealthWthrIdxServiceV2/getOakPollenRiskndxV2?numOfRows=4000&pageNo=1&dataType=JSON&areaNo=&time=2026091506&authKey=...
#  (소나무는 getPinePollenRiskndxV2, 파라미터 동일)
{"response":{"header":{"resultCode":"99","resultMsg":"해당지수자료 제공기간이 아닙니다! [자료제공기간 3월 ~ 6월]"}}}
#  → 오류가 아니라 제공기간 밖 요청에 대한 정상 안내. 요청 시각을 4~6월로 지정하면 resultCode 00
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
BASE = "https://apihub.kma.go.kr/api/typ02/openApi/HealthWthrIdxServiceV2"
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  r.raise_for_status()
  response = r.json().get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code in ("03", "99"):  # 03 자료 없음, 99 제공기간 외 → 빈 결과
   return ([], 0)
  if code != "00":
   raise RuntimeError(head.get("resultMsg", code))
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):  # 1건이면 dict로 옴
   items = [items]
  return (items, int(body.get("totalCount", len(items))))

def fetch_sido_summary(service, time_, auth_key):
  all_items, page = ([], 1)
  while True:
    items, total = request_items(
      f"{BASE}/{service}",
      {
        "numOfRows": 1000,
        "pageNo": page,
        "dataType": "JSON",
```

```
            "areaNo": "",
            "time": time_,
            "authKey": auth_key,
        },
    )
    if not items:
      break
          raise ValueError("꽃가루 페이지 중간 누락")
        return pd.DataFrame()
    all_items.extend(items)
    if len(all_items) >= total:
        break
    page += 1
    if page > 100:  # 안전장치: 최대 100쪽
        break
    time.sleep(0.3)
if not all_items: return pd.DataFrame()
df = pd.DataFrame(all_items)
cols = ["today", "tomorrow", "dayaftertomorrow"]
if "todayaftertomorrow" in df:
    df = df.rename(columns={"todayaftertomorrow": "dayaftertomorrow"})
for col in cols:
    df[col] = pd.to_numeric(df[col], errors="coerce")
    df[col] = df[col].where(df[col].isin([0, 1, 2, 3]))
df["sido_code"] = df["areaNo"].astype(str).str[:2]
return df.groupby("sido_code")[cols].mean()
```

요소	자료제공기간 (공식 안내)	응답 코드	2026-09-15 조회	전국 지점 수
참나무	4월 ~ 6월	resultCode 99	기간 아님	-
소나무	4월 ~ 6월	resultCode 99	기간 아님	-
잡초류	8월 ~ 10월	resultCode 00	정상	3,851

❷ 꽃가루 종류별 자료제공기간 확인

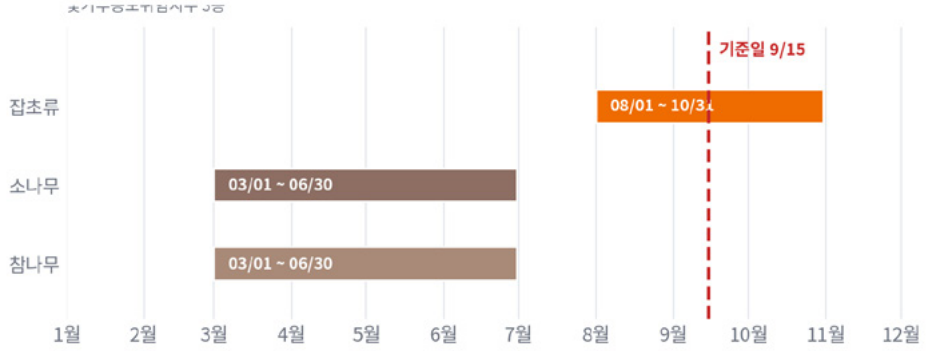


그림 38-1. 꽃가루 종류별 자료제공기간 확인

기상지수

❸ 2026년 9월 15일 지점별 잡초류 위험지수

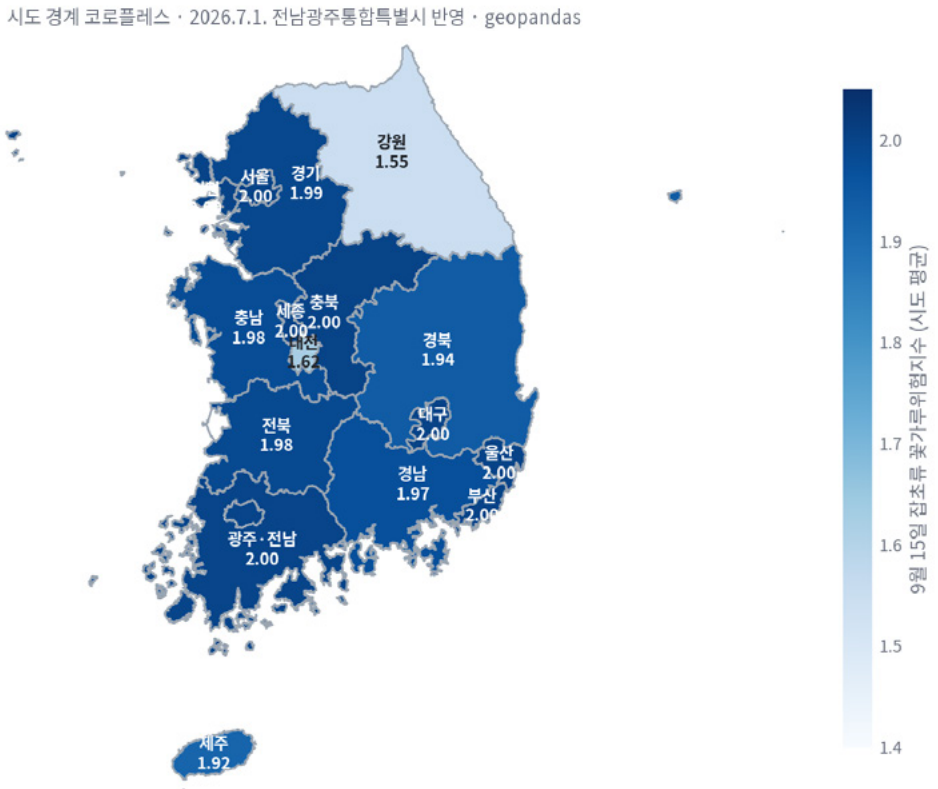


그림 38-2. 2026년 9월 15일 지점별 잡초류 위험지수

❹ 시도별 평균 위험지수 차이

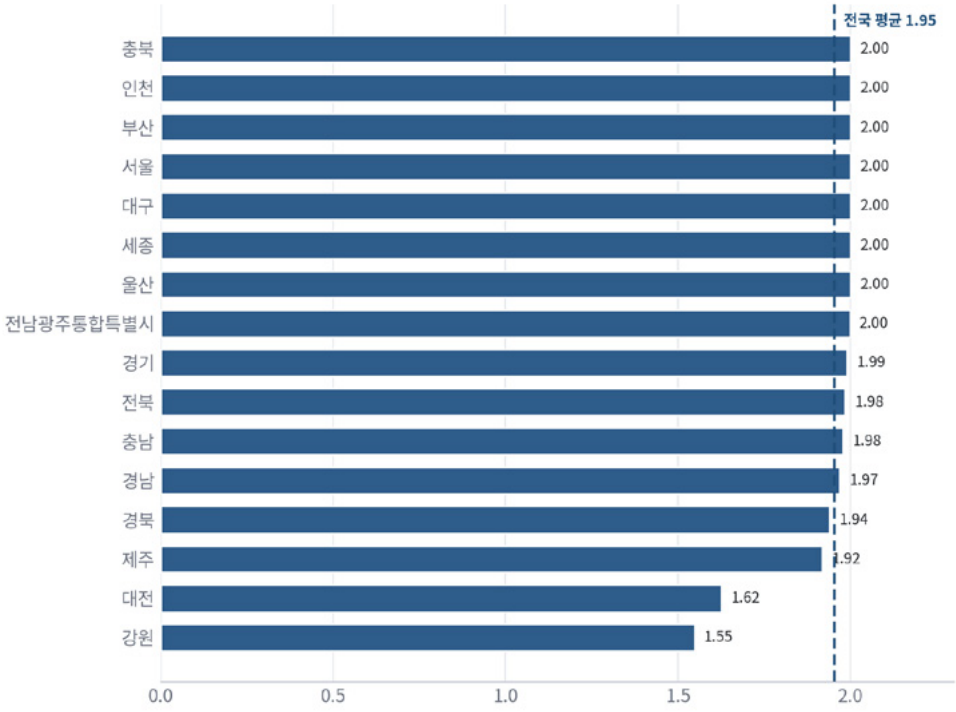


그림 38-3. 시도별 평균 위험지수 차이

16개 시도의 지점 수와 사흘간 예보값은 다음과 같습니다.

시도	지점 수	9/15	9/16	9/17
울산	61	2.00	1.66	1.54
세종	26	2.00	2.00	2.00
대구	160	2.00	2.00	1.96
제주	48	1.92	1.92	1.92
대전	88	1.62	1.58	1.08
강원	212	1.55	1.58	1.51

❺ 위험단계별 지점 수 비교

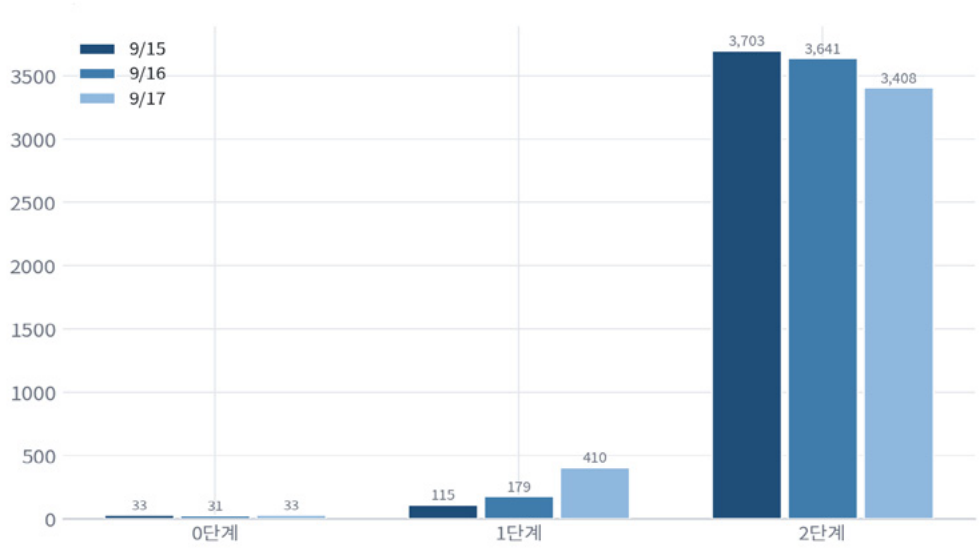


그림 38-4. 위험단계별 지점 수 비교

❻ 9월 15~17일 시도별 예보 변화

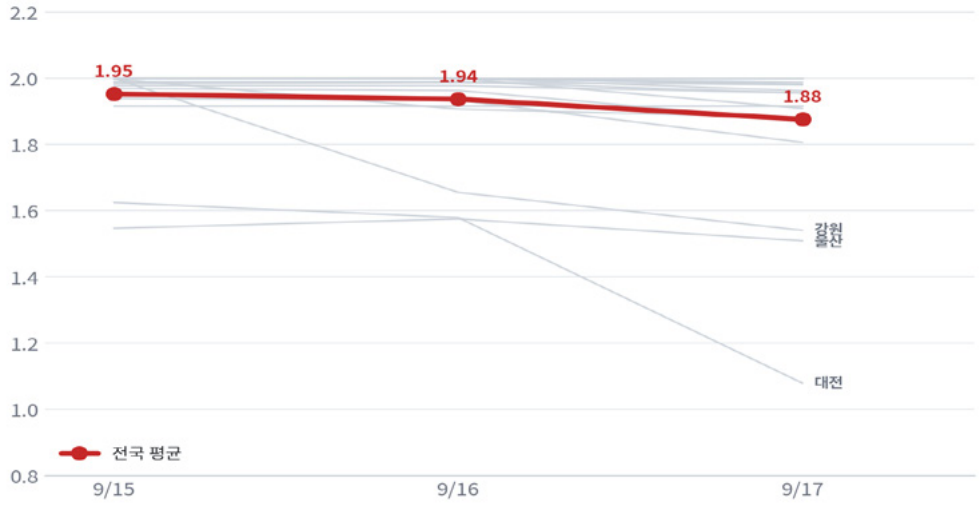


그림 38-5. 9월 15~17일 시도별 예보 변화

기상지수

❼ 강원과 최고 평균 7개 시도 비교

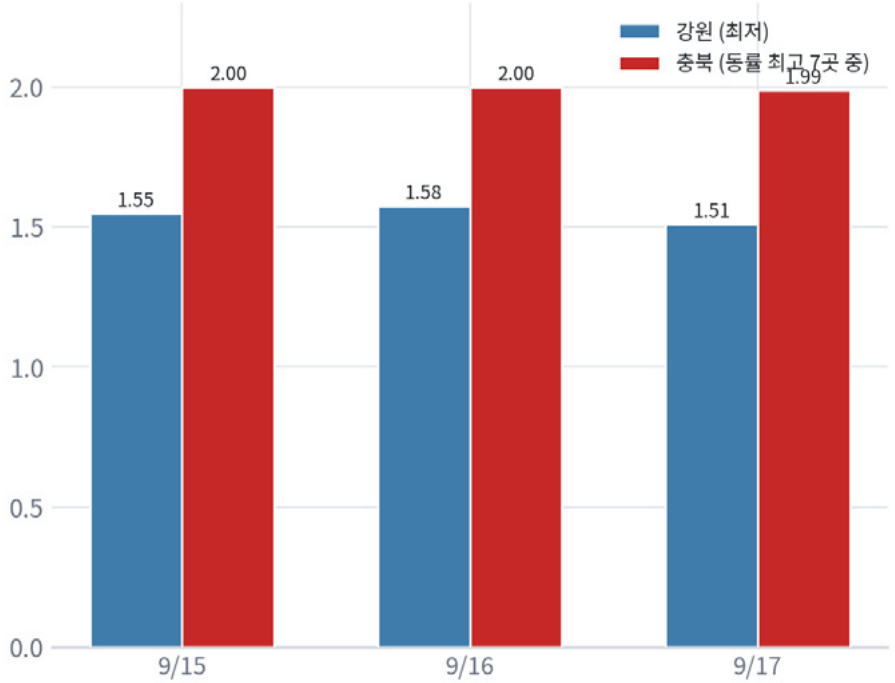


그림 38-6. 강원과 최고 평균 7개 시도 비교

39.
기상청 도로날씨정보

노선별 안개·비·눈 분포 분석

데이터 활용 경로(API)

기상청 API허브 > 융합기상 > 교통안전

❶ CCTV 판독자료 읽기

응답의 위경도와 5분 간격 자료로 지도와 지점별 시계열을 작성합니다.

```
https://apihub.kma.go.kr/api/typ01/url/cctv_ana_info.php?tm1=202608311200&tm2=202608311200
&eqp=0&disp=1&authKey=...
# YYMMDDHHMI EQP_ID STN_NM LAT LON HT WH_CD FOG_CD RN_CD SN_CD
202608311200 0150T00004 대산 35.31872 126.56783 -99.0 21 00 21 00
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
WH = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

범주와 상태는 공식 코드표로 확인합니다. 코드표에 없는 값은 원래 코드로 표시합니다.

```
def parse_road(text):
    """한 행에 수집시각·CCTV ID·지점명·위경도·고도와 네 개의 코드가 들어 있다. 지점명에 공백이 섞일 수 있어 뒤에서부터 잘라 산출합니다."""

    rows = []
    for line in text.splitlines():
        s = line.strip().rstrip("=").strip()
        if not s or s.startswith("#"):
            continue
        p = s.split()
        if len(p) < 10:
            continue
        sn, rn, fog, wh = (p.pop(), p.pop(), p.pop(), p.pop())
        ht, lon, lat = (p.pop(), p.pop(), p.pop())
        tm, eqp = (p.pop(0), p.pop(0))
        rows.append(
            {
                "시각": tm,
                "EQP_ID": eqp,
                "지점명": " ".join(p),
                "위도": float(lat),
                "경도": float(lon),
                "WH_CD": wh,
                "FOG_CD": fog,
                "RN_CD": rn,
```

기상지수

```
            "SN_CD": sn,
        }
    )
  return pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "시각",
        "EQP_ID",
        "지점명",
        "위도",
        "경도",
        "WH_CD",
        "FOG_CD",
        "RN_CD",
        "SN_CD",
      ]
    ),
  )
```

코드	뜻	코드	뜻
00	맑음	11	안개 약
12	안개 중	13	안개 강
21	비 약	22	비 중
23	비 강	31	눈 약
32	눈 중	33	눈 강

❷ 노선별 CCTV 관측지점 지도 작성

노선명에 따라 색을 나누어 도로별 분포를 비교합니다.

```
def net_table(obs, info):
  d = obs.merge(info, on="EQP_ID", how="left")
  g = d.groupby("노선").agg(
    지점=("EQP_ID", "count"), 남단=("위도", "min"), 북단=("위도", "max")
  )
  return (d, g.sort_values("지점", ascending=False).round(3).reset_index())
```

노선	지점	남단	북단
서해안선	163	34.844	37.424
중앙선	108	35.204	37.843
중부내륙선	80	35.268	37.522
호남선	77	35.005	36.075
영동선	62	37.246	37.769
서울도시-내부순환선	1	37.724	37.724

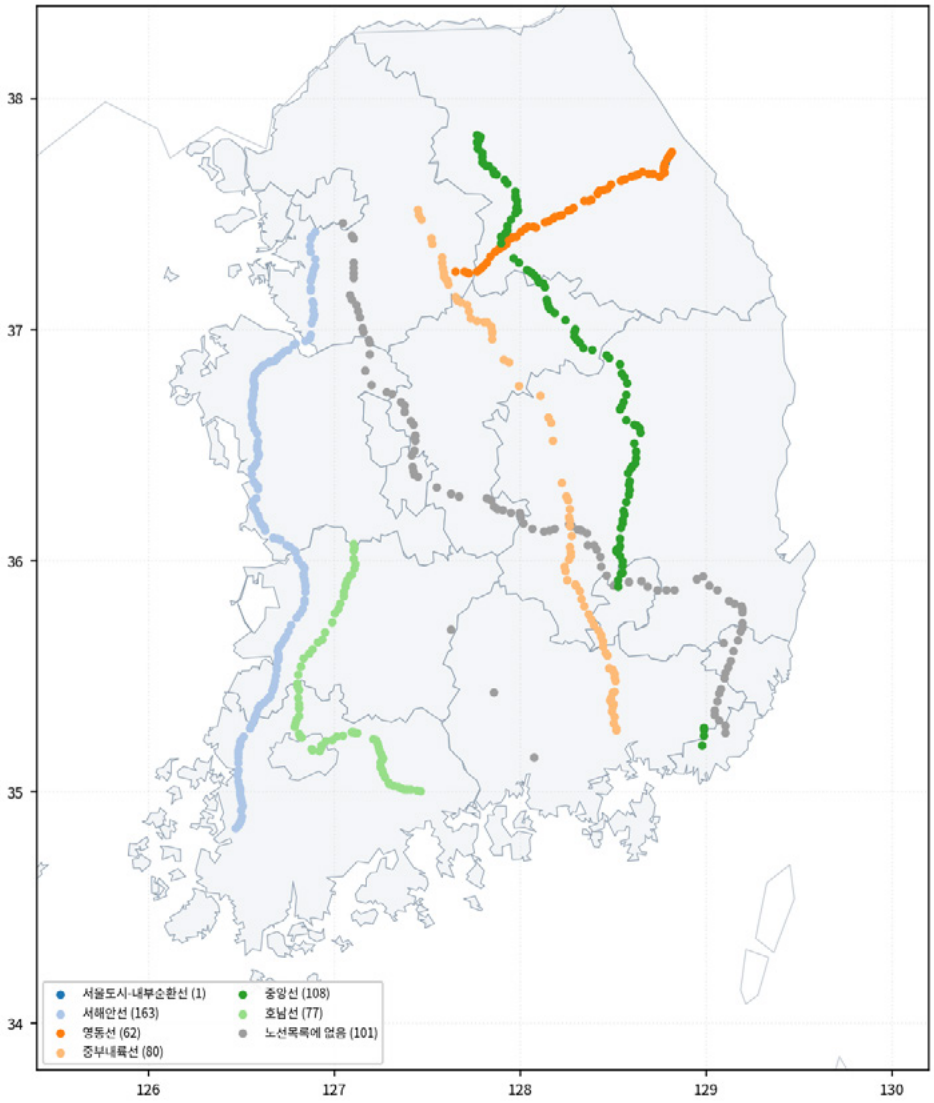


그림 39-1. 노선별 CCTV 관측지점 지도 작성

❸ 같은 시각의 도로날씨 분포

```
def state_table(d):
  g = d.groupby(["WH_CD", "상태"]).size().reset_index(name="지점 수")
  g["비율(%)"] = (g["지점 수"] / len(d) * 100).round(1)
  return g.sort_values("지점 수", ascending=False)
```

WH_CD	상태	지점 수	비율(%)
00	맑음	506	85.5
21	비 약	78	13.2
11	안개 약	3	0.5
13	안개 강	3	0.5
12	안개 중	2	0.3

기상지수

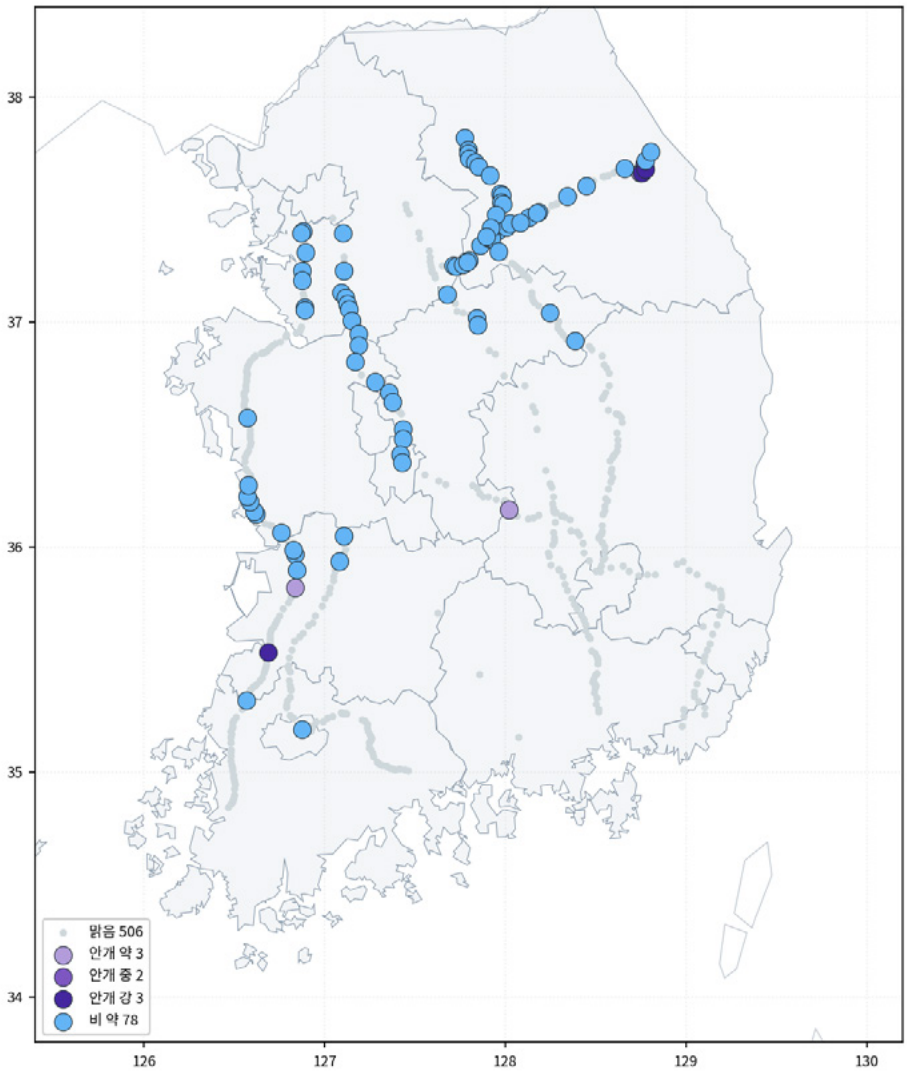


그림 39-2. 같은 시각의 도로날씨 분포

❹ 노선별 도로날씨 판정 비율 비교

노선명을 연결해 노선별 판정 비율을 집계합니다.

```
def route_table(d, min_n=8):
    """노선마다 비를 맞고 있는 구간의 비율. 구간이 적은 노선은 뺀다."""
    g = d.dropna(subset=["노선"]).groupby("노선")
    out = g.apply(
        lambda x: pd.Series(
            {
                "구간 수": len(x),
                "비": int((x["WH_CD"].str[0] == "2").sum()),
                "안개": int((x["WH_CD"].str[0] == "1").sum()),
                "맑음": int((x["WH_CD"] == "00").sum()),
            }
        ),
        include_groups=False,
    )
    out["비 비율(%)"] = (out["비"] / out["구간 수"] * 100).round(1)
    out["안개 비율(%)"] = (out["안개"] / out["구간 수"] * 100).round(1)
    return (
        out[out["구간 수"] >= min_n]
        .sort_values("비 비율(%)", ascending=False)
        .reset_index()
    )
```

