Package: lpjmlstats (via r-universe)

January 10, 2025

Title Statistical tools for LPJmL data analysis

Version 0.6.0

Description This package provides statistical tools for LPJmL data analysis to be used for benchmarking LPJmL outputs.

License AGPL-3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Encoding UTF-8

URL https://github.com/PIK-LPJmL/lpjmlstats

BugReports https://github.com/PIK-LPJmL/lpjmlstats/issues

Imports dplyr, units, lpjmlkit, Matrix, methods, memoise, rlang, jsonlite, stringr, rmarkdown, knitr, colorspace, ggplot2, kableExtra, patchwork, tibble, cli, raster, tidyselect, R6, rnaturalearth

Suggests testthat (>= 3.0.0), maps, sf, covr

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 3.5.0)

Date 2024-11-11

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev make libicu-dev libpng-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://pik-piam.r-universe.dev

RemoteUrl https://github.com/PIK-LPJmL/lpjmlstats

RemoteRef HEAD

RemoteSha dfe2b257c5d93f4fc7f827cc6f78c0c8aa2da507

Contents

*.LPJmLDataCalc *Multiplication of two LPJmLDataCalc objects*

Description

Multiply an LPJmLDataCalc object by another LPJmLDataCalc object

Usage

```
## S3 method for class 'LPJmLDataCalc'
o1 * o2
```

Arguments

o1	An LPJmLDataCalc object.
o2	An LPJmLDataCalc object.

Value

An LPJmLDataCalc object.

+.LPJmLDataCalc *Addition of two LPJmLDataCalc objects*

Description

Add an LPJmLDataCalc object to another LPJmLDataCalc object

Usage

```
## S3 method for class 'LPJmLDataCalc'
o1 + o2
```

Arguments

o1	An LPJmLDataCalc object.
o2	An LPJmLDataCalc object.

Value

An LPJmLDataCalc object.

-.LPJmLDataCalc *Subtraction of two LPJmLDataCalc objects*

Description

Subtract an LPJmLDataCalc object from another LPJmLDataCalc object

Usage

```
## S3 method for class 'LPJmLDataCalc'
o1 - o2
```

Arguments

o1	An LPJmLDataCalc object.
o2	An LPJmLDataCalc object.

Value

An LPJmLDataCalc object.

.as_LPJmLDataCalc *Coerce an LPJmLData object into an LPJmLDataCalc object*

Description

Function to coerce (convert) an LPJmLData object into an LPJmLDataCalc object with extended functionality.

Usage

```
.as_LPJmLDataCalc(obj)
```

Arguments

obj	LPJmLData object or an array with the following order of dimensions: 1. space, 2. time, 3. band.

Value

An LPJmLDataCalc object.

/.LPJmLDataCalc *Division of two LPJmLDataCalc objects*

Description

Divide an LPJmLDataCalc object by another LPJmLDataCalc object

Usage

```
## S3 method for class 'LPJmLDataCalc'
o1 / o2
```

Arguments

o1	An LPJmLDataCalc object.
o2	An LPJmLDataCalc object.

Value

An LPJmLDataCalc object.

aggregate *Aggregate an LPJmLDataCalc object*

Description

Function to aggregate the full data of an LPJmLDataCalc object by applying summary statistics along the cell and/or time dimensions.

Usage

```
aggregate(x, ref_area = "terr_area", ...)
```

Arguments

x	LPJmLDataCalc object to be aggregated.
ref_area	string either terr_area or cell_area. Specifies the reference area to be used as a multiplier for the weighted_sum and weighted_mean aggregation methods. Should be the area of each cell on which the value "lives", assuming it has the given value only on that area and the value zero elsewhere (see mathematical support).

`...` one or several key-value pairs. Keys represent the dimension to be aggregated and values specify the target aggregation units and the desired summary statistic.

Aggregation unit and statistic are given in a list, by the syntax `list(to = [aggregation unit], stat =`

If only a string is given instead of a list it is used as the aggregation unit and the summary statistic defualts to `mean` for time and `weighted_sum` for cell.

Options for the cell dimension

The aggregation units for the cell dimension can be either an LPJmlRegionData object or a string with the following options

- `countries`: The regions defined in the countries of the world file.
- `global`: A dynamically created region that fully contains all cells of the grid. The aggregation method for space has the following options:
- `sum`: The values of all cells belonging to each region are summed up. If a cell belongs to a region only partially, we assume that the quantity is distributed uniformly over the cell area and multiply the value by the fraction of the cell that is part of the region before summing up.
- `mean`: First sums up the values of all cells belonging to each region as described for `sum` and then divides by the number of cells belonging to the region. Again we account for partial belonging of cells to regions (if it exists) by only counting the fraction of the cell that is part of the region in the divisor.
- `weighted_sum`: Similar to the `sum` option but multiplies the value of each cell by a reference area before summing up. The reference area default is the `terr_area` output which needs to exist in the same directory as the output to be aggregated. Other reference areas can be specified by setting the `reference_area` parameter.
- `weighted_mean`: Similar to the `mean` option but multiplies the value of each cell by a reference area before summing up. Also, the resulting sum is then divided by the total reference area of each region instead of the number of cells.

Options for the time dimension

For the time dimension these aggregation units are available:

- `sim_period`: The full simulation period.
- `years`: Aggregates the data to annual values.

The only available aggregation method is `mean` which takes the unweighted mean of the values.

Value

An aggregated `LPJmLDataCalc` object.

Examples

```
## Not run:
# Example 1
# Load an example LPJmLDataCalc object
soiln <- load_soiln_calc()
```

```
# Aggregate the data to countries of the world
soiln_countries <- aggregate(soiln, cell = "countries")

soiln_countries$data # look at country time series

# Example 2
# Load an example LPJmLDataCalc object
soiln <- load_soiln_calc()

# Aggregate the to global region
soiln <- aggregate(soiln, cell = list(to = "global", stat = "weighted_sum"))

soiln$data # look at global time series

# Example 3
# Load an example LPJmLDataCalc object
soiln <- load_soiln_calc()

# Take the mean of the data over the full simulation period
# and a weighted mean over the cells
soiln <- aggregate(soiln, time = "sim_period",
                   cell = list(to = "global", stat = "weighted_mean"))

# Look at the resulting value
soiln$data

## End(Not run)
```

---

benchmark *Benchmark one or several LPJmL runs*

Description

Function to benchmark one or several under test LPJmL runs against a baseline run.

