

# Package ‘climaemet’

May 8, 2026

**Type** Package

**Title** Climate AEMET Tools

**Version** 1.5.1

**Description** Tools to download the climatic data of the Spanish Meteorological Agency (AEMET) directly from R using their API and create scientific graphs (climate charts, trend analysis of climate time series, temperature and precipitation anomalies maps, warming stripes graphics, climatograms, etc.).

**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), 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)

**VignetteBuilder** quarto

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

**Config/testthat/edition** 3

**Config/testthat/parallel** false

**Copyright** © AEMET. See file COPYRIGHTS

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**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](mailto:diego.hernangomezherrero@gmail.com)>  
**Repository** CRAN  
**Date/Publication** 2026-03-23 21:30:03 UTC

## Contents

aemet_alerts . . . . .	3
aemet_alert_zones . . . . .	5
aemet_api_key . . . . .	6
aemet_beaches . . . . .	7
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 . . . . .	21
aemet_monthly . . . . .	22
aemet_munic . . . . .	24
aemet_normal . . . . .	25
aemet_stations . . . . .	26
climaemet_9434_climatogram . . . . .	27
climaemet_9434_temp . . . . .	28
climaemet_9434_wind . . . . .	28
climatestripes_station . . . . .	29
climatogram_normal . . . . .	30
climatogram_period . . . . .	32
dms2decdegrees . . . . .	33
first_day_of_year . . . . .	34
get_data_aemet . . . . .	35
ggclimat_walter_lieth . . . . .	36
ggstripes . . . . .	38
ggwindrose . . . . .	40

<i>aemet_alerts</i>	3
<i>windrose_days</i> . . . . .	42
<i>windrose_period</i> . . . . .	43

**Index** **45**

---

<i>aemet_alerts</i>	<i>AEMET Meteorological warnings</i>
---------------------	--------------------------------------

---

**Description**

**[Experimental]** Get a database of current meteorological alerts.

**Usage**

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

**Arguments**

- ccaa*            A vector of names for autonomous communities or NULL to get all the autonomous communities.
- lang*            Language of the results. It can be "es" (Spanish) or "en" (English).
- verbose*        Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
- return\_sf*      Logical TRUE or FALSE. Should the function return an *sf* spatial object? If FALSE (the default value) it returns a *tibble*. Note that you need to have the *sf* package installed.
- extract\_metadata*    Logical TRUE/FALSE. On TRUE the output is a *tibble* with the description of the fields. See also *get\_metadata\_aemet()*.
- progress*        Logical, display a *cli::cli\_progress\_bar()* object. If *verbose* = TRUE won't 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.

Other `aemet_api_data`: `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 names of CCAAs
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
)

# If any alert
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 TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <code>sf</code> spatial object? If FALSE (the default value) it returns a <code>tibble</code> . Note that you need to have the <code>sf</code> package 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/el tiempo/prediccion/avisos/ayuda>. See also Annex 2 and Annex 3 docs, linked in that page.

## See Also

[aemet\\_alerts\(\)](#)

Other `aemet_api_data`: [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 beaches
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 will store 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 `.Renviro`n 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 API key provided to you from the AEMET formatted in quotes. A key can be acquired at <a href="https://opendata.aemet.es/centrodedescargas/inicio">https://opendata.aemet.es/centrodedescargas/inicio</a> . You can install several API keys as a character vector; see <b>Details</b> .
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 `rappdirs::user_cache_dir("climaemet", "R")`.

**See Also**

Other aemet\_auth: [aemet\\_detect\\_api\\_key\(\)](#)