노선	구간 수	비	안개	맑음	비 비율(%)	안개 비율(%)
영동선	62	20	5	37	32.3	8.1
중앙선	108	17	0	91	15.7	0
서해안선	163	18	2	143	11	1.2
호남선	77	3	0	74	3.9	0
중부내륙선	80	3	0	77	3.8	0

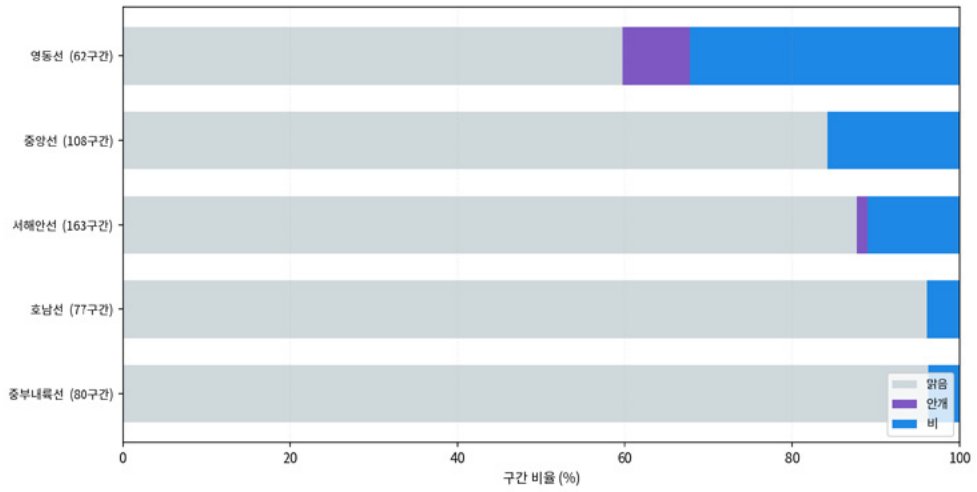


그림 39-3. 노선별 도로날씨 판정 비율 비교

❺ 5분 간격 강수 판정 변화

하루의 5분 자료를 받아 시각별 비 판정 지점 수를 비교합니다.

```
def load_series(path):
    """5분마다 전 지점을 코드별로 세어 둔 표."""
    cols = ["00", "11", "12", "13", "21", "22", "23", "31", "32", "33"]
    rows, on = ([], False)
    for line in Path(path).read_text("utf-8").splitlines():
        s = line.strip()
        if s.startswith("#"):
            on = s.startswith("#DAY")
            continue
        if not on or not s:
            continue
        p = s.split()
        if len(p) != 12:
            continue
        d = {
            "시각": pd.to_datetime(p[0], format="%Y%m%d%H%M"),
            "관측 지점 수": int(p[1]),
        }
        for c, v in zip(cols, p[2:]):
            d[c] = int(v)
        rows.append(d)
    d = pd.DataFrame(rows)
    d["비"] = d[["21", "22", "23"]].sum(axis=1)
    d["안개"] = d[["11", "12", "13"]].sum(axis=1)
    d["눈"] = d[["31", "32", "33"]].sum(axis=1)
    return d
```

기상지수

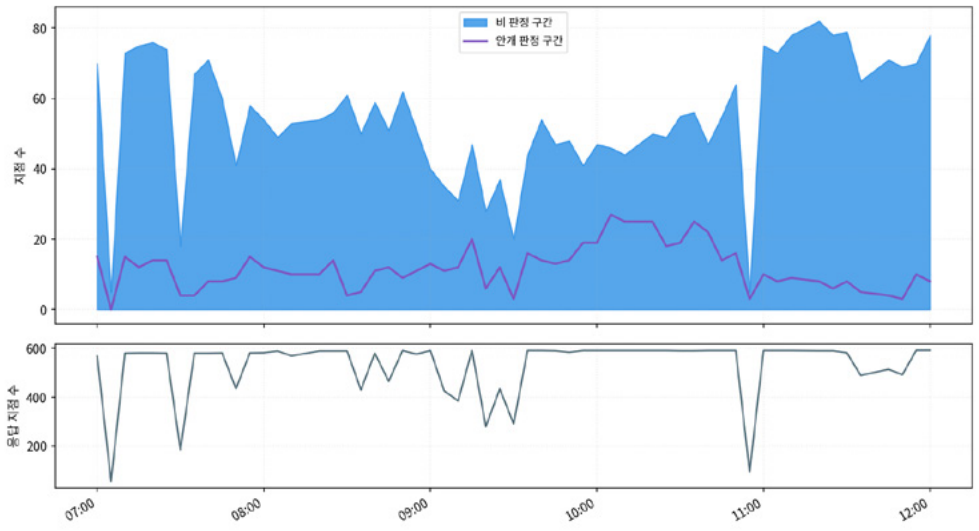


그림 39-4. 5분 간격 강수 판정 변화

❻ 겨울철 눈 판정 구간 확인

겨울 시각을 조회해 눈 판정 코드를 같은 방식으로 지도에 표시합니다.

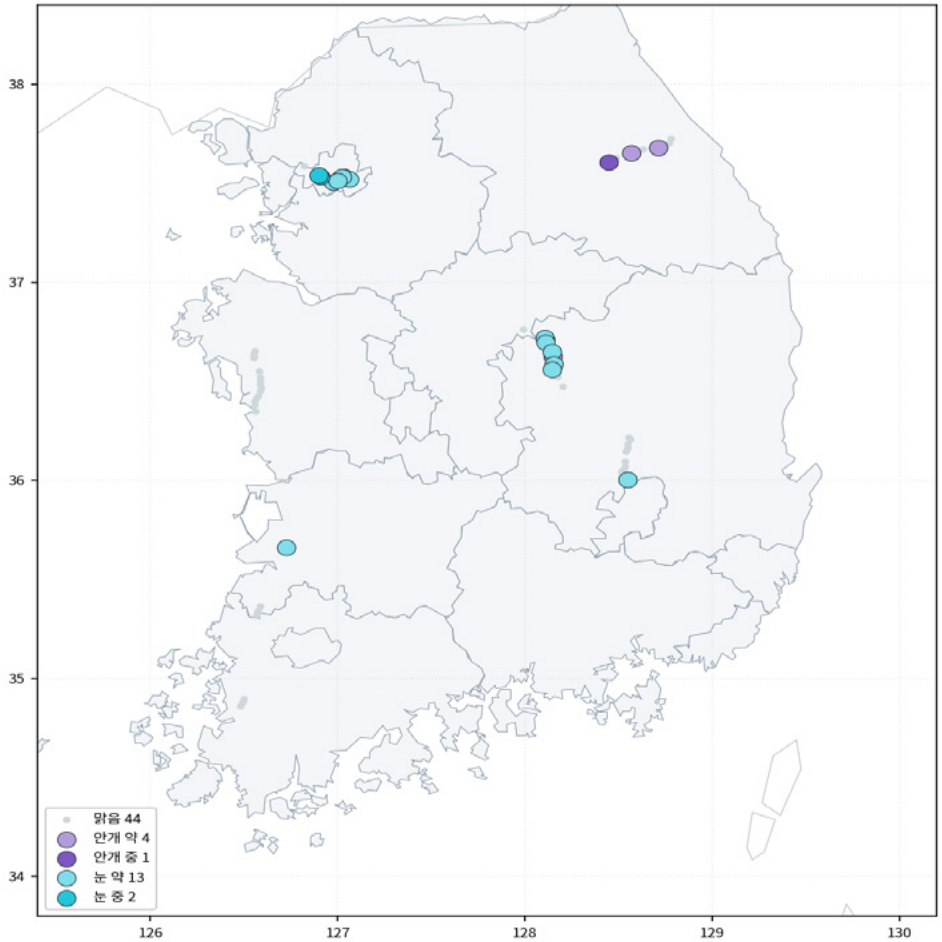


그림 39-5. 겨울철 눈 판정 구간 확인

❼ 종합 코드와 안개·비·눈 코드 비교

```
def code_cross(d):
    """CCTV 상태코드는 비트 조합이 아니다. 개별 코드별 빈도를 집계합니다."""
    g = (
        d.groupby(["WH_CD", "FOG_CD", "RN_CD", "SN_CD"], dropna=False)
        .size()
        .reset_index(name="건수")
    )
    g["상태"] = g["WH_CD"].astype(str).str.zfill(2).map(lambda c: WH.get(c, c))
    return g
```

WH_CD	상태	성분	지점 수
00	맑음	없음	506
21	비 약	비 약	78
11	안개 약	안개 약	3
13	안개 강	안개 강	3
12	안개 중	안개 중	2

기상지수

40. 서리발생예측정보

지역별 서리 발생확률 비교

데이터 활용 경로(API)
기상청 API허브 > 융합기상 > 산업

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests numpy pandas matplotlib cartopy  # 터미널에서 실행
```

❷ 서리 발생확률 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
FROST_URL = "https://apihub.kma.go.kr/api/typ02/openApi/FrstFcstInfoService/getFrstOcurFcst"
def get_frost(base_date, lat, lon, base_time="0500"):
    """생산일(KST)을 명시. HTTP 상태와 resultCode를 차례로 확인한다."""
    rows = []
    for page in range(1, 1000):
        params = {
            "dataType": "JSON",
            "pageNo": page,
            "numOfRows": 1000,
            "BASE_DATE": str(base_date),
            "BASE_TIME": base_time,
            "LAT": float(lat),
            "LON": float(lon),
            "authKey": API_KEY,
        }
        r = requests.get(FROST_URL, params=params, timeout=60)
        r.raise_for_status()
        response = r.json().get("response", {})
        code = str(response.get("header", {}).get("resultCode", ""))
        if code == "03":
            break
        if code != "00":
            raise RuntimeError(f"resultCode {code}")
        body = response.get("body", {})
```

```
    items = (body.get("items") or {}).get("item", [])
    if isinstance(items, dict):
      items = [items]
    if not items:
      break
    rows.extend(items)
    if len(rows) >= int(body.get("totalCount", len(rows))):
      break
  return rows
def frost_values(items):
  out = {}
  for item in items:
    day = pd.to_numeric(item.get("DAY"), errors="coerce")
    value = pd.to_numeric(item.get("VALUE"), errors="coerce")
    if pd.notna(day):
      out[int(day)] = (
        float(value) if pd.notna(value) and 0 <= value <= 100 else np.nan
      )
  return out
```

실행용 생산일은 입력으로 지정하고, 2025~2026년 그림·표의 날짜는 기존 분석결과의 재현 조건으로 구분합니다. 자료가 없는 날·좌표의 빈 응답은 결측으로 유지합니다. 과거 조회 가능 기간과 실제 응답을 확인한 뒤 기간을 지정합니다.

```
BASE_DATE = input("조회 가능한 서리 생산일(KST, YYYYMMDD): ").strip()
items = get_frost(BASE_DATE, 37.57, 126.98)
for x in items:
  print(x)
```

```
{'BASE_DATE': '20251110', 'FCST_DATE': '20251111', 'VALUE': '0',  'BASE_TIME': '0500', 'FCST_TIME':
'0000', 'DAY': 1}
{'BASE_DATE': '20251110', 'FCST_DATE': '20251112', 'VALUE': '37', 'BASE_TIME': '0500', 'FCST_TIME':
'0000', 'DAY': 2}
```

❸ 전국 확률 지도 표출

```
import time, numpy as np, pandas as pd
lats = np.arange(33.0, 38.6, 0.5)
lons = np.arange(125.5, 129.6, 0.5)
rows = []
for la in lats:
  for lo in lons:
    it = get_frost(BASE_DATE, round(la, 2), round(lo, 2))

    p = frost_values(it)
    rows.append({'위도': la, '경도': lo, **{f'{k}일': v for k, v in p.items()}})
    time.sleep(0.3)
df = pd.DataFrame(rows).reindex(columns=["위도", "경도", "1일", "2일"])
```

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
cm = plt.get_cmap('BuPu').copy()
cm.set_bad('#00000000')       # 값 없는 칸은 밑그림이 비쳐 보이게
fig = plt.figure(figsize=(4.5, 5))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([125.4, 130.1, 33.0, 38.7], crs=PC)
```

기상지수

```
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
               linewidth=0.6, edgecolor='#37474f')
piv = df.pivot(index='위도', columns='경도', values='1일')
m = ax.pcolormesh(piv.columns, piv.index, np.ma.masked_invalid(piv.values),
                  cmap=cm, vmin=0, vmax=100, shading='nearest',
                  transform=PC, alpha=0.85, zorder=4)
fig.colorbar(m, ax=ax, label='발생확률 (%)')
plt.show()
```

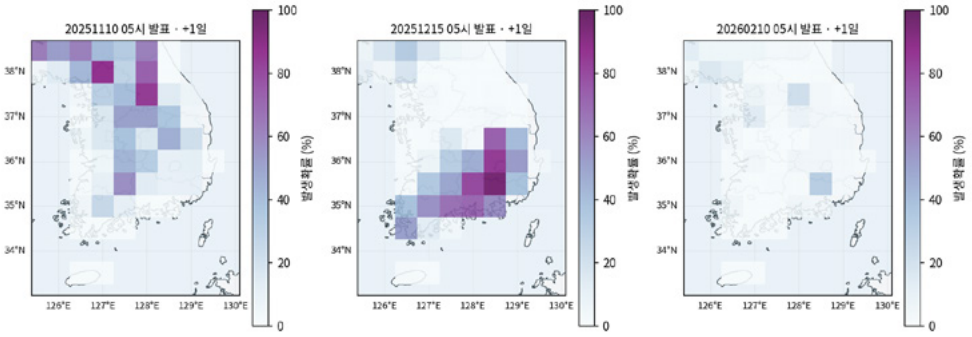


그림 40-1. 기준일 세 개의 +1일 서리 발생확률. 회색은 값이 오지 않은 해상 격자

기준일	값이 온 격자	+1일 평균(%)	+1일 50% 이상	+2일 평균(%)	+2일 50% 이상
20251110	55	23.2	10	38.6	22
20251215	55	20.4	10	5.7	0
20260210	55	3.6	0	61.9	38

표 40-1. 기준일별 확률 요약

❹ 선행일별 확률 비교

같은 응답의 D+1·D+2 값을 격자별로 비교합니다.

```
print(df[['1일', '2일']].describe().round(1))
print(df[['1일', '2일']].corr().round(3))
```

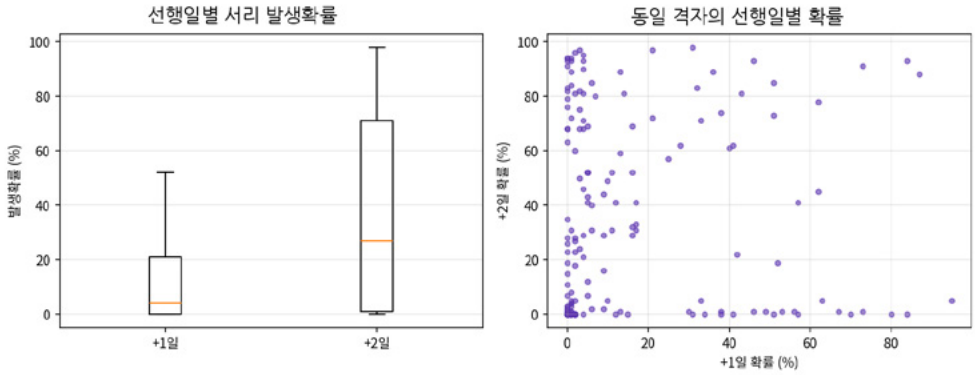


그림 40-2. 선행일별 확률 분포와 같은 격자의 두 값

선행일	평균(%)	중앙값(%)	0%인 격자	50% 이상 격자
+1일	15.7	4	42	20
+2일	35.4	27	37	60

표 40-2. 선행일별 확률 분포 (기준일 세 개 합계)

❺ 지점별 시계열 표출

같은 10개 지점을 10월 15일~4월 15일에 주 단위로 조회해 계절 변화를 비교합니다.

```
stns = {'서울': (37.57, 126.98), '춘천': (37.90, 127.73), '대관령': (37.68, 128.72),
    '청주': (36.64, 127.44), '안동': (36.57, 128.71), '전주': (35.82, 127.15),
    '대구': (35.88, 128.62), '광주': (35.16, 126.89), '부산': (35.10, 129.03),
    '제주': (33.51, 126.53)}
dates = pd.date_range(input("조회 가능한 기간 시작일(YYYY-MM-DD): "), periods=27, freq="7D").
strftime("%Y%m%d")
out = []
for d in dates:
  for nm, (la, lo) in stns.items():
    it = get_frost(d, la, lo)


    p = frost_values(it)
    out.append({'지점': nm, '기준일': d, '확률': p.get(1)})
    time.sleep(0.3)
ts = pd.DataFrame(out).pivot(index='지점', columns='기준일', values='확률')
```

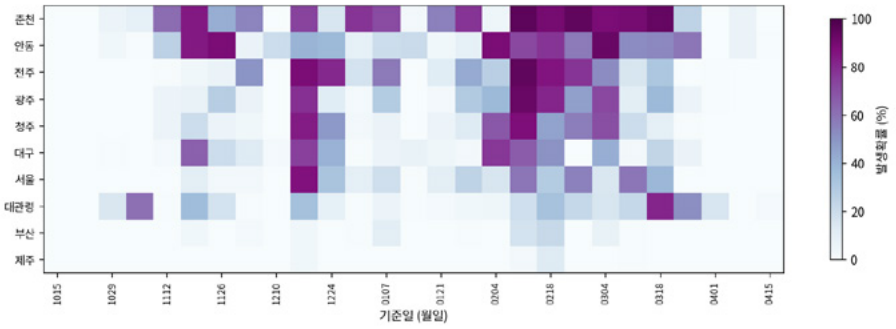


그림 40-3. 지점별 주 단위 +1일 서리 발생확률

지점	평균(%)	최고(%)	최고 기준일	50% 첫 기준일
춘천	45.4	96	20260211	20251112
안동	34.7	93	20260304	20251119
전주	27.8	95	20260211	20251203
광주	22	93	20260211	20251217
청주	20.7	88	20260211	20251217
대구	19	77	20260204	20251119
서울	17.6	87	20251217	20251217
대관령	17.1	82	20260318	20251105
부산	2.6	22	20260218	-
제주	0.8	13	20260218	-

표 40-3. 지점별 확률 요약 (27주)

기상지수

41. 전력기상지수

지역별 지수 변화와 체감온도 비교

데이터 활용 경로(API)

기상청 API허브 > 융합기상 > 산업

❶ 전력기상지수 응답 읽기

응답의 단기예보 격자별 위경도·체감온도를 연결합니다.

```
https://apihub.kma.go.kr/api/typ08/getPowerWeather?tm=202609010500&authKey=...
[{"crtnTime":"202609010500","fcstTime":"202609011100",
  "lat":"39.882","lon":"124.3089","x":"15","y":"177",
  "data":{"elec_idx":"89.71","senta_idx":"25.59"}}, ···]
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
PICK = {} # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
def save_power_csv(items, path="power.csv"):
  d = parse_power(items)
  d.to_csv(path, index=False, encoding="utf-8-sig")
  return d
```

원시 JSON의 자료 목록(items)을 parse_power()로 읽고 save_power_csv(items)로 저장합니다. load_pack("power.csv")는 원래 fcstTime을 보존해 시계열을 계산합니다. 격자 식별자를 시·도나 관측지점명으로 해석하지 않습니다.

