Package ‘nasapower’

August 23, 2021

Type Package
Title NASA POWER API Client
Version 4.0.0
URL https://docs.ropensci.org/nasapower/
BugReports https://github.com/ropensci/nasapower/issues
Description Client for 'NASA' 'POWER' global meteorology, surface solar energy and climatology data 'API'. 'POWER' (Prediction Of Worldwide Energy Resource) data are freely available for download with a spatial resolution of 0.5 x 0.625 degree latitude and longitude for meteorology and 1 x 1 degree latitude and longitude for solar parameters with various temporal resolutions depending on the POWER parameter and community. This work is funded through the 'NASA' Earth Science Directorate Applied Science Program. For more on the data themselves, the methodologies used in creating, a web-based data viewer and web access, please see <https://power.larc.nasa.gov/>.
Depends R (>= 3.5.0)
License MIT + file LICENSE
Imports crul, lubridate, jsonlite, readr, tibble
RoxygenNote 7.1.1.9001
Encoding UTF-8
Language en-US
NeedsCompilation no
Repository CRAN
Suggests knitr, rmarkdown, testthat, spelling
VignetteBuilder knitr
X-schema.org-applicationCategory Tools
X-schema.org-keywords NASA, meteorological-data, weather, global, weather, weather-data, meteorology, NASA-POWER, agroclimatology, earth-science, data-access, climate-data
X-schema.org-isPartOf https://ropensci.org
get_power

Author  Adam H. Sparks [aut, cre] ([https://orcid.org/0000-0002-0061-8359]),
         Scott Chamberlain [rev] ([https://orcid.org/0000-0003-1444-9135]), Scott
         Chamberlain reviewed nasapower for rOpenSci, see
         https://github.com/ropensci/software-review/issues/155,
         Hazel Kavili [rev] (Hazel Kavili reviewed nasapower for rOpenSci, see
         https://github.com/ropensci/software-review/issues/155),
         Alison Boyer [rev] (Alison Boyer reviewed nasapower for rOpenSci, see
         https://github.com/ropensci/software-review/issues/155),
         Fernando Miguez [ctb] ([https://orcid.org/0000-0002-4627-8329]),
         Fernando Miguez provided assistance in identifying improper missing
         value handling in the POWER data, see
         <https://github.com/femiguez/apsimx/pull/26>)

Maintainer  Adam H. Sparks <adamhsparks@gmail.com>

Date/Publication  2021-08-23 06:50:02 UTC

R topics documented:

get_power  ................................................................. 2
query_parameters  ...................................................... 6

Index  8

get_power  Get NASA POWER data from the POWER web API

Description

Get POWER global meteorology and surface solar energy climatology data and return a tidy data
frame tibble::tibble() object. All options offered by the official POWER API are supported.
Requests are formed to submit one request per point. There is no need to make synchronous requests
for multiple parameters for a single point or regional request.

Usage

get_power(
  community,
  pars,
  temporal_api = NULL,
  lonlat,
  dates = NULL,
  site_elevation = NULL,
  wind_elevation = NULL,
  wind_surface = NULL,
  temporal_average = NULL
)
get_power

Arguments

community  A character vector providing community name: “ag”, “re” or “sb”. See argument
details for more.

pars  A character vector of solar, meteorological or climatology parameters to down-
load. When requesting a single point of x, y coordinates, a maximum of twenty
(20) pars can be specified at one time, for “daily”, “monthly” and “climatology”
temporal_api. If the temporal_api is specified as “hourly” only 15 pars can
be specified in a single query. See temporal_api for more.

temporal_api  Temporal API end-point for data being queried, supported values are “hourly”,
“daily”, “monthly” or “climatology”. See argument details for more.

lonlat  A numeric vector of geographic coordinates for a cell or region entered as x, y
coordinates. See argument details for more.

dates  A character vector of start and end dates in that order,
e.g., dates = c("1983-01-01", "2017-12-31"). Not used when
temporal_api is set to “climatology”. See argument details for more.

