Package ‘epwshiftr’

October 13, 2022

Title Create Future 'EnergyPlus' Weather Files using 'CMIP6' Data

Version 0.1.3


Imports checkmate, data.table (>= 1.12.4), eplusr, future.apply, fst, jsonlite, progressr, psychrolib, rappdirs, RNetCDF, units

Suggests testthat, pingr, covr

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/ideas-lab-nus/epwshiftr

BugReports https://github.com/ideas-lab-nus/epwshiftr/issues

RoxygenNote 7.1.1

Collate 'coord.R' 'utils.R' 'epwshiftr-package.R' 'esgf.R' 'morph.R'

'netcdf.R'

NeedsCompilation no

Author Hongyuan Jia [aut, cre] (<https://orcid.org/0000-0002-0075-8183>), Adrian Chong [aut] (<https://orcid.org/0000-0002-9486-4728>)

Maintainer Hongyuan Jia <hongyuanjia@outlook.com>

Repository CRAN

Date/Publication 2021-05-26 13:10:02 UTC

R topics documented:

  epwshiftr-package ........................................................... 2
  esgf_query ................................................................. 3
  extract_data ............................................................... 7
epwshiftr-package

epwshiftr: Create future EnergyPlus Weather files using CMIP6 data

Description

Query, download climate change projection data from the CMIP6 (Coupled Model Intercomparison Project Phase 6) project in the ESGF (Earth System Grid Federation) platform, and create future EnergyPlus Weather (EPW) files adjusted from climate changes using data from Global Climate Models (GCM).

Package options

- epwshiftr.verbose: If TRUE, more detailed message will be printed. Default: FALSE.
- epwshiftr.dir: The directory to store package data, including CMIP6 model output file index and etc. If not set, the current user data directory will be used.

Author(s)

Hongyuan Jia

See Also

Useful links:

- https://github.com/ideas-lab-nus/epwshiftr
- Report bugs at https://github.com/ideas-lab-nus/epwshiftr/issues
**esgf_query**

*Query CMIP6 data using ESGF search RESTful API*

---

**Description**

Query CMIP6 data using ESGF search RESTful API

**Usage**

```r
esgf_query(
    activity = "ScenarioMIP",
    variable = c("tas", "tasmax", "tasmin", "hurs", "hursmax", "hursmin", "pr", "rsds", "rlds", "psl", "sfcWind", "clt"),
    frequency = "day",
    experiment = c("ssp126", "ssp245", "ssp370", "ssp585"),
    variant = "r1i1p1f1",
    replica = FALSE,
    latest = TRUE,
    resolution = c("100 km", "50 km"),
    type = "Dataset",
    limit = 10000L,
    data_node = NULL
)
```

**Arguments**

- **activity**: A character vector indicating activity identifiers. Default: "ScenarioMIP". Possible values:
- "AerChemMIP": Aerosols and Chemistry Model Intercomparison Project,
- "C4MIP": Coupled Climate Carbon Cycle Model Intercomparison Project,
- "CDRMIP": Carbon Dioxide Removal Model Intercomparison Project,
- "CFMIP": Cloud Feedback Model Intercomparison Project,
- "CMIP": CMIP DECK: 1pctCO2, abrupt4xCO2, amip, esm-piControl, esm-historical, historical, and piControl experiments,
- "CORDEX": Coordinated Regional Climate Downscaling Experiment,
- "DAMIP": Detection and Attribution Model Intercomparison Project,
- "DCPP": Decadal Climate Prediction Project,
- "DynVarMIP": Dynamics and Variability Model Intercomparison Project,
- "FAFMIP": Flux-Anomaly-Forced Model Intercomparison Project,
- "GMMIP": Global Monsoons Model Intercomparison Project,
- "GeoMIP": Geoengineering Model Intercomparison Project,
- "HighResMIP": High-Resolution Model Intercomparison Project,
variables
A character vector indicating variable identifiers. The 12 most related variables for EPW are set as defaults. If NULL, all possible variables are returned. Default: c("tas", "tasmax", "tasmin", "hurs", "hursmax", "hursmin", "psl", "rss", "rls", "sfcWind", "pr", "clt"). where:

- `tas`: Near-surface (usually, 2 meter) air temperature, units: K.
- `tasmax`: Maximum near-surface (usually, 2 meter) air temperature, units: K.
- `tasmin`: Minimum near-surface (usually, 2 meter) air temperature, units: K.
- `hurs`: Near-surface relative humidity, units: %.
- `hursmax`: Maximum near-surface relative humidity, units: %.
- `hursmin`: Minimum near-surface relative humidity, units: %.
- `psl`: Sea level pressure, units: Pa.
- `rss`: Surface downwelling shortwave radiation, units: W m^-2.
- `rls`: Surface downwelling longwave radiation, units: W m^-2.
- `sfcWind`: Near-surface (usually, 10 meters) wind speed, units: m s^-1.
- `pr`: Precipitation, units: kg m^-2 s^-1.
- `clt`: Total cloud area fraction for the whole atmospheric column, as seen from the surface or the top of the atmosphere. Units: %.

frequencies
A character vector of sampling frequency. If NULL, all possible frequencies are returned. Default: "day". Possible values:

