Package 'climaemet'

June 3, 2026

Type Package

Title Tools for AEMET Climate Data

Version 1.6.0

Description Download meteorological and climate data from the Spanish Meteorological Agency (AEMET) directly in R using the AEMET API. Create scientific visualizations, including climate charts, climate time series trend analyses, temperature and precipitation anomaly maps, warming stripes and climatograms.

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), tibble (>= 3.0.3), 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), withr (>= 3.0.0)

VignetteBuilder quarto

Config/Needs/website cpp11, crosstalk, devtools, geoR, gifski, gstat, leaflet, reactable, scales, tidyterra, tidyverse, usethis, styler

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Config/testthat/parallel false

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

NeedsCompilation no

Author Manuel Pizarro [aut, cph] (ORCID: <https://orcid.org/0000-0002-6981-0154>), Diego Hernangómez [aut, cre] (ORCID: <https://orcid.org/0000-0001-8457-4658>), Gema Fernández-Avilés [aut] (ORCID: <https://orcid.org/0000-0001-5934-1916>), AEMET [cph] (ROR: <https://ror.org/04kxf1r09>)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2026-06-03 14:20:07 UTC

Contents

`aemet_alerts` *AEMET meteorological alerts*

Description

[Experimental] Get current meteorological alerts.

Usage

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

Arguments

`ccaa`	Character vector with names for autonomous communities or `NULL` to get all autonomous communities.
`lang`	Language of the results. It can be `"es"` (Spanish) or `"en"` (English).
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.
`extract_metadata`	Logical. If `TRUE`, the output is a tibble with the description of the fields. See also `get_metadata_aemet()`.
`progress`	Logical. Displays a `cli::cli_progress_bar()` object. If `verbose = TRUE`, it will not be displayed.

Value

A tibble or a **sf** object.

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

aemet_alert_zones(). See also mapSpain::esp_codelist, mapSpain::esp_dict_region_code() to get the names of the autonomous communities.

AEMET data functions: aemet_alert_zones(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_last_obs(), aemet_monthly, aemet_normal, aemet_stations()

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_alert_zones` *AEMET alert zones*

Description

Get AEMET alert zones.

Usage

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

Arguments

`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.

Details

The first result of each call per session is temporarily cached in `tempdir()` to avoid unnecessary API calls.

Value

A tibble or a **sf** object.

Source

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

See Also

`aemet_alerts()`

AEMET data functions: `aemet_alerts()`, `aemet_beaches()`, `aemet_daily_clim()`, `aemet_extremes_clim()`, `aemet_forecast_beaches()`, `aemet_forecast_daily()`, `aemet_forecast_fires()`, `aemet_last_obs()`, `aemet_monthly`, `aemet_normal`, `aemet_stations()`

Examples

```
library(tibble)
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(dplyr)
library(ggplot2)

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

# Coast zones are identified by a "C" in COD_Z.
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_api_key *Install an AEMET API key*

Description

This function stores your AEMET API key on your local machine so it can be called securely without being stored in your code.

Alternatively, you can install the API key manually:

- Run `Sys.setenv(AEMET_API_KEY = "Your_Key")`. You will need to run this command in each session (similar to `install = FALSE`).
- Write this line in your `.Renviron` file: `AEMET_API_KEY = "Your_Key"` (same behavior as `install = TRUE`). This stores your API key permanently.

Usage

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

Arguments

apikey	The AEMET API key formatted in quotes. A key can be acquired at https://opendata.aemet.es/centrodedescargas/inicio. You can install several API keys as a character vector. See **Details**.
overwrite	If TRUE, overwrites an existing AEMET_API_KEY already set on your local machine.
install	If TRUE, installs the key on your local machine for use in future sessions. Defaults to FALSE.

Details

You can pass several apikey values as a character vector c(api1, api2). In this case, multiple AEMET_API_KEY values are generated. 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

Invisibly returns NULL.

Note

To locate your API key on your local machine, run tools::R_user_dir("climaemet", "config").

See Also

AEMET API key helpers: aemet_detect_api_key()

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 *AEMET beaches*

Description

Get AEMET beaches.

Usage

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

Arguments

verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
return_sf	Logical. If TRUE, the function returns an sf spatial object. If FALSE (the default value), it returns a tibble. The **sf** package must be installed.

Details

The first result of the API call in each session is temporarily cached in `tempdir()` to avoid unnecessary API calls.

Value

A tibble or a **sf** object.

API key

You need to set your API key globally using `aemet_api_key()`. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

See Also

`aemet_forecast_beaches()`

AEMET data functions: `aemet_alert_zones()`, `aemet_alerts()`, `aemet_daily_clim()`, `aemet_extremes_clim()`, `aemet_forecast_beaches()`, `aemet_forecast_daily()`, `aemet_forecast_fires()`, `aemet_last_obs()`, `aemet_monthly`, `aemet_normal`, `aemet_stations()`

Examples

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

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