**Examples**

```
# Don't run these examples!

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

  # You can check it with:
  Sys.getenv("AEMET_API_KEY")
}

if (FALSE) {
  # If you need to overwrite an existing key:
  aemet_api_key("222222abc", overwrite = TRUE, install = TRUE)

  # You can 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 TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <code>sf</code> spatial object? If FALSE (the default value) it returns a <code>tibble</code> . Note that you need to have the <code>sf</code> package installed.

**Details**

The first result of the API call on each session is (temporarily) cached in the assigned `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()`.

**See Also**

[aemet\\_forecast\\_beaches\(\)](#)

Other `aemet_api_data`: [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	<i>Daily/annual climatology values</i>
------------------	--

---

## 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 <a href="#">aemet_stations()</a> ) or "all" for all the stations.
start, end	Character string with start and end date. See <b>Details</b> .
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <a href="#">sf</a> spatial object? If FALSE (the default value) it returns a <a href="#">tibble</a> . Note that you need to have the <a href="#">sf</a> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <a href="#">cli::cli_progress_bar()</a> object. If verbose = TRUE won't be displayed.

**Details**

start and end arguments should 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\(\)](#).

**See Also**

[aemet\\_api\\_key\(\)](#), [as.Date\(\)](#)

Other [aemet\\_api\\_data](#): [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 if 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

Other `aemet_auth`: [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 <a href="#">aemet_stations()</a> ).
parameter	Character string as temperature ("T"), precipitation ("P") or wind ("V") parameter.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <a href="#">sf</a> spatial object? If FALSE (the default value) it returns a <a href="#">tibble</a> . Note that you need to have the <a href="#">sf</a> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <a href="#">cli::cli_progress_bar()</a> object. If verbose = TRUE won't 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\(\)](#).

**See Also**

[aemet\\_api\\_key\(\)](#)

Other aemet\_api\_data: [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

*Forecast database for beaches*

---

**Description**

Get a database of daily weather forecasts for a beach. Beach database can be accessed with [aemet\\_beaches\(\)](#).

**Usage**

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

**Arguments**

x	A vector of beaches codes to extract. See <a href="#">aemet_beaches()</a> .
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <i>sf</i> spatial object? If FALSE (the default value) it returns a <i>tibble</i> . Note that you need to have the <i>sf</i> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <i>tibble</i> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <code>cli::cli_progress_bar()</code> object. If verbose = TRUE won't be displayed.

**Value**

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

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**See Also**

[aemet\\_beaches\(\)](#) for beaches codes.

Other `aemet_api_data`: [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\(\)](#)

Other 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 in beaches of Palma (ES)",
    subtitle = "Forecast 3-days",
    x = "Date",
    y = "Temperature (Celsius)",
    color = "Beach"
  )
```

---

`aemet_forecast_daily` *Forecast database by municipality*

---

**Description**

Get a database of daily or hourly weather forecasts for a given municipality.

**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 vector of municipality codes to extract. For convenience, <b>climaemet</b> provides this data on the dataset <a href="#">aemet_munic</a> (see <code>municipio</code> field) as of January 2024.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <code>cli::cli_progress_bar()</code> object. If <code>verbose = TRUE</code> won't be displayed.

**Details**

Forecasts format provided by the AEMET API have a complex structure. Although **climaemet** returns a [tibble](#), each forecasted value is provided as a nested [tibble](#). [aemet\\_forecast\\_tidy\(\)](#) helper function 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](#). Forecasted 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\(\)](#).

**See Also**

[aemet\\_munic](#) for municipality codes and **mapSpain** package for working with `sf` objects of municipalities (see [mapSpain::esp\\_get\\_munic\(\)](#) and **Examples**).

Other aemet\_api\_data: `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()`

Other forecasts: `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)

# Vars 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
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 = "a")
    ) +
    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 fires forecast*

---

## Description

Get a [SpatRaster](#) as provided by [terra](#) with the daily meteorological risk level for wildfires.

**Usage**

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

**Arguments**

area	The area, being: <ul style="list-style-type: none"> <li>• "p" for Mainland Spain and Balearic Islands.</li> <li>• "c" for Canary Islands.</li> </ul>
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .

**Details**

The `SpatRaster` provides 5 (`factor()`) levels with the following meaning:

- "1": Low risk.
- "2": Moderate risk.
- "3": High risk.
- "4": Very high risk.
- "5": 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` object.

