Package: gdx (via r-universe)

August 15, 2024

Type Package

Title Interface package for GDX files in R

Version 1.53.1

Date 2024-05-17

Description A wrapper package for the gdxrrw extending its functionality and allowing to read and write GDX files directly in R.

License BSD_2_clause + file LICENSE

URL https://github.com/pik-piam/gdx, https://doi.org/10.5281/zenodo.1158598

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

Depends gdxrrw (>= 1.0.2), magclass (>= 2.43)

Imports utils

Suggests covr

Encoding UTF-8

LazyData yes

RoxygenNote 7.3.1

Config/Keywords tool

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

RemoteUrl https://github.com/pik-piam/gdx

RemoteRef HEAD

RemoteSha 3a901f395740f5aa97f41f38acff12fb3ca8eca5

Contents

---

gdx-package *gdx: Interface package for GDX files in R*

Description

A wrapper package for the gdxrrw extending its functionality and allowing to read and write GDX files directly in R.

Author(s)

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

Authors:

- Anastasis Giannousakis <giannou@pik-potsdam.de>
- Markus Bonsch Bonsch

Other contributors:

- Lavinia Baumstark Baumstark <baumstark@pik-potsdam.de> (Potsdam Institute for Climate Impact Research) [contributor]

See Also

Useful links:

- https://github.com/pik-piam/gdx
- doi:10.5281/zenodo.1158598
- Report bugs at https://github.com/pik-piam/gdx/issues

addDimPrideGdx *addDimPrideGdx*

Description

Function to expand Pride Model GDX's.

Usage

```
addDimPrideGdx(file)
```

Arguments

file	File name of the gdx file

Author(s)

Anastasis Giannousakis

Examples

```
## Not run: addDimPrideGdx("bla.gdx")
```

calc_scaling *calc_scaling*

Description

This function creates a GAMS file with scaling of variables. The scaling is calculated based on a gdx file containing all variables of a run.

Usage

```
calc_scaling(gdx, file=NULL, magnitude=2)
```

Arguments

gdx	a GDX list as created by readGDX, or the file name of a gdx file
file	A file name the scaling GAMS code should be written to. If NULL the code is returned by the function
magnitude	The order of magnitude for which variables should be scaled. All variables with average absolute values which are either below 10e(-magnitude) or above 10e(magnitude) will be scaled.

Value

A vector with the scaling GAMS code if file=NULL, otherwise nothing is returned.

Author(s)

Jan Philipp Dietrich

See Also

out,readGDX

Examples

```
## Not run: calc_scaling("fulldata.gdx")
```

`expand.set` *expand.set*

Description

Function which expands a GAMS set based on a comparison set. If strings in the set, which should be expanded, exist in the fullset, they are used directly, otherwise it is searched for a set in the gdx file having that name

Usage

```
expand.set(gdx,x,fullset=NULL)
```

Arguments

`gdx`	a GDX list as created by readGDX, or the file name of a gdx file (file name is recommended as this speeds up the code)
`x`	A vector of strings which should be used as a set. Strings are either set element names or names of whole sets.
`fullset`	a vector of strings used as comparison. If only a single string is supplied it is checked whether this string exists as set in the given gdx file and expanded. In all other cases it is assumed that the strings are single set elements. x must be a subset of fullset! If fullset is NULL it is just assumed that x contains only set names which all have to be expanded.

Value

The expanded set vector.

Author(s)

Jan Philipp Dietrich

See Also

readGDX

Examples

```
## Not run: expand.set("fulldata.gdx","kbe","kcr")
```

---

gdx_rename *gdx_rename*

Description

Function to replace or delete names of variables, parameters or equations in gdx files (aliases and sets are currently not supported, but support could be added if required).