```
identical(beaches, beaches2)

# Select and map beaches
library(dplyr)
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` *Daily/annual climatology values*

Description

Get climatology values for a station or for all the available stations. Note that `aemet_daily_period()` and `aemet_daily_period_all()` are shortcuts of `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	Character string with station identifier code(s) (see aemet_stations()) or "all" for all the stations.
start, end	Character strings with start and end dates. See **Details**.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
return_sf	Logical. If TRUE, the function returns an sf spatial object. If FALSE (the default value), it returns a tibble. The **sf** package must be installed.
extract_metadata	Logical. If TRUE, the output is a tibble with the description of the fields. See also get_metadata_aemet().
progress	Logical. Displays a cli::cli_progress_bar() object. If verbose = TRUE, it will not be displayed.

Details

start and end arguments must be:

- For aemet_daily_clim(): A Date object or a string with format YYYY-MM-DD ("2020-12-31") coercible with as.Date().
- For aemet_daily_period() and aemet_daily_period_all(): A string representing the year(s) to be extracted: "2020", "2018".

Value

A tibble or a **sf** object.

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

aemet_api_key(), as.Date()

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_last_obs(), aemet_monthly, aemet_normal, aemet_stations()

Examples

```
library(tibble)
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 whether an AEMET API key is present for the current session*

Description

Detects whether an API key is available in the current session:

- If an API key is already set as an environment variable, it is preserved.
- If no environment variable is set and an API key has been stored permanently via aemet_api_key(), it is loaded.

Usage

```
aemet_detect_api_key(...)

aemet_show_api_key(...)
```

Arguments

...	Ignored.

Value

TRUE or FALSE. aemet_show_api_key() displays your stored API keys.

See Also

AEMET API key helpers: aemet_api_key()

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

Get recorded extreme values for a station.

Usage

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

Arguments

`station`	Character string with station identifier code(s). See `aemet_stations()`.
`parameter`	Character string with the parameter to retrieve: temperature (`"T"`), precipitation (`"P"`) or wind (`"V"`).
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.
`extract_metadata`	Logical. If `TRUE`, the output is a tibble with the description of the fields. See also `get_metadata_aemet()`.
`progress`	Logical. Displays a `cli::cli_progress_bar()` object. If `verbose = TRUE`, it will not be displayed.

Value

A tibble or a **sf** object. If the function encounters a parsing error, it returns the results as a `list()` object.

API key

You need to set your API key globally using `aemet_api_key()`. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

See Also

`aemet_api_key()`

AEMET data functions: `aemet_alert_zones()`, `aemet_alerts()`, `aemet_beaches()`, `aemet_daily_clim()`, `aemet_forecast_beaches()`, `aemet_forecast_daily()`, `aemet_forecast_fires()`, `aemet_last_obs()`, `aemet_monthly`, `aemet_normal`, `aemet_stations()`

Examples