**Source**

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

**See Also**

Other `aemet_api_data`: `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()`

Other forecasts: `aemet_forecast_beaches()`, `aemet_forecast_daily()`, `aemet_forecast_tidy()`

## Examples

```
aemet_forecast_fires(extract_metadata = TRUE)

# Extract alerts
alerts <- aemet_forecast_fires()

alerts

# Nice plotting with terra
library(terra)
plot(alerts, all_levels = TRUE)

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

# SpatVector
cyl <- vect(cyl)

fires_cyl <- crop(alerts, 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 on 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 database extracted with <a href="#">aemet_forecast_daily()</a> or <a href="#">aemet_forecast_hourly()</a> .
var	Name of the desired var to extract

**Value**

A vector of characters (`aemet_forecast_vars_available()`) or a `tibble` (`aemet_forecast_tidy()`).

**See Also**

Other forecasts: `aemet_forecast_beaches()`, `aemet_forecast_daily()`, `aemet_forecast_fires()`

**Examples**

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

# Vars available
aemet_forecast_vars_available(hourly)

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

library(dplyr)
# Make hour - Need lubridate to adjust timezones
temp_end <- temp |>
  mutate(
    forecast_time = lubridate::force_tz(
      as.POSIXct(fecha) + hora,
      tz = "Europe/Madrid"
    )
  )

# Add also 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 = .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

*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 <a href="#">aemet_stations()</a> ) or "all" for all the stations.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <a href="#">sf</a> spatial object? If FALSE (the default value) it returns a <a href="#">tibble</a> . Note that you need to have the <a href="#">sf</a> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <a href="#">cli::cli_progress_bar()</a> object. If verbose = TRUE won't be displayed.

**Value**

A [tibble](#) or a [sf](#) object

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**See Also**

Other `aemet_api_data`: [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*

---

**Description**

Get monthly/annual climatology values for a station or all the stations. `aemet_monthly_period()` and `aemet_monthly_period_all()` allows 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 <a href="#">aemet_stations()</a> ).
year	Numeric value as date (format: YYYY).
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <a href="#">sf</a> spatial object? If FALSE (the default value) it returns a <a href="#">tibble</a> . Note that you need to have the <a href="#">sf</a> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <a href="#">tibble</a> with the description of the fields. See also <a href="#">get_metadata_aemet()</a> .
progress	Logical, display a <a href="#">cli::cli_progress_bar()</a> object. If verbose = TRUE won't be displayed.
start	Numeric value as start year (format: YYYY).
end	Numeric value as 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\(\)](#).

### See Also

Other `aemet_api_data`: [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
```

*Data set with all the municipalities of Spain*

---

## Description

A [tibble](#) with all the 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 code 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\(\)](#)

Other dataset: [climaemet\\_9434\\_climatogram](#), [climaemet\\_9434\\_temp](#), [climaemet\\_9434\\_wind](#)

## Examples

```
data(aemet_munic)
```

```
aemet_munic
```

---

aemet_normal	<i>Normal climatology values</i>
--------------	----------------------------------

---

### Description

Get normal climatology values for a station (or all the stations with `aemet_normal_clim_all()`). Standard climatology from 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 <code>aemet_stations()</code> ) or "all" for all the stations.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <code>sf</code> spatial object? If FALSE (the default value) it returns a <code>tibble</code> . Note that you need to have the <code>sf</code> package installed.
extract_metadata	Logical TRUE/FALSE. On TRUE the output is a <code>tibble</code> with the description of the fields. See also <code>get_metadata_aemet()</code> .
progress	Logical, display a <code>cli::cli_progress_bar()</code> object. If verbose = TRUE won't be displayed.

### Value

A `tibble` or a `sf` object.

### API Key

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

**Note**

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

**See Also**

Other aemet\_api\_data: [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	<i>AEMET stations</i>
----------------	-----------------------

---

**Description**

Get AEMET stations.

**Usage**

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

**Arguments**

verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
return_sf	Logical TRUE or FALSE. Should the function return an <code>sf</code> spatial object? If FALSE (the default value) it returns a <code>tibble</code> . Note that you need to have the <code>sf</code> package installed.

**Details**

The first result of the API call on each session is (temporarily) cached in the assigned `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()`.

