Package 'amadeus'

January 16, 2025

Title Accessing and Analyzing Large-Scale Environmental Data

Version 1.2.3

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 (>= 4.1.0)

Imports dplyr, sf, sftime, stats, terra, methods, data.table, httr, rvest, exactextractr, utils, stringi, testthat (>= 3.0.0), stars, tidyr, rlang, nhdplusTools, archive, collapse, Rdpack

Suggests covr, withr, knitr, rmarkdown, lwgeom, FNN, doRNG, devtools, stringr, tigris, spelling

RdMacros Rdpack

Encoding UTF-8

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.3.2

Config/Needs/website tidyverse/tidytemplate

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

Repository CRAN

Date/Publication 2025-01-16 22:00:06 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"),
   from,
   locs,
   locs_id = "site_id",
   ...
)
```

Arguments

`covariate`	character(1). Covariate type.
`from`	character. Single or multiple from strings.
`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"`.
`...`	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_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"

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_ecoregion *Calculate ecoregions covariates*

Description

Extract ecoregions covariates (U.S. EPA Ecoregions Level 2/3) at point locations. Returns a `data.frame` object containing `locs_id` and binary (0 = point not in ecoregion; 1 = point in ecoregion) variables for each ecoregion.

Usage

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

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.
`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 object with dummy variables and attributes of:

- `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",
  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",
  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`).
`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_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 reflects the elevation statistic, spatial resolution of `from`, and circular buffer radius (ie. Breakline Emphasis at 7.5 arc-second resolution with 0 meter buffer: breakline_emphasis_r75_0).

Usage

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

`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",
  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`).
`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://earthdata.nasa.gov/data/catalog/sedac-ciesin-sedac-groads-v1-1.00`.

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 locations. Returns a `data.frame` object containing `locs_id`, date, and binary variable for wildfire smoke plume density inherited from `from` (0 = point not covered by wildfire smoke plume; 1 = point covered by wildfire smoke plume).

Usage

```
calculate_hms(from, locs, locs_id = NULL, radius = 0, 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).
`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_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_koppen_geiger

Calculate climate classification covariates

Description

Extract climate classification values at point locations. Returns a `data.frame` object containing `locs_id` and binary (0 = point not in climate region; 1 = point in climate region) variables for each climate classification region.

Usage

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

Arguments

`from`	SpatVector(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.
`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

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",
  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`).
`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

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 in multiple CPU threads*

Description

calculate_modis essentially runs calculate_modis_daily function in each thread (subprocess). Based on daily resolution, each day's workload will be distributed to each thread. With product argument, the files are processed by a customized function where the unique structure and/or characteristics of the products are considered.

Usage

```
calculate_modis(
  from = NULL,
  locs = NULL,
  locs_id = "site_id",
  radius = c(0L, 1000L, 10000L, 50000L),
  preprocess = process_modis_merge,
  name_covariates = NULL,
  subdataset = NULL,
  fun_summary = "mean",
  package_list_add = NULL,
  export_list_add = NULL,
  max_cells = 3e+07,
  geom = FALSE,
  ...
)
```

Arguments

`from`	character. List of paths to MODIS/VIIRS files.
`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.
`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.
`package_list_add`	character. A vector with package names to load these in each thread. Note that `sf`, `terra`, `exactextractr`, `doParallel`, `parallelly` and `dplyr` are the default packages to be loaded.
`export_list_add`	character. A vector with object names to export to each thread. It should be minimized to spare memory.
`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`.
`...`	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.

Note

Overall, this function and dependent routines assume that the file system can handle concurrent access to the (network) disk by multiple processes. File system characteristics, package versions, and hardware settings and specification can affect the processing efficiency. `locs` is expected to be convertible to `sf` object. `sf`, `SpatVector`, and other class objects that could be converted to `sf` can be used. 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. 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_modis_daily` *A single-date MODIS worker*

Description

The function operates at MODIS/VIIRS products on a daily basis. Given that the raw hdf files are downloaded from NASA, standard file names include a data retrieval date flag starting with letter "A". Leveraging that piece of information, the function will select files of scope on the date of interest. Please note that this function does not provide a function to filter swaths or tiles, so it is strongly recommended to check and pre-filter the file names at users' discretion.

Usage

```
calculate_modis_daily(
  from = NULL,
  locs = NULL,
  locs_id = "site_id",
  radius = 0L,
  date = NULL,
  name_extracted = NULL,
  fun_summary = "mean",
  max_cells = 3e+07,
  geom = FALSE,
  ...
)
```

Arguments

`from`	SpatRaster. Preprocessed objects.
`locs`	SpatVector/sf/sftime object. Locations where MODIS values are summarized.
`locs_id`	character(1). Field name where unique site identifiers are stored. Default is `"site_id"`
`radius`	numeric. Radius to generate circular buffers.
`date`	Date(1). date to query.
`name_extracted`	character. Names of calculated covariates.
`fun_summary`	function. Summary function for multilayer rasters. Passed to `foo`. See `exactextractr::exact_extract` for details.
`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`.
`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`. See `exactextractr::exact_extract` for details.
`...`	Placeholders.

Value

a data.frame or SpatVector object.

Author(s)

Insang Song

See Also

- Preprocessing: `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")
calculate_modis_daily(
  from = mod06l2_warp, # dervied from process_modis() example
  locs = locs,
  locs_id = "id",
  radius = 0,
  date = "2024-01-01",
  name_extracted = "cloud_fraction_0",
  fun_summary = "mean",
  max_cells = 3e7
)

## 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",
  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`).
`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", 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.
`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,
  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.
`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",
  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`).

`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_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),
  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)`.
`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",
  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`).
`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 chemical in `from`.

