Package 'madrat'

May 29, 2019

Type Package

Title May All Data be Reproducible and Transparent (MADRaT) *

Version 1.61.0

Date 2019-05-29

Description Provides a framework which should improve reproducibility and transparency in data processing. It provides functionality such as automatic meta data creation and management, rudimentary quality management, data caching, work-flow management and data aggregation.
* The title is a wish not a promise. By no means we expect this package to deliver everything what is needed to achieve full reproducibility and transparency, but we believe that it supports efforts in this direction.

Depends R(>= 2.10.0), magclass(>= 4.49)

Imports spam, tools, utils, digest, methods, parallel, stringr, reshape2, stats, nnls, rlang, assertthat

Suggests igraph(>= 1.0.1), knitr, rmarkdown, testthat, ggplot2

URL https://github.com/pik-piam/madrat, https://doi.org/10.5281/zenodo.1115490

BugReports https://github.com/pik-piam/madrat/issues

License BSD_2_clause + file LICENSE

LazyData no

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Jan Philipp Dietrich [aut, cre],
Lavinia Baumstark [aut],
Stephen Wirth [aut],
Anastasis Giannousakis [aut],
Renato Rodrigues [aut],
Benjamin Leon Bodirsky [aut],
Ulrich Kreidenweis [aut]

Maintainer Jan Philipp Dietrich <dietrich@pik-potsdam.de>

Repository CRAN

Date/Publication 2019-05-29 13:30:03 UTC

R topics documented:

madrat-package *May All Data be Reproducible And Transparent (madrat) **

Description

Package provides a basic framework which should improve reproducibility and transparency in data processing. It provides functionality such as automatic meta data creation and management, rudimentary quality management, data caching, work-flow management an data aggregation.

Details

* The title is a wish not a promise. By no means we expect this package to deliver everything what is needed to achieve full reproducibility and transparency, but we believe that it supports efforts in this direction.

calcOutput *calcOutput*

Description

Calculate a specific output for which a calculation function exists package. The function is a wrapper for specific functions designed for the different possible output types.

Usage

```
calcOutput(type, aggregate = TRUE, file = NULL, years = NULL,
  round = NULL, supplementary = FALSE, append = FALSE,
  na_warning = TRUE, try = FALSE, ...)
```

Arguments

type	output type, e.g. "TauTotal". A list of all available source types can be retrieved with function getCalculations.
aggregate	Boolean indicating whether output data aggregation should be performed or not, "GLO" (or "glo") for aggregation to one global region, "REG+GLO" (or "reg-glo") for a combination of regional and global data.
file	A file name. If given the output is written to that file in the outputfolder as specified in the config.
years	A vector of years that should be returned. If set to NULL all available years are returned.
round	A rounding factor. If set to NULL no rounding will occur.
supplementary	boolean deciding whether supplementary information such as weight should be returned or not. If set to TRUE a list of elements will be returned!

append	boolean deciding whether the output data should be appended in the existing file. Works only when a file name is given in the function call.
na_warning	boolean deciding whether NAs in the data set should create a warning or not
try	if set to TRUE the calculation will only be tried and the script will continue even if the underlying calculation failed. If set to TRUE calculation will stop with an error in such a case.
...	Additional settings directly forwarded to the corresponding calculation function

Value

magpie object with the requested output data either on country or on regional level depending on the choice of argument "aggregate" or a list of information if supplementary is set to TRUE.

Note

The underlying calc-functions are required to provide a list of information back to `calcOutput`. Following list entries should be provided:

- x - the data itself as magclass object
- weight - a weight for the spatial aggregation
- unit - unit of the provided data
- description - a short description of the data
- note (optional) - additional notes related to the data
- isocountries (optional | default = TRUE (mostly) or FALSE (if global)) - a boolean indicating whether data is in iso countries or not (the latter will deactivate several features such as aggregation)
- mixed_aggregation (optional | default = FALSE) - boolean which allows for mixed aggregation (weighted mean mixed with summations). If set to TRUE weight columns filled with NA will lead to summation, otherwise they will trigger an error.
- min (optional) - Minimum value which can appear in the data. If provided calcOutput will check whether there are any values below the given threshold and warn in this case
- max (optional) - Maximum value which can appear in the data. If provided calcOutput will check whether there are any values above the given threshold and warn in this case
- aggregationFunction (optional | default = toolAggregate) - Function to be used to aggregate data from country to regions. The function must have the argument `x` for the data itself and `rel` for the relation mapping between countries and regions and must return the data as magpie object in the spatial resolution as defined in rel.
- aggregationArguments (optional) - List of additional, named arguments to be supplied to the aggregation function. In addition to the arguments set here, the function will be supplied with the arguments `x`, `rel` and if provided/deviating from the default also `weight` and `mixed_aggregation`.

Author(s)

Jan Philipp Dietrich

See Also

setConfig, calcTauTotal,

Examples

```
## Not run:

a <- calcOutput(type="TauTotal")


## End(Not run)
```

---

calcTauTotal *Calculate total tau*

Description

This function prepares total tau values for use. As the source data already provides all required information this function purely removes not required data and moves xref values to the weighting object which is required for aggregation.