**Note**

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

**See Also**

Other aemet\_api\_data: [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") period 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\(\)](#)

Other dataset: [aemet\\_munic](#), [climaemet\\_9434\\_temp](#), [climaemet\\_9434\\_wind](#)

Other climatogram: [climatogram\\_normal\(\)](#), [climatogram\\_period\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#)

## Examples

```
data(climaemet_9434_climatogram)
```

---

climaemet_9434_temp	<i>Average annual temperatures for Zaragoza Airport ("9434") period 1950-2020</i>
---------------------	---

---

## 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

Other dataset: [aemet\\_munic](#), [climaemet\\_9434\\_climatogram](#), [climaemet\\_9434\\_wind](#)

Other stripes: [climatestripes\\_station\(\)](#), [ggstripes\(\)](#)

## Examples

```
data(climaemet_9434_temp)
```

---

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

---

## 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**

Other dataset: [aemet\\_munic](#), [climaemet\\_9434\\_climatogram](#), [climaemet\\_9434\\_temp](#)

Other wind: [ggwindrose\(\)](#), [windrose\\_days\(\)](#), [windrose\\_period\(\)](#)

**Examples**

```
data(climaemet_9434_wind)
```

---

```
climatestripes_station
```

*Station climate stripes graph*

---

**Description**

Plot climate stripes graph for a station.

**Usage**

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

**Arguments**

<code>station</code>	Character string with station identifier code(s) (see <a href="#">aemet_stations()</a> ).
<code>start</code>	Numeric value as start year (format: YYYY).
<code>end</code>	Numeric value as end year (format: YYYY).
<code>with_labels</code>	Character string as yes/no. Indicates whether to use labels for the graph or not.

verbose Logical TRUE/FALSE. Provides information about the flow of information between the client and server.

... Arguments passed on to [ggstripes](#)

n\_temp Numeric value as 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

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**See Also**

[ggstripes\(\)](#)

Other aemet\_plots: [climatogram\\_normal\(\)](#), [climatogram\\_period\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#), [ggstripes\(\)](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#), [windrose\\_period\(\)](#)

Other stripes: [climaemet\\_9434\\_temp](#), [ggstripes\(\)](#)

**Examples**

```
# Don't run example
if (FALSE) {
  # Data download 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 of a Walter & Lieth climatic diagram from normal climatology data for a station. This climatogram are great for showing 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 <a href="#">aemet_stations()</a> ).
labels	Character string as month labels for the X axis: "en" (english), "es" (spanish), "fr" (french), etc.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
ggplot2	TRUE/FALSE. On TRUE the function uses <a href="#">ggclimat_walter_lieth()</a> , if FALSE uses <a href="#">climatol::diagwl()</a> .
...	Further arguments to <a href="#">climatol::diagwl()</a> or <a href="#">ggclimat_walter_lieth()</a> , depending on the value of <b>ggplot2</b> .

**Value**

A plot.

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**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**

Other aemet\_plots: [climatestripes\\_station\(\)](#), [climatogram\\_period\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#), [ggstripes\(\)](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#), [windrose\\_period\(\)](#)

Other climatogram: [climaemet\\_9434\\_climatogram](#), [climatogram\\_period\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#)

**Examples**

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

---

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

---

**Description**

Plot of a Walter & Lieth climatic diagram from monthly climatology data for a station. This climatogram are great for showing 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 <a href="#">aemet_stations()</a> ).
start	Numeric value as start year (format: YYYY).
end	Numeric value as end year (format: YYYY).
labels	Character string as month labels for the X axis: "en" (english), "es" (spanish), "fr" (french), etc.
verbose	Logical TRUE/FALSE. Provides information about the flow of information between the client and server.
ggplot2	TRUE/FALSE. On TRUE the function uses <a href="#">ggclimat_walter_lieth()</a> , if FALSE uses <a href="#">climatol::diagwl()</a> .
...	Further arguments to <a href="#">climatol::diagwl()</a> or <a href="#">ggclimat_walter_lieth()</a> , depending on the value of <b>ggplot2</b> .

**Value**

A plot.

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**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**

Other aemet\_plots: [climatestripes\\_station\(\)](#), [climatogram\\_normal\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#), [ggstripes\(\)](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#), [windrose\\_period\(\)](#)

Other climatogram: [climaemet\\_9434\\_climatogram](#), [climatogram\\_normal\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#)

**Examples**

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

---

dms2decdegrees

*Converts dms format to decimal degrees*

---

**Description**

Converts degrees, minutes and seconds to decimal degrees.

**Usage**

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

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

**Arguments**

input            Character string as dms coordinates.

**Value**

A numeric value.

**Note**

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

**See Also**

Other helpers: [climaemet\\_news\(\)](#), [first\\_day\\_of\\_year\(\)](#)

**Examples**

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

---

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

---

**Description**

Get first and last day of year.

**Usage**