Usage

```
calculate_tri(
  from = NULL,
  locs,
  locs_id = "site_id",
  radius = c(1000L, 10000L, 50000L),
  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"`.

`radius`	Circular buffer radius. Default is `c(1000, 10000, 50000)` (meters)
`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",
  radius = 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 = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`parameter_code`	integer(1). length of 5. EPA pollutant parameter code. For details, please refer to AQS parameter codes
`resolution_temporal`	character(1). Name of column containing POC values. Currently, no value other than `"daily"` works.
`year`	character(1 or 2). length of 4. 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. 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 `TRUE`.
`remove_zip`	logical(1). Remove zip file from directory_to_download. Default `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`.

Value

- 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 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`.

Examples

```
## Not run:
download_aqs(
  parameter_code = 88101,
  resolution_temporal = "daily",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## 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 = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  hash = FALSE
)
```

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`.

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", "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"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  hash = FALSE,
  ...
)
```

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`.
`...`	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_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"

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_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) Ecorgions. Level 3 data, where all pieces of information in the higher levels are included, are downloaded.

Usage

```
download_ecoregion(
 epa_certificate_path = system.file("extdata/cacert_gaftp_epa.pem", package = "amadeus"),
  certificate_url =
    "http://cacerts.digicert.com/DigiCertGlobalG2TLSRSASHA2562020CA1-1.crt",
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`epa_certificate_path`	character(1). Path to the certificate file for EPA DataCommons. Default is 'extdata/cacert_gaftp_epa.pem' under the package installation path.
`certificate_url`	character(1). URL to certificate file. See notes for details.
`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.
`unzip`	logical(1). Unzip zip files. Default `TRUE`.
`remove_zip`	logical(1). Remove zip file from `directory_to_download`. Default `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`.

Value

- 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`.

Note

For EPA Data Commons certificate errors, follow the steps below:

1. Click Lock icon in the address bar at https://gaftp.epa.gov
2. Click Show Certificate
3. Access Details
4. Find URL with *.crt extension Currently we bundle the pre-downloaded crt and its PEM (which is accepted in wget command) file in ./inst/extdata. The instruction above is for certificate updates in the future.

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

## 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) 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"),
  date = c("2018-01-01", "2018-01-01"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

collection	character(1). GEOS-CF data collection file name.
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. Sub-directories will be created within directory_to_save for each GEOS-CF collection.
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.
hash	logical(1). By setting TRUE the function will return an rlang::hash_file() hash character corresponding to the downloaded files. Default is FALSE.

Value

- For hash = FALSE, NULL
- For hash = TRUE, an rlang::hash_file character.
- netCDF (.nc4) files will be stored in a collection-specific folder within directory_to_save.

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,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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 and National Geospatial-Intelligence Agency.

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 = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`statistic`	character(1). Available statistics include `"Breakline Emphasis"`, `"Systematic Subsample"`, `"Median Statistic"`, `"Minimum Statistic"`, `"Mean Statistic"`, `"Maximum Statistic"`, and `"Standard Deviation Statistic"`.
`resolution`	character(1). Available resolutions include `"7.5 arc-seconds"`, `"15 arc-seconds"`, and `"30 arc-seconds"`.
`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`.

Value

- 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)

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

## 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 = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

`variables`	character(1). Variable(s) name(s). 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").
`year`	character(1 or 2). length of 4. Year or start/end years for downloading data.
`directory_to_save`	character(1). Directory(s) to save downloaded 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.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

- For `hash = FALSE`, NULL
- For `hash = TRUE`, an `rlang::hash_file` character.
- netCDF (.nc) files will be stored in a variable-specific folder within `directory_to_save`.

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 = "Precipitation",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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), v1 (1980-2010).

Usage

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

Arguments

`data_region`	character(1). Data can be downloaded for `"Global"`, `"Africa"`, `"Asia"`, `"Europe"`, `"Americas"`, `"Oceania East"`, and `"Oceania West"`.
`data_format`	character(1). Data can be downloaded as `"Shapefile"` or `"Geodatabase"`. (Only `"Geodatabase"` available for `"Global"` region).
`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 shapefiles ("/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.
`unzip`	logical(1). Unzip zip files. Default is `TRUE`.
`remove_zip`	logical(1). Remove zip files 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`.

Value

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

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://earthdata.nasa.gov/data/catalog/sedac-ciesin-sedac-groads-v1-1.00`.

Examples

```
## Not run:
download_groads(
  data_region = "Americas",
  data_format = "Shapefile",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## 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 = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`data_format`	character(1). "Shapefile" or "KML".
`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. If `data_format = "Shapefile"`, two sub-directories will be created for the downloaded zip files ("/zip_files") and the unzipped shapefiles ("/data_files"). If `data_format = "KML"`, a single sub-directory ("/data_files") will be created.

`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 zip files. Default is `TRUE`. (Ignored if `data_format = "KML"`.)
`remove_zip`	logical(1). Remove zip files from directory_to_download. Default is `FALSE`. (Ignored if `data_format = "KML"`.)
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

