

# Package: madrat (via r-universe)

July 2, 2026

**Type** Package

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

**Version** 3.37.1

**Date** 2026-07-01

**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.

**License** BSD\_2\_clause + file LICENSE

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

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

**Depends** magclass (>= 7.2.0), R (>= 2.10.0)

**Imports** callr, digest, filelock (>= 1.0.3), igraph (>= 2.1.1), Matrix, methods, pkgload, renv, stringi, tools, utils, withr, yaml

**Suggests** covr, ggplot2, graphics, grDevices, knitr, rmarkdown, terra, testthat, tibble

**VignetteBuilder** knitr

**Config/Keywords** tool

**Encoding** UTF-8

**Config/Needs/website** tidyverse/tidytemplate

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Jan Philipp Dietrich [aut, cre] (affiliation: Potsdam Institute for Climate Impact Research, ORCID: <<https://orcid.org/0000-0002-4309-6431>>), Pascal Sauer [aut] (affiliation: Potsdam Institute for Climate Impact Research, ORCID: <<https://orcid.org/0000-0002-6856-8239>>), Lavinia Baumstark [aut] (affiliation: Potsdam Institute for Climate Impact Research), Stephen Wirth [aut] (affiliation: Potsdam Institute for Climate Impact Research), Anastasis Giannousakis [aut], Renato Rodrigues [aut] (affiliation: Potsdam Institute for Climate Impact Research), Benjamin Leon Bodirsky [aut] (affiliation: Potsdam Institute for Climate Impact Research), Debbora Leip [aut] (affiliation: Potsdam Institute for Climate Impact Research), Michael Crawford [aut] (affiliation: Potsdam Institute for Climate Impact Research), Ulrich Kreidenweis [aut], David Klein [aut] (affiliation: Potsdam Institute for Climate Impact Research), Patrick Rein [aut] (affiliation: Potsdam Institute for Climate Impact Research, ORCID: <<https://orcid.org/0000-0001-9454-8381>>)

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

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-07-02 07:20:02 UTC

**RemoteUrl** <https://github.com/cran/madrat>

**RemoteRef** HEAD

**RemoteSha** f2136656ecd1b7f9ba1830c760ad27f0ffd6faa

## Contents

madrat-package . . . . .	4
addMapping . . . . .	5
cacheCleanup . . . . .	6
cacheCopy . . . . .	7
calcOutput . . . . .	7
calcTauTotal . . . . .	11
compareData . . . . .	11
compareMadratOutputs . . . . .	12
convertTau . . . . .	13
downloadSource . . . . .	14
findBottlenecks . . . . .	15
fullEXAMPLE . . . . .	16
getCalculations . . . . .	17
getConfig . . . . .	18
getDependencies . . . . .	19
getFromComment . . . . .	20
getISOList . . . . .	21
getLinkFunction . . . . .	22
getLocation . . . . .	22

getMadratGraph . . . . .	23
getMadratInfo . . . . .	24
getMadratMessage . . . . .	24
getSources . . . . .	25
installedMadratUniverse . . . . .	26
madratAttach . . . . .	27
metadataGFZ . . . . .	28
pucAggregate . . . . .	29
putMadratMessage . . . . .	30
readSource . . . . .	31
readTau . . . . .	32
redirectSource . . . . .	33
redirectTau . . . . .	35
regionscode . . . . .	35
resetMadratMessages . . . . .	36
retrieveData . . . . .	37
returnCalcOutput . . . . .	39
setConfig . . . . .	41
toolAggregate . . . . .	44
toolCodeLabels . . . . .	47
toolCompareStatusLogs . . . . .	48
toolConditionalReplace . . . . .	49
toolConvertMapping . . . . .	49
toolCountry2isocode . . . . .	50
toolCountryFill . . . . .	51
toolFillWithRegionAvg . . . . .	53
toolFillYears . . . . .	54
toolFixWeight . . . . .	55
toolGetMapping . . . . .	56
toolISOhistorical . . . . .	57
toolManualDownload . . . . .	58
toolNAreplace . . . . .	59
toolOrderCells . . . . .	60
toolSplitSubtype . . . . .	61
toolSubtypeSelect . . . . .	62
toolTimeAverage . . . . .	63
toolTimeSpline . . . . .	63
toolXlargest . . . . .	64
vcat . . . . .	65
visualizeDependencies . . . . .	66

---

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 and 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.

## Author(s)

**Maintainer:** Jan Philipp Dietrich <dietrich@pik-potsdam.de> ([ORCID](#)) (affiliation: Potsdam Institute for Climate Impact Research)

Authors:

- Jan Philipp Dietrich <dietrich@pik-potsdam.de> ([ORCID](#)) (affiliation: Potsdam Institute for Climate Impact Research)
- Pascal Sauer <pascal.sauer@pik-potsdam.de> ([ORCID](#)) (affiliation: Potsdam Institute for Climate Impact Research)
- Lavinia Baumstark <lavinia@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Stephen Wirth <wirth@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Anastasis Giannousakis <giannou@pik-potsdam.de>
- Renato Rodrigues <Renato.Rodrigues@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Benjamin Leon Bodirsky <bodirsky@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Debhora Leip <leip@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Michael Crawford <crawford@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Ulrich Kreidenweis <kreidenweis@pik-potsdam.de>
- David Klein <dklein@pik-potsdam.de> (affiliation: Potsdam Institute for Climate Impact Research)
- Patrick Rein <patrick.rein@pik-potsdam.de> ([ORCID](#)) (affiliation: Potsdam Institute for Climate Impact Research)

**See Also**

Useful links:

- <https://github.com/pik-piam/madrat>
- [doi:10.5281/zenodo.1115490](https://doi.org/10.5281/zenodo.1115490)
- Report bugs at <https://github.com/pik-piam/madrat/issues>

---

addMapping

*addMapping*

---

**Description**

Function which adds another mapping to the current list of extramappings in the madrat configuration (see [setConfig](#)) and stores the mapping in the mapping folder as well as output folder.

**Usage**

```
addMapping(filename, mapping = NULL)
```

**Arguments**

filename	The name of the the region mapping that should added including file ending (e.g. "regionmappingREMIND.csv"). Supported formats are currently ".csv" and ".rds".
mapping	Mapping provided as data.frame, or NULL. If a mapping is provided the data will be written in the mapping file of the given file (potentially replacing existing data). If NULL the mapping from the given file is used.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[setConfig](#)

Other mapping: [toolConvertMapping\(\)](#), [toolGetMapping\(\)](#)

**Examples**

```
## Not run:  
addMapping("regionmappingH12.csv")  
  
## End(Not run)
```

---

`cacheCleanup`*cacheCleanup*

---

**Description**

Delete files older than the specified number of days, based on file time metadata (per default atime = last access time).

**Usage**

```
cacheCleanup(  
  daysThreshold,  
  path = getConfig("cachefolder", verbose = FALSE),  
  timeType = c("atime", "mtime", "ctime"),  
  ask = TRUE,  
  readlineFunction = readline  
)
```

**Arguments**

<code>daysThreshold</code>	Files older than this many days are deleted/returned.
<code>path</code>	Path to where to look for old files.
<code>timeType</code>	Which file metadata time should be used. One of atime (last access time, default), mtime (last modify time), ctime (last metadata change).
<code>ask</code>	Whether to ask before deleting.
<code>readlineFunction</code>	Only needed for testing. A function to prompt the user for input.

**Details**

File time metadata is not available on all systems and the semantics are also system dependent, so please be careful and check that the correct files are deleted. This function will return a `data.frame` containing all files that would be deleted if the user answers 'n' to the question. If deleting files fails a warning is created.

**Value**

If the user answers 'n', a `data.frame` as returned by `base::file.info`, containing only files older than `<daysThreshold>` days.

**See Also**

Other cache management: [cacheCopy\(\)](#)

---

`cacheCopy`*cacheCopy*

---

**Description**

Copy cache files which were used for a given preprocessing.

**Usage**

```
cacheCopy(file, target = NULL, filter = NULL)
```

**Arguments**

<code>file</code>	Path to a log file or content of a log as character vector.
<code>target</code>	Folder to which the files should be copied. If NULL no data is copied.
<code>filter</code>	Regular expression to filter the cache files shown in the log file.

**Value**

A vector of cache files which match the given log information and filter.