```
first_day_of_year(year = NULL)
```

```
last_day_of_year(year = NULL)
```

**Arguments**

year                    Numeric value as year (format: YYYY).

**Value**

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

**See Also**

Other helpers: [climaemet\\_news\(\)](#), [dms2decdegrees\(\)](#)

**Examples**

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

---

get_data_aemet	<i>Client tool for AEMET API</i>
----------------	----------------------------------

---

### Description

Client tool to get data and metadata from AEMET and convert json to [tibble](#).

### Usage

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

### Arguments

apidest	Character string as destination URL. See <a href="https://opendata.aemet.es/dist/index.html">https://opendata.aemet.es/dist/index.html</a> .
verbose	Logical TRUE/FALSE. 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

Some examples on how to use these functions on `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)

# We can 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 and Lieth climatic diagram on R*  
*<https://CRAN.R-project.org/package=ggplot2>***ggplot2**

---

## Description

Plot of a Walter and Lieth climatic diagram of 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",
  pfc col = "#9BAEE2",
  sfcol = "#3C6FC4",
  shem = FALSE,
  p3line = FALSE,
  ...
)
```

## Arguments

<code>dat</code>	Monthly climatic data for which the diagram will be plotted.
<code>est</code>	Name of the climatological station.
<code>alt</code>	Altitude of the climatological station.

<code>per</code>	Period on which the averages have been computed.
<code>mlab</code>	Month labels for the X axis. Use 2-digit language code ("en", "es", etc.). See <a href="#">readr::locale()</a> for info.
<code>pcol</code>	Color for precipitation.
<code>tcol</code>	Color for temperature.
<code>pfcol</code>	Fill color for probable frosts.
<code>sfcol</code>	Fill color for sure frosts.
<code>shem</code>	Set to TRUE for southern hemisphere stations.
<code>p3line</code>	Set to TRUE to draw a supplementary precipitation line referenced to three times the temperature (as suggested by Bogdan Rosca).
<code>...</code>	Other graphic arguments

### Details

See Details on [climatol::diagwl\(\)](#).

Climatic data must be passed as a 4x12 matrix or `data.frame` of monthly (January to December) data, 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\(\)](#).

### References

- Walter, H. K., Harnickell, E., Lieth, F. H. H., & Rehder, H. (1967). *Klimadiagramm-weltatlas*. Jena: Fischer, 1967.

### See Also

[climatol::diagwl\(\)](#), [readr::locale\(\)](#)

Other `aemet_plots`: [climatestripes\\_station\(\)](#), [climatogram\\_normal\(\)](#), [climatogram\\_period\(\)](#), [ggstripes\(\)](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#), [windrose\\_period\(\)](#)

Other `climatogram`: [climaemet\\_9434\\_climatogram](#), [climatogram\\_normal\(\)](#), [climatogram\\_period\(\)](#)

