Package: openmeteo (via r-universe)

October 31, 2024

Title Retrieve Weather Data from the Open-Meteo API

Version 0.2.4

Description A client for the Open-Meteo API that retrieves Open-Meteo weather data in a tidy format. No API key is required. The API specification is located at <https://open-meteo.com/en/docs>.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports httr, tibble, tidyr, tibblify, dplyr, yaml, testthat (>= 3.0.0)

Suggests httptest

Config/testthat/edition 3

Repository https://tpisel.r-universe.dev

RemoteUrl https://github.com/tpisel/openmeteo

RemoteRef HEAD

RemoteSha 10cc1fd7b7bb92c601b8fc4bd91494fa3e6f928c

Contents

`air_quality` *Retrieve air quality data from the Open-Meteo API*

Description

`air_quality()` calls the Open-Meteo Air Quality API to obtain pollutant, pollen, and particulate data. Historical and forecasted data is available.

Refer to the API documentation at: https://open-meteo.com/en/docs/air-quality-api

Usage

```
air_quality(
  location,
  start = NULL,
  end = NULL,
  hourly = NULL,
  timezone = "auto"
)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start`, `end`	Start and end dates in ISO 8601 (e.g. "2020-12-31"). If no dates are supplied, data for the next 5 days will be provided by default.
`hourly`	Required. An air quality variable accepted by the API, or list thereof. See details below.
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Details

You will need to specify at least one air quality variable, such as PM10 or Carbon Monoxide, that you want forecasted data for. These variables are sampled or aggregated at *hourly* intervals, and can be supplied as a list to request multiple variables over the same time period.

Example hourly air quality variables include:

Variable	Description
`pm10`	Particulate matter smaller than 10 micrometers across
`carbon_monoxide`	10m concentration in micrograms per cubic meter
`european_aqi`	European Air Quality Index
`us_aqi`	United States Air Quality Index
`dust`	Saharan dust particles 10m above ground

Full documentation for the forecast API is available at: https://open-meteo.com/en/docs/air-quality-api

Value

Requested air quality data for the given location and time, as a tidy tibble.

Examples

```
# obtain Carbon Monoxide levels for Beijing over the next 5 days
air_quality("Beijing", hourly = "carbon_monoxide")
```

---

climate_forecast *Retrieve climate change forecasts from the Open-Meteo API*

Description

`climate_forecast()` calls the Open-Meteo Climate Change Forecast API to obtain long-range weather projections from a range of climate models.

Refer to the API documentation at: https://open-meteo.com/en/docs/climate-api

Usage

```
climate_forecast(
  location,
  start,
  end,
  daily = NULL,
  response_units = NULL,
  model = NULL,
  downscaling = TRUE,
  timezone = "auto"
)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start, end`	Required. Future start and end dates in ISO 8601 (e.g. "2020-12-31").
`daily`	Required. A weather variable accepted by the API, or list thereof. See details below.
`response_units`	Supply to convert temperature, windspeed, or precipitation units. This defaults to: `list(temperature_unit = "celsius", windspeed_unit = "kmh", precipitation_unit = "mm"`
`model`	Supply to specify a climate model for forecasted values (refer to the API documentation).

`downscaling`	Enable (default) or disable statistical downscaling with ERA5-Land (10 km).
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Details

You will need to specify at least one weather variable, such as temperature, that you want projected forecasts for. The models currently only provide weather data aggregated at *daily* intervals. Multiple variables can be supplied as a list.

Example daily climate forecast variables include:

Variable	Description
`temperature_2m_max`	Maximum daily air temperature at 2 meters above ground
`precipitation_sum`	Sum of rain, showers, and snow over the preceding day
`windspeed_10m_max`	Maximum daily wind speed at 10 meters above ground

Different climate change models can be specified, which may differ in the weather variables predicted. Models include:

Model	Origin	Resolution
`EC_Earth3P_HR`	Europe	29 km
`FGOALS_f3_H`	China	28 km
`MRI_AGCM3_2_S`	Japan	20 km

For all models and their available fields, refer to the full documentation for the climate API at: https://open-meteo.com/en/docs/climate-api

Value

Requested climate forecast data for the given location and time period, as a tidy tibble.

