

Package: climaemet (via r-universe)

October 12, 2026

Type Package

Title Access and Visualize AEMET Weather and Climate Data

Version 1.6.0

Description Provides access to meteorological observations, forecasts, alerts and climatology data from the Spanish Meteorological Agency (AEMET) through the 'AEMET OpenData' API [<https://opendata.aemet.es/>](https://opendata.aemet.es/). Includes tools for working with tabular and spatial data and for creating Walter-Lieth climate diagrams, warming stripes and wind roses.

License GPL-3

URL <https://ropenspain.github.io/climaemet/>,
<https://github.com/rOpenSpain/climaemet>

BugReports <https://github.com/rOpenSpain/climaemet/issues>

Depends R (>= 4.1.0)

Imports cli (>= 3.0.0), dplyr (>= 1.0.0), ggplot2 (>= 3.5.0), httr2 (>= 1.0.0), jsonlite (>= 1.7.0), rappdirs (>= 0.3.3), readr (>= 1.4.0), rlang (>= 0.4.6), tidyr (>= 1.1.0), tools, xml2

Suggests climatol (>= 3.1.2), gganimate (>= 1.0.5), jpeg (>= 0.1.8), knitr, lifecycle, lubridate, mapSpain, quarto, scales, sf (>= 1.0.0), terra (>= 1.8-10), testthat (>= 3.2.0), tibble, withr (>= 3.0.0)

VignetteBuilder quarto

Config/Needs/website classInt, crosstalk, geoR, gifski, gstat, leaflet, pak, reactable, scales, styler, tidyterra, tidyverse, usethis

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.1

Config/testthat/edition 3

Config/testthat/parallel false

Copyright © AEMET. See file inst/COPYRIGHTS

Encoding UTF-8

LazyData true

X-schema.org-applicationCategory Meteorology

X-schema.org-isPartOf <https://ropenspain.es/>

X-schema.org-keywords aemet, climate, cran, data, forecast-api, r,
r-package, ropenspain, rstats, science, spain, weather-api

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev libx11-dev

Repository <https://dieghernan.r-universe.dev>

Date/Publication 2026-10-12 17:57:07 UTC

RemoteUrl <https://github.com/rOpenSpain/climaemet>

RemoteRef HEAD

RemoteSha d97cac106f5ebb89d9ec00f48c419fb31b26c96e

Contents

aemet_alert_zones	3
aemet_alerts	4
aemet_api_key	6
aemet_beaches	8
aemet_daily_clim	9
aemet_detect_api_key	11
aemet_extremes_clim	12
aemet_forecast_beaches	13
aemet_forecast_daily	14
aemet_forecast_fires	17
aemet_forecast_tidy	19
aemet_last_obs	20
aemet_monthly_clim	22
aemet_munic	23
aemet_normal_clim	24
aemet_stations	25
climaemet_9434_climatogram	27
climaemet_9434_temp	27
climaemet_9434_wind	28
climaemet_news	29
climatestripes_station	29
climatogram_normal	31
climatogram_period	32
dms2decdegrees	33
first_day_of_year	34
get_data_aemet	35
ggclimat_walter_lieth	36
ggstripes	38
ggwindrose	39

<i>aemet_alert_zones</i>	3
windrose_days	41
windrose_period	42
Index	45

<i>aemet_alert_zones</i>	<i>AEMET alert zones</i>
--------------------------	--------------------------

Description

Retrieves the AEMET geographical zones used for meteorological alerts.

Usage

```
aemet_alert_zones(verbose = FALSE, return_sf = FALSE)
```

Arguments

<code>verbose</code>	A logical value. If TRUE, displays information about the exchange between the client and server.
<code>return_sf</code>	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.

Value

A [tibble](#) or an [sf](#) object.

Caching

The first result retrieved in each session is temporarily cached in [tempdir\(\)](#) to avoid unnecessary requests.

Source

<https://www.aemet.es/es/eltiempo/prediccion/avisos/ayuda>. See also Annex 2 and Annex 3 documents, linked from that page.

See Also

Weather alerts: [aemet_alerts\(\)](#)

AEMET locations: [aemet_beaches\(\)](#), [aemet_stations\(\)](#)

Examples

```
library(dplyr)
alert_zones <- aemet_alert_zones()
alert_zones

# Cached during this R session.
alert_zones2 <- aemet_alert_zones(verbose = TRUE)

identical(alert_zones, alert_zones2)

# Select and map alert zones.
library(ggplot2)

# Galicia.
alert_zones_sf <- aemet_alert_zones(return_sf = TRUE) |>
  filter(COD_CCAA == "71")

# Coast zones have codes ending in "C".
alert_zones_sf$type <- ifelse(grepl("C$", alert_zones_sf$COD_Z),
  "Coast", "Mainland"
)

ggplot(alert_zones_sf) +
  geom_sf(aes(fill = NOM_PROV)) +
  facet_wrap(~type) +
  scale_fill_brewer(palette = "Blues")
```

aemet_alerts

AEMET meteorological alerts

Description

[Experimental] Retrieves current meteorological alerts issued by AEMET.

Usage

```
aemet_alerts(
  ccaa = NULL,
  lang = c("es", "en"),
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)
```

Arguments

ccaa	A character vector of autonomous community names or NULL to retrieve all autonomous communities.
lang	The language of the results, either "es" (Spanish) or "en" (English).
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

Source

<https://www.aemet.es/en/eltiempo/prediccion/avisos> and <https://www.aemet.es/es/eltiempo/prediccion/avisos/ayuda> for API status and alerts reference, including Annex 2 and Annex 3 documentation.

See Also

See [mapSpain::esp_codelist](#) and [mapSpain::esp_dict_region_code\(\)](#) for autonomous community names.

Weather alerts: [aemet_alert_zones\(\)](#)

Weather observations: [aemet_last_obs\(\)](#)

Examples

```
# Display CCAA names.
library(dplyr)
aemet_alert_zones() |>
  select(NOM_CCAA) |>
  distinct()

# Base map.
cbasemap <- mapSpain::esp_get_ccaa(ccaa = c(
  "Galicia", "Asturias", "Cantabria",
```

```

    "Euskadi"
  ))

# Alerts.
alerts_north <- aemet_alerts(
  ccaa = c("Galicia", "Asturias", "Cantabria", "Euskadi"),
  return_sf = TRUE
)

# Plot if there are alerts.
if (inherits(alerts_north, "sf")) {
  library(ggplot2)
  library(lubridate)

  alerts_north$day <- date(alerts_north$effective)

  ggplot(alerts_north) +
    geom_sf(data = cbasemap, fill = "grey60") +
    geom_sf(aes(fill = `AEMET-Meteoalerta nivel`)) +
    geom_sf(
      data = cbasemap, fill = "transparent", color = "black",
      linewidth = 0.5
    ) +
    facet_grid(vars(`AEMET-Meteoalerta fenomeno`), vars(day)) +
    scale_fill_manual(values = c(
      "amarillo" = "yellow", naranja = "orange",
      "rojo" = "red"
    ))
}

```

aemet_api_key

Install an AEMET OpenData API key

Description

Stores an AEMET OpenData API key on your local machine so it can be used without including it in your code.