Usage

```
calcTauTotal()
```

Value

Total tau data and corresponding weights as a list of two MAgPIE objects

Author(s)

Jan Philipp Dietrich

See Also

calcOutput, readTau, convertTau

Examples

```
## Not run:
calcOutput("TauTotal")


## End(Not run)
```

convertTau *Convert Tau*

Description

Convert landuse intensity data (tau) to data on ISO country level.

Usage

```
convertTau(x)
```

Arguments

x	MAgPIE object containing tau values and corresponding weights xref at 0.5deg cellular level.

Value

Tau data and weights as MAgPIE object aggregated to country level

Author(s)

Jan Philipp Dietrich

diagnosticSources *diagnosticSources*

Description

Function to extract sources from a diagnostics.csv file

Usage

```
diagnosticSources(file)
```

Arguments

file	path to a diagnostics.csv file (as created by retrieveData)

Value

a vector with sources detected in the diagnostics.csv

Author(s)

Jan Philipp Dietrich

See Also

retrieveData

`downloadSource` *downloadSource*

Description

Download a source. The function is a wrapper for specific functions designed for the different possible source types.

Usage

```
downloadSource(type, overwrite = FALSE)
```

Arguments

`type`	source type, e.g. "IEA". A list of all available source types can be retrieved with function `getSources("download")`.
`overwrite`	Boolean deciding whether existing data should be overwritten or not.

Author(s)

Jan Philipp Dietrich

See Also

setConfig, readSource

Examples

```
## Not run:
a <- downloadSource("Tau")

## End(Not run)
```

`file2destination` *Tool: file2destination*

Description

Function which maps a file of the data output folder to the destination path in the corresponding folder. Mapping is stored in a the data output folder in a file called 'file2destination.txt'

Usage

```
file2destination(file, destination)
```

Arguments

`file`	The name of the file which should be mapped to a destination
`destination`	The path relative to the main folder of the model to which the file should be copied. In the case that the file should be copied to more than one destination within the model data should be provided as a vector of destinations.

Author(s)

Jan Philipp Dietrich

See Also

`calcOutput`, `retrieveData`

Examples

```
## Not run:
file2destination("example.txt","example/folder")

## End(Not run)
```

`fingerprint` *Tool: fingerprint*

Description

Function which creates a fingerprint of a folder together with some R objects (e.g. functions). Based on the fingerprint it is possible to decide whether there were some changes in the given folder and the given objects or not. If the fingerprint is unchanged also all files and objects stayed the same, otherwise the fingerprint changes

Usage

```
fingerprint(folder, ...)
```

Arguments

folder	A folder containing objects for which the fingerprint should be created (all files in that folder and all sub-folders will be considered)
...	R data objects that should be considered for the fingerprint as well (e.g. functions, variables,...)

Value

A md5-based fingerprint of all provided sources

Author(s)

Jan Philipp Dietrich

See Also

readSource

Examples

```
## Not run:
fingerprint(".",ls,c)

## End(Not run)
```

---

fullEXAMPLE *fullExample*

Description

Example for class of fullX functions. Can be used as template for a new function or for testing the basic functionality

Usage

```
fullEXAMPLE(rev = 0)
```

Arguments

rev	data revision which should be used as input (positive numeric). setConfig (e.g. for setting the mainfolder if not already set properly).

Author(s)

Jan Philipp Dietrich

See Also

readSource,getCalculations,calcOutput,setConfig ,file2destination

Examples

```
## Not run:
retrieveData("Test",rev=12,regionmapping="regionmappingH12.csv")

## End(Not run)
```

---

getCalculations *getCalculations*

Description

This function can be used to retrieve a list of currently available sources and outputs (based on the availability of corresponding conversion functions in the loaded data data processing packages.)

Usage

```
getCalculations(prefix = "calc", packages = getConfig("packages"),
  globalenv = getConfig("globalenv"))
```

Arguments

prefix	Type of calculations. Available options are "download" (source download), "read" (source read), "correct" (source corrections), "convert" (source conversion to ISO countries), "calc" (further calculations), and "full" (collections of calculations)
packages	A character vector with packages for which the available Sources/Calculations should be returned
globalenv	Boolean deciding whether sources/calculations in the global environment should be included or not

Value

A data frame containing all currently available outputs of all loaded data processing packages including its name, its function call and its package origin.

Author(s)

Jan Philipp Dietrich

See Also

readSource, setConfig

Examples

```
print(getCalculations())
print(getCalculations("read"))
```

---

getConfig *getConfig*

Description

This function returns the madrat config which is currently loaded. If no configuration has been loaded so far the configuration will be initialized with default settings or system settings (if available).

Usage

```
getConfig(option = NULL, raw = FALSE, verbose = TRUE)
```

Arguments

option	The option for which the setting should be returned. If set to NULL all options are returned.
raw	If set to FALSE some settings will be calculated, e.g. if the cache folder is set to FALSE the full path will be calculated using the main folder, or if the verbosity is not set the default verbosity will be returned. If raw is set to TRUE settings are returned as they are currently stored.
verbose	boolean deciding whether status information/updates should be shown or not

Value

A config list with all settings currently set for the madrat package

Author(s)

Jan Philipp Dietrich

See Also

setConfig, initializeConfig

getDependentCalls *getDependentCalls*

Description

Function to retrieve all calls of calcOutput and redSource inside a calcOutput or readSource call.