**Author(s)**

Jan Philipp Dietrich

**See Also**

Other cache management: [cacheCleanup\(\)](#)

---

`calcOutput`*calcOutput*

---

**Description**

Calculate a specific output for which a calculation function exists. 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,
  signif = NULL,
  supplementary = FALSE,
  append = FALSE,
  warnNA = TRUE,
  na_warning = NULL,
  try = FALSE,
  regionmapping = NULL,
  writeArgs = NULL,
  temporalmapping = NULL,
  ...,
  outputStatistics = NULL
)
```

**Arguments**

type	output type, e.g. "TauTotal". A list of all available source types can be retrieved with function <a href="#">getCalculations</a> .
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	Number of decimal places to round to. Ignored if NULL. See <a href="#">round()</a> for details.
signif	Number of significant digits to round to. Ignored if NULL. See <a href="#">signif()</a> for details.
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.
warnNA	boolean deciding whether NAs in the data set should create a warning or not
na_warning	deprecated, please use warnNA instead
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 FALSE calculation will stop with an error in such a case. This setting will be overwritten by the global setting debug=TRUE, in which try will be always interpreted as TRUE.

regionmapping	alternative regionmapping to use for the given calculation. It will temporarily overwrite the global setting just for this calculation.
writeArgs	a list of additional, named arguments to be supplied to the corresponding write function
temporalmapping	to aggregate yearly values in the produced magpie object to period values. A data frame consisting of the columns 'period', 'year' and 'weight'. Years in the magpie object will be mapped from years to periods as indicated in temporalmapping by calculating the weighted average using the 'weight' column. Requires magpie object to have exactly one temporal sub-dimension.
...	Additional settings directly forwarded to the corresponding calculation function
outputStatistics	a single name of a statistic function ("summary", "sum", or "count") or a vector of such names that denote which statistics should be computed on the data before aggregation. Disabled by default.

### 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
- **class** (optional | default = "magpie") - Class of the returned object. If set to something other than "magpie" most functionality, such as aggregation or unit tests will not be available and is switched off!
- **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

- **structure.spatial** (optional) - regular expression describing the name structure of all names in the spatial dimension (e.g. `^[A-Z]{3}$`). Names will be checked against this regular expression and disagreements will be reported via a warning.
- **structure.temporal** (optional) - regular expression describing the name structure of all names in the temporal dimension (e.g. `^[0-9]{4}$`). Names will be checked against this regular expression and disagreements will be reported via a warning.
- **structure.data** (optional) - regular expression describing the name structure of all names in the data dimension (e.g. `^[a-z]*\\. [a-z]*$`). Names will be checked against this regular expression and disagreements will be reported via a warning.
- **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`.
- **putInPUC** (optional) boolean which decides whether this calculation should be added to a puc file which contains non-aggregated data and can be used to later on aggregate the data to resolutions of own choice. If not set `calcOutput` will try to determine automatically, whether a file is being required for the puc file or not, but in more complex cases (e.g. if calculations below top-level have to be run as well) this setting can be used to manually tweak the puc file list. CAUTION: Incorrect settings will cause corrupt puc files, so use this setting with extreme care and only if necessary.
- **cache** (optional) boolean which decides whether a cache file should be written (if caching is active) or not. Default setting is `TRUE`. This can be for instance useful, if the calculation itself is quick, but the corresponding file sizes are huge. Or if the caching for the given data type does not support storage in RDS format. CAUTION: Deactivating caching for a data set which should be part of a PUC file will corrupt the PUC file. Use with care.
- **clean\_magpie** (optional) boolean, if set to `FALSE` `magclass::clean_magpie` will not be run on the result

### Author(s)

Jan Philipp Dietrich, Patrick Rein

### See Also

[setConfig](#), [calcTauTotal](#),

Other core workflow: [downloadSource\(\)](#), [readSource\(\)](#), [retrieveData\(\)](#), [returnCalcOutput\(\)](#)

### Examples

```
## Not run:
a <- calcOutput(type = "TauTotal")

## End(Not run)
```

---

calcTauTotal	<i>Calculate total tau</i>
--------------	----------------------------

---

**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(source = "paper")
```

**Arguments**

source            data source, either "paper" (default) or "historical".

**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)
```

---

compareData	<i>compareData</i>
-------------	--------------------

---

**Description**

Compares the content of two data archives and looks for similarities and differences

**Usage**

```
compareData(x, y, tolerance = 10^-5, yearLim = NULL)
```

**Arguments**

x	Either a tgz file or a folder containing data sets
y	Either a tgz file or a folder containing data sets
tolerance	tolerance level below which differences will get ignored
yearLim	year until when the comparison should be performed. Useful to check if data is identical until a certain year.

**Author(s)**

Jan Philipp Dietrich, Florian Humpenoeder

**See Also**

[setConfig](#), [calcTauTotal](#),

Other validation: [compareMadratOutputs\(\)](#), [toolCompareStatusLogs\(\)](#)

---

compareMadratOutputs    *Compare a madrat function's output with and without your changes*

---

**Description**

With ‘compareMadratOutputs’ you can easily compare the output of a madrat function (read, calc, ...) with and without your changes. First, run ‘compareMadratOutputs’ without your changes, so a ‘.rds’ file with the original output is saved. Then apply your changes and run ‘compareMadratOutputs’ again to compare the new output to the original output.

**Usage**

```
compareMadratOutputs(package, functionName, subtypes, overwriteOld = FALSE)
```

**Arguments**

package	[character(1)] The package where the given function is located. It will be attached via ‘library’.
functionName	[character(1)] The name of the function from which you want to compare outputs. Must be a madrat function whose name starts with read, correct, convert, or calc.
subtypes	[character(n)] The subtypes you want to check. For calc functions this must be NULL.
overwriteOld	If TRUE: overwrite a "*-old-*.rds" previously created with compareMadratOutputs.

**Details**

If there are differences a ‘<functionName>-new.rds’ containing the new output is saved for closer inspection. All files are created in the current working directory.

**Value**

Invisibly the result of `'waldo::compare'` or `'all.equal'` if a comparison was made, otherwise a named list of the outputs for each subtype.

**Author(s)**

Pascal Sauer

**See Also**

Other validation: [compareData\(\)](#), [toolCompareStatusLogs\(\)](#)

**Examples**

```
## Not run:
# save original output to readTau-old.rds
compareMadratOutputs("madrat", "readTau", c("paper", "historical"))

# now apply your changes to madrat:::readTau, reinstall madrat, restart the R session

# compare new output to original output from readTau-old.rds
compareMadratOutputs("madrat", "readTau", c("paper", "historical"))

## 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

---

downloadSource	<i>downloadSource</i>
----------------	-----------------------

---

### Description

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

### Usage

```
downloadSource(type, subtype = NULL, overwrite = FALSE, numberOfTries = 300)
```

### Arguments

type	source type, e.g. "IEA". A list of all available source types can be retrieved with function <a href="#">getSources("download")</a> .
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.
overwrite	Boolean deciding whether existing data should be overwritten or not.
numberOfTries	Integer determining how often readSource will check whether a running download is finished before exiting with an error. Between checks readSource will wait 30 seconds. Has no effect if the sources that should be read are not currently being downloaded.

### Note

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

- **url** - full path to the file that should be downloaded
- **title** - title of the data source
- **author** - author(s) of the data set
- **license** - license of the data set. Put **unknown** if not specified.
- **description** - description of the data source
- **unit** - unit(s) of the data
- **doi** (optional) - a DOI URL to the data source
- **version** (optional) - version number of the data set
- **release\_date** (optional) - release date of the data set
- **reference** (optional) - A reference for the data set (e.g. a paper, if the data was derived from it)

This user-provided data is enriched by automatically derived metadata:

- **call** - Information about the used madrat function call to download the data will check whether there are any values below the given threshold and warn in this case
- **accessibility** - A measure of quality for the accessibility of the data. Currently it distinguished between **iron** (manual access), **silver** (automatic access via URL) and **gold** (automatic access via DOI).

Besides the names above (user-provided and automatically derived) it is possible to add custom metadata entries by extending the return list with additional, named entries.

### Author(s)

Jan Philipp Dietrich, David Klein, Pascal Sauer

### See Also

[setConfig](#), [readSource](#)

Other core workflow: [calcOutput\(\)](#), [readSource\(\)](#), [retrieveData\(\)](#), [returnCalcOutput\(\)](#)

### Examples

```
## Not run:
a <- downloadSource("Tau", subtype = "historical")

## End(Not run)
```

---

findBottlenecks	<i>findBottlenecks</i>
-----------------	------------------------

---

### Description

Analyzes a log from a retrieveData run, extracts runtime information for all called functions and identifies most critical bottlenecks.

### Usage

```
findBottlenecks(file, unit = "min", cumulative = TRUE)
```

### Arguments

file	path to a log file or content of a log as character vector
unit	unit for runtime information, either "s" (seconds), "min" (minutes) or "h" (hours)
cumulative	boolean deciding whether calls to the same function should be aggregated or not

### Value

A named list with one entry per retrieveData call found in the log. The names are the retrieveData types and each entry is a data.frame sorted by net runtime showing for the different data processing functions their total runtime "time" (including the execution of all sub-functions) and net runtime "net" (excluding the runtime of sub-functions) and their share of total runtime.

**Author(s)**

Jan Philipp Dietrich

---

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, dev = "", extra = "Example argument")
```

**Arguments**

rev	data revision which should be used/produced. Will be converted to <a href="#">numeric_version</a> when called via <a href="#">retrieveData</a> .
dev	development suffix to distinguish development versions for the same data revision. This can be useful to distinguish parallel lines of development.
extra	additional argument which - when changed - does not require a re-computation of the portable unaggregated collection (puc) file. <a href="#">setConfig</a> (e.g. for setting the mainfolder if not already set properly).

**Author(s)**

Jan Philipp Dietrich

**See Also**

[readSource](#), [getCalculations](#), [calcOutput](#), [setConfig](#)

**Examples**

```
## Not run:  
retrieveData("example", rev = "2.1.2", dev = "test", regionmapping = "regionmappingH12.csv")  
  
## End(Not run)
```

---

getCalculations	<i>getCalculations</i>
-----------------	------------------------

---

### 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, vector of types or search term (e.g. "readcalc"). 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](#)

Other available functions: [getLocation\(\)](#), [getSources\(\)](#), [installedMadratUniverse\(\)](#)

### 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, print = FALSE)
```

### 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
print	if TRUE and verbose is TRUE a configuration overview will also get printed

### Value

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

### Note

getConfig is primarily designed to make the overall madrat configuration available to system tools of the madrat framework. There are only a few exceptions for which configuration settings are also readable from within a download-, read-, convert-, correct-, calc- or full-function. These exceptions are the setting "debug" (which can be used to add additional debug messages when active), the "tmpfolder" which can be used to temporarily store data and the setting "hash" (which can only be accessed from within a full function and can there be used to apply the identical hash algorithm for other calculations in which hashing is being used). Besides that "regionmapping" and "extramappings" can also be read from within calc- and full-functions but their use is at least for the calc-functions discouraged as it either might lead to incorrect caching behavior, or - if implemented correctly - lead to significant slow-downs of overall calculations. All other settings are currently still accessible but trigger a warning that this option will soon be removed. So, please make sure that your code runs without reading these options! As a background note: Read access to these settings will be restricted as they otherwise would allow access to code elements or data in a form which is violating the overall madrat logic and thereby can lead to erroneous results.