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

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

## 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 = FALSE,
  remove_command = FALSE,
  unzip = FALSE,
  hash = FALSE
)
```

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`. Not working for this function since HUC data is in 7z format.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

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_koppen_geiger` *Download climate classification data*

Description

The `download_koppen_geiger()` function accesses and downloads climate classification data from the *Present and future Köppen-Geiger climate classification maps at 1-km resolution*(link for article; link for 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 = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`data_resolution`	character(1). Available resolutions are `"0.0083"` degrees (approx. 1 km), `"0.083"` degrees (approx. 10 km), and `"0.5"` degrees (approx. 50 km).
`time_period`	character(1). Available times are `"Present"` (1980-2016) and `"Future"` (2071-2100). ("Future" classifications are based on scenario RCP8.5).
`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 shapefiles ("/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.
`unzip`	logical(1). Unzip zip files. Default is `TRUE`.
`remove_zip`	logical(1). Remove zip files 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`.

Value

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

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

## 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.

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"),
  date = c("2018-01-01", "2018-01-01"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

`collection`	character(1). MERRA-2 data collection file name.
`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). `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`).
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`. the text file containing download commands.

Value

- For `hash = FALSE`, NULL
- For `hash = TRUE`, an `rlang::hash_file` character.
- netCDF (.nc4) files will be stored in a collection-specific folder within `directory_to_save`.

Author(s)

Mitchell Manware, Insang Song

References

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst1_2d_ asm_ Nx: 2d,3-Hourly,Instantaneous,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/3Z173KIE2TPD, `https://disc.gsfc.nasa.gov/datasets/M2I1NXASM_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst1_2d_ int_ Nx: 2d,1-Hourly,Instantaneous,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.” doi:10.5067/G0U6NGQ3BLE0, `https://disc.gsfc.nasa.gov/datasets/M2I1NXINT_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst1_2d_ lfo_ Nx: 2d,1-Hourly,Instantaneous,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/RCMZA6TL70BG, `https://disc.gsfc.nasa.gov/datasets/M2I1NXLFO_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_3d_ asm_ Np: 3d,3-Hourly,Instantaneous,Pressure-Level,Assimilation,Assimilated Meteorological Fields V5.12.4.” doi:10.5067/QBZ6MG944HW0, `https://disc.gsfc.nasa.gov/datasets/M2I3NPASM_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_3d_ aer_ Nv: 3d,3-Hourly,Instantaneous,Model-Level,Assimilation,Aerosol Mixing Ratio V5.12.4.” doi:10.5067/LTVB4GPCOTK2, `https://disc.gsfc.nasa.gov/datasets/M2I3NVAER_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_3d_ asm_ Nv: 3d,3-Hourly,Instantaneous,Model-Level,Assimilation,Assimilated Meteorological Fields V5.12.4.” doi:10.5067/WWQSXQ8IVFW8, https://disc.gsfc.nasa.gov/datasets/M2I3NVASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_3d_ chm_ Nv: 3d,3-Hourly,Instantaneous,Model-Level,Assimilation,Carbon Monoxide and Ozone Mixing Ratio V5.12.4.” doi:10.5067/HO9OVZWF3KW2, https://disc.gsfc.nasa.gov/datasets/M2I3NVCHM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_3d_ gas_ Nv: 3d,3-Hourly,Instantaneous,Model-Level,Assimilation,Aerosol Mixing Ratio Analysis Increments V5.12.4.” doi:10.5067/96BUID8HGGX5, https://disc.gsfc.nasa.gov/datasets/M2I3NVGAS_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst3_2d_ gas_ Nx: 2d,3-Hourly,Instantaneous,Single-Level,Assimilation,Aerosol Optical Depth Analysis V5.12.4.” doi:10.5067/HNGA0EWW0R09, https://disc.gsfc.nasa.gov/datasets/M2I3NXGAS_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst6_3d_ ana_ Np: 3d,6-Hourly,Instantaneous,Pressure-Level,Analysis,Analyzed Meteorological Fields V5.12.4.” doi:10.5067/A7S6XP56VZWS, https://disc.gsfc.nasa.gov/datasets/M2I6NPANA_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 inst6_3d_ ana_ Nv: 3d,6-Hourly,Instantaneous,Model-Level,Analysis,Analyzed Meteorological Fields V5.12.4.” doi:10.5067/IUUF4WB9FT4W, https://disc.gsfc.nasa.gov/datasets/M2I6NVANA_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 statD_2d_ slv_ Nx: 2d,Monthly,Aggregated Statistics,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/KVIMOMCUO83U, https://disc.gsfc.nasa.gov/datasets/M2SMNXSLV_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 statD_2d_ slv_ Nx: 2d,Daily,Aggregated Statistics,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/9SC1VNTWGWV3, https://disc.gsfc.nasa.gov/datasets/M2SDNXSLV_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ adg_ Nx: 2d,3-Hourly,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics (extended) V5.12.4.” doi:10.5067/HM00OHQBHKTP, https://disc.gsfc.nasa.gov/datasets/M2T1NXADG_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ aer_ Nx: 2d,1-Hourly,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics V5.12.4.” doi:10.5067/KLICLTZ8EM9D, https://disc.gsfc.nasa.gov/datasets/M2T1NXAER_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ chm_ Nx: 2d,3-Hourly,Time-Averaged,Single-Level,Assimilation,Carbon Monoxide and Ozone Diagnostics V5.12.4.” doi:10.5067/3RQ5YS674DGQ, https://disc.gsfc.nasa.gov/datasets/M2T1NXCHM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ csp_ Nx: 2d,1-Hourly,Time-averaged,Single-Level,Assimilation,COSP Satellite Simulator V5.12.4.” doi:10.5067/H0VVAD8F6MX5, https://disc.gsfc.nasa.gov/datasets/M2T1NXCSP_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ flx_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4.” doi:10.5067/7MCPBJ41Y0K6, https://disc.gsfc.nasa.gov/datasets/M2T1NXFLX_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ int_ Nx: 2d,1-Hourly,Time-averaged,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.”

doi:10.5067/Q5GVUVUIVGO7, https://disc.gsfc.nasa.gov/datasets/M2T1NXINT_5.12.4/summary.