Usage

```
benchmark(
  baseline_dir,
  under_test_dirs,
  settings = default_settings,
  metric_options = NULL,
  author = "",
  description = "",
  pdf_report = TRUE,
  ...
)
```

Arguments

`baseline_dir`	Path to directory containing the baseline run.
`under_test_dirs`	List of paths to directories containing the under test run results.
`settings`	List that defines for each output which metrics to use. The list has to have the following structure: • `var1` = Vector of metric classes to use for variable `var1` • `var2` = Vector of metric classes to use for variable `var2` • ...
`metric_options`	List that defines options for the metrics. The list has to have the following structure: • `metric1` = List of options for metric `metric1` • `metric2` = List of options for metric `metric2`
`author`	Name of the author of the benchmark.
`description`	Description of the purpose of the benchmark.
`pdf_report`	Logical, if TRUE a pdf report will be created with the create_pdf_report function.
`...`	additional arguments to be passed to create_pdf_report

Details

In order for the benchmarking to work, all the output files specified in the settings have to be present in the baseline and all under test directories. All output files need to be with ".bin" extension and with meta files of ".bin.json" format. All output paths given to the function need to be distinct. In each output directory there must be a grid and a terr_area file corresponding to the outputs. For each variable the structure of the output files has to be same in each directory (i.e. same cells, same time steps, same bands).

The internal benchmarking process is structured as follows:

1. Create simulation table with meta information of all considered simulations and the short simulation identifiers.
2. Retrieve all summaries of outputs from the baseline and under test runs of the variable by applying the summary method of each metric to all lpjml outputs of variables that are designated to be evaluated with this metric, as specified in the settings. The results are organized in variable groups and stored in the var_grp_list attributes of the metrics. See Metric for details.
3. Add the comparison items to the variable groups, by applying the compare method of each metric to the combination of baseline summary with each under test summary of the variable groups stored in that metric.
4. Apply unit conversions to all data objects of the metrics, as specified in the unit conversion table. See set_lpjmlstats_settings.

Value

A benchmarkResult object containing the numerical results of the benchmarking. This data object is basically a list of all metrics used in the benchmarking. See Metric for the way a metric structures benchmarking results. In addition the benchmarkResult object contains meta information. Of particular importance is the simulation table, which contains the simulation names, paths and the short simulation identifier that are used in the benchmarkResult object.

The function get_benchmark_meta_data can be used to retrieve the meta information.

The data structure of the benchmarkResult object is depicted here:

See Also

create_pdf_report

Examples

```
## Not run:
# Example 1
# Most basic benchmarking with default settings
benchmark("path_to_baseline_results", "path_to_under_test_results")

# Example 2
# Specifying author and description, as well as filename for pdf report
# is recommended. Also, it can make sense to store the benchmarkResult object
# for later analysis.
BM_resu <- benchmark("path_to_baseline_results",
                     "path_to_under_test_results",
                     author = "anonymous",
                     description = "This is a test",
                     output_file = "myBenchmark.pdf")

saveRDS(BM_resu, "bm_results.rds")

# Example 3
# Quick benchmarking that only looks at specific outputs with
# specific metrics and doesn't generate pdf report.
# In addition only the first 10 years are considered
# which gives another significant speedup.
settings <- list(
 vegc = c(GlobSumTimeAvgTable),
 soilc = c(GlobSumTimeAvgTable),
 # this give an aggregation to a single value for baseline and under test
 # and their comparison, displayed in a table
 mgpp = c(GlobSumTimeseries),
 # this gives a time series for baseline and under test
 # displayed as line plots
 mnpp = c(TimeAvgMap)
 # this gives a time average for baseline and under test
 # displayed as maps
)
BM_data <- benchmark("path_to_baseline_results",
                     "path_to_under_test_results",
```

```
                    settings = settings,
                    pdf_report = FALSE)

# Example 4
# Benchmark soiltemp in addition to default settings
# with a special metric
settings <- c(default_settings, # use default settings
              list(msoiltemp1 = c(GlobAvgTimeAvgTable, TimeAvgMap))
              # GlobAvgTimeAvgTable uses a weighted average over space
              # instead of the standard weighted sum
              )
BM_data <- benchmark("path_to_baseline_results",
                     "path_to_under_test_results",
                     settings = settings)

# Example 5
# Benchmark multiple under test runs against the baseline
BM_data <- benchmark("path_to_baseline_results",
                    list("path_to_under_test_results1",
                    "path_to_under_test_results2")
                    )

# Example 6
# Benchmark with custom metric options
metric_options <- list(
  GlobSumTimeAvgTable = list(font_size = 12), # use larger font size in table
  TimeAvgMap = list(highlight = "soilc")      # plots a larger map for soilc
)
BM_data <- benchmark("path_to_baseline_results",
                     "path_to_under_test_results",
                     metric_options = metric_options)

# Example 7
# Benchmark only maize harvest
# The benchmarking allows to select only specific bands of an output
settings <- list(`pft_harvest.pft$rainfed maize; irrigated maize`
                 = c(GlobSumTimeAvgTable))
benchmark("path_to_baseline_results", "path_to_under_test_results",
          settings)

## End(Not run)
```

`build_global_region` *Construct global region object that fully contains all cells given in a grid.*

Description

Construct global region object that fully contains all cells given in a grid.

Usage

```
build_global_region(grid)
```

Arguments

grid	An LPJmLGridData object containing the grid.

Value

An LPJmLRegionData object containing the global region.

See Also

LPJmLRegionData

CellSubsetAnnAvgTimeseries *CellSubsetAnnAvgTimeseries*

Description

CellSubsetAnnAvgTimeseries metric. See Metric for the documentation of metrics in general.

