Package 'amadeus'

May 21, 2026

Title Accessing and Analyzing Large-Scale Environmental Data

Version 2.0.0

Maintainer Kyle Messier <kyle.messier@nih.gov>

Description Functions are designed to facilitate access to and utility with large scale, publicly available environmental data in R. The package contains functions for downloading raw data files from web URLs (download_data()), processing the raw data files into clean spatial objects (process_covariates()), and extracting values from the spatial data objects at point and polygon locations (calculate_covariates()). These functions call a series of source-specific functions which are tailored to each data sources/datasets particular URL structure, data format, and spatial/temporal resolution. The functions are tested, versioned, and open source and open access. For sum_edc() method details, see Messier, Akita, and Serre (2012) <doi:10.1021/es203152a>.

Depends R (>= 4.2.0)

Imports dplyr, sf, sftime, stats, terra (>= 1.8-50), methods, data.table, httr2, exactextractr, utils, stringi, stars, tidyr, rlang, archive, collapse, Rdpack

Suggests covr, devtools, doRNG, FNN, furrr, ggplot2, knitr, lwgeom, maps, mirai, nhdplusTools, rmarkdown, spelling, stringr, targets, testthat (>= 3.0.0), tigris, withr

RdMacros Rdpack

Encoding UTF-8

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.3

Config/Needs/website tidyverse/tidytemplate

Config/testhat/edition 3

License MIT + file LICENSE

URL https://niehs.github.io/amadeus/

BugReports https://github.com/NIEHS/amadeus/issues

Language en-US

NeedsCompilation no

Author Mitchell Manware [aut, ctb] (ORCID:
<https://orcid.org/0009-0003-6440-6106>),
Insang Song [aut, ctb] (ORCID: <https://orcid.org/0000-0001-8732-3256>),
Eva Marques [aut, ctb] (ORCID: <https://orcid.org/0000-0001-9817-6546>),
Mariana Alifa Kassien [aut, ctb] (ORCID:
<https://orcid.org/0000-0003-2295-406X>),
Elizabeth Scholl [ctb] (ORCID: <https://orcid.org/0000-0003-2727-1954>),
Kyle Messier [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9508-9623>),
Spatiotemporal Exposures and Toxicology Group [cph]

Repository CRAN

Date/Publication 2026-05-21 21:00:08 UTC

Contents

`as_mysftime` *Create an* `sftime` *object*

Description

Create a `sftime` object from one of `data.frame`, `data.table`, `sf`, `sftime`, `SpatRaster`, `SpatRasterDataset`, `SpatVector`

Usage

```
as_mysftime(x, ...)
```

Arguments

`x`	an object of class `data.frame`, `data.table`, `sf`, `sftime`, `SpatRaster`, `SpatRasterDataset` or `SpatVector`
`...`	if x is a data.frame or data.table: lonname, latname, timename and crs arguments are required. If x is a sf or sftime, timename argument is required. If x is a terra::SpatRaster, varname argument is required.

Value

an `sftime` object with constrained time column name

Author(s)

Eva Marques

See Also

check_mysftime, sf_as_mysftime, data.frame, data.table::data.table, terra::rast, terra::sds, terra::vect

`calculate_covariates` *Calculate covariates wrapper function*

Description

The `calculate_covariates()` function extracts values at point locations from a SpatRaster or SpatVector object returned from `process_covariates()`. `calculate_covariates()` and the underlying source-specific covariate functions have been designed to operate on the processed objects. To avoid errors, **do not edit the processed SpatRaster or SpatVector objects before passing to** `calculate_covariates()`.

Usage

```
calculate_covariates(
 covariate = c("modis", "koppen-geiger", "koeppen-geiger", "koppen", "koeppen", "geos",
  "dummies", "gmted", "sedac_groads", "groads", "roads", "ecoregions", "ecoregion",
  "hms", "smoke", "gmted", "narr", "geos", "sedac_population", "population", "nlcd",
  "merra", "merra2", "gridmet", "terraclimate", "tri", "nei", "mcd14ml", "prism",
  "cropscape", "cdl", "huc", "edgar", "goes", "goes_adp", "GOES", "drought", "spei",
   "eddi", "usdm"),
 from,
 locs,
 locs_id = "site_id",
 .by_time = NULL,
 weights = NULL,
 ...
)
```

Arguments

`covariate`	character(1). Covariate type.
`from`	character, SpatRaster, SpatVector, or data.frame depending on the selected `covariate` route.
`locs`	sf/SpatVector. Unique locations. Should include a unique identifier field named `locs_id`
`locs_id`	character(1). Name of unique identifier. Default is `"site_id"`.
`.by_time`	NULL or character(1). Name of the time column to use temporal summarization unit token. `NULL` (default) disables temporal summarization.
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Passed through to the underlying source-specific function for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`...`	Arguments passed to each covariate calculation function.

Value

Calculated covariates as a data.frame or SpatVector object

Note

`covariate` argument value is converted to lowercase.

Author(s)

Insang Song

See Also

- `calculate_modis`: "modis", "MODIS"
- `calculate_koppen_geiger`: "koppen-geiger", "koeppen-geiger", "koppen"
- `calculate_ecoregion`: "ecoregion", "ecoregions"
- `calculate_temporal_dummies`: "dummies", "Dummies"
- `calculate_hms`: "hms", "smoke", "HMS"
- `calculate_gmted`: "gmted", "GMTED"
- `calculate_narr`: "narr", "NARR"
- `calculate_geos`: "geos", "geos_cf", "GEOS"
- `calculate_goes`: "goes", "goes_adp", "GOES"
- `calculate_population`: "population", "sedac_population"
- `calculate_groads`: "roads", "groads", "sedac_groads"
- `calculate_nlcd`: "nlcd", "NLCD"
- `calculate_tri`: "tri", "TRI"
- `calculate_nei`: "nei", "NEI"
- `calculate_merra2`: "merra", "MERRA", "merra2", "MERRA2"
- `calculate_gridmet`: "gridMET", "gridmet"
- `calculate_terraclimate`: "terraclimate", "TerraClimate"
- `calculate_prism`: "prism", "PRISM"
- `calculate_cropscape`: "cropscape", "cdl"
- `calculate_huc`: "huc", "HUC"
- `calculate_edgar`: "edgar"
- `calculate_drought`: "drought", "spei", "eddi", "usdm"

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_covariates(
  covariate = "narr",
  from = narr, # derived from process_covariates() example
  locs = loc,
  locs_id = "id",
```

```
  geom = FALSE
)

## End(Not run)
```

`calculate_cropscape` *Calculate Cropscape covariates*

Description

Extract Cropscape (CDL) values at point locations. Returns a `data.frame` object containing `locs_id` and crop specific cell fractions.

Usage

```
calculate_cropscape(
  from,
  locs,
  locs_id = "site_id",
  radius = 0,
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output from `process_cropscape()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`radius`	integer(1). Circular buffer distance around site locations. (Default = 0).
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`geom`	FALSE/"sf"/"terra".. Should the function return with geometry? Default is `FALSE`, options with geometry are "sf" or "terra". The coordinate reference system of the `sf` or `SpatVector` is that of `from`.
`...`	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Insang Song

See Also

process_cropscape()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_cropscape(
  from = cropscape, # derived from process_cropscape() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  geom = FALSE
)

## End(Not run)
```

---

calculate_drought *Calculate drought index covariates*

Description

The calculate_drought() function extracts drought index values at point locations from an object returned by process_drought(). Three source datasets are supported:

- **SPEI / EDDI** (SpatRaster): cell values are extracted at each location using the standard raster-extraction pipeline (calc_prepare_locs() -> calc_worker() -> calc_return_locs()). Time column format is "YYYY-MM-DD".
- **USDM** (SpatVector polygons): the drought monitor class (DM, integer 0-4) at each location is determined via spatial overlay. A time column of class Date is populated from the date attribute of from.

When .by_time is supplied the extracted result is passed through calc_summarize_by() using the same semantics as all other calculate_*() functions in this package.

Usage

```
calculate_drought(
  from,
  locs,
  locs_id = "site_id",
  radius = 0L,
  fun = "mean",
  weights = NULL,
  geom = FALSE,
  .by_time = NULL,
  ...
)
```

Arguments

from	SpatRaster or SpatVector. Output of process_drought(). • SpatRaster for SPEI or EDDI sources. • SpatVector (polygons) for USDM source.
locs	data.frame, character (path to CSV), SpatVector, or sf object. Point locations at which to extract values.
locs_id	character(1). Name of the unique location identifier column in locs. Default "site_id".
radius	integer(1). Circular buffer radius in metres around each site location used for extraction. For SPEI/EDDI this controls raster buffering; for USDM, radius > 0 additionally returns class proportions within the buffer. Default 0L.
fun	character(1). Summary function applied to raster cells within the buffer (SPEI/EDDI only). Default "mean".
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE, "sf", or "terra". Whether to attach geometry to the returned object. Default FALSE.
.by_time	NULL or character(1). Name of the time column to use temporal summarization unit token. NULL disables "time".
...	Reserved for future use; currently ignored.

Value

A data.frame (default) or SpatVector/sf object (when geom is set) with columns:

<locs_id> Location identifier.

time Date of the observation (Date or "YYYY-MM-DD" character).

<value_column> Extracted drought index or class value.

When .by_time is non-NULL, rows are aggregated to the specified resolution via calc_summarize_by().

Note

- The column name for extracted drought values follows the pattern "<source>_<timescale>_<radius>" (e.g. "spei_01_0") for SPEI/EDDI, and "usdm_dm_0" for USDM.
- For USDM with radius > 0, proportion columns are added as "usdm_dm_<class>_<radius>" for classes 0–4.

Author(s)

Insang Song

See Also

process_drought, download_drought, calc_summarize_by

Examples

```
## Not run:
locs <- data.frame(site_id = "001", lon = -97.5, lat = 35.5)
## SPEI example
spei <- process_drought(
  source = "spei",
  path = "./data/drought",
  date = c("2020-01-01", "2020-12-31"),
  timescale = 1L
)
calculate_drought(
  from = spei,
  locs = locs,
  locs_id = "site_id",
  radius = 0L,
  fun = "mean"
)
## USDM example
usdm <- process_drought(
  source = "usdm",
  path = "./data/drought",
  date = c("2020-01-07", "2020-03-31")
)
calculate_drought(
  from = usdm,
  locs = locs,
  locs_id = "site_id"
)

## End(Not run)
```

---

calculate_ecoregion *Calculate ecoregions covariates*

Description

Extract ecoregions covariates (U.S. EPA Ecoregions Level 2/3) at point or polygon locations. Returns a `data.frame` object containing `locs_id` and either dummy indicators (`frac = FALSE`) or area fractions (`frac = TRUE`) for each ecoregion.

Usage

```
calculate_ecoregion(
  from = NULL,
  locs,
  locs_id = "site_id",
  colnames = c("coded", "full_ecoregion"),
  frac = FALSE,
  drop = FALSE,
```

```
  weights = NULL,
  geom = FALSE,
  radius = 0,
  ...
)
```

Arguments

from	SpatVector(1). Output of process_ecoregion.
locs	sf/SpatVector. Unique locs. Should include a unique identifier field named locs_id
locs_id	character(1). Name of unique identifier.
colnames	character(1). Naming convention for ecoregion indicator columns. Default is "coded" for the existing numeric key-based names. Use "full_ecoregion" to emit sanitized full ecoregion names.
frac	logical(1). Default FALSE. If FALSE, returns binary dummy indicators (0/1). If TRUE, returns fractional overlap values.
drop	logical(1). Default FALSE. If TRUE, remove ecoregion columns that are all 0 or NA across returned locations.
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
radius	numeric(1). Circular buffer size (meters) around point locations. Use 0 (default) for exact point extraction.
...	Placeholders.

Value

a data.frame or SpatVector object with ecoregion indicator/fraction variables and attributes of:

- Indicator names are controlled by colnames: "coded" (default) creates key-based names such as DUM_E2083_00000 and DUM_E3064_00000 when frac = FALSE, or FRC_E2083_00000 and FRC_E3064_00000 when frac = TRUE; "full_ecoregion" creates sanitized name-based columns such as DUM_E2_SOUTHEASTERN_USA_PLAINS_00000 / FRC_E2_SOUTHEASTERN_USA_PLAINS_00000 and DUM_E3_NORTHERN_PIEDMONT_00000 / FRC_E3_NORTHERN_PIEDMONT_00000 (duplicates are suffixed, e.g. _1).
- attr(., "ecoregion2_code"): Ecoregion lv.2 code and key
- attr(., "ecoregion3_code"): Ecoregion lv.3 code and key

Author(s)

Insang Song

See Also

process_ecoregion

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_ecoregion(
  from = ecoregion, # derived from process_ecoregion() example
  locs = loc,
  locs_id = "id",
  colnames = "coded",
  geom = FALSE
)

## End(Not run)
```

---

calculate_edgar *Calculate EDGAR covariates*

Description

Extract EDGAR gridded emissions values at point locations. For `radius = 0`, cell values are extracted directly. For `radius > 0`, means are calculated over a circular buffer around each location.

Usage

```
calculate_edgar(
  from,
  locs,
  locs_id = "site_id",
  radius = 0,
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatRaster(1). Output from `process_edgar()`.
locs	data.frame, character to file path, SpatVector, or sf object.
locs_id	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
radius	numeric(1). Circular buffer distance around site locations. Default is `0`.

weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Mariana Alifa Kassien, Insang Song

See Also

process_edgar()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires data that is
##       not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_edgar(
  from = edgar, # derived from process_edgar() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  geom = FALSE
)

## End(Not run)
```

---

calculate_geos *Calculate atmospheric composition covariates*

Description

Extract atmospheric composition values at point locations. Returns a data.frame object containing locs_id, date and hour, vertical pressure level, and atmospheric composition variable. Atmospheric composition variable column name reflects variable and circular buffer radius.

Usage

```
calculate_geos(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatRaster(1). Output of process_geos().
locs	data.frame, characater to file path, SpatVector, or sf object.
locs_id	character(1). Column within locations CSV file containing identifier for each unique coordinate location.
radius	integer(1). Circular buffer distance around site locations. (Default = 0).
fun	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = mean).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used when .by_time is provided.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object. When .by_time is provided, rows are aggregated using calc_summarize_by().

Author(s)

Mitchell Manware

See Also

process_geos()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_geos(
  from = geos, # derived from process_geos() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

`calculate_gmted` *Calculate elevation covariates*

Description

Extract elevation values at point locations. Returns a `data.frame` object containing `locs_id`, year of release, and elevation variable. Elevation variable column name follows the pattern `gmted_<radius>` (for example, `gmted_0` or `gmted_100`).

Usage

```
calculate_gmted(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output from `process_gmted()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`radius`	integer(1). Circular buffer distance around site locations. (Default = 0).
`fun`	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = `mean`).

weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders

Value

a data.frame or SpatVector object

Author(s)

Mitchell Manware

See Also

process_gmted()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_gmted(
  from = gmted, # derived from process_gmted() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

---

calculate_goes *Calculate NOAA GOES ADP covariates*

Description

Extract NOAA GOES Aerosol Detection Product (ADP) values at point locations from a SpatRaster returned by process_goes(). Returns a data.frame (or SpatVector / sf) containing locs_id, time, and the extracted variable column ({variable}_{radius}).

Usage

```
calculate_goes(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatRaster(1). Output from process_goes().
locs	data.frame, character file path, SpatVector, or sf object with point locations.
locs_id	character(1). Column name for unique location identifier.
radius	integer(1). Circular buffer radius in metres around each site (default 0 = point extraction).
fun	character(1). Summary function for buffered extractions (default "mean").
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.
geom	FALSE/"sf"/"terra". Return geometry with results. Default FALSE. The CRS is inherited from from.
...	Placeholders.

Value

a data.frame or SpatVector object.

Author(s)

Mitchell Manware

See Also

process_goes

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires downloaded
##       and processed data.
## Not run:
loc <- data.frame(id = "001", lon = -95.0, lat = 34.5)
calculate_goes(
  from = goes,  # derived from process_goes() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean"
)

## End(Not run)
```

`calculate_gridmet` *Calculate gridMET covariates*

Description

Extract gridMET values at point locations. Returns a `data.frame` object containing `locs_id` and gridMET variable. gridMET variable column name reflects the gridMET variable and circular buffer radius.

Usage

```
calculate_gridmet(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output from `process_gridmet()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`radius`	integer(1). Circular buffer distance around site locations. (Default = 0).
`fun`	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = `mean`).

weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Mitchell Manware

See Also

process_gridmet()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_gridmet(
  from = gridmet, # derived from process_gridmet() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

---

calculate_groads *Calculate roads covariates*

Description

Prepared groads data is clipped with the buffer polygons of radius. The total length of the roads are calculated. Then the density of the roads is calculated by dividing the total length from the area of the buffer. terra::linearUnits() is used to convert the unit of length to meters.

Usage

```
calculate_groads(
  from = NULL,
  locs = NULL,
  locs_id = NULL,
  radius = 1000,
  fun = "sum",
  drop = FALSE,
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatVector(1). Output of process_groads.
locs	data.frame, characater to file path, SpatVector, or sf object.
locs_id	character(1). Column within locations CSV file containing identifier for each unique coordinate location.
radius	integer(1). Circular buffer distance around site locations. (Default = 1000).
fun	function(1). Function used to summarize the length of roads within sites location buffer (Default is sum).
drop	logical(1). Should locations with zero roads in the extraction buffer be dropped from results? Default is FALSE (retain all locations).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Note

Unit is km / sq km. The returned data.frame object contains a $time column to represent the temporal range covered by the dataset. For more information, see https://data.nasa.gov/dataset/global-roads-open-access-data-set-version-1-groadsv1.

