

Package: ropenmeteo (via r-universe)

November 2, 2024

Title Wrappers for 'Open-Meteo' API

Version 0.1

Description Wrappers for the Application Programming Interface from the <<https://open-meteo.com>> project along with helper functions. The <<https://open-meteo.com>> project streamlines access to a range of publicly historical and forecast meteorology data from agencies across the world.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports stringr, dplyr, tidyr, purrr, lubridate, glue, imputeTS, httr2, jsonlite

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0), readr, tibble, spelling

Config/testthat/edition 3

VignetteBuilder knitr

Language en-US

URL <http://flare-forecast.org/ropenmeteo/>

BugReports <https://github.com/flare-forecast/ropenmeteo/issues>

Repository <https://flare-forecast.r-universe.dev>

RemoteUrl <https://github.com/flare-forecast/ropenmeteo>

RemoteRef HEAD

RemoteSha 67a0568505c39251b02d1ffbbe48a766d140461b

Contents

add_longwave	2
convert_to_efi_standard	3

daily_to_hourly	4
get_climate_projections	4
get_ensemble_forecast	5
get_forecast	7
get_historical_weather	8
get_seasonal_forecast	9
glm_variables	10
six_hourly_to_hourly	10
write_glm_format	11

Index	12
--------------	-----------

add_longwave	<i>Add longwave to ensemble forecast dataframe using Idso and Jackson (1969). Requires cloud cover and temperature variables in input data frame.</i>
--------------	---

Description

Add longwave to ensemble forecast dataframe using Idso and Jackson (1969). Requires cloud cover and temperature variables in input data frame.

Usage

```
add_longwave(df)
```

Arguments

df	data frame output from one of the functions that gets a data frame from the API (e.g., <code>get_ensemble_forecast</code>). The data frame must have <code>cloud_cover</code> and <code>temperature_2m</code> as variables
----	---

Value

data frame with the same columns as the input df but with `longwave_radiation` added as a variable

Examples

```
file <- system.file("extdata", "test-data.csv", package="ropenmeteo")
df <- readr::read_csv(file, show_col_types = FALSE)
df |>
add_longwave()
```

`convert_to_efi_standard`

Convert units and names to CF and Ecological Forecasting Initiative standard

Description

Output units:

- air_temperature: K
- relative_humidity: proportion
- surface_downwelling_longwave_flux_in_air: W m⁻²
- surface_downwelling_shortwave_flux_in_air: W m⁻²
- precipitation_flux: kg m⁻² s⁻¹
- wind_speed: m s⁻¹
- air_pressure: Pa
- cloud_cover: proportion

Usage

```
convert_to_efi_standard(df)
```

Arguments

`df` data frame output by `get_ensemble_forecast`

Value

data frame

Examples

```
file <- system.file("extdata", "test-data.csv", package="ropenmeteo")
df <- readr::read_csv(file, show_col_types = FALSE)
df |>
  add_longwave() |>
  convert_to_efi_standard()
```

daily_to_hourly	<i>Convert daily climate projections to hourly time-step</i>
-----------------	--

Description

Convert daily climate projections to hourly time-step

Usage

```
daily_to_hourly(df, latitude, longitude)
```

Arguments

df	data frame output by <code>get_climate_projections()</code> that has daily values for the variable
latitude	latitude degree north
longitude	longitude degree east

Value

data frame with an hourly time step

Examples

```
get_climate_projections(  
  latitude = 37.30,  
  longitude = -79.83,  
  start_date = Sys.Date(),  
  end_date = Sys.Date() + lubridate::years(1),  
  model = "EC_Earth3P_HR",  
  variables = glm_variables(product = "climate_projection", time_step = "daily")) |>  
  daily_to_hourly(latitude = 37.30, longitude = -79.83)
```

get_climate_projections	<i>Download point-level climate projections using open-meteo API</i>
-------------------------	--

Description

Download point-level climate projections using open-meteo API

Usage

```
get_climate_projections(
  latitude,
  longitude,
  site_id = NULL,
  start_date,
  end_date,
  model = "EC_Earth3P_HR",
  variables = c("temperature_2m_mean")
)
```