Usage

```
getDependentCalls(type, ...)
```

Arguments

type	Type for which the calls it depends on should be retrieved
...	additional arguments passed on

Value

a list containing n lists depending on the number of Calculations available for type. Each list contains the associated calls as character vectors

Author(s)

Stephen Wirth, Jan Philipp Dietrich

Examples

```
getDependentCalls("FAO")
getDependentCalls("fullMAGPIE")
```

getISOlist *get official ISO list*

Description

Function which returns the ISO list which is used as default for the input data preparation. It contains the countries to which all source data has to be up- or downscaled to.

Usage

```
getISOlist(type = "all")
```

Arguments

type	Determines what countries should be returned. "all" returns all countries, "important" returns all countries which are above the population threshold set in the configuration and "dispensable" returns all countries which are below the threshold.

Value

vector of default ISO country codes.

Note

Please always use this function instead of directly referring to the data object as the format in this data list might change in the future!

Author(s)

Jan Philipp Dietrich

See Also

getSources, getCalculations

Examples

```
getISOlist()
getISOlist("dispensable")
```

---

getMainfolder *getMainfolder*

Description

Functions checks for a global setting of the mainfolder (either by setting the environment variable "MADRAT_MAINFOLDER" or by setting the R option with the same name). If none of these is available the user will be asked for a directory. If this is not provided a temporary folder will be used.

Usage

```
getMainfolder(verbose = TRUE)
```

Arguments

verbose	boolean deciding whether status information/updates should be shown or not

Author(s)

Jan Philipp Dietrich

See Also

initializeConfig, getConfig, setConfig

---

getSources *getSources*

Description

These functions can be used to retrieve a list of currently available sources and outputs (based on the availability of corresponding conversion functions in the loaded data data processing packages.)

Usage

```
getSources(type = NULL, packages = getConfig("packages"),
  globalenv = getConfig("globalenv"))
```

Arguments

type	Type of source, either set to "global" "regional", "download", "correct" or NULL. Global returns all global sources (non-regional), regional returns sources with regional data, "download" returns source for which a download function is available, "correct" returns sources for which a correct function is available and NULL returns all available sources
packages	A character vector with packages for which the available Sources/Calculations should be returned
globalenv	Boolean deciding whether sources/calculations in the global environment should be included or not

Value

A vector containing all currently available sources or outputs of all loaded data processing packages.

Note

Please be aware that these functions assume that required source files do exist and are set correctly in the corresponding config file.

Author(s)

Jan Philipp Dietrich

See Also

readSource, setConfig

Examples

```
print(getSources())
```

`initializeConfig` *initializeConfig*

Description

Checks whether configuration already has been set. If not, it will be initialized with default settings or (if available) system settings.

Usage

```
initializeConfig(verbose = TRUE)
```

Arguments

`verbose`	boolean deciding whether status information/updates should be shown or not

Author(s)

Jan Philipp Dietrich

See Also

`getMainfolder`, `getConfig`, `setConfig`

`madapply` *madapply*

Description

Wrapper Function that executes multiple function calls on one core using apply or on the supplied number of cores using parLapply if the parallel option is set to TRUE. Make sure the package `parallel` is installed and runs on your machine before setting parallel to TRUE.

Usage

```
madapply(X = NULL, MARGIN = NULL, FUN = NULL, exports = NULL,
  evals = NULL, ...)
```

Arguments

X	a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list
MARGIN	vector specifying the dimensions to use.
FUN	the function to be applied to each element of X
exports	A list containing a list for each environment which contains imports. Each list contains a character vector and a character or an expression. The first listing all Objects and Functions to be Exported. The second lists the associated environment. The environment in which the expression is evaluated is the environment of FUN.
evals	List of expressions to evaluate. See details for more information
...	additional arguments to pass to FUN: beware of partial matching to earlier arguments.

Details

If your are using madlapply insice a calcFunction you don't have to export or evaluate any objects and functions or packages.

Value

A list with one entry for each element in X

Author(s)

Stephen Wirth, Jan Philipp Dietrich

Examples

```
library(madrat)
library(magclass)

input <- array(2, c(10,5,1))
wraper <- function(){
# create a variable which has to be exported to the workers
# madlapply call
 resultnopar <- madapply(X=input,MARGIN=c(1,2), FUN=mean, #stating X and FUN
               exports=list(list(c("input"),expression(environment()))),
                # listing the objects or function
                #to be exported and their origin environments
              evals=c("magclass" )) # libraries to be evaluated
                return(resultnopar)
 }

 res <- wraper()
```

madlapply *madlapply*

Description

Wrapper Function that executes multiple function calls on one core using lapply or on the supplied number of cores using parLapply if the parallel option is set to TRUE. Make sure the package parallel is installed and runs on your machine before setting parallel to TRUE.

Usage

```
madlapply(X = NULL, FUN = NULL, exports = NULL, evals = NULL, ...)
```

Arguments

X	a vector (atomic or list) or an expression object. Other objects (including classed objects) will be coerced by base::as.list
FUN	the function to be applied to each element of X
exports	A list containing a list for each environment which contains imports. Each list contains a character vector and a character or an expression. The first listing all Objects and Functions to be Exported. The second lists the associated environment. The environment in which the expression is evaluated is the environment of FUN.
evals	List of expressions to evaluate. See details for more information
...	additional arguments to pass to FUN: beware of partial matching to earlier arguments.

Details

If your are using madlapply insice a calcFunction you don't have to export or evaluate any objects and functions or packages.