Super classes

lpjmlstats::Metric -> lpjmlstats::GlobSumTimeseries -> lpjmlstats::GlobAvgTimeseries -> CellSubsetAnnAvgTimeseries

Public fields

`m_options` List of metric options specific to this metric:

- `font_size`: integer, font size of the plot (default 6)
- `year_subset`: character vector, defines which calendar years the metric considers, i.e., a data subset that the metric works with; e.g., c("1995", "1996") (default "1901" - "2019").
- `cell`: cells to be subsetted (default 10000)
- `num_cols`: integer, number of columns in the plot grid in the report (default 2)
- `var_subheading`: logical, if TRUE, a linebreak and a subheading will be inserted before plots for a new variable are added to the report. with the name of the variable will be added. Both things are intended to visually seperate the plots of different variables and to better organize the report, especially if the metric generates many plots for each variable. (default FALSE)
- `band_subheading`: analogous to var_subheading but for bands (default FALSE)

`description` Description used in the report

`title` Section header used in the report

Methods

Public methods:

- `CellSubsetAnnAvgTimeseries$summarize()`
- `CellSubsetAnnAvgTimeseries$clone()`

Method `summarize()`**:** Subset the cells and compute an annual average.

Usage:

```
CellSubsetAnnAvgTimeseries$summarize(lpjml_data)
```

Arguments:

`lpjml_data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CellSubsetAnnAvgTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CellSubsetTimeseries` *CellSubsetTimeseries*

Description

CellSubsetTimeseries metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeseries` -> `lpjmlstats::GlobAvgTimeseries` -> `lpjmlstats::CellSubsetAnnAvgTimeseries` -> `CellSubsetTimeseries`

Public fields

`description` Description used in the report

`title` Section header used in the report

Methods

Public methods:

- `CellSubsetTimeseries$summarize()`
- `CellSubsetTimeseries$clone()`

Method `summarize()`**:** Subset the cells.

Usage:

```
CellSubsetTimeseries$summarize(lpjml_data)
```

Arguments:

`lpjml_data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CellSubsetTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`create_literature_pdf` *Function to create a pdf with a table with literature values*

Description

Function to create a pdf with a table with literature values

Usage

```
create_literature_pdf(output_file = "literature_values.pdf", ...)
```

Arguments

`output_file`	filename of the output pdf, can include directory
`...`	additional parameters passed to rmarkdown::render

`create_pdf_report` *Generate a pdf report from a benchmarkResult object.*

Description

Generate a pdf report from a benchmarkResult object.

Usage

```
create_pdf_report(benchmark_result, output_file = "benchmark.pdf", ...)
```

Arguments

`benchmark_result`	benchmarkResult object created by the `benchmark` function
`output_file`	file of the output pdf, including filename and directory `render`
`...`	additional arguments passed to `render`

Details

Each metric has its own section in the report. The content of the section is generated by the plot and plot_arrange function of the metric. The metric results are displayed in the same order as they were specified in the benchmark settings.

Examples

```
## Not run:
  create_pdf_report(BM_data, "myBenchmark.pdf")

## End(Not run)
```

---

default_settings *Default settings for the Benchmarking*

Description

Default settings for the Benchmarking

Usage

```
default_settings
```

Format

An object of class `list` of length 28.

---

get_benchmark_meta_data *Function that returns the meta data of a benchmarkResult object*

Description

Function that returns the meta data of a benchmarkResult object

Usage

```
get_benchmark_meta_data(benchmark_result)
```

Arguments

benchmark_result	A benchmarkResult object

Value

A list with the meta data of the benchmarkResult object. The list contains the author, the description and a simulation identification table. The latter is a tibble with the columns sim_paths, lpjml_version, sim_names, sim_ident and sim_type.

`GlobAvgAnnAvgTimeseries` *GlobAvgAnnAvgTimeseries*

Description

GlobAvgAnnAvgTimeseries metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeseries` -> `GlobAvgAnnAvgTimeseries`

Public fields

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobAvgAnnAvgTimeseries$summarize()`
- `GlobAvgAnnAvgTimeseries$clone()`

Method `summarize()`**:** Take the mean for each year and then the global weighted mean over the cells.

Usage:

```
GlobAvgAnnAvgTimeseries$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobAvgAnnAvgTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobAvgTimeAvgTable *GlobAvgTimeAvgTable*

Description

GlobAvgTimeAvgTable metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeAvgTable` -> `GlobAvgTimeAvgTable`

Public fields

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobAvgTimeAvgTable$summarize()`
- `GlobAvgTimeAvgTable$clone()`

Method `summarize()`**:** First take global weighted mean, then average over all time steps.

Usage:

```
GlobAvgTimeAvgTable$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobAvgTimeAvgTable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobAvgTimeseries *GlobAvgTimeseries*

Description

GlobAvgTimeseries metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeseries` -> `GlobAvgTimeseries`

Public fields

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobAvgTimeseries$summarize()`
- `GlobAvgTimeseries$clone()`

Method `summarize()`**:** Take the global weighted mean over the cells.

Usage:

```
GlobAvgTimeseries$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobAvgTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobSumAnnAvgTimeseries *GlobSumAnnAvgTimeseries*

Description

GlobSumAnnAvgTimeseries metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeseries` -> `GlobSumAnnAvgTimeseries`

Public fields

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobSumAnnAvgTimeseries$summarize()`
- `GlobSumAnnAvgTimeseries$clone()`

Method `summarize()`**:** Take the mean for each year and then the global weighted sum over the cells.

Usage:

```
GlobSumAnnAvgTimeseries$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumAnnAvgTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobSumAnnTimeseriesFPC *GlobSumAnnTimeseriesFPC*

Description

GlobSumAnnTimeseriesFPC metric

Super classes

lpjmlstats::Metric -> lpjmlstats::GlobSumTimeseries -> lpjmlstats::GlobSumAnnAvgTimeseries -> GlobSumAnnTimeseriesFPC