**Author(s)**

Jan Philipp Dietrich

**See Also**[setConfig](#), [initializeConfig](#)Other madrat configuration: [madratAttach\(\)](#), [setConfig\(\)](#)

getDependencies

*getDependencies***Description**

Returns information about dependencies of a madrat-based calc- read- or full-function.

**Usage**

```

getDependencies(
    name,
    direction = "in",
    graph = NULL,
    type = NULL,
    self = FALSE,
    ...
)

```

**Arguments**

name	name of the function to be analyzed
direction	Character string, either "in", "out", "both", "full", "din" or "dout". If "in" all sources feeding into the function are listed. If "out" consumer of the function are listed. If "both" the union of "in" and "out" is returned. If "full" the full network this function is connected to is shown, including indirect connections to functions which neither source nor consume the given function but serve as sources to other consumer functions. "din" and "dout" (short for "direct in" and "direct out") behave like "in" and "out" but only show direct calls in or from the function (ignoring the network of functions attached to it).
graph	A madrat graph as returned by <a href="#">getMadratGraph</a> . Will be created with <a href="#">getMadratGraph</a> if not provided.
type	type filter. Only dependencies of that type will be returned. Currently available types are "calc", "read" and "tool"
self	boolean defining whether the function itself, which is analyzed, should be included in the output, or not
...	Additional arguments for <a href="#">getMadratGraph</a> in case that no graph is provided (otherwise ignored)

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getCalculations](#), [getMadratGraph](#), [getMadratInfo](#)

Other dependency graph: [getMadratGraph\(\)](#), [getMadratInfo\(\)](#), [visualizeDependencies\(\)](#)

---

getFromComment

*getFromComment*

---

**Description**

Helper function extract a metadata comment

**Usage**

```
getFromComment(x, name)
```

**Arguments**

x	object the metadata should be extracted from
name	name of the metadata to be extracted (e.g. unit)

**Author(s)**

Jan Philipp Dietrich

**Examples**

```
x <- as.magpie(1)
getComment(x) <- c(" description: example description", " unit: kg")
getFromComment(x, "unit")
getFromComment(x, "description")
```

---

getISOList	<i>get official ISO list</i>
------------	------------------------------

---

### 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", threshold = 1)
```

### 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.
threshold	Population threshold in million capita which determines whether the country is put into the "important" or "dispensable" class (default = 1 mio. people)

### 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

[getSource](#), [getCalculations](#)

Other country tools: [toolCountry2isocode\(\)](#), [toolCountryFill\(\)](#), [toolFillWithRegionAvg\(\)](#), [toolISOhistorical\(\)](#)

### Examples

```
head(getISOList())  
head(getISOList("dispensable"))
```

---

getLinkFunction	<i>getLinkFunction</i>
-----------------	------------------------

---

**Description**

Returns a function that creates a symlink, hardlink, junction, or copy of files and directories, depending on OS capabilities (usually symlinks are not supported on Windows).

**Usage**

```
getLinkFunction()
```

**Value**

A function with arguments "from" and "to" which should behave like file.symlink on all platforms.

**Author(s)**

Pascal Sauer

**See Also**

Other source redirection: [redirectSource\(\)](#)

---

getLocation	<i>getLocation</i>
-------------	--------------------

---

**Description**

Returns names of packages in which functions matching the description are found

**Usage**

```
getLocation(name, packages = installedMadratUniverse(), globalenv = TRUE)
```

**Arguments**

name	name of the function to be found. Can be either the full name (e.g. "calcTauTotal"), or just the type name (e.g. "TauTotal").
packages	A character vector with packages in which should be looked for the function
globalenv	Boolean deciding whether functions in the global environment should be included or not

**Value**

vector of packages in which a function matching the description could be found

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getCalculations](#), [getDependencies](#)

Other available functions: [getCalculations\(\)](#), [getSources\(\)](#), [installedMadratUniverse\(\)](#)

---

getMadratGraph

*getMadratGraph*

---

**Description**

Function returns the madrat graph of all linkages of full, calc, and read functions of the given madrat based packages. Linkages to subfunctions of read functions (i.e. download, correct or convert functions) are not listed separately, but collectively referred to through the corresponding read function.

**Usage**

```
getMadratGraph(  
  packages = installedMadratUniverse(),  
  globalenv = getConfig("globalenv")  
)
```

**Arguments**

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 with 4 columns: from (source function), from\_package (package the source function originates from), to (function which is using the source), to\_package (package of the using function)

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getCalculations](#), [getConfig](#)

Other dependency graph: [getDependencies\(\)](#), [getMadratInfo\(\)](#), [visualizeDependencies\(\)](#)

---

getMadratInfo            *getMadratInfo*

---

### Description

Collects and returns detailed information about the currently loaded network of madrat functions.

### Usage

```
getMadratInfo(graph = NULL, cutoff = 5, extended = FALSE, ...)
```

### Arguments

graph	A madrat graph as returned by <a href="#">getMadratGraph</a> . Will be created with <a href="#">getMadratGraph</a> if not provided.
cutoff	Integer introducing a cutoff of items to be returned for outputs which can become quite verbose.
extended	Will add additional outputs which has been removed from standard output due to limited usefulness.
...	Additional arguments for <a href="#">getMadratGraph</a> in case that no graph is provided (otherwise ignored)

### Author(s)

Jan Philipp Dietrich

### See Also

[getCalculations](#), [getMadratGraph](#)

Other dependency graph: [getDependencies\(\)](#), [getMadratGraph\(\)](#), [visualizeDependencies\(\)](#)

---

getMadratMessage            *getMadratMessage*

---

### Description

Read a madrat message from the madrat environment. The madrat environment behaves similar like global options, except that 1) messages will also be stored in cache files and restored when a cache file is being loaded and 2) messages are always stored in lists with messages split by function calls where the message was triggered.

### Usage

```
getMadratMessage(name = NULL, fname = NULL)
```

**Arguments**

name	The category in which the message should be stored
fname	function name. If specified only messages belonging to the functions history will be returned (this includes entries from the function itself, but also entries from functions which were called by this function).

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getMadratMessage](#)

Other madrat messages: [putMadratMessage\(\)](#), [resetMadratMessages\(\)](#), [vcats\(\)](#)

**Examples**

```
putMadratMessage("test", "This is a toast", fname = "readTau")
getMadratMessage("test", fname = "calcTauTotal")
```

---

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

**Arguments**

name	name of function for which sources should get returned. If not specified, all sources in the specified environment are returned
type	Type of source, either set to "read", "convert", "correct", "download" or NULL. If specified, a vector containing the sources with the corresponding function type are returned, otherwise a data.frame with all sources and their available function types is returned.

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 or data.frame containing all corresponding sources

**Note**

Please be aware that these functions only check the availability of corresponding functions of the package, not whether the functions will properly work.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[readSource](#), [setConfig](#)

Other available functions: [getCalculations\(\)](#), [getLocation\(\)](#), [installedMadratUniverse\(\)](#)

**Examples**

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

---

`installedMadratUniverse`

*installedMadratUniverse*

---

**Description**

Returns a name vector of installed packages which supposedly belong to the madrat universe. They are currently derived as the union of

- all packages registered under `getConfig("packages")`,
- all packages with a name starting with "mr" or "ms" (as the usual indicator for madrat-packages and madrat-support-packages), and
- all packages having madrat as either a Depends or Imports dependency.

**Usage**

```
installedMadratUniverse()
```

**Value**

A name vector of installed packages which supposedly belong to the madrat universe

**Author(s)**

Jan Philipp Dietrich

**See Also**

[setConfig](#)

Other available functions: [getCalculations\(\)](#), [getLocation\(\)](#), [getSources\(\)](#)

**Examples**

```
## Not run:  
installedMadratsUniverse()  
  
## End(Not run)
```

---

madratAttach

*madratAttach / madratDetach*

---

**Description**

Attaches the madrat functions of a package to the currently active madrat universe or detaches it again from it.

**Usage**

```
madratAttach(package)
```

```
madratDetach(package)
```

**Arguments**

package            name of the package to be loaded. Alternative, the path to the package.

**Functions**

- `madratDetach()`: detach package from madrat universe

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getConfig](#), [setConfig](#)

Other madrat configuration: [getConfig\(\)](#), [setConfig\(\)](#)

**Examples**

```
## Not run:  
madratAttach("madrat")  
  
## End(Not run)
```

---

metadataGFZ

*metadataGFZ*

---

**Description**

Function to extract metadata information of a data set hosted at GFZ dataservices (<https://dataservices.gfz-potsdam.de/portal/>).

**Usage**