Author(s)

Insang Song

See Also

process_groads

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_groads(
  from = groads, # derived from process_groads() example
  locs = loc,
  locs_id = "id",
  radius = 1000,
  fun = "sum",
  geom = FALSE
)

## End(Not run)
```

---

calculate_hms *Calculate wildfire smoke covariates*

Description

Extract wildfire smoke plume values at point or buffered locations. Returns a `data.frame` object containing `locs_id`, date, and either binary indicators (`frac = FALSE`) or fractional overlap values (`frac = TRUE`) for wildfire smoke plume density inherited from `from`.

Usage

```
calculate_hms(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  weights = NULL,
  .by_time = NULL,
  frac = FALSE,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatVector(1). Output of `process_hms()`.
`locs`	data.frame, characater to file path, SpatVector, or sf object.

locs_id	character(1). Column within locations CSV file containing identifier for each unique coordinate location.
radius	integer(1). Circular buffer distance around site locations. (Default = 0).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used when .by_time is provided. When supplied, HMS indicators are summarized by sum (smoke-day counts) for frac = FALSE, or mean for frac = TRUE.
frac	logical(1). Default FALSE. If FALSE, return binary 0/1 smoke indicators by density class. If TRUE, return fractional overlap by density class.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object. When .by_time is provided, rows are aggregated using calc_summarize_by().

Author(s)

Mitchell Manware

See Also

process_hms()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_hms(
  from = hms, # derived from process_hms() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  geom = FALSE
)

## End(Not run)
```

`calculate_huc` *Calculate HUC covariates*

Description

Extract HUC IDs at point locations. Returns a `data.frame` object containing `locs_id` and HUC IDs.

Usage

```
calculate_huc(
  from,
  locs,
  locs_id = "site_id",
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatVector(1). Output from `process_huc()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`geom`	FALSE/"sf"/"terra".. Should the function return with geometry? Default is `FALSE`, options with geometry are "sf" or "terra". The coordinate reference system of the `sf` or `SpatVector` is that of `from`.
`...`	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Insang Song

See Also

`process_huc()`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_huc(
  from = huc, # derived from process_huc() example
  locs = loc,
  locs_id = "id",
  geom = FALSE
)

## End(Not run)
```

`calculate_koppen_geiger` *Calculate climate classification covariates*

Description

Extract Koppen-Geiger climate classes at point or buffered locations. Returns a `data.frame` with `locs_id`, a `description` column, and either binary indicators (`frac = FALSE`) or fractional overlap values (`frac = TRUE`) for climate groups A-E.

Usage

```
calculate_koppen_geiger(
  from = NULL,
  locs = NULL,
  locs_id = "site_id",
  weights = NULL,
  geom = FALSE,
  frac = FALSE,
  radius = 0,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output of `process_koppen_geiger()`.
`locs`	sf/SpatVector. Unique locs. Should include a unique identifier field named `locs_id`
`locs_id`	character(1). Name of unique identifier.
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.

geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
frac	logical(1). Default FALSE. If FALSE, return binary 0/1 indicators by climate group. If TRUE, return fractional overlap in the extraction footprint.
radius	numeric(1). Circular buffer size (meters) around point locations. Use 0 (default) for exact point extraction.
...	Placeholders.

Value

a data.frame or SpatVector object with climate columns named like DUM_CLRGA_00000 (frac = FALSE) or FRC_CLRGA_100000 (frac = TRUE) where the suffix reflects the extraction radius.

Note

The returned object contains a $description column to represent the temporal range covered by the dataset. For more information, see https://www.nature.com/articles/sdata2018214.

Author(s)

Insang Song

See Also

process_koppen_geiger

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_koppen_geiger(
  from = kg, # derived from process_koppen_geiger() example
  locs = loc,
  locs_id = "id",
  geom = FALSE
)

## End(Not run)
```

calculate_lagged *Calculate temporally lagged covariates*

Description

The calculate_lagged() function calculates daily temporal lagged covariates from the output of calculate_covariates() or calc_*().

Usage

```
calculate_lagged(from, date, lag, locs_id, time_id = "time", geom = FALSE)
```

Arguments

from	data.frame(1). A data.frame containing calculated covariates returned from calculate_covariates() or calc_*().
date	character(2). Start and end dates of desired lagged covariates. Length of 10 each, format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
lag	integer(1). Number of lag days.
locs_id	character(1). Name of unique identifier.
time_id	character(1). Column containing time values.
geom	logical(1). Should the function return a SpatVector? Default is FALSE. The coordinate reference system of the SpatVector is that of from. To return as a SpatVector, from must also be a SpatVector

Value

a data.frame object

Note

In order to calculate temporally lagged covariates, from must contain at least the number of lag days before the desired start date. For example, if date = c("2024-01-01", "2024-01-31) and lag = 1, from must contain data starting at 2023-12-31. If from contains geometry features, calculate_lagged will return a column with geometry features of the same name. calculate_lagged() assumes that all columns other than time_id, locs_id, and fixed columns of "lat" and "lon", follow the genre, variable, lag, buffer radius format adopted in calc_setcolumns().

See Also

calculate_covariates()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
terracliamte_covar <- calculate_terraclimate(
  from = terraclimate, # derived from process_terraclimate() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)
calculate_lagged(
  from = terracliamte_covar,
  locs_id = "id",
  date = c("2023-01-02", "2023-01-10"),
  lag = 1,
  time_id = "time"
)

## End(Not run)
```

`calculate_merra2` *Calculate meteorological and atmospheric covariates*

Description

Extract meteorological and atmospheric values at point locations. Returns a `data.frame` object containing `locs_id`, date and hour, vertical pressure level, and meteorological or atmospheric variable. Variable column name reflects variable and circular buffer radius.

Usage

```
calculate_merra2(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatRaster(1). Output of process_merra2().
locs	data.frame, characater to file path, SpatVector, or sf object.
locs_id	character(1). Column within locations CSV file containing identifier for each unique coordinate location.
radius	integer(1). Circular buffer distance around site locations. (Default = 0).
fun	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = mean).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders

Value

a data.frame or SpatVector object. When .by_time is provided, rows are aggregated using calc_summarize_by().

Author(s)

Mitchell Manware

See Also

calculate_geos(), process_merra2()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_merra2(
  from = merra2, # derived from process_merra2() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

`calculate_modis` *Calculate MODIS product covariates*

Description

`calculate_modis` orchestrates daily extraction using `calculate_modis_daily()`. In raw-path mode, files are grouped by inferred date, preprocessed for each day, and then extracted over requested radii. With product-specific preprocessing, files are handled according to each product's structure and naming conventions.

Usage

```
calculate_modis(
  from = NULL,
  from_secondary = NULL,
  locs = NULL,
  locs_id = "site_id",
  radius = c(0L, 1000L, 10000L, 50000L),
  preprocess = amadeus::process_modis_merge,
  name_covariates = NULL,
  subdataset = NULL,
  fun_summary = "mean",
  .by_time = NULL,
  weights = NULL,
  package_list_add = NULL,
  export_list_add = NULL,
  max_cells = 3e+07,
  geom = FALSE,
  scale = NULL,
  fusion_method = c("mean", "primary_first", "secondary_first"),
  ...
)
```

Arguments

`from`	character, SpatRaster, or SpatVector. Either a list of MODIS/VIIRS file paths (raw path mode), a preprocessed raster (direct raster mode), or processed MODIS fire detections as a SpatVector with `time`, `fire_count`, and `frp` fields.
`from_secondary`	character or SpatRaster. Optional secondary input for fused temporal coverage in raster/path workflows. Type must match `from` (`character` with `character`, or `SpatRaster` with `SpatRaster`).
`locs`	sf/SpatVector object. Unique locs where covariates will be calculated.
`locs_id`	character(1). Site identifier. Default is `"site_id"`
`radius`	numeric. Radii to calculate covariates. Default is `c(0, 1000, 10000, 50000)`.
`preprocess`	function. Function to handle HDF files in raw path mode. Ignored when `from` is a `SpatRaster` or `SpatVector`.

name_covariates
: character. Name header of covariates. e.g., "MOD_NDVIF_0_". The calculated covariate names will have a form of "{name_covariates}{zero-padded buffer radius in meters}", e.g., 'MOD_NDVIF_0_50000' where 50 km radius circular buffer was used to calculate mean NDVI value.

subdataset
: Indices, names, or search patterns for subdatasets. Find detail usage of the argument in notes.

fun_summary
: character or function. Function to summarize extracted raster values.

.by_time
: NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.

weights
: NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.

package_list_add
: character. Reserved for backward compatibility; currently not used by calculate_modis().

export_list_add
: character. Reserved for backward compatibility; currently not used by calculate_modis().

max_cells
: integer(1). Maximum number of cells to be read at once. Higher values will expedite processing, but will increase memory usage. Maximum possible value is 2^31 - 1. See exactextractr::exact_extract for details.

geom
: FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.

scale
: character(1). Scale expression to be applied to the raw values. It is crucial that users review the technical documentation of the MODIS product they are using to ensure proper scale. An example for the MOD11A1 product's LST_Day_1km variable (land surface temperature) would be scale = "* 0.02". Default is NULL, which applies no scale.

fusion_method
: character(1). Fusion method used only when from_secondary is provided. Options are "mean" (pixel-wise mean with na.rm = TRUE), "primary_first" (use from first), and "secondary_first" (use from_secondary first).

...
: Arguments passed to preprocess.

Value

A data.frame or SpatVector with an attribute:

- attr(., "dates_dropped"): Dates with insufficient tiles. Note that the dates mean the dates with insufficient tiles, not the dates without available tiles. When .by_time is provided, rows are summarized with calc_summarize_by() semantics.

Note

locs is expected to be convertible to sf object. sf, SpatVector, and other class objects that could be converted to sf can be used. In raw path mode, preprocess is called once per inferred day using a single-date value. Temporal aggregation across extracted rows should be done with .by_time.

Common arguments in `preprocess` functions such as `date` and `path` are automatically detected and passed to the function. Please note that `locs` here and `path` in `preprocess` functions are assumed to have a standard naming convention of raw files from NASA. The argument `subdataset` should be in a proper format depending on `preprocess` function:

- `process_modis_merge()`: Regular expression pattern. e.g., `"^LST_"`
- `process_modis_swath()`: Subdataset names. e.g., `c("Cloud_Fraction_Day", "Cloud_Fraction_Night")`
- `process_blackmarble()`: Subdataset number. e.g., for VNP46A2 product, 3L.

For MOD13/MYD13 families, Terra and Aqua composites are 16-day phased products offset by 8 days; combining both can improve effective temporal coverage. This behavior aligns with NASA MOD13 product guidance: https://modis.gsfc.nasa.gov/data/dataprod/mod13.php

For MCD19A2 MAIAC, common sub-datasets include both optical depth and plume injection height layers. Typical selectors are `"(Optical_Depth|Injection_Height)"` for both families or `"(Injection_Height)"` when targeting plume height only.

For MOD14A1/MYD14A1 fire grids, the `FireMask` raw values are commonly interpreted as:

Raw value	Meaning	Binary fire mask?
0	not processed, missing input	NA / no observation
1	obsolete, not used since Collection 1	NA
2	not processed, other reason	NA
3	non-fire water pixel	0
4	cloud, land or water	NA or 0, depending on analysis
5	non-fire land pixel	0
6	unknown, land or water	NA
7	fire, low confidence	1, or exclude for stricter mask
8	fire, nominal confidence	1
9	fire, high confidence	1

Dates with less than 80 percent of the expected number of tiles, which are determined by the mode of the number of tiles, are removed. Users will be informed of the dates with insufficient tiles. The result data.frame will have an attribute with the dates with insufficient tiles.

See Also

This function leverages the calculation of single-day MODIS covariates:

- `calculate_modis_daily()`

Also, for preprocessing, please refer to:

- `process_modis_merge()`
- `process_modis_swath()`
- `process_blackmarble()`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
locs <- data.frame(lon = -78.8277, lat = 35.95013, id = "001")
locs <- terra::vect(locs, geom = c("lon", "lat"), crs = "EPSG:4326")
calculate_modis(
  from =
    list.files("./data", pattern = "VNP46A2.", full.names = TRUE),
  locs = locs,
  locs_id = "site_id",
  radius = c(0L, 1000L),
  preprocess = process_modis_merge,
  name_covariates = "cloud_fraction_0",
  subdataset = "Cloud_Fraction",
  fun_summary = "mean"
)

## End(Not run)
```

---

calculate_narr *Calculate meteorological covariates*

Description

Extract meteorological values at point locations. Returns a `data.frame` object containing `locs_id`, date, vertical pressure level, and meteorological variable. Meteorological variable column name reflects variable and circular buffer radius.

Usage

```
calculate_narr(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output of `process_narr()`.
`locs`	data.frame, characater to file path, SpatVector, or sf object.

locs_id	character(1). Column within locations CSV file containing identifier for each unique coordinate location.
radius	integer(1). Circular buffer distance around site locations. (Default = 0).
fun	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = mean).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used when .by_time is provided.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders

Value

a data.frame or SpatVector object

Author(s)

Mitchell Manware

See Also

process_narr

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_narr(
  from = narr, # derived from process_narr() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

`calculate_nei` *Calculate road emissions covariates*

Description

Calculate road emissions covariates

Usage

```
calculate_nei(
  from = NULL,
  locs = NULL,
  locs_id = "site_id",
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatVector(1). Output of `process_nei()`.
`locs`	sf/SpatVector. Locations at NEI values are joined.
`locs_id`	character(1). Unique site identifier column name. Unused but kept for compatibility.
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`geom`	FALSE/"sf"/"terra".. Should the function return with geometry? Default is `FALSE`, options with geometry are "sf" or "terra". The coordinate reference system of the `sf` or `SpatVector` is that of `from`.
`...`	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Insang Song, Ranadeep Daw

See Also

`process_nei`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_nei(
  from = nei, # derived from process_nei example
  locs = loc,
  locs_id = "id"
)

## End(Not run)
```

`calculate_nlcd` *Calculate land cover covariates*

Description

Compute ratio of land cover class in circle buffers around points. Returns a `data.frame` object containing `locs_id`, longitude, latitude, time (year), and computed ratio for each land cover class.

Usage

```
calculate_nlcd(
  from,
  locs,
  locs_id = "site_id",
  mode = c("exact", "terra"),
  radius = 1000,
  drop = FALSE,
  weights = NULL,
  max_cells = 5e+07,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output of `process_nlcd()`.
`locs`	terra::SpatVector of points geometry
`locs_id`	character(1). Unique identifier of locations
`mode`	character(1). One of `"exact"` (using `exactextractr::exact_extract()`) or `"terra"` (using `terra::freq()`). Ignored if `locs` are points.
`radius`	numeric (non-negative) giving the radius of buffer around points.
`drop`	logical(1). Default `FALSE`. For buffered outputs (`radius > 0`), retain NLCD class columns even when all values are 0 (`drop = FALSE`) or remove class columns that are all 0 across all locations (`drop = TRUE`).

weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
max_cells	integer(1). Maximum number of cells to be read at once. Higher values may expedite processing, but will increase memory usage. Maximum possible value is 2^31 - 1. Only valid when mode = "exact". See exactextractr::exact_extract for details.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Note

NLCD is available in U.S. only. Users should be aware of the spatial extent of the data. The results are different depending on mode argument. The "terra" mode is less memory intensive but less accurate because it counts the number of cells intersecting with the buffer. The "exact" may be more accurate but uses more memory as it will account for the partial overlap with the buffer.

See Also

process_nlcd

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_nlcd(
  from = nlcd, # derived from process_nlcd() example
  locs = loc,
  locs_id = "id",
  mode = "exact",
  geom = FALSE
)

## End(Not run)
```

`calculate_population` *Calculate population density covariates*

Description

Extract population density values at point locations. Returns a `data.frame` object containing `locs_id`, year, and population density variable. Population density variable column name reflects spatial resolution of `from` and circular buffer radius.

Usage

```
calculate_population(
  from,
  locs,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output of `process_population()`.
`locs`	data.frame, characater to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`radius`	integer(1). Circular buffer distance around site locations. (Default = 0).
`fun`	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = `mean`).
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`geom`	FALSE/"sf"/"terra".. Should the function return with geometry? Default is `FALSE`, options with geometry are "sf" or "terra". The coordinate reference system of the `sf` or `SpatVector` is that of `from`.
`...`	Placeholders

Value

a data.frame or SpatVector object

Author(s)

Mitchell Manware

See Also

process_population()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_population(
  from = pop, # derived from process_population() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

---

calculate_prism *Calculate PRISM covariates*

Description

Extract PRISM values at point locations. Returns a `data.frame` object containing `locs_id` and PRISM variable. PRISM variable column name reflects the PRISM variable and circular buffer radius.

Usage

```
calculate_prism(
  from,
  locs,
  locs_id = "site_id",
  radius = 0,
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output from `process_prism()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.

radius	integer(1). Circular buffer distance around site locations. (Default = 0).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
.by_time	NULL or character(1). Optional time grouping key used with .by_time for temporal summaries.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Insang Song

See Also

process_prism()

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_prism(
  from = prism, # derived from process_prism() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  geom = FALSE
)

## End(Not run)
```

---

calculate_temporal_dummies

Calculate temporal dummy covariates

Description

Calculate temporal dummy covariates at point locations. Returns a data.frame object with locs_id, year binary variable for each value in year, and month and day of week binary variables.

Usage

```
calculate_temporal_dummies(
  locs,
  locs_id = "site_id",
  year = seq(2018L, 2022L),
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

locs	data.frame with a temporal field named "time"
locs_id	character(1). Unique site identifier column name. Default is "site_id".
year	integer. Year domain to dummify. Default is seq(2018L, 2022L).
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_temporal_dummies(
  locs = loc,
  locs_id = "id",
  year = seq(2018L, 2022L)
)

## End(Not run)
```

`calculate_terraclimate` *Calculate TerraClimate covariates*

Description

Extract TerraClimate values at point locations. Returns a `data.frame` object containing `locs_id` and TerraClimate variable. TerraClimate variable column name reflects the TerraClimate variable and circular buffer radius. The `$time` column will contain the year and month ("YYYYMM") as TerraClimate products have monthly temporal resolution.