Usage

```
aemet_api_key(apikey, overwrite = FALSE, install = FALSE)
```

Arguments

apikey	A character vector of AEMET OpenData API keys. Acquire a key at https://opendata.aemet.es/centrodedescargas/inicio . You can install multiple API keys at once. See Details .
overwrite	A logical value. If TRUE, overwrites an existing AEMET_API_KEY environment variable.

`install` A logical value. If TRUE, installs the key on your local machine for use in future sessions. Defaults to FALSE.

Details

Alternatively, set the key for the current session with `Sys.setenv(AEMET_API_KEY = "Your_Key")`, equivalent to `install = FALSE`. To store it permanently, add `AEMET_API_KEY = "Your_Key"` to `.Renviron`, equivalent to `install = TRUE`.

You can pass multiple `apikey` values as a character vector, such as `c(api1, api2)`. In this case, multiple `AEMET_API_KEY` values are stored. In each subsequent API call, **climaemet** chooses the API key with the highest remaining quota.

This is useful when performing batch queries to avoid API throttling.

Value

NULL, invisibly.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

Note

To locate the stored API key, run `tools::R_user_dir("climaemet", "config")`.

See Also

AEMET OpenData API functions: [aemet_detect_api_key\(\)](#), [get_data_aemet\(\)](#)

Examples

```
# Do not run these examples.

if (FALSE) {
  aemet_api_key("111111abc", install = TRUE)

  # Check it with:
  Sys.getenv("AEMET_API_KEY")
}

if (FALSE) {
  # Overwrite an existing key:
  aemet_api_key("222222abc", overwrite = TRUE, install = TRUE)

  # Check it with:
  Sys.getenv("AEMET_API_KEY")
}
```

aemet_beaches	<i>AEMET beaches</i>
---------------	----------------------

Description

Retrieves the beaches available from the AEMET OpenData API.

Usage

```
aemet_beaches(verbose = FALSE, return_sf = FALSE)
```

Arguments

verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.

Value

A [tibble](#) or an [sf](#) object.

Caching

The first result retrieved in each session is temporarily cached in [tempdir\(\)](#) to avoid unnecessary requests.

See Also

[aemet_forecast_beaches\(\)](#).
AEMET locations: [aemet_alert_zones\(\)](#), [aemet_stations\(\)](#)

Examples

```
library(dplyr)
beaches <- aemet_beaches()
beaches

# Cached during this R session.
beaches2 <- aemet_beaches(verbose = TRUE)

identical(beaches, beaches2)

# Select and map beaches.
library(ggplot2)
library(mapSpain)

# Alicante / Alacant.
```

```
beaches_sf <- aemet_beaches(return_sf = TRUE) |>
  filter(ID_PROVINCIA == "03")

prov <- mapSpain::esp_get_prov("Alicante")

ggplot(prov) +
  geom_sf() +
  geom_sf(
    data = beaches_sf, shape = 4, size = 2.5,
    color = "blue"
  )
```

aemet_daily_clim	<i>Daily and annual climatology values</i>
------------------	--

Description

Retrieves climatology values for one station or all available stations. [aemet_daily_period\(\)](#) and [aemet_daily_period_all\(\)](#) are shortcuts for [aemet_daily_clim\(\)](#).

Usage

```
aemet_daily_clim(
  station = "all",
  start = Sys.Date() - 7,
  end = Sys.Date(),
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_daily_period(
  station,
  start = as.integer(format(Sys.Date(), "%Y")),
  end = start,
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_daily_period_all(
  start = as.integer(format(Sys.Date(), "%Y")),
  end = start,
  verbose = FALSE,
  return_sf = FALSE,
```

```

    extract_metadata = FALSE,
    progress = TRUE
  )

```

Arguments

station	A character vector of station identifiers (see aemet_stations()) or "all" for all stations.
start, end	Character strings containing the start and end dates. See Details .
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Details

For [aemet_daily_clim\(\)](#), start and end must be Date objects or strings in YYYY-MM-DD format, such as "2020-12-31", that can be coerced with [as.Date\(\)](#). For [aemet_daily_period\(\)](#) and [aemet_daily_period_all\(\)](#), they must be strings representing the years to extract, such as "2018" and "2020".

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

[aemet_stations\(\)](#) for station identifiers.

Climatology: [aemet_extremes_clim\(\)](#), [aemet_monthly_clim\(\)](#), [aemet_normal_clim\(\)](#)

Examples

```

library(dplyr)
obs <- aemet_daily_clim(c("9434", "3195"))
glimpse(obs)

# Metadata.

```

```
meta <- aemet_daily_clim(c("9434", "3195"), extract_metadata = TRUE)

glimpse(meta$campos)
```

aemet_detect_api_key *Check for an AEMET OpenData API key*

Description

Detects whether an API key is available in the current session. An existing environment variable is preserved. Otherwise, a key stored permanently with [aemet_api_key\(\)](#) is loaded.

Usage

```
aemet_detect_api_key(...)

aemet_show_api_key(...)
```

Arguments

... Ignored.

Value

[aemet_detect_api_key\(\)](#) returns a [logical](#) value, TRUE if an API key is available and FALSE otherwise. [aemet_show_api_key\(\)](#) returns a [character](#) vector containing the available API keys.

See Also

AEMET OpenData API functions: [aemet_api_key\(\)](#), [get_data_aemet\(\)](#)

Examples

```
aemet_detect_api_key()

# Caution: This may reveal API keys.
if (FALSE) {
  aemet_show_api_key()
}
```

aemet_extremes_clim *Extreme values for a station*

Description

Retrieves recorded extreme values for one or more stations.

Usage

```
aemet_extremes_clim(
  station = NULL,
  parameter = "T",
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)
```

Arguments

station	A character vector of station identifiers. See aemet_stations() .
parameter	A character string specifying the parameter to retrieve: temperature ("T"), precipitation ("P") or wind ("V").
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Value

A [tibble](#) or an [sf](#) object. If the function encounters a parsing error, it returns a [list](#).

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

[aemet_stations\(\)](#) for station identifiers.

Climatology: [aemet_daily_clim\(\)](#), [aemet_monthly_clim\(\)](#), [aemet_normal_clim\(\)](#)

Examples

```
obs <- aemet_extremes_clim(c("9434", "3195"))
dplyr::glimpse(obs)
```

aemet_forecast_beaches

Forecast weather at beaches

Description

Retrieves daily weather forecasts for one or more beaches. Use [aemet_beaches\(\)](#) to obtain beach codes.

Usage

```
aemet_forecast_beaches(
  x,
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)
```

Arguments

x	A character vector of beach codes to extract. See aemet_beaches() .
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

[aemet_beaches\(\)](#) for beach codes.