Usage

```
gdx_rename(file, ..., set_name = NULL)
```

Arguments

file	File name of the gdx file in which the objects or set entries should be renamed
...	For level=="objects": listing of renamings/deletions that should be done in the form oldname="newname" (or oldname=0 for delete). Alternatively a named vector could be provided c(oldname="newname",bla="blub",deleteme=0). For level=="set_names": A named vector should be provided c(newname1="oldname1",newname2="oldname It can select only some of the old set entries or use some old entries for more than one new set entry.
set_name	If you want to rename entries of a set you have to specify the name of the set. In the default case (set_name=NULL) you can rename objects.

Author(s)

Jan Philipp Dietrich, Lavinia Baumstark

See Also

readGDX, writeGDX

Examples

```
## Not run:

#list all objects in the given gdx file
gdx_rename("bla.gdx")

#replace oldname with newname and oldname2 with newname2
gdx_rename("bla.gdx",oldname="newname",oldname2="newname2")

#delete "deleteme"
gdx_rename("bla.gdx",deleteme=0)

# rename and select set entries of the set "testset"
gdx_rename("bla.gdx",set_name="testset",c(newentry1="oldentry1",newentry2="oldentry2"))

## End(Not run)
```

---

inp *inp* - ***DEPRECEATED!***

Description

This function is depreceated, please do not use it! Please see the note below for more information. Function to savely read parameters and sets from GDX file. The function creates a warning if the parameter does not exist in the gdx file. Please use this function when you write own GDX output functions.

Usage

```
inp(gdx, name, ..., react = "warning", as.magpie = FALSE)
```

Arguments

gdx	a GDX list as created by readGDX, or the file name of a gdx file (file name is recommended as this speeds up the code)
name	name of the parameter or set that should be read
...	Additional names. If the attempt to read 'name' fails, these will be tried in the order given by the user until one is successfully read.
react	determines the reaction, when the parameter or set does not exist. Available options are "warning" (NULL is returned and a warning is send that the parameter or set is missing), "silent" (NULL is returned, but no warning is given) and "error" (The function throws out an error)
as.magpie	If TRUE the content is returned as a MAgPIE object, otherwise as array

Value

The parameter if it exists, otherwise NULL

Note

This function is depreceated! Please use `readGDX(...,format="first_found")` instead! This will basically give you the same functionality with the only difference that it will return a magpie object instead of an array.

Author(s)

Jan Philipp Dietrich

See Also

`out`, `readGDX`

Examples

```
## Not run: inp("fulldata.gdx","x_modelstat")
```

`marginals_check` *marginals_check*

Description

This function returns a list of the highest marginals found in gdx file allowing to detect the most critical constraints in a given solution.

Usage

```
marginals_check(gdx, ifilter=NULL, efilter=NULL, limit=1e5,
scientific=TRUE, unlist=FALSE, name="o(v|q)*")
```

Arguments

`gdx`	a GDX list as created by readGDX, or the file name of a gdx file
`ifilter`	Inclusion filter. A vector of strings containing regular expressions which describe strings that have to be part of the found marginal names. Otherwise they will be excluded.
`efilter`	Exclusion filter. A vector of strings containing regular expressions which describe strings must not be part of the found marginal names. Otherwise they will be excluded.
`limit`	Lower limit for the absolute value of the marginal in order to be part of the output.
`scientific`	Boolean which decides whether the marginals should be written scientfically (e.g. 1e+2) or the default output scheme should be used.
`unlist`	If TRUE a vector sorted by the rank of the marginal will be returned, otherwise a structured list of outputs will be returned.
`name`	search string defining the objects that should be read from gdx file, with *-autocompletion. Can also be a vector containing more than one search strings

Value

A vector or list containing all marginals which absolute values are above the given limit and which agree with the given filters.