Usage

```
calculate_terraclimate(
  from = NULL,
  locs = NULL,
  locs_id = NULL,
  radius = 0,
  fun = "mean",
  weights = NULL,
  .by_time = NULL,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster(1). Output from `process_terraclimate()`.
`locs`	data.frame. character to file path, SpatVector, or sf object.
`locs_id`	character(1). Column within `locations` CSV file containing identifier for each unique coordinate location.
`radius`	integer(1). Circular buffer distance around site locations. (Default = 0).
`fun`	character(1). Function used to summarize multiple raster cells within sites location buffer (Default = `mean`).
`weights`	`NULL`, `SpatRaster`, polygon `SpatVector`/`sf`, or file path. Optional weights raster for weighted extraction. If `NULL` (default), unweighted extraction is performed.
`.by_time`	NULL or character(1). Optional time grouping key used with `.by_time` for temporal summaries.
`geom`	FALSE/"sf"/"terra".. Should the function return with geometry? Default is `FALSE`, options with geometry are "sf" or "terra". The coordinate reference system of the `sf` or `SpatVector` is that of `from`.
`...`	Placeholders.

Value

a data.frame or SpatVector object

Note

TerraClimate data has monthly temporal resolution, so the `$time` column will contain the year and month in YYYYMM format (ie. January, 2018 = 201801).

Author(s)

Mitchell Manware

See Also

`process_terraclimate()`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_terraclimate(
  from = terraclimate, # derived from process_terraclimate() example
  locs = loc,
  locs_id = "id",
  radius = 0,
  fun = "mean",
  geom = FALSE
)

## End(Not run)
```

`calculate_tri` *Calculate toxic release covariates*

Description

Calculate toxic release values for polygons or isotropic buffer point locations. Returns a `data.frame` object containing `locs_id` and variables for each processed TRI field in `from`. Target fields are derived from metadata attached by `process_tri()`, with a fallback to non-coordinate columns in `from`.

Usage

```
calculate_tri(
  from = NULL,
  locs,
  locs_id = "site_id",
  decay_range = c(1000L, 10000L, 50000L),
  C0 = NULL,
  use_threshold = TRUE,
  weights = NULL,
  geom = FALSE,
  ...
)
```

Arguments

from	SpatVector(1). Output of process_tri().
locs	sf/SpatVector. Locations where TRI variables are calculated.
locs_id	character(1). Unique site identifier column name. Default is "site_id".
decay_range	Circular buffer radius. Default is c(1000, 10000, 50000) (meters)
C0	NULL or character vector of column names in from. Optional source-value columns used by sum_edc(). If NULL and there is one TRI target field, that field is inferred with a warning. If NULL and there are multiple TRI target fields, each TRI target field is used as its own source values (for example STACK_AIR_*).
use_threshold	logical(1). Passed to sum_edc(). If TRUE (default), include only source points within 5 * decay_range. If FALSE, include all source points in from.
weights	NULL, SpatRaster, polygon SpatVector/sf, or file path. Optional weights raster for weighted extraction. If NULL (default), unweighted extraction is performed.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.
...	Placeholders.

Value

a data.frame or SpatVector object

Note

U.S. context.

Author(s)

Insang Song, Mariana Kassien

See Also

sum_edc, process_tri

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
loc <- data.frame(id = "001", lon = -78.90, lat = 35.97)
calculate_tri(
  from = tri, # derived from process_tri() example
  locs = loc,
  locs_id = "id",
  decay_range = c(1e3L, 1e4L, 5e4L)
)

## End(Not run)
```

---

download_aqs *Download air quality data*

Description

The download_aqs() function accesses and downloads Air Quality System (AQS) data from the U.S. Environmental Protection Agency's (EPA) Pre-Generated Data Files.

Usage

```
download_aqs(
  parameter_code = 88101,
  resolution_temporal = "daily",
  year = c(2018, 2022),
  url_aqs_download = "https://aqs.epa.gov/aqsweb/airdata/",
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

parameter_code	integer(1). EPA pollutant parameter code. See Details for a short list of common codes.
resolution_temporal	character(1). Currently only "daily" is supported.
year	integer(1 or 2). Year or start/end years for downloading data.
url_aqs_download	character(1). URL to the AQS pre-generated datasets.
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Details

Common AQS parameter codes include:

- 88101 — PM2.5 - Local Conditions
- 88502 — Acceptable PM2.5 AQI & Speciation Mass
- 81102 — PM10 Total 0-10um STP
- 44201 — Ozone
- 42602 — Nitrogen dioxide (NO2)
- 42401 — Sulfur dioxide (SO2)
- 42101 — Carbon monoxide

This list is not exhaustive; for the full official table, see the linked EPA AQS parameter code table.

Value

invisible list with download results; or hash character if hash=TRUE

Note

AQS data does not require authentication. AQS measurements are generally intended for use as dependent variables, so the package supports download and processing for AQS but does not expose AQS through `calculate_covariates()`.

Author(s)

Mariana Kassien, Insang Song, Mitchell Manware

References

U.S. Environmental Protection Agency (2023). "Air Quality System Data Mart [internet database]." https://www.epa.gov/outdoor-air-quality-data.

See Also

EPA AQS Parameter Codes

Examples

```
## Not run:
download_aqs(
  parameter_code = 88101,
  resolution_temporal = "daily",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_cropscape *Download CropScape data*

Description

Accesses and downloads United States Department of Agriculture CropScape Cropland Data Layer data from the USDA National Agricultural Statistics Service or the George Mason University website.

Usage

```
download_cropscape(
  year = seq(1997, 2023),
  source = c("USDA", "GMU"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

year	integer(1). Year of the data to download.
source	character(1). Data source, one of c("USDA", "GMU"). • "USDA" will download the national data from the USDA website (available in 2008-last year). • "GMU" will download the data from the George Mason University website (available in 1997-last year).
directory_to_save	character(1). Directory to download files.
acknowledgement	logical(1). By setting TRUE the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.
download	logical(1). FALSE will generate a *.txt file containing all download commands. By setting TRUE the function will download all of the requested data files.
remove_command	logical(1). Remove (TRUE) or keep (FALSE) the text file containing download commands.
unzip	logical(1). Unzip the downloaded compressed files. Default is FALSE.
hash	logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.
show_progress	logical(1). Show download progress. Default is TRUE.
max_tries	integer(1). Maximum download retry attempts. Default is 20.
rate_limit	numeric(1). Minimum seconds between requests. Default is 2.

Value

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- Yearly comma-separated value (CSV) files will be stored in directory_to_save.

Note

JSON files should be found at STAC catalog of OpenLandMap

Author(s)

Insang Song

Examples

```
## Not run:
download_cropscape(
  year = 2020,
  source = "USDA",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
```

```
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## End(Not run)
```

`download_data` *Download raw data wrapper function*

Description

The `download_data()` function accesses and downloads atmospheric, meteorological, and environmental data from various open-access data sources.

Usage

```
download_data(
 dataset_name = c("aqs", "ecoregion", "ecoregions", "geos", "goes", "goes_adp", "GOES",
   "gmted", "koppen", "koppengeiger", "merra2", "merra", "modis", "narr", "nlcd",
   "noaa", "sedac_groads", "sedac_population", "groads", "population", "hms", "smoke",
   "tri", "nei", "gridmet", "terraclimate", "huc", "cropscape", "cdl", "prism", "edgar",
    "improve", "IMPROVE", "drought", "spei", "eddi", "usdm"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  hash = FALSE,
  nasa_earth_data_token = NULL,
  rate_limit = 2,
  ...
)
```

Arguments

`dataset_name`	character(1). Dataset to download.
`directory_to_save`	character(1). Directory to save / unzip (if zip files are downloaded) data.
`acknowledgement`	logical(1). By setting `TRUE` the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.
`nasa_earth_data_token`	character(1) or NULL. NASA EarthData authentication token. Required for NASA EarthData datasets: `"geos"`, `"merra2"`, `"modis"`, `"sedac_groads"` / `"groads"`, and `"sedac_population"` / `"population"`. Can be a token string, a path to a file containing the token, or `NULL` to read from the `NASA_EARTHDATA_TOKEN` environment variable. Ignored for datasets that do not use NASA EarthData authentication.

rate_limit	numeric(1). Minimum seconds between HTTP requests (default 2). Passed to the underlying download function.
...	Additional arguments passed to each download function.

Value

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- Data files will be downloaded and stored in respective sub-directories within directory_to_save. File format and sub-directory names depend on data source and dataset of interest.

Note

- All download function names are in download_* formats

Author(s)

Insang Song

See Also

For details of each download function per dataset, Please refer to:

- download_aqs: "aqs", "AQS"
- download_ecoregion: "ecoregions", "ecoregion"
- download_geos: "geos"
- download_goes: "goes", "goes_adp", "GOES"
- download_gmted: "gmted", "GMTED"
- download_koppen_geiger: "koppen", "koppengeiger"
- download_merra2: "merra2", "merra", "MERRA", "MERRA2"
- download_narr: "narr"
- download_nlcd: "nlcd", "NLCD"
- download_hms: "noaa", "smoke", "hms"
- download_groads: "sedac_groads", "groads"
- download_population: "sedac_population", "population"
- download_modis: "modis", "MODIS"
- download_tri: "tri", "TRI"
- download_nei: "nei", "NEI"
- download_gridmet: "gridMET", "gridmet"
- download_terraclimate: "TerraClimate", "terraclimate"
- download_huc: "huc"
- download_cropscape: "cropscape", "cdl"
- download_prism: "prism"
- download_edgar: "edgar"
- download_improve: "improve", "IMPROVE"
- download_drought: "drought", "spei", "eddi", "usdm"

Examples

```
## Not run:
download_data(
  dataset_name = "narr",
  variables = "weasd",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE
)

## End(Not run)
```

---

download_drought *Download drought index data*

Description

The `download_drought()` function downloads drought index data from publicly available sources. Three source datasets are supported:

- **SPEI** (Standardized Precipitation-Evapotranspiration Index): Multi-year netCDF files by timescale from https://spei.csic.es.
- **EDDI** (Evaporative Demand Drought Index): Weekly raster files by timescale from https://www.drought.gov/data-maps-tools/evaporative-demand-drought-index-eddi.
- **USDM** (U.S. Drought Monitor): Weekly drought class shapefiles from https://droughtmonitor.unl.edu.

Usage

```
download_drought(
  source = c("spei", "eddi", "usdm"),
  date = c("2020-01-01", "2020-12-31"),
  timescale = 1L,
  directory_to_save = NULL,
  acknowledgement = FALSE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 3L,
  rate_limit = 2,
  unzip = TRUE,
  remove_zip = FALSE,
  ...
)
```

Arguments

source	character(1). Drought data source. One of "spei", "eddi", or "usdm".
date	character(1 or 2). Single date or start/end dates. Format "YYYY-MM-DD". For SPEI/EDDI the year component selects the annual file(s); for USDM the full date is used to select weekly release(s).
timescale	integer(1). Accumulation timescale in months (SPEI/EDDI only; ignored for USDM). Typical values are 1, 3, 6, 12, 24, 48. Default is 1L.
directory_to_save	character(1). Directory to save downloaded data.
acknowledgement	logical(1). Must be TRUE to proceed.
hash	logical(1). Return rlang::hash_file() hash of downloaded files. Default FALSE.
show_progress	logical(1). Show download progress bar. Default TRUE.
max_tries	integer(1). Maximum retry attempts. Default 3L.
rate_limit	numeric(1). Minimum seconds between HTTP requests. Default 2.
unzip	logical(1). Unzip downloaded zip archives (USDM only). Default TRUE.
remove_zip	logical(1). Remove zip archives after unzipping (USDM only). Default FALSE.
...	Reserved for future use; currently ignored.

Value

invisible(NULL) when hash = FALSE; a character hash string when hash = TRUE.

Note

- SPEI and EDDI are raster products; USDM is a polygon product (shapefile). Their process_drought() and calculate_drought() handling differ accordingly.
- No authentication is required for any of these sources.

Author(s)

Insang Song

See Also

process_drought, calculate_drought

Examples

```
## Not run:
download_drought(
  source = "spei",
  date = c("2020-01-01", "2020-12-31"),
  timescale = 1L,
  directory_to_save = "./data/drought",
```

```
  acknowledgement = TRUE
)
download_drought(
  source = "usdm",
  date = c("2020-01-07", "2020-03-31"),
  directory_to_save = "./data/drought",
  acknowledgement = TRUE
)

## End(Not run)
```

`download_ecoregion` *Download ecoregion data*

Description

The `download_ecoregion()` function accesses and downloads United States Ecoregions data from the U.S. Environmental Protection Agency's (EPA) Ecoregions.

Usage

```
download_ecoregion(
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`directory_to_save`	character(1). Directory to save data.
`acknowledgement`	logical(1). Must be TRUE to proceed.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). Deprecated, ignored.
`unzip`	logical(1). Unzip zip files (default TRUE).
`remove_zip`	logical(1). Remove zip files after unzipping (default FALSE).
`show_progress`	logical(1). Show download progress (default TRUE)
`hash`	logical(1). Return hash of downloaded files (default FALSE)
`max_tries`	integer(1). Maximum retry attempts (default 20)
`rate_limit`	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

Ecoregion data does not require authentication.

Author(s)

Insang Song

References

Omernik JM, Griffith GE (2014). "Ecoregions of the Conterminous United States: Evolution of a Hierarchical Spatial Framework." *Environmental Management*, **54**(6), 1249–1266. ISSN 0364-152X, 1432-1009, doi:10.1007/s0026701403641, https://link.springer.com/article/10.1007/s00267-014-0364-1.

Examples

```
## Not run:
download_ecoregion(
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_edgar *Download EDGAR Emissions Data*

Description

Constructs and optionally downloads EDGAR emissions data URLs based on user-specified inputs including species, temporal resolution, emission sectors, and file formats.

Usage

```
download_edgar(
  species = c("BC", "CO", "NH3", "NMVOC", "NOx", "OC", "PM10", "PM2.5", "SO2"),
  version = "8.1",
  temp_res = NULL,
  sector_yearly = NULL,
  sector_monthly = NULL,
  sector_voc = NULL,
  format = "nc",
  output = "emi",
  year_range = NULL,
```

```
  voc = NULL,
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

species	Character vector. One or more species to download. Supported values: "BC", "CO", "NH3", "NMVOC", "NOx", "OC", "PM10", "PM2.5", "SO2". Input is case-insensitive and supports "pm2.5" or "pm25".
version	Character. EDGAR data version. Supported values: "8.1" for most recent version data or "8.1_voc" for VOC speciation data.
temp_res	Character. Temporal resolution for specification with version 8.1. One of "yearly", "monthly", or "timeseries". temp_res is not needed for version=8.1_voc and will be ignored if specified.
sector_yearly	Character vector or NULL. Emission sectors for yearly data. If NULL, totals will be used. Possible values include: "AGS", "AWB", "CHE", "ENE", "IND", "MNM", "NMM", "PRU_SOL", "RCO", "REF_TRF", "SWD_INC", "SWD_LDF", "TNR_Aviation_CDS", "TNR_Aviation_CRS", "TNR_Aviation_LTO", "TNR_Aviation_SPS", "TNR_Other", "TNR_Ship", "TRO", "WWT"
sector_monthly	Character vector or NULL. Emission sectors for monthly data. If NULL, the function will use full-species files (not sector-specific). Supported values: "AGRICULTURE", "BUILDINGS", "FUEL_EXPLOITATION", "IND_COMBUSTION", "IND_PROCESSES", "POWER_INDUSTRY", "TRANSPORT", "WASTE".
sector_voc	Character vector or NULL. Emission sectors for VOC speciation data. If NULL, the function will use full-species files (not sector-specific). Supported values: "AGRICULTURE", "BUILDINGS", "FUEL_EXPLOITATION", "IND_COMBUSTION", "IND_PROCESSES", "POWER_INDUSTRY", "TRANSPORT", "WASTE".
format	Character. File format to download. Typically "nc" (NetCDF) or "txt". Flux output and monthly outputs are only supported in .nc format
output	Character. Output type. Supported values include "emi" for emissions and "flx" for fluxes.
year_range	Numeric vector of length 1, 2 or NULL. Year range, e.g., 2021, or c(2021, 2022). If NULL, uses all available years (1970-2022 for yearly data, 2000-2022 for monthly and VOC speciation data)
voc	Integer vector or NULL. Used for VOC speciation in version "8.1_voc". Accepts integers from 1 to 25. See: https://edgar.jrc.ec.europa.eu/dataset_ap81_VOC_spec#p3 for reference on speciation groups and VOC numbers.

directory_to_save
: character(1). Directory to save data. Two sub-directories will be created for the downloaded zip files ("/zip_files") and the unzipped data files ("/data_files").

acknowledgement
: logical(1). By setting TRUE the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.

download
: logical(1). FALSE will generate a *.txt file containing all download commands. By setting TRUE the function will download all of the requested data files.

remove_command
: logical(1). Remove (TRUE) or keep (FALSE) the text file containing download commands. Default is FALSE.

unzip
: logical(1). Unzip zip files. Default is TRUE.

remove_zip
: logical(1). Remove zip file from directory_to_download. Default is FALSE.

hash
: logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.

show_progress
: logical(1). Show download progress. Default is TRUE.

max_tries
: integer(1). Maximum download retry attempts. Default is 20.

rate_limit
: numeric(1). Minimum seconds between requests. Default is 2.

Value

A list of download URLs (character). Optionally downloads available files and warns about missing ones.

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- Zip and/or data files will be downloaded and stored in directory_to_save.

Author(s)

Mariana Alifa Kassien

Examples

```
## Not run:
download_edgar(
species = "CO",
acknowledgement = TRUE,
temp_res = "yearly",
sector_yearly = "ENE",
year_range = c(2021, 2022)
)

## End(Not run)
## Not run:
download_edgar(
species = "PM2.5",
acknowledgement = TRUE,
```

```
temp_res = "monthly",
sector_monthly = c("TRANSPORT", "WASTE")
)

## End(Not run)
## Not run:
download_edgar(
species = "SO2",
acknowledgement = TRUE,
temp_res = "timeseries"
)

## End(Not run)
```

---

download_geos *Download atmospheric composition data*

Description

The download_geos() function accesses and downloads various atmospheric composition collections from NASA's Global Earth Observing System (GEOS) compositional forecast model.

Usage

```
download_geos(
  collection = c("aqc_tavg_1hr_g1440x721_v1", "chm_tavg_1hr_g1440x721_v1",
    "met_tavg_1hr_g1440x721_x1", "xgc_tavg_1hr_g1440x721_x1",
    "chm_inst_1hr_g1440x721_p23", "met_inst_1hr_g1440x721_p23"),
  nasa_earth_data_token = NULL,
  date = c("2018-01-01", "2018-01-01"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

collection	character(1). GEOS-CF data collection file name.
nasa_earth_data_token	character(1) or NULL. NASA EarthData authentication token.
date	character(1 or 2). Date range "YYYY-MM-DD" format
directory_to_save	character(1). Directory to save data.

acknowledgement	logical(1). Must be TRUE to proceed
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

Due to NASA data access policies, downloads require a valid NASA Earthdata token for authentication. Use `setup_nasa_token()` for setup.