site_elevation  A user-supplied value for elevation at a single point in metres. If provided this
will return a corrected atmospheric pressure value adjusted to the elevation pro-
vided. Only used with lonlat as a single point of x, y coordinates, not for use
with “global” or with a regional request.

wind_elevation  A user-supplied value for elevation at a single point in metres. Wind Eleva-
tion values in Meters are required to be between 10m and 300m. Only used with
lonlat as a single point of x, y coordinates, not for use with “global” or
with a regional request. If this parameter is provided, the wind-surface param-
eter is required with the request, see https://power.larc.nasa.gov/docs/
methodology/meteorology/wind/.

wind_surface  A user-supplied wind surface for which the corrected wind-speed is to be sup-
plied. See wind-surface section for more detail.

temporal_average  Deprecated. This argument has been superseded by temporal_api to align with
the new POWER API terminology.

Value

A data frame as a POWER.Info class, an extension of the tibble::tibble, object of POWER data includ-
ing location, dates (not including “climatology”) and requested parameters. A decorative header of
metadata is included in this object.

Argument details for “community”

there are three valid values, one must be supplied. This will affect the units of the parameter and
the temporal display of time series data.

ag  Provides access to the Agroclimatology Archive, which contains industry-friendly parameters
formatted for input to crop models.

sb  Provides access to the Sustainable Buildings Archive, which contains industry-friendly param-
eters for the buildings community to include parameters in multi-year monthly averages.
re Provides access to the Renewable Energy Archive, which contains parameters specifically tailored to assist in the design of solar and wind powered renewable energy systems.

Argument details for `temporal_api`

There are four valid values.

- **hourly** The hourly average of parameters by hour, day, month and year.
- **daily** The daily average of parameters by day, month and year.
- **monthly** The monthly average of parameters by month and year.
- **climatology** Provide parameters as 22-year climatologies (solar) and 30-year climatologies (meteorology); the period climatology and monthly average, maximum, and/or minimum values.

Argument details for `lonlat`

**For a single point** To get a specific cell, 1/2 x 1/2 degree, supply a length-two numeric vector giving the decimal degree longitude and latitude in that order for data to download, e.g., `lonlat = c(-179.5,-89.5)`.

**For regional coverage** To get a region, supply a length-four numeric vector as lower left (lon, lat) and upper right (lon, lat) coordinates, e.g., `lonlat = c(min, min, max, max)` in that order for a given region, e.g., a bounding box for the south western corner of Australia: `lonlat = c(112.5,-55.5,115.5,-50.5)`. Maximum area processed is 4.5 x 4.5 degrees (100 points).

**For global coverage** To get global coverage for “climatology”, supply “global” while also specifying “climatology” for the `temporal_api`.

Argument details for `dates`

If one date only is provided, it will be treated as both the start date and the end date and only a single day’s values will be returned, e.g., `dates = "1983-01-01"`. When `temporal_api` is set to “monthly”, use only two year values (YYYY), e.g. `dates = c(1983,2010)`. This argument should not be used when `temporal_api` is set to “climatology” and will be ignored if set.

**wind-surface**

There are 17 surfaces that may be used for corrected wind-speed values using the following equation:

\[ WSC_{ht} = WS_{10m} \times \left(\frac{h_{gt}}{WS_{10m}}\right)^{\alpha} \]

Valid surface types are described here.

- **vegtype_1** 35-m broadleaf-evergreen trees (70% coverage)
- **vegtype_2** 20-m broadleaf-deciduous trees (75% coverage)
- **vegtype_3** 20-m broadleaf and needleleaf trees (75% coverage)
- **vegtype_4** 17-m needleleaf-evergreen trees (75% coverage)
- **vegtype_5** 14-m needleleaf-deciduous trees (50% coverage)
- **vegtype_6** Savanna: 18-m broadleaf trees (30%) & groundcover
get_power

vegtype_7  0.6-m perennial groundcover (100%)
vegtype_8  0.5-m broadleaf shrubs (variable %) & groundcover
vegtype_9  0.5-m broadleaf shrubs (10%) with bare soil
vegtype_10 Tundra: 0.6-m trees/shrubs (variable %) & groundcover
vegtype_11 Rough bare soil
vegtype_12 Crop: 20-m broadleaf-deciduous trees (10%) & wheat
vegtype_20 Rough glacial snow/ice
seaice  Smooth sea ice
openwater  Open water
airportice  Airport: flat ice/snow
airportgrass  Airport: flat rough grass

Note
The associated metadata shown in the decorative header are not saved if the data are exported to a file format other than a native R data format, e.g., .Rdata, .rda or .rds.

