

# Package ‘madrat’

October 11, 2021

**Type** Package

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

**Version** 2.3.2

**Date** 2021-10-07

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

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

**Depends** R(>= 2.10.0), magclass(>= 5.7.0)

**Imports** igraph (>= 1.0.1), Matrix, tools, utils, digest, methods, stringi, yaml, withr

**Suggests** knitr, rmarkdown, testthat, tibble, ggplot2, covr

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

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

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

**LazyData** no

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**VignetteBuilder** knitr

**NeedsCompilation** no

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

David Klein [aut],  
 Pascal Führlich [aut]

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

**Repository** CRAN

**Date/Publication** 2021-10-11 07:40:14 UTC

## R topics documented:

madrat-package . . . . .	3
cacheArgumentsHash . . . . .	3
cacheCopy . . . . .	4
cacheGet . . . . .	5
cacheName . . . . .	6
cachePut . . . . .	7
calcOutput . . . . .	8
calcTauTotal . . . . .	10
compareData . . . . .	11
convertTau . . . . .	11
downloadSource . . . . .	12
findBottlenecks . . . . .	13
fingerprint . . . . .	14
fullEXAMPLE . . . . .	15
getCalculations . . . . .	16
getCode . . . . .	17
getConfig . . . . .	18
getDependencies . . . . .	19
getISOList . . . . .	20
getMadratGraph . . . . .	21
getMadratInfo . . . . .	22
getMainfolder . . . . .	22
getNonDefaultArguments . . . . .	23
getSources . . . . .	24
initializeConfig . . . . .	25
installedMadratUniverse . . . . .	26
madapply . . . . .	26
madlapply . . . . .	27
metadataGFZ . . . . .	27
prepFunctionName . . . . .	28
readSource . . . . .	29
readTau . . . . .	30
regionscode . . . . .	31
retrieveData . . . . .	32
robustOrder . . . . .	33
setConfig . . . . .	33
toolAggregate . . . . .	36
toolCodeLabels . . . . .	39
toolConditionalReplace . . . . .	40

toolConvertMapping . . . . .	40
toolCountry2isocode . . . . .	41
toolCountryFill . . . . .	42
toolendmessage . . . . .	43
toolFillWithRegionAvg . . . . .	44
toolFillYears . . . . .	46
toolGetMapping . . . . .	46
toolISOhistorical . . . . .	47
toolNAreplace . . . . .	48
toolOrderCells . . . . .	49
toolSplitSubtype . . . . .	50
toolstartmessage . . . . .	51
toolSubtypeSelect . . . . .	52
toolTimeAverage . . . . .	53
toolTimeSpline . . . . .	53
toolXlargest . . . . .	54
vcats . . . . .	55

**Index** **56**

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

**Description**

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

**Details**

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

cacheArgumentsHash *Tool: cacheArgumentsHash*

**Description**

Calculate hash from given function arguments for given call

**Usage**

cacheArgumentsHash(call, args = NULL)

**Arguments**

call	A function as a string or symbol. Passing a vector of functions is possible, but is only intended for corresponding read/correct/convert functions. If multiple functions in a vector define arguments with the same name but different default values only the default defined in the first function is considered.
args	A list of named arguments used to call the given function(s). If duplicates of arguments exists the first occurrence of the argument will be used.

**Value**

A hash representing the given arguments hash for the given call. NULL, if no argument deviates from the default argument settings.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[cachePut](#), [cacheName](#), [getNonDefaultArguments](#)

**Examples**

```

madrat:::cacheArgumentsHash("madrat:::readTau", args = list(subtype = "historical"))
madrat:::cacheArgumentsHash("madrat:::readTau", args = list(subtype = "paper"))
calls <- c(madrart:::readTau, madrat:::convertTau)
madrat:::cacheArgumentsHash(calls, args = list(subtype = "historical"))

```

---

cacheCopy

*cacheCopy*

---

**Description**

Copy cache files which were used for a given preprocessing

**Usage**

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

**Arguments**

file	path to a log file or content of a log as character vector
target	folder to which the files should be copied. If NULL no data is copied.
filter	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