Value

A list with one entry for each element in X

Author(s)

Stephen Wirth, Jan Philipp Dietrich

Examples

```
library(madrat)
library(magclass)

input <- array(2, c(10,5,1))
```

```
powwrap <- function(){
# create a variable which has to be exported to the workers
pow <- function(exponent){
 # as magclass is not part of base, therefore the library
 #(magclass has to be evaluated on all the workers)
 return(as.magpie(
 input^exponent))
 }

 #actuall madlapply call
 resultnopar <- madlapply(X=c(2:10), FUN=pow, #stating X and FUN
               exports=list(list(c("input"),expression(environment()))),
                # listing the objects or function
                #to be exported and their origin environments
              evals=c("magclass" )) # libraries to be evaluated

                return(resultnopar)
 }

 res <- powwrap()
```

---

plotDiagnostics *Diagnostics plot*

Description

Function that plot diagnostic data from a madrat run and shows the network of function executions and distribution of time consumption

Usage

```
plotDiagnostics(filename, scale = "sqrt", cumulate_time = FALSE,
  png = NULL)
```

Arguments

filename	path to a diagnostics file as created with the setConfig argument diagnostics
scale	defines how time should scale the size of the vertices in the network. Available options are linear (lin), logarithmic (log) and square-root (sqrt).
cumulate_time	if set to TRUE the size of each node will reflect the cumulated execution time of the corresponding function (including the execution time of all underlying functions). If set to FALSE it will only show the net time spend in this function (excluding underlying functions).
png	png file name to save the plot as a png. If !NULL the function will wait until the interactive plot window will be closed and will then print the picture as it was in this window (still experimental!).

Author(s)

Jan Philipp Dietrich

See Also

toolstartmessage, setConfig

Examples

```
## Not run:
plotDiagnostics("diagnostics.csv")

## End(Not run)
```

---

prepFunctionName *prepFunctionName*

Description

Function to prepare a function call for a given type and prefix

Usage

```
prepFunctionName(type, prefix = "calc", ignore = NULL,
  error_on_missing = TRUE)
```

Arguments

type	name of calculation/source
prefix	Type of calculations. Available options are "download" (source download), "read" (source read), "correct" (source corrections), "convert" (source conversion to ISO countries), "calc" (further calculations), and "full" (collections of calculations)
ignore	vector of arguments which should be ignored (not be part of the function call)
error_on_missing	boolean deciding whether a missing type should throw an error or return NULL

Value

A function call as character to the specified function with corresponding package as attribute

Author(s)

Jan Philipp Dietrich

See Also

readSource, setConfig

Examples

```
print(madrat:::prepFunctionName("Tau","read"))
print(madrat:::prepFunctionName("TauTotal","calc"))
print(madrat:::prepFunctionName("EXAMPLE","full"))
```

---

readSource *readSource*

Description

Read in a source file and convert it to a MAgPIE object. The function is a wrapper for specific functions designed for the different possible source types.

Usage

```
readSource(type, subtype = NULL, convert = TRUE)
```

Arguments

type	source type, e.g. "IEA". A list of all available source types can be retrieved with function getSources.
subtype	For some sources there are subtypes of the source, for these source the subtype can be specified with this argument. If a source does not have subtypes, subtypes should not be set.
convert	Boolean indicating whether input data conversion should be done or not. In addition it can be set to "onlycorrect" for sources with a separate correctXXX-function.

Value

magpie object with the temporal and data dimensionality of the source data. Spatial will either agree with the source data or will be on ISO code country level depending on your choice for the argument "convert".

Author(s)

Jan Philipp Dietrich, Anastasis Giannousakis, Lavinia Baumstark

See Also

setConfig, ' downloadSource, readTau

Examples

```
## Not run:
a <- readSource("Tau","paper")

## End(Not run)
```

`readTau` *Read Tau*

Description

Read-in landuse intensity data (tau) following the methodology published in Dietrich J.P., Schmitz C., Mueller C., Fader M., Lotze-Campen H., Popp A., Measuring agricultural land-use intensity - A global analysis using a model-assisted approach, Ecological Modelling, Volume 232, 10 May 2012, Pages 109-118, ISSN 0304-3800, 10.1016/j.ecolmodel.2012.03.002. The data itself comes from the script prepare_data.R stored in the PIK svn at http://subversion/svn/magpie/paper/ti-paper/sonstiges/scripts/scripts2011. The file which is normally used is called tau_data_1995-2005.mz.

Usage

```
readTau(subtype = "paper")
```

Arguments

`subtype`	Type of Tau data that should be read. Available types are: • `paper`: numbers as they are reported in the paper (cellular, crop-specific) • `historical`: historic tau values on iso country level for total tau factor. This numbers were calculated by taking FAO yields and norming it to the 1995 tau values of the paper (faoyields*tau95/mean(faoyields[1995:2005]))

Value

Tau data and weights as MAgPIE object in original resolution

Author(s)

Jan Philipp Dietrich

See Also

`readSource`

Examples

```
## Not run:  a <- readSource("Tau")
```

regionscode *Tool: regionscode*

Description

Given a regionmapping (mapping between ISO countries and regions) the function calculates a regionscode which is basically the md5sum of a reduced form of the mapping. The regionscode is unique for each regionmapping and can be used to clearly identify a given regionmapping. In addition several checks are performed to make sure that the given input is a proper regionmapping

Usage

```
regionscode(mapping)
```

Arguments

mapping	Either a path to a mapping or an already read-in mapping as data.frame.