```
def parse_power(items):
  """항목 하나가 (예보시각 × 격자점) 이다. 격자에 위경도가 함께 붙어 있어
  좌표표를 따로 만들 필요가 없다."""
  rows = []
  for it in items:
    d = it.get("data", {})
    rows.append(
      {
        "발표시각": it["crtnTime"],
        "예보시각": it["fcstTime"],
        "위도": float(it["lat"]),
        "경도": float(it["lon"]),
        "격자x": int(it["x"]),
        "격자y": int(it["y"]),
        "전력기상지수": float(d["elec_idx"]),
        "체감온도(℃)": float(d["senta_idx"]),
      }
```

```
    )
  d = pd.DataFrame(
    rows,
    columns=(
      None
      if rows
      else [
        "발표시각",
        "예보시각",
        "위도",
        "경도",
        "격자x",
        "격자y",
        "전력기상지수",
        "체감온도(℃)",
      ]
    ),
  )
  for c in ["발표시각", "예보시각"]:
    d[c] = pd.to_datetime(d[c], format="%Y%m%d%H%M").dt.tz_localize(
      "Asia/Seoul"
    )
  return d
```

❷ 전력기상지수와 체감온도 분포 비교

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def load_pack(path):
    """parse_power 출력 CSV에서 시각통계·격자평균·시계열을 계산한다."""
    d = pd.read_csv(path, encoding="utf-8-sig")
    for column in ["예보시각", "발표시각"]:
        d[column] = pd.to_datetime(d[column], utc=True).dt.tz_convert(
            "Asia/Seoul"
        )
    if d.duplicated(["발표시각", "예보시각", "격자x", "격자y"]).any():
        raise ValueError("중복 전력기상 자료")
    if d["발표시각"].nunique() != 1:
        raise ValueError("발표시각 한 개씩 비교")
    t = (
        d.groupby("예보시각")
        .agg(
            격자수=("전력기상지수", "count"),
            지수평균=("전력기상지수", "mean"),
            체감온도평균=("체감온도(℃)", "mean"),
            지수최대=("전력기상지수", "max"),
            지수최소=("전력기상지수", "min"),
        )
        .reset_index()
    )
    d["위도칸"] = np.floor(d["위도"] / 0.2) * 0.2 + 0.1
    d["경도칸"] = np.floor(d["경도"] / 0.2) * 0.2 + 0.1
    cells = (
        d.groupby(["위도칸", "경도칸"])[["전력기상지수", "체감온도(℃)"]]
        .mean()
        .reset_index()
        .rename(columns={"위도칸": "위도", "경도칸": "경도"})
    )
    d["지점"] = d["격자x"].astype(str) + "," + d["격자y"].astype(str)
    return (t, cells, d)

def map_summary(c):
    return {
        "0.2도 칸 수": len(c),
        "전력기상지수 범위": f"{c['전력기상지수'].min():.0f} ~ {c['전력기상지수'].max():.0f}",
        "체감온도 범위": f"{c['체감온도(℃)'].min():.1f} ~ {c['체감온도(℃)'].max():.1f} ℃",
        "지수 120 이상 칸": f"{int((c['전력기상지수'] >= 120).sum())}칸 ({(c['전력기상지수'] >= 120).mean() * 100:.0f} %)",
        "읽는 법": "지수는 100이 기준이다. 값이 클수록 그 시각 그 자리의 기상 조건이 전력수요를 밀어 올린다는 뜻이다",
    }
```

기상지수

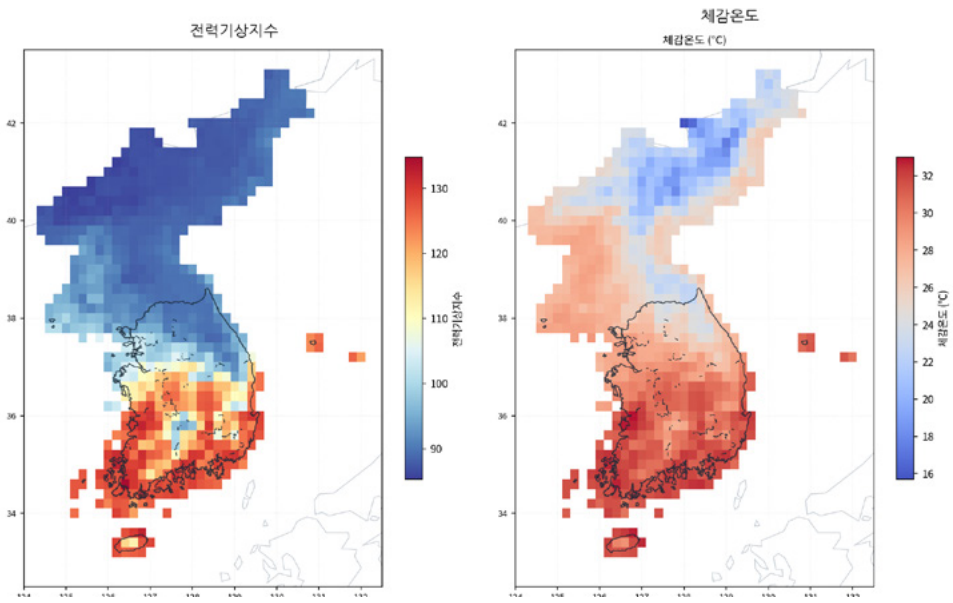


그림 41-1. 전력기상지수와 체감온도 분포 비교

❸ 사흘간 전력기상지수 변화

예보시각별 격자 평균·최대·최소로 사흘간 변화를 비교합니다.

```
def day_table(t):
  d = t.copy()
  d["날짜"] = d["예보시각"].dt.strftime("%m-%d")
  g = d.groupby("날짜").agg(
    예보시각수=("지수평균", "size"),
    지수평균=("지수평균", "mean"),
    지수최대=("지수최대", "max"),
    체감온도평균=("체감온도평균", "mean"),
  )
  return g.round(2).reset_index()
```

날짜	예보시각수	지수평균	지수최대	체감온도평균
09-01	13	114.79	194.6	25.32
09-02	24	101.24	193.9	23.6
09-03	24	98.75	191	22.81
09-04	1	106.17	152.2	20.68

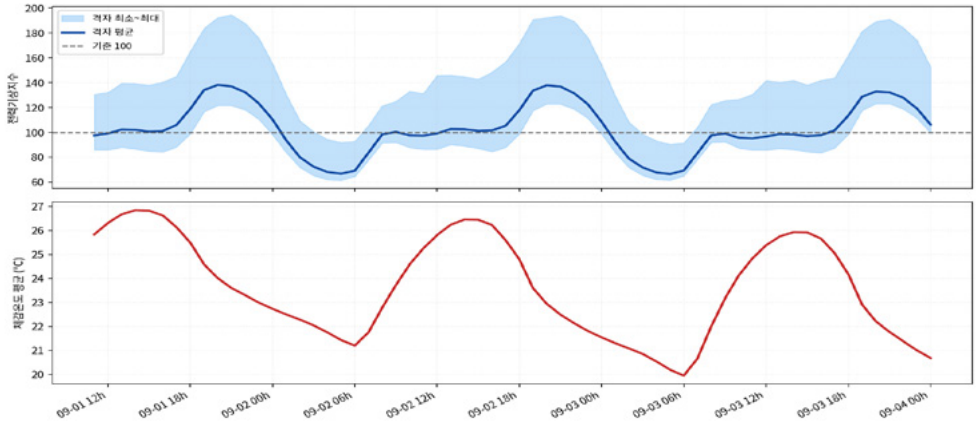


그림 41-2. 사흘간 전력기상지수 변화

❹ 도시별 지수 상승 시간 비교

도시 좌표에 가장 가까운 격자를 추출해 시계열을 비교합니다.

```
def pt_table(d):
  g = d.groupby("지점").agg(
    최소=("전력기상지수", "min"),
    최대=("전력기상지수", "max"),
    평균=("전력기상지수", "mean"),
    체감최고=("체감온도(℃)", "max"),
  )
  g["하루 진폭"] = (g["최대"] - g["최소"]).round(1)
  return g.round(1).sort_values("최대", ascending=False).reset_index()
```

지점	최소	최대	평균	체감최고	하루 진폭
제주	80.3	189.2	132.4	32.7	108.9
부산	79.6	187.9	131.7	32.7	108.3
대구	72.2	184.8	122	31.9	112.6
광주	74.9	182.7	128.2	32.9	107.8
서울	70.1	154	105.9	29.5	83.9
강릉	66.7	140.9	102.5	28.4	74.2

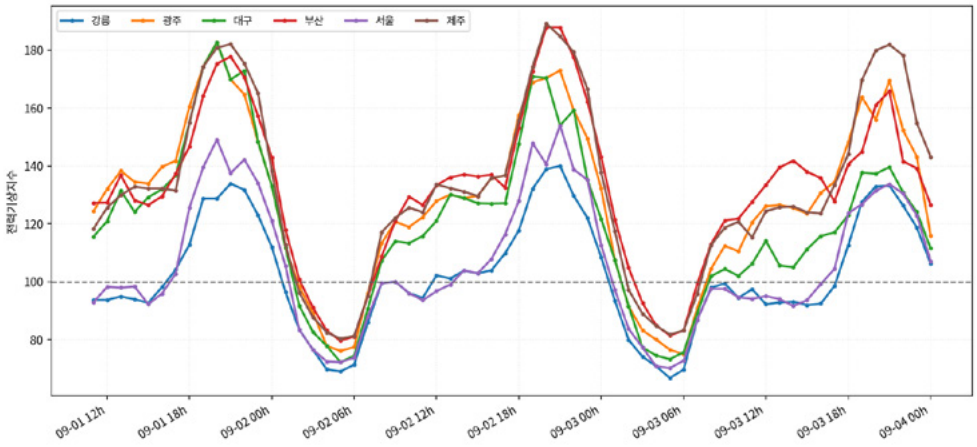


그림 41-3. 도시별 지수 상승 시간 비교

❺ 공간·시간별 체감온도와 지수의 관계

```
def rel_fit(c):
  """체감온도 구간별로 전력기상지수와의 회귀계수를 비교합니다."""

  a, b = np.polyfit(c["체감온도(℃)"], c["전력기상지수"], 1)
  r = np.corrcoef(c["체감온도(℃)"], c["전력기상지수"])[0, 1]
  lo = c[c["체감온도(℃)"] < 27]
  hi = c[c["체감온도(℃)"] >= 27]
  a_lo = np.polyfit(lo["체감온도(℃)"], lo["전력기상지수"], 1)[0]
  a_hi = np.polyfit(hi["체감온도(℃)"], hi["전력기상지수"], 1)[0]
  return {
    "칸 수": len(c),
    "전체 직선 기울기": f"{a:+.1f} /℃",
    "27℃ 미만 기울기": f"{a_lo:+.1f} /℃ ({len(lo)}칸)",
    "27℃ 이상 기울기": f"{a_hi:+.1f} /℃ ({len(hi)}칸)",
    "상관계수 r": f"{r:.3f}",
    "설명력 r²": f"{r ** 2:.3f}",
    "읽는 법": "한 시각 안에서도 관계는 직선이 아니다. 낮은 쪽은 바닥에 눌려 있고 27℃를 넘어서면서 급하게 꺾여 오른다",
  }
```

기상지수

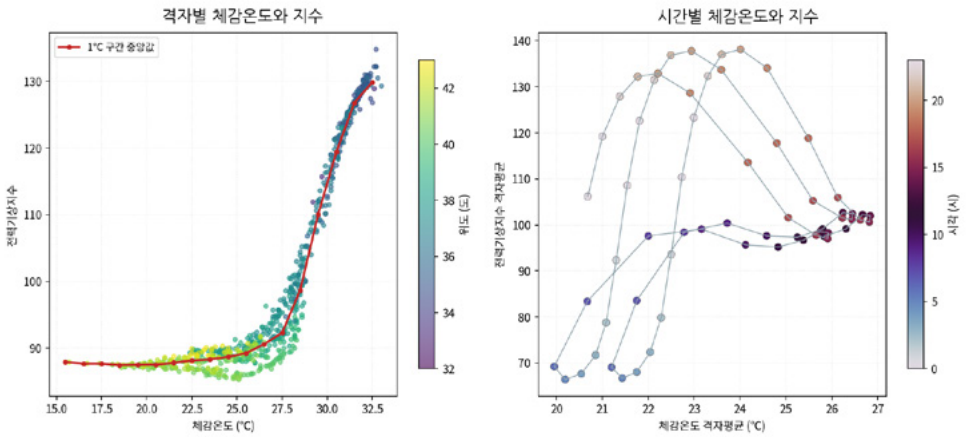


그림 41-4. 공간·시간별 체감온도와 지수의 관계

42.
도로위험기상정보
노선별 위험구간 확인

데이터 활용 경로(API)
기상청 API허브 > 융합기상 > 교통안전

❶ 도로위험자료 다운로드와 링크 읽기

```
https://apihub.kma.go.kr/api/typ04/url/road_file_down.php?tm=202608301600&roadNum=045&auth
Key=...
# ZIP 이 오고 그 안에 링크판과 1 km 격자판이 들어 있다
# ZIP 은 압축하지 않고 담아 둔(stored) 것이라 그대로 읽힌다
LINK_ID,F_NODE,T_NODE,F_LON,F_LAT,T_LON,T_LAT,FOG
1570001502,1570012800,1570012700,128.438251,35.673519,128.439663,35.671051,0
# F_ 는 선분의 시작점, T_ 는 끝점 · 마지막 열이 위험 단계
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
FOG_LEVEL = {} # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
```

범주와 상태는 공식 코드표로 확인합니다. 코드표에 없는 값은 원래 코드로 표시합니다.

```
def get_road(tm, route="045", retry=3):
  """노선·시각별 도로위험기상 ZIP 자료를 조회합니다. 링크자료와 1 km 격자자료를 읽습니다."""

  tm_utc = (
    pd.to_datetime(tm, format="%Y%m%d%H%M") - pd.Timedelta(hours=9)
  ).strftime("%Y%m%d%H%M")
  params = {
    "tm": tm_utc,
    "roadNum": str(route).lstrip("R").zfill(3),
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(
        "https://apihub.kma.go.kr/api/typ04/url/road_file_down.php",
        params=params,
        timeout=120,
      )
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      z = zipfile.ZipFile(io.BytesIO(r.content))
      return {
        n: z.read(n).decode("euc-kr", "replace")
```

기상지수

```
            for n in z.namelist()
            if not n.endswith("/")
        }
    except Exception as exc:
        if i == retry - 1:
            raise
        log(f"{tm} 재시도 {i + 1}: {type(exc).__name__}")
        time.sleep(2 * (i + 1))
def parse_link(text):
  """링크별 시작점·끝점 좌표와 위험정보를 읽습니다."""

  return pd.read_csv(io.StringIO(text))
```

❷ 링크별 위험단계 지도 작성

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def read_link(path):
  d = pd.read_csv(path)
  d["위험도"] = d["FOG"].map(lambda v: FOG_LEVEL.get(int(v), str(v)))
  d["길이(m)"] = [
    km(a, b, c, e) * 1000
    for a, b, c, e in zip(d["F_LON"], d["F_LAT"], d["T_LON"], d["T_LAT"])
  ]
  d["방위(도)"] = [
    bearing(a, b, c, e)
    for a, b, c, e in zip(d["F_LON"], d["F_LAT"], d["T_LON"], d["T_LAT"])
  ]
  d["중앙경도"] = (d["F_LON"] + d["T_LON"]) / 2
  d["중앙위도"] = (d["F_LAT"] + d["T_LAT"]) / 2
  return d
def km(lon1, lat1, lon2, lat2):
  """두 점 사이 거리(km)."""
  r, rad = (6371.0, math.radians)
  a = (
    math.sin(rad(lat2 - lat1) / 2) ** 2
    + math.cos(rad(lat1))
    * math.cos(rad(lat2))
    * math.sin(rad(lon2 - lon1) / 2) ** 2
  )
  return 2 * r * math.asin(math.sqrt(a))
def bearing(lon1, lat1, lon2, lat2):
  """선분이 향한 방향(도). 0 이 북쪽이다."""
  rad = math.radians
  dl = rad(lon2 - lon1)
  y = math.sin(dl) * math.cos(rad(lat2))
  x = math.cos(rad(lat1)) * math.sin(rad(lat2)) - math.sin(
    rad(lat1)
  ) * math.cos(rad(lat2)) * math.cos(dl)
  return (math.degrees(math.atan2(y, x)) + 360) % 360
```

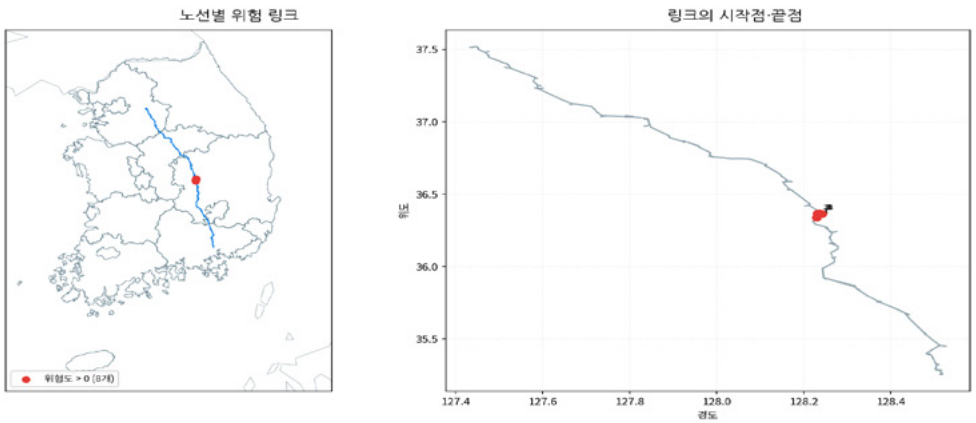


그림 42-1. 링크별 위험단계 지도 작성

❸ 링크별 길이와 방향 계산

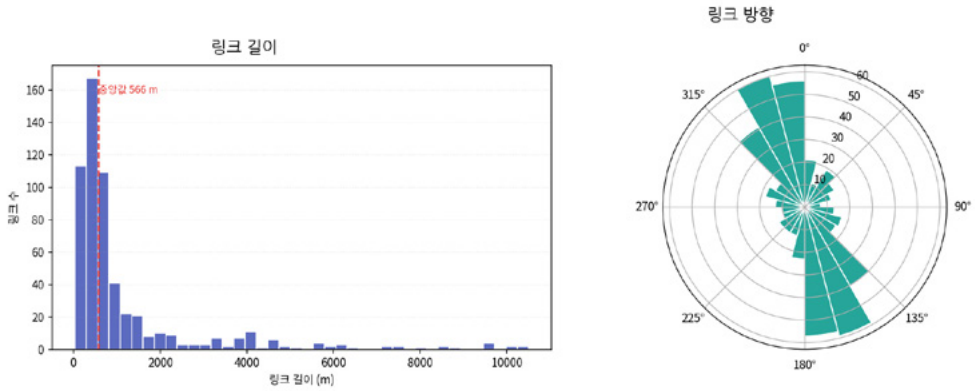


그림 42-2. 링크별 길이와 방향 계산

❹ 링크 중점으로 노선 방향축 구성

```
def axis_dist(d):
  """링크 중점의 주성분 방향으로 투영거리를 계산합니다. 남쪽 끝을 기준으로 한 노선 축 거리이며 실제 도로거리와 구분합니다."""

  x = d["중앙경도"].values * 111 * np.cos(np.radians(d["중앙위도"].values))
  y = d["중앙위도"].values * 111
  p = np.c_[x - x.mean(), y - y.mean()]
  u, sv, vt = np.linalg.svd(p, full_matrices=False)
  ax = vt[0]
  if ax[1] < 0:
    ax = -ax
  t = p @ ax
  return (t - t.min(), float(sv[0] ** 2 / (sv**2).sum()))
def with_dist(d):
  """노선 축 거리를 붙이고 그 순서로 세운다."""
  c = d.copy()
  c["축거리(km)"], c["축설명력"] = axis_dist(d)
  return c.sort_values("축거리(km)").reset_index(drop=True)
```

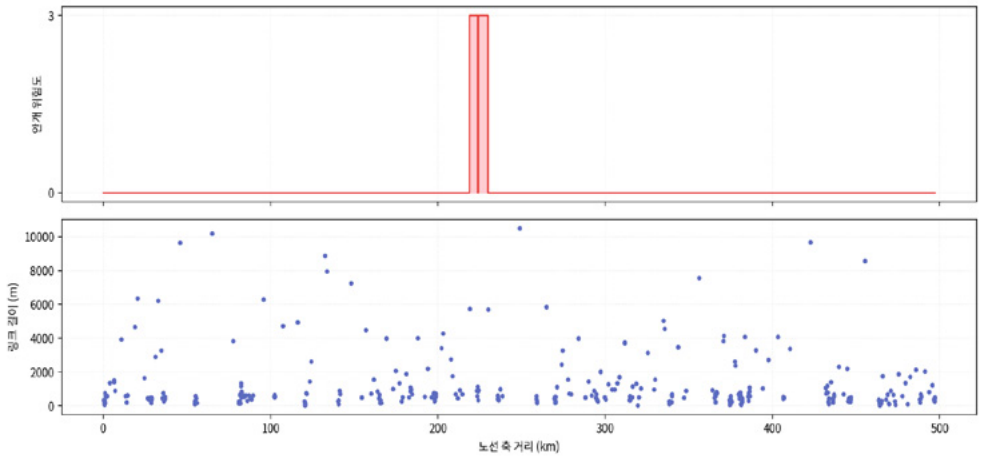


그림 42-3. 링크 중점으로 노선 방향축 구성

FOG	위험도	링크수	길이합(km)	비율(%)
0	없음	564	673.7	98.6
3	경고	8	16.61	1.4

기상지수

❺ 인접 위험 링크의 구간 집계

```
def clusters(c, gap_km=2.0):
    """위험 링크를 노선 축 거리와 지정 간격에 따라 구간으로 묶습니다."""

    r = c[c["FOG"] > 0].sort_values("축거리(km)")
    rows, start, prev, n, mx = ([], None, None, 0, 0)
    for _, x in r.iterrows():
        if start is None or x["축거리(km)"] - prev > gap_km:
            if start is not None:
                rows.append(
                    {
                        "시작(km)": round(start, 2),
                        "끝(km)": round(prev, 2),
                        "길이(km)": round(prev - start, 2),
                        "링크 수": n,
                        "최고 위험도": mx,
                    }
                )
            start, n, mx = (x["축거리(km)"], 0, 0)
        prev = x["축거리(km)"]
        n += 1
        mx = max(mx, int(x["FOG"]))
    if start is not None:
        rows.append(
            {
                "시작(km)": round(start, 2),
                "끝(km)": round(prev, 2),
                "길이(km)": round(prev - start, 2),
                "링크 수": n,
                "최고 위험도": mx,
            }
        )
    return pd.DataFrame(rows)
```

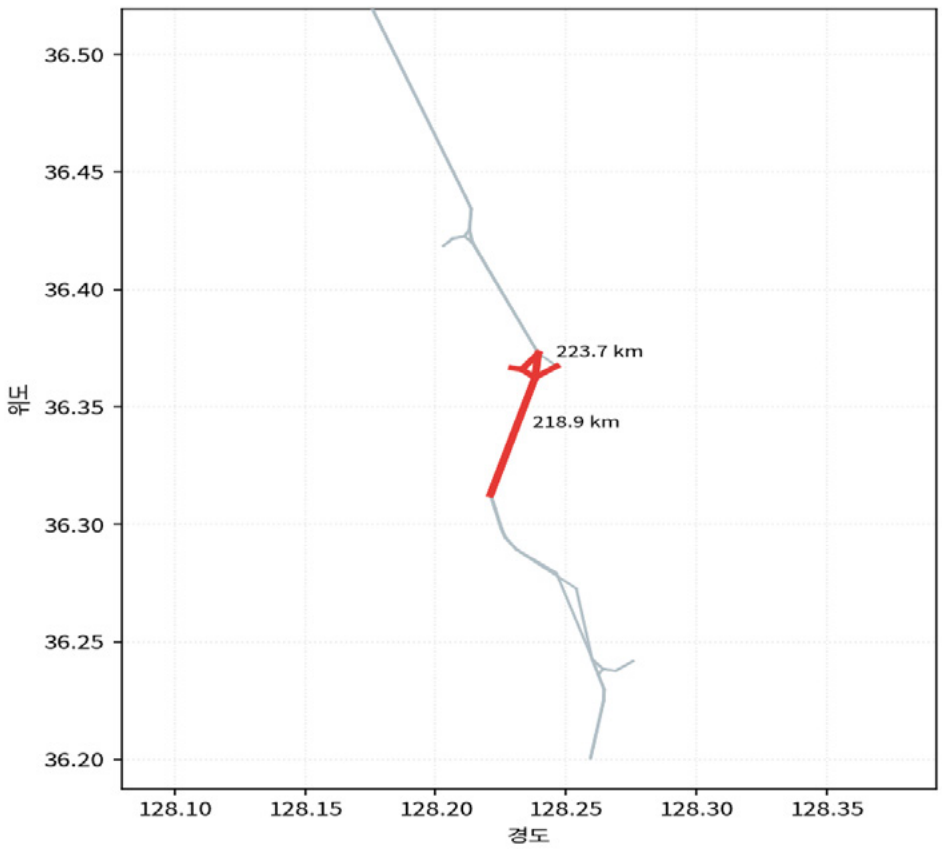


그림 42-4. 인접 위험 링크의 구간 집계

시작(km)	끝(km)	길이(km)	링크 수	최고 위험도
218.94	218.96	0.02	2	3
222.97	224.26	1.28	6	3

LINK_ID	F_LON	F_LAT	T_LON	T_LAT	길이(m)	FOG	위험도
3580008302	128.238	36.3625	128.221	36.3127	5747.06	3	경고
3580008303	128.239	36.3729	128.238	36.3625	1154.55	3	경고
3580008402	128.221	36.3128	128.238	36.3625	5726.49	3	경고
3580008403	128.238	36.3625	128.239	36.3729	1161.46	3	경고
3580053901	128.229	36.3667	128.233	36.366	349.477	3	경고
3580056800	128.233	36.366	128.238	36.3625	593.104	3	경고
3580056901	128.233	36.366	128.239	36.3729	954.869	3	경고
3580206500	128.246	36.3674	128.238	36.3625	923.086	3	경고

❻ 1 km 단위 위험도 요약

링크 중점을 1 km 구간별로 집계해 개수·최고 위험도를 비교합니다.