**Examples**

```

library(ggplot2)

w1 <- ggclimat_walter_lieth(
  climaemet::climaemet_9434_climatogram,
  alt = "249",
  per = "1981-2010",
  est = "Zaragoza Airport"
)

w1

# As it is a ggplot object we can modify it

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

*Warming stripes graph*


---

**Description**

Plot different "climate stripes" or "warming stripes" using **ggplot2**. This 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

<code>data</code>	a data.frame with <code>date(year)</code> and <code>temperature(temp)</code> variables.
<code>plot_type</code>	plot type (with labels, background, stripes with line trend and animation). Accepted values are "background", "stripes", "trend" or "animation".
<code>plot_title</code>	character string to be used for the graph title.
<code>n_temp</code>	Numeric value as the number of colors of the palette. (default 11).
<code>col_pal</code>	Character string indicating the name of the <code>hcl.pals()</code> color palette to be used for plotting.
<code>...</code>	further arguments passed to <code>ggplot2::theme()</code> .

## Value

A **ggplot2** object

## API Key

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

## 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 alert about risks of climate change. For more details see [ShowYourStripes](#).

## See Also

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

Other `aemet_plots`: `climatestripes_station()`, `climatogram_normal()`, `climatogram_period()`, `ggclimat_walter_lieth()`, `ggwindrose()`, `windrose_days()`, `windrose_period()`

Other stripes: `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 as the number of direction bins to plot (petals on the rose). Valid values are 4, 8 or 16.
n_speeds	Numeric value as the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	Numeric vector containing the cut points for the wind speed intervals, or NA (default).
col_pal	Character string indicating the name of the <code>hcl.pals()</code> color palette to be used for plotting.
legend_title	Character string to be used for the legend title.
calm_wind	Numeric value as 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 the facets used to plot the various 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 <b>Examples</b> .
...	further arguments (ignored).

**Value**

A **ggplot2** object.

**API Key**

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

**See Also**

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

Other `aemet` plots: `climatestripes_station()`, `climatogram_normal()`, `climatogram_period()`, `ggclimat_walter_lieth()`, `ggstripes()`, `windrose_days()`, `windrose_period()`

Other `wind`: `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 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 days period*


---

### Description

Plot a windrose showing the wind speed and direction for a station over a days period.

### 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 <a href="#">aemet_stations()</a> ) or "all" for all the stations.
start	Character string as start date (format: "YYYY-MM-DD").
end	Character string as end date (format: "YYYY-MM-DD").
n_directions	Numeric value as the number of direction bins to plot (petals on the rose). Valid values are 4, 8 or 16.
n_speeds	Numeric value as the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	Numeric vector containing the cut points for the wind speed intervals, or NA (default).
col_pal	Character string indicating the name of the <a href="#">hcl.pals()</a> color palette to be used for plotting.
calm_wind	Numeric value as 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 TRUE/FALSE. Provides information about the flow of information between the client and server.

**Value**

A **ggplot2** object.

**API Key**

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

**See Also**

`aemet_daily_clim()`

Other `aemet_plots`: `climatestripes_station()`, `climatogram_normal()`, `climatogram_period()`, `ggclimat_walter_lieth()`, `ggstripes()`, `ggwindrose()`, `windrose_period()`

Other `wind`: `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 <a href="#">aemet_stations()</a> ) or "all" for all the stations.
start	Numeric value as start year (format: YYYY).
end	Numeric value as end year (format: YYYY).
n_directions	Numeric value as the number of direction bins to plot (petals on the rose). Valid values are 4, 8 or 16.
n_speeds	Numeric value as the number of equally spaced wind speed bins to plot. This is used if speed_cuts is NA (default 5).
speed_cuts	Numeric vector containing the cut points for the wind speed intervals, or NA (default).
col_pal	Character string indicating the name of the <a href="#">hcl.pals()</a> color palette to be used for plotting.
calm_wind	Numeric value as 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 TRUE/FALSE. Provides information about the flow of information between the client and server.

**Value**

A **ggplot2** object

**API Key**

You need to set your API Key globally using [aemet\\_api\\_key\(\)](#).

**See Also**

[aemet\\_daily\\_period\(\)](#)

Other aemet\_plots: [climatestripes\\_station\(\)](#), [climatogram\\_normal\(\)](#), [climatogram\\_period\(\)](#), [ggclimat\\_walter\\_lieth\(\)](#), [ggstripes\(\)](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#)

Other wind: [climaemet\\_9434\\_wind](#), [ggwindrose\(\)](#), [windrose\\_days\(\)](#)

**Examples**