```
library(tibble)
obs <- aemet_extremes_clim(c("9434", "3195"))
glimpse(obs)
```

---

aemet_forecast_beaches

Beach forecast dataset

Description

Get daily weather forecasts for one or more beaches. Beach codes can be accessed with `aemet_beaches()`.

Usage

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

Arguments

`x`	Character vector with beach codes to extract. See `aemet_beaches()`.
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.
`extract_metadata`	Logical. If `TRUE`, the output is a tibble with the description of the fields. See also `get_metadata_aemet()`.
`progress`	Logical. Displays a `cli::cli_progress_bar()` object. If `verbose = TRUE`, it will not be displayed.

Value

A tibble or a sf object.

API key

You need to set your API key globally using `aemet_api_key()`. 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.

AEMET data functions: `aemet_alert_zones()`, `aemet_alerts()`, `aemet_beaches()`, `aemet_daily_clim()`, `aemet_extremes_clim()`, `aemet_forecast_daily()`, `aemet_forecast_fires()`, `aemet_last_obs()`, `aemet_monthly`, `aemet_normal`, `aemet_stations()`

Forecast functions: `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 in beaches of Palma (ES)",
    subtitle = "3-day forecast",
    x = "Date",
    y = "Temperature (Celsius)",
    color = "Beach"
  )
```

---

aemet_forecast_daily *Municipality forecast dataset*

Description

Get 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	Character vector with municipality codes to extract. For convenience, **climaemet** provides these data in the aemet_munic dataset (see municipio field) as of January 2024.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
extract_metadata	Logical. If TRUE, the output is a tibble with the description of the fields. See also get_metadata_aemet().
progress	Logical. Displays a cli::cli_progress_bar() object. If verbose = TRUE, it will not be displayed.

Details

Forecasts provided by the AEMET 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 a simple tibble describing the value of each field of the forecast is returned.

Value

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

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

aemet_munic for municipality codes and **mapSpain** package for working with `sf` objects of municipalities (see `mapSpain::esp_get_munic()` and **Examples**).

AEMET data functions: `aemet_alert_zones()`, `aemet_alerts()`, `aemet_beaches()`, `aemet_daily_clim()`, `aemet_extremes_clim()`, `aemet_forecast_beaches()`, `aemet_forecast_fires()`, `aemet_last_obs()`, `aemet_monthly`, `aemet_normal`, `aemet_stations()`

Forecast functions: `aemet_forecast_beaches()`, `aemet_forecast_fires()`, `aemet_forecast_tidy()`

Examples

```
# Select a city
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 = "º")
  ) +
  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 with mapSpain
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

Get a SpatRaster with the daily wildfire risk level.

Usage

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

Arguments

area	Forecast area. Accepted values are: • "p" for mainland Spain and Balearic Islands. • "c" for Canary Islands.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
extract_metadata	Logical. If TRUE, the output is a tibble with the description of the fields. See also get_metadata_aemet().

Details

The SpatRaster provides six factor() levels with the following meaning:

- "1": Very low risk.
- "2": Low risk.
- "3": Moderate risk.
- "4": High risk.
- "5": Very high risk.
- "6": Extreme risk.

The resulting object has several layers, each one representing the forecast for the upcoming 7 days. It also has additional attributes provided by the **terra** package, such as terra::time() and terra::coltab().

Value

A tibble or a SpatRaster.

Source

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

See Also

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_last_obs(), aemet_monthly, aemet_normal, aemet_stations()

Forecast functions: aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_tidy()

Examples

```
aemet_forecast_fires(extract_metadata = TRUE)

# Extract alerts.
alerts <- aemet_forecast_fires()

alerts

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

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

# 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 *Helper functions for extracting forecasts*

Description

[Experimental] Helpers for aemet_forecast_daily() and aemet_forecast_hourly():

- aemet_forecast_vars_available() extracts the values available in the dataset.
- aemet_forecast_tidy() produces a tibble with the forecast for var.