Forecasts: [aemet_forecast_daily\(\)](#), [aemet_forecast_fires\(\)](#), [aemet_forecast_tidy\(\)](#)

Examples

```
# Forecast for beaches in Palma, Mallorca.
library(dplyr)
library(ggplot2)

palma_b <- aemet_beaches() |>
  filter(ID_MUNICIPIO == "07040")

forecast_b <- aemet_forecast_beaches(palma_b$ID_PLAYA)
glimpse(forecast_b)

ggplot(forecast_b) +
  geom_line(aes(fecha, tagua_valor1, color = nombre)) +
  facet_wrap(~nombre, ncol = 1) +
  labs(
    title = "Water temperature at beaches in Palma (ES)",
    subtitle = "3-day forecast",
    x = "Date",
    y = "Temperature (Celsius)",
    color = "Beach"
  )
```

aemet_forecast_daily *Forecast weather in municipalities*

Description

Retrieves daily or hourly weather forecasts for one or more municipalities.

Usage

```
aemet_forecast_daily(
  x,
  verbose = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_forecast_hourly(
  x,
  verbose = FALSE,
  extract_metadata = FALSE,
```

```
    progress = TRUE
  )
```

Arguments

x	A character vector of municipality codes to extract. For convenience, climaemet provides these data in the aemet_munic dataset (see <code>municipio</code> field) as of January 2025.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a <code>cli::cli_progress_bar()</code> unless <code>verbose = TRUE</code> .

Details

Forecasts provided by the AEMET OpenData API have a complex structure. Although **climaemet** returns a [tibble](#), each forecast value is provided as a nested tibble. The [aemet_forecast_tidy\(\)](#) helper can unnest these values and provide a single unnested tibble for the requested variable.

If `extract_metadata = TRUE`, the function returns a simple tibble describing each forecast field.

Value

A nested [tibble](#). Forecast values can be extracted with [aemet_forecast_tidy\(\)](#). See also **Details**.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

- [aemet_munic](#) provides municipality codes.
- **mapSpain** provides `sf` objects of municipalities through [mapSpain::esp_get_munic\(\)](#). See also **Examples**.

Forecasts: [aemet_forecast_beaches\(\)](#), [aemet_forecast_fires\(\)](#), [aemet_forecast_tidy\(\)](#)

Examples

```
# Select cities.
data("aemet_munic")
library(dplyr)
munis <- aemet_munic |>
  filter(municipio_nombre %in% c("Santiago de Compostela", "Lugo")) |>
  pull(municipio)
```

```

daily <- aemet_forecast_daily(munis)

# Metadata.
meta <- aemet_forecast_daily(munis, extract_metadata = TRUE)
glimpse(meta$campos)

# Variables available.
aemet_forecast_vars_available(daily)

# This is nested.
daily |>
  select(municipio, fecha, nombre, temperatura)

# Select and unnest.
daily_temp <- aemet_forecast_tidy(daily, "temperatura")

# This is not nested.
daily_temp

# Wrangle and plot.
daily_temp_end <- daily_temp |>
  select(
    elaborado, fecha, municipio, nombre, temperatura_minima,
    temperatura_maxima
  ) |>
  tidyr::pivot_longer(cols = contains("temperatura"))

# Plot.
library(ggplot2)
ggplot(daily_temp_end) +
  geom_line(aes(fecha, value, color = name)) +
  facet_wrap(~nombre, ncol = 1) +
  scale_color_manual(
    values = c("red", "blue"),
    labels = c("max", "min")
  ) +
  scale_x_date(
    labels = scales::label_date_short(),
    breaks = "day"
  ) +
  scale_y_continuous(
    labels = scales::label_comma(suffix = "°")
  ) +
  theme_minimal() +
  labs(
    x = "", y = "",
    color = "",
    title = "Forecast: 7-day temperature",
    subtitle = paste(
      "Forecast produced on",
      format(daily_temp_end$elaborado[1], usetz = TRUE)
    )
  )

```

```

    )

# Spatial data.
library(mapSpain)
library(sf)

lugo_sf <- esp_get_munic(munic = "Lugo") |>
  select(LAU_CODE)

daily_temp_end_lugo_sf <- daily_temp_end |>
  filter(nombre == "Lugo" & name == "temperatura_maxima") |>
  # Join by LAU_CODE.
  left_join(lugo_sf, by = c("municipio" = "LAU_CODE")) |>
  st_as_sf()

ggplot(daily_temp_end_lugo_sf) +
  geom_sf(aes(fill = value)) +
  facet_wrap(~fecha) +
  scale_fill_gradientn(
    colors = c("blue", "red"),
    guide = guide_legend()
  ) +
  labs(
    main = "Forecast: 7-day max temperature",
    subtitle = "Lugo, ES"
  )

```

aemet_forecast_fires *AEMET wildfire risk forecast*

Description

Retrieves daily wildfire risk levels as either tabular data or a [SpatRaster](#).

Usage

```

aemet_forecast_fires(
  area = c("p", "c"),
  verbose = FALSE,
  extract_metadata = FALSE
)

```

Arguments

area	A character string specifying the forecast area: "p" for mainland Spain and the Balearic Islands or "c" for the Canary Islands.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.

extract_metadata

A logical value. If TRUE, returns a [tibble](#) describing the response fields. See [get_metadata_aemet\(\)](#).

Details

The `SpatRaster` provides six [factor\(\)](#) levels: "1" for very low risk, "2" for low risk, "3" for moderate risk, "4" for high risk, "5" for very high risk and "6" for extreme risk.

The resulting object has several layers, each representing one of the next seven forecast days. It also has additional attributes provided by [terra](#), including [terra::time\(\)](#) and [terra::coltab\(\)](#).

Value

A [tibble](#) or a [SpatRaster](#).

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

Source

<https://www.aemet.es/en/eltiempo/prediccion/incendios>.

See Also

Forecasts: [aemet_forecast_beaches\(\)](#), [aemet_forecast_daily\(\)](#), [aemet_forecast_tidy\(\)](#)

Examples

```
aemet_forecast_fires(extract_metadata = TRUE)

# Extract alerts.
alerts <- aemet_forecast_fires()

alerts

# Plot the raster.
library(terra)
plot(alerts, all_levels = TRUE)

# Zoom in on an area.
cyl <- mapSpain::esp_get_ccaa("Castilla y Leon", epsg = 4326)

# Convert to a SpatVector.
cyl <- vect(cyl)

fires_cyl <- crop(alerts, cyl)
title <- names(fires_cyl)[1]
```

```
plot(fires_cyl[[1]], main = title, all_levels = TRUE)
plot(cyl, add = TRUE)
```

aemet_forecast_tidy *Extract values from forecasts*

Description

[Experimental] [aemet_forecast_vars_available\(\)](#) lists the variables in output from [aemet_forecast_daily\(\)](#) or [aemet_forecast_hourly\(\)](#). [aemet_forecast_tidy\(\)](#) extracts the forecast for var as a [tibble](#).