Public fields

`title` Section header used in the report

Methods

Public methods:

- GlobSumAnnTimeseriesFPC$summarize()
- GlobSumAnnTimeseriesFPC$new()
- GlobSumAnnTimeseriesFPC$clone()

Method `summarize()`: Weigh by natural stand fraction and then do the same as GlobSumAnnAvgTimeseries

Usage:

```
GlobSumAnnTimeseriesFPC$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Method `new()`: initialize with an extended description

Usage:

```
GlobSumAnnTimeseriesFPC$new()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
GlobSumAnnTimeseriesFPC$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobSumAnnTimeseriesPFT_harvest

GlobSumAnnTimeseriesPFT_harvest

Description

GlobSumAnnTimeseriesPFT_harvest metric

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeseries` -> `lpjmlstats::GlobSumAnnAvgTimeseries` -> `GlobSumAnnTimeseriesPFT_harvest`

Public fields

`title` Section header used in the report

Methods

Public methods:

- `GlobSumAnnTimeseriesPFT_harvest$summarize()`
- `GlobSumAnnTimeseriesPFT_harvest$new()`
- `GlobSumAnnTimeseriesPFT_harvest$clone()`

Method `summarize()`**:** Weigh by cft_frac and then do the same as GlobSumAnnAvgTimeseries

Usage:

```
GlobSumAnnTimeseriesPFT_harvest$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Method `new()`**:** initialize with an extended description

Usage:

```
GlobSumAnnTimeseriesPFT_harvest$new()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumAnnTimeseriesPFT_harvest$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

GlobSumTimeAvgTable *GlobSumTimeAvgTable*

Description

GlobSumTimeAvgTable metric. See Metric for the documentation of metrics in general.

Super class

lpjmlstats::Metric -> GlobSumTimeAvgTable

Public fields

`m_options` List of metric options specific to this metric:

- `font_size`: integer, font size of the table (default 7)
- `disp_digits`: integer, number of significant digits to display (default 4)
- `year_subset`: character vector, defines which calendar years the metric considers, i.e., a data subset that the metric works with; e.g., c("1995", "1996") (default 1991:2000).
- `cell_subset`: character vector, defines which cells to subset (default NULL)

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobSumTimeAvgTable$summarize()`
- `GlobSumTimeAvgTable$compare()`
- `GlobSumTimeAvgTable$plot()`
- `GlobSumTimeAvgTable$arrange_plots()`
- `GlobSumTimeAvgTable$clone()`

Method `summarize()`**:** First take global weighted sum, then average over all time steps of the simulation period. The result is a scalar for each band.

Usage:

```
GlobSumTimeAvgTable$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `compare()`**:** Calculate difference and relative difference to the baseline.

Usage:

```
GlobSumTimeAvgTable$compare(var_grp)
```

Arguments:

`var_grp` variable group

Method `plot()`**:** Create a table of the results.

Usage:

```
GlobSumTimeAvgTable$plot()
```

Returns: A tibble with the results

Method `arrange_plots()`**:** Style the table to be displayed in the report.

Usage:

```
GlobSumTimeAvgTable$arrange_plots(table)
```

Arguments:

`table` A tibble with the results

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumTimeAvgTable$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GlobSumTimeAvgTableFPC` *GlobSumTimeAvgTableFPC*

Description

GlobSumTimeAvgTableFPC metric

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeAvgTable` -> `GlobSumTimeAvgTableFPC`

Public fields

`title` Section header used in the report

Methods

Public methods:

- `GlobSumTimeAvgTableFPC$summarize()`
- `GlobSumTimeAvgTableFPC$new()`
- `GlobSumTimeAvgTableFPC$clone()`

Method `summarize()`**:** Weigh by natural stand fraction and then do the same as GlobSumTimeAvgTable

Usage:

```
GlobSumTimeAvgTableFPC$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Method `new()`**:** initialize with an extended description

Usage:

```
GlobSumTimeAvgTableFPC$new()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumTimeAvgTableFPC$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GlobSumTimeAvgTablePFT_harvest`

GlobSumTimeAvgTablePFT_harvest

Description

GlobSumTimeAvgTablePFT_harvest metric

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::GlobSumTimeAvgTable` -> `GlobSumTimeAvgTablePFT_harvest`

Public fields

`title` Section header used in the report

Methods

Public methods:

- `GlobSumTimeAvgTablePFT_harvest$summarize()`
- `GlobSumTimeAvgTablePFT_harvest$new()`
- `GlobSumTimeAvgTablePFT_harvest$clone()`

Method `summarize()`**:** Weigh by cft_frac and then do the same as GlobSumTimeAvgTable

Usage:

```
GlobSumTimeAvgTablePFT_harvest$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Method `new()`**:** initialize with an extended description

Usage:

```
GlobSumTimeAvgTablePFT_harvest$new()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumTimeAvgTablePFT_harvest$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GlobSumTimeseries` *GlobSumTimeseries*

Description

GlobSumTimeseries metric. See Metric for the documentation of metrics in general.

Super class

`lpjmlstats::Metric` -> `GlobSumTimeseries`

Public fields

`m_options` List of metric options specific to this metric:

- `font_size`: integer, font size of the plot (default 6)
- `year_subset`: character vector, defines which calendar years the metric considers, i.e., a data subset that the metric works with; e.g., c("1995", "1996") (default "1901" - "2019").
- `cell_subset`: character vector, defines which cells to subset (default NULL)
- `num_cols`: integer, number of columns in the plot grid in the report (default 2)
- `var_subheading`: logical, if TRUE, a linebreak and a subheading will be inserted before plots for a new variable are added to the report. Both things are intended to visually seperate the plots of different variables and to better organize the report, especially if the metric generates many plots for each variable. (default FALSE)
- `band_subheading`: analogous to var_subheading but for bands (default FALSE)

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `GlobSumTimeseries$summarize()`
- `GlobSumTimeseries$plot()`
- `GlobSumTimeseries$arrange_plots()`
- `GlobSumTimeseries$clone()`

Method `summarize()`**:** Take a global weighted sum of the output.

Usage:

```
GlobSumTimeseries$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `plot()`**:** Create a time series plot of the results.