```
def cell_1km(d):
    """링크 중점을 노선 축의 1 km 구간별로 집계합니다. 링크 수 집계와 실제 위험구간 길이는 구분합니다."""

    e = d.copy()
    e["x"] = np.floor(e["중앙경도"] * 111 * np.cos(np.radians(e["중앙위도"])))
    e["y"] = np.floor(e["중앙위도"] * 111)
    t = (
        e.groupby(["x", "y"])
        .agg(
            링크수=("LINK_ID", "size"),
            최고위험도=("FOG", "max"),
            경도=("중앙경도", "mean"),
            위도=("중앙위도", "mean"),
            길이합=("길이(m)", "sum"),
        )
        .reset_index()
    )
    return t
```

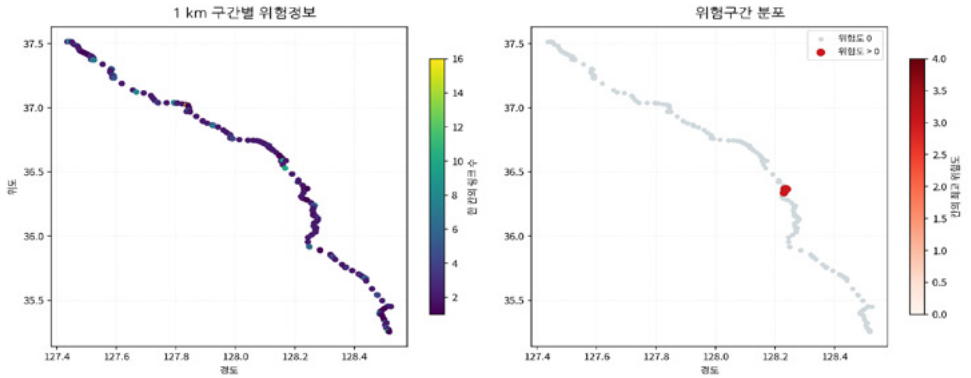


그림 42-5. 1 km 단위 위험도 요약

경도	위도	링크수	길이합	최고위험도
127.838	37.0209	16	4560.39	0
128.515	35.2585	11	3348.5	0
128.161	36.5875	10	4122.04	0
127.47	37.4835	8	3806.45	0
127.442	37.5174	8	3125.87	0
128.437	35.6726	8	4648.67	0

데이터 활용사례 50

지진화산

Earthquakes and Volcanoes



지진화산특정보

43. 지진해일정보

특보 발표 이력과 대상지역 확인

데이터 활용 경로(API)

기상청 API허브 > 지진/화산 > 지진해일정보

❶ 발표 이력 요청과 응답

```
# 1) 최근 3일 조회 (typ02, EqkInfoService) → 응답 예시: 해당 기간 발표 없음(정상)
GET /api/typ02/openApi/EqkInfoService/getTsunamiMsgList?fromTmFc=20260912&toTmFc=20260915&dataType=JSON&numOfRows=100&pageNo=1&authKey=...
{"response":{"header":{"resultCode":"03","resultMsg":"NO_DATA"}}}
# 2) 전체 이력 조회 (typ09, urlTsnmList.do) → 실호출 응답(2026-11-08 확인) (2020~2026 범위로 호출)
GET /api/typ09/url/tsnm/urlTsnmList.do?orderTy=xml&frDate=20200101&laDate=20260915&authKey=...
<alert><tsunam>
 <info><msgKo>지진해일정보</msgKo><flagLocal>N</flagLocal><tmIssue>2024-01-01 16:35:31</tmIssue>
  <eqkTm>2024-01-01 16:10:00</eqkTm><eqkMag>7.4</eqkMag><eqkDepth>-</eqkDepth>
  <eqkLoc>일본 도야마현(혼슈) 도야마 북쪽 90km 해역</eqkLoc><eqkLat>37.50</eqkLat><eqkLon>137.20</eqkLon>
  <wrnHighLoc>-</wrnHighLoc><wrnLowLoc>-</wrnLowLoc></info>
 <info>... 20:06:18 발표 ...</info> <info>... 18:51:17 발표 ...</info>
</tsunam></alert> (2020-01-01~2026-09-15 전체 기간 중 발표 이력은 이 3건뿐)
```

두 API의 조회 범위와 응답 결과는 다음과 같습니다.

API	조회 가능 범위	2026-09-15 기준 조회 결과	비고
typ02 EqkInfoService	최근자료 조회 (fromTmFc~toTmFc)	NO_DATA (정상)	실시간 모니터링용
typ09 urlTsnmList.do	frDate~laDate (orderTy=xml, 2020~2026 범위 1회 조회 가능)	3건 (2024-01-01)	이력 조회·분석용

❷ 조회된 발표 사례: 2024년 도야마현 지진

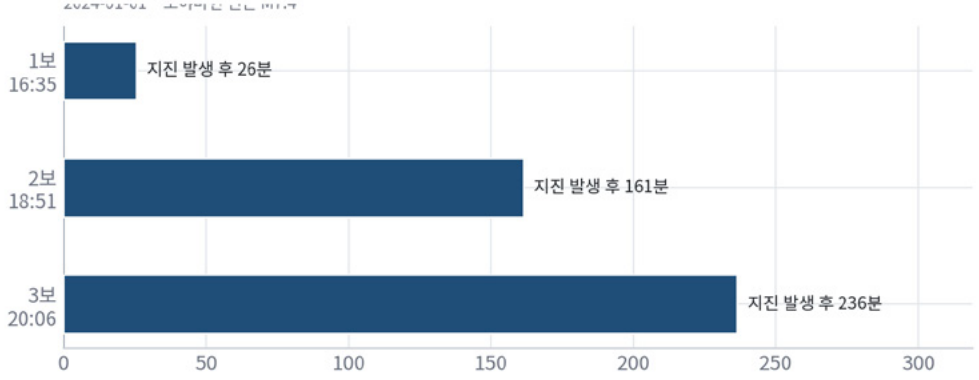


그림 43-1. 조회된 발표 사례: 2024년 도야마현 지진

❸ 3건의 특보 관련 필드 확인

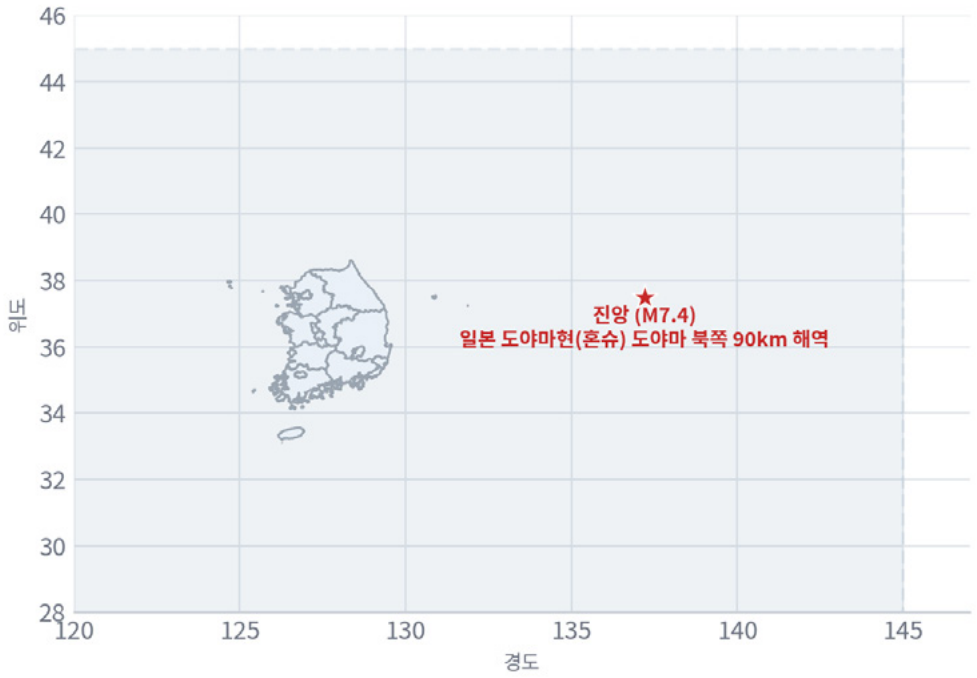


그림 43-2. 3건의 특보 관련 필드 확인

❹ 빈 응답과 조회 오류 구분

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
TYP02_URL = "https://apihub.kma.go.kr/api/typ02/openApi/EqkInfoService/getTsunamiMsgList"
def request_items(url, params):
  r = requests.get(url, params=params, timeout=60)
  r.raise_for_status()
  response = r.json().get("response", {})
  head = response.get("header", {})
  code = str(head.get("resultCode", ""))
  if code == "03":  # 자료 없음(정상) → 빈 결과
   return ([], 0)
  if code != "00":
   raise RuntimeError(head.get("resultMsg", code))
  body = response.get("body", {})
  items = (body.get("items") or {}).get("item", [])
  if isinstance(items, dict):  # 1건이면 dict로 옴
   items = [items]
  return (items, int(body.get("totalCount", len(items))))

def poll_tsunami_info(auth_key):
```

지진화산특정보

```
    today = datetime.now(KST).date()
    params = {
        "fromTmFc": (today - timedelta(days=2)).strftime("%Y%m%d"),
        "toTmFc": today.strftime("%Y%m%d"),
        "dataType": "JSON",
        "numOfRows": 100,
        "pageNo": 1,
        "authKey": auth_key,
    }
    items, total = request_items(TYP02_URL, params)
    if not items:
        return None
return items  # numOfRows=100: 3일치 발표문은 한 쪽에 모두 포함
```

44.
지진정보
발생 위치·규모·진도 분석

데이터 활용 경로(API)
기상청 API허브 > 지진/화산 > 국내·외 지진정보

❶ 라이브러리 설치
필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas numpy matplotlib cartopy  # 터미널에서 실행
```

❷ 지진 통보문 조회
API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
api_url = 'https://apihub.kma.go.kr/api/typ09/url/eqk/urlNewNotiEqk.do'
F = ['msgCode', 'cntDiv', 'arDiv', 'eqArCdNm', 'eqPt', 'nkDiv', 'tmIssue',
   'eqDate', 'magMl', 'magDiff', 'eqDt', 'eqLt', 'eqLn',
   'majorAxis', 'minorAxis', 'depthDiff', 'jdLoc', 'jdLocA']
def get_eqk(fr, la):
  res = requests.get(
    api_url,
    params={
      "fe": "f",
      "orderTy": "xml",
      "frDate": fr,
      "laDate": la,
      "authKey": API_KEY,
    },
    timeout=60,
  )
  res.raise_for_status()
  root = ET.fromstring(res.content)
  rows = []
  for info in root.iter():
    if info.tag.split("}")[-1] != "info":
      continue
    values = {
      el.tag.split("}")[-1]: "".join(el.itertext()).strip() for el in info
    }
    if any(("등록된 내용이 없습니다" in value for value in values.values())):
```

지진화산특정보

```
        continue
      if not values.get("eqDate"):
        continue
      rows.append([values.get(field, "") for field in F])
    return pd.DataFrame(rows, columns=F)
END = '20260920'  # 표 44-1 기준일(발표 기준). 최신 자료는 오늘 날짜로 변경
d = pd.concat([get_eqk(f'{y}0101', min(f'{y}1231', END)) for y in range(2018, 2027)])
print(d['msgCode'].value_counts())
국외지진정보   1215
지진정보     734
지진속보      17
지진조기경보     1
지진현장경보     1
```

통보문 종류	건수	국내(%)	보유 시작	보유 끝
국외지진정보	1215	0	2018-01-10	2026-09-20
지진정보	734	99.6	2018-01-01	2026-09-15
지진속보	17	94.1	2018-02-11	2025-02-07
지진조기경보	1	100	2021-12-14	2021-12-14
지진현장경보	1	100	2024-06-12	2024-06-12

표 44-1. 통보문 종류별 건수와 보유 범위

```
d = d.replace('-', np.nan)
for c in ['magMl', 'magDiff', 'eqDt', 'eqLt', 'eqLn',
      'majorAxis', 'minorAxis', 'depthDiff']:
  d[c] = pd.to_numeric(d[c], errors='coerce')
d['발생'] = pd.to_datetime(d['eqDate'], format='%Y%m%d%H%M%S')
d['발표'] = pd.to_datetime(d['tmIssue'], format='%Y%m%d%H%M%S')
```

❸ 국내·국외 지진 분포

위치·규모·깊이를 각각 좌표·점 크기·색으로 표시합니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
dom = d[(d['cntDiv'] == 'Y') & (d['msgCode'] == '지진정보')].dropna(
  subset=['eqLt', 'eqLn', 'magMl'])
fig = plt.figure(figsize=(6.2, 6.4))
ax = fig.add_subplot(111, projection=PC)
ax.set_extent([123.5, 132.5, 32.5, 43.2], crs=PC)
ax.add_feature(cfeature.OCEAN.with_scale('10m'), facecolor='#eaf2f8')
ax.add_feature(cfeature.LAND.with_scale('10m'), facecolor='#f4f6f8')
ax.add_feature(cfeature.COASTLINE.with_scale('10m'),
        linewidth=0.6, edgecolor='#37474f')
sc = ax.scatter(dom['eqLn'], dom['eqLt'], s=4 + 4 * dom['magMl'] ** 2,
         c=dom['eqDt'], cmap='viridis_r', vmin=0, vmax=25,
         alpha=0.75, edgecolors='#263238', linewidths=0.3,
         transform=PC)
fig.colorbar(sc, ax=ax, extend='max', label='깊이 (km)')
plt.show()
```

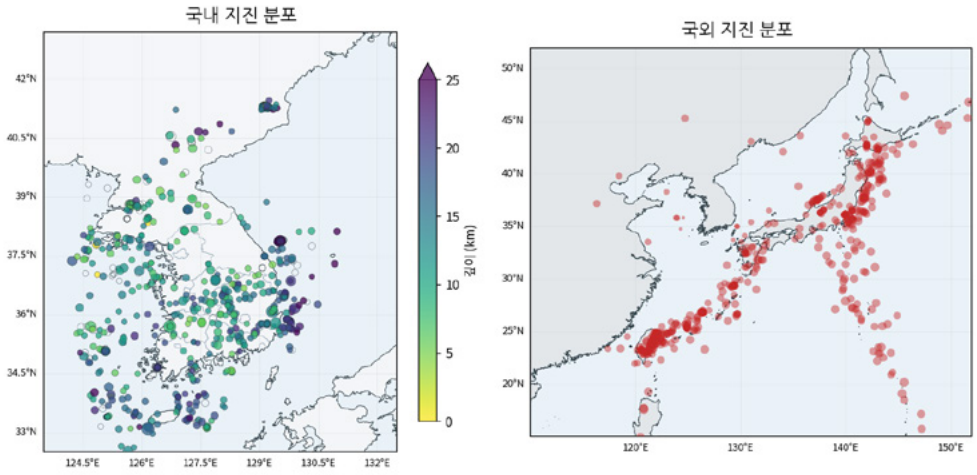


그림 44-1. 국내·국외 지진 통보문 분포(2018-01-01~2026-09-20 발표기준)

발생시각	위치	규모	깊이(km)	최대진도
2021-12-14 17:19	제주 서귀포시 서남서쪽 41km 해역	4.9	17	V
2024-06-12 08:26	전북 부안군 남남서쪽 4km 지역	4.8	8	V
2018-02-11 05:03	경북 포항시 북구 북서쪽 5km 지역	4.6	14	-
2023-05-15 06:27	강원 동해시 북동쪽 52km 해역	4.5	32	III
2019-04-19 11:16	강원 동해시 북동쪽 54km 해역	4.3	32	IV
2019-02-10 12:53	경북 포항시 북구 동북동쪽 50km 해역	4.1	21	III
2022-10-29 08:27	충북 괴산군 북동쪽 11km 지역	4.1	12	V
2021-08-21 09:40	전북 군산시 어청도 서남서쪽 124km 해역	4.0	7	II
2023-11-30 04:55	경북 경주시 동남동쪽 19km 지역	4.0	12	V
2019-06-27 15:19	북한 황해북도 송림 북동쪽 9km 지역	3.9	-	I

표 44-2. 국내 규모 상위 10건

❹ 규모와 최대진도

```
ROM = ['I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII']
jd = dom.dropna(subset=['jdLoc'])
print(jd.groupby('jdLoc')['magMl'].agg(['size', 'median'])
    .reindex([r for r in ROM if (jd['jdLoc'] == r).any()]))
    size  median
jdLoc
I    438    2.3
II    95    2.2
III   98    2.4
IV    33    2.7
V     10    3.5
```

지진화산특정보

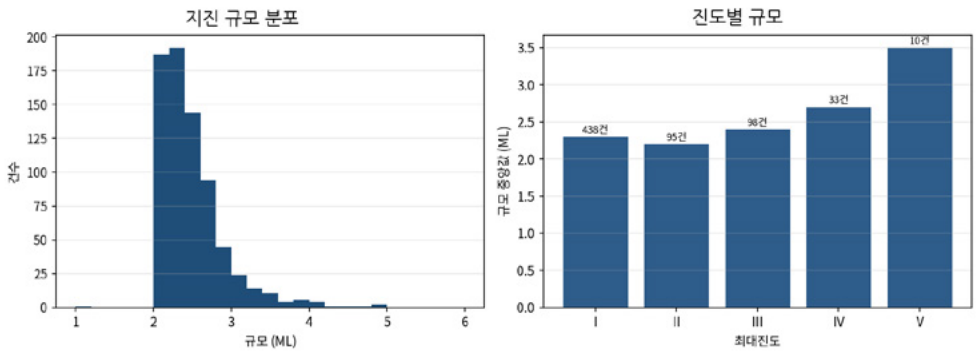


그림 44-2. 국내 지진 규모 분포와 진도 등급별 규모

최대진도	건수	규모 중앙값	규모 최대	깊이 중앙값(km)
I	438	2.3	3.9	16
II	95	2.2	4.0	14
III	98	2.4	4.5	13
IV	33	2.7	4.3	10
V	10	3.5	4.9	12

표 44-3. 최대진도 등급별 규모와 깊이

❺ 지진 발표 소요시간과 결측 필드

```
d['지연'] = (d['발표'] - d['발생']).dt.total_seconds() / 60
print(d.groupby('msgCode')['지연'].median().round(1))
국외지진정보  19.2
지진정보  4.0
지진속보  0.3
```

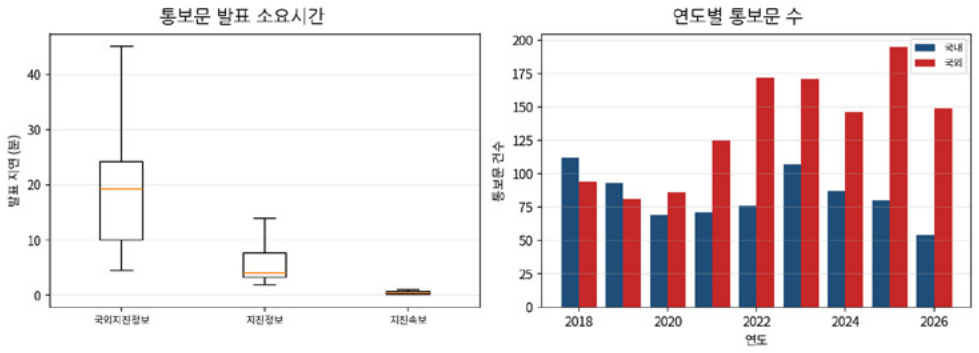


그림 44-3. 통보문 종류별 발표 지연과 연도별 발표 건수

통보문 종류	건수	지연 중앙값(분)	지연 25%(분)	지연 75%(분)
국외지진정보	1214	19.2	10	24.2
지진정보	734	4	3.2	7.7
지진속보	17	0.3	0.2	0.7

표 44-4. 발표 지연 (분)

```
UNC = ['magDiff', 'majorAxis', 'minorAxis', 'depthDiff', 'jdLoc']
for name, g in {'국내 지진정보': dom,
                '지진속보': d[d['msgCode'] == '지진속보'],
                '국외지진정보': d[d['cntDiv'] == 'N']}.items():
    print(name, (g[UNC].notna().mean() * 100).round(0).tolist())
```

필드	국내 지진정보	지진속보	국외지진정보
규모불확도	79%	0%	0%
위치불확도(장축)	79%	0%	0%
위치불확도(단축)	79%	0%	0%
깊이불확도	79%	0%	0%
최대진도	92%	94%	1%

표 44-5. 통보문 종류별 필드 유효자료 비율

연도	국내	국외	합계
2018	112	94	206
2019	93	81	174
2020	69	86	155
2021	71	125	196
2022	76	172	248
2023	107	171	278
2024	87	146	233
2025	80	195	275
2026	54	149	203

표 44-6. 연도별 통보문 건수

데이터 활용사례 50

기상연구

Meteorological Research



해양기상

45. 기상관측선 관측

항적별 해양기상 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 해양관측 > 해양기상부이·파고부이관측

❶ 해상 관측자료 조회와 관측선 추출

시각별 위경도를 연결해 항적과 관측값을 분석합니다.

```
https://apihub.kma.go.kr/api/typ01/url/sea_obs.php?tm=202607050900&stn=0&help=0&authKey=...
# TP,    TM,   STN_ID, STN_KO,    LON,     LAT,  WH, WD,  WS, WS_GST, TW,  TA,    PA, HM
J, 202607050900,  22003, 기상1호, 124.4206, 32.5503, 0.6, 201, 4.0,   9.2,  , 23.5, 1006.5, 99
#  TP : B 부이 · C 파고부이 · D 표류부이 · L 등표 · N 조위 · F 연안방재 · G 파랑계 · J 기상1호
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
SEA_URL = "https://apihub.kma.go.kr/api/typ01/url/sea_obs.php"
```

PORT : 분석 기준 항구의 (경도,위도). 실제 항구 메타데이터 적용
필수 입력: PORT. 자료의 격자 크기·도법·단위·결측 코드는 파일 헤더 또는 API 명세로 확인합니다.
조회시각은 자료의 실제 생산시각을 지정하고, 관련 함수에 같은 입력값을 전달합니다.