- "1hr": sampled hourly,
- "1hrCM": monthly-mean diurnal cycle resolving each day into 1-hour means,
- "1hrPt": sampled hourly, at specified time point within an hour,
- "3hr": sampled every 3 hours,
- "3hrPt": sampled 3 hourly, at specified time point within the time period,
- "6hr": sampled every 6 hours,
- "6hrPt": sampled 6 hourly, at specified time point within the time period,
- "day": daily mean samples,
- "dec": decadal mean samples,
- "fx": fixed (time invariant) field,
• "mon": monthly mean samples,
• "monC": monthly climatology computed from monthly mean samples,
• "monPt": sampled monthly, at specified time point within the time period,
• "subhrPt": sampled sub-hourly, at specified time point within an hour,
• "yr": annual mean samples,
• "yrPt": sampled yearly, at specified time point within the time period

experiment A character vector indicating root experiment identifiers. The Tier-1 experiment of activity ScenarioMIP are set as defaults. If NULL, all possible experiment are returned. Default: c("ssp126", "ssp245", "ssp370", "ssp585").

source A character vector indicating model identifiers. Defaults are set to 11 sources which give outputs of all 4 experiment of activity ScenarioMIP with daily frequency, i.e. "AWI-CM-1-1-MR", "BCC-CSM2-MR", "CESM2", "CESM2-WACCM", "EC-Earth3", "EC-Earth3-Veg", "GFDL-ESM4", "INM-CM4-8", "INM-CMS-0", "MPI-ESM1-2-HR" and "MRI-ESM2-0". If NULL, all possible sources are returned.

variant A character vector indicating label constructed from 4 indices stored as global attributes in format r<k>i<l>p<m>f<n> described below. Default: "r1i1p1f1". If NULL, all possible variants are returned.

• r: realization_index (<k>) = realization number (integer >0)
• i: initialization_index (<l>) = index for variant of initialization method (integer >0)
• p: physics_index (<m>) = index for model physics variant (integer >0)
• f: forcing_index (<n>) = index for variant of forcing (integer >0)

replica Whether the record is the "master" copy, or a replica. Use FALSE to return only originals and TRUE to return only replicas. Default: FALSE.

latest Whether the record is the latest available version, or a previous version. Use TRUE to return only the latest version of all records and FALSE to return previous versions. Default: FALSE.

resolution A character vector indicating approximate horizontal resolution. Default: c("50 km", "100 km"). If NULL, all possible resolutions are returned.

type A single string indicating the intrinsic type of the record. Should be either "Dataset" or "File". Default: "Dataset".

limit An integer indicating the maximum of matched records to return. Should be <= 10,000. Default: 10000.

data_node A character vector indicating data nodes to be queried. Default to NULL, which means all possible data nodes.

Details

The Earth System Grid Federation (ESGF) is an international collaboration for the software that powers most global climate change research, notably assessments by the Intergovernmental Panel on Climate Change (IPCC).

The ESGF search service exposes a RESTful URL that can be used by clients to query the contents of the underlying search index, and return results matching the given constraints.ributed capabilities
of the ESGF search, the URL at any Index Node can be used to query that Node only, or all Nodes in the ESGF system. `esgf_query()` uses the LLNL (Lawrence Livermore National Laboratory) Index Node.

**Value**

A `data.table::data.table` with an attribute named `response` which is a list converted from json response. If no matched data is found, an empty `data.table` is returned. Otherwise, the columns of returned data varies based on the type:

- If "Dataset", returned columns are:

<table>
<thead>
<tr>
<th>No.</th>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dataset_id</td>
<td>Character</td>
<td>Dataset universal identifier</td>
</tr>
<tr>
<td>2</td>
<td>mip_era</td>
<td>Character</td>
<td>Activity’s associated CMIP cycle. Will always be “CMIP6”</td>
</tr>
<tr>
<td>3</td>
<td>activity_drs</td>
<td>Character</td>
<td>Activity DRS (Data Reference Syntax)</td>
</tr>
<tr>
<td>4</td>
<td>institution_id</td>
<td>Character</td>
<td>Institution identifier</td>
</tr>
<tr>
<td>5</td>
<td>source_id</td>
<td>Character</td>
<td>Model identifier</td>
</tr>
<tr>
<td>6</td>
<td>experiment_id</td>
<td>Character</td>
<td>Root experiment identifier</td>
</tr>
<tr>
<td>7</td>
<td>member_id</td>
<td>Character</td>
<td>A compound construction from sub_experiment_id and variant_label</td>
</tr>
<tr>
<td>8</td>
<td>table_id</td>
<td>Character</td>
<td>Table identifier, i.e. sampling frequency identifier</td>
</tr>
<tr>
<td>9</td>
<td>frequency</td>
<td>Character</td>
<td>Sampling frequency</td>
</tr>
<tr>
<td>10</td>
<td>grid_label</td>
<td>Character</td>
<td>Grid identifier</td>
</tr>
<tr>
<td>11</td>
<td>version</td>
<td>Character</td>
<td>Approximate date of model output file</td>
</tr>
<tr>
<td>12</td>
<td>nominal_resolution</td>
<td>Character</td>
<td>Approximate horizontal resolution</td>
</tr>
<tr>
<td>13</td>
<td>variable_id</td>
<td>Character</td>
<td>Variable identifier</td>
</tr>
<tr>
<td>14</td>
<td>variable_long_name</td>
<td>Character</td>
<td>Variable long name</td>
</tr>
<tr>
<td>15</td>
<td>variable_units</td>
<td>Character</td>
<td>Units of variable</td>
</tr>
<tr>
<td>16</td>
<td>data_node</td>
<td>Character</td>
<td>Data node to download the model output file</td>
</tr>
<tr>
<td>17</td>
<td>dataset_pid</td>
<td>Character</td>
<td>A unique string that helps identify the dataset</td>
</tr>
</tbody>
</table>