Examples

```
# Obtain projected precipitation for the North Pole in 2050
climate_forecast(c(90, 0),
  "2050-06-01", "2050-07-01",
  daily = "precipitation_sum"
)

# Obtain projected temperatures for Madrid in 2050 in Fahrenheit, with ESMI1
climate_forecast("Madrid",
  "2050-06-01", "2050-07-01",
  daily = "temperature_2m_max",
  model = "MPI_ESM1_2_XR",
  response_units = list(temperature_unit = "fahrenheit")
)
```

geocode *Geocode a location using the Open-Meteo geocoding API*

Description

Call the Open-Meteo Geocoding API to retrieve co-ordinates and other information for a given place name. The closest n matching records can be requested.

Geocoding API documentation is available at: https://open-meteo.com/en/docs/geocoding-api.

Usage

```
geocode(location_name, n_results = 1, language = "en", silent = TRUE)
```

Arguments

location_name	Required. The location name to search for via fuzzy matching.
n_results	The number of matching locations provided in response, sorted by relevance (default 1, up to 100).
language	Desired response language for place names (lower-cased two-letter string, i.e. "en" for *London*,"fr" for *Londres*).
silent	If FALSE, the top match will be printed to the console, to aid in confirming the match is correct when used within other functions.

Value

Details for each matching location (latitude, longitude, elevation, population, timezone, and administrative areas)

Examples

```
# obtain co-ordinates of Sydney
gc <- geocode("Sydney")
sydney_coords <- c(gc$latitude, gc$longitude)
sydney_coords

# elevation of Kathmandu
geocode("kathmandu")$elevation

# 10 places named 'Paris'
geocode("paris", 10)
```

`marine_forecast` *Retrieve marine conditions data from the Open-Meteo API*

Description

`marine_forecast()` calls the Open-Meteo Marine Forecast API to obtain swell and wave data for a given location. Limited historical data is also available via this API.

Refer to the API documentation at: https://open-meteo.com/en/docs/marine-weather-api

Usage

```
marine_forecast(
  location,
  start = NULL,
  end = NULL,
  hourly = NULL,
  daily = NULL,
  response_units = NULL,
  timezone = "auto"
)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start, end`	Start and end dates in ISO 8601 (e.g. "2020-12-31"). If no dates are supplied, data for the next 7 days will be provided by default.
`hourly, daily`	At least one required. A marine weather variable accepted by the API, or list thereof. See details below.
`response_units`	Supply to convert response units for wave heights. This defaults to: `list(length_unit="metric") for`
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Details

You will need to specify at least one variable to retrieve, such as wave height, that you want data for. These variables are sampled or aggregated at *hourly* or *daily* intervals, and can be supplied as a list to request multiple variables over the same time period.

Example *Hourly* variables include:

Variable	Description
`wave_height`	Wave height of significant mean waves
`wind_wave_height`	Wave height of significant wind waves
`swell_wave_height`	Wave height of significant swell waves

`wind_wave_direction`	Mean direction of wind waves
`swell_wave_period`	Mean period between swell waves

Example *Daily* variables include:

Variable	Description
`wave_height_max`	Maximum wave height for mean waves
`wind_wave_direction_dominant`	Dominant wave direction of wind waves
`swell_wave_period_max`	Maximum wave period of swell waves

Full documentation for the marine API is available at: https://open-meteo.com/en/docs/marine-weather-api

Value

Requested marine conditions data for the given location and time, as a tidy tibble.

Examples

```
# Obtain maximum wave heights in Nazare, Portugal, over the next week
marine_forecast("Nazare", daily = "wave_height_max")
```

`openmeteo` *openmeteo: retrieve weather data from the Open-Meteo API*

Description

`openmeteo` provides functions for accessing the Open-Meteo weather API, enabling the desired weather data or forecasts to be retrieved in a tidy data format. An API key is *not* required to access the Open-Meteo API.

Open-Meteo provides several API endpoints through the following functions:

Core Weather APIs

- `weather_forecast()` - retrieve weather forecasts for a location
- `weather_history()` - retrieve historical weather observations for a location
- `weather_now()` - simple function to return current weather for a location
- `weather_variables()` - retrieve a shortlist of valid forecast or historical weather variables provided

Other APIs

- `geocode()` - return the co-ordinates and other data for a location name
- `climate_forecast()` - return long-range climate modelling for a location
- `river_discharge()` - return flow volumes for the nearest river

- `marine_forecast()` - return ocean conditions data for a location
- `air_quality()` - return air quality data for a location

Please review the API documentation at https://open-meteo.com/ for details regarding the data available, its types, units, and other caveats and considerations.

`river_discharge` *Retrieve river discharge data from the Open-Meteo API*

Description

`river_discharge()` calls the Open-Meteo Global Flood API to obtain simulated river discharge from the nearest river. Data obtained from the Global Flood Awareness System (GloFAS). Forecasts and historical data is available.

Refer to the API documentation at: https://open-meteo.com/en/docs/flood-api

Usage