Author(s)

Mitchell Manware, Insang Song

References

Keller CA, Knowland KE, Duncan BN, Liu J, Anderson DC, Das S, Lucchesi RA, Lundgren EW, Nicely JM, Nielsen E, Ott LE, Saunders E, Strode SA, Wales PA, Jacob DJ, Pawson S (2021). "Description of the NASA GEOS Composition Forecast Modeling System GEOS-CF v1.0." *Journal of Advances in Modeling Earth Systems*, **13**(4), e2020MS002413. ISSN 1942-2466, 1942-2466, doi:10.1029/2020MS002413.

Examples

```
## Not run:
download_geos(
  collection = "aqc_tavg_1hr_g1440x721_v1",
  date = "2024-01-01",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

download_gmted *Download elevation data*

Description

The download_gmted() function accesses and downloads Global Multi-resolution Terrain Elevation Data (GMTED2010) from U.S. Geological Survey.

Usage

```
download_gmted(
  statistic = c("Breakline Emphasis", "Systematic Subsample", "Median Statistic",
    "Minimum Statistic", "Mean Statistic", "Maximum Statistic",
    "Standard Deviation Statistic"),
  resolution = c("7.5 arc-seconds", "15 arc-seconds", "30 arc-seconds"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

statistic	character(1). Available statistics.
resolution	character(1). Available resolutions.
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

GMTED data does not require authentication.

Author(s)

Mitchell Manware, Insang Song

References

Danielson JJ, Gesch DB (2011). "Global multi-resolution terrain elevation data 2010 (GMTED2010)." Open-File Report 2011-1073, U.S. Geological Survey. Series: Open-File Report, https://doi.org/10.3133/ofr20111073.

Examples

```
## Not run:
download_gmted(
  statistic = "Breakline Emphasis",
  resolution = "7.5 arc-seconds",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_goes *Download NOAA GOES ADP data*

Description

The `download_goes()` function accesses and downloads NOAA GOES-16 or GOES-18 Aerosol Detection Product (ADP) files from the NOAA Open Data Dissemination (NODD) AWS S3 bucket. Files are in NetCDF format and contain aerosol detection variables (e.g. `"Smoke"`, `"Dust"`) on the GOES fixed geostationary grid.

Usage

```
download_goes(
  date = c("2024-01-01", "2024-01-01"),
  satellite = "16",
  product = "ADP-C",
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
```

```
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

date	character(1 or 2). Date (YYYY-MM-DD) or start and end dates.
satellite	character(1). GOES satellite number: "16" (East, default) or "18" (West).
product	character(1). ADP scan sector: "ADP-C" (CONUS, default), "ADP-F" (Full Disk), or "ADP-M" (Mesoscale).
directory_to_save	character(1). Directory to save downloaded files.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
show_progress	logical(1). Show download progress (default TRUE).
hash	logical(1). Return hash of downloaded files (default FALSE).
max_tries	integer(1). Maximum retry attempts (default 20).
rate_limit	numeric(1). Minimum seconds between requests (default 2).

Value

invisible list with download results; or hash character if hash = TRUE

Note

- GOES data does not require authentication.
- GOES-16 (East) covers the Americas; GOES-18 (West) covers the western hemisphere and Pacific.
- ADP-C (CONUS) scans are produced approximately every 5 minutes. A single day may contain several hundred files.
- GOES ADP files use the GOES fixed geostationary projection. Use process_goes() to load and reproject to EPSG:4326.

Author(s)

Mitchell Manware

Examples

```
## Not run:
download_goes(
  date = "2024-01-01",
  satellite = "16",
  product = "ADP-C",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_gridmet` *Download gridMET data*

Description

The `download_gridmet` function accesses and downloads gridded surface meteorological data from the University of California Merced Climatology Lab's gridMET dataset.

Usage

```
download_gridmet(
  variables = NULL,
  year = c(2018, 2022),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`variables`	character. Variable(s) name(s).
`year`	integer(1 or 2). Year or start/end years for downloading data.
`directory_to_save`	character(1). Directory to save data.
`acknowledgement`	logical(1). Must be TRUE to proceed.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). Deprecated, ignored.
`show_progress`	logical(1). Show download progress (default TRUE)

hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

gridMET data does not require authentication.

Author(s)

Mitchell Manware

References

Abatzoglou JT (2013). "Development of gridded surface meteorological data for ecological applications and modelling." *International journal of climatology*, **33**(1), 121–131.

Examples

```
## Not run:
download_gridmet(
  variables = "pr",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_groads *Download roads data*

Description

The download_groads() function accesses and downloads roads data from NASA's Global Roads Open Access Data Set (gROADS).

Usage

```
download_groads(
 data_region = c("Americas", "Global", "Africa", "Asia", "Europe", "Oceania East",
    "Oceania West"),
  data_format = c("Shapefile", "Geodatabase"),
  nasa_earth_data_token = NULL,
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

data_region	character(1). Data region.
data_format	character(1). "Shapefile" or "Geodatabase".
nasa_earth_data_token	character(1) or NULL. NASA EarthData authentication token. Can be a token string, a path to a file containing the token, or NULL to read from the NASA_EARTHDATA_TOKEN environment variable.
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

gROADS data is hosted on NASA EarthData and requires a valid NASA EarthData token for authentication. Set the `NASA_EARTHDATA_TOKEN` environment variable or pass the token directly via `nasa_earth_data_token`. Use `setup_nasa_token()` for setup.

Author(s)

Mitchell Manware, Insang Song

References

Center For International Earth Science Information Network-CIESIN-Columbia University, Information Technology Outreach Services-ITOS-University Of Georgia (2013). "Global Roads Open Access Data Set, Version 1 (gROADSv1)." doi:10.7927/H4VD6WCT, https://data.nasa.gov/dataset/global-roads-open-access-data-set-version-1-groadsv1.

Examples

```
## Not run:
download_groads(
  data_region = "Americas",
  data_format = "Shapefile",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_hms *Download wildfire smoke data*

Description

The `download_hms()` function accesses and downloads wildfire smoke plume coverage data from NOAA's Hazard Mapping System Fire and Smoke Product.

Usage

```
download_hms(
  data_format = "Shapefile",
  date = c("2018-01-01", "2018-01-01"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
```

```
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

data_format	character(1). "Shapefile" or "KML".
date	character(1 or 2). Date range "YYYY-MM-DD" format
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

HMS data does not require authentication.

Author(s)

Mitchell Manware, Insang Song

References

(????). "Hazard Mapping System Fire and Smoke Product: Hazard Mapping System." https://www.ospo.noaa.gov/products/land/hms.html#about. https://www.ospo.noaa.gov/products/land/hms.html#about.

Examples

```
## Not run:
download_hms(
  data_format = "Shapefile",
  date = "2024-01-01",
  directory_to_save = tempdir(),
```

```
  acknowledgement = TRUE
)

## End(Not run)
```

`download_huc` *Download National Hydrography Dataset (NHD) data*

Description

NHDPlus data provides the most comprehensive and high-resolution hydrography data. This function downloads **national** dataset from NHDPlus Version 2.1 on USGS Amazon S3 storage.

Usage

```
download_huc(
  region = c("Lower48", "Islands"),
  type = c("Seamless", "OceanCatchment"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = FALSE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`region`	character(1). One of `c("Lower48", "Islands")`. When `"Islands"` is selected, the data will be downloaded for Hawaii, Puerto Rico, and Virgin Islands.
`type`	character(1). One of `c("Seamless", "OceanCatchment")`.
`directory_to_save`	character(1). Directory to download files.
`acknowledgement`	logical(1). By setting `TRUE` the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.
`download`	logical(1). `FALSE` will generate a *.txt file containing all download commands. By setting `TRUE` the function will download all of the requested data files.
`remove_command`	logical(1). Remove (`TRUE`) or keep (`FALSE`) the text file containing download commands.
`unzip`	logical(1). Unzip the downloaded compressed files. Default is `FALSE`. Supports ".7z" extraction via **archive**.

hash	logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.
show_progress	logical(1). Show download progress. Default is TRUE.
max_tries	integer(1). Maximum download retry attempts. Default is 20.
rate_limit	numeric(1). Minimum seconds between requests. Default is 2.

Value

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- Downloaded files will be stored in directory_to_save.

Note

For HUC, set type = "Seamless". HUC12 layer presents in the seamless geodatabase. Users can aggregate HUC12 layer to make HUC6, HUC8, HUC10, etc. For whom wants to download a specific region, please visit Get NHDPlus Data

Author(s)

Insang Song

References

U.S. Geological Survey (2023). "National Hydrography Dataset (NHD) – USGS National Map Downloadable Data Collection." https://www.usgs.gov/national-hydrography.

Examples

```
## Not run:
download_huc(
  region = "Lower48",
  type = "Seamless",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## End(Not run)
```

`download_improve` *Download IMPROVE aerosol monitoring data*

Description

The `download_improve()` function accesses and downloads IMPROVE (Interagency Monitoring of Protected Visual Environments) data files from the VIEWS/VIBE data export service hosted at CIRA/CSU. Annual files are downloaded as `.txt.zip` archives and extracted to pipe-delimited `.txt` files containing aerosol measurements at federal-land monitoring stations.

Usage

```
download_improve(
  year = c(2018, 2022),
  product = c("raw", "rhr2", "rhr3"),
  url_improve = "https://vibe.cira.colostate.edu/data/export/",
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`year`	integer(1 or 2). Year or start/end years.
`product`	character(1). Product selector: `"raw"` (aerosol, default), `"rhr2"` (Regional Haze Rule II), or `"rhr3"` (Regional Haze Rule III).
`url_improve`	character(1). Base URL to the IMPROVE data export service.
`directory_to_save`	character(1). Directory to save downloaded files.
`acknowledgement`	logical(1). Must be `TRUE` to proceed.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). Deprecated, ignored.
`show_progress`	logical(1). Show download progress (default `TRUE`).
`hash`	logical(1). Return hash of downloaded files (default `FALSE`).
`max_tries`	integer(1). Maximum retry attempts (default 20).
`rate_limit`	numeric(1). Minimum seconds between requests (default 2).

Value

invisible list with download results; or hash character if `hash = TRUE`.

Note

- IMPROVE data does not require authentication.
- Three product types are available: `"raw"` (IMPAER — speciated aerosol mass concentrations), `"rhr2"` (IMPRHR2 — Regional Haze Rule II light extinction), `"rhr3"` (IMPRHR3 — Regional Haze Rule III deciview index).
- Site metadata is handled by `process_improve` using an embedded table; annual downloads include measurement files only.
- IMPROVE monitors ~$1 \mu g/m^3$ precision instruments deployed at Class I and other federal land areas.

Author(s)

Insang Song, Mitchell Manware

See Also

`process_improve`

Examples

```
## Not run:
download_improve(
  year = 2022,
  product = "raw",
  directory_to_save = "./data/improve/",
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_koppen_geiger

Download climate classification data

Description

The `download_koppen_geiger()` function accesses and downloads climate classification data.

Usage

```
download_koppen_geiger(
  data_resolution = c("0.0083", "0.083", "0.5"),
  time_period = c("Present", "Future"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

data_resolution	character(1). Available resolutions.
time_period	character(1). "Present" (1980-2016) or "Future" (2071-2100).
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

Köppen-Geiger data does not require authentication.

Author(s)

Mitchell Manware, Insang Song

References

Beck HE, McVicar TR, Vergopolan N, Berg A, Lutsko NJ, Dufour A, Zeng Z, Jiang X, Van Dijk AIJM, Miralles DG (2023). "High-resolution (1 km) Köppen-Geiger maps for 1901–2099 based on constrained CMIP6 projections." *Scientific Data*, **10**(1), 724. ISSN 2052-4463, doi:10.1038/s41597023025496, https://www.nature.com/articles/s41597-023-02549-6. Beck HE, Zimmermann NE, McVicar TR, Vergopolan N, Berg A, Wood EF (2018). "Present and future Köppen-Geiger climate classification maps at 1-km resolution." *Scientific data*, **5**(1), 1–12. doi:10.1038/sdata.2018.214.

Examples

```
## Not run:
download_koppen_geiger(
  data_resolution = "0.0083",
  time_period = "Present",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_merra2` *Download meteorological and atmospheric data*

Description

The `download_merra2()` function accesses and downloads various meteorological and atmospheric collections from NASA's Modern-Era Retrospective analysis for Research and Applications, Version 2 (MERRA-2) model, and the daily corrected Global Fire Weather Index (FWI) product derived from MERRA-2 weather inputs.

Usage

```
download_merra2(
  collection = c("inst1_2d_asm_Nx", "inst1_2d_int_Nx", "inst1_2d_lfo_Nx",
    "inst3_3d_asm_Np", "inst3_3d_aer_Nv", "inst3_3d_asm_Nv", "inst3_3d_chm_Nv",
    "inst3_3d_gas_Nv", "inst3_2d_gas_Nx", "inst6_3d_ana_Np", "inst6_3d_ana_Nv",
    "statD_2d_slv_Nx", "tavg1_2d_adg_Nx", "tavg1_2d_aer_Nx", "tavg1_2d_chm_Nx",
    "tavg1_2d_csp_Nx", "tavg1_2d_flx_Nx", "tavg1_2d_int_Nx", "tavg1_2d_lfo_Nx",
    "tavg1_2d_lnd_Nx", "tavg1_2d_ocn_Nx", "tavg1_2d_rad_Nx", "tavg1_2d_slv_Nx",
    "tavg3_3d_mst_Ne", "tavg3_3d_trb_Ne", "tavg3_3d_nav_Ne", "tavg3_3d_cld_Np",

    "tavg3_3d_mst_Np", "tavg3_3d_rad_Np", "tavg3_3d_tdt_Np", "tavg3_3d_trb_Np",
    "tavg3_3d_udt_Np", "tavg3_3d_odt_Np", "tavg3_3d_qdt_Np", "tavg3_3d_asm_Nv",
   "tavg3_3d_cld_Nv", "tavg3_3d_mst_Nv", "tavg3_3d_rad_Nv", "tavg3_2d_glc_Nx", "fwi"),
  nasa_earth_data_token = NULL,
  date = c("2018-01-01", "2018-01-01"),
```

```
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

collection	character(1). MERRA-2 data collection file name, or "fwi" for the daily corrected Global Fire Weather Index product (MERRA2.CORRECTED).
nasa_earth_data_token	character(1) or NULL. NASA EarthData authentication token.
date	character(1 or 2). length of 10. Date or start/end dates for downloading data. Format "YYYY-MM-DD" (ex. January 1, 2018 = "2018-01-01").
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). By setting TRUE the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
hash	logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.
show_progress	logical(1). Show download progress (default TRUE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

Due to NASA data access policies, standard MERRA-2 GES DISC downloads require a valid NASA Earthdata token for authentication. Use setup_nasa_token() for setup. The "fwi" collection is hosted on the public GlobalFWI portal and does not require Earthdata authentication.

Author(s)

Mitchell Manware, Insang Song, Kyle Messier

Examples

```
## Not run:
download_merra2(
  collection = "inst1_2d_int_Nx",
  date = "2024-01-01",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_modis` *Download MODIS product files*

Description

Downloads MODIS data using httr2 with robust retry logic and rate limiting. This function queries NASA's CMR API for available granules and downloads relevant tiles based on the specified extent.

Usage

```
download_modis(
  product = c("MOD09GA", "MYD09GA", "MOD09GQ", "MYD09GQ", "MOD09A1", "MYD09A1",
    "MOD09Q1", "MYD09Q1", "MOD11A1", "MYD11A1", "MOD11A2", "MYD11A2", "MOD11B1",
    "MYD11B1", "MOD13A1", "MYD13A1", "MOD13A2", "MYD13A2", "MOD13Q1", "MYD13Q1",
    "MOD13A3", "MYD13A3", "MCD12Q1", "MOD14A1", "MYD14A1", "MOD14A2", "MYD14A2",
   "MOD14CM1", "MYD14CM1", "MOD16A2", "MYD16A2", "MCD64A1", "MCD64CMQ", "MOD06_L2",
    "MCD14ML", "MCD19A2", "VNP46A2", "VNP64A1"),
  version = "061",
  nasa_earth_data_token = NULL,
  date = c("2023-09-01", "2023-09-01"),
  extent = c(-125, 22, -64, 50),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`product`	character(1). MODIS product code
`version`	character(1). Default is `"061"`, meaning v061.

`nasa_earth_data_token`
: character(1) or NULL. NASA EarthData authentication token. For security, recommended options (in priority order):
  - NULL (default): Reads from NASA_EARTHDATA_TOKEN environment variable
  - File path: e.g., "~/.nasa_earthdata_token"
  - Token string: Direct token (not recommended for scripts)

  Use `setup_nasa_token()` for interactive setup.

`date`
: character(1 or 2). Date range "YYYY-MM-DD" format

`extent`
: numeric(4). Bounding box `c(min_lon, max_lon, min_lat, max_lat)`. Default covers continental US: `c(-125, 22, -64, 50)`.

`directory_to_save`
: character(1). Directory to save data.

`acknowledgement`
: logical(1). Must be `TRUE` to proceed with download

`download`
: logical(1). DEPRECATED. Downloads now happen automatically. Set to FALSE to skip downloading (generates file list only).

`remove_command`
: logical(1). Deprecated, ignored.

`show_progress`
: logical(1). Show download progress (default TRUE)

`hash`
: logical(1). Return hash of downloaded files (default FALSE)

`max_tries`
: integer(1). Maximum download retry attempts (default 20)

`rate_limit`
: numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

Due to NASA data access policies, downloads require a valid NASA Earthdata token for authentication. For security, it's recommended to store your token in an environment variable or file rather than in your code. Use `setup_nasa_token()` for easy, secure token setup.