- If "File", returned columns are:

<table>
<thead>
<tr>
<th>No.</th>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>file_id</td>
<td>Character</td>
<td>Model output file universal identifier</td>
</tr>
<tr>
<td>2</td>
<td>dataset_id</td>
<td>Character</td>
<td>Dataset universal identifier</td>
</tr>
<tr>
<td>3</td>
<td>mip_era</td>
<td>Character</td>
<td>Activity’s associated CMIP cycle. Will always be “CMIP6”</td>
</tr>
<tr>
<td>4</td>
<td>activity_drs</td>
<td>Character</td>
<td>Activity DRS (Data Reference Syntax)</td>
</tr>
<tr>
<td>5</td>
<td>institution_id</td>
<td>Character</td>
<td>Institution identifier</td>
</tr>
<tr>
<td>6</td>
<td>source_id</td>
<td>Character</td>
<td>Model identifier</td>
</tr>
<tr>
<td>7</td>
<td>experiment_id</td>
<td>Character</td>
<td>Root experiment identifier</td>
</tr>
<tr>
<td>8</td>
<td>member_id</td>
<td>Character</td>
<td>A compound construction from sub_experiment_id and variant_label</td>
</tr>
<tr>
<td>9</td>
<td>table_id</td>
<td>Character</td>
<td>Table identifier, i.e. sampling frequency identifier</td>
</tr>
<tr>
<td>10</td>
<td>frequency</td>
<td>Character</td>
<td>Sampling frequency</td>
</tr>
<tr>
<td>11</td>
<td>grid_label</td>
<td>Character</td>
<td>Grid identifier</td>
</tr>
<tr>
<td>12</td>
<td>version</td>
<td>Character</td>
<td>Approximate date of model output file</td>
</tr>
<tr>
<td>13</td>
<td>nominal_resolution</td>
<td>Character</td>
<td>Approximate horizontal resolution</td>
</tr>
<tr>
<td>14</td>
<td>variable_id</td>
<td>Character</td>
<td>Variable identifier</td>
</tr>
</tbody>
</table>
extract_data

variable_long_name  Character  Variable long name
variable_units     Character  Units of variable
datetime_start    POSIXct  Start date and time of simulation
datetime_end      POSIXct  End date and time of simulation
file_size         Character  Model output file size in Bytes
data_node          Character  Data node to download the model output file
file_url           Character  Model output file download url from HTTP server
tracking_id        Character  A unique string that helps identify the output file

References

https://github.com/ESGF/esgf.github.io/wiki/ESGF_Search_REST_API

Examples

## Not run:
esgf_query(variable = "rss", experiment = "ssp126", resolution = "100 km", limit = 1)
esgf_query(variable = "rss", experiment = "ssp126", type = "File", limit = 1)

## End(Not run)

extract_data

Extract data

Description

extract_data() takes an epw_cmip6_coord object generated using match_coord() and extracts CMIP6 data using the coordinates and years of interest specified.

Usage

extract_data(
  coord,
  years = NULL,
  unit = FALSE,
  out_dir = NULL,
  by = NULL,
  keep = is.null(out_dir),
  compress = 100
)
Arguments

coord
An epw_cmip6_coord object created using `match_coord()`

years
An integer vector indicating the target years to be included in the data file. All other years will be excluded. If NULL, no subsetting on years will be performed. Default: NULL.

unit
If TRUE, units will be added to values using `units::set_units()`.

out_dir
The directory to save extracted data using `fst::write_fst()`. If NULL, all data will be kept in memory by default. Default: NULL.

by
A character vector of variable names used to split data during extraction. Should be a subset of:

- "experiment": root experiment identifiers
- "source": model identifiers
- "variable": variable identifiers
- "activity": activity identifiers
- "frequency": sampling frequency
- "variant": variant label
- "resolution": approximate horizontal resolution

If NULL and out_dir is given, file name `data.fst` will be used. Default: NULL.

keep
Whether keep extracted data in memory. Default: TRUE if out_dir is NULL, and FALSE otherwise.

compress
A single integer in the range 0 to 100, indicating the amount of compression to use. Lower values mean larger file sizes. Default: 100.

Details

`extract_data()` uses `future.apply` underneath. You can use your preferable future backend to speed up data extraction in parallel. By default, `extract_data()` uses `future::sequential` backend, which runs things in sequential.