```
def get_sea(tm, stn=0, retry=3):
    """해상 관측자료. tm은 KST 기준이며 요청시간에서 가장 가까운 생산시각의 자료가 반환된다.
    일곱 종류(B·C·D·L·N·F·G·J)가 함께 오고, 관측선은 맨 앞 칸이 J 다."""
    params = {"tm": tm, "stn": stn, "help": 0, "authKey": API_KEY}
    for i in range(retry):
        try:
            r = requests.get(SEA_URL, params=params, timeout=60)
            if not r.ok:
                raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
            r.encoding = "euc-kr"
            return r.text
        except Exception as exc:
            if i == retry - 1:
                raise
            log(f"{tm} 재시도 {i + 1}: {type(exc).__name__}")
            time.sleep(2 * (i + 1))
def parse_ship(text):
    """한 응답에서 관측선 줄만 추출합니다. 열은 TP, TM, STN_ID, STN_KO, LON, LAT, WH, WD, WS, WS_GST, TW, TA,
PA, HM."""
```

```
rows = []
for line in text.splitlines():
  if not line.startswith("J,"):
    continue
  f = [s.strip() for s in line.rstrip("=, ").split(",")]
  rows.append(
    {
      "시각": f[1],
      "경도": float(f[4]),
      "위도": float(f[5]),
      "유의파고(m)": float(f[6]),
      "풍향(도)": float(f[7]),
      "풍속(m/s)": float(f[8]),
      "돌풍(m/s)": float(f[9]),
      "기온(°C)": float(f[11]),
      "해면기압(hPa)": float(f[12]),
      "습도(%)": float(f[13]),
    }
  )
return pd.DataFrame(
  rows,
  columns=(
    None
    if rows
    else [
      "시각",
      "경도",
      "위도",
      "유의파고(m)",
      "풍향(도)",
      "풍속(m/s)",
      "돌풍(m/s)",
      "기온(℃)",
      "해면기압(hPa)",
      "습도(%)",
    ]
  ),
)
```

❷ 일별 위치로 연간 항적 작성

하루 한 시각의 위치를 연결해 연간 항적을 그립니다. 모항에서 멀어진 기간을 항차 후보로 구분합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def load_pack(path):
  """수집한 관측선 자료를 읽고 결측과 분석영역 밖 좌표를 처리합니다."""


  cols = [
    "시각",
    "경도",
    "위도",
    "유의파고(m)",
    "풍향(도)",
    "풍속(m/s)",
    "돌풍(m/s)",
    "수온(°C)",
    "기온(°C)",
    "해면기압(hPa)",
    "습도(%)",
  ]
  d = pd.read_csv(path, comment="#", names=cols, header=None)
  d["시각"] = pd.to_datetime(d["시각"].astype(str), format="%Y%m%d%H%M")
  ok = d["경도"].between(120, 136) & d["위도"].between(29, 45)
  return d[ok].sort_values("시각").reset_index(drop=True)
def km(lon1, lat1, lon2, lat2):
  """두 좌표 사이의 근사 거리(km)를 계산합니다. 경도 차이에 평균 위도의 코사인을 적용하며 실제 항해거리와 구분합니다."""
  la = np.radians((np.asarray(lat1) + np.asarray(lat2)) / 2)
  dx = (np.asarray(lon2) - np.asarray(lon1)) * 111.32 * np.cos(la)
  dy = (np.asarray(lat2) - np.asarray(lat1)) * 110.57
  return np.hypot(dx, dy)
```

해양기상

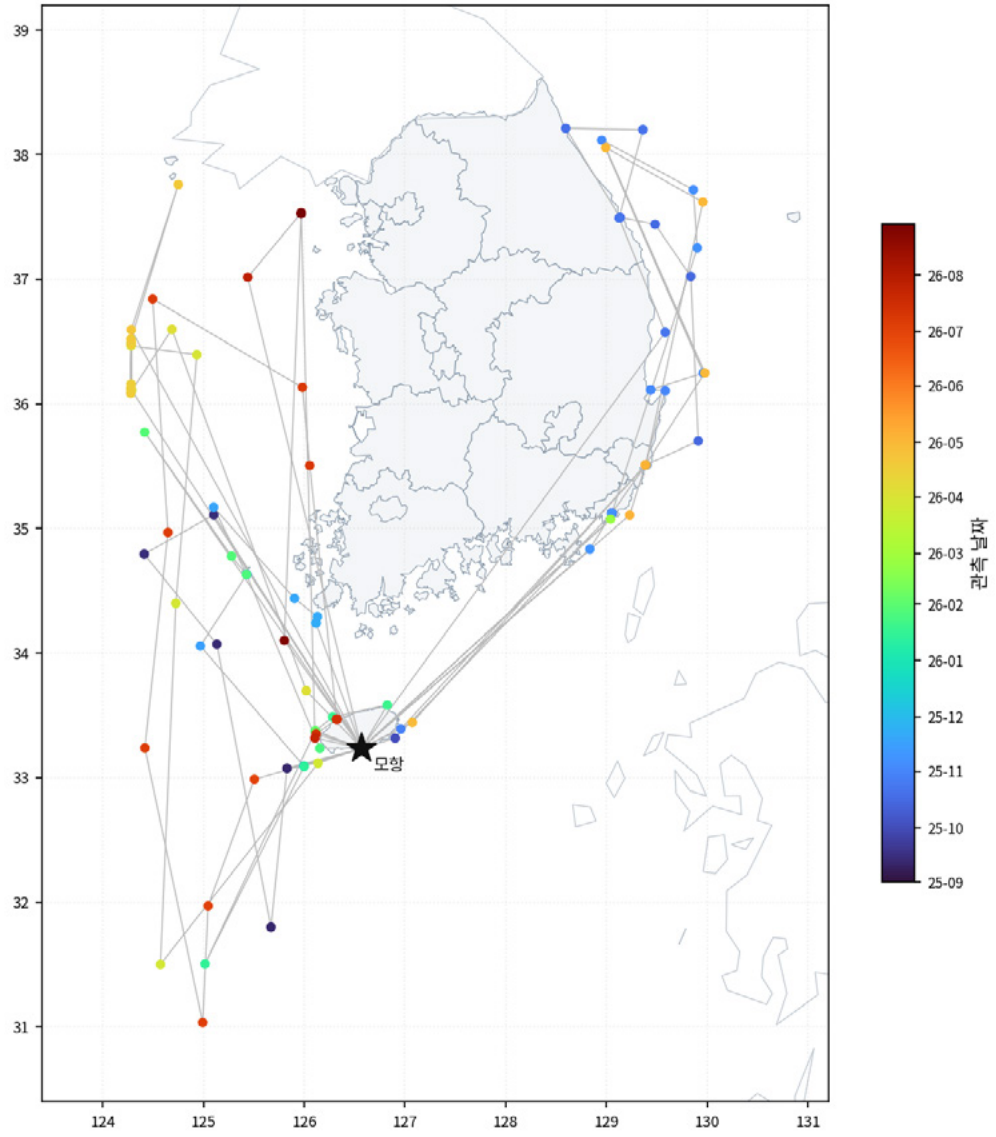


그림 45-1. 일별 위치로 연간 항적 작성

❸ 모항 이탈 기간별 항차 구분

모항에서 떨어진 날이 연속되는 기간을 한 항차로 구분합니다.

```
def port_days(d, r=5.0, *, PORT):
    """모항에서 반경 r km 안에 있으면 정박으로 본다."""
    dist = km(d["경도"], d["위도"], PORT[0], PORT[1])
    return dist <= r
def cruise_table(d, *, PORT):
    """항차를 나눈다. 모항을 떠난 날부터 돌아온 날까지가 한 항차다."""
    day = d[d["시각"].dt.hour == 12].copy()
    day["정박"] = port_days(day, PORT=PORT)
    day["항차"] = (day["정박"] != day["정박"].shift()).cumsum()
    rows = []
    for _, g in day[~day["정박"]].groupby("항차"):
        far = km(g["경도"], g["위도"], PORT[0], PORT[1]).max()
        rows.append(
            {
                "출항": g["시각"].min().strftime("%Y-%m-%d"),
                "귀항": g["시각"].max().strftime("%Y-%m-%d"),
```

```
            "일수": len(g),
            "최북단": round(g["위도"].max(), 2),
            "최서단": round(g["경도"].min(), 2),
            "모항에서 최대(km)": round(far, 0),
            "평균 풍속(m/s)": round(g["풍속(m/s)"].mean(), 1),
        }
    )
    return pd.DataFrame(
        rows,
        columns=(
            None
            if rows
            else [
                "출항",
                "귀항",
                "일수",
                "최북단",
                "최서단",
                "모항에서 최대(km)",
                "평균 풍속(m/s)",
            ]
        ),
    )
```

출항	귀항	일수	최북단	최서단	모항에서 최대(km)	평균 풍속 (m/s)
2025-09-11	2025-09-15	5	35.11	124.41	263	6.1
2025-10-01	2025-10-23	23	38.21	126.9	604	4.2
2025-10-30	2025-11-09	11	38.12	126.96	581	6.8
2026-07-02	2026-07-15	14	36.84	124.42	441	5.7
2026-07-23	2026-07-27	5	37.53	125.44	478	6.9
2026-08-26	2026-08-29	4	37.53	125.8	478	2.8

❹ 1시간 간격 항적과 속력 계산

```
def add_move(d):
    """앞 자료와의 거리·시간으로 속력을 산출합니다. 속력이 곧 항해와 정박을 가른다."""
    e = d.copy()
    e["이동(km)"] = np.r_[
        0.0, km(e["경도"][:-1], e["위도"][:-1], e["경도"][1:], e["위도"][1:])
    ]
    dt = e["시각"].diff().dt.total_seconds().fillna(0) / 3600
    e["간격(h)"] = dt
    e["속력(kt)"] = np.where(
        dt > 0, e["이동(km)"] / dt.replace(0, np.nan) / 1.852, 0
    )
    e.loc[e["간격(h)"] > 3, ["이동(km)", "속력(kt)"]] = np.nan
    e["항해중"] = e["속력(kt)"] > 1.0
    return e
```

해양기상

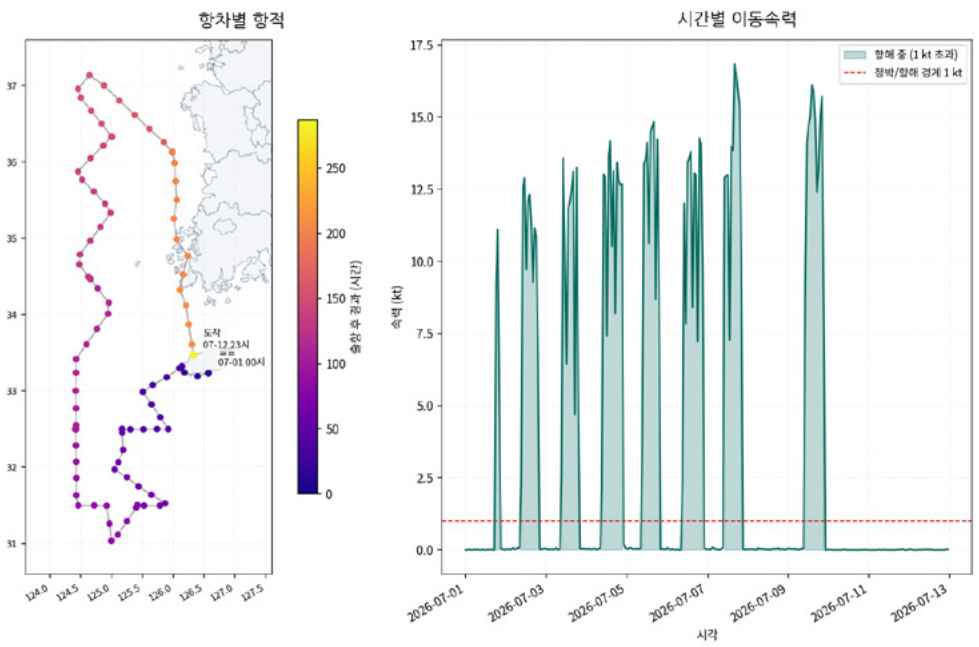


그림 45-2. 1시간 간격 항적과 속력 계산

❺ 항적별 바람·파고·기압 비교

이동 관측선의 시계열에는 시간 변화와 위치 변화가 함께 반영됩니다. 항해 구간을 표시해 관측값 변화를 비교합니다.

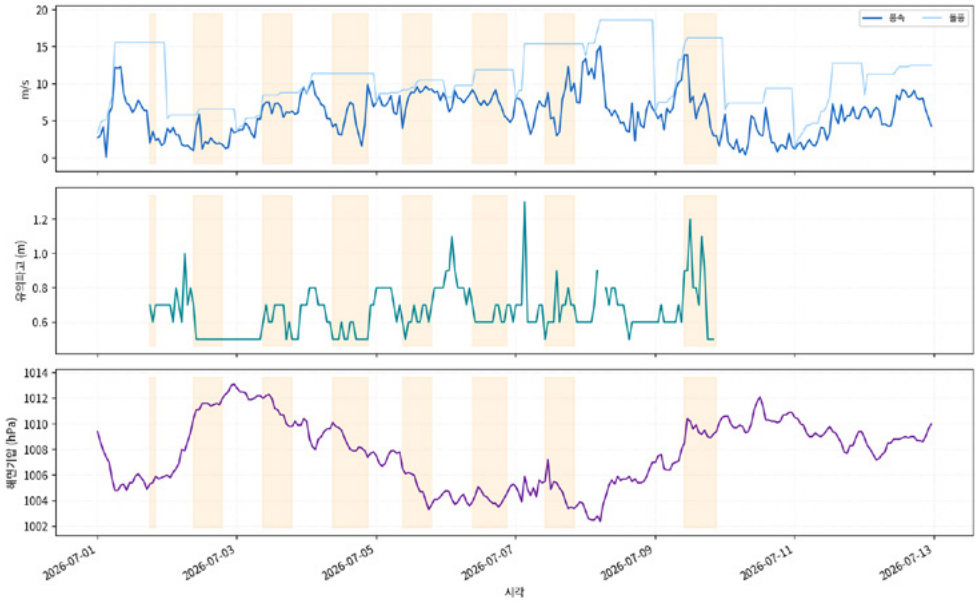


그림 45-3. 항적별 바람·파고·기압 비교

❻ 항해·정박 구간별 자료 활용

속력과 위치로 항해·정박 구간을 구분합니다.

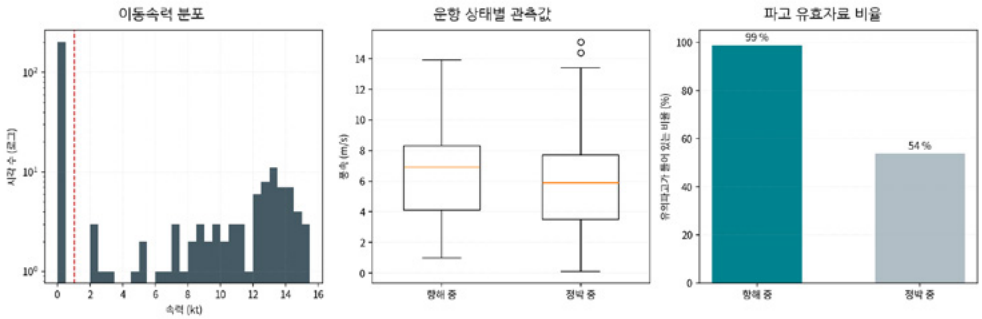


그림 45-4. 항해·정박 구간별 자료 활용

기상기후데이터 카탈로그

데이터 활용사례 50

기후지진통계

Climate and Earthquake Statistics



평년값

46. 기후평년값

지역별 기후 특성과 평년기간별 변화 비교

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS) > 2. 지상 평년값 조회

❶ 기후평년값 읽기

지역 차이와 지점별 기후 변화를 분석합니다.

```
https://apihub.kma.go.kr/api/typ01/url/sfc_norm1.php?tmst=2021&norm=M&stn=0&MM1=1&DD1=1
&MM2=12&DD2=31&authKey=...
# ST, STN, MM, TA, TA_MAX, TA_MIN, RN, ···, RD_01, RD_10
2021, 108, 8, 26.1, 30.0, 23.0, 348, ···, 14.9, 11.8
#  tmst : 1991(1961~1990) 2001(1971~2000) 2011(1981~2010) 2021(1991~2020)
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
STN_NAME = {}  # 표시명 사전. 공식 코드표로 보강; 미등록 코드는 그대로 표시
def get_norm(tmst=2021, norm="M", stn=0):
  response = requests.get(
    "https://apihub.kma.go.kr/api/typ01/url/sfc_norm1.php",
    params={
      "tmst": tmst,
      "norm": norm,
      "stn": stn,
      "MM1": 1,
      "DD1": 1,
      "MM2": 12,
      "DD2": 31,
      "authKey": API_KEY,
    },
    timeout=120,
  )
  if not response.ok:
    raise RuntimeError(f"평년값 HTTP {response.status_code}")
  response.encoding = "euc-kr"
  return parse_norm(response.text)
def save_norm_files(directory="normals", stn=0):
  target = Path(directory)
  target.mkdir(parents=True, exist_ok=True)
  for tmst, norm in [(2021, "Y"), (2021, "M"), (2011, "Y")]:
    get_norm(tmst, norm, stn).to_csv(
      target / f"norm_{tmst}_{norm}.csv", index=False, encoding="utf-8-sig"
    )
  return str(target)
```

save_norm_files()로 공식 연(Y)·월(M) 평년값 CSV를 생성한 뒤 load_pack("normals")로 읽습니다. 연평균기온·연강수량·연강수일수는 공식 연평년값을 사용합니다. 월자료는 연교차·여름강수비율 및 월별 분포에 사용합니다. 지점명 사전을 제공하지 않으면 지점번호를 표시합니다.

```
def parse_norm(text):
    """응답의 실제 열 이름을 사용. 월/일/연 형식 차이를 고정 위치로 추측하지 않는다."""
    columns, rows = (None, [])
    for line in text.splitlines():
        line = line.strip()
        if not line:
            continue
        if line.startswith("#"):
            candidate = line.lstrip("# ")
            if re.match("ST\\s*,\\s*STN", candidate):
                columns = [c.strip() for c in candidate.rstrip("=,").split(",")]
            continue
        if columns is None:
            raise ValueError("평년값 응답은 헤더 행부터 시작")
        fields = [c.strip() for c in line.rstrip("=,").split(",")]
        if len(fields) != len(columns):
            raise ValueError("평년값 행의 열 수 = 헤더 열 수")
        rows.append(dict(zip(columns, fields)))
    d = pd.DataFrame(rows, columns=columns or [])
    for c in d:
        d[c] = pd.to_numeric(d[c], errors="coerce")
    for c in d.columns.difference(["ST", "STN", "MM", "DD"]):
        d[c] = d[c].mask(d[c] <= -99)
    return d.rename(
        columns={"ST": "평년기간", "STN": "지점", "MM": "월", "DD": "일"}
    )
```

지점	지점명	연평균기온	연강수량	연강수일수	연교차	여름강수 비율(%)
100	대관령	7.1	1695	102	26.6	51.2
216	태백	9	1308	89	26.1	53
95	철원	10.3	1354	79	29.7	64
159	부산	15	1577	77	22.5	49.6
184	제주	16.2	1502	94	21.1	43.5
189	서귀포	16.9	1990	95	20	43.2

❷ 지점별 연평균기온 분포

지점별 12개월 평년기온을 평균해 지도에 표시합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def load_pack(path):
    """save_norm_files의 공식 연·월 CSV에서 분석 표를 생성한다."""
    directory = Path(path)
    a21 = pd.read_csv(directory / "norm_2021_Y.csv", encoding="utf-8-sig")
    m21 = pd.read_csv(directory / "norm_2021_M.csv", encoding="utf-8-sig")
    a11 = pd.read_csv(directory / "norm_2011_Y.csv", encoding="utf-8-sig")
    if a21["지점"].duplicated().any() or a11["지점"].duplicated().any():
        raise ValueError("연평년 지점 중복")
    a = a21[["지점", "TA", "RN", "RD_01"]].rename(
        columns={"TA": "연평균기온", "RN": "연강수량", "RD_01": "연강수일수"}
    )
    climate = m21.groupby("지점").agg(
        최난월기온=("TA", "max"), 최한월기온=("TA", "min")
    )
    summer = (
        m21[m21["월"].isin([6, 7, 8])]
        .groupby("지점")["RN"]
        .sum(min_count=3)
        .rename("여름강수")
    )
```

평년값

```
a = a.merge(climate, on="지점", how="left").merge(
  summer, on="지점", how="left"
)
a["연교차"] = a["최난월기온"] - a["최한월기온"]
a["여름강수비율(%)"] = a["여름강수"] / a["연강수량"].replace(0, np.nan) * 100
a["지점명"] = a["지점"].map(lambda x: STN_NAME.get(x, str(x)))
monthly = []
for station, g in m21.groupby("지점"):
  g = g.sort_values("월")
  if list(g["월"]) != list(range(1, 13)):
    continue
  monthly.append(
    dict(
      지점=station,
      **{
        c: g[c].tolist()
        for c in ["TA", "TA_MAX", "TA_MIN", "RN"]
        if c in g
      }
    )
  )
old = a11[["지점", "TA"]].rename(columns={"TA": "연평균기온_1981_2010"})
return (a, pd.DataFrame(monthly, columns=None if monthly else ["지점"]), old)
```

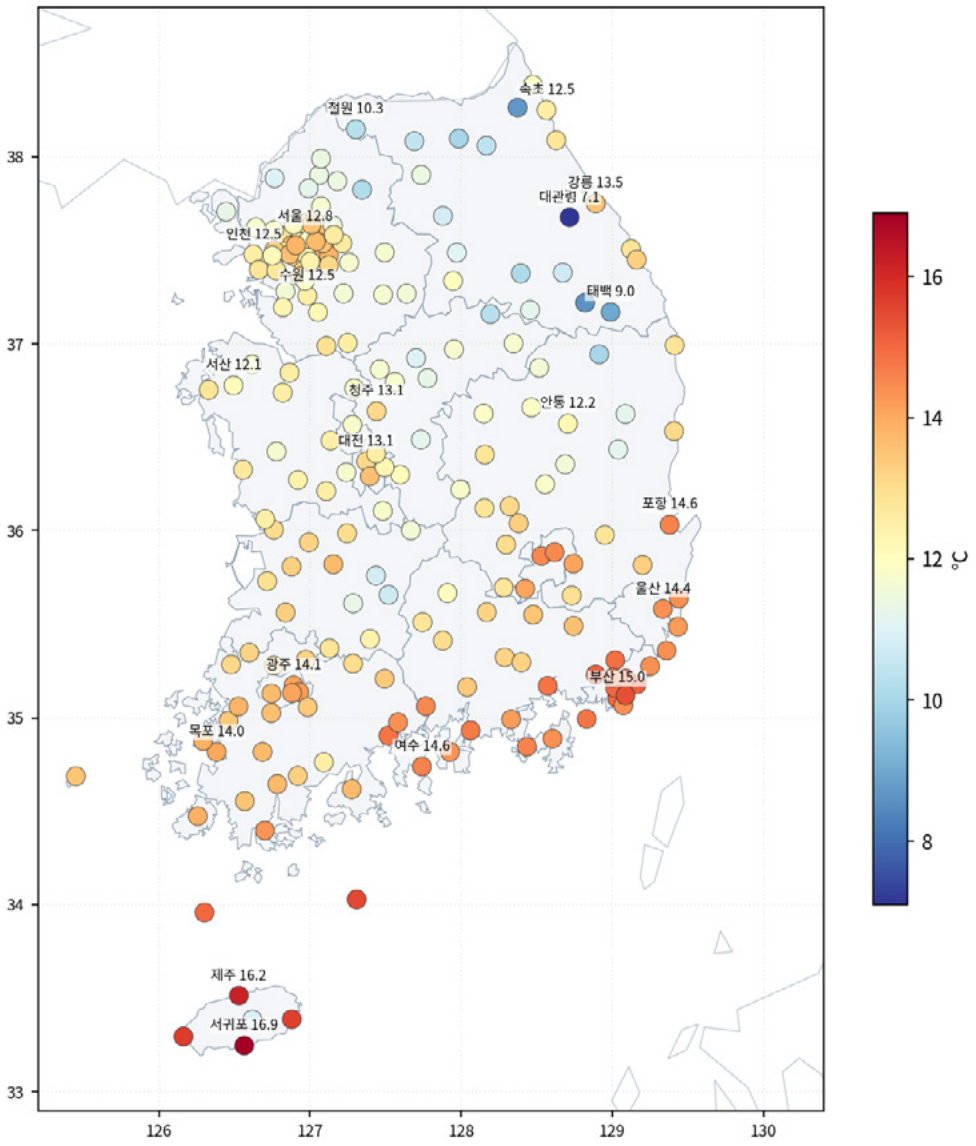


그림 46-1. 지점별 연평균기온 분포

❸ 연강수량과 여름철 강수 비중 비교

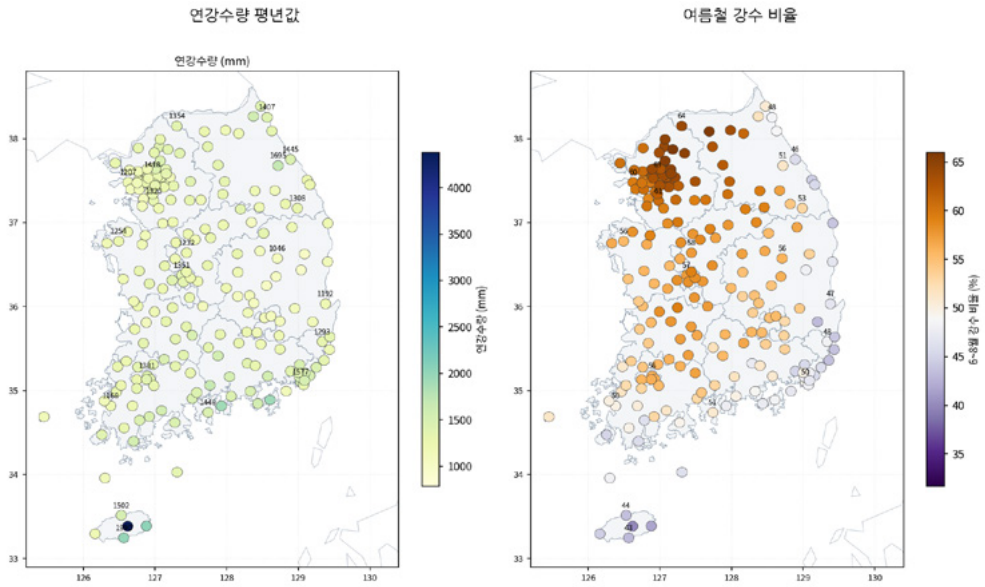


그림 46-2. 연강수량과 여름철 강수 비중 비교

❹ 해안·내륙의 기온 연교차 비교

지도와 산점도로 해안·내륙 차이를 비교합니다.