```
river_discharge(location, start = NULL, end = NULL, daily = NULL, model = NULL)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start, end`	Start and end dates in ISO 8601 (e.g. "2020-12-31"). If no dates are supplied, data for the next 3 months will be provided by default.
`daily`	Required. A weather variable accepted by the API, or list thereof. See details below.
`model`	Supply to specify a model for forecasted values (defaults to latest GloFAS model).

Details

You will need to specify at least one river discharge variable to retrieve data for. These variables are sampled or aggregated at daily intervals, and can be supplied as a list to request multiple variables over the same time period.

Example daily forecast variables include:

Variable	Description
`river_discharge`	Daily river discharge rate in m³/s
`river_discharge_median`	Median over ensemble members (forecasts only)
`river_discharge_max`	Maximum over ensemble members (forecasts only)

Full documentation for the forecast API is available at: https://open-meteo.com/en/docs/flood-api

Value

Requested river discharge data (m³/s) for the given location and time period, as a tidy tibble.

Examples

```
# obtain historical flood data for Brisbane
river_discharge("Brisbane",
  "2022-01-01", "2022-02-01",
  daily = "river_discharge"
)
```

`weather_forecast` *Retrieve weather forecasts from the Open-Meteo API*

Description

`weather_forecast()` calls the Open-Meteo Weather Forecast API to obtain meteorological forecasts for a given location.

Refer to the API documentation at: https://open-meteo.com/en/docs

Usage

```
weather_forecast(
  location,
  start = NULL,
  end = NULL,
  hourly = NULL,
  daily = NULL,
  response_units = NULL,
  model = NULL,
  timezone = "auto"
)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start, end`	Start and end dates in ISO 8601 (e.g. "2020-12-31"). If no dates are supplied, data for the next 7 days will be provided by default.
`hourly, daily`	At least one required. A weather variable accepted by the API, or list thereof. See details below.
`response_units`	Supply to convert temperature, windspeed, or precipitation units. This defaults to: `list(temperature_unit = "celsius", windspeed_unit = "kmh", precipitation_unit = "mm"`

`model`	Supply to specify a model for forecasted values (defaults to autoselection of best model).
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Details

You will need to specify at least one weather variable, such as temperature, that you want forecasted data for. These variables are sampled or aggregated at *hourly* or *daily* intervals, and can be supplied as a list to request multiple variables over the same time period.

Example *Hourly* forecast variables include:

Variable	Description
`temperature_2m`	Air temperature at 2 meters above ground
`precipitation`	Sum of rain, showers, and snow over the preceding hour
`windspeed_10m`	Wind speed at 10 meters above ground
`cloudcover`	Total cloud cover as an area fraction
`pressure_msl`	Atmospheric air pressure at mean sea level

Example *Daily* forecast variables include:

Variable	Description
`temperature_2m_max`	Maximum daily air temperature at 2 meters above ground
`precipitation_sum`	Sum of rain, showers, and snow over the preceding day
`windspeed_10m_max`	Maximum daily wind speed at 10 meters above ground

Full documentation for the forecast API is available at: https://open-meteo.com/en/docs

You can also call `weather_variables()` to retrieve an (incomplete) shortlist of valid hourly and daily weather variables.

Value

Requested weather forecast data for the given location and time, as a tidy tibble.

Examples

```
# obtain temperature forecasts for the South Pole's next 7 days
weather_forecast(c(-90, 0), hourly = "temperature_2m")

# obtain temperature and precipitation forecasts for NYC in Imperial units
weather_forecast("nyc",
  hourly = c("temperature_2m", "precipitation"),
  response_units = list(
    temperature_unit = "fahrenheit",
    precipitation_unit = "inch"
  )
)
```