Pawson S (2020). “MERRA-2 tavg1_2d_ lfo_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/L0T5GEG1NYFA, https://disc.gsfc.nasa.gov/datasets/M2T1NXLFO_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ lnd_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Land Surface Diagnostics V5.12.4.” doi:10.5067/RKPHT8KC1Y1T, https://disc.gsfc.nasa.gov/datasets/M2T1NXLND_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ ocn_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Ocean Surface Diagnostics V5.12.4.” doi:10.5067/Y67YQ1L3ZZ4R, https://disc.gsfc.nasa.gov/datasets/M2T1NXOCN_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ rad_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/Q9QMY5PBNV1T, https://disc.gsfc.nasa.gov/datasets/M2T1NXRAD_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg1_2d_ slv_ Nx: 2d,1-Hourly,Time-Averaged,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/VJAFPLI1CSIV, https://disc.gsfc.nasa.gov/datasets/M2T1NXSLV_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ mst_ Ne: 3d,3-Hourly,Time-Averaged,Model-Level Edge,Assimilation,Moist Processes Diagnostics V5.12.4.” doi:10.5067/JRUZ3SJ3ZJ72, https://disc.gsfc.nasa.gov/datasets/M2T3NEMST_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ trb_ Ne: 3d,3-Hourly,Time-Averaged,Model-Level Edge,Assimilation,Turbulence Diagnostics V5.12.4.” doi:10.5067/4I7ZI35QRH8K, https://disc.gsfc.nasa.gov/datasets/M2T3NETRB_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ nav_ Ne: 3d,3-Hourly,Time-Averaged, Vertical Coordinates V5.12.4.” doi:10.5067/N5WAKNS1UYQN, https://disc.gsfc.nasa.gov/datasets/M2T3NENAV_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ cld_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Cloud Diagnostics V5.12.4.” doi:10.5067/TX10URJSKT53, https://disc.gsfc.nasa.gov/datasets/M2T3NPCLD_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ mst_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Moist Processes Diagnostics V5.12.4.” doi:10.5067/0TUFO90Q2PMS, https://disc.gsfc.nasa.gov/datasets/M2T3NPMST_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ rad_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/3UGE8WQXZAOK, https://disc.gsfc.nasa.gov/datasets/M2T3NPRAD_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ tdt_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Temperature Tendencies V5.12.4.” doi:10.5067/9NCR9DDDOPFI, https://disc.gsfc.nasa.gov/datasets/M2T3NPTDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ trb_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Turbulence Diagnostics V5.12.4.” doi:10.5067/ZRRJPGWL8AVL, https://disc.gsfc.nasa.gov/datasets/M2T3NPTRB_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ udt_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Wind Tendencies V5.12.4.” doi:10.5067/CWV0G3PPPWFW, https://disc.gsfc.nasa.gov/datasets/M2T3NPUDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ odt_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Ozone Tendencies V5.12.4.” doi:10.5067/S0LYTK57786Z, https://disc.gsfc.nasa.gov/datasets/M2T3NPODT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ qdt_ Np: 3d,3-Hourly,Time-Averaged,Pressure-Level,Assimilation,Moist Tendencies V5.12.4.” doi:10.5067/A9KWADY78YHQ, https://disc.gsfc.nasa.gov/datasets/M2T3NPQDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ asm_ Nv: 3d,3-Hourly,Time-Averaged,Model-Level,Assimilation,Assimilated Meteorological Fields V5.12.4.” doi:10.5067/SUOQESM06LPK, https://disc.gsfc.nasa.gov/datasets/M2T3NVASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ cld_ Nv: 3d,3-Hourly,Time-Averaged,Model-Level,Assimilation,Cloud Diagnostics V5.12.4.” doi:10.5067/F9353J0FAHIH, https://disc.gsfc.nasa.gov/datasets/M2T3NVCLD_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ mst_ Nv: 3d,3-Hourly,Time-Averaged,Model-Level,Assimilation,Moist Processes Diagnostics V5.12.4.” doi:10.5067/ZXTJ28TQR1TR, https://disc.gsfc.nasa.gov/datasets/M2T3NVMST_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_3d_ rad_ Nv: 3d,3-Hourly,Time-Averaged,Model-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/7GFQKO1T43RW, https://disc.gsfc.nasa.gov/datasets/M2T3NVRAD_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavg3_2d_ glc_ Nx: 2d,3-Hourly,Time-Averaged,Single-Level,Assimilation,Land Ice Surface Diagnostics V5.12.4.” doi:10.5067/9ETB4TT5J6US, https://disc.gsfc.nasa.gov/datasets/M2T3NXGLC_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_2d_ asm_ Nx: 2d,Monthly mean,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/5ESKGQTZG7FO, https://disc.gsfc.nasa.gov/datasets/M2IMNXASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_2d_ int_ Nx: 2d,Monthly mean,Instantaneous,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.” doi:10.5067/KVTU1A8BWFSJ, https://disc.gsfc.nasa.gov/datasets/M2IMNXINT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_2d_ lfo_ Nx: 2d,Monthly mean,Instantaneous,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/11F99Y6TXN99, https://disc.gsfc.nasa.gov/datasets/M2IMNXLFO_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_2d_ gas_ Nx: 2d,Monthly mean,Instantaneous,Single-Level,Assimilation,Aerosol Optical Depth Analysis V5.12.4.” doi:10.5067/XOGNBQEPLUC5, https://disc.gsfc.nasa.gov/datasets/M2IMNXGAS_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_3d_ asm_ Np: 3d,Monthly mean,Instantaneous,Pressure-Level,Assimilation,Assimilated Meteorological Fields V5.12.4.” doi:10.5067/2E096JV59PK7, https://disc.gsfc.nasa.gov/datasets/M2IMNPASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instM_3d_ ana_ Np: 3d,Monthly mean,Instantaneous,Pressure-Level,Analysis,Analyzed Meteorological Fields V5.12.4.” doi:10.5067/V92O8XZ30XBI, https://disc.gsfc.nasa.gov/datasets/M2IMNPANA_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ adg_ Nx: 2d,Monthly mean,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics (extended) V5.12.4.” doi:10.5067/RZIK2TV7PP38, https://disc.gsfc.nasa.gov/datasets/M2TMNXADG_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ aer_ Nx: 2d,Monthly mean,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics V5.12.4.” doi:10.5067/FH9A0MLJPC7N, `https://disc.gsfc.nasa.gov/datasets/M2TMNXAER_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ chm_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Carbon Monoxide and Ozone Diagnostics V5.12.4.” doi:10.5067/WMT31RKEXK8I, `https://disc.gsfc.nasa.gov/datasets/M2TMNXCHM_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ csp_ Nx: 2d,Monthly mean,Time-averaged,Single-Level,Assimilation,COSP Satellite Simulator V5.12.4.” doi:10.5067/BZPOTGJOQKLU, `https://disc.gsfc.nasa.gov/datasets/M2TMNXCSP_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ flx_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4.” doi:10.5067/0JRLVL8YV2Y4, `https://disc.gsfc.nasa.gov/datasets/M2TMNXFLX_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ int_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.” doi:10.5067/FQPTQ4OJ22TL, `https://disc.gsfc.nasa.gov/datasets/M2TMNXINT_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ lfo_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/5V7K6LJD44SY, `https://disc.gsfc.nasa.gov/datasets/M2TMNXLFO_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ lnd_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Land Surface Diagnostics V5.12.4.” doi:10.5067/8S35XF81C28F, `https://disc.gsfc.nasa.gov/datasets/M2TMNXLND_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ ocn_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Ocean Surface Diagnostics V5.12.4.” doi:10.5067/4IASLIDL8EEC, `https://disc.gsfc.nasa.gov/datasets/M2TMNXOCN_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ rad_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/OU3HJDS973O0, `https://disc.gsfc.nasa.gov/datasets/M2TMNXRAD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ slv_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/AP1B0BA5PD2K, `https://disc.gsfc.nasa.gov/datasets/M2TMNXSLV_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_2d_ glc_ Nx: 2d,Monthly mean,Time-Averaged,Single-Level,Assimilation,Land Ice Surface Diagnostics V5.12.4.” doi:10.5067/5W8Q3I9WUFGX, `https://disc.gsfc.nasa.gov/datasets/M2TMNXGLC_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ cld_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Cloud Diagnostics