Value

An epw_cmip6_data object, which is basically a list of 3 elements:

- epw: An eplusr::Epw object whose longitude and latitude are used to extract CMIP6 data. It is the same object as created in `match_coord()`
- meta: A list containing basic meta data of input EPW, including city, state_province, country, latitude and longitude.
- data: An empty `data.table::data.table()` if keep is FALSE or a `data.table::data.table()` of 12 columns if keep is TRUE:

<table>
<thead>
<tr>
<th>No.</th>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>activity_drs</td>
<td>Character</td>
<td>Activity DRS (Data Reference Syntax)</td>
</tr>
<tr>
<td>2</td>
<td>institution_id</td>
<td>Character</td>
<td>Institution identifier</td>
</tr>
<tr>
<td>3</td>
<td>source_id</td>
<td>Character</td>
<td>Model identifier</td>
</tr>
<tr>
<td>4</td>
<td>experiment_id</td>
<td>Character</td>
<td>Root experiment identifier</td>
</tr>
</tbody>
</table>
### future_epw

Create future EPW files using morphed data

#### Description

Create future EPW files using morphed data

#### Usage

```r
future_epw(
  morphed,
  by = c("experiment", "source", "interval"),
  dir = ".",
  separate = TRUE,
  overwrite = FALSE
)
```

#### Arguments

- **morphed**: An epw_cmip6_morphed object created using `morphing_epw()`.  
- **by**: A character vector of columns to be used as grouping variables when creating EPW files. Should be a sub-set of:
  - "experiment": root experiment identifiers  
  - "source": model identifiers  
  - "variable": variable identifiers  
  - "activity": activity identifiers  
  - "frequency": sampling frequency  
  - "variant": variant label
get_data_dir

- "resolution": approximate horizontal resolution
- "longitude": averaged longitude of input data
- "latitude": averaged latitude of input data

dir
The parent directory to save the generated EPW files. If not exist, it will be created first. Default: ".", i.e., current working directory.

separate
If TRUE, each EPW file will be saved into a separate folder using grouping variables specified in by.

overwrite
If TRUE, overwrite existing files if they exist. Default: FALSE.

Value
A list of generated eplusr::Epw objects, invisibly

Examples
```r
options(epwshiftr.dir = tempdir())
get_data_dir()
```

---

**init_cmip6_index**

*Build CMIP6 experiment output file index*

**Description**

init_cmip6_index() will search the CMIP6 model output file using esgf_query(), return a data.table::data.table() containing the actual NetCDF file url to download, and store it into user data directory for future use.

---

**get_data_node**

*Get data nodes which store CMIP6 output*

**Description**

Get data nodes which store CMIP6 output

**Usage**

get_data_node(speed_test = FALSE, timeout = 3)

**Arguments**

- **speed_test**: If TRUE, use pingr::ping() to perform connection speed test on each data node. A ping column is appended in returned data.table which stores each data node response in milliseconds. This feature needs pingr package already installed. Default: FALSE.
- **timeout**: Timeout for a ping response in seconds. Default: 3.

**Value**

A data.table::data.table() of 2 or 3 (when speed_test is TRUE) columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_node</td>
<td>character</td>
<td>Web address of data node</td>
</tr>
<tr>
<td>status</td>
<td>character</td>
<td>Status of data node. &quot;UP&quot; means OK and &quot;DOWN&quot; means currently not available</td>
</tr>
<tr>
<td>ping</td>
<td>double</td>
<td>Data node response in milliseconds during speed test</td>
</tr>
</tbody>
</table>

**Examples**

get_data_node()
Usage

```
init_cmip6_index(
  activity = "ScenarioMIP",
  variable = c("tas", "tasmax", "tasmin", "hurs", "hursmax", "hursmin", "pr", "rlds", "psl", "sfcWind", "clt"),
  frequency = "day",
  experiment = c("ssp126", "ssp245", "ssp370", "ssp585"),
             "EC-Earth3-Veg", "GFDL-ESM4", "INM-CM4-8", "INM-CM5-0", "MPI-ESM1-2-HR",
             "MRI-ESM2-0"),
  variant = "r1i1p1f1",
  replica = FALSE,
  latest = TRUE,
  resolution = c("100 km", "50 km"),
  limit = 10000L,
  data_node = NULL,
  years = NULL,
  save = FALSE
)
```