```
metadataGFZ(doi)
```

**Arguments**

doi                   DOI of a data set hosted at GFZ dataservices

**Value**

a list with entries "license", "citation", "authors" and "year"

**Author(s)**

Jan Philipp Dietrich

**See Also**

[toolstartmessage](#), [vcat](#)

Other source downloading: [toolManualDownload\(\)](#)

**Examples**

```
## Not run:  
metadataGFZ("10.5880/pik.2019.004")  
  
## End(Not run)
```

---

pucAggregate	<i>pucAggregate</i>
--------------	---------------------

---

### Description

Function which takes a puc-file ("portable unaggregated collection") as created via [retrieveData](#) and computes the corresponding aggregated collection with the provided arguments (e.g. the provided region mapping). The resulting tgz-file containing the collection will be put to the madrat outputfolder as defined in `getConfig("outputfolder")`.

### Usage

```
pucAggregate(  
  puc,  
  regionmapping = getConfig("regionmapping"),  
  ...,  
  renv = TRUE,  
  strict = FALSE  
)
```

### Arguments

puc	path to a puc-file
regionmapping	region mapping to be used for aggregation.
...	(Optional) Settings that should be changed in addition. NOTE: which settings can be modified varies from puc to puc. Allowed settings are typically listed in the file name of the puc file after the revision number.
renv	Boolean which determines whether data should be aggregated from within a renv environment (recommended) or not. If activated, renv will check which packages in which versions were used to create the puc file, download, install and load these packages and run the aggregation with them. Otherwise, the packages in the currently used environment are being used.
strict	Boolean or NULL which allows to trigger a strict mode. During strict mode warnings will be taken more seriously and will cause 1. to have the number of warnings as prefix of the created tgz file and 2. will prevent <code>retrieveData</code> from creating a puc file. If set to NULL the setting will be read from the puc file.

### Author(s)

Jan Philipp Dietrich

### See Also

[retrieveData,localConfig](#)

Other aggregation: [toolAggregate\(\)](#)

**Examples**

```
## Not run:
pucAggregate("rev1_example.puc", regionmapping = "regionmappingH12.csv")

## End(Not run)
```

---

<code>putMadratMessage</code>	<i>putMadratMessage</i>
-------------------------------	-------------------------

---

**Description**

Store a madrat message in the madrat environment. The madrat environment behaves similar like global options, except that 1) messages will also be stored in cache files and restored when a cache file is being loaded and 2) messages are always stored in lists with messages split by function calls where the message was triggered.

**Usage**

```
putMadratMessage(name, value, fname = -1, add = FALSE)
```

**Arguments**

<code>name</code>	The category in which the message should be stored
<code>value</code>	The message that should be recorded as character. Alternatively, if name is not set, it is also possible to provide a complete list of the structure value[[name]][[fname]] where name and fname correspond to the category name and function name entries (e.g. value = list(test = list(readTau = "This is a toast"))).
<code>fname</code>	function name the entry belongs to or the frame number from which the function name should be derived from (e.g. -1 to receive function name from parent function).
<code>add</code>	boolean deciding whether the value should be added to an existing value (TRUE) or overwrite it (FALSE)

**Author(s)**

Jan Philipp Dietrich

**See Also**

[putMadratMessage](#)

Other madrat messages: [getMadratMessage\(\)](#), [resetMadratMessages\(\)](#), [vcats\(\)](#)

**Examples**

```
putMadratMessage("test", "This is a toast", fname = "readTau")
getMadratMessage("test", fname = "calcTauTotal")
```

---

readSource	<i>readSource</i>
------------	-------------------

---

### 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,
  subset = NULL,
  convert = TRUE,
  supplementary = FALSE
)
```

### Arguments

type	A character string referring to the source type, e.g. "IEA" which would internally call a function called 'readIEA' (the "wrapped function"). A list of available source types can be retrieved with function <a href="#">getSources</a> .
subtype	A character string. For some sources there are subtypes of the source, for these sources the subtype can be specified with this argument. If a source does not have subtypes, subtypes should not be set.
subset	A character string. Similar to subtype a source can also have subsets. A subsets can be used to only read part of the data. This can in particular make sense for huge data sets where reading in the whole data set might be impractical or even infeasible.
convert	Boolean indicating whether input data conversion to ISO countries should be done or not. In addition it can be set to "onlycorrect" for sources with a separate correctXXX-function.
supplementary	Boolean deciding whether a list including the actual data and metadata, or just the actual data is returned.

### Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

**Note**

If a magpie object is returned magclass::clean\_magpie is run and if convert = TRUE ISO code country level is checked.

The underlying read-functions can return a magpie object or a list of information (preferred) back to readSource. In list format the object should have the following structure:

- **x** - the data itself as magclass object
- **unit** (optional) - unit of the provided data
- **description** (optional) - a short description of the data
- **note** (optional) - additional notes related to the data
- **class** (optional | default = "magpie") - Class of the returned object. If set to something other than "magpie" most functionality will not be available and is switched off!
- **cache** (optional) boolean which decides whether a cache file should be written (if caching is active) or not. Default setting is TRUE. This can be for instance useful, if the calculation itself is quick, but the corresponding file sizes are huge. Or if the caching for the given data type does not support storage in RDS format. **CAUTION:** Deactivating caching for a data set which should be part of a PUC file will corrupt the PUC file. Use with care.

**Author(s)**

Jan Philipp Dietrich, Anastasis Giannousakis, Lavinia Baumstark, Pascal Sauer

**See Also**

[setConfig](#), [downloadSource](#), [readTau](#)

Other core workflow: [calcOutput\(\)](#), [downloadSource\(\)](#), [retrieveData\(\)](#), [returnCalcOutput\(\)](#)

**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.

**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")  
  
## End(Not run)
```

---

redirectSource	<i>redirectSource</i>
----------------	-----------------------

---

**Description**

redirectSource will call a source specific redirect function if it exists (called e.g. redirectTau), in which case the arguments are passed on to that function. If such a function is not available [redirect](#) is called.

**Usage**

```
redirectSource(type, target, ..., linkOthers = TRUE, local = TRUE)
```

**Arguments**

type	Dataset name, e.g. "Tau" to set the source folder that <code>readTau</code> will use
target	Either path to the new source folder that should be used instead of the default, or NULL to remove the redirection, or a vector of paths to files which are then symlinked into a temporary folder that is then used as target folder; if the vector is named the names are used as relative paths in the temporary folder, e.g. target = c('a/b/c.txt' = "~/d/e/f.txt") would create a temporary folder with subfolders a/b and there symlink c.txt to ~/d/e/f.txt.
...	Additional arguments, passed on to the specific redirect function.
linkOthers	If target is a list of files, whether to symlink all other files in the original source folder to the temporary folder.
local	If TRUE the redirection is only temporary and will be reset when the function which calls redirect is finished. Set to FALSE for a permanent/global redirection or to an environment for more control.

**Value**

The result of the specific redirect function or `redirect`.

**Author(s)**

Pascal Sauer

**See Also**

Other source redirection: `getLinkFunction()`

**Examples**

```
## Not run:
f <- function() {
  redirectSource("Tau", target = "~/TauExperiment")
  # the following call will change directory
  # into ~/TauExperiment instead of <getConfig("sourcefolder")>/Tau
  readSource("Tau")
}
f()
# Tau is only redirected in the local environment of f,
# so it will use the usual source folder here
readSource("Tau")

## End(Not run)
```

---

redirectTau	<i>redirectTau</i>
-------------	--------------------

---

**Description**

redirectTau will be called by redirectSource when type = "Tau". Redirects the Tau source folder to the target folder.

**Usage**

```
redirectTau(target, ...)
```

**Arguments**

target	The target folder or files.
...	Passed on to <a href="#">redirect</a> .

**Author(s)**

Pascal Sauer

**Examples**

```
## Not run:  
redirectSource("Tau", "a/different/tau-source-folder")  
a <- readSource("Tau", "paper")  
  
## End(Not run)
```

---

regionscode	<i>Tool: regionscode</i>
-------------	--------------------------

---

**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 = NULL, label = FALSE, strict = TRUE)
```

**Arguments**

mapping	Either a path to a mapping or an already read-in mapping as data.frame. If set to NULL (default) the regionscode of the region mapping set in the madrat config will be returned.
label	logical deciding whether the corresponding label of a regionscode should be returned instead of the regionscode.
strict	If set to TRUE region mappings with mapping to ISO countries with exactly 2 columns or more than 2 columns (if the first column contains irrelevant information which will be deleted automatically) will be accepted. In this case data will be transformed and even cases with different ordering will yield the same regionscode. If set to FALSE all these checks will be ignored and the regionscode will be just computed on the object as it is. Please be aware the regionscode will differ with strict mode on or off!

**Value**

A md5-based regionscode which describes the given mapping or, if label=TRUE and a corresponding label is available, the label belonging to the regionscode

**Author(s)**

Jan Philipp Dietrich

**See Also**

[toolCodeLabels](#), [fingerprint](#), [digest](#)

Other code labeling: [toolCodeLabels\(\)](#)

**Examples**

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

---

resetMadratMessages    *resetMadratMessages*

---

**Description**

Delete stored madrat messages from the madrat environment. The madrat environment behaves similar like global options, except that 1) messages will also be stored in cache files and restored when a cache file is being loaded and 2) messages are always stored in lists with messages split by function calls where the message was triggered.