Usage:

```
GlobSumTimeseries$plot()
```

Returns: A list of time series ggplots

Method `arrange_plots()`**:** Arrange the time series plots side by side with legends pooled together in the top left

Usage:

```
GlobSumTimeseries$arrange_plots(plotlist)
```

Arguments:

`plotlist` List of time series ggplots

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GlobSumTimeseries$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LPJmLDataCalc` *LPJmLDataCalc*

Description

An extended LPJmLData class that enables arithmetic and statistics.

Super class

`lpjmlkit::LPJmLData` -> `LPJmLDataCalc`

Active bindings

`data` the data array

`.data_with_unit` Returns the internal enclosed unit object !Internal method only to be used for package development!

`.meta` Returns the actual LPJmLMetaDataCalc object !Internal method only to be used for package development!

Methods

Public methods:

- LPJmLDataCalc$new()
- LPJmLDataCalc$aggregate()
- LPJmLDataCalc$add_band()
- LPJmLDataCalc$get_ref_area()
- LPJmLDataCalc$plot()
- LPJmLDataCalc$.check_internal_integrity()
- LPJmLDataCalc$.plot_aggregated()
- LPJmLDataCalc$.add()
- LPJmLDataCalc$.subtract()
- LPJmLDataCalc$.multiply()
- LPJmLDataCalc$.divide()
- LPJmLDataCalc$.convert_unit()
- LPJmLDataCalc$.set_unit()
- LPJmLDataCalc$apply_unit_conversion_table()
- LPJmLDataCalc$add_grid()
- LPJmLDataCalc$clone()

Method `new()`**:** Create a new LPJmLDataCalc object; to be used internally or explicitly !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$new(lpjml_data)
```

Arguments:

`lpjml_data` an LPJmLData object.

Method `aggregate()`**:** See aggregate.

Usage:

```
LPJmLDataCalc$aggregate(ref_area = "terr_area", ...)
```

Arguments:

`ref_area` See aggregate.

`...` See aggregate.

Method `add_band()`**:** Add a band to the object by applying a function to the band vector for each spacial and temporal unit

Usage:

```
LPJmLDataCalc$add_band(band_name, fun)
```

Arguments:

`band_name` Name of band

`fun` function

Method `get_ref_area()`**:** Get the reference area of the LPJmLDataCalc object. For an area density variable the reference area should be the area of each cell on which the variable is defined.

Usage:

```
LPJmLDataCalc$get_ref_area(ref_area)
```

Arguments:

`ref_area` A string that can be

- `terr_area` terrestrial area (land area including inland water bodies)
- `cell_area` full area of each cell

Returns: An `LPJmLDataCalc` object with the reference area as variable.

Method `plot()`**:** Plot an LPJmLDataCalc object

The function acts a wrapper of plot.LPJmLData from lpjmlkit, but allows for plotting data in more formats.

In case of non-aggregated data plot.LPJmLData is directly called. In case of aggregated data the value for each region is assigned to all pixels that belong to the region. If a pixel belong to a region only partially, the value is multiplied by the fraction of that pixel belonging to the region. If a pixel belongs to multiple regions, the sum of all respective region values (multiplied by the fractions) is taken. The pixel values are then again plotted with plot.LPJmLData.

Usage:

```
LPJmLDataCalc$plot(...)
```

Arguments:

`...` Arguments passed to LPJmLData plot method.

Method `.check_internal_integrity()`**:** Check consistency of data and meta data !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.check_internal_integrity()
```

Method `.plot_aggregated()`**:** Plot aggregated data. Performs a very simple disaggregation to create LPJmLData obj that can be plotted with plot.LPJmLData. For each pixel the values of all regions that contain the pixel are multiplied by the fractions and summed up.

Usage:

```
LPJmLDataCalc$.plot_aggregated(...)
```

Arguments:

`...` Arguments to be passed to plot.LPJmLData

Method `.add()`**:** Addition of two LPJmLDataCalc objects !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.add(lpjml_calc_obj)
```

Arguments:

`lpjml_calc_obj` An `LPJmLData` object.

Method `.subtract()`**:** Subtraction of two LPJmLDataCalc objects !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.subtract(lpjml_calc_obj)
```

Arguments:

`lpjml_calc_obj` An `LPJmLData` object.

Method `.multiply()`**:** Multiplication of two LPJmLDataCalc objects !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.multiply(lpjml_calc_obj)
```

Arguments:

`lpjml_calc_obj` An `LPJmLData` object.

Method `.divide()`**:** Division of two LPJmLDataCalc objects !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.divide(lpjml_calc_obj)
```

Arguments:

`lpjml_calc_obj` An `LPJmLData` object.

Method `.convert_unit()`**:** Unit conversion of LPJmLDataCalc object !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.convert_unit(unit)
```

Arguments:

`unit` A string with the unit to convert to.

Method `.set_unit()`**:** Set unit of LPJmLDataCalc object !Internal method only to be used for package development!

Usage:

```
LPJmLDataCalc$.set_unit(unit_str)
```

Arguments:

`unit_str` A string with the unit to be set.

Method `apply_unit_conversion_table()`**:** Apply unit conversion from conversion table

Usage:

```
LPJmLDataCalc$apply_unit_conversion_table(path_to_table = NULL)
```

Arguments:

`path_to_table` A string with the path to the conversion table.

Method `add_grid()`**:** Add a grid to the LPJmLDataCalc object Wrapper for the `add_grid` method of the `LPJmLData` class.

Usage:

```
LPJmLDataCalc$add_grid()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LPJmLDataCalc$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LPJmLMetaDataCalc` *LPJmL meta data class*

Description

A meta data container for the LPJmLDataCalc class that extends the LPJmLMetaData such that aggregation can be tracked.

Super class

`lpjmlkit::LPJmLMetaData` -> `LPJmLMetaDataCalc`

Active bindings

`space_aggregation` boolean, Indication weather the data has been subject to space aggregation.