---

cacheGet

*Tool: cacheGet*

---

**Description**

Load fitting cache data (if available)

**Usage**

```
cacheGet(prefix, type, args = NULL, graph = NULL, ...)
```

**Arguments**

prefix	function prefix (e.g. "calc" or "read")
type	output type (e.g. "TauTotal")
args	a list of named arguments used to call the given function
graph	A madrat graph as returned by <a href="#">getMadratGraph</a> . Will be created with <a href="#">getMadratGraph</a> if not provided.
...	Additional arguments for <a href="#">getMadratGraph</a> in case that no graph is provided (otherwise ignored)

**Value**

cached data, if cache is available, otherwise NULL

**Author(s)**

Jan Philipp Dietrich

**See Also**

[cachePut](#), [cacheName](#)

**Examples**

```
madrat:::cacheGet("calc", "TauTotal", packages="madrat")
```

---

cacheName	<i>Tool: cacheName</i>
-----------	------------------------

---

### Description

Load fitting cache data (if available)

### Usage

```
cacheName(
  prefix,
  type,
  args = NULL,
  graph = NULL,
  mode = "put",
  packages = getConfig("packages"),
  globalenv = getConfig("globalenv")
)
```

### Arguments

prefix	function prefix (e.g. "calc" or "read")
type	output type (e.g. "TauTotal")
args	a list of named arguments used to call the given function
graph	A madrat graph as returned by <a href="#">getMadratGraph</a> . Will be created with <a href="#">getMadratGraph</a> if not provided.
mode	Context in which the function is used. Either "get" (loading) or "put" (writing). In case of "put" the potential file name is returned. When set to "get", a file name will only be returned if the file exists (otherwise NULL) and in combination which <code>setConfig(forcecache=TRUE)</code> even a cache file with deviating hash might get selected.
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

cached data, if cache is available, otherwise NULL

### Note

`setConfig(forcecache=TRUE)` strongly affects the behavior of `cacheName`. In read model it will also return cache names with deviating hashes if no fitting cache file is found (in that case it will just return the newest one). In write mode the hash in the name will be left out since due to cache forcing it cannot be guaranteed that the cache file agrees with the state represented by the hash.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[cachePut](#), [cacheName](#)

**Examples**

```
madrat:::cacheName("calc", "TauTotal")
```

---

cachePut

*Tool: cachePut*

---

**Description**

Save data to cache

**Usage**

```
cachePut(x, prefix, type, args = NULL, graph = NULL, ...)
```

**Arguments**

x	data that should be written to cache
prefix	function prefix (e.g. "calc" or "read")
type	output type (e.g. "TauTotal")
args	a list of named arguments used to call the given function
graph	A madrat graph as returned by <a href="#">getMadratGraph</a> Will be created with <a href="#">getMadratGraph</a> if not provided.
...	Additional arguments for <a href="#">getMadratGraph</a> in case that no graph is provided (otherwise ignored)

**Author(s)**

Jan Philipp Dietrich

**See Also**

[cachePut](#), [cacheName](#)

**Examples**

```
## Not run:  
example <- 1  
madrat:::cachePut(example, "calc", "Example", packages="madrat")  
  
## End(Not run)
```

---

 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,
  supplementary = FALSE,
  append = FALSE,
  na_warning = TRUE,
  try = FALSE,
  ...
)
```

### Arguments

type	output type, e.g. "TauTotal". A list of all available source types can be retrieved with function <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	A rounding factor. If set to NULL no rounding will occur.
supplementary	boolean deciding whether supplementary information such as weight should be returned or not. If set to TRUE a list of elements will be returned!
append	boolean deciding whether the output data should be appended in the existing file. Works only when a file name is given in the function call.
na_warning	boolean deciding whether NAs in the data set should create a warning or not
try	if set to TRUE the calculation will only be tried and the script will continue even if the underlying calculation failed. If set to 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.
...	Additional settings directly forwarded to the corresponding calculation function

**Value**

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

**Note**

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

- **x** - the data itself as magclass object
- **weight** - a weight for the spatial aggregation
- **unit** - unit of the provided data
- **description** - a short description of the data
- **note** (optional) - additional notes related to the data
- **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.

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

**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)
```