Arguments

latitude	latitude degree north
longitude	longitude degree east
site_id	name of site location (optional, default = NULL)
start_date	Number of days in the future for forecast (starts at current day)
end_date	Number of days in the past to include in the data
model	id of forest model https://open-meteo.com/en/docs/climate-api
variables	vector of name of variable(s) https://open-meteo.com/en/docs/climate-api

Value

data frame with the results from the call to the open-meteo API. The data frame is in a long format and has the following columns: "datetime", "reference_datetime", "site_id", "model_id", "ensemble", "variable", "prediction", "unit".

Examples

```
get_climate_projections(
  latitude = 37.30,
  longitude = -79.83,
  start_date = Sys.Date(),
  end_date = Sys.Date() + lubridate::years(1),
  model = "EC_Earth3P_HR",
  variables = c("temperature_2m_mean"))
```

get_ensemble_forecast *Download point-level ensemble weather forecasting using open-meteo API*

Description

Download point-level ensemble weather forecasting using open-meteo API

Usage

```
get_ensemble_forecast(  
  latitude,  
  longitude,  
  site_id = NULL,  
  forecast_days,  
  past_days,  
  model = "gfs_seamless",  
  variables = c("relative_humidity_2m", "precipitation", "wind_speed_10m", "cloud_cover",  
    "temperature_2m", "shortwave_radiation")  
)
```

Arguments

latitude	latitude degree north
longitude	longitude degree east
site_id	name of site location (optional, default = NULL)
forecast_days	Number of days in the future for forecast (starts at current day)
past_days	Number of days in the past to include in the data
model	id of forest model https://open-meteo.com/en/docs/ensemble-api
variables	vector of name of variable(s) https://open-meteo.com/en/docs/ensemble-api

Value

data frame with the results from the call to the open-meteo API. The data frame is in a long format and has the following columns: "datetime", "reference_datetime", "site_id", "model_id", "ensemble", "variable", "prediction", "unit".

Examples

```
get_ensemble_forecast(  
  latitude = 37.30,  
  longitude = -79.83,  
  forecast_days = 7,  
  past_days = 2,  
  model = "gfs_seamless",  
  variables = c("temperature_2m"))
```

get_forecast	<i>Download point-level ensemble weather forecasting using open-meteo API</i>
--------------	---

Description

Download point-level ensemble weather forecasting using open-meteo API

Usage

```
get_forecast(  
  latitude,  
  longitude,  
  site_id = NULL,  
  forecast_days,  
  past_days,  
  model = "generic",  
  variables = c("temperature_2m")  
)
```

Arguments

latitude	latitude degree north
longitude	longitude degree east
site_id	name of site location (optional, default = NULL)
forecast_days	Number of days in the future for forecast (starts at current day)
past_days	Number of days in the past to include in the data
model	id of forest model https://open-meteo.com/en/docs/climate-api . Default = "generic"
variables	vector of name of variable(s) https://open-meteo.com/en/docs/ensemble-api .

Value

data frame (in long format)

Examples

```
get_forecast(latitude = 37.30,  
             longitude = -79.83,  
             forecast_days = 7,  
             past_days = 2,  
             model = "generic",  
             variables = c("temperature_2m"))
```

`get_historical_weather`

Download point-level historical weather (ERA5) using open-meteo API

Description

Download point-level historical weather (ERA5) using open-meteo API

Usage

```
get_historical_weather(  
  latitude,  
  longitude,  
  site_id = NULL,  
  start_date,  
  end_date,  
  variables = c("relative_humidity_2m", "precipitation", "wind_speed_10m", "cloud_cover",  
               "temperature_2m", "shortwave_radiation")  
)
```

Arguments

latitude	latitude degree north
longitude	longitude degree east
site_id	name of site location (optional, default = NULL)
start_date	earliest date requested. Must be on or after 1950-01-01
end_date	latest date requested
variables	vector of name of variable(s) https://open-meteo.com/en/docs/ensemble-api

Value

data frame with the results from the call to the open-meteo API. The data frame is in a long format and has the following columns: "datetime", "site_id", "model_id", "variable", "prediction", "unit".