Arguments

- **activity**: A character vector indicating activity identifiers. Default: "ScenarioMIP". Possible values:
  - "AerChemMIP": Aerosols and Chemistry Model Intercomparison Project,
  - "C4MIP": Coupled Climate Carbon Cycle Model Intercomparison Project,
  - "CDRMIP": Carbon Dioxide Removal Model Intercomparison Project,
  - "CFMIP": Cloud Feedback Model Intercomparison Project,
  - "CMIP": CMIP DECK: 1pctCO2, abrupt4xCO2, amip, esm-piControl, esm-historical, historical, and piControl experiments,
  - "CORDEX": Coordinated Regional Climate Downscaling Experiment,
  - "DAMIP": Detection and Attribution Model Intercomparison Project,
  - "DCPP": Decadal Climate Prediction Project,
  - "DynVarMIP": Dynamics and Variability Model Intercomparison Project,
  - "FAFMIP": Flux-Anomaly-Forced Model Intercomparison Project,
  - "GMMIP": Global Monsoons Model Intercomparison Project,
  - "GeoMIP": Geoengineering Model Intercomparison Project,
  - "HighResMIP": High-Resolution Model Intercomparison Project,
  - "ISMIP6": Ice Sheet Model Intercomparison Project for CMIP6,
  - "LS3MIP": Land Surface, Snow and Soil Moisture,
  - "LUMIP": Land-Use Model Intercomparison Project,
  - "OMIP": Ocean Model Intercomparison Project,
  - "PAMIP": Polar Amplification Model Intercomparison Project,
  - "PMIP": Palaeoclimate Modelling Intercomparison Project,
  - "RFMIP": Radiative Forcing Model Intercomparison Project,
variable
A character vector indicating variable identifiers. The 12 most related variables for EPW are set as defaults. If NULL, all possible variables are returned. Default: c("tas", "tasmax", "tasmin", "hurs", "hursmax", "hursmin", "psl", "rss", "rls", "sfcWind", "pr", "clt"), where:

- tas: Near-surface (usually, 2 meter) air temperature, units: K.
- tasmax: Maximum near-surface (usually, 2 meter) air temperature, units: K.
- tasmin: Minimum near-surface (usually, 2 meter) air temperature, units: K.
- hurs: Near-surface relative humidity, units: %.
- hursmax: Maximum near-surface relative humidity, units: %.
- hursmin: Minimum near-surface relative humidity, units: %.
- psl: Sea level pressure, units: Pa.
- rsds: Surface downwelling shortwave radiation, units: W m\textsuperscript{-2}.
- rlds: Surface downwelling longwave radiation, units: W m\textsuperscript{-2}.
- sfcWind: Near-surface (usually, 10 meters) wind speed, units: m s\textsuperscript{-1}.
- pr: Precipitation, units: kg m\textsuperscript{-2} s\textsuperscript{-1}.
- clt: Total cloud area fraction for the whole atmospheric column, as seen from the surface or the top of the atmosphere. Units: %.

frequency
A character vector of sampling frequency. If NULL, all possible frequencies are returned. Default: "day". Possible values:

- "1hr": sampled hourly,
- "1hrCM": monthly-mean diurnal cycle resolving each day into 1-hour means,
- "1hrPt": sampled hourly, at specified time point within an hour,
- "3hr": sampled every 3 hours,
- "3hrPt": sampled 3 hourly, at specified time point within the time period,
- "6hr": sampled every 6 hours,
- "6hrPt": sampled 6 hourly, at specified time point within the time period,
- "day": daily mean samples,
- "dec": decadal mean samples,
- "fx": fixed (time invariant) field,
- "mon": monthly mean samples,
- "monC": monthly climatology computed from monthly mean samples,
- "monPt": sampled monthly, at specified time point within the time period,
- "subhrPt": sampled sub-hourly, at specified time point within an hour,
- "yr": annual mean samples,
- "yrPt": sampled yearly, at specified time point within the time period

experiment
A character vector indicating root experiment identifiers. The **Tier-1** experiment of activity ScenarioMIP are set as defaults. If NULL, all possible experiment are returned. Default: c("ssp126", "ssp245", "ssp370", "ssp585").
source

A character vector indicating model identifiers. Defaults are set to 11 sources which give outputs of all 4 experiment of activity ScenarioMIP with daily frequency, i.e. “AWI-CM-1-1-MR”, “BCC-CSM2-1-MR”, “CESM2”, “CESM2-WACCM”, “EC-Earth3”, “EC-Earth3-Veg”, “GFDL-ESM4”, “INM-CM4-8”, “INM-CMS-0”, “MPI-ESM1-2-HR” and “MRI-ESM2-0”. If NULL, all possible sources are returned.

variant

A character vector indicating label constructed from 4 indices stored as global attributes in format \( r<0>i<1>p<0>f<n> \) described below. Default: “r1i1p1f1”. If NULL, all possible variants are returned.

- \( r \): realization_index \((<k>) \) = realization number \((integer >0)\)
- \( i \): initialization_index \((<l>) \) = index for variant of initialization method \((integer >0)\)
- \( p \): physics_index \((<m>) \) = index for model physics variant \((integer >0)\)
- \( f \): forcing_index \((<n>) \) = index for variant of forcing \((integer >0)\)

replica

Whether the record is the "master" copy, or a replica. Use FALSE to return only originals and TRUE to return only replicas. Default: FALSE.

latest

Whether the record is the latest available version, or a previous version. Use TRUE to return only the latest version of all records and FALSE to return previous versions. Default: FALSE.

resolution

A character vector indicating approximate horizontal resolution. Default: c("50 km", "100 km"). If NULL, all possible resolutions are returned.

limit

An integer indicating the maximum of matched records to return. Should be <= 10,000. Default: 10000.

data_node

A character vector indicating data nodes to be queried. Default to NULL, which means all possible data nodes.

years

An integer vector indicating the target years to be include in the data file. All other years will be excluded. If NULL, no subsetting on years will be performed. Default: NULL.

save

If TRUE, the results will be saved into user data directory. Default: FALSE.