```
# Don't run example
if (FALSE) {
  # Data download may take a few minutes...
  windrose_period("9434",
    start = 2000, end = 2010,
    speed_cuts = 4
  )
}
```

# Index

- \* **aemet\_api\_data**
    - aemet\_alert\_zones, 5
    - aemet\_alerts, 3
    - aemet\_beaches, 7
    - aemet\_daily\_clim, 9
    - aemet\_extremes\_clim, 12
    - aemet\_forecast\_beaches, 13
    - aemet\_forecast\_daily, 14
    - aemet\_forecast\_fires, 17
    - aemet\_last\_obs, 21
    - aemet\_monthly, 22
    - aemet\_normal, 25
    - aemet\_stations, 26
  - \* **aemet\_api**
    - get\_data\_aemet, 35
  - \* **aemet\_auth**
    - aemet\_api\_key, 6
    - aemet\_detect\_api\_key, 11
  - \* **aemet\_plots**
    - climatestripes\_station, 29
    - climatogram\_normal, 30
    - climatogram\_period, 32
    - ggclimat\_walter\_lieth, 36
    - ggstripes, 38
    - ggwindrose, 40
    - windrose\_days, 42
    - windrose\_period, 43
  - \* **climatogram**
    - climaemet\_9434\_climatogram, 27
    - climatogram\_normal, 30
    - climatogram\_period, 32
    - ggclimat\_walter\_lieth, 36
  - \* **dataset**
    - aemet\_munic, 24
    - climaemet\_9434\_climatogram, 27
    - climaemet\_9434\_temp, 28
    - climaemet\_9434\_wind, 28
  - \* **forecasts**
    - aemet\_forecast\_beaches, 13
    - aemet\_forecast\_daily, 14
    - aemet\_forecast\_fires, 17
    - aemet\_forecast\_tidy, 19
  - \* **forecast**
    - aemet\_munic, 24
  - \* **helpers**
    - dms2decdegrees, 33
    - first\_day\_of\_year, 34
  - \* **stripes**
    - climaemet\_9434\_temp, 28
    - climatestripes\_station, 29
    - ggstripes, 38
  - \* **wind**
    - climaemet\_9434\_wind, 28
    - ggwindrose, 40
    - windrose\_days, 42
    - windrose\_period, 43
- 
- aemet\_alert\_zones, 4, 5, 8, 10, 13, 14, 16, 18, 22, 23, 26, 27
  - aemet\_alert\_zones(), 4
  - aemet\_alerts, 3, 5, 8, 10, 13, 14, 16, 18, 22, 23, 26, 27
  - aemet\_alerts(), 5
  - aemet\_api\_key, 6, 11
  - aemet\_api\_key(), 8, 10–15, 22, 23, 25, 26, 30–32, 37, 39, 41, 43, 44
  - aemet\_beaches, 4, 5, 7, 10, 13, 14, 16, 18, 22, 23, 26, 27
  - aemet\_beaches(), 13, 14
  - aemet\_daily (aemet\_daily\_clim), 9
  - aemet\_daily\_clim, 4, 5, 8, 9, 13, 14, 16, 18, 22, 23, 26, 27
  - aemet\_daily\_clim(), 43
  - aemet\_daily\_period (aemet\_daily\_clim), 9
  - aemet\_daily\_period(), 44
  - aemet\_daily\_period\_all (aemet\_daily\_clim), 9
  - aemet\_detect\_api\_key, 7, 11

- aemet\_extremes\_clim, [4](#), [5](#), [8](#), [10](#), [12](#), [14](#), [16](#), [18](#), [22](#), [23](#), [26](#), [27](#)
- aemet\_forecast\_beaches, [4](#), [5](#), [8](#), [10](#), [13](#), [13](#), [16](#), [18](#), [20](#), [22](#), [23](#), [26](#), [27](#)
- aemet\_forecast\_beaches(), [8](#)
- aemet\_forecast\_daily, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [14](#), [18](#), [20](#), [22](#), [23](#), [26](#), [27](#)
- aemet\_forecast\_daily(), [19](#), [24](#)
- aemet\_forecast\_fires, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [16](#), [17](#), [20](#), [22](#), [23](#), [26](#), [27](#)
- aemet\_forecast\_hourly  
(aemet\_forecast\_daily), [14](#)
- aemet\_forecast\_hourly(), [19](#), [24](#)
- aemet\_forecast\_tidy, [14](#), [16](#), [18](#), [19](#)
- aemet\_forecast\_tidy(), [15](#), [19](#), [20](#)
- aemet\_forecast\_vars\_available  
(aemet\_forecast\_tidy), [19](#)
- aemet\_forecast\_vars\_available(), [19](#), [20](#)