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	Name of the desired variable to extract.

Value

A character vector from aemet_forecast_vars_available() or a tibble from aemet_forecast_tidy().

See Also

Forecast functions: 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. This needs lubridate to adjust time zones.
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, ocaso) |>
  distinct_all() |>
  group_by(nombre) |>
  mutate(
    ocaso_end = lubridate::force_tz(
      as.POSIXct(fecha) + ocaso,
      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 = ocaso_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 = "º")) +
  labs(
    x = "", y = "",
    title = "Forecast: Temperature",
    subtitle = paste("Forecast produced on", format(temp_end$elaborado[1],
      usetz = TRUE
    ))
  )
```

`aemet_last_obs` *Last observation values for a station*

Description

Get last observation values for a station.

Usage

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

Arguments

`station`	Character string with station identifier code(s) (see `aemet_stations()`) or `"all"` for all the stations.
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.

extract_metadata
: Logical. If TRUE, the output is a tibble with the description of the fields. See also get_metadata_aemet().

progress
: Logical. Displays a cli::cli_progress_bar() object. If verbose = TRUE, it will not be displayed.

Value

A tibble or a sf object.

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_monthly, aemet_normal, aemet_stations()

Examples

```
library(tibble)
obs <- aemet_last_obs(c("9434", "3195"))
glimpse(obs)
```

---

aemet_monthly *Monthly/annual climatology values*

Description

Get monthly/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	Character string with station identifier code(s). See aemet_stations().
year	Numeric value with year (format: YYYY).
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
return_sf	Logical. If TRUE, the function returns an sf spatial object. If FALSE (the default value), it returns a tibble. The **sf** package must be installed.
extract_metadata	Logical. If TRUE, the output is a tibble with the description of the fields. See also get_metadata_aemet().
progress	Logical. Displays a cli::cli_progress_bar() object. If verbose = TRUE, it will not be displayed.
start	Numeric value with the start year (format: YYYY).
end	Numeric value with the end year (format: YYYY).

Value

A tibble or a **sf** object.

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_last_obs(), aemet_normal, aemet_stations()

Examples

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

---

aemet_munic *Municipalities of Spain*

Description

A tibble with all municipalities of Spain as defined by the INE (Instituto Nacional de Estadistica) 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(), aemet_forecast_hourly()

Included datasets: climaemet_9434_climatogram, climaemet_9434_temp, climaemet_9434_wind

Examples

```
data(aemet_munic)

aemet_munic
```

aemet_normal *Normal climatology values*

Description

Get normal climatology values for a station, or for all stations with `aemet_normal_clim_all()`. Standard climatology covers 1981 to 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`	Character string with station identifier code(s) (see `aemet_stations()`) or `"all"` for all the stations.
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`return_sf`	Logical. If `TRUE`, the function returns an `sf` spatial object. If `FALSE` (the default value), it returns a tibble. The **sf** package must be installed.
`extract_metadata`	Logical. If `TRUE`, the output is a tibble with the description of the fields. See also `get_metadata_aemet()`.
`progress`	Logical. Displays a `cli::cli_progress_bar()` object. If `verbose = TRUE`, it will not be displayed.

Value

A tibble or a **sf** object.

API key

You need to set your API key globally using `aemet_api_key()`. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

Note

Code modified from project https://github.com/SevillaR/aemet.

See Also

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_last_obs(), aemet_monthly, aemet_stations()

Examples

```
library(tibble)
obs <- aemet_normal_clim(c("9434", "3195"))
glimpse(obs)
```

---

aemet_stations *AEMET stations*

Description

Get AEMET stations.

Usage

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

Arguments

verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
return_sf	Logical. If TRUE, the function returns an sf spatial object. If FALSE (the default value), it returns a tibble. The **sf** package must be installed.

Details

The first result of the API call in each session is temporarily cached in tempdir() to avoid unnecessary API calls.

Value

A tibble or a **sf** object.

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

Note

Code modified from project https://github.com/SevillaR/aemet.

See Also

AEMET data functions: aemet_alert_zones(), aemet_alerts(), aemet_beaches(), aemet_daily_clim(), aemet_extremes_clim(), aemet_forecast_beaches(), aemet_forecast_daily(), aemet_forecast_fires(), aemet_last_obs(), aemet_monthly, aemet_normal

Examples

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

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

identical(stations, stations2)
```