**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

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

---

convertTau	<i>Convert Tau</i>
------------	--------------------

---

**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)
```

**Arguments**

type                source type, e.g. "IEA". A list of all available source types can be retrieved with function `getSources("download")`.

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.

**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

### See Also

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

### Examples

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

---

findBottlenecks      *findBottlenecks*

---

### 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 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

---

fingerprint	<i>Tool: fingerprint</i>
-------------	--------------------------

---

**Description**

Function which creates a unique fingerprint for a madrat function based on the code of the function itself, other madrat functions which are called by this function and of all source folders involved in the process. The fingerprint can serve as an indication whether the workflow for the given function has been most likely changed, or not. If all involved source folders and the code of all involved functions remains the same, also the fingerprint will stay the same, otherwise it will change. Hence, it can be to figure out whether a cache file can be used for further calculations, or whether the calculation should be redone.

**Usage**

```
fingerprint(name, details = FALSE, graph = NULL, ...)
```

**Arguments**

name	Name of the function to be analyzed
details	Boolean indicating whether additional details in form of an attribute with underlying hash information should be added or not
graph	A madrat graph as returned by <a href="#">getMadratGraph</a> . Will be created with <a href="#">getMadratGraph</a> if not provided.
...	Additional arguments for <a href="#">getMadratGraph</a> in case that no graph is provided (otherwise ignored)

**Value**

A md5-based fingerprint of all provided sources

**Note**

For a better performance only the first 300 bytes of each file and the corresponding file size is hashed. As the fingerprint function only takes madrat-based functions into account (e.g. read-functions or calc-functions), but does ignore all other functions there might be cases where calculations actually changed, but the fingerprint is still the same. In a similar fashion it is possible that the fingerprint changes even though the workflow stayed the same (as the dependencies are sometimes overestimated).

**Author(s)**

Jan Philipp Dietrich, Pascal Führlich

**See Also**

[readSource](#)

**Examples**

```
madrat::fingerprint("toolGetMapping", package = "madrat")
```

---

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 = "")
```

**Arguments**

rev	data revision which should be used/produced. Format must be compatible 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. <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)
```

---

<code>getCalculations</code>	<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**

<code>prefix</code>	Type of calculations, vector of types or search term (e.g. "readlcalc"). 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)
<code>packages</code>	A character vector with packages for which the available Sources/Calculations should be returned
<code>globalenv</code>	Boolean deciding whether sources/calculations in the global environment should be included or not

**Value**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

**Examples**

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

---

`getCode`*getCode*

---

**Description**

Extract function code from madrat-style functions in specified packages

**Usage**

```
getCode(  
  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 named vector with condensed function code

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getMadratGraph](#)

---

`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**

<code>option</code>	The option for which the setting should be returned. If set to NULL all options are returned.
<code>raw</code>	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.
<code>verbose</code>	boolean deciding whether status information/updates should be shown or not
<code>print</code>	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

**Author(s)**

Jan Philipp Dietrich

**See Also**

[setConfig](#), [initializeConfig](#)

---

getDependencies	<i>getDependencies</i>
-----------------	------------------------

---

**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](#)

---

`getISOList`*get official ISO list*

---

**Description**

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

**Usage**

```
getISOList(type = "all")
```

**Arguments**

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

**Value**

vector of default ISO country codes.

**Note**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getSources](#), [getCalculations](#)

**Examples**

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

---

getMadratGraph	<i>getMadratGraph</i>
----------------	-----------------------

---

### 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](#)

---

getMadratInfo	<i>getMadratInfo</i>
---------------	----------------------

---

### 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](#)

---

getMainfolder	<i>getMainfolder</i>
---------------	----------------------

---

### Description

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

### Usage

```
getMainfolder(verbose = TRUE, .testmode = FALSE)
```

**Arguments**

verbose	boolean deciding whether status information/updates should be shown or not
.testmode	boolean switch only relevant for internal testing (will simulate user inputs)

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

---

getNonDefaultArguments

*getNonDefaultArguments*

---

**Description**

Given a function and an argument list, identify which arguments are different from their default.