```
def range_fit(g):
    """기온 연교차와 위도·경도의 관계를 회귀로 비교합니다. 경도는 동서 위치의 대용변수이며 해안까지의 거리는 아닙니다."""

    a, b = np.polyfit(g["위도"], g["연교차"], 1)
    r = np.corrcoef(g["위도"], g["연교차"])[0, 1]
    return {
      "지점 수": len(g),
      "연교차 범위": f"{g['연교차'].min():.1f} ~ {g['연교차'].max():.1f} ℃",
      "가장 큰 곳": f"{g.loc[g['연교차'].idxmax(), '지점']}번 {g['연교차'].max():.1f} ℃",
      "가장 작은 곳": f"{g.loc[g['연교차'].idxmin(), '지점']}번 {g['연교차'].min():.1f} ℃",
      "위도 1도당": f"{a:+.2f} ℃",
      "상관계수 r": f"{r:.3f}",
    }
```

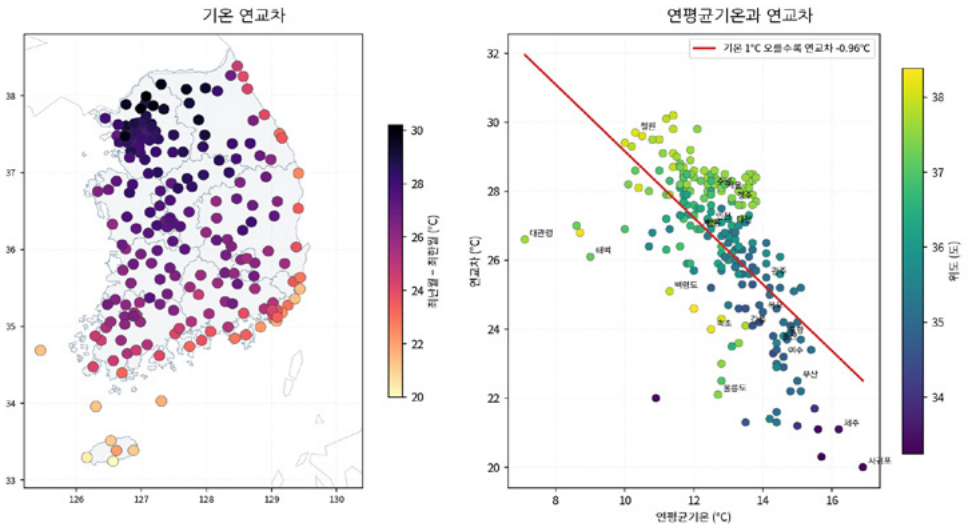


그림 46-3. 해안·내륙의 기온 연교차 비교

평년값

❺ 대표 지점의 월별 기온·강수량 비교

대표 지점의 월별 값을 비교해 기온뿐 아니라 강수의 계절 분포 차이도 확인합니다.

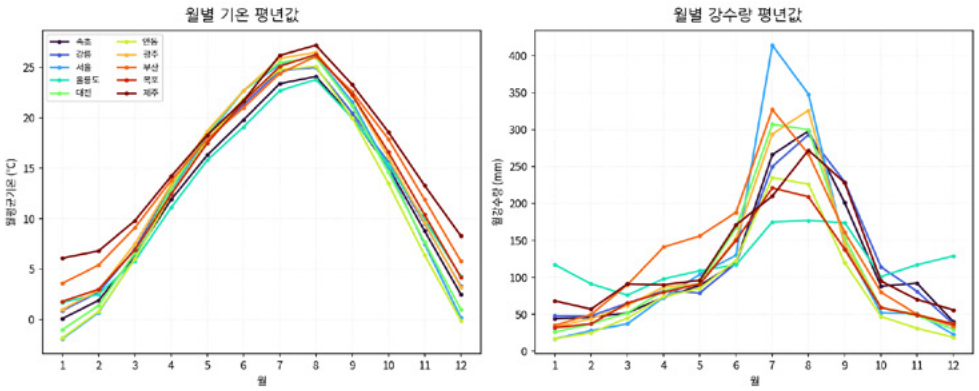


그림 46-4. 대표 지점의 월별 기온·강수량 비교

❻ 1981~2010년과 1991~2020년 평년값 비교

```
def diff_table(a, b):
    """같은 지점의 1981~2010년과 1991~2020년 평년값을 비교합니다. 두 기간은 20년이 겹치므로 차이를 단순한 10년 변화량으로 해석하지 않습니다."""

    m = a.merge(b, on="지점")
    m["차이"] = (m["연평균기온"] - m["연평균기온_1981_2010"]).round(2)
    m["지점명"] = m["지점"].map(lambda s: STN_NAME.get(s, ""))
    return m.sort_values("차이", ascending=False)
```

지점	지점명	연평균기온_1981_2010	연평균기온	차이
131	청주	12.5	13.1	0.6
119	수원	12	12.5	0.5
100	대관령	6.6	7.1	0.5
129	서산	11.9	12.1	0.2
115	울릉도	12.5	12.7	0.2
95	철원	10.2	10.3	0.1

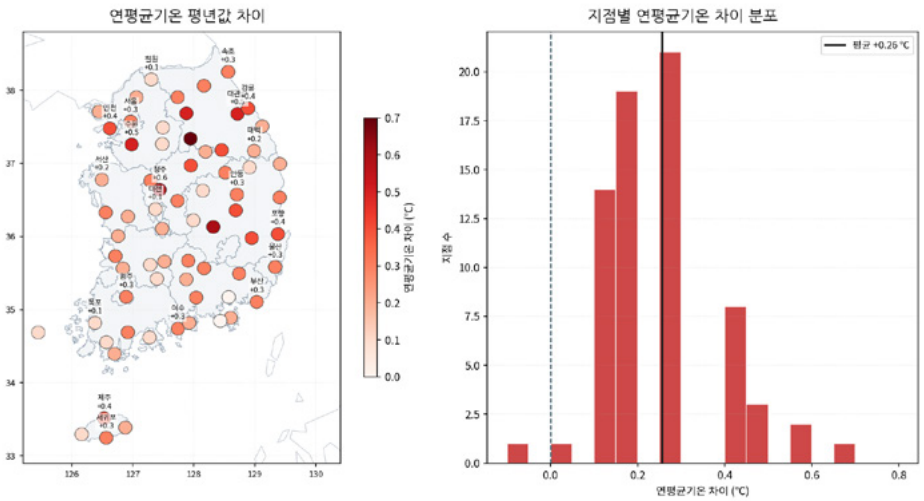


그림 46-5. 1981~2010년과 1991~2020년 평년값 비교

47. 강수일수

지역별 발생 빈도와 장기 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS)

❶ 일자료 조회와 강수일수 집계

전국 지점별 분포·추세·강도별 빈도를 비교합니다.

```
https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php?tm1=20240101&tm2=20241231&stn=0&disp=0&help=0&authKey=...
# YYMMDD STN  WS_AVG …  TA_AVG  TA_MAX …  TA_MIN …  RN_DAY …
20240101  90   1.0 …    2.7    6.1 …   0.9 …   -9.0 …
20240101 100   1.6 …   -2.6    1.1 …  -6.2 …    0.5 …
# 한 해치 = 약 3만 5천 줄 · 12 MB. 지점번호(STN)로 나눠 세면 된다.
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
DD_URL = "https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php"
COL = {"TM": 0, "STN": 1, "TA_AVG": 10, "TA_MAX": 11, "TA_MIN": 13, "RN_DAY": 38, "SD_NEW": 47} #
help=1 헤더와 대조
def get_annual_summary(year):
  url = "https://apihub.kma.go.kr/api/typ02/openApi/SfcYearlyInfoService/getYearSumry"
  rows = []
  for page in range(1, 1000):
    response = requests.get(
      url,
      params={
        "year": year,
        "dataType": "JSON",
        "pageNo": page,
        "numOfRows": 1000,
        "authKey": API_KEY,
      },
      timeout=60,
    )
    response.raise_for_status()
    payload = response.json().get("response", {})
    code = str(payload.get("header", {}).get("resultCode", ""))
    if code == "03":
      break
    if code != "00":
      raise RuntimeError(f"resultCode {code}")
    body = payload.get("body", {})
```

통계분석

```
    def flatten(value):
      if isinstance(value, dict):
        if "stn_id" in value:
          return [value]
        return [row for child in value.values() for row in flatten(child)]
      if isinstance(value, list):
        return [row for child in value for row in flatten(child)]

      return []

    batch = flatten(body.get("items", {}))
    rows.extend(batch)

    if not batch or page * 1000 >= int(body.get("totalCount", len(rows))):
      break


  def number(row, key, scale=1):
    value = pd.to_numeric(row.get(key), errors="coerce")
    return value * scale if pd.notna(value) and value >= 0 else np.nan

  return pd.DataFrame(
    [
      {
        "지점": int(r["stn_id"]),
        "지점명": r.get("stn_ko", str(r["stn_id"])),
        "연도": year,
        "강수일수": number(r, "va_lst_18"),
        "강수일수1mm": number(r, "va_lst_17"),
        "연강수량": number(r, "va_lst_11", 0.1),
      }
      for r in rows
    ]
  )
```

강수일수는 공식 연요약 va_lst_18(0.1mm 이상), 1mm 이상일수는 va_lst_17, 연강수량은 va_lst_11×0.1mm를 사용합니다. 연요약은 10mm·80mm 이상일수를 제공하지 않으므로 임의 값으로 채우지 않습니다. 연요약에 없는 요소는 별도 통계 API 또는 의미가 확인된 관측자료를 사용합니다.

10mm·80mm 이상일수는 일자료의 양수 강수량으로 셉니다. 연간 일자료가 365일(윤년366일) 확보되고, 0.1mm 이상일수·양수 강수량 합이 공식 연요약과 일치한 경우에만 수록합니다. 음수 강수량을 0으로 바꾸지 않습니다. 불일치한 연도는 추가 지표를 결측으로 남기며 경향 분석에서 유효 연수를 확인합니다.

```
def get_daily(year, stn=0, retry=3):
  """한 해치 전국 일자료를 한 번에 받는다. stn=0 이면 그날 자료가 있는 모든 지점이 함께 반환됩니다."""

  params = {
    "tm1": f"{year}0101",
    "tm2": f"{year}1231",
    "stn": stn,
    "disp": 0,
    "help": 0,
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(DD_URL, params=params, timeout=180)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      return r.text
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{year} 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(2 * (i + 1))
```

```
def count_year(text):
    """ASOS 일자료를 지점별 집계하고 요소별 유효 관측일수를 기록한다."""
    out = {}
    seen = set()
    for line in text.splitlines():
        if not line.strip() or line.lstrip().startswith("#"):
            continue
        fields = line.split()
        if len(fields) < 50:
            raise ValueError("ASOS 일자료 행은 50개 열 이상")
        station = int(fields[1])
        day = fields[0]
        if (station, day) in seen:
            raise ValueError("지점·날짜 중복 자료")
        seen.add((station, day))
        a = out.setdefault(
            station,
            dict(
                n=0,
                ntmax=0,
                ntmin=0,
                nr=0,
                ns=0,
                rd=0,
                r10=0,
                r80=0,
                hw=0,
                tn=0,
                ic=0,
                sn=0,
                rn=0.0,
                ta=0.0,
            ),
        )
        average, maximum, minimum = map(
            float, [fields[10], fields[11], fields[13]]
        )
        rain, snow = (float(fields[38]), float(fields[47]))
        if np.isfinite(average) and -80 < average < 60:
            a["ta"] += average
            a["n"] += 1
        if np.isfinite(maximum) and -80 < maximum < 60:
            a["ntmax"] += 1
            a["hw"] += maximum >= 33
            a["ic"] += maximum < 0
        if np.isfinite(minimum) and -80 < minimum < 60:
            a["ntmin"] += 1
            a["tn"] += minimum >= 25
        if np.isfinite(rain) and rain >= 0:
            a["nr"] += 1
            a["rn"] += rain
            a["rd"] += rain >= 0.1
            a["r10"] += rain >= 10
            a["r80"] += rain >= 80
        if np.isfinite(snow) and snow >= 0:
            a["ns"] += 1
            a["sn"] += snow >= 0.1
    return out
```

실행 예는 서울(108) 2020~2025년 자료입니다. 아래 전국 30년 도표와 조회조건이 다릅니다. 전국 비교에는 stn=0과 동일 분석기간을 지정하고, 분석기간 전체에 관측이 있는 공통지점과 요소별 유효일수를 적용합니다.

❷ 30년 평균 강수일수 분포

지점별 30년 평균 강수일수를 지도에 표시합니다.

통계분석

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def collect_annual(years, stn=108, min_valid=350):
  import calendar


  frames = []
  for year in years:
    if year >= datetime.now(KST).year:
      raise ValueError("완료된 연도를 지정")
    official = get_annual_summary(year)
    if official.empty:
      continue
    if stn:
      official = official[official["지점"] == stn]
    daily_text = get_daily(year, stn)
    daily = count_year(daily_text)
    official = official.copy()
    official["강수일수10mm"] = np.nan
    official["강수일수80mm"] = np.nan
    official["유효일수"] = np.nan
    official["일자료연요약일치"] = False
    expected_days = 366 if calendar.isleap(year) else 365
    for index, row in official.iterrows():
      observed = daily.get(int(row["지점"]))
      if observed is None:
        continue
      official.loc[index, "유효일수"] = observed["n"]
      consistent = (
        observed["n"] == expected_days
        and observed["rd"] == row["강수일수"]
        and np.isclose(observed["rn"], row["연강수량"], rtol=0, atol=0.05)
      )
      if consistent:
        official.loc[index, "강수일수10mm"] = observed["r10"]
        official.loc[index, "강수일수80mm"] = observed["r80"]
        official.loc[index, "일자료연요약일치"] = True
    frames.append(official)
  return (
    pd.concat(frames, ignore_index=True)
    if frames
    else pd.DataFrame(
      columns=[
        "지점",
        "지점명",
        "연도",
        "유효일수",
        "강수일수",
        "강수일수1mm",
        "연강수량",
        "강수일수10mm",
        "강수일수80mm",
        "일자료연요약일치",
      ]
    )
  )

# 실행 예: 한 지점으로 확인 후 전국 조회(stn=0)로 확장
# 실행 예: d = collect_annual(range(2020, 2026), stn=108)
# 실행 예: d.to_csv("annual_statistics.csv", index=False, encoding="utf-8-sig")
def load_pack(path):
  """collect_annual 결과 CSV 읽기. 구 CSV 형식 사용하지 않음."""
  d = pd.read_csv(path, encoding="utf-8-sig", dtype={"지점": int, "연도": int})
  required = {"지점", "지점명", "연도", "유효일수"}
  if not required.issubset(d.columns):
    raise ValueError("collect_annual 결과 CSV를 입력")
  return d
def mean_by_stn(d, cols):
  """30년을 지점마다 평균한다. 연수가 다르면 평균도 다르므로 연수를 남긴다."""
  g = d.groupby(["지점", "지점명"])[cols].mean().round(1).reset_index()
  g["연수"] = d.groupby(["지점", "지점명"]).size().values
  return g
```

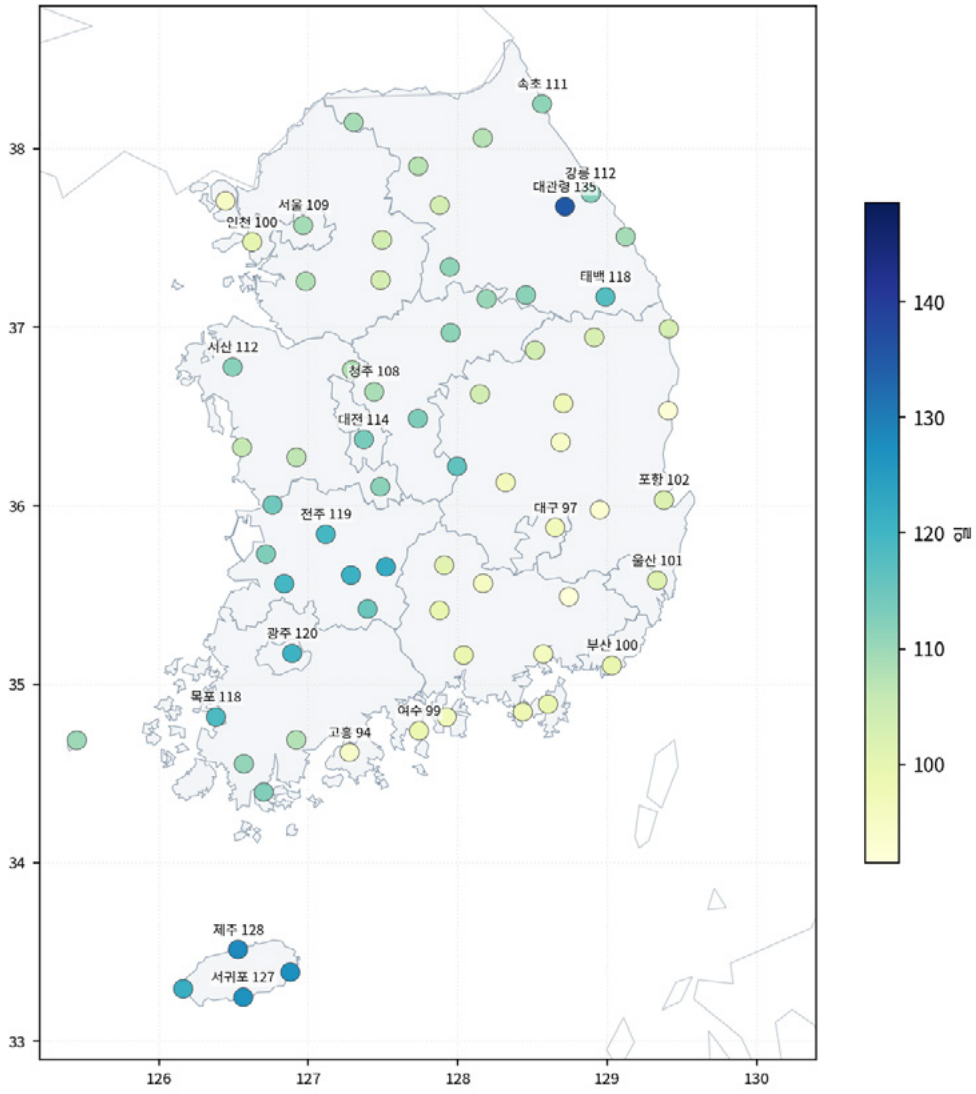


그림 47-1. 30년 평균 강수일수 분포

❸ 강수량 기준별 발생일수 비교

```
def rain_mix(g):
  """비가 온 날을 세기별로 나눈다. 같은 강수일수라도 속이 다르다."""
  d = g.copy()
  d["약한비"] = (d["강수일수"] - d["강수일수10mm"]).round(1)
  d["보통비"] = (d["강수일수10mm"] - d["강수일수80mm"]).round(1)
  d["큰비"] = d["강수일수80mm"].round(1)
  d["강수강도"] = (d["연강수량"] / d["강수일수"]).round(1)
  return d
```

지점명	강수일수	약한비	보통비	큰비	연강수량	강수강도
울릉도	148.6	105.6	40.8	2.2	1541	10.4
대관령	135.2	96.4	36	2.8	1575.7	11.7
제주	128.2	90.3	35.1	2.8	1492	11.6
군산	114.6	82.1	30.4	2.1	1291	11.3
대전	113.6	77.7	33.4	2.5	1388.7	12.2
보은	113	77.6	33.4	2	1345	11.9

통계분석

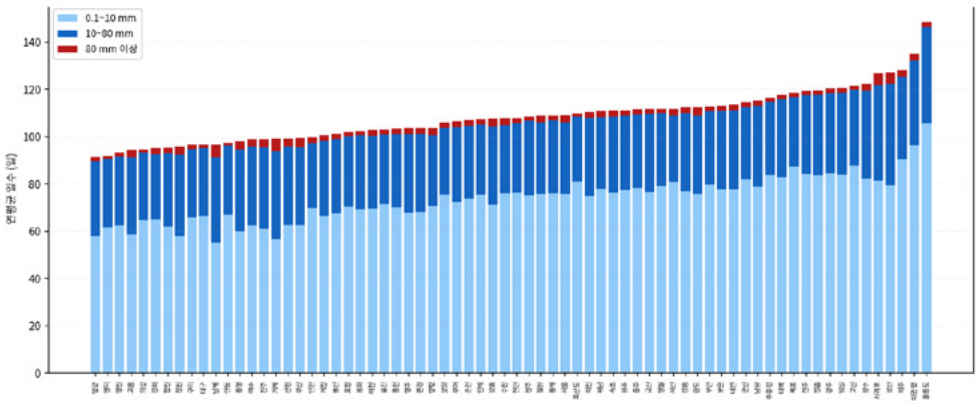


그림 47-2. 강수량 기준별 발생일수 비교

❹ 30년간 전국·지점별 강수일수 추세

연도별 전국 평균으로 전체 추세를, 지점별 기울기 지도로 지역 차이를 확인합니다.

```
def nation_year(d, cols):
    """해마다 전국 지점을 평균한다. 지점 수가 해마다 조금씩 달라지므로 함께 남긴다."""
    g = d.groupby("연도")[cols].mean().round(2)
    g["지점수"] = d.groupby("연도").size()
    return g.reset_index()
def slope_by_stn(d, col):
    """지점마다 30년 기울기를 잰다. 단위는 '10년당 며칠'."""
    rows = []
    for (stn, name), g in d.groupby(["지점", "지점명"]):
        if len(g) < 25:
            continue
        a = np.polyfit(g["연도"], g[col], 1)[0]
        rows.append(
            {
                "지점": stn,
                "지점명": name,
                "10년당": round(a * 10, 2),
                "평균": round(g[col].mean(), 1),
            }
        )
    return pd.DataFrame(
        rows, columns=None if rows else ["지점", "지점명", "10년당", "평균"]
    ).sort_values("10년당", ascending=False)
```

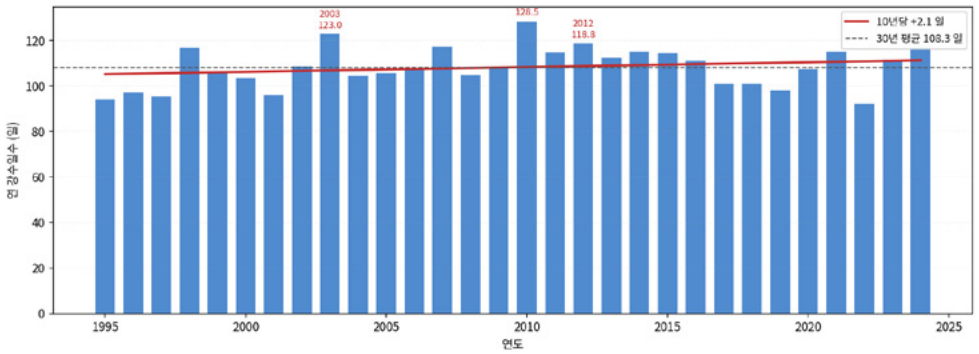


그림 47-3. 30년간 전국·지점별 강수일수 추세

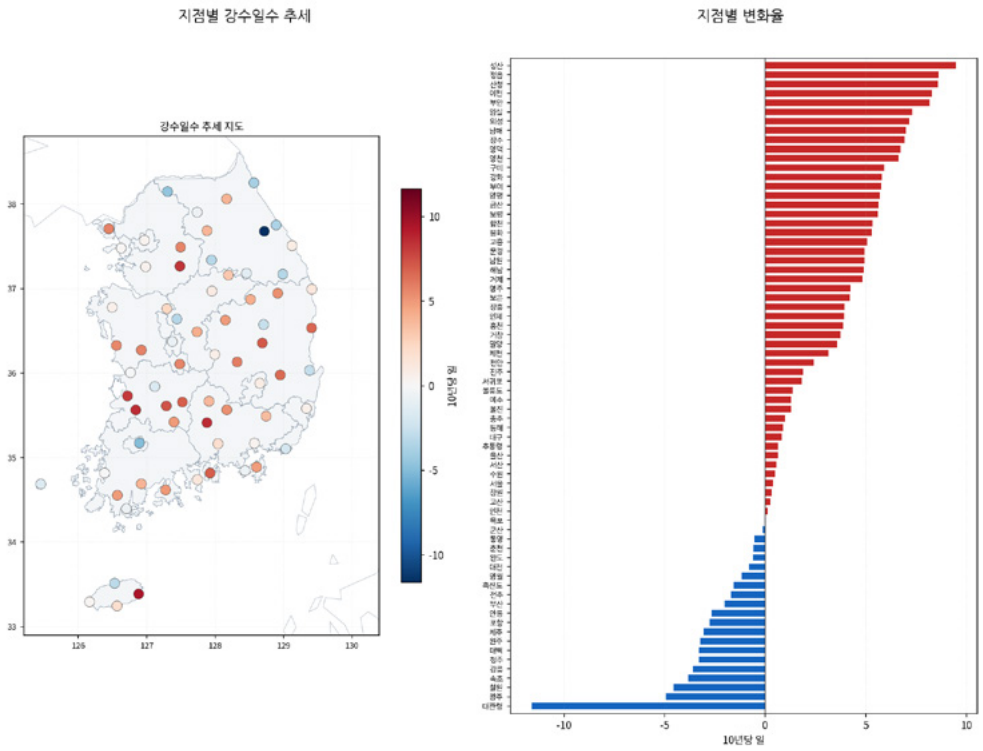


그림 47-4. 30년간 전국·지점별 강수일수 추세

❺ 강수일수와 강수강도 변화 비교

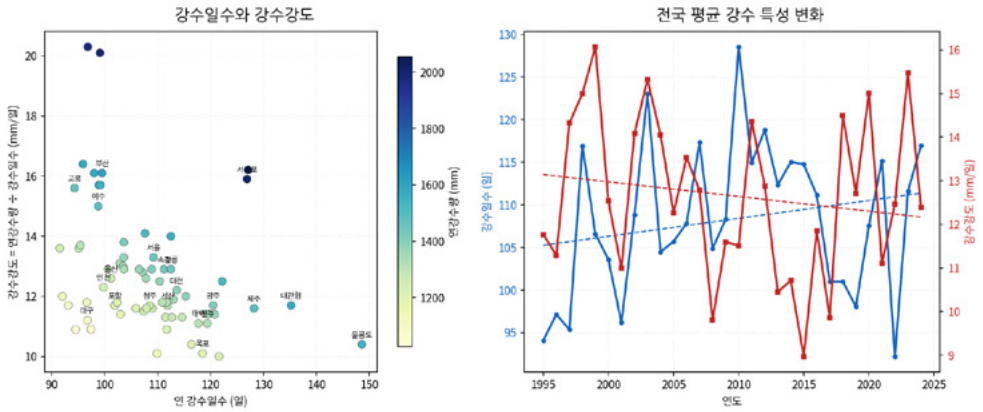


그림 47-5. 강수일수와 강수강도 변화 비교

통계분석

48. 폭염일수

지역별 발생 빈도와 장기 변화 분석

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS)

