

# Package: nasapower (via r-universe)

December 18, 2024

**Type** Package

**Title** NASA POWER API Client

**Version** 4.2.2

**Description** An API client for NASA POWER global meteorology, surface solar energy and climatology data API. POWER (Prediction Of Worldwide Energy Resources) data are freely available for download with varying spatial resolutions dependent on the original data and with several 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/>>.

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/nasapower/>

**BugReports** <https://github.com/ropensci/nasapower/issues>

**Depends** R (>= 3.5.0)

**Imports** cli, crul, jsonlite, lubridate, readr, rlang, tibble (>= 3.0.2)

**Suggests** knitr, purrr, rmarkdown, spelling, testthat (>= 3.0.0), vcr (>= 0.6.0)

**VignetteBuilder** knitr

**Config/Needs/documentation** roxylint

**Config/roxylint** list(linters = roxylint::tidy)

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Encoding** UTF-8

**Language** en-US

**NeedsCompilation** no

**Roxygen** list(markdown = TRUE, roclets = c("`collate", "`namespace",  
 "`rd", "`roxygen2::roxygen2"))

**RoxygenNote** 7.3.2

**X-schema.org-applicationCategory** Tools

**X-schema.org-isPartOf** <https://ropensci.org>

**X-schema.org-keywords** NASA, meteorological-data, weather, global,  
 weather, weather-data, meteorology, NASA-POWER,  
 agroclimatology, earth-science, data-access, climate-data

**Config/pak/sysreqs** libssl-dev libx11-dev

**Repository** <https://ropensci.r-universe.dev>

**RemoteUrl** <https://github.com/ropensci/nasapower>

**RemoteRef** main

**RemoteSha** 892ab920b6745fa3548945b4269466fd719504c1

## Contents

get_power . . . . .	2
query_groupings . . . . .	6
query_parameters . . . . .	7
query_surfaces . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

get\_power

*Get NASA POWER data from the POWER 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. See section on “Rate Limiting” for more.

## Usage

```
get_power(
  community = c("ag", "re", "sb"),
  pars,
  temporal_api = c("daily", "monthly", "hourly", "climatology"),
  lonlat,
  dates = NULL,
  site_elevation = NULL,
  wind_elevation = NULL,
```

```

    wind_surface = NULL,
    time_standard = c("LST", "UTC")
  )

```

### Arguments

community	A case-insensitive character vector providing community name: “AG”, “RE” or “SB”. See argument details for more.
pars	case-insensitive character vector of solar, meteorological or climatology parameters to download. 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. These values are checked internally for validity before sending the query to the POWER API.
temporal_api	A case-insensitive character vector providing the temporal API end-point for data being queried, supported values are “hourly”, “daily”, “monthly” or “climatology”. Defaults to “daily”. See argument details for more.
lonlat	A numeric vector of geographic coordinates for a cell or region entered as x, y (longitude, latitude) 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 provided. 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 Elevation values are required to be between 10 and 300 metres. 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 parameter is required with the request, see <a href="https://power.larc.nasa.gov/docs/methodology/meteorology/wind/">https://power.larc.nasa.gov/docs/methodology/meteorology/wind/</a> .
wind_surface	A user-supplied wind surface for which the corrected wind-speed is to be supplied. See wind-surface section for more detail.
time_standard	POWER provides two different time standards. <ul style="list-style-type: none"> <li>• Universal Time Coordinated (UTC): is the standard time measure that used by the world.</li> <li>• Local Solar Time (LST): A 15 degree swath that represents solar noon at the middle longitude of the swath. Defaults to LST.</li> </ul>

### Value

A data frame as a POWER.Info class, an extension of the `tibble::tibble`, object of POWER data including 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 parameters 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 pars by hour, day, month and year, the time zone is LST by default.

**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.

**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(xmin, ymin, xmax, ymax) 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:

$$WSC_{hgt} = WS_{10m} \times \left( \frac{hgt}{WS_{50m}} \right)^\alpha$$

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  
**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

### Rate limiting

The POWER API endpoints limit queries to prevent server overloads due to repetitive and rapid requests. If you find that the API is throttling your queries, I suggest that you investigate the use of `limit_rate()` from **ratelimitr** to create self-limiting functions that will respect the rate limits that the API has in place. It is considered best practice to check the **POWER website** for the latest rate limits as they differ between temporal APIs and may change over time as the project matures.

### 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 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\_groupings

*Query the POWER API for detailed information on available parameter groupings*

---

## Description

Queries the POWER API returning detailed information on available parameter groupings grouped by community followed by temporal API or if `global = TRUE`, grouped by climatology, then by the available types of parameters.

## Usage

```
query_groupings(global = FALSE)
```

**Arguments**

`global` Boolean; should the query return global parameter groupings and attribute information? Defaults to FALSE returning details for point data.

**Value**

A [list](#) object of information on parameter groupings in the POWER API.

**Author(s)**

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

**Examples**

```
# fetch groupings for parameters
query_groupings()

# fetch groupings for global parameters
query_groupings(global = TRUE)
```

---

query_parameters	<i>Query the POWER API for detailed information on available parameters</i>
------------------	---

---

**Description**

Queries the POWER API returning detailed information on available parameters. For a list of all available parameters, use `parameters`

**Usage**

```
query_parameters(
  community = NULL,
  pars = NULL,
  temporal_api = NULL,
  metadata = FALSE
)
```

**Arguments**

`community` An optional character vector providing community name: “ag”, “sb” or “re”.

`pars` An optional character string of a single solar, meteorological or climatology parameter to query. If none is provided, all are returned.

`temporal_api` An optional character vector indicating the temporal API end-point for data being queried, supported values are “hourly”, “daily”, “monthly” or “climatology”.

**metadata** Boolean; retrieve extra parameter metadata? This is only applicable if you supply the `community` and `temporal_api`, if these values are not provided it will be ignored. Defaults to `FALSE`.

### Value

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

### 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

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

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



---

query_surfaces	<i>Query the POWER API for Detailed Information on Wind Type Surfaces</i>
----------------	---

---

**Description**

Queries the POWER API returning detailed information on all (or just one) wind elevation surface alias and attribute information.

**Usage**

```
query_surfaces(surface_alias = NULL)
```

**Arguments**

`surface_alias` An optional character vector providing a wind surface alias available from the POWER API. All values are returned if this value is not provided.

**Value**

A [list](#) object of information for the requested wind surface(s).

**Author(s)**

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

**Examples**

```
# fetch all wind surface information
query_surfaces()

# fetch surface information for `airportgrass`
query_surfaces(surface_alias = "airportgrass")
```

# Index

`get_power`, 2

`list`, 7–9

`query_groupings`, 6

`query_parameters`, 7

`query_surfaces`, 9

`tibble::tibble`, 3

`tibble::tibble()`, 2