---

climaemet_9434_climatogram

Climatogram data for Zaragoza Airport ("9434"), 1981-2010

Description

Normal data for Zaragoza Airport (1981-2010). This is an example dataset used to plot climatograms.

Format

A data.frame with columns 1 to 12 (months) and rows:

p_mes_md Precipitation (mm).

tm_max_md Maximum temperature (Celsius).

tm_min_md Minimum temperature (Celsius).

ta_min_md Absolute monthly minimum temperature (Celsius).

Source

AEMET.

See Also

ggclimat_walter_lieth(), climatogram_period(), climatogram_normal()

Included datasets: aemet_munic, climaemet_9434_temp, climaemet_9434_wind

Climatogram functions: climatogram_normal(), climatogram_period(), ggclimat_walter_lieth()

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 (Celsius).

Source

AEMET.

See Also

Included datasets: `aemet_munic`, `climaemet_9434_climatogram`, `climaemet_9434_wind`

Warming stripes functions: `climatestripes_station()`, `ggstripes()`

Examples

```
data(climaemet_9434_temp)
```

`climaemet_9434_wind` *Wind conditions for Zaragoza Airport ("9434"), 2000-2020*

Description

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

Format

A tibble with columns:

fecha Date of observation.

dir Wind directions (0-360).

velmedia Average wind speed (km/h).

Source

AEMET.

See Also

Included datasets: `aemet_munic`, `climaemet_9434_climatogram`, `climaemet_9434_temp`

Wind functions: `ggwindrose()`, `windrose_days()`, `windrose_period()`

Examples

```
data(climaemet_9434_wind)
```

---

climatestripes_station

Station climate stripes plot

Description

Plot a climate stripes graph for a station.

Usage

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

Arguments

`station`	Character string with station identifier code(s). See `aemet_stations()`.
`start`	Numeric value with the start year (format: `YYYY`).
`end`	Numeric value with the end year (format: `YYYY`).
`with_labels`	Character string, either `"yes"` or `"no"`, to indicate whether plot labels are displayed.

verbose — Logical. If TRUE, provides information about the flow of information between the client and server.

... — Arguments passed on to ggstripes

n_temp Numeric value with the number of colors of the palette. (default 11).

col_pal Character string indicating the name of the hcl.pals() color palette to be used for plotting.

Value

A **ggplot2** object. See help("ggplot2").

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

Note

"Warming stripes" charts are a conceptual idea of Professor Ed Hawkins (University of Reading) and are specifically designed to be as simple as possible and to warn about climate change risks. For more details, see ShowYourStripes.

See Also

ggstripes()

Plotting functions: climatogram_normal(), climatogram_period(), ggclimat_walter_lieth(), ggstripes(), ggwindrose(), windrose_days(), windrose_period()

Warming stripes functions: climaemet_9434_temp, 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` *Walter & Lieth climatic diagram from normal climatology values*

Description

Plot a Walter & Lieth climatic diagram from normal climatology values for a station. This climatogram is a great way to show a summary of climate conditions for a place over a time period (1981-2010).

Usage

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

Arguments

`station`	Character string with station identifier code(s). See `aemet_stations()`.
`labels`	Character string with month labels for the x-axis: `"en"` (English), `"es"` (Spanish), `"fr"` (French), etc.
`verbose`	Logical. If `TRUE`, provides information about the flow of information between the client and server.
`ggplot2`	Logical. 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.

API key

You need to set your API key globally using `aemet_api_key()`. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

Note

The code is based on code from the CRAN package **climatol**.