Author(s)

Jan Philipp Dietrich

See Also

readGDX

Examples

```
## Not run: marginals_check("fulldata.gdx")
```

`old_readGDX` *old_readGDX*

Description

Function to read gdx files in R. - **DEPRECEATED!**

Usage

```
old_readGDX(
  file,
  ...,
  restore_zeros = NULL,
  types = c("sets", "equations", "parameters", "variables", "aliases"),
  field = "All",
  format = "simplest"
)
```

Arguments

`file`	File name of the gdx file
`...`	search strings defining the objects that should be read from gdx file, with *-autocompletion. Can also be vectors containing more than one search strings
`restore_zeros`	Dummy argument for downwards compatibility. Not used anymore! Function will now always try to restore zeros
`types`	Types of objects that should be extracted. Available options are "sets", "equations", "parameters" and "variables".

`field` Defining what kind of information should be returned. "All" means all available data. Other options are "l" (level value), "m" (marginal), "lo" (lower bound), "up" (upper bound) and "s" (scaling factor). In the case that the level value is not part of the field value (all options other than "All" and "l") only values for equations and variables are returned as all other types do not have this kind of information

`format` Output format. Four choices are currently available "detailed", "simple", "simplest", "compact" and "raw". "detailed" is the old default which returns a list of lists separating the outputs first in type and afterwards in variable names. "simple" returns a list of variables. If there is more than one object returned "simplest" behaves exactly the same as "simple". However, if only one object is read from gdx file the array itself is returned completely getting rid of the list structure. "raw" is the data in the format as it comes from `rgdx`. This is especially useful the data should be written again to a gdx file without having much transformations in between. "simplest" is set by default because it is the most convenient output. However, as "simplest" either returns a list of array or an array itself it is harder to use in other functions. "compact" behaves like "simple" with the only difference that data is returned as magclass rather than arrays, which can be especially quite useful for sparse data. Or if you anyway plan to work with magclass objects rather than arrays.

Value

The gdx objects read in the format set with the argument `format`.

Note

This function is depreceated! Please use `readGDX` instead!

Author(s)

Jan Philipp Dietrich

See Also

`readGDX`, `writeGDX`

Examples

```
## Not run: old_readGDX("bla.gdx","blub*")
```

out *out*

Description

Function to savely returns parameters. Function returns either the output or writes it to a file. Please use this function when you write own GDX output functions.

Usage

```
out(x,file)
```

Arguments

x	an object that can be converted to a MAgPIE object
file	file name of a file it should be written to. NULL, if x should be returned instead to be written to a file.

Value

NULL or x as MAgPIE object

Author(s)

Jan Philipp Dietrich

See Also

in,readGDX

Examples

```
out(12,NULL)
```

readGDX *readGDX*

Description

Function to read gdx files in R. It is partly a reimplementation of readGDX which is now based on magclass structures rather than array structures.

Usage

```
readGDX(
  gdx,
  ...,
  types = c("sets", "equations", "parameters", "variables", "aliases"),
  field = "All",
  format = "simplest",
  restore_zeros = TRUE,
  react = "warning",
  spatial = NULL,
  temporal = NULL,
  select = NULL,
  collapseNames = TRUE,
  magpie_cells = TRUE
)
```

Arguments

gdx	Either file name of a gdx file or an already read in gdx (in the latter case readGDX just acts as a filter. This can be useful if you want to apply several functions on the same gdx file. In that case you could read in the gdx first and then filter the data you need using readGDX.)
...	search strings defining the objects that should be read from gdx file, with *-autocompletion. Can also be vectors containing more than one search strings
types	Types of objects that should be extracted. Available options are "sets", "equations", "parameters", "variables" and "aliases".
field	Defining what kind of information should be returned. "All" means all available data. Other options are "l" (level value), "m" (marginal), "lo" (lower bound), "up" (upper bound) and "s" (scaling factor). In the case that the level value is not part of the field value (all options other than "All" and "l") only data for equations and variables are returned as all other types do not have this kind of information. WARNING: field has to be set to "All" if the data is planned to be written back to a GDX. Otherwise writeGDX will not work!
format	Output format. Five choices are currently available `detailed`, `simple`, `simplest`, `raw` and `first_found`. Instead of writing the full format name each format has its own abbreviation as shown below. **list("detailed")** This is the old default which returns a list of lists separating the outputs first in type and afterwards in variable names. **(d)** This is the old default which returns a list of lists separating the outputs first in type and afterwards in variable names. **list("simple")** This returns a list of outputs. **(s)** This returns a list of outputs. **list("simplest")** Behaves like "simple" if more than one object is returned. However, if only one object is read from gdx file the magpie object itself is returned getting rid of the surrounding list structure. This is the recommended format for interactive use.