- aemet\_last\_obs, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [16](#), [18](#), [21](#), [23](#), [26](#), [27](#)
- aemet\_monthly, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [16](#), [18](#), [22](#), [22](#), [26](#), [27](#)
- aemet\_monthly\_clim(aemet\_monthly), [22](#)
- aemet\_monthly\_period(aemet\_monthly), [22](#)
- aemet\_monthly\_period\_all  
(aemet\_monthly), [22](#)
- aemet\_munic, [15](#), [24](#), [27–29](#)
- aemet\_normal, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [16](#), [18](#), [22](#), [23](#), [25](#), [27](#)
- aemet\_normal\_clim(aemet\_normal), [25](#)
- aemet\_normal\_clim\_all(aemet\_normal), [25](#)
- aemet\_show\_api\_key  
(aemet\_detect\_api\_key), [11](#)
- aemet\_stations, [4](#), [5](#), [8](#), [10](#), [13](#), [14](#), [16](#), [18](#), [22](#), [23](#), [26](#), [26](#)
- aemet\_stations(), [10](#), [12](#), [21](#), [23](#), [25](#), [29](#), [31](#), [32](#), [42](#), [44](#)
- as.Date(), [10](#)
- cli::cli\_progress\_bar(), [3](#), [10](#), [12](#), [13](#), [15](#), [21](#), [23](#), [25](#)
- climaemet\_9434\_climatogram, [24](#), [27](#), [28](#), [29](#), [31](#), [33](#), [37](#)
- climaemet\_9434\_temp, [24](#), [27](#), [28](#), [29](#), [30](#), [39](#)
- climaemet\_9434\_wind, [24](#), [27](#), [28](#), [28](#), [41](#), [43](#), [44](#)
- climaemet\_news, [34](#)
- climatestripes\_station, [28](#), [29](#), [31](#), [33](#), [37](#), [39](#), [41](#), [43](#), [44](#)
- climatestripes\_station(), [39](#)
- climatogram\_normal, [27](#), [30](#), [30](#), [33](#), [37](#), [39](#), [41](#), [43](#), [44](#)
- climatogram\_normal(), [27](#)
- climatogram\_period, [27](#), [30](#), [31](#), [32](#), [37](#), [39](#), [41](#), [43](#), [44](#)
- climatogram\_period(), [27](#)
- climatol::diagwl(), [31](#), [32](#), [36](#), [37](#)
- dms2decdegrees, [33](#), [34](#)
- dms2decdegrees\_2(dms2decdegrees), [33](#)
- factor(), [18](#)
- first\_day\_of\_year, [34](#), [34](#)
- get\_data\_aemet, [35](#)
- get\_metadata\_aemet(get\_data\_aemet), [35](#)
- get\_metadata\_aemet(), [3](#), [10](#), [12](#), [13](#), [15](#), [18](#), [21](#), [23](#), [25](#)
- ggclimat\_walter\_lieth, [27](#), [30](#), [31](#), [33](#), [36](#), [39](#), [41](#), [43](#), [44](#)
- ggclimat\_walter\_lieth(), [27](#), [31](#), [32](#)
- ggplot2::theme(), [39](#), [41](#)
- ggstripes, [28](#), [30](#), [31](#), [33](#), [37](#), [38](#), [41](#), [43](#), [44](#)
- ggstripes(), [30](#)
- ggstripes\_station  
(climatestripes\_station), [29](#)
- ggwindrose, [29–31](#), [33](#), [37](#), [39](#), [40](#), [43](#), [44](#)
- hcl.pals(), [30](#), [39](#), [40](#), [42](#), [44](#)
- httr2::resp\_body\_raw(), [35](#)
- httr2::resp\_body\_string(), [35](#)
- last\_day\_of\_year(first\_day\_of\_year), [34](#)
- mapSpain::esp\_codelist, [4](#)
- mapSpain::esp\_dict\_region\_code(), [4](#)
- mapSpain::esp\_get\_munic(), [15](#)
- readr::locale(), [37](#)
- sf, [3](#), [5](#), [8](#), [10](#), [12](#), [13](#), [21](#), [23](#), [25](#), [26](#)
- SpatRaster, [17](#), [18](#)
- tempdir(), [5](#), [8](#), [26](#)
- terra::coltab(), [18](#)
- terra::time(), [18](#)
- tibble, [3](#), [5](#), [8](#), [10](#), [12–15](#), [18–26](#), [28](#), [29](#), [35](#)
- windrose\_days, [29–31](#), [33](#), [37](#), [39](#), [41](#), [42](#), [44](#)
- windrose\_period, [29–31](#), [33](#), [37](#), [39](#), [41](#), [43](#), [43](#)