**Usage**

```
getNonDefaultArguments(call, args = NULL)
```

**Arguments**

call	A function name as a string or symbol. Passing a vector of functions is possible, but is only intended for corresponding read/correct/convert functions. If multiple functions in a vector define arguments with the same name but different default values only the default defined in the first function is considered.
args	A list of named arguments used to call the given function(s). If duplicates of arguments exists the first occurrence of the argument will be used.

**Value**

A subset of args that is used by the function/s and is different from default values.

**Author(s)**

Jan Philipp Dietrich

**See Also**

[cacheArgumentsHash](#), [toolstartmessage](#)

**Examples**

```

madrat::

```

---

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](#)

**Examples**

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

---

`initializeConfig`      *initializeConfig*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

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 loaded madrat packages and all packages with a name starting with "mr" or "ms" (as the usual indicator for madrat-packages and madrat-support-packages).

**Usage**

```
installedMadratUniverse()
```

**Value**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

[setConfig](#)

**Examples**

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

---

madapply                      *madapply*

---

**Description**

This function is defunct and will be completely removed soon.

**Usage**

```
madapply(...)
```

**Arguments**

...                      placeholder

**Author(s)**

Jan Philipp Dietrich

---

*madlapply*

*madlapply*

---

**Description**

This function is defunct and will be completely removed soon.

**Usage**

```
madlapply(...)
```

**Arguments**

... placeholder

**Author(s)**

Jan Philipp Dietrich

---

*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](#)

**Examples**

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

## End(Not run)
```

---

<code>prepFunctionName</code>	<i>prepFunctionName</i>
-------------------------------	-------------------------

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

## Examples

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

---

readSource

*readSource*

---

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Author(s)

Jan Philipp Dietrich, Anastasis Giannousakis, Lavinia Baumstark

## See Also

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

### Examples

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

---

readTau	<i>Read Tau</i>
---------	-----------------

---

### 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")
```

---

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 colums (if the first colum 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](#)

### Examples

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

---

`retrieveData`*retrieveData*

---

**Description**

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

**Usage**

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

**Arguments**

<code>model</code>	The names of the model for which the data should be provided (e.g. "magpie").
<code>rev</code>	data revision which should be used/produced. Format must be compatible to <a href="#">numeric_version</a> .
<code>dev</code>	development suffix to distinguish development versions for the same data revision. This can be useful to distinguish parallel lines of development.
<code>cachetype</code>	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.
<code>...</code>	(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> !)

**Author(s)**

Jan Philipp Dietrich, Lavinia Baumstark

**See Also**

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

**Examples**

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

---

robustOrder	<i>robustOrder, robustSort</i>
-------------	--------------------------------

---

**Description**

`robustOrder`: A wrapper around `base::order` that always uses the locale independent method = "radix". If the argument `x` is a character vector it is converted to utf8 first. `robustSort`: A convenience function using `order` to sort a vector using radix sort. The resulting vector will have the same encoding as the input although internally character vectors are converted to utf8 before ordering.

**Usage**

```
robustOrder(..., na.last = TRUE, decreasing = FALSE, method = "radix")
```

**Arguments**

<code>...</code>	One or more vectors of the same length
<code>na.last</code>	If TRUE missing values are put last, if FALSE they are put first, if NA they are removed
<code>decreasing</code>	If TRUE decreasing/descending order, if FALSE increasing/ascending order. For the "radix" method, this can be a vector of length equal to the number of arguments in <code>...</code> . For the other methods, it must be length one.
<code>method</code>	Default is "radix", which is locale independent. The alternatives "auto" and "shell" should not be used in madrat because they are locale dependent.