Author(s)
Adam H. Sparks <adamhsparks@gmail.com>

References
https://power.larc.nasa.gov/docs/methodology/ https://power.larc.nasa.gov

Examples

# Fetch daily "ag" community temperature, relative humidity and precipitation
# for January 1 1985 at Kingsthorpe, Queensland, Australia
ag_d <- get_power(  
  community = "ag",  
  lonlat = c(151.81, -27.48),  
  pars = c("RH2M", "T2M", "PRECTOTCORR"),  
  dates = "1985-01-01",  
  temporal_api = "daily"
)

ag_d

# Fetch single point climatology for air temperature
ag_c_point <- get_power(  
  community = "ag",  
  pars = "T2M",  
  c(151.81, -27.48),  
  temporal_api = "climatology"
)
ag_c_point

# Fetch global ag climatology for air temperature
ag_c_global <- get_power(
  community = "ag",
  pars = "T2M",
  lonlat = "global",
  temporal_api = "climatology"
)

ag_c_global

# Fetch interannual solar cooking parameters for a given region
sse_i <- get_power(
  community = "re",
  lonlat = c(112.5, -55.5, 115.5, -50.5),
  dates = c("1984", "1985"),
  temporal_api = "monthly",
  pars = c("CLRSKY_SFC_SW_DWN", "ALLSKY_SFC_SW_DWN")
)

sse_i

---

query_parameters  
Query the POWER API for detailed information on parameters

Description

Queries the POWER API returning detailed information on available parameters.

Usage

query_parameters(community = NULL, par = NULL, temporal_api = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>community</td>
<td>An optional character vector providing community name: “ag”, “sb” or “re”.</td>
</tr>
<tr>
<td>par</td>
<td>An optional character vector of a single solar, meteorological or climatology parameter to query. If unsure, omit this argument for a full list of all the parameters available for each temporal API and community.</td>
</tr>
<tr>
<td>temporal_api</td>
<td>An optional character vector indicating the temporal API end-point for data being queried, supported values are “hourly”, “daily”, “monthly” or “climatology”.</td>
</tr>
</tbody>
</table>
query_parameters

Details

If par is not provided all possible parameters for the provided community, community and temporal API, temporal_api will be returned. If only a single parameter is supplied with no community or temporal_api then the complete attribute information for that parameter will be returned for all possible communities and temporal APIs combinations. If all three values are provided, only the information for that specific combination of parameter, temporal API and community will be returned.

Value

A list object of information for the requested parameter(s) (if requested), community and temporal API.

Argument details for temporal_api

There are four valid values.

- **hourly** The hourly average of pars by hour, day, month and year.
- **daily** The daily average of pars by day, month and year.
- **monthly** The monthly average of pars by month and year.
- **climatology** Provide parameters as 22-year climatologies (solar) and 30-year climatologies (meteorology); the period climatology and monthly average, maximum, and/or minimum values.

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```r
# fetch the complete set of attribute information for "T2M".
query_parameters(par = "T2M")

# fetch complete temporal and community specific attribute information
# for "T2M" in the "ag" community for the "hourly" temporal API.
query_parameters(par = "T2M",
                 community = "ag",
                 temporal_api = "hourly")

# fetch complete temporal and community specific attribute information
# for all parameters in the "ag" community for the "hourly" temporal API.
query_parameters(community = "ag",
                 temporal_api = "hourly")
```
Index

get_power, 2
list, 7
query_parameters, 6
tibble::tibble, 3
tibble::tibble(), 2