❶ 일자료 조회와 폭염일수 집계

```
https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php?tm1=20180101&tm2=20181231&stn=0&disp=0&help=0&authKey=...
# YYMMDD STN ··· TA_AVG TA_MAX TA_MAX_TM TA_MIN TA_MIN_TM ···
20180801 108 ···  33.6  39.6    1510  27.8    0530 ···
#  TA_MAX ≥ 33 이면 폭염일. TA_MIN ≥ 25 는 공식 열대야가 아닌 일최저 고온 참고지표입니다.
#  -9 · -99 는 값이 없다는 뜻이므로 세지 않는다.
```

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
DD_URL = "https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php"
COL = {"TM": 0, "STN": 1, "TA_AVG": 10, "TA_MAX": 11, "TA_MIN": 13, "RN_DAY": 38, "SD_NEW": 47} # help=1 헤더와 대조
def get_annual_summary(year):
  url = "https://apihub.kma.go.kr/api/typ02/openApi/SfcYearlyInfoService/getYearSumry"
  rows = []
  for page in range(1, 1000):
    response = requests.get(
      url,
      params={
        "year": year,
        "dataType": "JSON",
        "pageNo": page,
        "numOfRows": 1000,
        "authKey": API_KEY,
      },
      timeout=60,
    )
    response.raise_for_status()
    payload = response.json().get("response", {})
    code = str(payload.get("header", {}).get("resultCode", ""))
    if code == "03":
      break
    if code != "00":
      raise RuntimeError(f"resultCode {code}")
    body = payload.get("body", {})


    def flatten(value):
      if isinstance(value, dict):
        if "stn_id" in value:
```

```
            return [value]
          return [row for child in value.values() for row in flatten(child)]
        if isinstance(value, list):
          return [row for child in value for row in flatten(child)]

        return []

      batch = flatten(body.get("items", {}))
      rows.extend(batch)

      if not batch or page * 1000 >= int(body.get("totalCount", len(rows))):
        break


    def number(row, key, scale=1):
      value = pd.to_numeric(row.get(key), errors="coerce")
      return value * scale if pd.notna(value) and value >= 0 else np.nan

    return pd.DataFrame(
      [
        {
          "지점": int(r["stn_id"]),
          "지점명": r.get("stn_ko", str(r["stn_id"])),
          "연도": year,
          "강수일수": number(r, "va_lst_18"),
          "강수일수1mm": number(r, "va_lst_17"),
          "연강수량": number(r, "va_lst_11", 0.1),
        }
        for r in rows
      ]
    )
def get_daily(year, stn=0, retry=3):
    """한 해치 전국 일자료를 한 번에 받는다. stn=0 이면 그날 자료가 있는 모든 지점이 함께 반환됩니다."""

    params = {
      "tm1": f"{year}0101",
      "tm2": f"{year}1231",
      "stn": stn,
      "disp": 0,
      "help": 0,
      "authKey": API_KEY,
    }
    for i in range(retry):
      try:
        r = requests.get(DD_URL, params=params, timeout=180)
        if not r.ok:
          raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
        return r.text
      except Exception as exc:
        if i == retry - 1:
          raise
        log(f"{year} 재시도 {i + 1}: {type(exc).__name__}")
        time.sleep(2 * (i + 1))
def count_year(text):
    """ASOS 일자료를 지점별 집계하고 요소별 유효 관측일수를 기록한다."""
    out = {}
    seen = set()
    for line in text.splitlines():
      if not line.strip() or line.lstrip().startswith("#"):
        continue
      fields = line.split()
      if len(fields) < 50:
        raise ValueError("ASOS 일자료 행은 50개 열 이상")
      station = int(fields[1])
      day = fields[0]
      if (station, day) in seen:
        raise ValueError("지점·날짜 중복 자료")
      seen.add((station, day))
      a = out.setdefault(
        station,
        dict(
          n=0,
```

통계분석

```
            ntmax=0,
            ntmin=0,
            nr=0,
            ns=0,
            rd=0,
            r10=0,
            r80=0,
            hw=0,
            tn=0,
            ic=0,
            sn=0,
            rn=0.0,
            ta=0.0,
        ),
    )
    average, maximum, minimum = map(
        float, [fields[10], fields[11], fields[13]]
    )
    rain, snow = (float(fields[38]), float(fields[47]))
    if np.isfinite(average) and -80 < average < 60:
        a["ta"] += average
        a["n"] += 1
    if np.isfinite(maximum) and -80 < maximum < 60:
        a["ntmax"] += 1
        a["hw"] += maximum >= 33
        a["ic"] += maximum < 0
    if np.isfinite(minimum) and -80 < minimum < 60:
        a["ntmin"] += 1
        a["tn"] += minimum >= 25
    if np.isfinite(rain) and rain >= 0:
        a["nr"] += 1
        a["rn"] += rain
        a["rd"] += rain >= 0.1
        a["r10"] += rain >= 10
        a["r80"] += rain >= 80
    if np.isfinite(snow) and snow >= 0:
        a["ns"] += 1
        a["sn"] += snow >= 0.1
    return out
```

실행 예는 서울(108) 2020~2025년 자료입니다. 아래 전국 30년 도표와 조회조건이 다릅니다. 전국 비교에는 stn=0과 동일 분석기간을 지정하고, 분석기간 전체에 관측이 있는 공통지점과 요소별 유효일수를 적용합니다.

❷ 30년 평균 폭염일수 분포

지점별 30년 평균 폭염일수를 지도에 표시합니다.

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def collect_annual(years, stn=108, min_valid=350):
    """조회→집계→CSV용 표. 전국 비교는 stn=0, 동일 지점 구성은 별도 선택."""
    rows = []
    for year in years:
        if year >= datetime.now(KST).year:
            raise ValueError("완료된 연도만 연간 통계에 사용")
        for station, a in count_year(get_daily(year, stn)).items():
            row = {
                "지점": station,
                "지점명": str(station),
                "연도": year,
                "유효일수": a["n"],
                "강수유효일수": a["nr"],
                "최고기온유효일수": a["ntmax"],
                "최저기온유효일수": a["ntmin"],
```

```
            "강수일수": a["rd"],
            "강수일수10mm": a["r10"],
            "강수일수80mm": a["r80"],
            "폭염일수": a["hw"],
            "일최저25℃이상일수(참고)": a["tn"],
            "결빙일수": a["ic"],
            "신적설일수": a["sn"],
            "연강수량": a["rn"],
            "연평균기온": a["ta"] / a["n"] if a["n"] else np.nan,
        }
        for key in ["강수일수", "강수일수10mm", "강수일수80mm", "연강수량"]:
            if a["nr"] < min_valid:
                row[key] = np.nan
        for key in ["폭염일수", "결빙일수"]:
            if a["ntmax"] < min_valid:
                row[key] = np.nan
        if a["ntmin"] < min_valid:
            row["일최저25℃이상일수(참고)"] = np.nan
        if a["ns"] < min_valid:
            row["신적설일수"] = np.nan
        if a["n"] < min_valid:
            row["연평균기온"] = np.nan
        rows.append(row)
    time.sleep(0.3)
  return pd.DataFrame(rows)
# 실행 예: 한 지점으로 확인 후 전국 조회(stn=0)로 확장
# 실행 예: d = collect_annual(range(2020, 2026), stn=108)
# 실행 예: d.to_csv("annual_statistics.csv", index=False, encoding="utf-8-sig")
def load_pack(path):
  """collect_annual 결과 CSV 읽기. 구 CSV 형식 사용하지 않음."""
  d = pd.read_csv(path, encoding="utf-8-sig", dtype={"지점": int, "연도": int})
  required = {"지점", "지점명", "연도", "유효일수"}
  if not required.issubset(d.columns):
    raise ValueError("collect_annual 결과 CSV를 입력")
  return d
def mean_by_stn(d, cols):
  """30년을 지점마다 평균한다. 연수가 다르면 평균도 다르므로 연수를 남긴다."""
  g = d.groupby(["지점", "지점명"])[cols].mean().round(1).reset_index()
  g["연수"] = d.groupby(["지점", "지점명"]).size().values
  return g
```

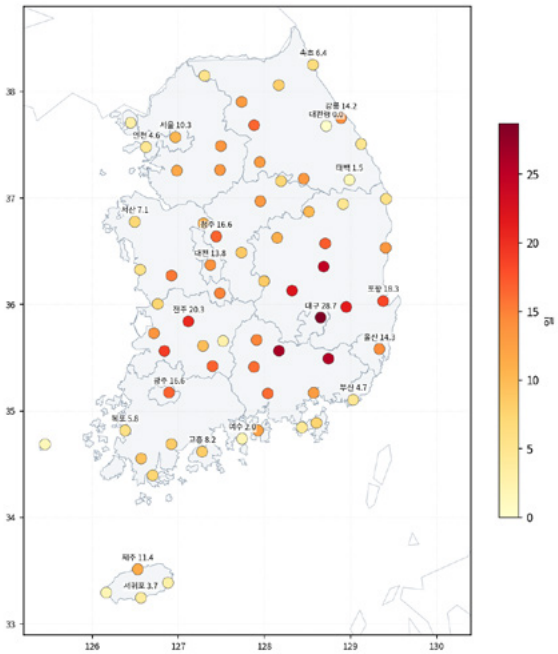


그림 48-1. 30년 평균 폭염일수 분포

통계분석

❸ 전국 평균 폭염일수의 연도별 변화

```
def nation_year(d, cols):
    """해마다 전국 지점을 평균한다. 지점 수가 해마다 조금씩 달라지므로 함께 남긴다."""
    g = d.groupby("연도")[cols].mean().round(2)
    g["지점수"] = d.groupby("연도").size()
    return g.reset_index()
def year_rank(ny, col, n=8):
    """더웠던 해를 순위로 세운다."""
    t = ny.nlargest(n, col)[["연도", col, "지점수"]].copy()
    t["연도"] = t["연도"].astype(int)
    t = t.rename(columns={col: f"{col}(전국평균)"})
    return t.reset_index(drop=True)
```

연도	폭염일수(전국평균)	지점수
2024	28.91	70
2018	28.69	70
2016	20.33	70
2013	16.01	70
1996	14.58	69
2004	13.96	70
2023	13.36	70
2012	12.91	70

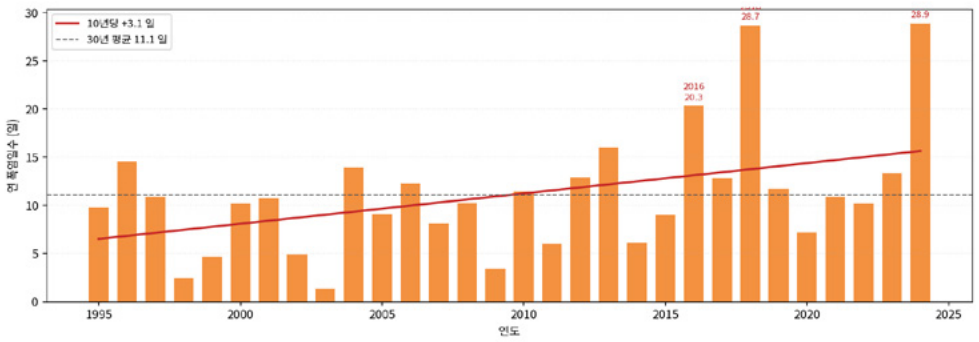


그림 48-2. 전국 평균 폭염일수의 연도별 변화

❹ 지점별 폭염일수 증가·감소 추세

지점별 30년 기울기를 지도에 표시해 증가·감소 지역을 비교합니다.

```
def slope_by_stn(d, col):
    """지점마다 30년 기울기를 잰다. 단위는 '10년당 며칠'."""
    rows = []
    for (stn, name), g in d.groupby(["지점", "지점명"]):
        if len(g) < 25:
            continue
        a = np.polyfit(g["연도"], g[col], 1)[0]
        rows.append(
            {
                "지점": stn,
                "지점명": name,
                "10년당": round(a * 10, 2),
                "평균": round(g[col].mean(), 1),
            }
        )
    return pd.DataFrame(
        rows, columns=None if rows else ["지점", "지점명", "10년당", "평균"]
    ).sort_values("10년당", ascending=False)
```

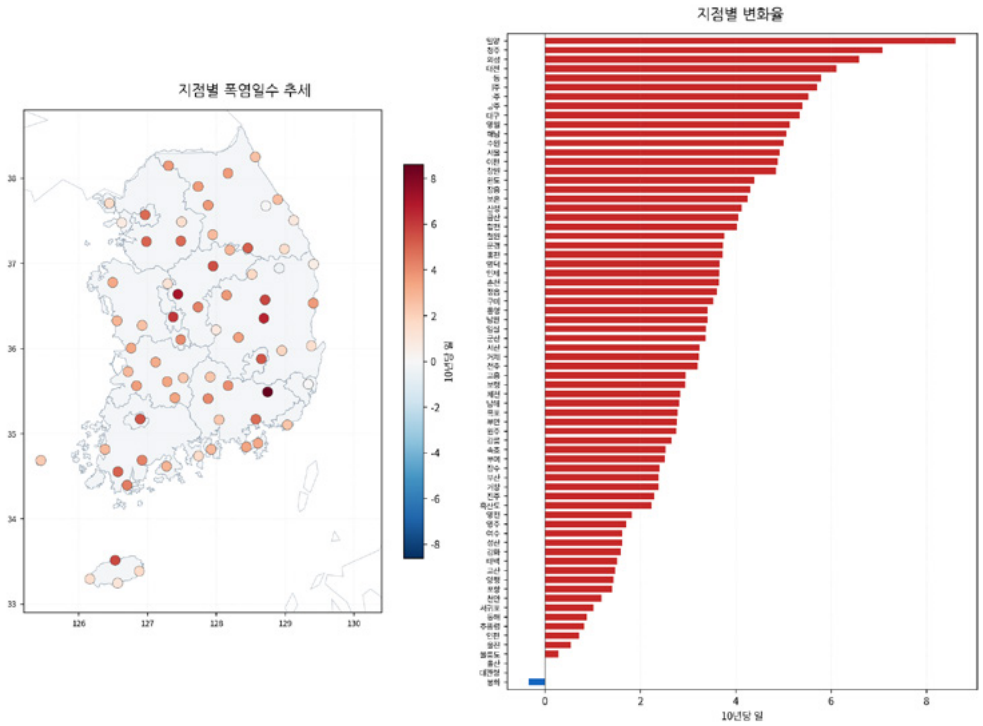


그림 48-3. 지점별 폭염일수 증가·감소 추세

지점	지점명	10년당	평균
288	밀양	8.61	25.6
131	청주	7.08	16.6
278	의성	6.58	24.6
133	대전	6.1	13.8
136	안동	5.78	17.1
184	제주	5.7	11.4
115	울릉도	0.28	0.7
100	대관령	0	0
152	울산	-0	14.3
271	봉화	-0.34	4.8

❺ 2024년 폭염일수와 30년 평균 비교

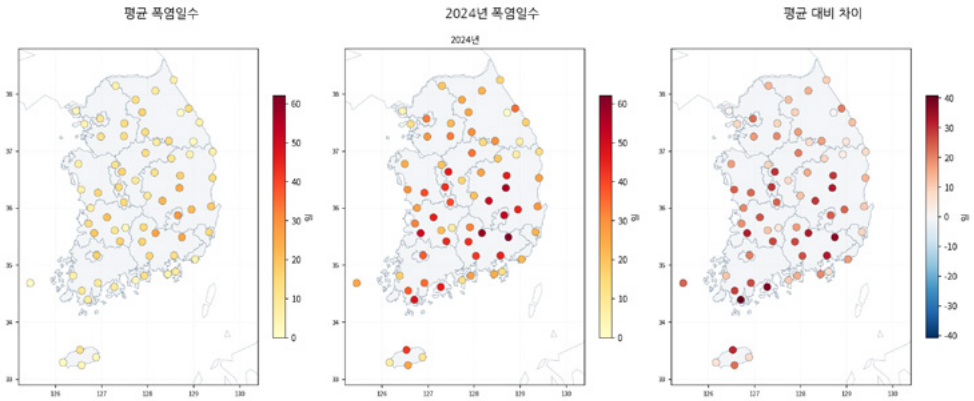


그림 48-4. 2024년 폭염일수와 30년 평균 비교

통계분석

❻ 대표 지점의 7년 이동평균 비교

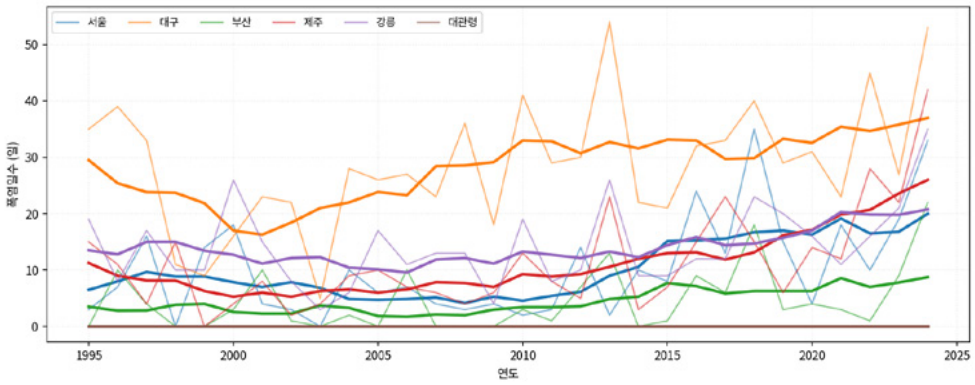


그림 48-5. 대표 지점의 7년 이동평균 비교

49.
열대야일수

일최저기온 참고지표 비교

데이터 활용 경로(API)

기상청 API허브 > 지상관측 > 종관기상관측(ASOS)

❶ 일자료 조회와 일최저 25℃ 이상일수 참고지표

※ 정확한 열대야 통계에는 이 일자료 계산을 사용하지 않습니다. 공식 열대야 통계 또는 18:01 ~익일 09:00의 밤최저기온을 산출할 수 있는 자료를 사용해야 합니다.

```
https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php?tm1=20240101&tm2=20241231&stn=0&disp=0&help=0&authKey=...
# YYMMDD STN ··· TA_AVG TA_MAX TA_MAX_TM TA_MIN TA_MIN_TM ···
20240812 159 ···  29.4   32.1     1436   27.3     0612 ···
#  주의: TA_MIN ≥ 25는 공식 열대야가 아님. 열대야는 당일 18:01~익일 09:00 밤최저기온 ≥ 25℃인 날이다.
```

이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)

```
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
DD_URL = "https://apihub.kma.go.kr/api/typ01/url/kma_sfcdd3.php"
COL = {"TM": 0, "STN": 1, "TA_AVG": 10, "TA_MAX": 11, "TA_MIN": 13, "RN_DAY": 38, "SD_NEW": 47} # help=1 헤더와 대조
def get_annual_summary(year):
    url = "https://apihub.kma.go.kr/api/typ02/openApi/SfcYearlyInfoService/getYearSumry"
    rows = []
    for page in range(1, 1000):
        response = requests.get(
            url,
            params={
                "year": year,
                "dataType": "JSON",
                "pageNo": page,
                "numOfRows": 1000,
                "authKey": API_KEY,
            },
            timeout=60,
        )
        response.raise_for_status()
        payload = response.json().get("response", {})
        code = str(payload.get("header", {}).get("resultCode", ""))
        if code == "03":
            break
        if code != "00":
            raise RuntimeError(f"resultCode {code}")
        body = payload.get("body", {})
```

통계분석

```
    def flatten(value):
      if isinstance(value, dict):
        if "stn_id" in value:
          return [value]
        return [row for child in value.values() for row in flatten(child)]
      if isinstance(value, list):
        return [row for child in value for row in flatten(child)]

      return []

    batch = flatten(body.get("items", {}))
    rows.extend(batch)

    if not batch or page * 1000 >= int(body.get("totalCount", len(rows))):
      break


  def number(row, key, scale=1):
    value = pd.to_numeric(row.get(key), errors="coerce")
    return value * scale if pd.notna(value) and value >= 0 else np.nan

  return pd.DataFrame(
    [
      {
        "지점": int(r["stn_id"]),
        "지점명": r.get("stn_ko", str(r["stn_id"])),
        "연도": year,
        "강수일수": number(r, "va_lst_18"),
        "강수일수1mm": number(r, "va_lst_17"),
        "연강수량": number(r, "va_lst_11", 0.1),
      }
      for r in rows
    ]
  )
def get_daily(year, stn=0, retry=3):
  """한 해치 전국 일자료를 한 번에 받는다. stn=0 이면 그날 자료가 있는 모든 지점이 함께 반환됩니다."""

  params = {
    "tm1": f"{year}0101",
    "tm2": f"{year}1231",
    "stn": stn,
    "disp": 0,
    "help": 0,
    "authKey": API_KEY,
  }
  for i in range(retry):
    try:
      r = requests.get(DD_URL, params=params, timeout=180)
      if not r.ok:
        raise RuntimeError(f"HTTP {r.status_code}: 권한·접속정책 확인")
      return r.text
    except Exception as exc:
      if i == retry - 1:
        raise
      log(f"{year} 재시도 {i + 1}: {type(exc).__name__}")
      time.sleep(2 * (i + 1))
def count_year(text):
  """ASOS 일자료를 지점별 집계하고 요소별 유효 관측일수를 기록한다."""
  out = {}
  seen = set()
  for line in text.splitlines():
    if not line.strip() or line.lstrip().startswith("#"):
      continue
    fields = line.split()
    if len(fields) < 50:
      raise ValueError("ASOS 일자료 행은 50개 열 이상")
    station = int(fields[1])
    day = fields[0]
    if (station, day) in seen:
      raise ValueError("지점·날짜 중복 자료")
    seen.add((station, day))
    a = out.setdefault(
      station,
```

```
        dict(
            n=0,
            ntmax=0,
            ntmin=0,
            nr=0,
            ns=0,
            rd=0,
            r10=0,
            r80=0,
            hw=0,
            tn=0,
            ic=0,
            sn=0,
            rn=0.0,
            ta=0.0,
        ),
    )
    average, maximum, minimum = map(
        float, [fields[10], fields[11], fields[13]]
    )
    rain, snow = (float(fields[38]), float(fields[47]))
    if np.isfinite(average) and -80 < average < 60:
        a["ta"] += average
        a["n"] += 1
    if np.isfinite(maximum) and -80 < maximum < 60:
        a["ntmax"] += 1
        a["hw"] += maximum >= 33
        a["ic"] += maximum < 0
    if np.isfinite(minimum) and -80 < minimum < 60:
        a["ntmin"] += 1
        a["tn"] += minimum >= 25
    if np.isfinite(rain) and rain >= 0:
        a["nr"] += 1
        a["rn"] += rain
        a["rd"] += rain >= 0.1
        a["r10"] += rain >= 10
        a["r80"] += rain >= 80
    if np.isfinite(snow) and snow >= 0:
        a["ns"] += 1
        a["sn"] += snow >= 0.1
return out
```

실행 예는 서울(108) 2020~2025년 자료입니다. 아래 전국 30년 도표와 조회조건이 다릅니다. 전국 비교에는 stn=0과 동일 분석기간을 지정하고, 분석기간 전체에 관측이 있는 공통지점과 요소별 유효일수를 적용합니다.