Both dates in `date` should be in the same year. Directory structure: input/modis/raw/{version}/{product}/{year}/{day_of_yea

Author(s)

Mitchell Manware, Insang Song

References

Lyapustin A, Wang Y (2022). "MODIS/Terra+Aqua Land Aerosol Optical Depth Daily L2G Global 1km SIN Grid V061." doi:10.5067/MODIS/MCD19A2.061, https://www.earthdata.nasa.gov/data/catalog/lpcloud-mcd19a2-061. MODIS Atmosphere Science Team (2017). "MODIS/Terra Clouds 5-Min L2 Swath 1km and 5km." doi:10.5067/MODIS/MOD06_L2.061, https://ladsweb.modaps.eosdis.nasa.gov/missions-and-measurements/products/MOD06_

L2. Vermote E, Wolfe R (2021). "MODIS/Terra Surface Reflectance Daily L2G Global 1km and 500m SIN Grid V061." doi:10.5067/MODIS/MOD09GA.061, https://www.earthdata.nasa.gov/data/catalog/lpcloud-mod09ga-061. Wan Z, Hook S, Hulley G (2021). "MODIS/Terra Land Surface Temperature/Emissivity Daily L3 Global 1km SIN Grid V061." doi:10.5067/MODIS/MOD11A1.061, https://www.earthdata.nasa.gov/data/catalog/lpcloud-mod11a1-061. Didan K (2021). "MODIS/Terra Vegetation Indices 16-Day L3 Global 1km SIN Grid V061." doi:10.5067/MODIS/MOD13A2.061, https://www.earthdata.nasa.gov/data/catalog/lpcloud-mod13a2-061. Román MO, Wang Z, Sun Q, Kalb V, Miller SD, Molthan A, Schultz L, Bell J, Stokes EC, Pandey B, Seto KC, Hall D, Oda T, Wolfe RE, Lin G, Golpayegani N, Devadiga S, Davidson C, Sarkar S, Praderas C, Schmaltz J, Boller R, Stevens J, Ramos González OM, Padilla E, Alonso J, Detrés Y, Armstrong R, Miranda I, Conte Y, Marrero N, MacManus K, Esch T, Masuoka EJ (2018). "NASA's Black Marble nighttime lights product suite." *Remote Sensing of Environment*, **210**, 113–143. ISSN 00344257, doi:10.1016/j.rse.2018.03.017, https://linkinghub.elsevier.com/retrieve/pii/S003442571830110X.

Examples

```
## Not run:
# RECOMMENDED: Set up token once (persists across sessions)
setup_nasa_token()

# Then download without specifying token
download_modis(
  product = "MOD09GA",
  version = "061",
  date = "2024-01-01",
  extent = c(-80, 35, -75, 40),
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

# ALTERNATIVE: Token from file
download_modis(
  product = "MOD09GA",
  version = "061",
  date = "2024-01-01",
  extent = c(-80, 35, -75, 40),
  nasa_earth_data_token = "~/.nasa_earthdata_token",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

# ALTERNATIVE: Set token for current session
Sys.setenv(NASA_EARTHDATA_TOKEN = "your_token_here")
download_modis(
  product = "MOD09GA",
  date = "2024-01-01",
  acknowledgement = TRUE
)

# Date range
download_modis(
```

```
  product = "MOD09GA",
  version = "061",
  date = c("2024-01-01", "2024-01-07"),
  extent = c(-80, 35, -75, 40),
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_narr` *Download meteorological data*

Description

The `download_narr` function accesses and downloads daily meteorological data from NOAA's North American Regional Reanalysis (NARR) model via the NOAA Physical Sciences Laboratory (PSL) NARR Dailies server (https://downloads.psl.noaa.gov/Datasets/NARR/Dailies/).

Usage

```
download_narr(
  variables = NULL,
  year = c(2018, 2022),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`variables`	character. Variable(s) name acronym. See the *Available NARR Variables* section below for the complete list of supported abbreviations.
`year`	integer(1 or 2). Year or start/end years for downloading data.
`directory_to_save`	character(1). Directory to save downloaded data files.
`acknowledgement`	logical(1). Must be TRUE to proceed with download.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). DEPRECATED, ignored.
`show_progress`	logical(1). Show download progress (default TRUE)

hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum download retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Available NARR Variables

The variables argument accepts one or more of the following abbreviations. Variables are grouped into three categories that determine the source URL path used for download.

Monolevel variables (single vertical level, surface / near-surface fields):

acpcp Convective precipitation

air.2m Air temperature at 2 m

air.sfc Air temperature at surface

albedo Surface albedo

apcp Total accumulated precipitation

bgrun Baseflow-groundwater runoff

bmixl.hl1 Blackadar mixing length scale at hybrid level 1

cape Convective available potential energy

ccond Canopy conductance

cdcon Convective cloud cover

cdlyr Non-convective cloud cover

cfrzr Categorical freezing rain

cicep Categorical ice pellets

cin Convective inhibition

cnwat Plant canopy surface water

crain Categorical rain

csnow Categorical snow

dlwrf Downward longwave radiation flux

dpt.2m Dew point temperature at 2 m

dswrf Downward shortwave radiation flux

evap Evaporation

gflux Ground heat flux

hcdc High cloud cover

hgt.tropo Geopotential height at tropopause

hlcy Storm relative helicity

hpbl Planetary boundary layer height

`lcdc` Low cloud cover

`lftx4` Best (4-layer) lifted index

`lhtfl` Latent heat net flux

`mcdc` Mid-cloud cover

`mconv.hl1` Horizontal moisture divergence at hybrid level 1

`mslet` Mean sea level pressure (ETA model reduction)

`mstav` Moisture availability

`pevap` Potential evaporation

`pottmp.hl1` Potential temperature at hybrid level 1

`pottmp.sfc` Potential temperature at surface

`prate` Precipitation rate

`pres.sfc` Surface pressure

`pres.tropo` Pressure at tropopause

`prmsl` Pressure reduced to mean sea level

`pr_wtr` Precipitable water

`rcq` Specific humidity tendency from all physics

`rcs` Snowfall water equivalent tendency

`rcsol` Solar radiative heating rates

`rct` Temperature tendency from all physics

`rhum.2m` Relative humidity at 2 m

`shtfl` Sensible heat net flux

`shum.2m` Specific humidity at 2 m

`snod` Snow depth

`snohf` Snow phase-change heat flux

`snom` Snow melt

`snowc` Snow cover

`soilm` Soil moisture content (0–200 cm layer)

`ssrun` Storm surface runoff

`tcdc` Total cloud cover

`tke.hl1` Turbulent kinetic energy at hybrid level 1

`ulwrf.ntat` Upward longwave radiation flux at nominal top of atmosphere

`ulwrf.sfc` Upward longwave radiation flux at surface

`ustm` U-component of storm motion

`uswrf.ntat` Upward shortwave radiation flux at nominal top of atmosphere

`uswrf.sfc` Upward shortwave radiation flux at surface

`uwnd.10m` U-component of wind at 10 m

`veg` Vegetation fraction

`vis` Visibility

`vstm` V-component of storm motion

`vvel.hl1` Vertical velocity at hybrid level 1

`vwnd.10m` V-component of wind at 10 m

`vwsh.tropo` Vertical wind shear at tropopause

`wcconv` Convective wetting of vegetation canopy

`wcinc` Wetting of vegetation canopy

`wcuflx` U-component of convective canopy moisture flux

`wcvflx` V-component of convective canopy moisture flux

`weasd` Water-equivalent accumulated snow depth

`wvconv` Convective column moisture convergence

`wvinc` Column moisture increase

`wvuflx` U-component of vertically-integrated moisture flux

`wvvflx` V-component of vertically-integrated moisture flux

Pressure level variables (29 atmospheric pressure levels from 1000 to 100 hPa; all levels are downloaded together):

`air` Air temperature

`hgt` Geopotential height

`omega` Vertical velocity (pressure / omega)

`shum` Specific humidity

`tke` Turbulent kinetic energy

`uwnd` U-component of wind

`vwnd` V-component of wind

Subsurface (soil) variables (4 soil layers):

`soill` Liquid volumetric soil moisture (non-frozen fraction)

`soilw` Volumetric soil moisture content

`tsoil` Soil temperature

Note

"Pressure levels" variables contain variable values at 29 atmospheric levels, ranging from 1000 hPa to 100 hPa. All pressure levels data will be downloaded for each variable.

The 88 variables supported by this function represent the complete set of variables available as individual NetCDF files on the PSL NARR Dailies server. The NARR archive also contains additional variables (e.g., cloud water mixing ratio, ice mixing ratio, surface friction velocity, momentum fluxes, and static land/soil properties) that are only present in the raw merged GRIB files (`merged_AWIP32.YYYYMMDDHH`) available at https://ftp.cpc.ncep.noaa.gov/NARR/. Those variables cannot be downloaded with this function.

Author(s)

Mitchell Manware, Insang Song, Kyle Messier

References

Mesinger F, DiMego G, Kalnay E, Mitchell K, Shafran PC, Ebisuzaki W, Jović D, Woollen J, Rogers E, Berbery EH, Ek MB, Fan Y, Grumbine R, Higgins W, Li H, Lin Y, Manikin G, Parrish D, Shi W (2006). "North American Regional Reanalysis." *Bulletin of the American Meteorological Society*, **87**(3), 343–360. ISSN 0003-0007, 1520-0477, doi:10.1175/BAMS873343.

Examples

```
## Not run:
download_narr(
  variables = c("weasd", "omega"),
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

# Multiple years
download_narr(
  variables = c("air.2m", "rhum.2m"),
  year = c(2020, 2022),
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_nei *Download road emissions data*

Description

The `download_nei()` function accesses and downloads road emissions data from the U.S Environmental Protection Agency's (EPA) National Emissions Inventory (NEI).

Usage

```
download_nei(
  epa_certificate_path = NULL,
  certificate_url = paste0("http://cacerts.digicert.com/",
    "DigiCertGlobalG2TLSRSASHA2562020CA1-1.crt"),
  year = c(2017L, 2020L),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
```

```
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

epa_certificate_path	TO BE DEPRECATED. Certificate path.
certificate_url	TO BE DEPRECATED. Certificate URL.
year	integer(1). Available years of NEI data.
directory_to_save	character(1). Directory to save data.
acknowledgement	logical(1). Must be TRUE to proceed.
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
unzip	logical(1). Unzip zip files (default TRUE).
remove_zip	logical(1). Remove zip files after unzipping (default FALSE).
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

NEI data does not require authentication.

Author(s)

Kyle Messier, Insang Song

References

United States Environmental Protection Agency (2024). "Air Emissions Inventories." https://www.epa.gov/air-emissions-inventories.

Examples

```
## Not run:
download_nei(
  year = c(2017L, 2020L),
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_nlcd` *Download National Land Cover Database (NLCD) data*

Description

Downloads NLCD data products from the Multi-Resolution Land Characteristics (MRLC) Consortium. NLCD provides nationwide land cover and land cover change information for the United States at a 30m resolution.

Usage

```
download_nlcd(
  product = "Land Cover",
  year = 2021,
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`product`	character(1). NLCD product type. One of: • "Land Cover" (default) • "Land Cover Change" • "Land Cover Confidence" • "Fractional Impervious Surface" • "Impervious Descriptor" • "Spectral Change Day of Year"
`year`	integer(1). Year of NLCD data (1985-2024). Default is 2021.

directory_to_save
: character(1). Directory to save downloaded files.

acknowledgement
: logical(1). Must be TRUE to proceed with download.

download
: logical(1). DEPRECATED. Downloads now happen automatically. Set to FALSE to skip downloading (generates file list only).

remove_command
: logical(1). Deprecated, ignored.

unzip
: logical(1). Unzip downloaded files? Default is TRUE.

remove_zip
: logical(1). Remove zip files after extraction? Default is FALSE.

show_progress
: logical(1). Show download progress? Default is TRUE.

hash
: logical(1). Return hash of downloaded files? Default is FALSE.

max_tries
: integer(1). Maximum download retry attempts. Default is 20.

rate_limit
: numeric(1). Minimum seconds between requests (default 2)

Value

invisible NULL; or hash character if hash=TRUE

Author(s)

Mitchell Manware, Insang Song, Kyle Messier

References

Dewitz J (2023). "National Land Cover Database (NLCD) 2021 Products." doi:10.5066/P9JZ7AO3.

Examples

```
## Not run:
# Download 2021 Land Cover
download_nlcd(
  product = "Land Cover",
  year = 2021,
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

# Download Land Cover Change for 2019
download_nlcd(
  product = "Land Cover Change",
  year = 2019,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  unzip = TRUE,
  remove_zip = TRUE
)

## End(Not run)
```

`download_population` *Download population density data*

Description

The `download_population()` function accesses and downloads population density data from NASA's UN WPP-Adjusted Population Density.

Usage

```
download_population(
  data_resolution = "60 minute",
  data_format = c("GeoTIFF", "ASCII", "netCDF"),
  year = "2020",
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2,
  nasa_earth_data_token = NULL
)
```

Arguments

`data_resolution`	character(1). Available resolutions.
`data_format`	character(1). "ASCII", "GeoTIFF", or "netCDF".
`year`	character(1). Available years or "all".
`directory_to_save`	character(1). Directory to save data.
`acknowledgement`	logical(1). Must be TRUE to proceed.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). Deprecated, ignored.
`unzip`	logical(1). Unzip zip files (default TRUE).
`remove_zip`	logical(1). Remove zip files after unzipping (default FALSE).
`show_progress`	logical(1). Show download progress (default TRUE)
`hash`	logical(1). Return hash of downloaded files (default FALSE)
`max_tries`	integer(1). Maximum retry attempts (default 20)

`rate_limit` numeric(1). Minimum seconds between requests (default 2)

`nasa_earth_data_token`
character(1). NASA EarthData bearer token. If NULL (default), reads from the `NASA_EARTHDATA_TOKEN` environment variable via `get_token()`.

Value

invisible list with download results; or hash character if hash=TRUE

Note

Population data may require NASA EarthData authentication depending on access method.

Author(s)

Mitchell Manware, Insang Song

References

Center For International Earth Science Information Network-CIESIN-Columbia University (2017). "Gridded Population of the World, Version 4 (GPWv4): Population Density, Revision 11." doi:10.7927/H49C6VHW, https://earthdata.nasa.gov/data/catalog/sedac-ciesin-sedac-gpwv4-popdens-r11-4.11.

Examples

```
## Not run:
# RECOMMENDED: Set up token once (persists across sessions)
setup_nasa_token()

download_population(
  data_resolution = "30 second",
  data_format = "GeoTIFF",
  year = "2020",
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

---

download_prism *Download PRISM data*

Description

Accesses and downloads Oregon State University's PRISM data from the PRISM Climate Group Web Service

Usage

```
download_prism(
  time,
 element = c("ppt", "tmin", "tmax", "tmean", "tdmean", "vpdmin", "vpdmax", "solslope",
    "soltotal", "solclear", "soltrans"),
  data_type = c("ts", "normals_800", "normals"),
  format = c("nc", "asc", "grib2"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE,
  show_progress = TRUE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`time` character(1). Length of 2, 4, 6, or 8. Time period for time series or normals. According to the PRISM Web Service Guide, acceptable formats include (disclaimer: the following is a direct quote; minimal formatting is applied): **Time Series**:

- `YYYYMMDD` for daily data (between yesterday and January 1st, 1981) – returns a single grid in a .zip file
- `YYYYMM` for monthly data (between last month and January 1981) – returns a single grid in a .zip file
- `YYYY` for annual data (between last year and 1981) - returns a single grid in a .zip file
- `YYYY` for historical data (between 1980 and 1895) - returns a single zip file containing 12 monthly grids for `YYYY` plus the annual.

Normals:

- Monthly normal: date is `MM` (i.e., 04 for April) or the value 14, which returns the annual normal
- Daily normal: date is `MMDD` (i.e., 0430 for April 30)

`element` character(1). Data element. One of `c("ppt", "tmin", "tmax", "tmean", "tdmean", "vpdmin", "vpdmax")` For normals, `c("solslope", "soltotal", "solclear", "soltrans")` are also accepted.

`data_type` character(1). Data type.

- `"ts"`: 4km resolution time series.
- `"normals_800"`: 800m resolution normals.
- `"normals"`: 4km resolution normals.

`format` character(1). Data format. Only applicable for `data_type = "ts"`.

directory_to_save
: character(1). Directory to download files.

acknowledgement
: logical(1). By setting TRUE the user acknowledges that the data downloaded using this function may be very large and use lots of machine storage and memory.

download
: logical(1). FALSE will generate a *.txt file containing all download commands. By setting TRUE the function will download all of the requested data files.

remove_command
: logical(1). Remove (TRUE) or keep (FALSE) the text file containing download commands.

unzip
: logical(1). Unzip the downloaded zip file to extract the data files (nc, grib2, etc.) into directory_to_save. Default is TRUE. The PRISM API always returns a zip regardless of the requested format.

remove_zip
: logical(1). Remove the zip file after unzipping. Default is FALSE. Only applies when unzip = TRUE.

hash
: logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.

show_progress
: logical(1). Show download progress. Default is TRUE.

max_tries
: integer(1). Maximum download retry attempts. Default is 20.

rate_limit
: numeric(1). Minimum seconds between requests. Default is 2.

Value

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- .bil (normals) or single grid files depending on the format choice will be stored in directory_to_save.

Author(s)

Insang Song

References

Daly C, Taylor GH, Gibson WP, Parzybok TW, Johnson GL, Pasteris PA (2000). "HIGH-QUALITY SPATIAL CLIMATE DATA SETS FOR THE UNITED STATES AND BEYOND." *Transactions of the ASAE*, **43**(6), 1957–1962. ISSN 2151-0059, doi:10.13031/2013.3101, http://elibrary.asabe.org/abstract.asp??JID=3&AID=3101&CID=t2000&v=43&i=6&T=1.