Details

For details on where the file index is stored, see rappdirs::user_data_dir().

Value

A data.table::data.table with 22 columns:

<table>
<thead>
<tr>
<th>No.</th>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>file_id</td>
<td>Character</td>
<td>Model output file universal identifier</td>
</tr>
<tr>
<td>2</td>
<td>dataset_id</td>
<td>Character</td>
<td>Dataset universal identifier</td>
</tr>
<tr>
<td>3</td>
<td>mip_era</td>
<td>Character</td>
<td>Activity’s associated CMIP cycle. Will always be “CMIP6”</td>
</tr>
<tr>
<td>4</td>
<td>activity_drs</td>
<td>Character</td>
<td>Activity DRS (Data Reference Syntax)</td>
</tr>
<tr>
<td>5</td>
<td>institution_id</td>
<td>Character</td>
<td>Institution identifier</td>
</tr>
<tr>
<td>6</td>
<td>source_id</td>
<td>Character</td>
<td>Model identifier</td>
</tr>
<tr>
<td>7</td>
<td>experiment_id</td>
<td>Character</td>
<td>Root experiment identifier</td>
</tr>
</tbody>
</table>
load_cmip6_index

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>member_id</td>
<td>Character</td>
<td>A compound construction from sub_experiment_id and variant_label</td>
</tr>
<tr>
<td>table_id</td>
<td>Character</td>
<td>Table identifier</td>
</tr>
<tr>
<td>frequency</td>
<td>Character</td>
<td>Sampling frequency</td>
</tr>
<tr>
<td>grid_label</td>
<td>Character</td>
<td>Grid identifier</td>
</tr>
<tr>
<td>version</td>
<td>Character</td>
<td>Approximate date of model output file</td>
</tr>
<tr>
<td>nominal_resolution</td>
<td>Character</td>
<td>Approximate horizontal resolution</td>
</tr>
<tr>
<td>variable_id</td>
<td>Character</td>
<td>Variable identifier</td>
</tr>
<tr>
<td>variable_long_name</td>
<td>Character</td>
<td>Variable long name</td>
</tr>
<tr>
<td>variable_units</td>
<td>Character</td>
<td>Units of variable</td>
</tr>
<tr>
<td>datetime_start</td>
<td>POSIXct</td>
<td>Start date and time of simulation</td>
</tr>
<tr>
<td>datetime_end</td>
<td>POSIXct</td>
<td>End date and time of simulation</td>
</tr>
<tr>
<td>file_size</td>
<td>Character</td>
<td>Model output file size in Bytes</td>
</tr>
<tr>
<td>data_node</td>
<td>Character</td>
<td>Data node to download the model output file</td>
</tr>
<tr>
<td>dataset_pid</td>
<td>Character</td>
<td>A unique string that helps identify the dataset</td>
</tr>
<tr>
<td>tracking_id</td>
<td>Character</td>
<td>A unique string that helps identify the output file</td>
</tr>
</tbody>
</table>

Note

Argument `limit` will only apply to Dataset query. `init_cmip6_index()` will try to get all model output files which match the dataset id.

Examples

```r
## Not run:
init_cmip6_index()

## End(Not run)
```

load_cmip6_index Load previously stored CMIP6 experiment output file index

Description

Load previously stored CMIP6 experiment output file index

Usage

```r
load_cmip6_index(force = FALSE)
```

Arguments

- `force` If `TRUE`, read the index file. Otherwise, return the cached index if exists. Default: FALSE.

Value

A `data.table::data.table` with 20 columns. For detail description on column, see `init_cmip6_index()`.
match_coord

Match coordinates of input EPW in the CMIP6 output file database

Description

match_coord() takes an EPW and uses its longitude and latitude to match corresponding values that meet specified threshold in NetCDF files.

Usage

match_coord(epw, threshold = list(lon = 1, lat = 1), max_num = NULL)

Arguments

epw

Possible values:

• A file path of EPW file
• An eplusr::Epw object
• A regular expression used to search locations in EnergyPlus Weather Database, e.g. "los angeles.*tmy3". You will be asked to select a matched EPW to download and read. It will be saved into tempdir(). Note that the search is case-insensitive

threshold

A list of 2 elements lon and lat specifying the absolute distance threshold used when matching longitude and latitude. Default: list(lon = 1.0, lat = 1.0)

max_num

The maximum number to be matched for both longitude and latitude when threshold is matched. Default is NULL, which means no limit

Details

match_coord() uses future.apply underneath. You can use your preferable future backend to speed up data extraction in parallel. By default, match_coord() uses future::sequential backend, which runs things in sequential.