**Author(s)**

Pascal Führtlich

**See Also**

[order](#)

---

<code>setConfig</code>	<i>setConfig</i>
------------------------	------------------

---

**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,
  pop_threshold = NULL,
  nolabels = NULL,
  forcecache = NULL,
  ignorecache = NULL,
  cachecompression = NULL,
  hash = NULL,
  delete_cache = NULL,
  diagnostics = NULL,
  nocores = NULL,
  debug = NULL,
  indentationCharacter = NULL,
  maxLengthLogMessage = NULL,
  .cfgchecks = TRUE,
  .verbose = TRUE,
  .local = FALSE
)

```

**Arguments**

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	Boolean deciding whether data should be read from cache if data is available and the up-to-date (data will always be written to the cache regardless of this setting)
verbosity	an integer value describing the verbosity of the functions (2 = full information, 1 = only warnings and execution information, 0 = only warnings, -1 = no information)
mainfolder	The mainfolder where all data can be found and should be written to.

sourcefolder	The folder in which all source data is stored (in sub-folders with the name of the source as folder name). In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/sources
cachefolder	The folder in which all cache files should be written to. In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/cache
mappingfolder	A folder containing all kinds of mappings (spatial, temporal or sectoral). In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/mappings
outputfolder	The folder all outputs should be written to. In the default case this argument is set to NA meaning that the default folder should be used which is <mainfolder>/output
pop_threshold	Population threshold in capita which determines whether the country is put into the "important" or "dispensable" class in <code>getISOList</code> . This distinction is used for different treatment of countries in notifications to set a focus on rather critical issues instead of flooding the user with information.
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. <code>readTau</code> , <code>calcTauTotal</code> ) or type (e.g. <code>Tau</code> , <code>TauTotal</code> ) 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. <code>readTau</code> , <code>calcTauTotal</code> ) or types (e.g. <code>Tau</code> , <code>TauTotal</code> ) called by <code>calcOutput</code> or <code>readSource</code> . 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 <code>digest</code> can be used.
delete_cache	Boolean deciding whether a temporary cache folder (as created by <code>retrieveInput</code> ) should be deleted after completion or not.
diagnostics	file name for additional diagnostics information (without file ending). 2 log files are written if a file name is provided (a compact version with the most relevant information and a full version with all available details).
nocores	integer number of cores to use
debug	Boolean which activates a debug mode. In debug mode all calculations will be executed with <code>try=TRUE</code> 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 <code>debug=TRUE</code> will add the suffix "debug" to the files created to avoid their use in productive runs. Furthermore, with <code>debug=TRUE</code> calculations will be rerun even if a corresponding <code>tgz</code> file already exists.

indentationCharacter	character used for indenting the output of nested function calls
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
.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

**Author(s)**

Jan Philipp Dietrich

**See Also**

[getConfig](#), [getISOList](#)

**Examples**

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

## End(Not run)
```

---

toolAggregate

*toolAggregate*

---

**Description**

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

**Usage**

```
toolAggregate(
  x,
  rel,
  weight = NULL,
  from = NULL,
  to = NULL,
  dim = 1,
  wdim = NULL,
  partrel = FALSE,
```

```

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

```

## 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 parameter 'from').
weight	magclass object containing weights which should be considered for a weighted aggregation. The provided weight should only contain positive values, but does not need to be normalized (any positive number $\geq 0$ is allowed). Please see the "details" section below for more information.
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.
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

## 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

## See Also

[calcOutput](#)

## Examples

```
# create example mapping
p <- magclass::maxample("pop")
mapping <- data.frame(from = getRegions(p),
  region = rep(c("REG1", "REG2"), 5),
  global = "GLO")
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 not 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](#)

### Examples

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

toolConditionalReplace

*toolConditionalReplace*

---

### Description

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

### Usage

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

### Arguments

x	magpie 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

---

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](#)

---

toolCountry2isocode    *toolCountry2isocode*

---

**Description**

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

**Usage**

```
toolCountry2isocode(country, warn = TRUE, type = 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
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](#)

**Examples**

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

---

toolCountryFill	<i>Tool: CountryFill</i>
-----------------	--------------------------