Usage

```
aemet_forecast_tidy(x, var)

aemet_forecast_vars_available(x)
```

Arguments

x A dataset extracted with [aemet_forecast_daily\(\)](#) or [aemet_forecast_hourly\(\)](#).
var The name of the forecast variable to extract.

Value

A [character](#) vector from [aemet_forecast_vars_available\(\)](#) or a [tibble](#) from [aemet_forecast_tidy\(\)](#).

See Also

Forecasts: [aemet_forecast_beaches\(\)](#), [aemet_forecast_daily\(\)](#), [aemet_forecast_fires\(\)](#)

Examples

```
# Hourly values.
hourly <- aemet_forecast_hourly(c("15030", "28079"))

# Variables available.
aemet_forecast_vars_available(hourly)

# Get temperature.
temp <- aemet_forecast_tidy(hourly, "temperatura")

library(dplyr)
# Create a forecast time and adjust its time zone.
temp_end <- temp |>
  mutate(
    forecast_time = lubridate::force_tz(
      as.POSIXct(fecha) + hora,
      tz = "Europe/Madrid"
```

```

    )
  )

# Add sunset and sunrise.
suns <- temp_end |>
  select(nombre, fecha, orto, ocase) |>
  distinct_all() |>
  group_by(nombre) |>
  mutate(
    ocase_end = lubridate::force_tz(
      as.POSIXct(fecha) + ocase,
      tz = "Europe/Madrid"
    ),
    orto_end = lubridate::force_tz(
      as.POSIXct(fecha) + orto,
      tz = "Europe/Madrid"
    ),
    orto_lead = lead(orto_end)
  ) |>
  tidyr::drop_na()

# Plot.

library(ggplot2)

ggplot(temp_end) +
  geom_rect(data = suns, aes(
    xmin = ocase_end, xmax = orto_lead,
    ymin = min(temp_end$temperatura),
    ymax = max(temp_end$temperatura)
  ), alpha = 0.4) +
  geom_line(aes(forecast_time, temperatura), color = "blue4") +
  facet_wrap(~nombre, nrow = 2) +
  scale_x_datetime(labels = scales::label_date_short()) +
  scale_y_continuous(labels = scales::label_number(suffix = "°")) +
  labs(
    x = "", y = "",
    title = "Forecast: temperature",
    subtitle = paste("Forecast produced on", format(temp_end$elaborado[1],
      usetz = TRUE
    ))
  )
)

```

aemet_last_obs

Latest observations from weather stations

Description

Retrieves the latest observations for one or more weather stations.

Usage

```
aemet_last_obs(  
  station = "all",  
  verbose = FALSE,  
  return_sf = FALSE,  
  extract_metadata = FALSE,  
  progress = TRUE  
)
```

Arguments

station	A character vector of station identifiers (see aemet_stations()) or "all" for all stations.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

[aemet_stations\(\)](#) for station identifiers.

Weather observations: [aemet_alerts\(\)](#)

Examples

```
obs <- aemet_last_obs(c("9434", "3195"))  
dplyr::glimpse(obs)
```

aemet_monthly_clim	<i>Monthly and annual climatology values</i>
--------------------	--

Description

Retrieves monthly or annual climatology values for one or more stations. [aemet_monthly_period\(\)](#) and [aemet_monthly_period_all\(\)](#) allow requests that span several years.

Usage

```
aemet_monthly_clim(
  station = NULL,
  year = as.integer(format(Sys.Date(), "%Y")),
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_monthly_period(
  station = NULL,
  start = as.integer(format(Sys.Date(), "%Y")),
  end = start,
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_monthly_period_all(
  start = as.integer(format(Sys.Date(), "%Y")),
  end = start,
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)
```

Arguments

station	A character vector of station identifiers. See aemet_stations() .
year	A numeric year in YYYY format.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.

extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.
start	A numeric value specifying the start year in YYYY format.
end	A numeric value specifying the end year in YYYY format.

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

[aemet_stations\(\)](#) for station identifiers.
Climatology: [aemet_daily_clim\(\)](#), [aemet_extremes_clim\(\)](#), [aemet_normal_clim\(\)](#)

Examples

```
obs <- aemet_monthly_clim(station = c("9434", "3195"), year = 2000)
dplyr::glimpse(obs)
```

aemet_munic	<i>Municipalities of Spain</i>
-------------	--------------------------------

Description

A [tibble](#) containing all municipalities of Spain as defined by the INE (Instituto Nacional de Estadística) as of January 2025.

Format

- A [tibble](#) with 8,132 rows and fields:
- municipio** INE code of the municipality.
 - municipio_nombre** INE name of the municipality.
 - cpro** INE code of the province.
 - cpro_nombre** INE name of the province.
 - codauto** INE code of the autonomous community.
 - codauto_nombre** INE name of the autonomous community.

Source

INE municipality codes by province: <https://www.ine.es/daco/daco42/codmun/diccionario25.xlsx>.

See Also

- [aemet_forecast_daily\(\)](#) retrieves daily municipality forecasts.
- [aemet_forecast_hourly\(\)](#) retrieves hourly municipality forecasts.

Included datasets: [climaemet_9434_climatogram](#), [climaemet_9434_temp](#), [climaemet_9434_wind](#)

Examples

```
data(aemet_munic)

aemet_munic
```

aemet_normal_clim	<i>Climatological normal values</i>
-------------------	-------------------------------------

Description

Retrieves climatological normal values for a station or for all stations with [aemet_normal_clim_all\(\)](#). The standard normal period is 1981–2010.

Usage

```
aemet_normal_clim(
  station = NULL,
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)

aemet_normal_clim_all(
  verbose = FALSE,
  return_sf = FALSE,
  extract_metadata = FALSE,
  progress = TRUE
)
```

Arguments

station	A character vector of station identifiers (see aemet_stations()) or "all" for all stations.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.

return_sf	A logical value. If TRUE, the function returns an sf spatial object. If FALSE (the default), it returns a tibble . sf must be installed.
extract_metadata	A logical value. If TRUE, returns a tibble describing the response fields. See get_metadata_aemet() .
progress	A logical value. If TRUE, displays a cli::cli_progress_bar() unless verbose = TRUE.

Value

A [tibble](#) or an [sf](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

Note

Code adapted from the project at <https://github.com/SevillaR/aemet>.

See Also

[aemet_stations\(\)](#) for station identifiers.

Climatology: [aemet_daily_clim\(\)](#), [aemet_extremes_clim\(\)](#), [aemet_monthly_clim\(\)](#)

Examples

```
obs <- aemet_normal_clim(c("9434", "3195"))
dplyr::glimpse(obs)
```

aemet_stations	<i>AEMET stations</i>
----------------	-----------------------

Description

Retrieves the weather stations available from the AEMET OpenData API.

Usage

```
aemet_stations(verbose = FALSE, return_sf = FALSE)
```

Arguments

<code>verbose</code>	A logical value. If TRUE, displays information about the exchange between the client and server.
<code>return_sf</code>	A logical value. If TRUE, the function returns an <code>sf</code> spatial object. If FALSE (the default), it returns a <code>tibble</code> . <code>sf</code> must be installed.