References

- Walter, H. K., Harnickell, E., Lieth, F. H. H., & Rehder, H. (1967). *Klimadiagramm-weltatlas*. Jena: Fischer, 1967.
- Guijarro J. A. (2023). *climatol: Climate Tools (Series Homogenization and Derived Products)*. R package version 4.0.0, https://climatol.eu.

See Also

Plotting functions: climatestripes_station(), climatogram_period(), ggclimat_walter_lieth(), ggstripes(), ggwindrose(), windrose_days(), windrose_period()

Climatogram functions: climaemet_9434_climatogram, climatogram_period(), ggclimat_walter_lieth()

Examples

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

---

climatogram_period *Walter & Lieth climatic diagram for a time period*

Description

Plot a Walter & Lieth climatic diagram from monthly climatology values for a station. This climatogram is a great way to show a summary of climate conditions for a place over a specific time period.

Usage

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

Arguments

station	Character string with station identifier code(s). See aemet_stations().
start	Numeric value with the start year (format: YYYY).
end	Numeric value with the end year (format: YYYY).
labels	Character string with month labels for the x-axis: "en" (English), "es" (Spanish), "fr" (French), etc.

verbose	Logical. If TRUE, provides information about the flow of information between the client and server.
ggplot2	Logical. 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.

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

Note

The code is based on code from the CRAN package **climatol**.

References

- Walter, H. K., Harnickell, E., Lieth, F. H. H., & Rehder, H. (1967). *Klimadiagramm-weltatlas*. Jena: Fischer, 1967.
- Guijarro J. A. (2023). *climatol: Climate Tools (Series Homogenization and Derived Products)*. R package version 4.0.0, https://climatol.eu.

See Also

Plotting functions: climatestripes_station(), climatogram_normal(), ggclimat_walter_lieth(), ggstripes(), ggwindrose(), windrose_days(), windrose_period()

Climatogram functions: climaemet_9434_climatogram, climatogram_normal(), ggclimat_walter_lieth()

Examples

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

dms2decdegrees *Convert* dms *format to decimal degrees*

Description

Convert degrees, minutes and seconds to decimal degrees.

Usage

```
dms2decdegrees(input = NULL)

dms2decdegrees_2(input = NULL)
```

Arguments

input	Character string with dms coordinates.

Value

A numeric value.

Note

Code for dms2decdegrees() was modified from project https://github.com/SevillaR/aemet.

See Also

Helper functions: climaemet_news(), first_day_of_year()

Examples

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

first_day_of_year *First and last day of a year*

Description

Get the first and last day of a year.

Usage

```
first_day_of_year(year = NULL)

last_day_of_year(year = NULL)
```

Arguments

year	Numeric value with year (format: YYYY).

Value

Character string with date (format: YYYY-MM-DD).

See Also

Helper functions: climaemet_news(), dms2decdegrees()

Examples

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

---

get_data_aemet *Client tool for the AEMET API*

Description

Client tool to retrieve data and metadata from AEMET and convert JSON to a tibble.

Usage

```
get_data_aemet(apidest, verbose = FALSE)

get_metadata_aemet(apidest, verbose = FALSE)
```

Arguments

apidest	Character string with a destination URL. See https://opendata.aemet.es/dist/index.html.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.

Value

A tibble (if possible) or the results of the query as provided by httr2::resp_body_raw() or httr2::resp_body_string().

Source

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

See Also

See examples of how to use these functions in vignette("extending-climaemet").