- PRISM Climate Group
- PRISM Web Service Guide

Examples

```
## Not run:
download_prism(
  time = "202104",
  element = "ppt",
  data_type = "ts",
```

```
  format = "nc",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE
)

## End(Not run)
```

`download_terraclimate` *Download TerraClimate data*

Description

The `download_terraclimate` function accesses and downloads climate and water balance data from the University of California Merced Climatology Lab's TerraClimate dataset.

Usage

```
download_terraclimate(
  variables = NULL,
  year = c(2018, 2022),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

`variables`	character. Variable(s) name(s).
`year`	integer(1 or 2). Year or start/end years for downloading data.
`directory_to_save`	character(1). Directory to save data.
`acknowledgement`	logical(1). Must be TRUE to proceed.
`download`	logical(1). DEPRECATED. Downloads happen automatically.
`remove_command`	logical(1). Deprecated, ignored.
`show_progress`	logical(1). Show download progress (default TRUE)
`hash`	logical(1). Return hash of downloaded files (default FALSE)
`max_tries`	integer(1). Maximum retry attempts (default 20)
`rate_limit`	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

TerraClimate data does not require authentication.

Author(s)

Mitchell Manware, Insang Song

References

Abatzoglou JT, Dobrowski SZ, Parks SA, Hegewisch KC (2018). "TerraClimate, a high-resolution global dataset of monthly climate and climatic water balance from 1958–2015." *Scientific data*, **5**(1), 1–12.

Examples

```
## Not run:
download_terraclimate(
  variables = "ppt",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE
)

## End(Not run)
```

`download_tri` *Download toxic release data*

Description

The `download_tri()` function accesses and downloads toxic release data from the U.S. Environmental Protection Agency's (EPA) Toxic Release Inventory (TRI) Program. The EPA TRI basic data files contain annual, facility-reported toxic chemical release and waste management information. EPA publishes TRI basic files in multiple annual variants under the same service endpoint: a nationwide file (`"US"`), state-specific files identified by two-letter postal abbreviations (for example `"AZ"` or `"NC"`), and a tribal file (`"tbl"`).

Usage

```
download_tri(
  year = c(2018L, 2022L),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  jurisdiction = "US",
```

```
  download = TRUE,
  remove_command = FALSE,
  show_progress = TRUE,
  hash = FALSE,
  max_tries = 20,
  rate_limit = 2
)
```

Arguments

year	integer(1 or 2). Year or start/end years for downloading data.
directory_to_save	character(1). Directory to download files.
acknowledgement	logical(1). Must be TRUE to proceed.
jurisdiction	character(1). TRI file variant to download. Use "US" for the nationwide file, a two-letter state or territory code such as "AZ" or "NC" for a jurisdiction-specific file, or "tbl" for the tribal file. Default is "US".
download	logical(1). DEPRECATED. Downloads happen automatically.
remove_command	logical(1). Deprecated, ignored.
show_progress	logical(1). Show download progress (default TRUE)
hash	logical(1). Return hash of downloaded files (default FALSE)
max_tries	integer(1). Maximum retry attempts (default 20)
rate_limit	numeric(1). Minimum seconds between requests (default 2)

Value

invisible list with download results; or hash character if hash=TRUE

Note

TRI data does not require authentication. State and tribal downloads are saved with jurisdiction-specific file names, while the U.S.-wide download keeps the historical tri_raw_<year>.csv naming pattern.

Author(s)

Mariana Kassien, Insang Song

References

United States Environmental Protection Agency (2024). "TRI Basic Data Files: Calendar Years 1987 – Present." https://www.epa.gov/toxics-release-inventory-tri-program/tri-data-action-0.

Examples

```
## Not run:
download_tri(
  year = 2021L,
  directory_to_save = tempdir(),
  jurisdiction = "NC",
  acknowledgement = TRUE
)

## End(Not run)
```

`dt_as_mysftime` *Convert a* `data.table` *to an* `sftime`

Description

Convert a `data.table` object to an `sftime`. `x` must be a `data.table` object with "lon", "lat", and "time" columns to describe the longitude, latitude, and time-orientation, respectively, of `x`.

Usage

```
dt_as_mysftime(x, lonname, latname, timename, crs)
```

Arguments

`x`	a `data.table`
`lonname`	character for longitude column name
`latname`	character for latitude column name
`timename`	character for time column name
`crs`	coordinate reference system

Value

an `sftime` object

Author(s)

Eva Marques

`get_geos_info` *Get GEOS variable lookup information*

Description

Returns a lookup table of available GEOS collection and variable selectors from locally downloaded GEOS-CF netCDF files. This helper inspects layer metadata only and does not read raster values into memory.

Usage

```
get_geos_info(path = NULL, include_file = FALSE, ...)
```

Arguments

`path`	character(1+) Path(s) to GEOS file(s) and/or directory(ies) containing GEOS-CF `.nc4` files.
`include_file`	logical(1). If `TRUE`, include a `file` column showing the source file for each collection-variable row. Default `FALSE`.
`...`	Placeholders.

Value

a `data.frame` with GEOS collection and variable selectors.

Author(s)

Kyle Messier

Examples

```
## Not run:
get_geos_info(path = "./data/geos")
get_geos_info(path = "./data/geos", include_file = TRUE)

## End(Not run)
```

`get_merra2_info` *Get MERRA2 variable lookup information*

Description

Returns a lookup table of available MERRA2 collection and variable selectors from locally downloaded MERRA2 netCDF files. This helper inspects layer metadata only and does not read raster values into memory.

Usage

```
get_merra2_info(path = NULL, include_file = FALSE, ...)
```

Arguments

`path`	character(1+) Path(s) to MERRA2 file(s) and/or directory(ies) containing MERRA2 `.nc4` files (and optional FWI `.nc` files).
`include_file`	logical(1). If `TRUE`, include a `file` column showing the source file for each collection-variable row. Default `FALSE`.
`...`	Placeholders.

Value

a `data.frame` with MERRA2 collection and variable selectors.

Author(s)

Kyle Messier

Examples

```
## Not run:
get_merra2_info(path = "./data/merra2")
get_merra2_info(path = "./data/merra2", include_file = TRUE)

## End(Not run)
```

`get_modis_info` *Get MODIS product subdataset lookup information*

Description

Returns a lookup table of available MODIS product and subdataset selectors from locally downloaded MODIS/VIIRS-style HDF/H5 files. This helper uses metadata inspection (`terra::describe(..., sds = TRUE)` and layer names) and does not read raster values into memory.

Usage

```
get_modis_info(path = NULL, include_file = FALSE, ...)
```

Arguments

`path`	character(1+) Path(s) to MODIS file(s) and/or directory(ies) containing `.hdf/.h5` files.
`include_file`	logical(1). If `TRUE`, include a `file` column showing the source file for each product-subdataset row. Default `FALSE`.
`...`	Placeholders.

Value

a `data.frame` with MODIS product and subdataset selectors.

Author(s)

Kyle Messier

Examples

```
## Not run:
get_modis_info(path = "./data/modis")
get_modis_info(path = "./data/modis", include_file = TRUE)

## End(Not run)
```

get_tri_info *Get TRI lookup information for chemicals or industries*

Description

Returns a lookup table from local TRI files. By default it returns chemical information (TRI_CHEMICAL_COMPOUND_ID, CHEMICAL, CASN). Set type = "industries" to return industry sector information (INDUSTRY_SECTOR_CODE, INDUSTRY_SECTOR).

Usage

```
get_tri_info(
  path = NULL,
  type = c("chemicals", "industries"),
  year = NULL,
  include_na = FALSE,
  ...
)
```

Arguments

path	character(1). Path to the directory with TRI CSV files (from download_tri).
type	character(1). Lookup table to return. One of "chemicals" (default) or "industries".
year	NULL or integer(1). Optional single year filter. If NULL (default), all years in path are included.
include_na	logical(1). If FALSE (default), rows where lookup fields are all missing are removed.
...	Placeholders.

Value

a data.frame containing the requested TRI lookup table.

Author(s)

Kyle Messier

Examples

```
## Not run:
get_tri_info(path = "./data")
get_tri_info(path = "./data", type = "industries")
get_tri_info(path = "./data", year = 2020)

## End(Not run)
```

process_aqs *Process U.S. EPA AQS daily CSV data*

Description

The process_aqs() function cleans and imports raw air quality monitoring sites from pre-generated daily CSV files, returning a single SpatVector or sf object. date is used to filter the raw data read from csv files. Filtered rows are then processed according to mode argument. Some sites report multiple measurements per day with and without exceptional events the internal procedure of this function keeps "Included" if there are multiple event types per site-time.

Usage

```
process_aqs(
  path = NULL,
  date = c("2018-01-01", "2022-12-31"),
  mode = c("date-location", "available-data", "location"),
  data_field = "Arithmetic.Mean",
  return_format = c("terra", "sf", "data.table"),
  extent = NULL,
  ...
)
```

Arguments

path	character(1). Directory path to daily measurement data.
date	character(1 or 2). Date (1) or start and end dates (2). Should be in "YYYY-MM-DD" format and sorted.
mode	character(1). One of • "date-location" (all dates * all locations) • "available-data" (date-location pairs with available data) • "location" (unique locations).
data_field	character(1). Data field to extract.
return_format	character(1). "terra" or "sf" or "data.table".
extent	numeric(4). Spatial extent of the resulting object. The order should be c(xmin, xmax, ymin, ymax). The coordinate system should be WGS84 (EPSG:4326).
...	Placeholders.

Value

a SpatVector, sf, or data.table object depending on the return_format

Note

Choose `date` and `mode` values with caution. The function may return a massive data.table depending on the time range, resulting in a long processing time or even a crash if data is too large for your computing environment to process. AQS data are generally intended for use as dependent variables, so `process_aqs()` does not have a companion route in `calculate_covariates()`.

See Also

- `download_aqs()`
- EPA, n.d., *AQS Parameter Codes*

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
aqs <- process_aqs(
  path = "./data/aqs_daily_example.csv",
  date = c("2022-12-01", "2023-01-31"),
  mode = "date-location",
  return_format = "terra"
)

## End(Not run)
```

---

process_blackmarble *Assign VIIRS Black Marble products corner coordinates to retrieve a merged raster*

Description

This function will return a `SpatRaster` object with georeferenced h5 files of Black Marble product. Referencing corner coordinates are necessary as the original h5 data do not include such information.

Usage

```
process_blackmarble(
  path = NULL,
  date = NULL,
  tile_df = process_blackmarble_corners(),
  subdataset = 3L,
  crs = "EPSG:4326",
  ...
)
```

Arguments

path	character. Full paths of h5 files.
date	character(1). Date to query.
tile_df	data.frame. Contains four corner coordinates in fields named c("xmin", "xmax", "ymin", "ymax"). See process_blackmarble_corners to generate a valid object for this argument.
subdataset	integer(1). Subdataset number to process. Default is 3L.
crs	character(1). terra::crs compatible CRS. Default is "EPSG:4326"
...	For internal use.

Value

a SpatRaster object

Author(s)

Insang Song

References

- Wang, Z. (2022). Black Marble User Guide (Version 1.3). NASA.

See Also

- terra::describe
- terra::merge
- process_blackmarble_corners

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
vnp46a2 <- process_blackmarble(
  path =
    list.files("./data", pattern = "VNP46A2.", full.names = TRUE),
  date = "2024-01-01",
  tile_df =
    process_blackmarble_corners(hrange = c(8, 10), vrange = c(4, 5)),
  subdataset = 3L,
  crs = "EPSG:4326"
)

## End(Not run)
```

process_covariates *Process raw data wrapper function*

Description

This function processes raw data files which have been downloaded by download_data. process_covariates and the underlying source-specific processing functions have been designed to operate on the raw data files. To avoid errors, **do not edit the raw data files before passing to** process_covariates.

Usage

```
process_covariates(
  covariate = c("modis_swath", "modis_merge", "mcd14ml", "koppen-geiger", "blackmarble",
    "koeppen-geiger", "koppen", "koeppen", "geos", "goes", "goes_adp", "GOES", "dummies",
    "gmted", "aqs", "hms", "smoke", "sedac_population", "population", "sedac_groads",
    "groads", "roads", "nlcd", "tri", "narr", "nei", "ecoregions", "ecoregion", "merra",
    "merra2", "gridmet", "terraclimate", "huc", "cropscape", "cdl", "prism", "edgar",
     "improve", "IMPROVE", "drought", "spei", "eddi", "usdm"),
  path = NULL,
  ...
)
```

Arguments

covariate	character(1). Covariate type.
path	character(1). Directory or file path to raw data depending on covariate value.
...	Arguments passed to each raw data processing function.

Value

SpatVector, SpatRaster, sf, data.table, or character depending on covariate type and selections.

Author(s)

Insang Song

See Also

- process_modis_swath: "modis_swath"
- process_modis_merge: "modis_merge"
- process_blackmarble: "blackmarble"
- process_koppen_geiger: "koppen-geiger", "koeppen-geiger", "koppen"
- process_ecoregion: "ecoregion", "ecoregions"
- process_nlcd: "nlcd", "NLCD"

- process_tri: "tri", "TRI"
- process_nei: "nei", "NEI"
- process_geos: "geos", "GEOS"
- process_goes: "goes", "goes_adp", "GOES"
- process_gmted: "gmted", "GMTED"
- process_aqs: "aqs", "AQS"
- process_edgar: "edgar"
- process_improve: "improve", "IMPROVE"
- process_hms: "hms", "smoke", "HMS"
- process_narr: "narr", "NARR"
- process_groads: "sedac_groads", "roads", "groads"
- process_population: "sedac_population", "population"
- process_merra2: "merra", "merra2", "MERRA2"
- process_gridmet: "gridmet", "gridMET"
- process_terraclimate: "terraclimate", "TerraClimate"
- process_huc: "huc", "HUC"
- process_cropscape: "cropscape", "cdl"
- process_prism: "prism", "PRISM"
- process_drought: "drought", "spei", "eddi", "usdm"

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
process_covariates(
  covariate = "narr",
  date = c("2018-01-01", "2018-01-10"),
  variable = "weasd",
  path = system.file("extdata", "examples", "narr", "weasd")
)

## End(Not run)
```

`process_cropscape` *Process CropScape data*

Description

This function imports and cleans raw CropScape data, returning a single `SpatRaster` object.

Reads CropScape file of selected `year`.

Usage

```
process_cropscape(path = NULL, year = 2021, extent = NULL, ...)
```

Arguments

path	character giving CropScape data path
year	numeric giving the year of CropScape data used
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatRaster object

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
cropscape <- process_cropscape(
  path = "./data/cropscape_example.tif",
  year = 2020
)

## End(Not run)
```

---

process_drought *Process drought index data*

Description

The process_drought() function imports and cleans raw drought index files returned by download_drought(), producing a harmonized output object ready for calculate_drought():

- **SPEI / EDDI** — returns a SpatRaster with one layer per time step, layer names in "<source>_<timescale>_YYYY-MM
format, CRS set to EPSG:4326.
- **USDM** — returns a SpatVector (polygon) with columns DM (drought-monitor class, integer 0–4), date (Date), and source ("usdm"), CRS EPSG:4326.

Usage

```
process_drought(
  source = c("spei", "eddi", "usdm"),
  path = NULL,
  date = c("2020-01-01", "2020-12-31"),
  timescale = 1L,
  extent = NULL,
  ...
)
```

Arguments

source	character(1). Drought data source. One of "spei", "eddi", or "usdm". When called through process_covariates(covariate = "spei") the alias is forwarded automatically.
path	character(1). Directory containing downloaded drought files (output of download_drought()).
date	character(1 or 2). Single date or start/end dates. Format "YYYY-MM-DD".
timescale	integer(1). Accumulation timescale in months (SPEI/EDDI only; ignored for USDM). Must match the timescale used in download_drought(). Default 1L.
extent	numeric(4) or SpatExtent. Optional spatial crop applied before returning. NULL (default) returns full extent.
...	Reserved for future use; currently ignored.

Value

- SpatRaster for SPEI or EDDI sources.
- SpatVector (polygons) for USDM source.

Note

- SPEI/EDDI files are expected to follow the naming convention produced by download_drought(): spei<timescale>.nc and either legacy eddi<timescale>mn<year>.nc or current EDDI_ETrs_<timescale>mn_<YYY
- USDM files are expected to be weekly shapefiles named USDM_<YYYYMMDD>.shp.
- Layer/column naming is standardised so that calculate_drought() can operate identically regardless of source.

Author(s)

Insang Song

See Also

download_drought, calculate_drought

Examples

```
## Not run:
## SPEI
spei <- process_drought(
  source = "spei",
  path = "./data/drought",
  date = c("2020-01-01", "2020-12-31"),
  timescale = 1L
)
## USDM
usdm <- process_drought(
  source = "usdm",
  path = "./data/drought",
  date = c("2020-01-07", "2020-03-31")
)

## End(Not run)
```

`process_ecoregion` *Process ecoregion data*

Description

The `process_ecoregion` function imports and cleans raw ecoregion data, returning a `SpatVector` object.

Usage

```
process_ecoregion(path = NULL, extent = NULL, ...)
```

Arguments

`path`	character(1). Path to Ecoregion Shapefiles
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`...`	Placeholders.

Value

a `SpatVector` object

Note

The function will fix Tukey's bridge in Portland, ME. This fix will ensure that the EPA air quality monitoring sites will be located within the ecoregion.

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
ecoregion <- process_ecoregion(
  path = "./data/epa_ecoregion.gpkg"
)

## End(Not run)
```

`process_edgar` *Process EDGAR emissions data*

Description

The `process_edgar()` function imports extracted EDGAR gridded emissions files and returns a single `SpatRaster` object. Raster formats supported by `terra::rast()` such as NetCDF (`.nc`, `.nc4`) and GeoTIFF (`.tif`, `.tiff`) are supported.

Usage

```
process_edgar(path = NULL, extent = NULL, ...)
```

Arguments

`path`	character. Directory containing extracted EDGAR raster files or one or more file paths.
`extent`	numeric(4) or SpatExtent giving the extent of the raster; if `NULL` (default), the entire raster is loaded.
`...`	Placeholders.

Value

a `SpatRaster` object

Note

`process_edgar()` currently supports gridded raster outputs from `download_edgar()` such as the default `format = "nc"`. Plain-text EDGAR downloads should be re-downloaded as raster outputs before processing.