V5.12.4.” doi:10.5067/J9R0LXGH48JR, `https://disc.gsfc.nasa.gov/datasets/M2TMNPCLD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ mst_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Moist Processes Diagnostics V5.12.4.” doi:10.5067/ZRZGD0DCK1CG, `https://disc.gsfc.nasa.gov/datasets/M2TMNPMST_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ rad_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/H3YGROBVBGFJ, `https://disc.gsfc.nasa.gov/datasets/M2TMNPRAD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ tdt_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Temperature Tendencies V5.12.4.” doi:10.5067/VILT59HI2MOY, https://disc.gsfc.nasa.gov/datasets/M2TMNPTDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ trb_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Turbulence Diagnostics V5.12.4.” doi:10.5067/2YOIQB5C3ACN, https://disc.gsfc.nasa.gov/datasets/M2TMNPTRB_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ udt_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Wind Tendencies V5.12.4.” doi:10.5067/YSR6IA5057XX, https://disc.gsfc.nasa.gov/datasets/M2TMNPUDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ odt_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Ozone Tendencies V5.12.4.” doi:10.5067/Z2KCWAV4GPD2, https://disc.gsfc.nasa.gov/datasets/M2TMNPODT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgM_3d_ qdt_ Np: 3d,Monthly mean,Time-Averaged,Pressure-Level,Assimilation,Moist Tendencies V5.12.4.” doi:10.5067/2ZTU87V69ATP, https://disc.gsfc.nasa.gov/datasets/M2TMNPQDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 const_2d_ asm_ Nx: 2d, constants.” doi:10.5067/ME5QX6Q5IGGU, https://disc.gsfc.nasa.gov/datasets/M2C0NXASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_2d_ asm_ Nx: 2d,Diurnal,Instantaneous,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/BOJSTZAO2L8R, https://disc.gsfc.nasa.gov/datasets/M2IUNXASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_2d_ int_ Nx: 2d,Diurnal,Instantaneous,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.” doi:10.5067/DGAB3HFEYMLY, https://disc.gsfc.nasa.gov/datasets/M2IUNXINT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_2d_ lfo_ Nx: 2d,Diurnal,Instantaneous,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/FC3BVJ88Y8A2, https://disc.gsfc.nasa.gov/datasets/M2IUNXLFO_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_2d_ gas_ Nx: 2d,Diurnal,Instantaneous,Single-Level,Assimilation,Aerosol Optical Depth Analysis V5.12.4.” doi:10.5067/TVJ4MHBED39L, https://disc.gsfc.nasa.gov/datasets/M2IUNXGAS_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_3d_ asm_ Np: 3d,Diurnal,Instantaneous,Pressure-Level,Assimilation,Assimilated Meteorological Fields V5.12.4.” doi:10.5067/6EGRBNEBMIYS, https://disc.gsfc.nasa.gov/datasets/M2IUNPASM_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 instU_3d_ ana_ Np: 3d,Diurnal,Instantaneous,Pressure-Level,Analysis,Analyzed Meteorological Fields V5.12.4.” doi:10.5067/TRD91YO9S6E7, https://disc.gsfc.nasa.gov/datasets/M2IUNPANA_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ adg_ Nx: 2d,Diurnal,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics (extended) V5.12.4.” doi:10.5067/YZJJXZTFCX6B, https://disc.gsfc.nasa.gov/datasets/M2TUNXADG_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ aer_ Nx: 2d,Diurnal,Time-averaged,Single-Level,Assimilation,Aerosol Diagnostics V5.12.4.” doi:10.5067/KPUMVXFEQLA1, https://disc.gsfc.nasa.gov/datasets/M2TUNXAER_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ chm_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Carbon Monoxide and Ozone Diagnostics V5.12.4.” doi:10.5067/5KFZ6GXRHZKN, `https://disc.gsfc.nasa.gov/datasets/M2TUNXCHM_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ csp_ Nx: 2d,Diurnal,Time-averaged,Single-Level,Assimilation,COSP Satellite Simulator V5.12.4.” doi:10.5067/9PH5QU4CL9E8, `https://disc.gsfc.nasa.gov/datasets/M2TUNXCSP_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ flx_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Surface Flux Diagnostics V5.12.4.” doi:10.5067/LUHPNWAKYIO3, `https://disc.gsfc.nasa.gov/datasets/M2TUNXFLX_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ int_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Vertically Integrated Diagnostics V5.12.4.” doi:10.5067/R2MPVU4EOSWT, `https://disc.gsfc.nasa.gov/datasets/M2TUNXINT_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ lfo_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Land Surface Forcings V5.12.4.” doi:10.5067/BTSNKAJND3ME, `https://disc.gsfc.nasa.gov/datasets/M2TUNXLFO_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ lnd_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Land Surface Diagnostics V5.12.4.” doi:10.5067/W0J15047CF6N, `https://disc.gsfc.nasa.gov/datasets/M2TUNXLND_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ ocn_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Ocean Surface Diagnostics V5.12.4.” doi:10.5067/KLNAVGAX7J66, `https://disc.gsfc.nasa.gov/datasets/M2TUNXOCN_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ rad_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/4SDCJYK8P9QU, `https://disc.gsfc.nasa.gov/datasets/M2TUNXRAD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ slv_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Single-Level Diagnostics V5.12.4.” doi:10.5067/AFOK0TPEVQEK, `https://disc.gsfc.nasa.gov/datasets/M2TUNXSLV_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_2d_ glc_ Nx: 2d,Diurnal,Time-Averaged,Single-Level,Assimilation,Land Ice Surface Diagnostics V5.12.4.” doi:10.5067/7VUPQC736SWX, `https://disc.gsfc.nasa.gov/datasets/M2TUNXGLC_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ cld_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Cloud Diagnostics V5.12.4.” doi:10.5067/EPW7T5UO0C0N, `https://disc.gsfc.nasa.gov/datasets/M2TUNPCLD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ mst_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Moist Processes Diagnostics V5.12.4.” doi:10.5067/ZRSN0JU27DK2, `https://disc.gsfc.nasa.gov/datasets/M2TUNPMST_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ rad_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Radiation Diagnostics V5.12.4.” doi:10.5067/H140JMDOWB0Y, `https://disc.gsfc.nasa.gov/datasets/M2TUNPRAD_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ tdt_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Temperature Tendencies V5.12.4.” doi:10.5067/QPO9E5TPZ8OF, `https://disc.gsfc.nasa.gov/datasets/M2TUNPTDT_5.12.4/summary`.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ trb_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Turbulence Diagnostics V5.12.4.” doi:10.5067/2A99C60CG7WC, https://disc.gsfc.nasa.gov/datasets/M2TUNPTRB_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ udt_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Wind Tendencies V5.12.4.” doi:10.5067/DO715T7T5PG8, https://disc.gsfc.nasa.gov/datasets/M2TUNPUDT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ odt_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Ozone Tendencies V5.12.4.” doi:10.5067/M8OJ09GZP23E, https://disc.gsfc.nasa.gov/datasets/M2TUNPODT_5.12.4/summary.

Global Modeling And Assimilation Office, Pawson S (2015). “MERRA-2 tavgU_3d_ qdt_ Np: 3d,Diurnal,Time-Averaged,Pressure-Level,Assimilation,Moist Tendencies V5.12.4.” doi:10.5067/S8HJXIR0BFTS, https://disc.gsfc.nasa.gov/datasets/M2TUNPQDT_5.12.4/summary.

Examples