Value

A md5-based regionscode which describes the given mapping

Author(s)

Jan Philipp Dietrich

See Also

fingerprint, digest

Examples

```
file <- system.file("extdata","regionmappingH12.csv",package="madrat")
regionscode(file)
```

retrieveData *retrieveData*

Description

Function to retrieve a predefined collection of calculations for a specific regionmapping.

Usage

```
retrieveData(model, rev = 0, cachetype = "rev", ...)
```

Arguments

model	The names of the model for which the data should be provided (e.g. "magpie").
rev	data revision which should be used/produced (positive numeric).
cachetype	defines what cache should be used. "rev" points to a cache shared by all calculations for the given revision, "def" points to the cache as defined in the current settings and "tmp" temporarily creates a cache folder for the calculations and deletes it afterwards again
...	(Optional) Settings that should be changed using setConfig (e.g. regionmapping).

Author(s)

Jan Philipp Dietrich, Lavinia Baumstark

See Also

calcOutput,setConfig

Examples

```
## Not run:
retrieveData("magpie",rev=2,regionmapping="regionmappingMAgPIE.csv")

## End(Not run)
```

setConfig *setConfig*

Description

This function manipulates the current madrat configuration. In general, NULL means that the argument remains as it is whereas all other inputs will overwrite the current setting.

Usage

```
setConfig(regionmapping = NULL, packages = NULL, globalenv = NULL,
  enablecache = NULL, verbosity = NULL, mainfolder = NULL,
  sourcefolder = NULL, cachefolder = NULL, mappingfolder = NULL,
  outputfolder = NULL, pop_threshold = NULL, forcecache = NULL,
  ignorecache = NULL, delete_cache = NULL, diagnostics = NULL,
  nocores = NULL, .cfgchecks = TRUE, .verbose = TRUE)
```

Arguments

regionmapping	The name of the csv file containing the region mapping that should be used for aggregation (e.g. "regionmappingREMIND.csv").
packages	A character vector with packages in which corresponding read and calc functions should be searched for
globalenv	Boolean deciding whether sources/calculations in the global environment should be included or not
enablecache	Boolean deciding whether data should be read from cache if data is available and the up-to-date (data will always be written to the cache regardless of this setting)
verbosity	an integer value describing the verbosity of the functions (2 = full information, 1 = only warnings and execution information, 0 = only warnings, -1 = no information)
mainfolder	The mainfolder where all data can be found and should be written to.
sourcefolder	The folder in which all source data is stored (in sub-folders with the name of the source as folder name). In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/sources
cachefolder	The folder in which all cache files should be written to. In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/cache
mappingfolder	A folder containing all kinds of mappings (spatial, temporal or sectoral). In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/mappings
outputfolder	The folder all outputs should be written to. In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/output

pop_threshold	Population threshold in capita which determines whether the country is put into the "important" or "dispensable" class in getISOlist. This distinction is used for different treatment of countries in notifications to set a focus on rather critical issues instead of flooding the user with information.
forcecache	Argument that allows to force madrat to read data from cache if the corresponding cache files exist. It is either a boolean to fully activate or deactivate the forcing or a vector of files (e.g. readTau, calcTauTotal) or type (e.g. Tau, TauTotal) that should be read from cache in any case.
ignorecache	Argument that allows madrat to ignore the forcecache argument for the given vector of files (e.g. readTau, calcTauTotal) or types (e.g. Tau, TauTotal) called by calcOutput or readSource. The top level function must always be part of this list.
delete_cache	Boolean deciding whether a temporary cache folder (as created by retrieveInput) should be deleted after completion or not.
diagnostics	file name for additional diagnostics information (without file ending). 3 diagnostic files will be written if a file name is provided (a csv showing the network of function executions, a log file showing the log and a full log showing the full amount of available information.)
nocores	integer number of cores to use for clusterApply calls
.cfgchecks	boolean deciding whether the given inputs to setConfig should be checked for consistency or just be accepted (latter is only necessary in very rare cases and should not be used in regular cases)
.verbose	boolean deciding whether status information/updates should be shown or not

Author(s)

Jan Philipp Dietrich

See Also

getConfig, getISOlist

Examples

```
 ## Not run:
   setConfig(forcecache=c("readSSPall","convertSSPall"))

## End(Not run)
```

toolAggregate *toolAggregate*

Description

(Dis-)aggregates a magclass object from one resolution to another based on a relation matrix or mapping

Usage

```
toolAggregate(x, rel, weight = NULL, from = NULL, to = NULL,
  dim = 1, partrel = FALSE, negative_weight = "warn",
  mixed_aggregation = FALSE, verbosity = 1)
```