Value

A `tibble` or an `sf` object.

Caching

The first result retrieved in each session is temporarily cached in `tempdir()` to avoid unnecessary requests.

API key

Queries to the AEMET OpenData API require an API key. Use `aemet_api_key()` to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

Note

Code adapted from the project at <https://github.com/SevillaR/aemet>.

See Also

- `aemet_last_obs()` retrieves latest station observations.
- `aemet_daily_clim()`, `aemet_monthly_clim()`, `aemet_normal_clim()` and `aemet_extremes_clim()` retrieve station climatology.

AEMET locations: `aemet_alert_zones()`, `aemet_beaches()`

Examples

```
library(dplyr)
stations <- aemet_stations()
stations

# Cached during this R session.
stations2 <- aemet_stations(verbose = TRUE)

identical(stations, stations2)
```

`climaemet_9434_climatogram`*Climatological normals for Zaragoza Airport ("9434"), 1981–2010*

Description

Climatological normal data for Zaragoza Airport (1981–2010). This example dataset is used to create Walter-Lieth climate diagrams.

Format

A [data frame](#) with four rows and 12 columns. Columns 1 through 12 represent months from January through December. Rows contain:

- `p_mes_md`: precipitation (mm).
- `tm_max_md`: maximum temperature (degrees Celsius).
- `tm_min_md`: minimum temperature (degrees Celsius).
- `ta_min_min`: absolute monthly minimum temperature (degrees Celsius).

Source

AEMET.

See Also

- [climatogram_normal\(\)](#) and [climatogram_period\(\)](#) retrieve climatology data.
- [ggclimat_walter_lieth\(\)](#) creates Walter-Lieth climate diagrams.

Included datasets: [aemet_munic](#), [climaemet_9434_temp](#), [climaemet_9434_wind](#)

Examples

```
data(climaemet_9434_climatogram)
```

`climaemet_9434_temp`*Average annual temperatures for Zaragoza Airport ("9434"), 1950–2020*

Description

Yearly observations of average temperature for Zaragoza Airport (1950–2020). This is an example dataset.

Format

A [tibble](#) with columns:

year Year of reference.

indicativo Identifier of the station.

temp Average temperature (degrees Celsius).

Source

AEMET.

See Also

- [climatestripes_station\(\)](#) retrieves annual temperature data.
- [ggstripes\(\)](#) creates warming stripe plots.

Included datasets: [aemet_munic](#), [climaemet_9434_climatogram](#), [climaemet_9434_wind](#)

Examples

```
data(climaemet_9434_temp)
```

climaemet_9434_wind	<i>Wind conditions for Zaragoza Airport ("9434"), 2000–2020</i>
---------------------	---

Description

Daily observations of wind speed and direction for Zaragoza Airport (2000–2020). This is an example dataset.

Format

A [tibble](#) with columns:

fecha Date of observation.

dir Wind direction (0-360 degrees).

velmedia Average wind speed (m/s).

Source

AEMET.

See Also

- [windrose_days\(\)](#) and [windrose_period\(\)](#) retrieve wind observations.
- [ggwindrose\(\)](#) creates wind rose plots.

Included datasets: [aemet_munic](#), [climaemet_9434_climatogram](#), [climaemet_9434_temp](#)

Examples

```
data(climaemet_9434_wind)
```

climaemet_news	Show	the	latest	Rhref https://CRAN.R-project.org/package=climaemet
----------------	------	-----	--------	---

Description

Opens the NEWS file for **climaemet**.

Usage

```
climaemet_news()
```

Value

NULL, invisibly. This function is called for its side effect.

See Also

Helper functions: [dms2decdegrees\(\)](#), [first_day_of_year\(\)](#)

Examples

```
## Not run:
climaemet_news()

## End(Not run)
```

climatestripes_station	Warming stripes for a weather station
------------------------	---------------------------------------

Description

Plots warming stripes for a weather station over a specified period.

Usage

```
climatestripes_station(
  station,
  start = 1950,
  end = 2020,
  with_labels = "yes",
  verbose = FALSE,
  ...
)
```

Arguments

station	A character string containing a station identifier. See aemet_stations() .
start	A numeric value specifying the start year in YYYY format.
end	A numeric value specifying the end year in YYYY format.
with_labels	A character string indicating whether to display plot labels, either "yes" or "no".
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
...	Arguments passed on to ggstripes
n_temp	The number of colors in the palette. Defaults to 11.
col_pal	A character string specifying an hcl.pals() color palette.

Value

A [ggplot](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

Note

Professor Ed Hawkins of the University of Reading developed the "warming stripes" concept to communicate climate change risks as simply as possible. For more details, see [ShowYourStripes](#).

See Also

[climaemet_9434_temp](#).

Warming stripes: [ggstripes\(\)](#)

Examples

```
# Do not run this example.
if (FALSE) {
  # Downloading data may take a few minutes.
  climatestripes_station(
    "9434",
    start = 2020,
    end = 2024,
    with_labels = "yes",
    col_pal = "Inferno"
  )
}
```

climatogram_normal	<i>Walter-Lieth climate diagram from climatological normals</i>
--------------------	---

Description

Plots a Walter-Lieth climate diagram from climatological normal values for a station. The diagram summarizes local climate conditions for 1981–2010.

Usage

```
climatogram_normal(
  station,
  labels = "en",
  verbose = FALSE,
  ggplot2 = TRUE,
  ...
)
```

Arguments

station	A character string containing a station identifier. See aemet_stations() .
labels	A character string specifying the language for the x-axis month labels, such as "en" (English), "es" (Spanish) or "fr" (French).
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
ggplot2	A logical value. If TRUE, the function uses ggclimat_walter_lieth() . If FALSE, it uses climatol::diagwl() .
...	Further arguments passed to climatol::diagwl() or ggclimat_walter_lieth() , depending on the value of ggplot2.

Value

A plot produced by [ggclimat_walter_lieth\(\)](#) or [climatol::diagwl\(\)](#), depending on ggplot2.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

References

Walter H, Lieth H (1967). *Klimadiagramm-Weltatlas*. VEB Gustav Fischer Verlag, Jena. Published in three installments, 1960-1967, <https://hdl.handle.net/2268.1/7079>.

Guijarro JA (2026). *climatol: Climate Tools (Series Homogenization and Derived Products)*. doi:10.32614/CRAN.package.climatol. R package version 4.5-0, <https://CRAN.R-project.org/package=climatol>.

See Also

[climaemet_9434_climatogram](#).

Walter-Lieth climate diagrams: [climatogram_period\(\)](#), [ggclimat_walter_lieth\(\)](#)

Examples

```
climatogram_normal("9434")
```

climatogram_period	<i>Walter-Lieth climate diagram for a time period</i>
--------------------	---

Description

Plots a Walter-Lieth climate diagram from monthly climatology values for a station over a specified time period.