---

**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**

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

verbosity	verbosity for information about filling important countries. 0 = warning will show up (recommended if filling of important countries is not expected), 1 = note will show up in reduced log file (default), 2 = info will show up in extended log file (recommended if filling of important countries is not critical and desired).
countrylist	character vector of official country names (if other than ISO)
...	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

**Examples**

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

---

toolendmessage	<i>Tool: End message</i>
----------------	--------------------------

---

**Description**

Function writes a process end message and performs some diagnostics

**Usage**

```
toolendmessage(startdata, level = NULL)
```

**Arguments**

startdata	a list containing diagnostic information provided by <a href="#">toolstartmessage</a>
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).

**Author(s)**

Jan Philipp Dietrich

**See Also**

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

**Examples**

```
innerFunction <- function() {
  startinfo <- madrat:::toolstartmessage(list(argumentsToPrint = 123), "+")
  vcat(1, "inner")
  madrat:::toolendmessage(startinfo, "-")
}
outerFunction <- function() {
  startinfo <- madrat:::toolstartmessage(list(), "+")
  vcat(1, "outer")
  innerFunction()
  madrat:::toolendmessage(startinfo, "-")
}
outerFunction()
```

---

toolFillWithRegionAvg *Tool: FillWithRegionAvg*

---

**Description**

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

**Usage**

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

**Arguments**

<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.

**Details**

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

**Value**

A MAGPIE object with the missing values filled.

**Author(s)**

Bjoern Soergel, Lavinia Baumstark

**Examples**

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

---

toolFillYears	<i>toolFillYears</i>
---------------	----------------------

---

**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

---

toolGetMapping	<i>Tool: GetMapping</i>
----------------	-------------------------

---

**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="#">codetoolConvertMapping</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 <code>getConfig("packages")</code>
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](#)

**Examples**

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

---

toolISOhistorical

*Tool: ISOhistorical*

---

**Description**

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

**Usage**

```

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

```

**Arguments**

<code>m</code>	MAGPIE object with ISO country codes in the spatial dimension
<code>mapping</code>	mapping of historical ISO countries to the standard ISO country list. For the default setting ( <code>mapping=NULL</code> ) the mapping stored as supplementary data in the <code>madrat</code> package is used. If provided as file the mapping needs to contain three columns "fromISO", "toISO" and "lastYear".
<code>additional_mapping</code>	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. <code>additional_mapping = c("TTT","TTX","y1111")</code> or <code>additional_mapping = list(c("TTT","TTX","y1111"),c("TTT","TTY","y1111"))</code>
<code>overwrite</code>	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 <code>overwrite=TRUE</code>

**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

---

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](#)

---

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



---

toolstartmessage      *Tool: Start message*

---

## Description

Function writes a process start message and performs some diagnostics

## Usage

```
toolstartmessage(argumentValues, level = NULL)
```

## Arguments

`argumentValues` list of the evaluated arguments of the calling function

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

## Value

a list containing diagnostic information required by [toolendmessage](#)

## Author(s)

Jan Philipp Dietrich, Pascal Führlich

## See Also

[toolendmessage](#), [vcats](#)

## Examples

```
innerFunction <- function() {
  startinfo <- madrat:::toolstartmessage(list(argumentsToPrint = 123), "+")
  vcat(1, "inner")
  madrat:::toolendmessage(startinfo, "-")
}
outerFunction <- function() {
  startinfo <- madrat:::toolstartmessage(list(), "+")
  vcat(1, "outer")
  innerFunction()
  madrat:::toolendmessage(startinfo, "-")
}
outerFunction()
```

---

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](#)

### 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

**Usage**

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

**Arguments**

x	magclass object that should be averaged
averaging_range	number of time steps to average (if annual=FALSE, please check consistency)
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
annual	if TRUE checks for completeness of annual coverage in the data set if FALSE no checks are performed

**Value**

the averaged data in magclass format

**Author(s)**

Kristine Karstens

---

toolTimeSpline	<i>toolTimeSpline</i>
----------------	-----------------------

---

**Description**

Smoothing a data set by replacing its values by its spline approximation using the given degrees of freedom.