Arguments

x	magclass object that should be (dis-)aggregated
rel	relation matrix, mapping or csv file containing a mapping. A mapping object should contain 2 columns in which each element of x is mapped to the category it should belong to after (dis-)aggregation
weight	magclass object containing weights which should be considered for a weighted aggregation. The provided weight should only contain positive values, but does not need to be normalized (any positive number>=0 is allowed). Please see the "details" section below for more information.
from	Name of the first column to be used in rel if it is a mapping (if not set the first or second column will be used).
to	Name of the second column to be used in rel if it is a mapping (if not set the second or third column will be used).
dim	Specifying the dimension of the magclass object that should be (dis-)aggregated. Either specified as an integer (1=spatial,2=temporal,3=data) or if you want to specify a sub dimension specified by name of that dimension or position within the given dimension (e.g. 3.2 means the 2nd data dimension, 3.8 means the 8th data dimension).
partrel	If set to TRUE allows that the relation matrix does contain less entries than x and vice versa. These values without relation are lost in the output.
negative_weight	Describes how a negative weight should be treated. "allow" means that it just should be accepted (dangerous), "warn" returns a warning and "stop" will throw an error in case of negative values
mixed_aggregation	boolean which allows for mixed aggregation (weighted mean mixed with summations). If set to TRUE weight columns filled with NA will lead to summation.
verbosity	Verbosity level of messages coming from the function: -1 = error, 0 = warning, 1 = note, 2 = additional information, >2 = no message

Details

Basically toolAggregate is doing nothing more than a normal matrix multiplication which is taking into account the 3 dimensional structure of MAgPIE objects. So, you can provide any kind of relation matrix you would like. However, for easier usability it is also possible to provide weights for a weighted (dis-)aggregation as a MAgPIE object. In this case rel must be a 1-0-matrix or a mapping between both resolutions. The weight needs to be provided in the higher spatial aggregation, meaning for aggregation the spatial resolution of your input data and in the case of disaggregation the spatial resolution of your output data. The temporal and data dimension must be either identical to the resolution of the data set that should be (dis-)aggregated or 1. If the temporal and/or data dimension is 1 this means that the same transformation matrix is applied for all years and/or all data columns. In the case that a column should be just summed up instead of being calculated as a weighted average you either do not provide any weight (than all columns are just summed up) or your set this specific weighting column to NA and mixed_aggregation to TRUE.

Value

the aggregated data in magclass format

Author(s)

Jan Philipp Dietrich, Ulrich Kreidenweis

See Also

calcOutput

Examples

```
# create example mapping
mapping <- data.frame(from=getRegions(population_magpie),to=rep(c("REG1","REG2"),5))
mapping

# run aggregation
toolAggregate(population_magpie,mapping)
# weighted aggregation
toolAggregate(population_magpie,mapping, weight=population_magpie)
```

---

toolConvertMapping *Tool: ConvertMapping*

Description

Function which converts mapping files between formats

Usage

```
toolConvertMapping(name, format = "rda", type = NULL,
  where = "mappingfolder")
```

Arguments

name	File name of the mapping file. Supported file types are currently csv (, or ; separated) and rda (which needs to have the data stored with the object name "data"!). Use codetoolConvertMapping to convert between both formats
format	format it should be converted to. Available is "csv" or "rda".
type	Mapping type (e.g. "regional", "cell", or "sectoral"). Can be set to NULL if file is not stored in a type specific subfolder
where	location to look for the mapping, either "mappingfolder" or the name of a package which contains the mapping

Value

the mapping as a data frame

Author(s)

Jan Philipp Dietrich

See Also

calcOutput, toolConvertMapping

Examples

```
toolGetMapping("regionmappingH12.csv", where="madrat")
```

---

toolCountry2isocode *toolCountry2isocode*

Description

Function used to convert country names from the long name to the ISO 3166-1 alpha 3 country code

Usage

```
toolCountry2isocode(country, warn = TRUE, type = "IEA",
  mapping = NULL)
```

Arguments

country	A vector of country names
warn	whether warnings should be printed now or in the end of the whole process as notes
type	the name of the source type
mapping	additional mappings as a names vector

Value

the ISO 3166-1 alpha 3 country code

Author(s)

Jan Philipp Dietrich, Anastasis Giannousakis

See Also

readSource,getSources

Examples

```
toolCountry2isocode("Germany")
toolCountry2isocode(c("Germany","Fantasyland"),mapping=c("Fantasyland"="BLA"))
```

---

toolCountryFill *Tool: CountryFill*

Description

This function expects a MAgPIE object with ISO country codes in the spatial dimension. These ISO codes are compared with the official ISO code country list (stored as supplementary data in the madrat package). If there is an ISO code in the data but not in the official list this entry is removed, if an entry of the official list is missing in the data this entry is added and set to the value of the argument fill.

Usage

```
toolCountryFill(x, fill = NA, no_remove_warning = NULL,
  overwrite = FALSE, verbosity = 1, ...)
```

Arguments

x	MAgPIE object with ISO country codes in the spatial dimension
fill	Number which should be used for filling the gaps of missing countries.
no_remove_warning	A vector of non-ISO country codes that exist in the data and that should be removed by CountryFill but without creating a warning (they will be removed in any case). You should use that argument if you are certain that the given entries should be actually removed from the data.
overwrite	logical deciding whether existing data should be overwritten, if there is a specific mapping provided for that country, or not

verbosity	verbosity for information about filling important countries. 0 = warning will show up (recommended if filling of important countries is not expected), 1 = note will show up in reduced log file (default), 2 = info will show up in extended log file (recommended if filling of important countries is not critical and desired).
...	Mappings between countries for which the data is missing and countries from which the data should be used instead for these countries (e.g. "HKG"="CHN" if HongKong should receive the value of China). This replacement usually only makes sense for intensive values.