❷ 30년 평균 일최저25℃ 이상일수 분포(참고)

```
# 응용 함수: 원시 API 응답과 다른 가공 입력을 사용하므로 입력 구조에 맞춰 자료를 준비한다.
def collect_annual(years, stn=108, min_valid=350):
    """조회→집계→CSV용 표. 전국 비교는 stn=0, 동일 지점 구성은 별도 선택."""
    rows = []
    for year in years:
        if year >= datetime.now(KST).year:
            raise ValueError("완료된 연도만 연간 통계에 사용")
        for station, a in count_year(get_daily(year, stn)).items():
            row = {
                "지점": station,
                "지점명": str(station),
                "연도": year,
                "유효일수": a["n"],
                "강수유효일수": a["nr"],
                "최고기온유효일수": a["ntmax"],
                "최저기온유효일수": a["ntmin"],
                "강수일수": a["rd"],
                "강수일수10mm": a["r10"],
```

통계분석

```
            "강수일수80mm": a["r80"],
            "폭염일수": a["hw"],
            "일최저25°C이상일수(참고)": a["tn"],
            "결빙일수": a["ic"],
            "신적설일수": a["sn"],
            "연강수량": a["rn"],
            "연평균기온": a["ta"] / a["n"] if a["n"] else np.nan,
        }
        for key in ["강수일수", "강수일수10mm", "강수일수80mm", "연강수량"]:
            if a["nr"] < min_valid:
                row[key] = np.nan
        for key in ["폭염일수", "결빙일수"]:
            if a["ntmax"] < min_valid:
                row[key] = np.nan
        if a["ntmin"] < min_valid:
            row["일최저25°C이상일수(참고)"] = np.nan
        if a["ns"] < min_valid:
            row["신적설일수"] = np.nan
        if a["n"] < min_valid:
            row["연평균기온"] = np.nan
        rows.append(row)
    time.sleep(0.3)
  return pd.DataFrame(rows)
# 실행 예: 한 지점으로 확인 후 전국 조회(stn=0)로 확장
# 실행 예: d = collect_annual(range(2020, 2026), stn=108)
# 실행 예: d.to_csv("annual_statistics.csv", index=False, encoding="utf-8-sig")
def load_pack(path):
  """collect_annual 결과 CSV 읽기. 구 CSV 형식 사용하지 않음."""
  d = pd.read_csv(path, encoding="utf-8-sig", dtype={"지점": int, "연도": int})
  required = {"지점", "지점명", "연도", "유효일수"}
  if not required.issubset(d.columns):
    raise ValueError("collect_annual 결과 CSV를 입력")
  return d
def mean_by_stn(d, cols):
  """30년을 지점마다 평균한다. 연수가 다르면 평균도 다르므로 연수를 남긴다."""
  g = d.groupby(["지점", "지점명"])[cols].mean().round(1).reset_index()
  g["연수"] = d.groupby(["지점", "지점명"]).size().values
  return g
```

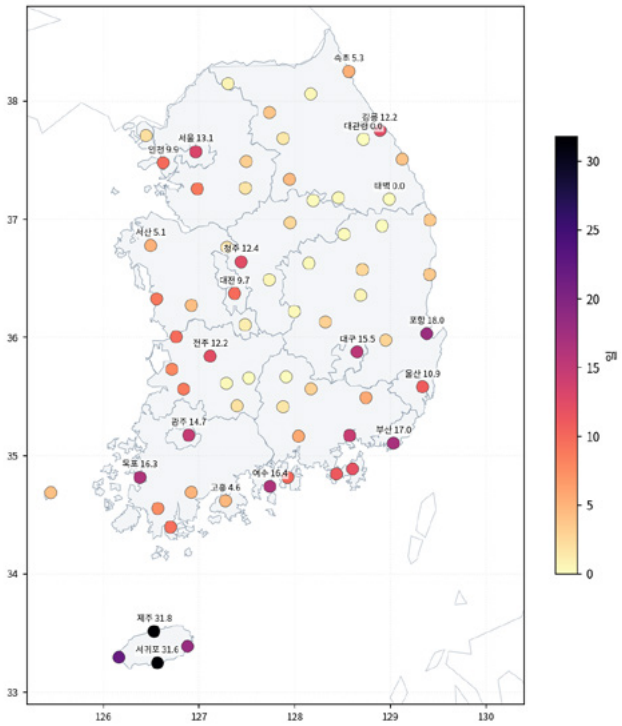


그림 49-1. 30년 평균 일최저25℃ 이상일수 분포(참고)

❸ 폭염일수와 일최저25℃ 이상일수 비교(참고)

같은 일자료로 계산한 폭염일수와 일최저25℃ 이상일수를 지점별로 비교합니다.

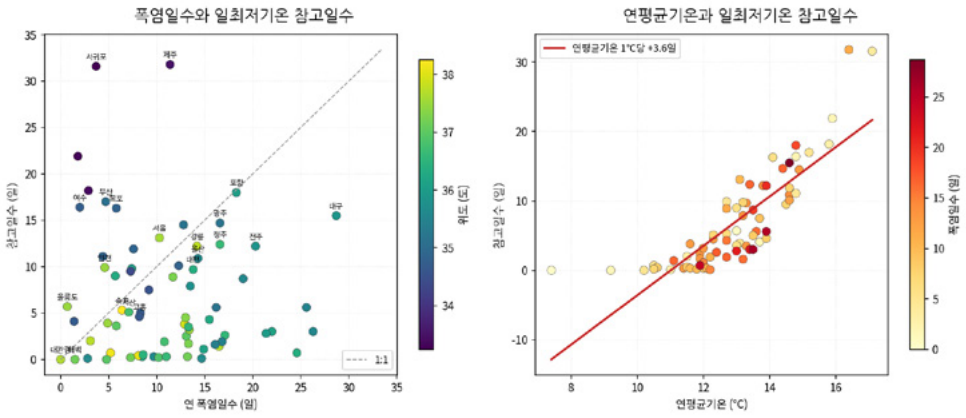


그림 49-2. 폭염일수와 일최저25℃ 이상일수 비교(참고)

❹ 30년간 전국 평균 변화

연도별 평균을 비교할 때 관측지점 구성과 유효 관측일수를 함께 확인합니다.

```
def nation_year(d, cols):
    """해마다 전국 지점을 평균한다. 지점 수가 해마다 조금씩 달라지므로 함께 남긴다."""
    g = d.groupby("연도")[cols].mean().round(2)
    g["지점수"] = d.groupby("연도").size()
    return g.reset_index()
def year_rank(ny, col, n=8):
    """더웠던 해를 순위로 세운다."""
    t = ny.nlargest(n, col)[["연도", col, "지점수"]].copy()
    t["연도"] = t["연도"].astype(int)
    t = t.rename(columns={col: f"{col}(전국평균)"})
    return t.reset_index(drop=True)
```

연도	일최저25℃ 이상일수(참고)(전국평균)	지점수
2024	23.61	70
2018	16.2	70
2013	13.16	70
2022	12.21	70
2010	10.21	70
2016	10.13	70
2017	9.36	70
2019	9.27	70

통계분석

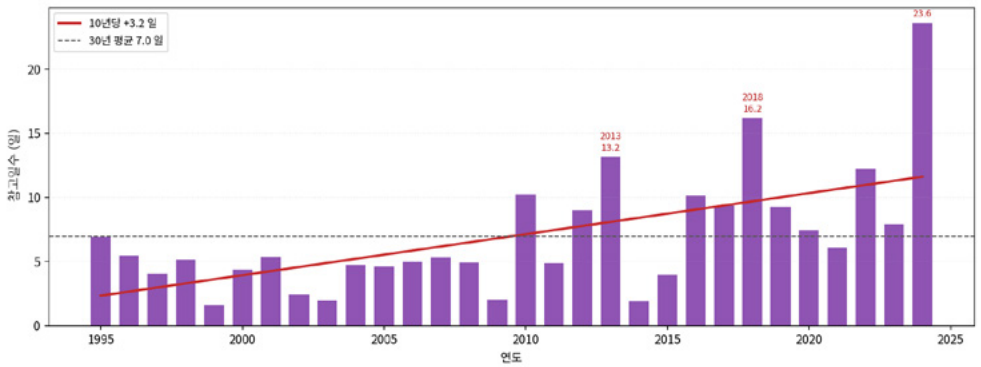


그림 49-3. 30년간 전국 평균 변화

❺ 지점별 증가 속도 비교

지점별 기울기를 지도와 막대그래프로 비교합니다.

```
def slope_by_stn(d, col):
  """지점마다 30년 기울기를 잰다. 단위는 '10년당 며칠'."""
  rows = []
  for (stn, name), g in d.groupby(["지점", "지점명"]):
    if len(g) < 25:
      continue
    a = np.polyfit(g["연도"], g[col], 1)[0]
    rows.append(
      {
        "지점": stn,
        "지점명": name,
        "10년당": round(a * 10, 2),
        "평균": round(g[col].mean(), 1),
      }
    )
  return pd.DataFrame(
    rows, columns=None if rows else ["지점", "지점명", "10년당", "평균"]
  ).sort_values("10년당", ascending=False)
```

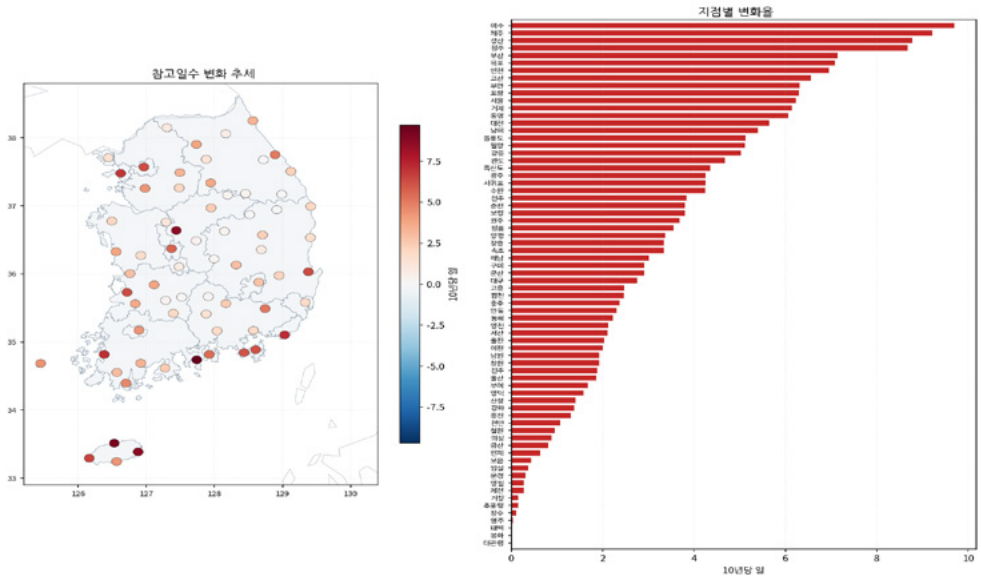


그림 49-4. 지점별 증가 속도 비교

지점	지점명	10년당	평균
168	여수	9.69	16.4
184	제주	9.22	31.8
188	성산	8.78	18.2
131	청주	8.68	12.4
159	부산	7.15	17
165	목포	7.08	16.3
272	영주	0.04	0.2
216	태백	0.02	0
100	대관령	0	0
271	봉화	0	0

❻ 대표 지점의 7년 이동평균 비교

7년 이동평균을 비교해 도시·해안·산지의 장기 변화 속도 차이를 확인합니다.

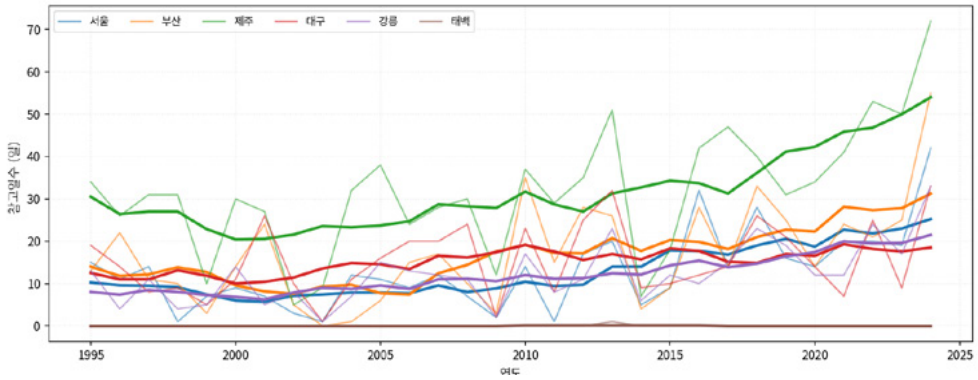


그림 49-5. 대표 지점의 7년 이동평균 비교

기상기후데이터 카탈로그

데이터 활용사례 50

메타데이터

Metadata



지점정보

50. 관측지점정보

관측망 분포와 지점 간 거리 분석

데이터 활용 경로(API)
기상청 API허브 > 지상관측 > 지상관측 지점정보

❶ 라이브러리 설치

필요한 라이브러리를 설치합니다. API 요청은 requests, 자료 처리는 pandas·numpy, 그래프는 matplotlib을 사용하며 지도에는 cartopy를 추가합니다.

```
python -m pip install requests pandas matplotlib cartopy numpy # 터미널에서 실행
```

❷ 관측지점 목록 조회

API허브에서 발급받은 인증키를 환경변수에 등록해 사용합니다.

```
# 이 사례의 코드 앞에서 실행(Python 3.9 이상 권장)
import os, math, re, time, io, gzip, struct, zipfile
from pathlib import Path
from datetime import datetime, timedelta, timezone
import xml.etree.ElementTree as ET
import requests
import pandas as pd
import numpy as np
API_KEY = os.environ["KMA_API_KEY"]
KST = timezone(timedelta(hours=9))
log = print
import os
API_KEY = os.environ["KMA_API_KEY"]
from io import StringIO
api_url = 'https://apihub.kma.go.kr/api/typ01/url/stn_inf.php'
def get_stn(inf):
  res = requests.get(
    api_url, params={"inf": inf, "help": 0, "authKey": API_KEY}, timeout=60
  )
  res.encoding = "euc-kr"
  rows = [
    ln.split()
    for ln in res.text.splitlines()
    if ln.strip() and (not ln.startswith("#"))
  ]
  df = pd.DataFrame(
    [[r[0], r[1], r[2], r[4]] for r in rows],
    columns=["지점", "경도", "위도", "고도"],
  )
  return df.astype({"지점": int, "경도": float, "위도": float, "고도": float})
sfc = get_stn('SFC')
aws = get_stn('AWS')
print(len(sfc), len(aws))
97 745
```

❸ 관측망 지도 표출

응답의 위경도로 관측망을 표시하고, cartopy로 해안선·시도 경계를 함께 그립니다.

```
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
PC = ccrs.PlateCarree()
def basemap(ax):
  ax.set_extent([124, 132, 32, 40], crs=PC)
  ax.add_feature(cfeature.COASTLINE.with_scale("10m"), linewidth=0.5)
  return ax
fig = plt.figure(figsize=(9.6, 5.8))
for i, (d, t, c) in enumerate([(sfc, '종관기상관측', '#C62828'),
                (aws, '방재기상관측', '#1F4E79')], start=1):
  ax = basemap(fig.add_subplot(1, 2, i, projection=PC))
  ax.scatter(d['경도'], d['위도'], s=10, c=c, linewidths=0,
        transform=PC, zorder=5)
  ax.set_title(f'{t} {len(d)}곳')
plt.show()
```

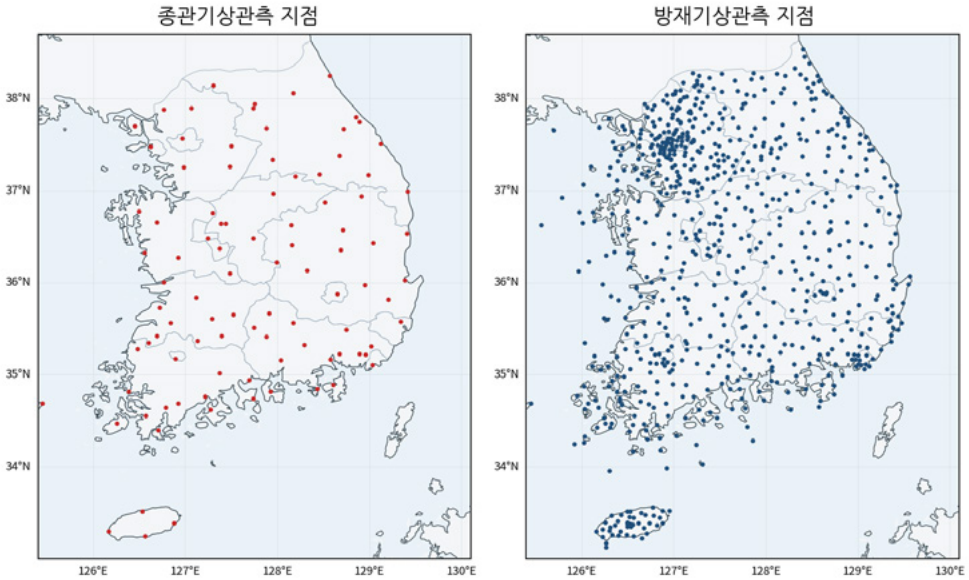


그림 50-1. 두 관측망의 지점 배치

❹ 가장 가까운 관측소까지의 거리

같은 관측망에서 각 지점의 최근접 거리를 계산해 관측망 간격을 비교합니다.

```
def haversine(lon1, lat1, lon2, lat2):
  R = 6371.0
  dlon, dlat = (np.radians(lon2 - lon1), np.radians(lat2 - lat1))
  h = (
    np.sin(dlat / 2) ** 2
    + np.cos(np.radians(lat1))
    * np.cos(np.radians(lat2))
    * np.sin(dlon / 2) ** 2
  )
  return 2 * R * np.arcsin(np.sqrt(np.clip(h, 0, 1)))
def nearest(d):
  if len(d) < 2:
    return np.full(len(d), np.nan)
  lon, lat = (d["경도"].values, d["위도"].values)
  out = []
  for i in range(len(d)):
    dist = haversine(lon[i], lat[i], lon, lat)
    dist[i] = np.inf
    out.append(dist.min())
```

지점정보

```
    return np.array(out)
near_sfc, near_aws = nearest(sfc), nearest(aws)
print(near_sfc.mean(), near_aws.mean())
26.763 8.434
```

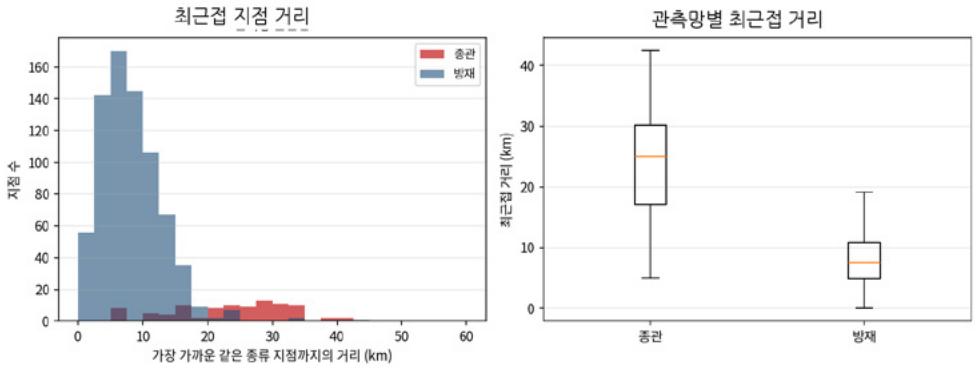


그림 50-2. 같은 관측망 안에서 가장 가까운 지점까지의 거리

관측망	지점 수	평균(km)	25%(km)	중앙값(km)	75%(km)	최대(km)
종관기상 관측	97	26.8	17.1	25	30.2	155.1
방재기상 관측	745	8.4	4.8	7.6	10.8	90

표 50-1. 관측망별 최근접 거리

❺ 지점 고도 분포

노장 고도를 구간별로 집계해 관측망의 고도 분포를 비교합니다.

```
bins = [0, 100, 200, 500, 1000, 2000]
tab = pd.DataFrame({
  '종관': pd.cut(sfc['고도'], bins, right=False).value_counts().sort_index(),
  '방재': pd.cut(aws['고도'], bins, right=False).value_counts().sort_index(),
})
print(tab)
print(aws.nlargest(6, '고도'))
```

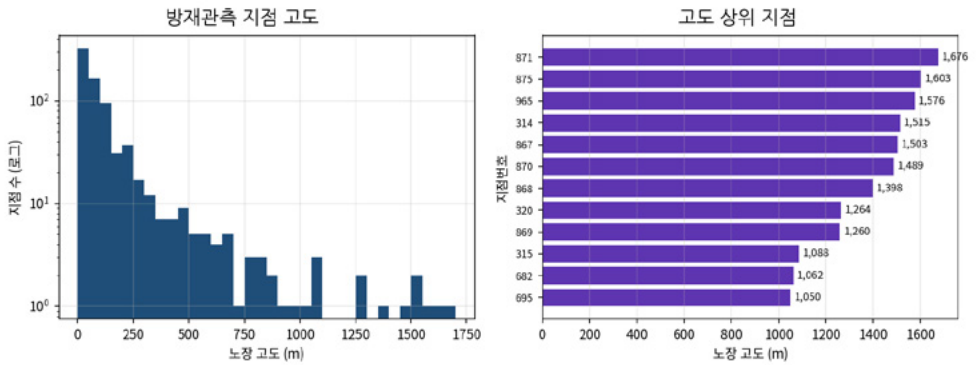


그림 50-3. 노장 고도 분포와 가장 높은 지점

고도 구간(m)	종관	방재	방재 비율(%)
0~100	69	487	65.4
100~200	14	126	16.9
200~500	12	89	11.9
500~1000	2	30	4
1000~2000	0	13	1.7

표 50-2. 고도 구간별 지점 수

지점번호	경도	위도	고도(m)
871	126.518	33.362	1675.8
875	128.46	38.121	1603.1
965	126.533	33.352	1576
314	127.743	35.866	1515.3
867	126.53	33.377	1503.3
870	126.556	33.37	1488.6

표 50-3. 노장 고도가 가장 높은 6곳

2026
기상기후데이터
카탈로그

발행일	2026년 11월
발행처	**기상청 국가기후데이터센터**
주소	대전광역시 서구 청사로 189 정부대전청사 1동 11~14층
홈페이지	https://data.kma.go.kr
기획디자인	엠케이디자인
발간번호	11-1360000-100251-01

게재된 글과 사진은 무단복제를 금합니다.

ISBN 979-11-6988-791-5(05450)

기상청 Korea Meteorological Administration
국가기후데이터센터

데이터로 만드는
더 안전한 내일

비매품/무료
05450
9 791169 887915
ISBN 979-11-6988-791-5 (PDF)