Examples

```
# Run this example only if 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 climatic diagram with* R*hrefhttps://CRAN.R-project.org/package=ggplot2***ggplot2**

Description

Plot a Walter & Lieth climatic diagram for a station. This function is an updated version of `climatol::diagwl()`, by Jose A. Guijarro.

Usage

```
ggclimat_walter_lieth(
  dat,
  est = "",
  alt = NA,
  per = NA,
  mlab = "es",
  pcol = "#002F70",
```

```
  tcol = "#ff0000",
  pfcol = "#9BAEE2",
  sfcol = "#3C6FC4",
  shem = FALSE,
  p3line = FALSE,
  ...
)
```

Arguments

dat	Monthly climate data for which the diagram will be plotted.
est	Name of the climatological station.
alt	Altitude of the climatological station.
per	Period used to compute the averages.
mlab	Month labels for the x-axis. Use a 2-digit language code ("en", "es", etc.). See readr::locale() for details.
pcol	Color for precipitation.
tcol	Color for temperature.
pfcol	Fill color for probable frosts.
sfcol	Fill color for sure frosts.
shem	Set to TRUE for southern hemisphere stations.
p3line	Set to TRUE to draw a supplementary precipitation line relative to three times the temperature (as suggested by Bogdan Rosca).
...	Further graphic arguments.

Details

See the details in climatol::diagwl().

Climate data must be passed as a 4 x 12 matrix or data.frame of monthly data (January to December) in the following order:

- Row 1: Mean precipitation.
- Row 2: Mean maximum daily temperature.
- Row 3: Mean minimum daily temperature.
- Row 4: Absolute monthly minimum temperature.

See climaemet_9434_climatogram for a sample dataset.

Value

A **ggplot2** object. See help("ggplot2").

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

References

- Walter, H. K., Harnickell, E., Lieth, F. H. H., & Rehder, H. (1967). *Klimadiagramm-weltatlas*. Jena: Fischer, 1967.
- Guijarro J. A. (2023). *climatol: Climate Tools (Series Homogenization and Derived Products)*. R package version 4.0.0, https://climatol.eu.

See Also

`climatol::diagwl()`, `readr::locale()`

Plotting functions: `climatestripes_station()`, `climatogram_normal()`, `climatogram_period()`, `ggstripes()`, `ggwindrose()`, `windrose_days()`, `windrose_period()`

Climatogram functions: `climaemet_9434_climatogram`, `climatogram_normal()`, `climatogram_period()`

Examples

```
library(ggplot2)

wl <- ggclimat_walter_lieth(
  climaemet::climaemet_9434_climatogram,
  alt = "249",
  per = "1981-2010",
  est = "Zaragoza Airport"
)

wl

# Since it is a ggplot object, we can modify it.

wl + 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 *Warming stripes graph*

Description

Plot different "climate stripes" or "warming stripes" using **ggplot2**. These graphics are visual representations of the change in temperature as measured in each location over the past 70-100+ years. Each stripe represents the temperature in that station averaged over a year.

Usage

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

Arguments

`data`	A data.frame with date (`year`) and temperature (`temp`) variables.
`plot_type`	Plot type. Accepted values are `"background"`, `"stripes"`, `"trend"` or `"animation"`.
`plot_title`	Character string to be used for the plot title.
`n_temp`	Numeric value with the number of colors of the palette. (default `11`).
`col_pal`	Character string indicating the name of the `hcl.pals()` color palette to be used for plotting.
`...`	Further arguments passed to `ggplot2::theme()`.

Value

A **ggplot2** object. See `help("ggplot2")`.

API key

You need to set your API key globally using `aemet_api_key()`. Query timeout can be controlled with `options(climaemet_timeout = 60)` (default value). See `httr2::req_timeout()` for details.

Note

"Warming stripes" charts are a conceptual idea of Professor Ed Hawkins (University of Reading) and are specifically designed to be as simple as possible and to warn about climate change risks. For more details, see ShowYourStripes.

See Also

`climatestripes_station()`, `ggplot2::theme()` for more possible arguments to pass to `ggstripes()`.