`time_aggregation` boolean, Indication weather the data has been subject to time aggregation.

`band_names_disp` named vector, versions of band names used for display, usually shorter

`pos_in_var_grp` list, position of the lpjml_calc inside of its var_grp.

`sim_ident` string, simulation identifier

`var_and_band_disp` string, variable name together with name of first band, e.g. `soiln$200`

Methods

Public methods:

- `LPJmLMetaDataCalc$new()`
- `LPJmLMetaDataCalc$.__set_space_aggregation__()`
- `LPJmLMetaDataCalc$.__set_time_aggregation__()`
- `LPJmLMetaDataCalc$print()`
- `LPJmLMetaDataCalc$.__set_sim_ident__()`
- `LPJmLMetaDataCalc$.__set_pos_in_var_grp__()`
- `LPJmLMetaDataCalc$clone()`

Method `new()`**:** Initialize the LPJmLMetaDataCalc object by copying all private attributes from an LPJmLMetaData object to private attributes of this object. !Internal method only to be used for package development!

Usage:

```
LPJmLMetaDataCalc$new(lpjml_meta)
```

Arguments:

`lpjml_meta` an LPJmLMetaData object.

Method `.__set_space_aggregation__()`: Save in metadata that data is in space_aggregation format !Internal method only to be used for package development!

Usage:

```
LPJmLMetaDataCalc$.__set_space_aggregation__(agg_method)
```

Arguments:

`agg_method` string indicating the aggregation method

Method `.__set_time_aggregation__()`: Save in metadata that data is in time_aggregation format !Internal method only to be used for package development!

Usage:

```
LPJmLMetaDataCalc$.__set_time_aggregation__(agg_method)
```

Arguments:

`agg_method` string indicating the aggregation method

Method `print()`: Wrapper for `LPJmLMetaData` print method.

Usage:

```
LPJmLMetaDataCalc$print(spaces = "", ...)
```

Arguments:

`spaces` string of spaces to be printed as prefix

`...` additional arguments passed to `LPJmLMetaData` print method

Method `.__set_sim_ident__()`: Set the simulation identifier !Internal method only to be used for package development!

Usage:

```
LPJmLMetaDataCalc$.__set_sim_ident__(sim_ident)
```

Arguments:

`sim_ident` string, simulation identifier

Method `.__set_pos_in_var_grp__()`: Set the position of the lpjml_calc inside of its var_grp. !Internal method only to be used for package development!

Usage:

```
LPJmLMetaDataCalc$.__set_pos_in_var_grp__(pos_in_var_grp)
```

Arguments:

`pos_in_var_grp` A list with the position of the lpjml_calc inside of the var_grp. The first entry is the type; can be "baseline", "under_test" or "compare". The second entry is the compare item if of type "compare", e.g. "diff". E.g. list("under_test") or list("compare", "diff").

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
LPJmLMetaDataCalc$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

LPJmLRegionData *LPJmLRegionData*

Description

A class that represents one or several regions in LPJmL. Based on an LPJmL grid, a region is defined as set of grid cells together with fractions. The fractions indicate the share of each grid cell that is part of the region. (e.g. 1 = the cell belongs completely to the region, 0 = the cell does not belong to the region at all).\ The underlying data structure is a sparse matrix, where the rows represent the regions, the columns represent the grid cells and the values represent the fractions (cells not belonging to a region do not take memory as only nonzero entries are stored in a sparse matrix).

Create a new `LPJmLRegionData` object; only used internally.

Active bindings

`region_matrix` object stores the region data as a sparse matrix.

`grid` `LPJmLGridData` object containing the underlying grid.

Methods

Public methods:

- `LPJmLRegionData$new()`
- `LPJmLRegionData$get_ncells_per_region()`
- `LPJmLRegionData$clone()`

Method `new()`**:** !Internal method only to be used by the package itself!

Usage:

```
LPJmLRegionData$new(grid, region_matrix)
```

Arguments:

`grid` `LPJmLGridData` object containing the underlying grid.

`region_matrix` object stores the region data as a sparse matrix.

Method `get_ncells_per_region()`**:** Get number of cells per region.

Usage:

```
LPJmLRegionData$get_ncells_per_region()
```

Details: For partially belonging cells the fraction of the cell that belongs to the region is counted.

Returns: A vector of length nrow(region_matrix) containing the number of cells per region.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LPJmLRegionData$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Metric` *Metric super class*

Description

A metric is a) a structured, generic processing pipeline to calculate numerical indicators for sets of lpjml outputs as well as to display these indicators in a report, and b) a structured data container that stores the numerical indicators resulting from a).

a) A metric defines a procedure to

1. **summarize** a complex, multidimensional LPJmL output in a meaningfull, potentially opinionated way, typically involving the reduction of its time and or space dimension,
2. **compare** how this summary statistic of an output variable changes, when going from an established baseline LPJmL version or configuration to a new version or configuration currently under test,
3. **plot** the results of 1. and 2. as a figure or table and
4. **arrange** these plots in a report, (e.g. styling, side by side arrangement)

See benchmark on how these steps come into play in the benchmarking process.

b) The summarized outputs as well as comparisons that are stored a metric are grouped by the different lpjml variables. A so called variable group (var_grp) contains

1. the summary of the baseline output of that variable
2. a list of all summarized under test outputs of that variable
3. a list of compare items (e.g. difference, relative difference). Each compare item is a list of comparisons of the baseline summary with each under test summary using the specific method of that item.

All variable groups are stored in the var_grp_list attribute of the metric.

As the cornerstone of the benchmarking process, all metrics illuminate the change of LPJmL results from different angles, and should together provide a comprehensive picture of the effects of modifications in code or settings.

See GlobSumTimeAvgTable for a typical example of a metric.

Public fields

`m_options` List of metric options Will be overwritten by the individual metric subclasses.

`var_grp_list` List of variable groups. Each variable group contains the summaries and the comparisons for one variable.