```
## Not run:
download_merra2(
  collection = "inst1_2d_int_Nx",
  date = "2024-01-01",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
)

## End(Not run)
```

`download_modis` *Download MODIS product files*

Description

Need maintenance for the directory path change in NASA EOSDIS. This function first retrieves the all hdf download links on a certain day, then only selects the relevant tiles from the retrieved links. Download is only done at the queried horizontal-vertical tile number combinations. An exception is MOD06_L2 product, which is produced every five minutes every day.

Usage

```
download_modis(
  product = c("MOD09GA", "MYD09GA", "MOD09GQ", "MYD09GQ", "MOD09A1", "MYD09A1",
    "MOD09Q1", "MYD09Q1", "MOD11A1", "MYD11A1", "MOD11A2", "MYD11A2", "MOD11B1",
    "MYD11B1", "MOD13A1", "MYD13A1", "MOD13A2", "MYD13A2", "MOD13A3", "MYD13A3",
    "MOD06_L2", "MCD19A2", "VNP46A2"),
  version = "61",
  horizontal_tiles = c(7, 13),
  vertical_tiles = c(3, 6),
  mod06_links = NULL,
  nasa_earth_data_token = NULL,
  date = c("2023-09-01", "2023-09-01"),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

`product`	character(1). One of `c("MOD09GA", "MOD11A1", "MOD06_L2", "MCD19A2", "MOD13A2", "VNP46A2")`
`version`	character(1). Default is `"61"`, meaning v061.
`horizontal_tiles`	integer(2). Horizontal tile numbers `c({start}, {end})`. Default is `c(7, 13)`.
`vertical_tiles`	integer(2). Vertical tile numbers `c({start}, {end})`. Default is `c(3, 6)`.
`mod06_links`	character(1). CSV file path to MOD06_L2 download links from NASA LAADS MOD06_L2. Default is `NULL`.
`nasa_earth_data_token`	character(1). Token for downloading data from NASA. Should be set before trying running the function.
`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"`). Note: ignored if `product == "MOD06_L2"`.
`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). Download data or only save wget commands.
`remove_command`	logical(1). Remove (`TRUE`) or keep (`FALSE`) the text file containing download commands.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

- For `hash = FALSE`, NULL
- For `hash = TRUE`, an `rlang::hash_file` character.
- HDF (.hdf) files will be stored in year/day_of_year sub-directories within `directory_to_save`.

Note

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

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://lpdaac.usgs.gov/products/mcd19a2v061/.

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://lpdaac.usgs.gov/products/mod09gav061/.

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://lpdaac.usgs.gov/products/mod11a1v061/.

Didan K (2021). "MODIS/Terra Vegetation Indices 16-Day L3 Global 1km SIN Grid V061." doi:10.5067/MODIS/MOD13A2.061, https://lpdaac.usgs.gov/products/mod13a2v061/.

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:
## NOTE: Examples are wrapped in `/dontrun{}` to avoid sharing sensitive
##       NASA EarthData tokden information.
# example with MOD09GA product
download_modis(
  product = "MOD09GA",
  version = "61",
  horizontal_tiles = c(8, 8),
  vertical_tiles = c(4, 4),
  date = "2024-01-01",
  nasa_earth_data_token = "./pathtotoken/token.txt",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE
)
# example with MOD06_L2 product
download_modis(
  product = "MOD06_L2",
  version = "61",
  horizontal_tiles = c(8, 8),
  vertical_tiles = c(4, 4),
  date = "2024-01-01",
  mod06_links =
    system.file(
      "extdata", "nasa", "LAADS_query.2024-08-02T12_49.csv",
      package = "amadeus"
    ),
  nasa_earth_data_token = "./pathtotoken/token.txt",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE
)
# example with VNP46A2 product
download_modis(
  product = "VNP46A2",
  version = "61",
  horizontal_tiles = c(8, 8),
  vertical_tiles = c(4, 4),
  date = "2024-01-01",
  nasa_earth_data_token = "./pathtotoken/token.txt",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,MOD09GA
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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.