**Usage**

```
resetMadratMessages(name = NULL, fname = NULL)
```

**Arguments**

name	The category for which the messages should be reset (if not set messages in all categories will be reset)
fname	function name for which the entries should be reset (if not specified messages for all function names will be reset)

**Author(s)**

Jan Philipp Dietrich

**See Also**

[putMadratMessage](#), [getMadratMessage](#)

Other madrat messages: [getMadratMessage\(\)](#), [putMadratMessage\(\)](#), [vcat\(\)](#)

**Examples**

```
putMadratMessage("test", "This is a toast", fname = "readTau")
getMadratMessage("test", fname = "calcTauTotal")
resetMadratMessages("test")
```

---

retrieveData

*retrieveData*

---

**Description**

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

**Usage**

```
retrieveData(
  model,
  rev = 0,
  dev = "",
  cachetype = "def",
  puc = identical(dev, ""),
  strict = FALSE,
  revn = TRUE,
  ...
)
```

**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. Will be converted to <a href="#">numeric_version</a> .
dev	development suffix to distinguish development versions for the same data revision. This can be useful to distinguish parallel lines of development.
cachetype	defines what cache should be used. "rev" points to a cache shared by all calculations for the given revision and sets forcecache to TRUE, "def" points to the cache as defined in the current settings and does not change forcecache setting.
puc	Boolean deciding whether a fitting puc file (if existing) should be read in and if a puc file (if not already existing) should be created.
strict	Boolean which allows to trigger a strict mode. During strict mode warnings will be taken more seriously and will cause 1. to have the number of warnings as prefix of the created tgz file and 2. will prevent retrieveData from creating a puc file.
renv	Boolean which determines whether calculations should run within a renv environment (recommended) or not (currently only applied in <a href="#">pucAggregate</a> ). If activated, renv will check which packages in which versions were used to create the puc file, download, install and load these packages and run the aggregation with them. Otherwise, the packages in the currently used environment are being used.
...	(Optional) Settings that should be changed using <code>setConfig</code> (e.g. regionmapping). or arguments which should be forwarded to the corresponding fullXYZ function (Please make sure that argument names in full functions do not match settings in <code>setConfig</code> !)

**Value**

Invisibly, the path to the newly created tgz archive.

**Note**

The underlying full-functions can optionally provide a list of information back to `retrieveData`. Following list entries are currently supported:

- **tag** (optional) - additional name tag which will be included in the file name of the aggregated collection (resulting tgz-file). This can be useful to highlight information in the file name which otherwise would not be visible.
- **pucTag** (optional) - identical purpose as **tag** but for the resulting unaggregated collections (puc-files).

**Author(s)**

Jan Philipp Dietrich, Lavinia Baumstark

**See Also**

[calcOutput](#), [setConfig](#)

Other core workflow: [calcOutput\(\)](#), [downloadSource\(\)](#), [readSource\(\)](#), [returnCalcOutput\(\)](#)

**Examples**

```
## Not run:
retrieveData("example", rev = "2.1.1", dev = "test", regionmapping = "regionmappingH12.csv")

## End(Not run)
```

---

returnCalcOutput	<i>Construct a list with the information required for calc-functions to return</i>
------------------	--

---

**Description**

A simple helper to have auto-completion for required and optional elements of calcOutput() return lists.

**Usage**

```
returnCalcOutput(
  x,
  weight,
  unit,
  description,
  note,
  class,
  isocountries,
  mixed_aggregatio,
  min,
  max,
  structure.spatial,
  structure.temporal,
  structure.data,
  aggregationFunction,
  aggregationArguments,
  putinPUC,
  cache
)
```

**Arguments**

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

<code>class</code>	(optional   default = "magpie") Class of the returned object. If set to something other than "magpie" most functionality, such as aggregation or unit tests will not be available and is switched off!
<code>isocountries</code>	(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)
<code>mixed_aggregatio</code>	(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.
<code>min</code>	(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
<code>max</code>	(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
<code>structure.spatial</code>	(optional) regular expression describing the name structure of all names in the spatial dimension (e.g. " <code>^[A-Z]{3}\$</code> "). Names will be checked against this regular expression and disagreements will be reported via a warning.
<code>structure.temporal</code>	(optional) regular expression describing the name structure of all names in the temporal dimension (e.g. " <code>^y[0-9]{4}\$</code> "). Names will be checked against this regular expression and disagreements will be reported via a warning.
<code>structure.data</code>	(optional) regular expression describing the name structure of all names in the data dimension (e.g. " <code>^[a-z]*\.[a-z]*\$</code> "). Names will be checked against this regular expression and disagreements will be reported via a warning.
<code>aggregationFunction</code>	(optional   default = <code>toolAggregate</code> ) Function to be used to aggregate data from country to regions. The function must have the argument <code>x</code> for the data itself and <code>rel</code> for the relation mapping between countries and regions and must return the data as magpie object in the spatial resolution as defined in <code>rel</code> .
<code>aggregationArguments</code>	(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 <code>x</code> , <code>rel</code> and if provided/deviating from the default also <code>weight</code> and <code>mixed_aggregation</code> .
<code>putinPUC</code>	(optional) boolean which decides whether this calculation should be added to a puc file which contains non-aggregated data and can be used to later on aggregate the data to resolutions of own choice. If not set calcOutput will try to determine automatically, whether a file is being required for the puc file or not, but in more complex cases (e.g. if calculations below top-level have to be run as well) this setting can be used to manually tweak the puc file list. CAUTION: Incorrect settings will cause corrupt puc files, so use this setting with extreme care and only if necessary.

cache (optional) boolean which decides whether a cache file should be written (if caching is active) or not. Default setting is TRUE. This can be for instance useful, if the calculation itself is quick, but the corresponding file sizes are huge. Or if the caching for the given data type does not support storage in RDS format. CAUTION: Deactivating caching for a data set which should be part of a PUC file will corrupt the PUC file. Use with care.

### Value

A list with the specified arguments.

### See Also

calcOutput

Other core workflow: [calcOutput\(\)](#), [downloadSource\(\)](#), [readSource\(\)](#), [retrieveData\(\)](#)

### Examples

```
calcSomething <- function() {
  x <- "some calculation"

  returnCalcOutput(x = x, weight = NULL, unit = "a unit",
                  description = "the description")
}
```

---

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. For values which can be reset to NULL (currently only "extramappings") you can achieve a reset by setting the value to "".

### Usage

```
setConfig(
  ...,
  regionmapping = NULL,
  extramappings = NULL,
  packages = NULL,
  globalenv = NULL,
  enablecache = NULL,
  verbosity = NULL,
  mainfolder = NULL,
  sourcefolder = NULL,
  cachefolder = NULL,
```

```

mappingfolder = NULL,
outputfolder = NULL,
pucfolder = NULL,
tmpfolder = NULL,
nolabels = NULL,
forcecache = NULL,
ignorecache = NULL,
cachecompression = NULL,
hash = NULL,
diagnostics = NULL,
debug = NULL,
maxLengthLogMessage = NULL,
redirections = NULL,
.cfgchecks = TRUE,
.verbose = TRUE,
.local = FALSE
)

localConfig(...)

```

### Arguments

...	Arguments forwarded to setConfig
regionmapping	The name of the csv file containing the region mapping that should be used for aggregation (e.g. "regionmappingREMIND.csv").
extramappings	Names of additional mappings supplementing the given region mapping. This allows for additional aggregation levels such as subnational aggregation.
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	Is deprecated and will be ignored. Please use ignorecache instead.
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
pucfolder	The path where portable unaggregated collection (puc) files are located. NA by default, which means <mainfolder>/puc
tmpfolder	Path to a temp folder for temporary storage of files. By default set to <mainfolder>/tmp
nolabels	vector of retrieve models (e.g. "EXAMPLE" in case of "fullEXAMPLE") which should NOT apply a replacement of known hashes with given code labels
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.
cachecompression	logical or character string specifying whether cache files use compression. TRUE corresponds to gzip compression, and character strings "gzip", "bzip2" or "xz" specify the type of compression.
hash	specifies the used hashing algorithm. Default is "xxhash32" and all algorithms supported by <a href="#">digest</a> can be used.
diagnostics	Either FALSE (default) to avoid the creation of additional log files or TRUE to write a log of diagnostic outputs to "diagnostics.log". Setting diagnostics to a file name is deprecated.
debug	Boolean which activates a debug mode. In debug mode all calculations will be executed with try=TRUE so that calculations do not stop even if the previous calculation failed. This can be helpful to get a full picture of errors rather than only seeing the first one. In addition debug=TRUE will add the suffix "debug" to the files created to avoid their use in productive runs. Furthermore, with debug=TRUE calculations will be rerun even if a corresponding tgz file already exists.
maxLengthLogMessage	in log messages evaluated arguments are printed if the resulting message is shorter than this value, otherwise arguments are shown as passed, potentially with unevaluated variable names
redirections	A list of source folder redirections, intended to be set by <a href="#">redirectSource</a> . See that function's documentation for more details.
.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
.local	boolean deciding whether options are only changed until the end of the current function execution OR environment for which the options should get changed.

## Functions

- `localConfig()`: A wrapper for `setConfig(..., .local = TRUE)`

## Note

`setConfig` must only be used before the data processing is started and changes in the configuration from within a `download-`, `read-`, `correct-`, `convert-`, `calc-`, or `full-`function are not allowed! Only allowed configuration update is to add another extramapping via [addMapping](#). Currently the use of `setConfig` within any of these functions will trigger a warning, which is planned to be converted into an error message in one of the next package updates!