Methods

Public methods:

- Metric$summarize()
- Metric$compare()
- Metric$plot()
- Metric$arrange_plots()
- Metric$capture_summary()
- Metric$store_summary()
- Metric$add_comparisons()
- Metric$add_compare_meta()
- Metric$transform_lpjml_calcs()
- Metric$generate_report_content()
- Metric$print_metric_header()
- Metric$print_metric_description()
- Metric$print_year_subset()
- Metric$clone()

Method `summarize()`**:** Pipeline to summarize the raw data. Will be overwritten by the individual metric subclasses.

Usage:

```
Metric$summarize(data)
```

Arguments:

`data` Raw data to be summarized

Method `compare()`**:** Pipeline to compare the baseline summary with each under test summary stored in the metric. Will be overwritten by the individual metric subclasses.

Usage:

```
Metric$compare(var_grp)
```

Arguments:

`var_grp` variable group

Method `plot()`**:** Function to plot the results of the metric. Will be overwritten by the individual metric subclasses.

Usage:

```
Metric$plot(var_grp)
```

Arguments:

`var_grp` variable group

Method `arrange_plots()`**:** Function to arrange all plots of the metric in the respective section of the report. Will be overwritten by the individual metric subclasses.

Usage:

```
Metric$arrange_plots(var_grp)
```

Arguments:

`var_grp` variable group

Method `capture_summary()`: !Package internal method!

Usage:

```
Metric$capture_summary(lpjml_calc, var, type)
```

Arguments:

`lpjml_calc` Raw data to be summarized

`var` Variable name

`type` Type of data ("baseline" or "under_test")

Method `store_summary()`: !Package internal method! Store the summary in the variable group

Usage:

```
Metric$store_summary(summary, var, type)
```

Arguments:

`summary` Summary to be stored

`var` Variable name

`type` Type of data ("baseline" or "under_test")

Method `add_comparisons()`: !Package internal method! Compare and store the comparison in all variable groups

Usage:

```
Metric$add_comparisons()
```

Method `add_compare_meta()`: !Package internal method! Add the position of the comparisons within the var_grp to meta

Usage:

```
Metric$add_compare_meta(var_grp)
```

Arguments:

`var_grp` variable group

Method `transform_lpjml_calcs()`: !Package internal method! Apply function to all lpjml_calcs in all eval groups and lists

Usage:

```
Metric$transform_lpjml_calcs(fun, ...)
```

Arguments:

`fun` Function to apply

`...` Additional arguments passed to fun

Method `generate_report_content()`: !Package internal method! Generate the full report content of the metric.

Usage:

```
Metric$generate_report_content()
```

Method `print_metric_header()`: !Package internal method! Function to print the metric header.

Usage:

```
Metric$print_metric_header()
```

Method `print_metric_description()`: !Package internal method! Function to print the metric description.

Usage:

```
Metric$print_metric_description()
```

Method `print_year_subset()`: !Package internal method! Function to print the year_subset metric option.

Usage:

```
Metric$print_year_subset()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Metric$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`plot.LPJmLDataCalc` *Plot an LPJmLDataCalc object*

Description

The function acts a wrapper of plot.LPJmLData from lpjmlkit, but allows for plotting data in more formats.

Usage

```
## S3 method for class 'LPJmLDataCalc'
plot(x, ...)
```

Arguments

`x`	LPJmLDataCalc object. In case of non-aggregated data plot.LPJmLData is directly called. In case of aggregated data the value for each region is assigned to all pixels that belong to the region. If a pixel belong to a region only partially, the value is multiplied by the fraction of that pixel belonging to the region. If a pixel belongs to multiple regions, the sum of all respective region values (multiplied by the fractions) is taken. The pixel values are then again plotted with plot.LPJmLData.
`...`	Arguments passed to LPJmLData plot method.

read_cow_regions *Read or create the cow regions as an LPJmLRegionData object*

Description

The COW = countries of the world data contains global country borders.

Usage

```
read_cow_regions()
```

Value

An LPJmLRegionData object containing the cow regions.

See Also

LPJmLRegionData

read_def_grid *Read default grid*

Description

The default grid is the standard global grid used in LPJmL.

Usage

```
read_def_grid()
```

Value

An LPJmLGridData object containing the default grid.

See Also

LPJmLGridData

read_io *Read in LPJmL input and output files as LPJmLDataCalc*

Description

The function acts a wrapper of read_io from lpjmlkit, but outputs an `LPJmLDataCalc` object.

Usage

```
read_io(..., output_type = "LPJmLDataCalc")
```

Arguments

`...`	Parameters that are passed to read_io
`output_type`	Can be either `LPJmLDataCalc`or `LPJmLData`.

Value

An `LPJmLDataCalc` object

set_lpjmlstats_settings *Set the package settings for lpjmlstats*

Description

This function configures various settings for the lpjmlstats package.

Usage

```
set_lpjmlstats_settings(...)
```

Arguments

`...`	Variable arguments to specify settings. The function accepts the following options: • `graphics_device`: A character string specifying the graphics device to be used for plotting the benchmarking results. Defaults to "png". Use "pdf" for vector graphics. • `pdf_plot_dpi`: A numeric value specifying the DPI for the PDF document. • `unit_table_path`: A character string specifying the path to the unit conversion table (a .csv file). Defaults to the conversion table in the package's `inst` folder. The specified file must exist and be a .csv file.

- `metrics_at_start`: A vector of strings to be matched against the names of the metrics. The matched metrics will be prioritized, that is run first and displayed at the report beginning. The prioritization will be in the same order as the vector.
- `file_extension`: A string indicating the file extension to be used by read_io.

Examples

```
## Not run:
set_lpjmlstats_settings(unit_table_path = "path/to/my_table.csv",
pdf_plot_dpi = 300)

## End(Not run)
```

`subset.LPJmLDataCalc` *Subset an LPJmLDataCalc object*

Description

Function to subset an LPJmLDataCalc object. The function acts as a wrapper of subset.LPJmLData from lpjmlkit, but outputs an `LPJmLDataCalc` object, in particular keeping its unit.

Usage