Usage

```
download_narr(
  variables = NULL,
  year = c(2018, 2022),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

`variables`	character. Variable(s) name acronym. See List of Variables in NARR Files for variable names and acronym codes.
`year`	character(1 or 2). length of 4. Year or start/end years for downloading data.
`directory_to_save`	character(1). Directory(s) to save downloaded 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.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

- For `hash = FALSE`, NULL
- For `hash = TRUE`, an `rlang::hash_file` character.
- netCDF (.nc) files will be stored in `directory_to_save`.

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.

Author(s)

Mitchell Manware, Insang Song

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,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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 = system.file("extdata/cacert_gaftp_epa.pem", package = "amadeus"),
  certificate_url =
    "http://cacerts.digicert.com/DigiCertGlobalG2TLSRSASHA2562020CA1-1.crt",
  year = c(2017L, 2020L),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  hash = FALSE
)
```

Arguments

`epa_certificate_path`	character(1). Path to the certificate file for EPA DataCommons. Default is 'extdata/cacert_gaftp_epa.pem' under the package installation path.
`certificate_url`	character(1). URL to certificate file. See notes for details.
`year`	Available years of NEI data. Default is `c(2017L, 2020L)`.
`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.

`unzip`	logical(1). Unzip the downloaded zip 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`.

Value

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

Note

For EPA Data Commons certificate errors, follow the steps below:

1. Click Lock icon in the address bar at https://gaftp.epa.gov
2. Click Show Certificate
3. Access Details
4. Find URL with *.crt extension Currently we bundle the pre-downloaded crt and its PEM (which is accepted in wget command) file in ./inst/extdata. The instruction above is for certificate updates in the future.

Author(s)

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

## End(Not run)
```

`download_nlcd` *Download land cover data*

Description

The `download_nlcd()` function accesses and downloads land cover data from the Multi-Resolution Land Characteristics (MRLC) Consortium's National Land Cover Database (NLCD) products data base.

Usage

```
download_nlcd(
  collection = "Coterminous United States",
  year = 2021,
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  unzip = TRUE,
  remove_zip = FALSE,
  hash = FALSE
)
```

Arguments

`collection`	character(1). `"Coterminous United States"` or `"Alaska"`.
`year`	integer(1). Available years for Coterminous United States include `2001`, `2004`, `2006`, `2008`, `2011`, `2013`, `2016`, `2019`, and `2021`. Available years for Alaska include `2001`, `2011`, and `2016`.
`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 shapefiles ("/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.
`unzip`	logical(1). Unzip zip files. Default is `TRUE`.
`remove_zip`	logical(1). Remove zip files 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`.

Value

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

Author(s)

Mitchell Manware, Insang Song

References

Dewitz J (2023). "National Land Cover Database (NLCD) 2021 Products." doi:10.5066/P9JZ7AO3.
Dewitz J (2024). "National Land Cover Database (NLCD) 2019 Products (ver. 3.0, February 2024)." doi:10.5066/P9KZCM54.

Examples

```
## Not run:
download_nlcd(
  collection = "Coterminous United States",
  year = 2021,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## 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, v4.11.

Usage

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

Arguments

`data_resolution`	character(1). Available resolutions are 30 second (approx. 1 km), 2.5 minute (approx. 5 km), 15 minute (approx. 30 km), 30 minute (approx. 55 km), and 60 minute (approx. 110 km).
`data_format`	character(1). Individual year data can be downloaded as `"ASCII"` or `"GeoTIFF"`. "all" years is downloaded as `"netCDF"`.
`year`	character(1). Available years are `2000`, `2005`, `2010`, `2015`, and `2020`, or `"all"` for all years.
`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 shapefiles ("/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.
`unzip`	logical(1). Unzip zip files. Default is `TRUE`.
`remove_zip`	logical(1). Remove zip files 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`.

Value

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

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:
download_population(
  data_resolution = "30 second",
  data_format = "GeoTIFF",
  year = "2020",
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = TRUE,
  unzip = FALSE
)

## 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 = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

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

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 = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

variables	character(1). Variable(s) name(s). See TerraClimate Direct Downloads for variable names and acronym codes.
year	character(1 or 2). length of 4. Year or start/end years for downloading data.
directory_to_save	character(1). Directory(s) to save downloaded 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.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

- For `hash = FALSE`, NULL
- For `hash = TRUE`, an `rlang::hash_file` character.
- netCDF (.nc) files will be stored in a variable-specific folder within `directory_to_save`.

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 = "Precipitation",
  year = 2023,
  directory_to_save = tempdir(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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.

Usage

```
download_tri(
  year = c(2018L, 2022L),
  directory_to_save = NULL,
  acknowledgement = FALSE,
  download = FALSE,
  remove_command = FALSE,
  hash = FALSE
)
```

Arguments

`year`	character(1 or 2). length of 4. Year or start/end years for downloading data.
`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.
`hash`	logical(1). By setting `TRUE` the function will return an `rlang::hash_file()` hash character corresponding to the downloaded files. Default is `FALSE`.

Value

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

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(),
  acknowledgement = TRUE,
  download = FALSE, # NOTE: download skipped for examples,
  remove_command = 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

`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.

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 = "full",
  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_blackmarble_corners`
Process Black Marble corners

Description

Tile corner generator for Black Marble products.

Black Marble products are in HDF5 format and are read without georeference with typical R geospatial packages. This function generates a `data.frame` of corner coordinates for assignment.

Usage

```
process_blackmarble_corners(hrange = c(5, 11), vrange = c(3, 6))
```

Arguments

`hrange`	integer(2). Both should be in 0-35.
`vrange`	integer(2). Both should be in 0-17.

Value

`data.frame` with xmin, xmax, ymin, and ymax fields

Author(s)

Insang Song

References

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

Examples

```
process_blackmarble_corners(hrange = c(1, 2), vrange = c(1, 2))
```

`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", "koppen-geiger", "blackmarble",
   "koeppen-geiger", "koppen", "koeppen", "geos", "dummies", "gmted", "hms", "smoke",
   "sedac_population", "population", "sedac_groads", "groads", "roads", "nlcd", "tri",
     "narr", "nei", "ecoregions", "ecoregion", "merra", "merra2", "gridmet",
     "terraclimate", "huc", "cropscape", "cdl", "prism"),
  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`, 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_gmted: "gmted", "GMTED"
- process_aqs: "aqs", "AQS"
- 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"

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_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_flatten_sds` *Process MODIS layers*

Description

Aggregate layers in a sub-dataset in sinusoidal MODIS products.