## Author(s)

Jan Philipp Dietrich

## See Also

[getConfig](#), [getIS0list](#)

Other madrat configuration: [getConfig\(\)](#), [madratAttach\(\)](#)

## Examples

```
## Not run:  
setConfig(forcecache = c("readSSPa11", "convertSSPa11"))  
  
## 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,  
  wdim = NULL,  
  partrel = FALSE,  
  negative_weight = "warn",
```

```

    mixed_aggregation = FALSE,
    verbosity = 1,
    zeroWeight = "warn"
)

```

### Arguments

x	magclass object that should be (dis-)aggregated
rel	relation matrix, mapping or file containing a mapping in a format supported by <a href="#">toolGetMapping</a> (currently csv, rds or rda). A mapping object consists of any number of columns, where one column contains all the elements in x. These elements are mapped to the corresponding values in another column, as described below (see parameters 'from' and 'to'). It is possible to not set rel as long as to is set and dim is chosen appropriately. In that case the relation mapping is extracted from the dimnames of the corresponding dimension, e.g. if your data contains a spatial subdimension "country" you can aggregate to countries via <code>toolAggregate(x, to = "country", dim = 1)</code> .
weight	magclass object containing weights to be used for a weighted (dis)aggregation. The provided weight does not need to be normalized, any number $\geq 0$ is allowed. See "details" section below.
from	Name of source column to be used in rel if it is a mapping (if not set the first column matching the data will be used).
to	Name of the target column to be used in rel if it is a mapping (if not set the column following column from will be used. If column from is the last column, the column before from is used). If data should be aggregated based on more than one column these columns can be specified via "+", e.g. "region+global" if the data should be aggregated to column regional as well as column global. In the to column, empty values (in the form of "") will not be included in the aggregated data (but used as normal aggregation targets). If rel is missing to refers to the aggregation target dimension name.
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).
wdim	Specifying the according weight dimension as chosen with dim for the aggregation object. If set to NULL the function will try to automatically detect the 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
zeroWeight	Describes how a weight sum of 0 for a category/aggregation target should be treated. "allow" accepts it and returns 0 (dangerous), "setNA" returns NA, "warn" throws a warning, "stop" throws an error, "fix" will set zero weights to $10^{-30}$ only where needed, see <a href="#">toolFixWeight</a> .

## 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 (then all columns are just summed up) or you 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, Pascal Sauer

## See Also

[calcOutput](#)

Other aggregation: [pucAggregate\(\)](#)

## Examples

```
# create example mapping
p <- magclass::maxample("pop")
mapping <- data.frame(from = magclass::getItems(p, dim = 1.1),
                     region = rep(c("REG1", "REG2"), 5),
                     global = "GLO")

print(mapping)

# run aggregation
toolAggregate(p, mapping)
# weighted aggregation
toolAggregate(p, mapping, weight = p)
# combined aggregation across two columns
toolAggregate(p, mapping, to = "region+global")
```

---

toolCodeLabels	<i>Tool: CodeLabels</i>
----------------	-------------------------

---

### Description

This function replaces a hash code (e.g. regioncode) or another cryptic code with a human readable code via a given dictionary. This can be useful to make outputs better readable in cases where hash codes are already known to the user. If no entry exists in the dictionary the hash code is returned again.

### Usage

```
toolCodeLabels(get = NULL, add = NULL)
```

### Arguments

get	A vector of hash codes which should be replaced
add	Additional entries that should be added to the dictionary. Need to be provided in the form of a named vector with the structure <code>c(&lt;label&gt;=&lt;hash&gt;)</code> , e.g. <code>c(h12="62eff8f7")</code>

### Value

A vector with either labels (if available) or hash codes (if no label was available).

### Author(s)

Jan Philipp Dietrich

### See Also

[regionscode](#)

Other code labeling: [regionscode\(\)](#)

### Examples

```
toolCodeLabels("62eff8f7")
```

---

toolCompareStatusLogs *toolCompareStatusLogs*

---

### Description

Compares status logs and returns a textual rendering of the diff between the two.

### Usage

```
toolCompareStatusLogs(  
    oldArchivePath = NULL,  
    newArchivePath = NULL,  
    oldLogPath = file.path(".", "status.log"),  
    newLogPath = file.path(".", "status.log"),  
    sections = c("changedStatistics", "changedCalls", "addedCalls", "removedCalls")  
)
```

### Arguments

oldArchivePath	Paths to the tgz-archive that serves as the old / new archive. If given, oldLogPath / newLogPath determines the name of the log file within the archive. If not given, oldLogPath / newLogPath need to be given.
newArchivePath	see above
oldLogPath	Paths to a log file, either within an archive, if an archive is given, or to a single file.
newLogPath	see above
sections	Vector of sections the output should include. Valid section names are: changed-Statistics, changedCalls, addedCalls, removedCalls

### Value

A printable string describing the changes that occurred between the old and the new log.

### Author(s)

Patrick Rein

### See Also

Other validation: [compareData\(\)](#), [compareMadratOutputs\(\)](#)

---

toolConditionalReplace  
*toolConditionalReplace*

---

**Description**

Sets values (NA, negative, ..) to value replaceby

**Usage**

```
toolConditionalReplace(x, conditions, replaceby = 0)
```

**Arguments**

x	maggie object
conditions	vector of conditions for values, that should be removed e.g. "is.na()", "< 0" (order matters)
replaceby	value which should be used instead (can be a vector of same length as conditions as well)

**Value**

return changed input data

**Author(s)**

Kristine Karstens

**See Also**

Other data manipulation: [toolFixWeight\(\)](#), [toolNAreplace\(\)](#), [toolOrderCells\(\)](#), [toolXlargest\(\)](#)

---

toolConvertMapping     *Tool: ConvertMapping*

---

**Description**

Function which converts mapping files between formats

**Usage**

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

**Arguments**

name	File name of the mapping file. Supported file types are currently csv (, or ; separated), rds and rda (which needs to have the data stored with the object name "data!").
format	format it should be converted to. Available is "csv", "rds" 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

**Author(s)**

Jan Philipp Dietrich

**See Also**

[calcOutput](#), [toolConvertMapping](#)

Other mapping: [addMapping\(\)](#), [toolGetMapping\(\)](#)

---

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,
  ignoreCountries = NULL,
  type = NULL,
  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
ignoreCountries	A vector of country names/codes that exist in the data and that should be removed 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.

type deprecated and will be removed soon!  
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](#)

Other country tools: [getISOlist\(\)](#), [toolCountryFill\(\)](#), [toolFillWithRegionAvg\(\)](#), [toolISOhistorical\(\)](#)

**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,  
  countrylist = NULL,  
  ...  
)
```

**Arguments**

<code>x</code>	MAGPIE object with ISO country codes in the spatial dimension
<code>fill</code>	Number which should be used for filling the gaps of missing countries.
<code>no_remove_warning</code>	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.
<code>overwrite</code>	logical deciding whether existing data should be overwritten, if there is a specific mapping provided for that country, or not
<code>verbosity</code>	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).
<code>countrylist</code>	character vector of official country names (if other than ISO)
<code>...</code>	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 Hong Kong should receive the value of China). This replacement usually only makes sense for intensive values. Can be also provided as a argument called "map" which contains a named vector of these mappings.

**Value**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

Other country tools: [getISOList\(\)](#), [toolCountry2isocode\(\)](#), [toolFillWithRegionAvg\(\)](#), [toolISOhistorical\(\)](#)

**Examples**

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

---

 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,
  noteThreshold = 1
)
```

### Arguments

<code>x</code>	MAGPIE object with country codes in the first and time steps in the second dimension.
<code>valueToReplace</code>	value that denotes missing data. Defaults to NA.
<code>weight</code>	MAGPIE object with weights for the weighted average. Must contain at least all the countries and years present in <code>x</code> . If no weights are specified, an unweighted average is performed.
<code>callToolCountryFill</code>	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.
<code>regionmapping</code>	Data frame containing the mapping between countries and regions. Expects column names <code>CountryCode</code> and <code>RegionCode</code> . Uses the currently set mapping if no mapping is specified.
<code>verbose</code>	Boolean variable indicating if the function should print out what it is doing. Can generate a lot of output for a large object.
<code>warningThreshold</code>	If more than this fraction of the countries in a given region and timestep have a missing value, throw a warning.
<code>noteThreshold</code>	If more than this fraction of the countries in a given region and timestep have a missing value, a note will be written.