**Usage**

```
toolTimeSpline(x, dof = NULL)
```

**Arguments**

x	magclass object that should be smoothed via a spline approximation
dof	degrees of freedom per 100 years (similar to an average range), is a proxy for the smoothness of the spline (smaller values = smoother)

**Value**

approximated data in magclass format

**Author(s)**

Kristine Karstens, Felicitas Beier

---

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.
...	further parameters will be handed on to calcOutput function type.

**Value**

vector with ISO country codes

**Author(s)**

Benjamin Leon Bodirsky, Jan Philipp Dietrich

**Examples**

```
toolXlargest(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)
```

## Arguments

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

## Author(s)

Jan Philipp Dietrich

## See Also

[readSource](#)

## Examples

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

# Index

cacheArgumentsHash, [3](#), [23](#)  
cacheCopy, [4](#)  
cacheGet, [5](#)  
cacheName, [4](#), [5](#), [6](#), [7](#)  
cachePut, [4](#), [5](#), [7](#), [7](#)  
calcOutput, [8](#), [10](#), [15](#), [32](#), [38](#), [41](#), [47](#), [49](#)  
calcTauTotal, [10](#), [10](#), [11](#)  
compareData, [11](#)  
convertTau, [10](#), [11](#)

digest, [31](#), [35](#)  
downloadSource, [12](#), [29](#)

findBottlenecks, [13](#)  
fingerprint, [14](#), [31](#)  
fullEXAMPLE, [15](#)

getCalculations, [8](#), [15](#), [16](#), [19–22](#)  
getCode, [17](#)  
getConfig, [18](#), [21](#), [23](#), [25](#), [36](#)  
getDependencies, [19](#)  
getISOList, [20](#), [35](#), [36](#)  
getMadratGraph, [5–7](#), [14](#), [17](#), [19](#), [21](#), [22](#)  
getMadratInfo, [19](#), [22](#)  
getMainfolder, [22](#), [25](#)  
getNonDefaultArguments, [4](#), [23](#)  
getSources, [12](#), [20](#), [24](#), [29](#), [42](#)

initializeConfig, [18](#), [23](#), [25](#)  
installedMadratUniverse, [26](#)

madapply, [26](#)  
madlapply, [27](#)  
madrat (madrat-package), [3](#)  
madrat-package, [3](#)  
metadataGFZ, [27](#)

numeric\_version, [15](#), [32](#)

order, [33](#)

prepFunctionName, [28](#)

readSource, [13](#), [15](#), [16](#), [25](#), [28](#), [29](#), [30](#), [42](#), [52](#),  
[55](#)  
readTau, [10](#), [29](#), [30](#)  
regionscode, [31](#), [39](#)  
retrieveData, [32](#)  
robustOrder, [33](#)

setConfig, [10](#), [11](#), [13](#), [15](#), [16](#), [18](#), [23](#), [25](#), [26](#),  
[28](#), [29](#), [32](#), [33](#)

toolAggregate, [36](#)  
toolCodeLabels, [31](#), [39](#)  
toolConditionalReplace, [40](#)  
toolConvertMapping, [40](#), [41](#), [47](#)  
toolCountry2isocode, [41](#)  
toolCountryFill, [42](#)  
toolendmessage, [43](#), [51](#)  
toolFillWithRegionAvg, [44](#)  
toolFillYears, [46](#)  
toolGetMapping, [37](#), [46](#)  
toolISOhistorical, [47](#)  
toolNAreplace, [48](#)  
toolOrderCells, [49](#)  
toolSplitSubtype, [50](#)  
toolstartmessage, [23](#), [28](#), [43](#), [44](#), [51](#)  
toolSubtypeSelect, [52](#)  
toolTimeAverage, [53](#)  
toolTimeSpline, [53](#)  
toolXlargest, [54](#)

vcat, [28](#), [44](#), [51](#), [55](#)