Value

An epw_cmip6_coord object, which is basically a list of 3 elements:

• epw: An eplusr::Epw object parsed from input epw argument
• meta: A list containing basic meta data of input EPW, including city, state_province, country, latitute and longitude.
morphing_epw

- coord: A `data.table::data.table()` which is basically CMIP6 index database with an appending new list column coord that contains matched latitudes and longitudes in each NetCDF file. Each element in coord contains 2 elements `lat` and `lon`, in which contains the 4 components describing the matched coordinates.
  - index: the indices of matched coordinates
  - value: the actual longitude or latitude in the NetCDF coordinate grids
  - dis: the distance between the coordinate values in NetCDF and input EPW
  - which: The value indices of longitude or latitude in the NetCDF coordinate grids. These values are used to extract the corresponding variable values

Examples

```r
# Not run:
# download an EPW from EnergyPlus website
epw <- eplusr::download_weather("los angeles.*TMY3", dir = tempdir(),
    type = "EPW", ask = FALSE)

match_coord(epw, threshold = list(lon = 1.0, lat = 1.0))
# End(Not run)
```

morphing_epw

**Morphing EPW weather variables**

**Description**

`morphing_epw()` takes an `epw_cmip6_data` object generated using `extract_data()` and calculates future core EPW weather variables using Morphing Method.

**Usage**

`morphing_epw(data, years = NULL, labels = NULL, methods = NULL)`

**Arguments**

- `data`: An `epw_cmip6_data` object generated using `extract_data()`.
- `years`: An integer vector indicating the target years to be considered. If NULL, all years in input data will be considered. Default: NULL.
- `labels`: A character or factor vector used for grouping input years. Usually are the outputs of `base::cut()`. labels should have the same length as years. If given, climate data of years grouped by labels will be averaged. Default: NULL.
- `methods`: A named character giving the methods of morphing procedures of each variables. Possible variable names are `tdb`, `rh`, `p`, `hor_ir`, `glob_rad`, `wind`. Possible values are: "stretch", "shift" and "combined". For example: `c(tdb = "stretch", rh = "shift")". "combined" is only applicable to tdb. The default morphing method for each variable is listed in the `Return` section. If NULL, the default methods will be used. Default: NULL.
Details

The EPW weather variables that get morphed are listed in details.

Value

An `epw_cmip6_morphed` object, which is basically a list of 12 elements:

<table>
<thead>
<tr>
<th>No.</th>
<th>Element</th>
<th>Type</th>
<th>Morphing Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>epw</code></td>
<td>eplusr::Epw</td>
<td>N/A</td>
<td>The original EPW file used for morphing</td>
</tr>
<tr>
<td>2</td>
<td><code>tdb</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of dry-bulb temperature after morphing</td>
</tr>
<tr>
<td>3</td>
<td><code>tdew</code></td>
<td>data.table::data.table()</td>
<td>Derived</td>
<td>Data of dew-point temperature after morphing</td>
</tr>
<tr>
<td>4</td>
<td><code>rh</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of relative humidity after morphing</td>
</tr>
<tr>
<td>5</td>
<td><code>p</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of atmospheric pressure after morphing</td>
</tr>
<tr>
<td>6</td>
<td><code>hor_ir</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of horizontal infrared radiation from the sky after morphing</td>
</tr>
<tr>
<td>7</td>
<td><code>glob_rad</code></td>
<td>data.table::data.table()</td>
<td>Derived</td>
<td>Data of global horizontal radiation after morphing</td>
</tr>
<tr>
<td>8</td>
<td><code>norm_rad</code></td>
<td>data.table::data.table()</td>
<td>Derived</td>
<td>Data of direct normal radiation after morphing</td>
</tr>
<tr>
<td>9</td>
<td><code>diff_rad</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of diffuse horizontal radiation after morphing</td>
</tr>
<tr>
<td>10</td>
<td><code>wind</code></td>
<td>data.table::data.table()</td>
<td>Stretch</td>
<td>Data of wind speed after morphing</td>
</tr>
<tr>
<td>11</td>
<td><code>total_cover</code></td>
<td>data.table::data.table()</td>
<td>Derived</td>
<td>Data of total sky cover after morphing</td>
</tr>
<tr>
<td>12</td>
<td><code>opaque_cover</code></td>
<td>data.table::data.table()</td>
<td>Derived</td>
<td>Data of opaque sky cover after morphing</td>
</tr>
</tbody>
</table>

Each `data.table::data.table()` listed above contains x columns

<table>
<thead>
<tr>
<th>No.</th>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>activity_drs</code></td>
<td>Character</td>
<td>Activity DRS (Data Reference Syntax)</td>
</tr>
<tr>
<td>2</td>
<td><code>institution_id</code></td>
<td>Character</td>
<td>Institution identifier</td>
</tr>
<tr>
<td>3</td>
<td><code>source_id</code></td>
<td>Character</td>
<td>Model identifier</td>
</tr>
<tr>
<td>4</td>
<td><code>experiment_id</code></td>
<td>Character</td>
<td>Root experiment identifier</td>
</tr>
<tr>
<td>5</td>
<td><code>member_id</code></td>
<td>Character</td>
<td>A compound construction from sub_experiment_id and variant_label</td>
</tr>
<tr>
<td>6</td>
<td><code>table_id</code></td>
<td>Character</td>
<td>Table identifier</td>
</tr>
<tr>
<td>7</td>
<td><code>lat</code></td>
<td>Double</td>
<td>The averaged values of input latitude</td>
</tr>
<tr>
<td>8</td>
<td><code>lon</code></td>
<td>Double</td>
<td>The averaged values of input longitude</td>
</tr>
<tr>
<td>9</td>
<td><code>interval</code></td>
<td>Factor</td>
<td>The year value of data morphed</td>
</tr>
<tr>
<td>10</td>
<td><code>Variable Name</code></td>
<td>Double</td>
<td>The morphed data, where <code>Variable Name</code> is the corresponding EPW weather variable</td>
</tr>
<tr>
<td>11</td>
<td><code>delta</code></td>
<td>Double</td>
<td>The shift factor. Will be <code>NA</code> for derived values</td>
</tr>
<tr>
<td>12</td>
<td><code>alpha</code></td>
<td>Double</td>
<td>The stretch factor. Will be <code>NA</code> for derived values</td>
</tr>
</tbody>
</table>

References

set_cmip6_index() takes a `data.table::data.table()` as input and set it as current index.