Usage

```
climatogram_period(
  station = NULL,
  start = 1990,
  end = 2020,
  labels = "en",
  verbose = FALSE,
  ggplot2 = TRUE,
  ...
)
```

Arguments

station	A character string containing a station identifier. See aemet_stations() .
start	A numeric value specifying the start year in YYYY format.
end	A numeric value specifying the end year in YYYY format.
labels	A character string specifying the language for the x-axis month labels, such as "en" (English), "es" (Spanish) or "fr" (French).
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.
ggplot2	A logical value. If TRUE, the function uses ggclimat_walter_lieth() . If FALSE, it uses climatol::diagwl() .
...	Further arguments passed to climatol::diagwl() or ggclimat_walter_lieth() , depending on the value of ggplot2.

Value

A plot produced by [ggclimat_walter_lieth\(\)](#) or [climatol::diagwl\(\)](#), depending on ggplot2.

API key

Queries to the AEMET OpenData API require an API key. Use `aemet_api_key()` to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

References

Walter H, Lieth H (1967). *Klimadiagramm-Weltatlas*. VEB Gustav Fischer Verlag, Jena. Published in three installments, 1960-1967, <https://hdl.handle.net/2268.1/7079>.

Guijarro JA (2026). *climatol: Climate Tools (Series Homogenization and Derived Products)*. doi:10.32614/CRAN.package.climatol. R package version 4.5-0, <https://CRAN.R-project.org/package=climatol>.

See Also

Walter-Lieth climate diagrams: `climatogram_normal()`, `ggclimat_walter_lieth()`

Examples

```
climatogram_period("9434", start = 2015, end = 2020, labels = "en")
```

dms2decdegrees	<i>Convert DMS coordinates to decimal degrees</i>
----------------	---

Description

Converts degrees, minutes and seconds to decimal degrees.

Usage

```
dms2decdegrees(input = NULL)
```

```
dms2decdegrees_2(input = NULL)
```

Arguments

`input` A character string containing DMS coordinates.

Value

A [numeric](#) value.

Note

Code for `dms2decdegrees()` was adapted from the project at <https://github.com/SevillaR/aemet>.

See Also

Helper functions: [climaemet_news\(\)](#), [first_day_of_year\(\)](#)

Examples

```
dms2decdegrees("055245W")
dms2decdegrees_2("-3° 40' 37'")
```

first_day_of_year	<i>First and last day of a year</i>
-------------------	-------------------------------------

Description

Returns the first or last calendar day of a year.

Usage

```
first_day_of_year(year = NULL)

last_day_of_year(year = NULL)
```

Arguments

year A numeric year in YYYY format.

Value

A [character](#) string containing a date in YYYY-MM-DD format.

See Also

Helper functions: [climaemet_news\(\)](#), [dms2decdegrees\(\)](#)

Examples

```
first_day_of_year(2000)
last_day_of_year(2020)
```

`get_data_aemet`*Query the AEMET OpenData API*

Description

Retrieves data and metadata from AEMET and converts JSON responses to a [tibble](#) when possible.

Usage

```
get_data_aemet(apidest, verbose = FALSE)
```

```
get_metadata_aemet(apidest, verbose = FALSE)
```

Arguments

<code>apidest</code>	A character string containing the destination URL. See https://opendata.aemet.es/dist/index.html .
<code>verbose</code>	A logical value. If TRUE, displays information about the exchange between the client and server.

Value

A [tibble](#) when possible, otherwise a [raw](#) vector or a [character](#) string as provided by `httr2::resp_body_raw()` or `httr2::resp_body_string()`.

Source

<https://opendata.aemet.es/dist/index.html>.

See Also

`vignette("extending-climaemet", package = "climaemet")` provides usage examples.
AEMET OpenData API functions: [aemet_api_key\(\)](#), [aemet_detect_api_key\(\)](#)

Examples

```
# Run only when AEMET_API_KEY is detected.

url <- "/api/valores/climatologicos/inventarioestaciones/todasestaciones"

get_data_aemet(url)

# Metadata.

get_metadata_aemet(url)

# Get data from any API endpoint.
```

```
# Plain text.

plain <- get_data_aemet("/api/prediccion/nacional/hoy")

cat(plain)

# An image.

image <- get_data_aemet("/api/mapasygraficos/analisis")

# Write and read.
tmp <- tempfile(fileext = ".gif")

writeBin(image, tmp)

gganimate::gif_file(tmp)
```

ggclimat_walter_lieth *Walter-Lieth climate diagram with R*
*project.org/package=ggplot2***ggplot2**

Description

Plots a Walter-Lieth climate diagram for a station using **ggplot2**.

Usage

```
ggclimat_walter_lieth(
  dat,
  est = "",
  alt = NA,
  per = NA,
  mlab = "es",
  pcol = "#002F70",
  tcol = "#ff0000",
  pfc col = "#9BAEE2",
  sfcol = "#3C6FC4",
  shem = FALSE,
  p3line = FALSE,
  ...
)
```

Arguments

<code>dat</code>	A data frame containing monthly climatology data.
<code>est</code>	A character string with the climatological station name.
<code>alt</code>	A numeric value with the station altitude in meters.

<code>per</code>	A character string describing the averaging period.
<code>mlab</code>	Month labels for the x-axis. Use a two-letter language code, such as "en" or "es". See <code>readr::locale()</code> for details.
<code>pcol</code>	A color for precipitation.
<code>tcol</code>	A color for temperature.
<code>pfcol</code>	A fill color for probable frosts.
<code>sfcol</code>	A fill color for certain frosts.
<code>shem</code>	A logical value. If TRUE, plots a Southern Hemisphere station.
<code>p3line</code>	Set to TRUE to draw a supplementary precipitation line relative to three times the temperature (as suggested by Bogdan Rosca).
<code>...</code>	Further graphic arguments.

Details

See the details in `climatol::diagwl()`.

Climatology data must be passed as a 4 by 12 matrix or data frame of monthly data from January to December. Rows must contain mean precipitation, mean maximum daily temperature, mean minimum daily temperature and absolute monthly minimum temperature, in that order.

See `climaemet_9434_climatogram` for a sample dataset.

Value

A `ggplot` object.

References

Walter H, Lieth H (1967). *Klimadiagramm-Weltatlas*. VEB Gustav Fischer Verlag, Jena. Published in three installments, 1960-1967, <https://hdl.handle.net/2268.1/7079>.

Guijarro JA (2026). *climatol: Climate Tools (Series Homogenization and Derived Products)*. doi:10.32614/CRAN.package.climatol. R package version 4.5-0, <https://CRAN.R-project.org/package=climatol>.

See Also

- `climatol::diagwl()` provides the original diagram implementation.
- `readr::locale()` provides language-specific month labels.