**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, Jan Philipp Dietrich

**See Also**

Other country tools: [getISOlist\(\)](#), [toolCountry2isocode\(\)](#), [toolCountryFill\(\)](#), [toolISOhistorical\(\)](#)

**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)
```

---

toolFillYears

*toolFillYears*

---

**Description**

Inter- and extrapolates a historical dataset for a given time period.

**Usage**

```
toolFillYears(x, years)
```

**Arguments**

x	MAgPIE object to be continued.
years	vector of years as digits or in mag year format

**Value**

MAgPIE object with completed time dimensionality.

**Author(s)**

Kristine Karstens

**See Also**

Other temporal processing: [toolTimeAverage\(\)](#), [toolTimeSpline\(\)](#)

---

toolFixWeight	<i>toolFixWeight</i>
---------------	----------------------

---

**Description**

Check the example below for the problem this function is solving. Instead of adding a small number (like  $10^{-10}$ ) to each entry in weight, set only those weights to  $10^{-30}$  where otherwise the total sum of the (dis)aggregated object would be different from the original.

**Usage**

```
toolFixWeight(weight, map, dim)
```

**Arguments**

weight	magclass object containing weights to be used for a weighted (dis)aggregation. The provided weight does not need to be normalized, any number $\geq 0$ is allowed.
map	a data frame where the first column contains coarse resolution items and the second column contains fine resolution items; fine resolution items must match items in weight
dim	which dim to fix (e.g. 1 or 3.2 or "region")

**Value**

weight, with weights set to  $10^{-30}$  only where otherwise the total sum of the (dis)aggregated object would be different from the original

**Author(s)**

Pascal Sauer

**See Also**

[toolAggregate](#)

Other data manipulation: [toolConditionalReplace\(\)](#), [toolNAreplace\(\)](#), [toolOrderCells\(\)](#), [toolXlargest\(\)](#)

**Examples**

```
x <- new.maggpie(c("A", "B"), fill = 100)
rel <- data.frame(c("A", "A", "B", "B"),
                 c("A1", "A2", "B1", "B2"))
weight <- new.maggpie(c("A1", "A2", "B1", "B2"), fill = 0)
weight["B1", , ] <- 1

y <- toolAggregate(x, rel, weight) # warning "Weight sum is 0"
stopifnot(sum(x) - sum(y) == 100) # total sum no longer equal, hence the warning
stopifnot(as.vector(y["B2", , ]) == 0)

# problematic hotfix
y <- toolAggregate(x, rel, weight + 10^-10) # no warning
stopifnot(sum(x) == sum(y)) # total sum is equal
stopifnot(as.vector(y["B2", , ]) > 0) # this is the problem, should still be 0

# proper fix
y <- toolAggregate(x, rel, weight, zeroWeight = "fix") # no warning
stopifnot(sum(x) == sum(y)) # total sum is equal
stopifnot(as.vector(y["B2", , ]) == 0) # and this is also still 0
```

---

 toolGetMapping

*Tool: GetMapping*


---

**Description**

Function which retrieves a mapping file

**Usage**

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

**Arguments**

name	File name of the mapping file. Supported file types are currently csv (, or ; separated), rds and rda (which needs to have the data stored with the object name "data!"). Use <a href="#">toolConvertMapping</a> 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", "local" (if the path is relative to your current directory) or the name of a package which contains the mapping. If set to NULL it will first try "local", then "mappingfolder" and afterwards scan all packages currently listed in getConfig("packages")
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
activecalc	If set, this argument helps to define the first package within which the mapping has to be sought for. This happens via finding in which package the active calc function is located.

**Value**

the mapping as a data frame

**Author(s)**

Jan Philipp Dietrich

**See Also**

[calcOutput](#), [toolConvertMapping](#)

Other mapping: [addMapping\(\)](#), [toolConvertMapping\(\)](#)

**Examples**

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

---

toolISOhistorical	<i>Tool: ISOhistorical</i>
-------------------	----------------------------

---

**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,
  additional_weight = NULL
)
```

**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. If provided as file the mapping needs to contain three columns "fromISO", "toISO" and "lastYear".
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
additional_weight	optional weight to be used for regional disaggregation, if not provided, the values of m in the "lastYear" are used as weight

**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

**See Also**

Other country tools: [getISOlist\(\)](#), [toolCountry2isocode\(\)](#), [toolCountryFill\(\)](#), [toolFillWithRegionAvg\(\)](#)

---

toolManualDownload      *Tool: ManualDownload*

---

**Description**

Support tool for the creation of download functions in cases where a fully automated data download is not an option (e.g. due to a missing API). The function can be used to print a step-by-step guide for the user how to manually retrieve the data and then asks for a (local) path where the data can be copied from.

**Usage**

```
toolManualDownload(
  instructions,
  intro = "Data must be downloaded manually",
  request = "Enter full path to the downloaded data:"
)
```

**Arguments**

instructions	Download instructions in form of a character vector describing how to manually retrieve the data.
intro	Introductory sentence to be shown first. Will not show up if set to NULL.
request	A prompt which should show up after the instructions to ask for the local download location.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[downloadSource](#)

Other source downloading: [metadataGFZ\(\)](#)

**Examples**

```
## Not run:
toolManualDownload(c("Log into website ABC",
                    "Download the data set XYZ"))

## End(Not run)
```

---

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](#)

Other data manipulation: [toolConditionalReplace\(\)](#), [toolFixWeight\(\)](#), [toolOrderCells\(\)](#), [toolXlargest\(\)](#)

---

toolOrderCells	<i>toolOrderCells</i>
----------------	-----------------------

---

**Description**

reorder numbered spatial units (cells, clusters) by number. Function will return the unmodified object, if the given subdimension does not exist or does not contain cell information.

**Usage**

```
toolOrderCells(x, dim = 1.2, na.rm = FALSE)
```

**Arguments**

x	magclass object that should be ordered
dim	subdimension which contains the cell information
na.rm	boolean deciding how to deal with non-integer information in cellular column. If FALSE, non-integer values will lead to a return of the unsorted object, if TRUE non-integer cells will be removed from the data set and the rest will get sorted

**Value**

ordered data in magclass format

**Author(s)**

Kristine Karstens, Jan Philipp Dietrich

**See Also**

Other data manipulation: [toolConditionalReplace\(\)](#), [toolFixWeight\(\)](#), [toolNAreplace\(\)](#), [toolXlargest\(\)](#)



---

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 or list. 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](#)

Other subtype handling: [toolSplitSubtype\(\)](#)

### Examples

```
files <- c(production="production.csv",
           production="production.csv",
           extent="forest_extent.csv")
toolSubtypeSelect("extent",files)
```

---

toolTimeAverage	<i>toolTimeAverage</i>
-----------------	------------------------

---

**Description**

average over time given an averaging range. Only works for data with equidistant time steps!

**Usage**

```
toolTimeAverage(x, averaging_range = NULL, cut = TRUE)
```

**Arguments**

x	magclass object that should be averaged with equidistant time steps
averaging_range	number of time steps to average
cut	if TRUE, all time steps at the start and end that can not be averaged correctly, will be removed if FALSE, time steps at the start and end will be averaged with high weights for start and end points

**Value**

the averaged data in magclass format

**Author(s)**

Kristine Karstens, Jan Philipp Dietrich

**See Also**

Other temporal processing: [toolFillYears\(\)](#), [toolTimeSpline\(\)](#)

---

toolTimeSpline	<i>Smooth a magclass time series with optional anchor years</i>
----------------	---

---

**Description**

Smooths a magclass time series using spline approximation with the given degrees of freedom. Optionally, specific years can be "pegged" (anchored) to stay close to their original values during smoothing. Anchoring is enforced by applying high weights to those years.

**Usage**

```
toolTimeSpline(x, dof = 5, peggedYears = NULL, anchorFactor = 10)
```

**Arguments**

x	A magclass object.
dof	Degrees-of-freedom per 100 years (higher -> more degrees of freedom, less smoothing; default 5).
peggedYears	Integer vector of years (e.g. 'c(2020, 2050, 2100)') to anchor during smoothing; NULL for none (default).
anchorFactor	Numeric multiplier for anchor weights (default 10); larger values more strongly enforce pegging.

**Value**

A magclass object of the same shape, with each time series spline-smoothed.

**Author(s)**

Kristine Karstens, Felicitas Beier, Michael Crawford

**See Also**

Other temporal processing: [toolFillYears\(\)](#), [toolTimeAverage\(\)](#)

---

toolXlargest	<i>toolXlargest</i>
--------------	---------------------

---

**Description**

Selects the countries with the highest values in a magpie object

**Usage**

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

**Arguments**

x	magclass object 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.

**Value**

vector with ISO country codes

**Author(s)**

Benjamin Leon Bodirsky, Jan Philipp Dietrich

**See Also**

Other data manipulation: [toolConditionalReplace\(\)](#), [toolFixWeight\(\)](#), [toolNAreplace\(\)](#), [toolOrderCells\(\)](#)

**Examples**

```
toolXlargest(magclass::maxample("pop"), range = 1:3)
```

---

vcat

*Tool: Verbosity Cat*


---

**Description**

Function which returns information based on the verbosity setting

**Usage**

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

**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 information, 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.
logOnly	option to only log warnings and error message without creating warnings or errors (expert use only).