Examples

```
get_historical_weather(  
  latitude = 37.30,  
  longitude = -79.83,  
  start_date = "2023-01-01",  
  end_date = Sys.Date(),  
  variables = c("temperature_2m"))
```

get_seasonal_forecast *Download point-level seasonal weather forecast using open-meteo API*

Description

Download point-level seasonal weather forecast using open-meteo API

Usage

```
get_seasonal_forecast(  
  latitude,  
  longitude,  
  site_id = NULL,  
  forecast_days,  
  past_days,  
  model = "cfs",  
  variables = c("temperature_2m")  
)
```

Arguments

latitude	latitude degree north
longitude	longitude degree east
site_id	name of site location (optional, default = NULL)
forecast_days	Number of days in the future for forecast (starts at current day)
past_days	Number of days in the past to include in the data
model	id of forest model https://open-meteo.com/en/docs/ensemble-api
variables	vector of name of variable(s) https://open-meteo.com/en/docs/ensemble-api

Value

data frame with the results from the call to the open-meteo API. The data frame is in a long format and has the following columns: "datetime", "reference_datetime", "site_id", "model_id", "ensemble", "variable", "prediction", "unit".

Examples

```
get_seasonal_forecast(  
  latitude = 37.30,  
  longitude = -79.83,  
  forecast_days = 30,  
  past_days = 5,  
  variables = glm_variables(product = "seasonal_forecast",  
                           time_step = "6hourly"))
```

`glm_variables` *Get set of variables required for the GLM model*

Description

Get set of variables required for the GLM model

Usage

```
glm_variables(product, time_step)
```

Arguments

<code>product</code>	api type: climate, forecast, ensemble_forecast, historical, seasonal_forecast
<code>time_step</code>	model and time-step: hourly, 6hour, daily

Value

a vector of variables requires by the GLM model; the vector can be used in the `variables` argument in the API function calls (e.g., `get_ensemble_forecast`).

Examples

```
glm_variables(product = "ensemble_forecast", time_step = "hourly")
```

`six_hourly_to_hourly` *Convert 6 hour seasonal forecast to hourly time-step*

Description

Convert 6 hour seasonal forecast to hourly time-step

Usage

```
six_hourly_to_hourly(df, latitude, longitude, use_solar_geom = TRUE)
```

Arguments

<code>df</code>	data frame with 6-hour time step
<code>latitude</code>	latitude degree north
<code>longitude</code>	long longitude degree east
<code>use_solar_geom</code>	use solar geometry to determine hourly solar radiation

Value

data frame with an hourly time step

Examples

```
get_seasonal_forecast(
  latitude = 37.30,
  longitude = -79.83,
  forecast_days = 30,
  past_days = 5,
  variables = glm_variables(product = "seasonal_forecast",
    time_step = "6hourly")) |>
six_hourly_to_hourly(
  latitude = 37.30,
  longitude = -79.83,
  use_solar_geom = TRUE)
```

write_glm_format	<i>Write ensemble forecast dataframe to General Lake Model formatted csv files</i>
------------------	--

Description

Write ensemble forecast dataframe to General Lake Model formatted csv files

Usage

```
write_glm_format(df, path)
```

Arguments

df	data frame output by get_ensemble_forecast()
path	directory where csv files will be written

Value

No return value, called to generate csv files in the GLM required format

Examples

```
file <- system.file("extdata", "test-data.csv", package="ropenmeteo")
df <- readr::read_csv(file, show_col_types = FALSE)
df |>
  add_longwave() |>
  write_glm_format(path = path)
```

Index

`add_longwave`, [2](#)

`convert_to_efi_standard`, [3](#)

`daily_to_hourly`, [4](#)

`get_climate_projections`, [4](#)

`get_ensemble_forecast`, [5](#)

`get_forecast`, [7](#)

`get_historical_weather`, [8](#)

`get_seasonal_forecast`, [9](#)

`glm_variables`, [10](#)

`six_hourly_to_hourly`, [10](#)

`write_glm_format`, [11](#)