Package 'weatherjoin'

January 29, 2026

Type Package

Title Join Gridded Weather Data to Event Tables

Version 0.2.0

URL https://github.com/hauae/weatherjoin

BugReports https://github.com/hauae/weatherjoin/issues

Description High-level tools to attach gridded weather data from the NASA POWER Project to event-based datasets. The package plans efficient spatio-temporal API calls via the 'nasapower' R package, caches downloaded segments locally, and joins weather variables back to the input table using exact or rolling joins. This package is not affiliated with or endorsed by NASA.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports data.table, jsonlite

Suggests nasapower, digest, fst, anytime, testthat (>= 3.0.0), knitr, rmarkdown, withr

Depends R (>= 4.1.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Przemek Dolowy [aut, cre] (affiliation: Harper Adams University)

Maintainer Przemek Dolowy <pdolowy@harper-adams.ac.uk>

Repository CRAN

Date/Publication 2026-01-29 18:50:01 UTC

Contents

`join_weather` *Join gridded weather data to an event table*

Description

Attach gridded weather variables from NASA POWER to rows of an event table. The function:

- standardizes/validates time input (single timestamp column or multiple time columns),
- plans efficient provider calls by clustering locations (default) and splitting sparse time ranges,
- caches downloaded weather segments locally and reuses them,
- joins weather back to events using exact or rolling joins.

Usage

```
join_weather(
  x,
  params,
  time,
  lat_col = "lat",
  lon_col = "lon",
  time_api = c("guess", "hourly", "daily"),
  tz = "UTC",
  roll = c("nearest", "last", "none"),
  roll_max_hours = NULL,
  spatial_mode = c("cluster", "exact", "by_group"),
  group_col = NULL,
  cluster_radius_m = 250,
  site_elevation = c("constant", "auto"),
  elev_constant = 100,
  elev_fun = NULL,
  community = "ag",
  cache_scope = c("user", "project"),
  cache_dir = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	A data.frame/data.table with event rows.
`params`	Character vector of NASA POWER parameter codes (e.g. `"T2M"`).
`time`	A single column name containing time (POSIXct/Date/character/numeric) OR a character vector of column names used to assemble a timestamp (e.g. `c("YEAR","MO","DY","HR")`).
`lat_col, lon_col`	Column names for latitude and longitude (decimal degrees).

time_api	One of "guess", "hourly", "daily". If "daily" is chosen while the input contains time-of-day information, timestamps are downsampled to dates (with a fixed hour). If "hourly" is chosen but the input has no time-of-day information, an error is raised.
tz	Time zone used to interpret/construct input timestamps (default "UTC"). Weather is requested from NASA POWER in UTC.
roll	Join behaviour when matching timestamps: "nearest" (default, recommended), "last", or "none" (exact). Rolling is applied when joining hourly weather to event times.
roll_max_hours	Maximum allowed time distance (hours) for a rolling match. If NULL, a safe default is used: 1 hour for hourly joins and 24 hours for daily joins.
spatial_mode	How to reduce many points to representative locations before calling POWER: "cluster" (default), "exact", or "by_group". Clustering reduces accidental explosion of provider calls and matches POWER's coarse spatial resolution.
group_col	Grouping column used when spatial_mode="by_group".
cluster_radius_m	Clustering radius in meters when spatial_mode="cluster".
site_elevation	Elevation strategy for POWER calls: "constant" or "auto". Elevation is resolved for representative locations and becomes part of the cache identity.
elev_constant	Constant elevation (meters) used when site_elevation="constant" and as a fallback for "auto".
elev_fun	Optional function function(lon, lat, ...) returning elevation (meters) for representative points.
community	Passed to nasapower::get_power() (e.g. "ag").
cache_scope	Where to store cache by default: "user" or "project".
cache_dir	Optional explicit cache directory. If NULL, determined by cache_scope.
verbose	If TRUE, print progress messages.
...	Passed through to nasapower::get_power().

Value

A data.table with weather columns appended. Rows with missing/invalid inputs keep their original values and receive NA weather.

See Also

wj_cache_list, wj_cache_clear, weatherjoin_options

`weatherjoin_options` *weatherjoin options*

Description

Most users will not need to change package options. Advanced configuration can be controlled via `options()`.

Details

Cache policy:

- `weatherjoin.cache_max_age_days` Cache entries older than this (days) are considered stale (default `60`).
- `weatherjoin.cache_refresh` When to refetch: one of `"if_missing"`, `"if_stale"`, `"always"` (default `"if_missing"`).
- `weatherjoin.cache_match_mode` Cache matching mode: `"cover"` (cached window covers requested) or `"exact"` (default `"cover"`).
- `weatherjoin.cache_param_match` Parameter matching for cache reuse: `"superset"` or `"exact"` (default `"superset"`).
- `weatherjoin.cache_pkg` Internal namespace used when `cache_scope="user"` (default `"weatherjoin"`).

Time splitting and call planning:

These options control how sparse time series are split into separate provider calls. They are primarily performance controls; incorrect values will not change the meaning of returned weather values, only how much data is downloaded and cached.

- `weatherjoin.split_penalty_hours` Gap threshold (hours). Larger values yield fewer, wider time windows (default `72`).
- `weatherjoin.pad_hours` Padding (hours) added to both ends of each planned time window (default `0`).
- `weatherjoin.max_parts` Maximum number of planned time windows per representative location (default `50`).

Time construction:

- `weatherjoin.dummy_hour` Hour used when constructing daily timestamps (default `12`).

Diagnostics:

- `weatherjoin.keep_rep_cols` If `TRUE`, keep representative-location diagnostics (rep_lon/rep_lat, distance, elevation) in outputs (default `FALSE`).

Use **withr** for temporary changes:

```
withr::local_options(list(
  weatherjoin.split_penalty_hours = 168,
  weatherjoin.max_parts = 25
))
```

wj_cache_clear *Clear cached weather data*

Description

Deletes cached files and (optionally) removes rows from the cache index.

Usage

```
wj_cache_clear(
  cache_dir = NULL,
  cache_scope = c("user", "project"),
  pkg = "weatherjoin",
  filter = NULL,
  keep_index = FALSE,
  dry_run = FALSE,
  verbose = TRUE
)
```

Arguments

cache_dir	Optional explicit cache directory.
cache_scope	Where to store cache by default: "user" or "project".
pkg	Package name used for "user" cache scope.
filter	Optional expression evaluated within the cache index to select entries to remove.
keep_index	If TRUE, leaves index rows (useful for debugging); default FALSE.
dry_run	If TRUE, prints what would be deleted but does not delete.
verbose	If TRUE, prints progress.

Value

Invisibly returns the rows selected for deletion.

wj_cache_list *List cached weather segments*

Description

Returns the cache index (one row per cached segment).

Usage

```
wj_cache_list(
  cache_dir = NULL,
  cache_scope = c("user", "project"),
  pkg = "weatherjoin"
)
```

Arguments

cache_dir	Optional explicit cache directory.
cache_scope	Where to store cache by default: "user" or "project".
pkg	Package name used for "user" cache scope.

Value

A data.table index of cached segments.

---

wj_cache_upgrade_index

Upgrade cache index schema

Description

Ensures the cache index contains required columns and correct types.

Usage

```
wj_cache_upgrade_index(
  cache_dir = NULL,
  cache_scope = c("user", "project"),
  pkg = "weatherjoin",
  verbose = TRUE
)
```

Arguments

cache_dir	Optional explicit cache directory.
cache_scope	Where to store cache by default: "user" or "project".
pkg	Package name used for "user" cache scope.
verbose	If TRUE, prints progress.

Value

The upgraded cache index.

Index