```
## S3 method for class 'LPJmLDataCalc'
subset(x, ...)
```

Arguments

`x`	LPJmLDataCalc object.
`...`	Parameters that are passed to subset.LPJmLData.

Value

An `LPJmLDataCalc` object.

TimeAvgMap *TimeAvgMap*

Description

TimeAvgMap metric. See Metric for the documentation of metrics in general.

Super class

lpjmlstats::Metric -> TimeAvgMap

Public fields

`m_options` List of metric options specific to this metric:

- `font_size`: integer, font size of the map plot (default 6)
- `highlight`: vector of strings, indicating which variables should be highlighted in the report, that is receive a larger plot at the beginning of report content of the metric. All variables with a name that contains at least one these strings as a substring, will not be plotted in the plotgrid (see `num_cols`) but before the plot grid starts. These plots are allowed to extent to full page width. (default NULL)
- `quantiles`: quantiles used to determine the lower and upper limits for the values in the map plot (default c(0.05, 0.95))
- `n_breaks`: number of breaks for each arm of the diverging color scale (default 3)
- `year_subset`: character vector, defines which calendar years the metric considers, i.e., a data subset that the metric works with; e.g., c("1995", "1996") (default 1991:2000).
- `cell_subset`: character vector, defines which cells to subset (default NULL)
- `num_cols`: integer, number of columns in the plot grid in the report (default 2)
- `var_subheading`: logical, if TRUE, a linebreak and a subheading will be inserted before plots for a new variable are added to the report. with the name of the variable will be added. Both things are intended to visually seperate the plots of different variables and to better organize the report, especially if the metric generates many plots for each variable. (default FALSE)
- `band_subheading`: analogous to var_subheading but for bands (default FALSE)

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `TimeAvgMap$summarize()`
- `TimeAvgMap$compare()`
- `TimeAvgMap$plot()`
- `TimeAvgMap$arrange_plots()`
- `TimeAvgMap$clone()`

Method `summarize()`**:** Take the mean over the simulation period for each cell.

Usage:

```
TimeAvgMap$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Returns: A summarized LPJmLDataCalc object

Method `compare()`**:** Compare the baseline summary with the under test summaries by subtracting the baseline from the under test.

Usage:

```
TimeAvgMap$compare(var_grp)
```

Arguments:

`var_grp` variable group

Method `plot()`**:** Create a map plot with country border overlay.

Usage:

```
TimeAvgMap$plot()
```

Returns: A list of map ggplots

Method `arrange_plots()`**:** Arrange the map plots side by side

Usage:

```
TimeAvgMap$arrange_plots(plotlist)
```

Arguments:

`plotlist` List of map ggplots

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TimeAvgMap$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TimeAvgMapTreeCover` *TimeAvgMapTreeCover*

Description

TimeAvgMapTreeCover metric

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::TimeAvgMap` -> `lpjmlstats::TimeAvgMapWithAbs` -> `TimeAvgMapTreeCover`

Public fields

`m_options` See TimeAvgMapWithAbs for the documentation of the options of this metric.

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `TimeAvgMapTreeCover$summarize()`
- `TimeAvgMapTreeCover$clone()`

Method `summarize()`**:** add up bands for tree fpcs

Usage:

```
TimeAvgMapTreeCover$summarize(data)
```

Arguments:

`data` LPJmLDataCalc object to be summarized

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TimeAvgMapTreeCover$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TimeAvgMapWithAbs` *TimeAvgMapWithAbs*

Description

TimeAvgMapWithAbs metric. See Metric for the documentation of metrics in general.

Super classes

`lpjmlstats::Metric` -> `lpjmlstats::TimeAvgMap` -> `TimeAvgMapWithAbs`

Public fields

`m_options` List of metric options specific to this metric:

- `font_size`: integer, font size of the map plot (default 6)
- `highlight`: vector of strings, indicating which variables should be highlighted in the report, that is receive a larger plot at the beginning of report content of the metric. All variables with a name that contains at least one these strings as a substring, will not be plotted in the plotgrid (see `num_cols`) but before the plot grid starts. These plots are allowed to extent to full page width. (default NULL)

- quantiles: quantiles used to determine the lower and upper limits for the values in the map plot (default c(0.05, 0.95))
- sep_cmp_lims: logical, if TRUE not all plots of a var_grp will have the same limits anymore, but the compare plots have their own separate limits (default TRUE)
- n_breaks: number of breaks for each arm of the diverging color scale (default 3)
- year_subset: character vector, defines which calendar years the metric considers, i.e., a data subset that the metric works with; e.g., c("1995", "1996") (default 1991:2000).
- cell_subset: character vector, defines which cells to subset (default NULL)
- num_cols: integer, number of columns in the plot grid in the report (default 3)
- var_subheading: logical, if TRUE, a linebreak and a subheading will be inserted before plots for a new variable are added to the report. with the name of the variable will be added. Both things are intended to visually seperate the plots of different variables and to better organize the report, especially if the metric generates many plots for each variable. (default FALSE)
- band_subheading: analogous to var_subheading but for bands (default FALSE)

`title` Section header used in the report

`description` Description used in the report

Methods

Public methods:

- `TimeAvgMapWithAbs$compare()`
- `TimeAvgMapWithAbs$plot()`
- `TimeAvgMapWithAbs$arrange_plots()`
- `TimeAvgMapWithAbs$clone()`

Method `compare()`**:** Compare the baseline summary with the under test summaries by subtracting the baseline from the under test.

Usage:

```
TimeAvgMapWithAbs$compare(var_grp)
```

Arguments:

`var_grp` variable group

Method `plot()`**:** Create a map plot with country border overlay.

Usage:

```
TimeAvgMapWithAbs$plot()
```

Returns: A list of map ggplots

Method `arrange_plots()`**:** Arrange the map plots side by side

Usage:

```
TimeAvgMapWithAbs$arrange_plots(plotlist)
```

Arguments:

`plotlist` List of map ggplots

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TimeAvgMapWithAbs$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index