**Author(s)**

Jan Philipp Dietrich

**See Also**

[readSource](#)

Other madrat messages: [getMadratMessage\(\)](#), [putMadratMessage\(\)](#), [resetMadratMessages\(\)](#)

**Examples**

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

## End(Not run)
```

---

visualizeDependencies *visualizeDependencies*

---

**Description**

Creates a graphical visualization of dependencies between functions in the mr-universe.

**Usage**

```
visualizeDependencies(
  ...,
  direction = "both",
  order = 2,
  filter = NULL,
  packages = getConfig("packages"),
  filename = NULL
)
```

**Arguments**

...	function(s) to be analyzed
direction	Character string, either "in", "out" or "both". If "in" all sources feeding into the function are listed. If "out" consumer of the function are listed. If "both" the union of "in" and "out" is returned.

order	order of dependencies. Order 1 would be only functions directly called from (in case of direction "in") or directly calling (in case of direction "out") are shown. Order 2 will also show direct dependencies of the order 1 dependencies, order 3 also the direct dependencies from order 2 dependencies, etc.
filter	regular expression to describe elements which should be excluded from visualization (e.g. "^tool" to exclude all tool functions)
packages	packages to use when searching dependencies
filename	If a filename is provided, the resulting graph will be saved

**Author(s)**

Debbora Leip, Jan Philipp Dietrich

**See Also**

[getDependencies](#), [getMadratGraph](#), [getMadratInfo](#)

Other dependency graph: [getDependencies\(\)](#), [getMadratGraph\(\)](#), [getMadratInfo\(\)](#)

# Index

- \* **aggregation**
    - [pucAggregate](#), 29
    - [toolAggregate](#), 44
  - \* **analysis**
    - [findBottlenecks](#), 15
  - \* **available functions**
    - [getCalculations](#), 17
    - [getLocation](#), 22
    - [getSources](#), 25
    - [installedMadratUniverse](#), 26
  - \* **cache management**
    - [cacheCleanup](#), 6
    - [cacheCopy](#), 7
  - \* **code labeling**
    - [regionscode](#), 35
    - [toolCodeLabels](#), 47
  - \* **core workflow**
    - [calcOutput](#), 7
    - [downloadSource](#), 14
    - [readSource](#), 31
    - [retrieveData](#), 37
    - [returnCalcOutput](#), 39
  - \* **country tools**
    - [getISOlist](#), 21
    - [toolCountry2isocode](#), 50
    - [toolCountryFill](#), 51
    - [toolFillWithRegionAvg](#), 53
    - [toolISOhistorical](#), 57
  - \* **data manipulation**
    - [toolConditionalReplace](#), 49
    - [toolFixWeight](#), 55
    - [toolINareplace](#), 59
    - [toolOrderCells](#), 60
    - [toolXlargest](#), 64
  - \* **dependency graph**
    - [getDependencies](#), 19
    - [getMadratGraph](#), 23
    - [getMadratInfo](#), 24
    - [visualizeDependencies](#), 66
  - \* **madrat configuration**
    - [getConfig](#), 18
    - [madratAttach](#), 27
    - [setConfig](#), 41
  - \* **madrat messages**
    - [getMadratMessage](#), 24
    - [putMadratMessage](#), 30
    - [resetMadratMessages](#), 36
    - [vcats](#), 65
  - \* **mapping**
    - [addMapping](#), 5
    - [toolConvertMapping](#), 49
    - [toolGetMapping](#), 56
  - \* **metadata comment**
    - [getFromComment](#), 20
  - \* **source downloading**
    - [metadataGFZ](#), 28
    - [toolManualDownload](#), 58
  - \* **source redirection**
    - [getLinkFunction](#), 22
    - [redirectSource](#), 33
  - \* **subtype handling**
    - [toolSplitSubtype](#), 61
    - [toolSubtypeSelect](#), 62
  - \* **temporal processing**
    - [toolFillYears](#), 54
    - [toolTimeAverage](#), 63
    - [toolTimeSpline](#), 63
  - \* **validation**
    - [compareData](#), 11
    - [compareMadratOutputs](#), 12
    - [toolCompareStatusLogs](#), 48
- [addMapping](#), 5, 44
- [addMapping\(\)](#), 50, 57
- [cacheCleanup](#), 6
- [cacheCleanup\(\)](#), 7
- [cacheCopy](#), 7
- [cacheCopy\(\)](#), 6

- calcOutput, [7](#), [11](#), [16](#), [38](#), [46](#), [50](#), [57](#), [60](#)
- calcOutput(), [15](#), [32](#), [38](#), [41](#)
- calcTauTotal, [10](#), [11](#), [12](#)
- compareData, [11](#)
- compareData(), [13](#), [48](#)
- compareMadratOutputs, [12](#)
- compareMadratOutputs(), [12](#), [48](#)
- convertTau, [11](#), [13](#)
  
- digest, [36](#), [43](#)
- downloadSource, [14](#), [32](#), [59](#)
- downloadSource(), [10](#), [32](#), [38](#), [41](#)
  
- findBottlenecks, [15](#)
- fingerprint, [36](#)
- fullEXAMPLE, [16](#)
  
- getCalculations, [8](#), [16](#), [17](#), [20](#), [21](#), [23](#), [24](#)
- getCalculations(), [23](#), [26](#), [27](#)
- getConfig, [18](#), [23](#), [27](#), [44](#)
- getConfig(), [27](#), [44](#)
- getDependencies, [19](#), [23](#), [67](#)
- getDependencies(), [23](#), [24](#), [67](#)
- getFromComment, [20](#)
- getISOList, [21](#), [44](#)
- getISOList(), [51](#), [52](#), [54](#), [58](#)
- getLinkFunction, [22](#)
- getLinkFunction(), [34](#)
- getLocation, [22](#)
- getLocation(), [17](#), [26](#), [27](#)
- getMadratGraph, [19](#), [20](#), [23](#), [24](#), [67](#)
- getMadratGraph(), [20](#), [24](#), [67](#)
- getMadratInfo, [20](#), [24](#), [67](#)
- getMadratInfo(), [20](#), [23](#), [67](#)
- getMadratMessage, [24](#), [25](#), [37](#)
- getMadratMessage(), [30](#), [37](#), [66](#)
- getSources, [14](#), [21](#), [25](#), [31](#), [51](#)
- getSources(), [17](#), [23](#), [27](#)
  
- initializeConfig, [19](#)
- installedMadratUniverse, [26](#)
- installedMadratUniverse(), [17](#), [23](#), [26](#)
  
- localConfig, [29](#)
- localConfig (setConfig), [41](#)
  
- madrat (madrat-package), [4](#)
- madrat-package, [4](#)
- madratAttach, [27](#)
- madratAttach(), [19](#), [44](#)
  
- madratDetach (madratAttach), [27](#)
- metadataGFZ, [28](#)
- metadataGFZ(), [59](#)
  
- numeric\_version, [16](#), [38](#)
  
- pucAggregate, [29](#), [38](#)
- pucAggregate(), [46](#)
- putMadratMessage, [30](#), [30](#), [37](#)
- putMadratMessage(), [25](#), [37](#), [66](#)
  
- readSource, [15–17](#), [26](#), [31](#), [33](#), [51](#), [62](#), [66](#)
- readSource(), [10](#), [15](#), [38](#), [41](#)
- readTau, [11](#), [32](#), [32](#), [34](#)
- redirect, [33–35](#)
- redirectSource, [33](#), [43](#)
- redirectSource(), [22](#)
- redirectTau, [35](#)
- regionscode, [35](#), [47](#)
- regionscode(), [47](#)
- resetMadratMessages, [36](#)
- resetMadratMessages(), [25](#), [30](#), [66](#)
- retrieveData, [16](#), [29](#), [37](#)
- retrieveData(), [10](#), [15](#), [32](#), [41](#)
- returnCalcOutput, [39](#)
- returnCalcOutput(), [10](#), [15](#), [32](#), [38](#)
- round(), [8](#)
  
- setConfig, [5](#), [10](#), [12](#), [15–17](#), [19](#), [26](#), [27](#), [32](#), [38](#), [41](#)
- setConfig(), [19](#), [27](#)
- signif(), [8](#)
  
- toolAggregate, [44](#), [55](#)
- toolAggregate(), [29](#)
- toolCodeLabels, [36](#), [47](#)
- toolCodeLabels(), [36](#)
- toolCompareStatusLogs, [48](#)
- toolCompareStatusLogs(), [12](#), [13](#)
- toolConditionalReplace, [49](#)
- toolConditionalReplace(), [55](#), [60](#), [65](#)
- toolConvertMapping, [49](#), [50](#), [56](#), [57](#)
- toolConvertMapping(), [5](#), [57](#)
- toolCountry2isocode, [50](#)
- toolCountry2isocode(), [21](#), [52](#), [54](#), [58](#)
- toolCountryFill, [51](#)
- toolCountryFill(), [21](#), [51](#), [54](#), [58](#)
- toolFillWithRegionAvg, [53](#)
- toolFillWithRegionAvg(), [21](#), [51](#), [52](#), [58](#)

toolFillYears, [54](#)  
toolFillYears(), [63](#), [64](#)  
toolFixWeight, [46](#), [55](#)  
toolFixWeight(), [49](#), [60](#), [65](#)  
toolGetMapping, [45](#), [56](#)  
toolGetMapping(), [5](#), [50](#)  
toolISOhistorical, [57](#)  
toolISOhistorical(), [21](#), [51](#), [52](#), [54](#)  
toolManualDownload, [58](#)  
toolManualDownload(), [28](#)  
toolNAreplace, [59](#)  
toolNAreplace(), [49](#), [55](#), [60](#), [65](#)  
toolOrderCells, [60](#)  
toolOrderCells(), [49](#), [55](#), [60](#), [65](#)  
toolSplitSubtype, [61](#)  
toolSplitSubtype(), [62](#)  
toolstartmessage, [28](#)  
toolSubtypeSelect, [62](#)  
toolSubtypeSelect(), [61](#)  
toolTimeAverage, [63](#)  
toolTimeAverage(), [55](#), [64](#)  
toolTimeSpline, [63](#)  
toolTimeSpline(), [55](#), [63](#)  
toolXlargest, [64](#)  
toolXlargest(), [49](#), [55](#), [60](#)

vcats, [28](#), [65](#)  
vcats(), [25](#), [30](#), [37](#)  
visualizeDependencies, [66](#)  
visualizeDependencies(), [20](#), [23](#), [24](#)