**Usage**

```r
set_cmip6_index(index, save = FALSE)
```

**Arguments**

- `index`: A `data.table::data.table()` containing the same column names and types as the output of `init_cmip6_index()`.
- `save`: If TRUE, besides loaded index, the index file saved to data directory will be also updated. Default: FALSE.

**Details**

`set_cmip6_index()` is useful when `init_cmip6_index()` may give you too much cases of which only some are of interest.

**Value**

A `data.table::data.table()`.

---

```r
summary_database() scan the directory specified and returns a `data.table()` containing summary information about all the CMIP6 files available against the output file index loaded using `load_cmip6_index()`.

**Usage**

```r
summary_database(
  dir,
  by = c("activity", "experiment", "variant", "frequency", "variable", "source", "resolution"),
  mult = c("skip", "latest"),
  append = FALSE,
  recursive = FALSE,
  update = FALSE,
  warning = TRUE
)
```
Arguments

dir
A single string indicating the directory where CMIP6 model output NetCDF files are stored.

by
The grouping column to summary the database status. Should be a subset of:
- "experiment": root experiment identifiers
- "source": model identifiers
- "variable": variable identifiers
- "activity": activity identifiers
- "frequency": sampling frequency
- "variant": variant label
- "resolution": approximate horizontal resolution

mult
Actions when multiple files match a same case in the CMIP6 index. If "latest", the file with latest modification time will be used. If "skip", all matched files will be skip and this case will be kept as unmatched. Default: "skip".

append
If TRUE, status of CMIP6 files will only be updated if they are not found in previous summary. This is useful if CMIP6 files are stored in different directories. Default: FALSE.

recursive
If TRUE, scan recursively into directories. Default: FALSE.

update
If TRUE, the output file index will be updated based on the matched NetCDF files in specified directory. If FALSE, only current loaded index will be updated, but the actual index database file saved in \texttt{get_data_dir()} will remain unchanged. Default: FALSE.

warning
If TRUE, warning messages will show when multiple files match a same case. Default: TRUE.

Details

\texttt{summary_database()} uses \texttt{future.apply} underneath. You can use your preferable future backend to speed up data extraction in parallel. By default, \texttt{summary_database()} uses \texttt{future::sequential} backend, which runs things in sequential.

Value

A \texttt{data.table::data.table()} containing corresponding grouping columns plus:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datetime_start</td>
<td>POSIXct</td>
<td>Start date and time of simulation</td>
</tr>
<tr>
<td>datetime_end</td>
<td>POSIXct</td>
<td>End date and time of simulation</td>
</tr>
<tr>
<td>file_num</td>
<td>Integer</td>
<td>Total number of file per group</td>
</tr>
<tr>
<td>file_size</td>
<td>Units (Mbytes)</td>
<td>Approximate total size of file</td>
</tr>
<tr>
<td>dl_num</td>
<td>Integer</td>
<td>Total number of file downloaded</td>
</tr>
<tr>
<td>dl_percent</td>
<td>Units (%)</td>
<td>Total percentage of file downloaded</td>
</tr>
<tr>
<td>dl_size</td>
<td>Units (Mbytes)</td>
<td>Total size of file downloaded</td>
</tr>
</tbody>
</table>
Also an attribute `not_matched` is added to the returned `data.table::data.table()` which contains meta data for those CMIP6 output files that are not covered by current CMIP6 output file index.

For the meaning of grouping columns, see `init_cmip6_index()`.

Examples

```r
## Not run:
summary_database()

summary_database(by = "experiment")

## End(Not run)
```
Index

base::cut, 17

data.table(), 19
data.table::data.table, 6, 14, 15
data.table::data.table(), 8, 11, 17–21

eplusr::Epw, 8, 10, 16, 18
epwhiftr (epwhiftr-package), 2
epwhiftr-package, 2
esgf_query, 3
esgf_query(), 11
extract_data, 7
extract_data(), 17

dst::write_fst(), 8
future.apply, 8, 16, 20
future_epw, 9

get_data_dir, 10
get_data_dir(), 20
get_data_node, 11

init_cmip6_index, 11
init_cmip6_index(), 15, 19, 21

load_cmip6_index, 15
load_cmip6_index(), 19

match_coord, 16
match_coord(), 7, 8
morphing_epw, 17
morphing_epw(), 9

pingr::ping(), 11
rappdirs::user_data_dir(), 10, 14

set_cmip6_index, 18
summary_database, 19

tempdir(), 16

units::set_units, 8