Value

A MAgPIE object with spatial entries for each country of the official ISO code country list.

Author(s)

Jan Philipp Dietrich

Examples

```
library(magclass)
x <- new.magpie("DEU",1994,"bla",0)
y <- toolCountryFill(x,99)
```

---

toolendmessage *Tool: End message*

Description

Function writes a process end message and performs some diagnostics

Usage

```
toolendmessage(startdata, level = NULL, id = "none")
```

Arguments

startdata	a list containing diagnostic information provided by toolstartmessage
level	This argument allows to establish a hierarchy of print statements. The hierarchy is preserved for the next vcat executions. Currently this setting can have 4 states: NULL (nothing will be changed), 0 (reset hierarchies), "+" (increase hierarchy level by 1) and "-" (decrease hierarchy level by 1).
id	additional id which uniquely identifies the process that just has been finished

Author(s)

Jan Philipp Dietrich

See Also

toolstartmessage, vcat

Examples

```
## Not run:
tmp <- function(bla=NULL) {
  startinfo <- toolstartmessage("+")
  print(bla)
  toolendmessage(startinfo,"-")
  }
tmp(bla=99)

## End(Not run)
```

---

toolFillWithRegionAvg *Tool: FillWithRegionAvg*

Description

This function fills missing values for countries with the (weighted) average of the respective region. The average is computed separately for every timestep. Currently only inputs with one data dimension are allowed as inputs. (If the filling should be performed over multiple data dimensions, call this function multiple times and bind the results together with magclass::mbind.)

Usage

```
toolFillWithRegionAvg(x, valueToReplace = NA, weight = NULL,
  callToolCountryFill = FALSE, regionmapping = NULL, verbose = TRUE,
  warningThreshold = 0.5)
```

Arguments

x	MAgPIE object with country codes in the first and time steps in the second dimension.
valueToReplace	value that denotes missing data. Defaults to NA.
weight	MAgPIE object with weights for the weighted average. Must contain at least all the countries and years present in x. If no weights are specified, an unweighted average is performed.

callToolCountryFill
: Boolean variable indicating whether the list of countries should first be filled to the official ISO code country list. Subsequently the newly added and previously missing values are filled with the region average.

regionmapping
: Data frame containing the mapping between countries and regions. Expects column names CountryCode and RegionCode. Uses the currently set mapping (as returned by toolMappingFile) if no mapping is specified.

verbose
: Boolean variable indicating if the function should print out what it is doing. Can generate a lot of output for a large object.

warningThreshold
: If more than this fraction of the countries in a given region and timestep have a missing value, throw a warning.

Details

toolFillWithRegionAvg can be used in conjunction with toolCountryFill() to first fill up the list of countries to the official ISO code country list, and then fill values with the regional average (see callToolCountryFill Option).

Value

A MAgPIE object with the missing values filled.

Author(s)

Bjoern Soergel, Lavinia Baumstark

Examples

```
x <- magclass::new.magpie(cells_and_regions = c('A','B','C','D'), years = c(2000,2005),
fill = c(1,NA,3,4,5,6,NA,8))
rel <- data.frame(CountryCode = c('A','B','C','D'), RegionCode = c('R1','R1','R1','R2'))
xfilled <- toolFillWithRegionAvg(x, regionmapping = rel)
```

---

toolGetMapping *Tool: GetMapping*

Description

Function which retrieves a mapping file

Usage

```
toolGetMapping(name, type = NULL, where = "mappingfolder",
  error.missing = TRUE, returnPathOnly = FALSE)
```

Arguments

name	File name of the mapping file. Supported file types are currently csv (, or ; separated) and rda (which needs to have the data stored with the object name "data"!). Use codetoolConvertMapping to convert between both formats
type	Mapping type (e.g. "regional", "cell", or "sectoral"). Can be set to NULL if file is not stored in a type specific subfolder
where	location to look for the mapping, either "mappingfolder" or the name of a package which contains the mapping
error.missing	Boolean which decides whether an error is returned if the mapping file does not exist or not.
returnPathOnly	If set to TRUE only the file path is returned

Value

the mapping as a data frame

Author(s)

Jan Philipp Dietrich

See Also

calcOutput, toolConvertMapping

Examples

```
toolGetMapping("regionmappingH12.csv", where="madrat")
```

---

toolISOhistorical *Tool: ISOhistorical*

Description

This function expects a MAgPIE object with ISO country codes in the spatial dimension. For this MAgPIE object the time of transition is calculated and for each the historic time filled by using the mapping stored as supplementary data in the madrat package. If you want to use a different mapping please specify it in the argument mapping

Usage

```
toolISOhistorical(m, mapping = NULL, additional_mapping = NULL,
  overwrite = FALSE)
```

Arguments

m	MAgPIE object with ISO country codes in the spatial dimension
mapping	mapping of historical ISO countries to the standard ISO country list. For the default setting (mapping=NULL) the mapping stored as supplementary data in the madrat package is used.
additional_mapping	vector or list of vectors to provide some specific mapping, first the old country code, second the new country code and last the last year of the old country, e.g. additional_mapping = c("TTT","TTX","y1111") or additional_mapping = list(c("TTT","TTX","y1111"),c("TTT","TTY","y1111"))
overwrite	if there are already historical data in the data source for years that are calculated in this function they will not be overwritten by default. To overwrite all data (e.g. if there are meaningless "0") choose overwrite=TRUE

Value

A MAgPIE object with spatial entries for each country of the official ISO code country list. Historical time is filled up, old countries deleted

Author(s)

Lavinia Baumstark

---

toolMappingFile *Tool: MappingFile*

Description

Functions which calculates the path to a mapping file

Usage

```
toolMappingFile(type, name, readcsv = FALSE, error.missing = TRUE,
  where = "mappingfolder")
```

Arguments

type	Mapping type ("regional", or "sectoral")
name	File name of the mapping file.
readcsv	if true, file read in
error.missing	Boolean which decides whether an error is returned if the mapping file does not exist or not.
where	location to look for the mapping, either "mappingfolder" or the name of a package which contains the mapping

Value

The complete path to the mapping file.