```
# will it rain tomorrow in Jakarta?
tomorrow <- Sys.Date() + 1
weather_forecast("jakarta", tomorrow, tomorrow, daily = "precipitation_sum")
```

`weather_history` *Retrieve historical weather data from the Open-Meteo API*

Description

`weather_history()` calls the Open-Meteo Historical Weather API to obtain weather data for a given location and historical time period.

Refer to the API documentation at: https://open-meteo.com/en/docs/historical-weather-api

Usage

```
weather_history(
  location,
  start,
  end,
  hourly = NULL,
  daily = NULL,
  response_units = NULL,
  model = NULL,
  timezone = "auto"
)
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`start, end`	Required. Start and end dates in ISO 8601 (e.g. "2020-12-31").
`hourly, daily`	At least one required. A weather variable accepted by the API, or list thereof. See details below.
`response_units`	Supply to convert temperature, windspeed, or precipitation units. This defaults to: `list(temperature_unit = "celsius", windspeed_unit = "kmh", precipitation_unit = "mm"`
`model`	Supply to specify a model for re-analysis.
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Details

You will need to specify at least one weather variable, such as temperature, that you want historical data for. These variables have been sampled or aggregated at *hourly* or *daily* intervals, and can be supplied as a list to request multiple variables over the same time period.

Example *Hourly* historical weather variables include:

Variable	Description
`temperature_2m`	Air temperature at 2 meters above ground
`precipitation`	Sum of rain, showers, and snow over the preceding hour
`windspeed_10m`	Wind speed at 10 meters above ground
`cloudcover`	Total cloud cover as an area fraction
`pressure_msl`	Atmospheric air pressure at mean sea level

Example *Daily* historical weather variables include:

Variable	Description
`temperature_2m_max`	Maximum daily air temperature at 2 meters above ground
`precipitation_sum`	Sum of rain, showers, and snow over the preceding day
`windspeed_10m_max`	Maximum daily wind speed at 10 meters above ground

Full documentation for the historical weather API is available at: https://open-meteo.com/en/docs/historical-weather-api

You can also call `weather_variables()` to retrieve an (incomplete) shortlist of valid hourly and daily weather variables.

Value

Specified weather forecast data for the given location and time

Examples

```
# obtain cloud cover history for London over 2020
weather_history("London",
  start = "2020-01-01",
  end = "2021-12-31",
  hourly = "cloudcover"
)
```

weather_now *Retrieve Current Weather from the Open-Meteo API*

Description

`weather_now()` calls the Open-Meteo weather API for the most recently recorded weather conditions a given location. Location is provided either as string or `c(latitude,longitude)`.

Usage

```
weather_now(location, response_units = NULL, timezone = "auto")
```

Arguments

`location`	Required. The location for which data will be retrieved. Supplied as either a `c(latitude,longitude)` WGS84 coordinate pair or a place name string (with co-ordinates obtained via `geocode()`).
`response_units`	Supply to convert temperature, windspeed, or precipitation units. This defaults to: `list(temperature_unit = "celsius", windspeed_unit = "kmh", precipitation_unit = "mm"`
`timezone`	specify timezone for time data as a string, i.e. "australia/sydney" (defaults to "auto", the timezone local to the specified `location`).

Value

Current weather conditions: temperature, windspeed, wind direction and weathercode.

Examples

```
# current weather in Montreal
weather_now("Montreal")

# current weather at the North Pole in Imperial units
weather_now(c(90, 0),
  response_units = list(
    temperature_unit = "fahrenheit",
    windspeed_unit = "mph"
  )
)
```

`weather_variables` *Retrieve valid hourly and daily weather variables*

Description

`weather_variables()` retrieves an incomplete list of *hourly* and *daily* variables accepted by `weather_forecast()` and `weather_history()`, such as temperature or precipitation.

Refer to the following documentation for the forecast and history API endpoints for detailed descriptions, units, and caveats:

Forecast API https://open-meteo.com/en/docs

Historical API https://open-meteo.com/en/docs/historical-weather-api

Usage

```
weather_variables()
```

Value

A list of valid hourly and daily weather variables

Examples

```
weather_variables()
```

Index