Author(s)

Mariana Alifa Kassien, Insang Song

See Also

`download_edgar()`, `calculate_edgar()`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires data that is
##       not included in the package.
## Not run:
edgar <- process_edgar(
  path = "./data/edgar",
  extent = c(-85, -75, 33, 37)
)

## End(Not run)
```

`process_geos` *Process atmospheric composition data*

Description

The `process_geos()` function imports and cleans raw atmospheric composition data, returning a single `SpatRaster` object.

Usage

```
process_geos(
  date = c("2018-01-01", "2018-01-10"),
  variable = NULL,
  path = NULL,
  extent = NULL,
  daily_agg = FALSE,
  fun = "mean",
  ...
)
```

Arguments

`date`	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
`variable`	character(1). GEOS-CF variable name(s). See *Notes* for collection-specific variable-name guidance.
`path`	character(1). Directory with downloaded netCDF (.nc4) files.
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`daily_agg`	logical(1). If `TRUE`, aggregate sub-daily layers to daily values using `fun`. Default `FALSE` preserves the original hourly output. Aggregation groups layers by variable/level and date so that pressure-level structure is preserved. Not meaningful for collections that are already daily.
`fun`	character(1). Aggregation function passed to `terra::tapp()` (e.g. `"mean"`, `"max"`, `"min"`, `"sum"`). Ignored when `daily_agg = FALSE`.
`...`	Placeholders.

Details

GEOS-CF netCDF collections currently supported by `download_geos()` are: `"aqc_tavg_1hr_g1440x721_v1"`, `"chm_tavg_1hr_g1440x721_v1"`, `"met_tavg_1hr_g1440x721_x1"`, `"xgc_tavg_1hr_g1440x721_x1"`, `"chm_inst_1hr_g1440x721_p23"`, and `"met_inst_1hr_g1440x721_p23"`.

Value

a `SpatRaster` object;

Note

Layer names of the returned `SpatRaster` object contain the variable, pressure level, date, and hour when `daily_agg = FALSE` (default). When `daily_agg = TRUE`, layer names contain the variable, pressure level, and date only, and `terra::time()` is set to midnight UTC of each date.

Collection-specific variable names accepted by `variable`:

Collection	Variables
aqc_tavg_1hr_g1440x721_v1	no2, co, so2, pm25_rh35_gcc, o3
chm_tavg_1hr_g1440x721_v1	ocpi, bcpo, pm25soa_rh35_gc, dst4, prpe, macr, pm25ss_rh35_gcc, hno4, ch4, nh3, h2c
met_tavg_1hr_g1440x721_x1	zl, zpbl, ps, v2m, v, q2m, u, t2m, troppb, q, t, v10m, t10m, u2m, q10m, ts, slp, cldtt, phi
xgc_tavg_1hr_g1440x721_x1	wetdepflx_nh4, aod550_dst6, wetdepflx_dst1, tropcol_io, totcol_o3, tropcol_hch
chm_inst_1hr_g1440x721_p23	pm25soa_rh35_gc, pm25ss_rh35_gcc, so2, co, o3, pm25oc_rh35_gcc, pm25du_rh35_gcc
met_inst_1hr_g1440x721_p23	omega, t, eth, q, epv, rh, slp, airdens, ps, h, th, v, u, airvol_chem

`variable` matching is case-insensitive (for example, `"o3"` matches `"O3"`).

Reference: NASA GEOS-CF OpenDAP catalog https://opendap.nccs.nasa.gov/dods/gmao/geos-cf/assim.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
geos <- process_geos(
  date = c("2024-01-01", "2024-01-10"),
  variable = "O3",
  path = "./data/aqc_tavg_1hr_g1440x721_v1"
)
## daily mean across all sub-daily layers per variable/level
geos_daily <- process_geos(
  date = c("2024-01-01", "2024-01-10"),
  variable = "O3",
  path = "./data/aqc_tavg_1hr_g1440x721_v1",
  daily_agg = TRUE,
```

```
  fun = "mean"
)

## End(Not run)
```

`process_gmted` *Process elevation data*

Description

The `process_gmted()` function imports and cleans raw elevation data, returning a single `SpatRaster` object.

Usage

```
process_gmted(variable = NULL, path = NULL, extent = NULL, ...)
```

Arguments

`variable`	vector(1). Vector containing the GMTED statistic first and the resolution second. (Example: variable = c("Breakline Emphasis", "7.5 arc-seconds")). • Statistic options: "Breakline Emphasis", "Systematic Subsample", "Median Statistic", "Minimum Statistic", "Mean Statistic", "Maximum Statistic", "Standard Deviation Statistic" • Resolution options: "30 arc-seconds", "15 arc-seconds", "7.5 arc-seconds"
`path`	character(1). Directory with downloaded GMTED "*_grd" folder containing .adf files.
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`...`	Placeholders.

Value

a `SpatRaster` object

Note

`SpatRaster` layer name indicates selected variable and resolution, and year of release (2010).

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
gmted <- process_gmted(
  variable = c("Breakline Emphasis", "7.5 arc-seconds"),
  path = "./data/be75_grd"
)

## End(Not run)
```

`process_goes` *Process NOAA GOES ADP data*

Description

The `process_goes()` function imports and cleans NOAA GOES-16/18 Aerosol Detection Product (ADP) NetCDF files downloaded by `download_goes()`, returning a single `SpatRaster` object with CRS `EPSG:4326`.

Usage

```
process_goes(
  date = c("2024-01-01", "2024-01-01"),
  variable = NULL,
  path = NULL,
  extent = NULL,
  daily_agg = FALSE,
  fun = "mean",
  ...
)
```

Arguments

`date`	character(1 or 2). Date (YYYY-MM-DD) or start and end dates.
`variable`	character(1). Variable name to extract: `"Smoke"` or `"Dust"`.
`path`	character(1+). Directory with downloaded GOES ADP NetCDF files or a vector of full NetCDF file paths.
`extent`	numeric(4) or SpatExtent. Crop extent (`xmin, xmax, ymin, ymax` in EPSG:4326). Default `NULL` loads the full raster.
`daily_agg`	logical(1). If `TRUE`, aggregate sub-daily layers to daily values using `fun`. Default `FALSE` preserves original sub-daily layers.
`fun`	character(1). Aggregation function passed to `terra::tapp()` (e.g. `"mean"` or `"sum"`). Ignored when `daily_agg = FALSE`.
`...`	Placeholders.

Value

a SpatRaster object

Note

- Layer names follow the convention {variable}_{YYYYMMDD}_{HHMMSS} when daily_agg = FALSE, e.g. "Smoke_20240101_000000". With daily_agg = TRUE, layer names contain {variable}_{YYYYMMDD} and terra::time() is set to midnight UTC.
- terra::time() is set to POSIXct UTC for each layer.
- Files with GOES geostationary projection are reprojected to EPSG:4326.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires downloaded
##       data files.
## Not run:
goes <- process_goes(
  date = c("2024-01-01", "2024-01-01"),
  variable = "Smoke",
  path = "./data/goes/"
)
goes_daily <- process_goes(
  date = c("2024-01-01", "2024-01-01"),
  variable = "Smoke",
  path = "./data/goes/",
  daily_agg = TRUE,
  fun = "mean"
)

## End(Not run)
```

---

process_gridmet *Process gridMET data*

Description

The process_gridmet() function imports and cleans raw gridded surface meteorological data, returning a single SpatRaster object.

Usage

```
process_gridmet(
  date = c("2023-09-01", "2023-09-10"),
  variable = NULL,
  path = NULL,
  extent = NULL,
  ...
)
```

Arguments

date	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
variable	character(1). Variable name or acronym code. See gridMET Generate Wget File for variable names and acronym codes. (Note: variable "Burning Index" has code "bi" and variable "Energy Release Component" has code "erc").
path	character(1). Directory with downloaded netCDF (.nc) files.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatRaster object

Note

Layer names of the returned SpatRaster object contain the variable acronym, and date.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
gridmet <- process_gridmet(
  date = c("2023-01-01", "2023-01-10"),
  variable = "Precipitation",
  path = "./data/pr"
)

## End(Not run)
```

process_groads *Process roads data*

Description

The process_groads() function imports and cleans raw road data, returning a single SpatVector object.

Usage

```
process_groads(path = NULL, extent = NULL, ...)
```

Arguments

path	character(1). Path to geodatabase or shapefiles.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatVector object

Note

U.S. context. The returned SpatVector object contains a $description column to represent the temporal range covered by the dataset. For more information, see https://data.nasa.gov/dataset/global-roads-open-access-data-set-version-1-groadsv1.

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
groads <- process_groads(
  path = "./data/groads_example.shp"
)

## End(Not run)
```

process_hms *Process wildfire smoke data*

Description

The process_hms() function imports and cleans raw wildfire smoke plume coverage data, returning a single SpatVector object.

Usage

```
process_hms(date = "2018-01-01", path = NULL, extent = NULL, ...)
```

Arguments

date	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
path	character(1). Directory with downloaded NOAA HMS data files.
extent	numeric(4) or SpatExtent giving the extent of the output if NULL (default), the entire data is returned
...	Placeholders.

Value

a SpatVector or character object

Note

process_hms() will return a character object if there are no wildfire smoke plumes present for the selected dates and density. The returned character will contain the density value and the sequence of dates for which no wildfire smoke plumes were detected (see "Examples"). If multiple density polygons overlap, the function will return the highest density value.

Author(s)

Mitchell Manware

Examples

```
hms <- process_hms(
  date = c("2018-12-30", "2019-01-01"),
  path = "../tests/testdata/hms/"
)
```

process_huc *Retrieve Hydrologic Unit Code (HUC) data*

Description

Retrieve Hydrologic Unit Code (HUC) data

Usage

```
process_huc(
  path,
  layer_name = NULL,
  huc_level = NULL,
  huc_header = NULL,
  extent = NULL,
  ...
)
```

Arguments

path	character. Path to the file or the directory containing HUC data.
layer_name	character(1). Layer name in the path
huc_level	character(1). Field name of HUC level
huc_header	character(1). The upper level HUC code header to extract lower level HUCs.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Arguments passed to nhdplusTools::get_huc()

Value

a SpatVector object

Author(s)

Insang Song

See Also

nhdplusTools::get_huc

Examples

```
## NOTE: Examples are wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
library(terra)
getf <- "WBD_National_GDB.gdb"
# check the layer name to read
terra::vector_layers(getf)
test1 <- process_huc(
  getf,
  layer_name = "WBDHU8",
  huc_level = "huc8"
)
test2 <- process_huc(
  getf,
  layer_name = "WBDHU8",
  huc_level = "huc8"
)
test3 <- process_huc(
  "",
  layer_name = NULL,
  huc_level = NULL,
  huc_header = NULL,
  id = "030202",
  type = "huc06"
)

## End(Not run)
```

`process_improve` *Process IMPROVE aerosol monitoring data*

Description

The `process_improve()` function reads pipe-delimited IMPROVE (Interagency Monitoring of Protected Visual Environments) measurement files downloaded by `download_improve()` and joins them with a site metadata table to attach geographic coordinates and auxiliary site attributes. Returns a `SpatVector`, `sf`, or `data.table` object.

Usage

```
process_improve(
  path = NULL,
  product = c("raw", "rhr2", "rhr3"),
  date = NULL,
  sites_file = NULL,
  return_format = c("terra", "sf", "data.table"),
  extent = NULL,
  ...
)
```

Arguments

path	character(1). Directory containing downloaded IMPROVE .txt files.
product	character(1). Product type: "raw" (default), "rhr2", or "rhr3".
date	character(1 or 2). Date ("YYYY-MM-DD") or start/end date pair to filter measurements. Defaults to no filtering when NULL.
sites_file	character(1) or NULL. Path to a site metadata file. When NULL (default), the function first looks for a file named improve_sites.txt inside path, then falls back to an embedded IMPROVE aerosol site table included in amadeus.
return_format	character(1). Return object type: "terra", "sf", or "data.table".
extent	numeric(4) or NULL. Optional crop extent c(xmin, xmax, ymin, ymax) in WGS84 / EPSG:4326. Applied only when return_format is "terra" or "sf".
...	Placeholders.

Details

Three product types are supported via product:

"raw" IMPAER speciated aerosol mass concentrations. Key columns: SiteCode, FactDate, ParamCode, FactValue, Units.

"rhr2" IMPRHR2 Regional Haze Rule II light extinction (bext, Mm^{-1}).

"rhr3" IMPRHR3 Regional Haze Rule III deciview index (dv).

Measurement values are **not** filtered by Status; callers may apply their own validity flags (e.g., keep only Status == "V0").

Value

a SpatVector, sf, or data.table object depending on return_format.

Note

IMPROVE data are measured on an every-third-day sampling schedule. Gaps between measurement dates are expected.

See Also

download_improve

Examples

```
improve <- process_improve(
  path = system.file("testdata/improve", package = "amadeus"),
  product = "raw",
  date = c("2022-01-01", "2022-01-31"),
  return_format = "data.table"
)
```

process_koppen_geiger *Process climate classification data*

Description

The process_koppen_geiger() function imports and cleans raw climate classification data, returning a single SpatRaster object.

Usage

```
process_koppen_geiger(path = NULL, extent = NULL, ...)
```

Arguments

path	character(1). Path to Koppen-Geiger climate zone raster file
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatRaster object

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
kg <- process_koppen_geiger(
  path = "./data/koppen_geiger_data.tif"
)

## End(Not run)
```

process_merra2 *Process meteorological and atmospheric data*

Description

The process_merra2() function imports and cleans raw atmospheric, meteorological, and MERRA2-based Fire Weather Index data, returning a single SpatRaster object.

Usage

```
process_merra2(
  date = c("2018-01-01", "2018-01-10"),
  variable = NULL,
  path = NULL,
  extent = NULL,
  daily_agg = FALSE,
  fun = "mean",
  ...
)
```

Arguments

date	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
variable	character(1). MERRA2 variable name(s). For daily corrected Fire Weather Index files (collection = "fwi" during download), use one of "DC", "DMC", "FFMC", "ISI", "BUI", or "FWI" (or the full raw layer name).
path	character(1). Directory with downloaded netCDF (.nc4 or .nc) files.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
daily_agg	logical(1). If TRUE, aggregate sub-daily layers to daily values using fun. Default FALSE preserves the original sub-daily output. Aggregation groups layers by variable/level and date. Silently ignored for FWI collections, which are already daily.
fun	character(1). Aggregation function passed to terra::tapp() (e.g. "mean", "max", "min", "sum"). Ignored when daily_agg = FALSE.
...	Placeholders.

Value

a SpatRaster object;

Note

Layer names of the returned `SpatRaster` object contain the variable, pressure level, date, and hour for standard MERRA-2 collections when `daily_agg = FALSE` (default). When `daily_agg = TRUE`, layer names contain the variable, pressure level, and date only, and `terra::time()` is set to midnight UTC of each date. For daily Fire Weather Index files, layer names contain the variable and date only regardless of `daily_agg`. Pressure level values utilized for layer names are taken directly from raw data and are not edited to retain pressure level information.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
merra2 <- process_merra2(
  date = c("2024-01-01", "2024-01-10"),
  variable = "CPT",
  path = "./data/inst1_2d_int_Nx"
)
## daily mean CPT
merra2_daily <- process_merra2(
  date = c("2024-01-01", "2024-01-10"),
  variable = "CPT",
  path = "./data/inst1_2d_int_Nx",
  daily_agg = TRUE,
  fun = "mean"
)

## End(Not run)
```

---

process_modis_daily *Process MODIS files as daily outputs*

Description

Process MODIS HDF/H5 files into day-specific rasters over a requested date range. This helper preserves daily slices instead of flattening a multi-day range into one merged result.

Usage

```
process_modis_daily(
  path = NULL,
  date = NULL,
  subdataset = NULL,
  fun_agg = "mean",
```

```
  path_secondary = NULL,
  fusion_method = c("mean", "primary_first", "secondary_first"),
  return_type = c("stack", "list"),
  ...
)
```

Arguments

path	character. Full list of HDF/H5 file paths.
date	character(1:2). Date or date range in "YYYY-MM-DD" format.
subdataset	character(1). Subdataset names to extract. Should conform to regular expression. See base::regex for details.
fun_agg	Function name or custom function to aggregate overlapping cell values. See fun description in terra::tapp for details.
path_secondary	character. Optional secondary list of HDF/H5 paths (for example, Aqua files) to fuse with path by date.
fusion_method	character(1). Fusion method when path_secondary is provided: "mean", "primary_first", or "secondary_first".
return_type	character(1). Return "stack" for a multi-layer SpatRaster (default) or "list" for a named list of daily SpatRaster objects.
...	Additional arguments passed to process_modis_merge.

Value

A day-preserving MODIS result as a SpatRaster (return_type = "stack") or named list (return_type = "list").

Author(s)

Insang Song

See Also

process_modis_merge, download_data

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
mod09ga_daily <- process_modis_daily(
  path = list.files("./data", pattern = "MOD09GA.", full.names = TRUE),
  date = c("2024-01-01", "2024-01-07"),
  subdataset = "sur_refl_b01_1",
  return_type = "list"
)

## End(Not run)
```

process_modis_merge *Process MODIS .hdf files*

Description

Get mosaic or merged raster from multiple MODIS hdf files.

Usage

```
process_modis_merge(
  path = NULL,
  date = NULL,
  subdataset = NULL,
  fun_agg = "mean",
  path_secondary = NULL,
  fusion_method = c("mean", "primary_first", "secondary_first"),
  ...
)
```

Arguments

path	character. Full list of hdf file paths. preferably a recursive search result from base::list.files.
date	character(1). date to query. Should be in "YYYY-MM-DD" format.
subdataset	character(1). subdataset names to extract. Should conform to regular expression. See base::regex for details. Default is NULL, which will result in errors. Users should specify which subdatasets will be imported.
fun_agg	Function name or custom function to aggregate overlapping cell values. See fun description in terra::tapp for details.
path_secondary	character. Optional secondary list of HDF/H5 paths (e.g., Aqua files) to fuse with path for improved temporal coverage.
fusion_method	character(1). Fusion method when path_secondary is provided: "mean", "primary_first", "secondary_first".
...	For internal use.

Value

a SpatRaster object

Note

Curvilinear products (i.e., swaths) will not be accepted. MODIS products downloaded by functions in amadeus, MODISTools, and luna are accepted.

Author(s)

Insang Song

See Also

download_data

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
mod09ga_merge <- process_modis_merge(
  path =
    list.files("./data", pattern = "MOD09GA.", full.names = TRUE),
  date = "2024-01-01",
  subdataset = "sur_refl_b01_1",
  fun_agg = "mean"
)

## End(Not run)
```

---

process_modis_swath *Mosaic MODIS swaths*

Description

This function will return a SpatRaster object with values of selected subdatasets. Swath data include curvilinear grids, which require warping/rectifying the original curvilinear grids into rectilinear grids. The function internally warps each of inputs then mosaic the warped images into one large SpatRaster object. Users need to select a subdataset to process. The full path looks like "HDF4_EOS:EOS_SWATH:{file_path}:mod06:subdataset", where file_path is the full path to the hdf file.