Some MODIS products consist of multi-layer subdatasets. This function aggregates multiple layers into single layer `SpatRaster`. `fun_agg` is applied at overlapping cells.

Usage

```
process_flatten_sds(path = NULL, subdataset = NULL, fun_agg = "mean", ...)
```

Arguments

`path`	character(1). Full path to MODIS HDF4/HDF5 file. Direct sub-dataset access is supported, for example, HDF4_EOS:EOS_GRID:{filename}:{base_grid_information}:{sub-dataset}
`subdataset`	character(1). Exact or regular expression filter of sub-dataset. See process_modis_sds for details.
`fun_agg`	character(1). Function name to aggregate layers. Should be acceptable to terra::tapp.
`...`	Placeholders.

Value

a `SpatRaster` object

Note

HDF values are read as original without scaling. Users should consult MODIS product documentation to apply proper scaling factor for post-hoc adjustment. If users have no preliminary information about MODIS sub-datasets, consider running `terra::describe(__filename__, sds = TRUE)` to navigate the full list of sub-datasets in the input file then consult the documentation of MODIS product.

Author(s)

Insang Song

See Also

terra::tapp, terra::rast, terra::describe

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_flatten <- process_flatten_sds(
  path =
    list.files("./data", pattern = "MOD09GA.", full.names = TRUE)[1],
  subdataset = process_modis_sds("MOD09GA"),
  fun_agg = "mean"
)

## 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,
  ...
)
```

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).
`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
`...`	Placeholders.

Value

a `SpatRaster` object;

Note

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

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"
)

## 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_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://earthdata.nasa.gov/data/catalog/sedac-ciesin-sedac-groads-v1-1.00`.

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_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 composition data, returning a single `SpatRaster` object.

Usage

```
process_merra2(
  date = c("2018-01-01", "2018-01-10"),
  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). MERRA2 variable name(s).
`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
`...`	Placeholders.

Value

a `SpatRaster` object;

Note

Layer names of the returned `SpatRaster` object contain the variable, pressure level, date, and hour. 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"
)

## End(Not run)
```

`process_modis_merge` *Process MODIS .hdf files*

Description

Get mosaicked or merged raster from multiple MODIS hdf files.

Usage

```
process_modis_merge(
  path = NULL,
  date = NULL,
  subdataset = NULL,
  fun_agg = "mean",
  ...
)
```

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.
`...`	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_sds` *Process MODIS sub-datasets*

Description

Selected MODIS sinusoidal grid product subdataset name selector. Four presets are supported. `custom_sel` supersedes presets of `product` values.

Usage

```
process_modis_sds(
  product = c("MOD11A1", "MOD13A2", "MOD09GA", "MCD19A2"),
  custom_sel = NULL,
  ...
)
```

Arguments

`product`	character(1). Product code.
`custom_sel`	character(1). Custom filter. If this value is not NULL, preset filter is overridden.
`...`	Placeholders.

Value

A character object that conforms to the regular expression. Details of regular expression in R can be found in regexp.

Note

Preset product codes and associated variables include

- "MOD11A1" - Land surface temperature (LST)
- "MOD13A2" - Normalized Difference Vegetation Index (NDVI)
- "MOD09GA" - Surface reflectance, and
- "MCD19A2" - Aerosol optical depth (AOD).

For a full list of available MODIS product codes, see the "Short Name" column at NASA LP DAAC Search Data Catalog. When utilizing a product code from this "Short Name" column, **do not include** the version number following the period. For example, if "Short Name" = MCD12C1.006, then `product = "MCD12C1"`.

Author(s)

Insang Song

See Also

calculate_modis

Examples

```
process_modis_sds(product = "MOD09GA")
```

`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_modis_warp` *Warp MODIS swath data into rectilinear grid raster*

Description

Swath data is a type of MODIS data, where curvilinear points are stored with varying resolution depending on the relative position of the sensor axis. As this type of data typically does not work well with planar spatial data, users should warp or rectify this data into a rectilinear raster. Main procedure is done with `stars::st_warp`, in which users are able to customize the threshold to fill potential gaps that appear where the target resolution is finer than the local resolution of curvilinear grid points.

Usage

```
process_modis_warp(
  path = NULL,
  cellsize = 0.1,
  threshold = cellsize * 4,
  crs = 4326,
  ...
)
```

Arguments

`path`	File path of MODIS swath with exact sub-dataset specification.
`cellsize`	numeric(1). Cell size (spatial resolution) of output rectilinear grid raster.
`threshold`	numeric(1). Maximum distance to fill gaps if occur.
`crs`	integer(1)/character(1). Coordinate system definition. Should be compatible with EPSG codes or WKT2. See `terra::crs` and `sf::st_crs` / EPSG
`...`	For internal use.

Value

a `stars` object

Note

This function handles one file at a time.

Author(s)

Insang Song

See Also

`terra::rectify`

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_warp <- process_modis_warp(
  path = paste0(
    "HDF4_EOS:EOS_SWATH:",
    list.files(
      "./data/mod06l2",
      full.names = TRUE,
      pattern = ".hdf"
    )[1],
    ":mod06:Cloud_Fraction"
  ),
  cellsize = 0.1,
  threshold = 0.4,
  crs = 4326
)

## 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 = c(1, 13, 12, 14, 20, 34, 36, 47, 48, 49),
  extent = NULL,
  ...
)
```

Arguments

`path`	character(1). Path to the directory with TRI CSV files
`year`	integer(1). Single year to select.
`variables`	integer. Column index of TRI data.
`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

Visit TRI Data and Tools to view the available years and variables.

Author(s)

Insang Song, Mariana Kassien

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(1, 13, 12, 14, 20, 34, 36, 47, 48, 49)
)

## 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,
  sedc_bandwidth = NULL,
  target_fields = NULL,
  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`.
`sedc_bandwidth`	numeric(1). Distance at which the source concentration is reduced to `exp(-3)` (approximately -95 %)
`target_fields`	character(varying). Field names in characters.
`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, "sedc_bandwidth")“: the bandwidth 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

```
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)
```

Index