(st - default setting) Behaves like "simple" if more than one object is returned. However, if only one object is read from gdx file the magpie object itself is returned getting rid of the surrounding list structure. This is the recommended format for interactive use.

list("raw") This returnes the data as it comes from `rgdx`. This is especially useful the data should be written again to a gdx file without having much transformations in between.

(r) This returnes the data as it comes from `rgdx`. This is especially useful the data should be written again to a gdx file without having much transformations in between.

list("first_found") This is a special format for the case that you would like to read in exactly one object but you do not know exactly what the name of the object is. Here, you can list all possible names of the object and the function will return the first object of the list which is found. This is especially useful writing read functions for gdx outputs of models in which the names of a data object might change over time but the function itself should work for all model versions. Having this format helps to make your gdx-based functions backwards compatible to older versions of a gdx file with different naming.

(f) This is a special format for the case that you would like to read in exactly one object but you do not know exactly what the name of the object is. Here, you can list all possible names of the object and the function will return the first object of the list which is found. This is especially useful writing read functions for gdx outputs of models in which the names of a data object might change over time but the function itself should work for all model versions. Having this format helps to make your gdx-based functions backwards compatible to older versions of a gdx file with different naming.

list("name") In this case the function returns the name of all objects found in the gdx which fit to the given search pattern and the given type as vector.

(n) In this case the function returns the name of all objects found in the gdx which fit to the given search pattern and the given type as vector.

`restore_zeros` Defines whether 0s, which are typically not stored in a gdx file, should be restored or ignored in the output. By default they will be restored. If possible, it is recommended to use restore_zeros=TRUE. It is faster but more memory consuming. If you get memory errors you should use restore_zeros=FALSE

`react` determines the reaction, when the object you would like to read in does not exist. Available options are "warning" (NULL is returned and a warning is send that the object is missing), "silent" (NULL is returned, but no warning is given) and "error" (The function throws out an error)

`spatial` argument to determine the spatial columns in the dataframe to be converted to a magclass object. Defaults to NULL. See `as.magpie` for more information.

`temporal` argument to determine the temporal columns in the dataframe to be converted to a magclass object. Defaults to NULL. See `as.magpie` for more information.

`select` preselection of subsets in the data coming from the gdx using the function `mselect`. Information has to be provided as a list of selections (e.g. `select=list(type="level")`). See `mselect` for more information.

collapseNames	Boolean which determines whether collapseNames should be applied in mselect or not.
magpie_cells	(boolean) determines whether a set "j" gets special treatment by replacing underscores in the set elements with dots. Active by default for historical reasons. Can be ignored in most cases. Makes only a difference, if 1) GDX element depends on set "j", 2) set "j" contains underscores.

Value

The gdx objects read in the format set with the argument format.

Author(s)

Jan Philipp Dietrich

See Also

writeGDX, mselect

Examples

```
## Not run:
readGDX("bla.gdx", "blub*")

## End(Not run)
```

---

writeGDX *writeGDX*

Description

Function to write gdx files in R.

Usage

```
writeGDX(x,file,period_with_y = TRUE)
```

Arguments

x	A list of objects or a single object that should be written to a gdx file. The format is identical to what you get as output by using readGDX (all formats provided by readGDX are supported by writeGDX as well).
file	File name of the gdx file
period_with_y	Keep "y" in period dimension

Author(s)

Jan Philipp Dietrich

See Also

readGDX

Examples

```
## Not run: writeGDX(readGDX("input.gdx"),"test.gdx")
```

Index