Usage

```
process_modis_swath(
  path = NULL,
  date = NULL,
  subdataset = NULL,
  suffix = ":mod06:",
  resolution = 0.05,
  ...
)
```

Arguments

path	character. Full paths of hdf files.
date	character(1). Date to query.
subdataset	character. Subdatasets to process. **Unlike other preprocessing functions, this argument should specify the exact subdataset name.** For example, when using MOD06_L2 product, one may specify `c("Cloud_Fraction", "Cloud_Optical_Thickness")`, etc. The subdataset names can be found in `terra::describe()` output.
suffix	character(1). Should be formatted `:{product}:`, e.g., `:mod06:`
resolution	numeric(1). Resolution of output raster. Unit is degree (decimal degree in WGS84).
...	For internal use.

Value

- a `SpatRaster` object (crs = `"EPSG:4326"`): if `path` is a single file with full specification of subdataset.
- a `SpatRaster` object (crs = `"EPSG:4326"`): if `path` is a list of files. In this case, the returned object will have the maximal extent of multiple warped layers

Author(s)

Insang Song

See Also

- `process_modis_warp()`, `stars::read_stars()`, `stars::st_warp()`
- GDAL HDF4 driver documentation
- `terra::describe()`: to list the full subdataset list with `sds = TRUE`
- `terra::sprc()`, `terra::rast()`

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
mod06l2_swath <- process_modis_swath(
  path = list.files(
    "./data/mod06l2",
    full.names = TRUE,
    pattern = ".hdf"
  ),
  date = "2024-01-01",
  subdataset = "Cloud_Fraction",
  suffix = ":mod06:",
  resolution = 0.05
)

## End(Not run)
```

process_narr *Process meteorological data*

Description

The process_narr() function imports and cleans raw meteorological data, returning a single SpatRaster object.

Usage

```
process_narr(
  date = "2023-09-01",
  variable = NULL,
  path = NULL,
  extent = NULL,
  ...
)
```

Arguments

date	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
variable	character(1). Variable name acronym. See List of Variables in NARR Files for variable names and acronym codes.
path	character(1). Directory with downloaded netCDF (.nc) files.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatRaster object

Note

Layer names of the returned SpatRaster object contain the variable acronym, pressure level, and date.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
process_narr(
  date = c("2018-01-01", "2018-01-10"),
  variable = "weasd",
  path = "./tests/testdata/narr/weasd"
)

## End(Not run)
```

`process_nei` *Process road emissions data*

Description

The `process_nei()` function imports and cleans raw road emissions data, returning a single `SpatVector` object.

NEI data comprises multiple csv files where emissions of 50+ pollutants are recorded at county level. With raw data files, this function will join a combined table of NEI data and county boundary, then perform a spatial join to target locations.

Usage

```
process_nei(path = NULL, county = NULL, year = c(2017, 2020), ...)
```

Arguments

`path`	character(1). Directory with NEI csv files.
`county`	`SpatVector`/`sf`. County boundaries.
`year`	integer(1) Year to use. Currently only 2017 or 2020 is accepted.
`...`	Placeholders.

Value

a `SpatVector` object

Note

Base files for `county` argument can be downloaded directly from U.S. Census Bureau or by using `tigris` package. This function does not reproject census boundaries. Users should be aware of the coordinate system of census boundary data for other analyses.

Author(s)

Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
nei <- process_nei(
  path = "./data",
  county = system.file("gpkg/nc.gpkg", package = "sf"),
  year = 2017
)

## End(Not run)
```

`process_nlcd` *Process land cover data*

Description

The `process_nlcd()` function imports and cleans raw land cover data, returning a single `SpatRaster` object.

Reads NLCD file of selected `year`.

Usage

```
process_nlcd(path = NULL, year = 2021, extent = NULL, ...)
```

Arguments

`path`	character giving nlcd data path
`year`	numeric giving the year of NLCD data used
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`...`	Placeholders.

Value

a `SpatRaster` object

Author(s)

Eva Marques, Insang Song

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
nlcd <- process_nlcd(
  path = "./data/",
  year = 2021
)

## End(Not run)
```

`process_population` *Process population density data*

Description

The `process_secac_population()` function imports and cleans raw population density data, returning a single `SpatRaster` object.

Usage

```
process_population(path = NULL, extent = NULL, ...)
```

Arguments

`path`	character(1). Path to GeoTIFF (.tif) or netCDF (.nc) file.
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`...`	Placeholders.

Value

a `SpatRaster` object

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
pop <- process_population(
  path = "./data/sedac_population_example.tif"
)

## End(Not run)
```

process_prism *Process PRISM data*

Description

This function imports and cleans raw PRISM data, returning a single SpatRaster object.

Reads time series or 30-year normal PRISM data.

Usage

```
process_prism(path = NULL, element = NULL, time = NULL, extent = NULL, ...)
```

Arguments

path	character giving PRISM data path Both file and directory path are acceptable.
element	character(1). PRISM element name
time	character(1). PRISM time name. Should be character in length of 2, 4, 6, or 8. "annual" is acceptable.
extent	numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded
...	Placeholders.

Value

a SpatRaster object with metadata of time and element.

Author(s)

Insang Song

See Also

terra::rast, terra::metags

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
prism <- process_prism(
  path = "./data/PRISM_ppt_stable_4kmM3_202104_nc.nc",
  element = "ppt",
  time = "202104"
)

## End(Not run)
```

`process_terraclimate` *Process TerraClimate data*

Description

The `process_terraclimate()` function imports and cleans climate and water balance data, returning a single `SpatRaster` object.

Usage

```
process_terraclimate(
  date = c("2023-09-01", "2023-09-10"),
  variable = NULL,
  path = NULL,
  extent = NULL,
  ...
)
```

Arguments

`date`	character(1 or 2). Date (1) or start and end dates (2). Format YYYY-MM-DD (ex. September 1, 2023 = "2023-09-01").
`variable`	character(1). Variable name or acronym code. See TerraClimate Direct Downloads for variable names and acronym codes.
`path`	character(1). Directory with downloaded netCDF (.nc) files.
`extent`	numeric(4) or SpatExtent giving the extent of the raster if `NULL` (default), the entire raster is loaded
`...`	Placeholders.

Value

a `SpatRaster` object

Note

Layer names of the returned `SpatRaster` object contain the variable acronym, year, and month.

TerraClimate data has monthly temporal resolution, so the first day of each month is used as a placeholder temporal value.

Author(s)

Mitchell Manware

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
terraclimate <- process_terraclimate(
  date = c("2023-01-01", "2023-01-10"),
  variable = "Precipitation",
  path = "./data/ppt"
)

## End(Not run)
```

`process_tri` *Process toxic release data*

Description

This function imports and cleans raw toxic release data, returning a single `SpatVector` (points) object for the selected `year`.

Usage

```
process_tri(
  path = NULL,
  year = 2018,
  variables = "STACK_AIR",
  chemical = NULL,
  industry_group = c("none", "industry_sector", "industry_sector_code", "both"),
  ignore_case = TRUE,
  extent = NULL,
  ...
)
```

Arguments

`path`	character(1). Path to the directory with TRI CSV files
`year`	integer(1). Single year to select.
`variables`	character. One or more regular expressions used to select TRI release variables by column name after normalization to underscore naming (for example, `STACK_AIR`, `FUGITIVE_AIR`, `WATER`). Default is `"STACK_AIR"`. Matching first uses raw TRI column names, then falls back to a normalized match where punctuation and spaces are converted to underscores (for example, `"ON-SITE RELEASE TOTAL"` matches `ON_SITE_RELEASE_TOTAL`). Recommended options include: • `FUGITIVE_AIR` • `STACK_AIR`

- WATER
- UNDERGROUND
- UNDERGROUND_CL_I
- UNDERGROUND_C_II_V
- LANDFILLS
- RCRA_C_LANDFILL
- OTHER_LANDFILLS
- LAND_TREATMENT
- SURFACE_IMPNDMNT
- RCRA_SURFACE_IM
- OTHER_SURFACE_I
- OTHER_DISPOSAL
- ON_SITE_RELEASE_TOTAL
- POTW_TRNS_RLSE
- POTW_TRNS_TRT
- POTW_TOTAL_TRANSFERS

`chemical` NULL or character. Optional one or more regular expressions used to filter chemicals. Patterns are matched against TRI_CHEMICAL_COMPOUND_ID, CHEMICAL, and CAS/CAS. values. If NULL (default), all chemicals are retained.

`industry_group` character(1). Optional additional grouping level. One of "none" (default), "industry_sector", "industry_sector_code", or "both".

`ignore_case` logical(1). If TRUE (default), regular expression matching in variables and chemical is case-insensitive.

`extent` numeric(4) or SpatExtent giving the extent of the raster if NULL (default), the entire raster is loaded

`...` Placeholders.

Value

a SpatVector object (points) in year year is stored in a field named "year".

Note

Use get_tri_info() to inspect available TRI chemical IDs/names/CAS numbers and industry sector codes in local TRI files. Visit TRI Data and Tools to view the available years and variables.

Author(s)

Kyle Messier

References

https://www.epa.gov/toxics-release-inventory-tri-program/tri-toolbox

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
tri <- process_tri(
  path = "./data",
  year = 2020,
  variables = c("STACK_AIR", "FUGITIVE_AIR"),
  chemical = "benzene",
  industry_group = "industry_sector"
)

## End(Not run)
```

`setup_nasa_token` *Set up NASA EarthData authentication*

Description

Interactive helper to securely set up NASA EarthData authentication. This function guides users through setting up their token in a secure way that won't be exposed in scripts or version control.

Usage

```
setup_nasa_token(method = c("renviron", "file", "session"), token = NULL)
```

Arguments

`method`	character(1). Setup method: • "renviron": Add to ~/.Renviron (recommended, persists across sessions) • "file": Save to ~/.nasa_earthdata_token file • "session": Set for current R session only
`token`	character(1). Your NASA EarthData token. If NULL, will prompt.

Value

invisible(NULL). Sets up authentication.

Examples

```
## Not run:
# Interactive setup (recommended)
setup_nasa_token()

# Save to .Renviron for permanent setup
setup_nasa_token(method = "renviron", token = "your_token_here")

# Save to file
```

```
setup_nasa_token(method = "file", token = "your_token_here")

# Current session only
setup_nasa_token(method = "session", token = "your_token_here")

## End(Not run)
```

`sftime_as_mysftime` *Convert an* `sftime` *to a* `mysftime`

Description

Convert an `sftime` object to a `mysftime` object. `x` must contain a time-defining column, identified in `timename`.

Usage

```
sftime_as_mysftime(x, timename)
```

Arguments

`x`	an `sftime` object
`timename`	character: name of time column in `x`

Value

an `sftime` object with specific format

Author(s)

Eva Marques

See Also

check_mysftime

sftime_as_sf *Convert an* sftime *to an* sf

Description

Convert an sftime object to an sf object. x must contain a time-defining column, identified in timename.

Usage

```
sftime_as_sf(x, keeptime = TRUE)
```

Arguments

x	an sftime object
keeptime	boolean: TRUE if user wants to keep time column as simple column (default = TRUE)

Value

an sf object

Author(s)

Eva Marques

sftime_as_spatraster *Convert an* sftime *to a* SpatRaster

Description

Convert an sftime object to a SpatRaster object. Returns a SpatRatser with one layer for each time step in x.

Usage

```
sftime_as_spatraster(x, varname)
```

Arguments

x	an sftime object
varname	variable to rasterize

Value

a SpatRaster object

Note

Running `sftime_as_spatraster` can take a long time if `x` is not spatially structured.

Author(s)

Eva Marques

See Also

terra::rast

`sftime_as_spatrds` *Convert an* `sftime` *to a* `SpatRasterDataset`

Description

Convert an `sftime` object to a `SpatRasterDataset` object.

Usage

```
sftime_as_spatrds(x)
```

Arguments

`x`	an `sftime` object

Value

an `SpatRasterDataset` object

Note

Running `sftime_as_spatrds` can take a long time if `x` is not spatially and temporally structured.

Author(s)

Eva Marques

See Also

terra::sds

sftime_as_spatvector *Convert an* sftime *to a* SpatVector

Description

Convert an sftime object to a SpatVector object.

Usage

```
sftime_as_spatvector(x)
```

Arguments

x	an sftime object

Value

a SpatVector object

Author(s)

Eva Marques

See Also

terra::vect

sf_as_mysftime *Convert an* sf *to an* sftime

Description

Convert an sf object to an sftime object. x must contain a time-defining column, identified in timename.

Usage

```
sf_as_mysftime(x, timename)
```

Arguments

x	an sf object
timename	character: name of time column in x

Value

an sftime object

Author(s)

Eva Marques

`spatraster_as_sftime` *Convert a* `SpatRaster` *to an* `sftime`

Description

Convert a `SpatRaster` object to an `sftime` object. `x` must contain a time-defining column, identified in `timename`.

Usage

```
spatraster_as_sftime(x, varname, timename = "time")
```

Arguments

`x`	a `SpatRaster` object
`varname`	character for variable column name in the sftime
`timename`	character for time column name in the sftime (default: "time")

Value

a `sftime` object

Author(s)

Eva Marques

See Also

terra::rast

`spatrds_as_sftime` *Convert a* `SpatRasterDataset` *to an* `sftime`

Description

Convert a `SpatRasterDataset` object to an `sftime` object. `x` must contain a time-defining column, identified in `timename`.

Usage

```
spatrds_as_sftime(x, timename = "time")
```

Arguments

x	a SpatRasterDataset object (~ list of named SpatRasters)
timename	character for time column name in the sftime (default: "time")

Value

an sftime object

Author(s)

Eva Marques

See Also

terra::sds

---

spatvector_as_sftime *Convert a* SpatVector *to an* sftime

Description

Convert a SpatVector object to an sftime object. x must contain a time-defining column, identified in timename.

Usage

```
spatvector_as_sftime(x, timename = "time")
```

Arguments

x	a SpatVector object
timename	character for time column name in x (default: "time")

Value

an sftime object

Author(s)

Eva Marques

See Also

terra::vect

sum_edc *Calculate isotropic Sum of Exponentially Decaying Contributions (SEDC) covariates*

Description

Calculate isotropic Sum of Exponentially Decaying Contributions (SEDC) covariates

Usage

```
sum_edc(
  from = NULL,
  locs = NULL,
  locs_id = NULL,
  decay_range = NULL,
  target_fields = NULL,
  C0 = NULL,
  use_threshold = TRUE,
  geom = FALSE
)
```

Arguments

from	SpatVector(1). Point locations which contain point-source covariate data.
locs	sf/SpatVector(1). Locations where the sum of exponentially decaying contributions are calculated.
locs_id	character(1). Name of the unique id field in point_to.
decay_range	numeric(1). Distance at which the source concentration is reduced to exp(-3) (approximately -95 %)
target_fields	character(varying). Field names in characters.
C0	NULL, character(1), or numeric vector of length nrow(from). Optional initial source values at pollutant locations. If NULL (default), all source values are set to 1. If character(1), the value is treated as a column name in from and used as source values.
use_threshold	logical(1). If TRUE (default), include only source points within 5 * decay_range from each target location. If FALSE, include all source points in from.
geom	FALSE/"sf"/"terra".. Should the function return with geometry? Default is FALSE, options with geometry are "sf" or "terra". The coordinate reference system of the sf or SpatVector is that of from.

Value

a data.frame (tibble) or SpatVector object with input field names with a suffix "_sedc" where the sums of EDC are stored. Additional attributes are attached for the EDC information.

- ‘attr(result, "decay_range")‘: the range where concentration reduces to approximately five percent
- ‘attr(result, "sedc_threshold")‘: the threshold distance at which emission source points are excluded beyond that

Note

The function is originally from chopin Distance calculation is done with terra functions internally. Thus, the function internally converts sf objects in `point_*` arguments to terra. The threshold should be carefully chosen by users.

Author(s)

Insang Song

References

Messier KP, Akita Y, Serre ML (2012). “Integrating Address Geocoding, Land Use Regression, and Spatiotemporal Geostatistical Estimation for Groundwater Tetrachloroethylene.” *Environmental Science & Technology*, **46**(5), 2772–2780. ISSN 0013-936X, doi:10.1021/es203152a.

Wiesner C (????). “Euclidean Sum of Exponentially Decaying Contributions Tutorial.”

Examples

```
## NOTE: Example is wrapped in `\dontrun{}` as function requires a large
##       amount of data which is not included in the package.
## Not run:
set.seed(101)
ncpath <- system.file("gpkg/nc.gpkg", package = "sf")
nc <- terra::vect(ncpath)
nc <- terra::project(nc, "EPSG:5070")
pnt_locs <- terra::centroids(nc, inside = TRUE)
pnt_locs <- pnt_locs[, "NAME"]
pnt_from <- terra::spatSample(nc, 10L)
pnt_from$pid <- seq(1, 10)
pnt_from <- pnt_from[, "pid"]
pnt_from$val1 <- rgamma(10L, 1, 0.05)
pnt_from$val2 <- rgamma(10L, 2, 1)

vals <- c("val1", "val2")
sum_edc(pnt_locs, pnt_from, "NAME", 1e4, vals)

## End(Not run)
```

Index