Walter-Lieth climate diagrams: `climatogram_normal()`, `climatogram_period()`

Examples

```
library(ggplot2)

wl <- ggclimat_walter_lieth(
  climaemet::climaemet_9434_climatogram,
  alt = "249",
  per = "1981-2010",
```

```

    est = "Zaragoza Airport"
  )

w1

# Modify the ggplot object.

w1 + theme(
  plot.background = element_rect(fill = "grey80"),
  panel.background = element_rect(fill = "grey70"),
  axis.text.y.left = element_text(
    colour = "black",
    face = "italic"
  ),
  axis.text.y.right = element_text(
    colour = "black",
    face = "bold"
  )
)

```

ggstripes

Plot warming stripes

Description

Plots warming stripes with **ggplot2**. These graphics represent temperature change at a location over at least 70 years. Each stripe shows the annual average temperature at that station.

Usage

```

ggstripes(
  data,
  plot_type = "stripes",
  plot_title = "",
  n_temp = 11,
  col_pal = "RdBu",
  ...
)

```

Arguments

<code>data</code>	A data frame with date (year) and temperature (temp) variables.
<code>plot_type</code>	A character string specifying the plot type: "background", "stripes", "trend" or "animation".
<code>plot_title</code>	A character string for the plot title.
<code>n_temp</code>	The number of colors in the palette. Defaults to 11.
<code>col_pal</code>	A character string specifying an <code>hcl.pals()</code> color palette.
<code>...</code>	Further arguments passed to <code>ggplot2::theme()</code> .

Value

A `ggplot` object.

Note

Professor Ed Hawkins of the University of Reading developed the "warming stripes" concept to communicate climate change risks as simply as possible. For more details, see [ShowYourStripes](#).

See Also

`ggplot2::theme()` for additional arguments to `ggstripes()` and `climaemet_9434_temp`.

Warming stripes: `climatestripes_station()`

Examples

```
library(ggplot2)

data <- climaemet::climaemet_9434_temp

ggstripes(data, plot_title = "Zaragoza Airport") +
  labs(subtitle = "(1950-2020)")

ggstripes(data, plot_title = "Zaragoza Airport", plot_type = "trend") +
  labs(subtitle = "(1950-2020)")
```

ggwindrose

Plot a wind rose

Description

Plots a wind rose showing wind speed and direction with **ggplot2**.

Usage

```
ggwindrose(
  speed,
  direction,
  n_directions = 8,
  n_speeds = 5,
  speed_cuts = NA,
  col_pal = "GnBu",
  legend_title = "Wind speed (m/s)",
  calm_wind = 0,
  n_col = 1,
  facet = NULL,
  plot_title = "",
  stack_reverse = FALSE,
```

```
    ...
  )
```

Arguments

<code>speed</code>	A numeric vector of wind speeds.
<code>direction</code>	A numeric vector of wind directions.
<code>n_directions</code>	The number of direction bins (petals) to plot. Valid values are 4, 8 or 16.
<code>n_speeds</code>	The number of equally spaced wind speed bins to plot when <code>speed_cuts</code> is NA. Defaults to 5.
<code>speed_cuts</code>	A numeric vector with the cut points for the wind speed intervals or NA (default).
<code>col_pal</code>	A character string specifying an <code>hcl.pals()</code> color palette.
<code>legend_title</code>	A character string or expression for the legend title.
<code>calm_wind</code>	The upper wind speed limit considered calm. Defaults to 0.
<code>n_col</code>	The number of plot columns. Defaults to 1.
<code>facet</code>	A character or factor vector of facets used to plot wind roses.
<code>plot_title</code>	A character string for the plot title.
<code>stack_reverse</code>	A logical value. If TRUE, reverses the stack order of speed cuts. See Examples .
<code>...</code>	Further arguments (ignored).

Value

A `ggplot` object.

See Also

`ggplot2::theme()` for additional arguments to pass to `ggwindrose()` and `climaemet_9434_wind`.
Wind roses: `windrose_days()`, `windrose_period()`

Examples

```
library(ggplot2)

speed <- climaemet::climaemet_9434_wind$velmedia
direction <- climaemet::climaemet_9434_wind$dir

rose <- ggwindrose(
  speed = speed,
  direction = direction,
  speed_cuts = seq(0, 16, 4),
  legend_title = "Wind speed (m/s)",
  calm_wind = 0,
  n_col = 1,
  plot_title = "Zaragoza Airport"
)
rose + labs(
  subtitle = "2000-2020",
```

```

    caption = "Source: AEMET"
  )

  # Reverse the stack.

  ggwindrose(
    speed = speed,
    direction = direction,
    speed_cuts = seq(0, 16, 4),
    legend_title = "Wind speed (m/s)",
    calm_wind = 0,
    n_col = 1,
    plot_title = "Zaragoza Airport",
    stack_reverse = TRUE
  ) +
  labs(
    subtitle = "2000-2020",
    caption = "Source: AEMET"
  )

```

windrose_days

Wind rose for a range of days

Description

Plots a wind rose showing wind speed and direction at a station over a period of days.

Usage

```

windrose_days(
  station,
  start = "2000-12-01",
  end = "2000-12-31",
  n_directions = 8,
  n_speeds = 5,
  speed_cuts = NA,
  col_pal = "GnBu",
  calm_wind = 0,
  legend_title = "Wind speed (m/s)",
  verbose = FALSE
)

```

Arguments

station	A character string containing a station identifier. See aemet_stations() .
start	A character string containing the start date in YYYY-MM-DD format.
end	A character string containing the end date in YYYY-MM-DD format.

n_directions	The number of direction bins (petals) to plot. Valid values are 4, 8 or 16.
n_speeds	The number of equally spaced wind speed bins to plot when speed_cuts is NA. Defaults to 5.
speed_cuts	A numeric vector with the cut points for the wind speed intervals or NA (default).
col_pal	A character string specifying an hcl.pals() color palette.
calm_wind	The upper wind speed limit considered calm. Defaults to 0.
legend_title	A character string or expression for the legend title.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.

Value

A [ggplot](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

- [aemet_daily_clim\(\)](#) retrieves daily climatology data.
- [climaemet_9434_wind](#) provides example wind observations.

Wind roses: [ggwindrose\(\)](#), [windrose_period\(\)](#)

Examples

```
windrose_days("9434",
  start = "2000-12-01",
  end = "2000-12-31",
  speed_cuts = 4
)
```

windrose_period	<i>Wind rose for a range of years</i>
-----------------	---------------------------------------

Description

Plots a wind rose showing wind speed and direction at a station over a time period.

Usage

```

windrose_period(
  station,
  start = 2000,
  end = 2010,
  n_directions = 8,
  n_speeds = 5,
  speed_cuts = NA,
  col_pal = "GnBu",
  calm_wind = 0,
  legend_title = "Wind speed (m/s)",
  verbose = FALSE
)

```

Arguments

station	A character string containing a station identifier. See aemet_stations() .
start	A numeric value specifying the start year in YYYY format.
end	A numeric value specifying the end year in YYYY format.
n_directions	The number of direction bins (petals) to plot. Valid values are 4, 8 or 16.
n_speeds	The number of equally spaced wind speed bins to plot when speed_cuts is NA. Defaults to 5.
speed_cuts	A numeric vector with the cut points for the wind speed intervals or NA (default).
col_pal	A character string specifying an hcl.pals() color palette.
calm_wind	The upper wind speed limit considered calm. Defaults to 0.
legend_title	A character string or expression for the legend title.
verbose	A logical value. If TRUE, displays information about the exchange between the client and server.