Author(s)

Jan Philipp Dietrich

See Also

calcOutput

Examples

```
## Not run: toolMappingFile("sectoral","structuremappingFE.csv")
```

---

toolNAreplace *Tool: NA replace*

Description

Functions removes NAs, NaNs and infinite values in x and weight

Usage

```
toolNAreplace(x, weight = NULL, replaceby = 0, val.rm = NULL)
```

Arguments

x	data
weight	aggregation weight
replaceby	value which should be used instead of NA. Either a single value or a MAgPIE object which can be expanded to the size of x (either same size or with lower dimensionality).
val.rm	vector of values that should in addition be removed in x

Value

a list containing x and weight

Author(s)

Benjamin Bodirsky, Jan Philipp Dietrich

See Also

calcOutput

`toolstartmessage` *Tool: Start message*

Description

Function writes a process start message and performs some diagnostics

Usage

```
toolstartmessage(level = NULL)
```

Arguments

`level`	This argument allows to establish a hierarchy of print statements. The hierarchy is preserved for the next vcat executions. Currently this setting can have 4 states: NULL (nothing will be changed), 0 (reset hierarchies), "+" (increase hierarchy level by 1) and "-" (decrease hierarchy level by 1).

Value

a list containing diagnostic information required by `toolendmessage`

Author(s)

Jan Philipp Dietrich

See Also

`toolendmessage`, `vcat`

Examples

```
## Not run:
tmp <- function(bla=NULL) {
  startinfo <- toolstartmessage("+")
  print(bla)
  toolendmessage(startinfo,"-")
  }
tmp(bla=99)

## End(Not run)
```

`toolSubtypeSelect` *Tool: SubtypeSelect*

Description

This function is a support function for the selection of a subtype in a readX function. In addition to the subtype selection it also performs some consistency checks.

Usage

```
toolSubtypeSelect(subtype, files)
```

Arguments

`subtype`	A chosen subtype (character)
`files`	A named vector. The names of the vector correspond to the allowed subtypes and the content of the vector are the corresponding file names.

Value

The file name corresponding to the given subtype

Author(s)

Jan Philipp Dietrich

See Also

`readSource`

Examples

```
subtype <- "extent"

files <-  c(protection="protection.csv",
             production="production.csv",
             extent="forest_extent.csv")
## Not run:
file <- toolSubtypeSelect(subtype,files)

## End(Not run)
```

toolXlargest *toolXlargest*

Description

Selects the largest countries of a country-level magpie object

Usage

```
toolXlargest(type, range = 1:20, years = NULL, elements = NULL, ...)
```

Arguments

type	calcOutput function that shall be used for ranking
range	the position of the countries in the top X which should be returned.
years	range of years that shall be summed for ranking. If NULL, the sum of all years is used.
elements	range of elements that shall be summed for ranking. If NULL, all elements are used.
...	further parameters will be handed on to calcOutput function type.

Value

vector with ISO country codes

Author(s)

Benjamin Leon Bodirsky, Jan Philipp Dietrich

Examples

```
## Not run:
top10 <- toolXlargest(type="TauTotal",range=1:10)

## End(Not run)
```

vcat *Tool: Verbosity Cat*

Description

Function which returns information based on the verbosity setting

Usage

```
vcat(verbosity, ..., level = NULL, fill = TRUE, show_prefix = TRUE)
```

Arguments

verbosity	The lowest verbosity level for which this message should be shown (verbosity = -1 means no information at all, 0 = only warnings, 1 = warnings and execution informations, 2 = full information). If the verbosity is set to 0 the message is written as warning, if the verbosity is set higher than 0 it is written as a normal cat message.
...	The message to be shown
level	This argument allows to establish a hierarchy of print statements. The hierarchy is preserved for the next vcat executions. Currently this setting can have 4 states: NULL (nothing will be changed), 0 (reset hierarchies), "+" (increase hierarchy level by 1) and "-" (decrease hierarchy level by 1).
fill	a logical or (positive) numeric controlling how the output is broken into successive lines. If FALSE (default), only newlines created explicitly by "\n" are printed. Otherwise, the output is broken into lines with print width equal to the option width if fill is TRUE, or the value of fill if this is numeric. Non-positive fill values are ignored, with a warning.
show_prefix	a logical defining whether a content specific prefix (e.g. "NOTE") should be shown in front of the message or not. If prefix is not shown it will also not show up in official statistics.

Author(s)

Jan Philipp Dietrich

See Also

readSource

Examples

```
## Not run:
vcat(2,"Hello world!")

## End(Not run)
```

Index