Plotting functions: `climatestripes_station()`, `climatogram_normal()`, `climatogram_period()`, `ggclimat_walter_lieth()`, `ggwindrose()`, `windrose_days()`, `windrose_period()`

Warming stripes functions: `climaemet_9434_temp`, `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 *Windrose (speed/direction) diagram*

Description

Plot a windrose showing the wind speed and direction using **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

`speed`	Numeric vector of wind speeds.
`direction`	Numeric vector of wind directions.
`n_directions`	Numeric value with the number of direction bins to plot (petals on the rose). Valid values are `4`, `8` or `16`.
`n_speeds`	Numeric value with the number of equally spaced wind speed bins to plot. This is used if `speed_cuts` is `NA` (default `5`).
`speed_cuts`	Numeric vector with the cut points for the wind speed intervals, or `NA` (default).

col_pal	Character string indicating the name of the hcl.pals() color palette to be used for plotting.
legend_title	Character string to be used for the legend title.
calm_wind	Numeric value with the upper limit for wind speed that is considered calm (default 0).
n_col	The number of columns of plots (default 1).
facet	Character or factor vector of facets used to plot windroses.
plot_title	Character string to be used for the plot title.
stack_reverse	Logical. If TRUE, the stack order of speed cuts is inverted. See **Examples**.
...	Further arguments (ignored).

Value

A **ggplot2** object. See help("ggplot2").

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

ggplot2::theme() for more possible arguments to pass to ggwindrose().

Plotting functions: climatestripes_station(), climatogram_normal(), climatogram_period(), ggclimat_walter_lieth(), ggstripes(), windrose_days(), windrose_period()

Wind functions: climaemet_9434_wind, 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` *Windrose (speed/direction) diagram of a station over a period of days*

Description

Plot a windrose showing the wind speed and direction for 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`	Character string with station identifier code(s) (see `aemet_stations()`) or `"all"` for all the stations.
`start`	Character string with the start date (format: `"YYYY-MM-DD"`).
`end`	Character string with the end date (format: `"YYYY-MM-DD"`).
`n_directions`	Numeric value with the number of direction bins to plot (petals on the rose). Valid values are `4`, `8` or `16`.

n_speeds	Numeric value with the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	Numeric vector with the cut points for the wind speed intervals, or NA (default).
col_pal	Character string indicating the name of the hcl.pals() color palette to be used for plotting.
calm_wind	Numeric value with the upper limit for wind speed that is considered calm (default 0).
legend_title	Character string to be used for the legend title.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.

Value

A **ggplot2** object. See help("ggplot2").

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

aemet_daily_clim()

Plotting functions: climatestripes_station(), climatogram_normal(), climatogram_period(), ggclimat_walter_lieth(), ggstripes(), ggwindrose(), windrose_period()

Wind functions: climaemet_9434_wind, ggwindrose(), windrose_period()

Examples

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

---

windrose_period *Windrose (speed/direction) diagram of a station over a time period*

Description

Plot a windrose showing the wind speed and direction for 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	Character string with station identifier code(s). See aemet_stations().
start	Numeric value with the start year (format: YYYY).
end	Numeric value with the end year (format: YYYY).
n_directions	Numeric value with the number of direction bins to plot (petals on the rose). Valid values are 4, 8 or 16.
n_speeds	Numeric value with the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	Numeric vector with the cut points for the wind speed intervals, or NA (default).
col_pal	Character string indicating the name of the hcl.pals() color palette to be used for plotting.
calm_wind	Numeric value with the upper limit for wind speed that is considered calm (default 0).
legend_title	Character string to be used for the legend title.
verbose	Logical. If TRUE, provides information about the flow of information between the client and server.

Value

A **ggplot2** object. See help("ggplot2").

API key

You need to set your API key globally using aemet_api_key(). Query timeout can be controlled with options(climaemet_timeout = 60) (default value). See httr2::req_timeout() for details.

See Also

aemet_daily_period()

Plotting functions: climatestripes_station(), climatogram_normal(), climatogram_period(), ggclimat_walter_lieth(), ggstripes(), ggwindrose(), windrose_days()

Wind functions: climaemet_9434_wind, 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