Value

A [ggplot](#) object.

API key

Queries to the AEMET OpenData API require an API key. Use [aemet_api_key\(\)](#) to set it globally. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See [httr2::req_timeout\(\)](#) for details.

See Also

- [aemet_daily_period\(\)](#) retrieves daily climatology data by period.
- [climaemet_9434_wind](#) provides example wind observations.

Wind roses: [ggwindrose\(\)](#), [windrose_days\(\)](#)

Examples

```
# Do not run this example.
if (FALSE) {
  # Downloading data may take a few minutes.
  windrose_period("9434",
    start = 2000, end = 2010,
    speed_cuts = 4
  )
}
```

Index

- * **aemet_api**
 - aemet_api_key, 6
 - aemet_detect_api_key, 11
 - get_data_aemet, 35
 - * **aemet_auth**
 - aemet_api_key, 6
 - aemet_detect_api_key, 11
 - * **aemet_low**
 - get_data_aemet, 35
 - * **alerts**
 - aemet_alert_zones, 3
 - aemet_alerts, 4
 - * **climatogram**
 - climatogram_normal, 31
 - climatogram_period, 32
 - ggclimat_walter_lieth, 36
 - * **climatology**
 - aemet_daily_clim, 9
 - aemet_extremes_clim, 12
 - aemet_monthly_clim, 22
 - aemet_normal_clim, 24
 - * **datasets**
 - aemet_munic, 23
 - climaemet_9434_climatogram, 27
 - climaemet_9434_temp, 27
 - climaemet_9434_wind, 28
 - * **forecasts**
 - aemet_forecast_beaches, 13
 - aemet_forecast_daily, 14
 - aemet_forecast_fires, 17
 - aemet_forecast_tidy, 19
 - * **helpers**
 - climaemet_news, 29
 - dms2decdegrees, 33
 - first_day_of_year, 34
 - * **locations**
 - aemet_alert_zones, 3
 - aemet_beaches, 8
 - aemet_stations, 25
 - * **observations**
 - aemet_alerts, 4
 - aemet_last_obs, 20
 - * **stripes**
 - climatestripes_station, 29
 - ggstripes, 38
 - * **wind**
 - ggwindrose, 39
 - windrose_days, 41
 - windrose_period, 42
- aemet_alert_zones, 3
- aemet_alert_zones(), 5, 8, 26
- aemet_alerts, 4
- aemet_alerts(), 3, 21
- aemet_api_key, 6
- aemet_api_key(), 5, 7, 10–13, 15, 18, 21, 23, 25, 26, 30, 31, 33, 35, 42, 43
- aemet_beaches, 8
- aemet_beaches(), 3, 13, 14, 26
- aemet_daily(aemet_daily_clim), 9
- aemet_daily_clim, 9
- aemet_daily_clim(), 9, 10, 12, 23, 25, 26, 42
- aemet_daily_period(aemet_daily_clim), 9
- aemet_daily_period(), 9, 10, 43
- aemet_daily_period_all(aemet_daily_clim), 9
- aemet_daily_period_all(), 9, 10
- aemet_detect_api_key, 11
- aemet_detect_api_key(), 7, 11, 35
- aemet_extremes_clim, 12
- aemet_extremes_clim(), 10, 23, 25, 26
- aemet_forecast_beaches, 13
- aemet_forecast_beaches(), 8, 15, 18, 19
- aemet_forecast_daily, 14
- aemet_forecast_daily(), 14, 18, 19, 24
- aemet_forecast_fires, 17
- aemet_forecast_fires(), 14, 15, 19
- aemet_forecast_hourly(aemet_forecast_daily), 14

, 19, 24
, 19
, 14, 15, 18, 19

 (aemet_forecast_tidy), 19
, 19
, 20
, 5, 26
, 22
, 10, 12, 25, 26

 (aemet_monthly_clim), 22
, 22

 (aemet_monthly_clim), 22
, 22
, 15, 23, 27, 28
 (aemet_normal_clim), 24
, 24
, 10, 12, 23, 26

 (aemet_normal_clim), 24
, 24

 (aemet_detect_api_key), 11
, 11
, 25
, 3, 8, 10, 12, 21–25,
 30–32, 41, 43
as.Date(), 10

character, 11, 19, 34, 35
cli::cli_progress_bar(), 5, 10, 12, 13, 15,
 21, 23, 25
climaemet_9434_climatogram, 24, 27, 28,
 32, 37
climaemet_9434_temp, 24, 27, 27, 28, 30, 39
climaemet_9434_wind, 24, 27, 28, 28, 40, 42,
 43
climaemet_news, 29
climaemet_news(), 34
climatestripes_station, 29
climatestripes_station(), 28, 39
climatogram_normal, 31
climatogram_normal(), 27, 33, 37
climatogram_period, 32
climatogram_period(), 27, 32, 37
climatol::diagwl(), 31, 32, 37

data frame, 27
dms2decdegrees, 33
dms2decdegrees(), 29, 33, 34
dms2decdegrees_2 (dms2decdegrees), 33

factor(), 18
first_day_of_year, 34
first_day_of_year(), 29, 34

get_data_aemet, 35
get_data_aemet(), 7, 11
get_metadata_aemet (get_data_aemet), 35
get_metadata_aemet(), 5, 10, 12, 13, 15, 18,
 21, 23, 25
ggclimat_walter_lieth, 36
ggclimat_walter_lieth(), 27, 31–33
ggplot, 30, 37, 39, 40, 42, 43
ggplot2::theme(), 38–40
ggstripes, 30, 38
ggstripes(), 28, 30, 39
ggstripes_station
 (climatestripes_station), 29
ggwindrose, 39
ggwindrose(), 28, 40, 42, 43

hcl.pals(), 30, 38, 40, 42, 43
httr2::req_timeout(), 5, 7, 10, 12, 13, 15,
 18, 21, 23, 25, 26, 30, 31, 33, 42, 43
httr2::resp_body_raw(), 35
httr2::resp_body_string(), 35

last_day_of_year (first_day_of_year), 34
list, 12
logical, 11

mapSpain::esp_codelist, 5
mapSpain::esp_dict_region_code(), 5
mapSpain::esp_get_munic(), 15

NULL, 7, 29
numeric, 33

raw, 35
readr::locale(), 37

sf, 3, 5, 8, 10, 12, 13, 15, 21–23, 25, 26
SpatRaster, 17, 18

tempdir(), 3, 8, 26
terra::coltab(), 18

`terra::time()`, 18
`tibble`, 3, 5, 8, 10, 12, 13, 15, 18, 19, 21–23,
25, 26, 28, 35

`windrose_days`, 41
`windrose_days()`, 28, 40, 43
`windrose_period`, 42
`windrose_period()`, 28, 40, 42