Package: remind2 (via r-universe)

September 4, 2024

Type Package

Title The REMIND R package (2nd generation)

Version 1.154.0

Date 2024-09-02

Description Contains the REMIND-specific routines for data and model output manipulation.

License LGPL-3

URL https://github.com/pik-piam/remind2

Depends magclass (>= 6.16.1)

Imports abind, assertr, data.table, dplyr (>= 1.1.1), gdx (>= 1.53.0), gdxrrw, ggplot2, gms, lucode2 (>= 0.43.0), lusweave, madrat (>= 3.13.0), mip (>= 0.149.2), openxlsx, piamInterfaces (>= 0.17.11), piamPlotComparison (>= 0.0.10), piamutils, plotly (>= 4.10.4), quitte (>= 0.3132.0), readr, remulator, reshape2, rlang, rmarkdown, tibble, tidyr, tidyselect, withr, digest

Suggests covr, gridExtra, htmltools, kableExtra, knitr, testthat, tidyverse

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/parallel true

Config/testthat/edition 3

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

RemoteUrl https://github.com/pik-piam/remind2

RemoteRef HEAD

RemoteSha 0089636796c56aad2eb80d2ae77d50516124df89

Contents

`remind2-package` *The REMIND R package (2nd generation)*

Description

Contains the REMIND-specific routines for data and model output manipulation.

Author(s)

Maintainer: Renato Rodrigues <renato.rodrigues@pik-potsdam.de>

Authors:

- Lavinia Baumstark
- Falk Benke
- Jan Philipp Dietrich

- Alois Dirnaichner
- Jakob Duerrwaechter
- Pascal Führlich
- Anastasis Giannousakis
- Robin Hasse
- Jérome Hilaire
- David Klein
- Johannes Koch
- Katarzyna Kowalczyk
- Antoine Levesque
- Aman Malik
- Anne Merfort
- Leon Merfort
- Simón Morena-Leiva
- Michaja Pehl
- Robert Pietzcker
- Sebastian Rauner
- Oliver Richters
- Marianna Rottoli
- Christof Schötz
- Felix Schreyer
- Kais Siala
- Björn Sörgel
- Mike Spahr
- Jessica Strefler
- Philipp Verpoort
- Pascal Weigmann
- Tonn Rüter <tonn.rueter@pik-potsdam.de>

See Also

Useful links:

- https://github.com/pik-piam/remind2

calcNetTrade *Calculate net trade from GDX file*

Description

Calculate net trade in a commodity from a GDX file and store it into a magpie object.

Usage

```
calcNetTrade(gdx, variable)
```

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)
variable	one of the traded commodities in REMIND: good, pecoal, pegas, peoil, pebiolc, peur, perm

Value

Trade data as MAgPIE object

Author(s)

Niklas Roming

See Also

readTrade, calcNetTradeValue

Examples

```
  ## Not run:
    coal <- calcNetTrade(gdx, "pecoal")
    gas <- calcNetTrade(gdx, "pegas")

## End(Not run)
```

`calcNetTradeValue` *Calculate monetary value of trade in primary energy from GDX file*

Description

Calculate monetary value of trade in primary energy carriers in a GDX file and return as a MagPIE object.

Usage

```
calcNetTradeValue(gdx, variable)
```

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)
`variable`	one of the primary energy tyes in REMIND that ar: pecoal, pegas, peoil, pebiolc, peur

Value

Value (in US$2005) of trade

Author(s)

Niklas Roming

See Also

`calcNetTrade`, `calcPrice`

Examples

```
  ## Not run:
    coal <- calcNetTradeValue(gdx, "pecoal")
    gas <- calcNetTradeValue(gdx, "pegas")

## End(Not run)
```

`calcPrice` *Calculates normalized prices of a commodity*

Description

Read PVP data of a certain commodity from a GDX file into a magpie object and normalize it with the 2005 goods price.

Usage

```
calcPrice(gdx, level = "glo", enty = "good", type = "nominal")
```

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)
`level`	spartiel resolution "glo" = global or "reg" = regional
`enty`	one of the traded commodities in REMIND: good, pecoal, pegas, peoil, pebiolc, peur, perm
`type`	calculate present values (PVP(t)/PVP_Good(2005)) or nominal values(PVP(t)/PVP_Good(t))

Value

normalized price data as MAgPIE object

Author(s)

Niklas Roming

Examples

```
  ## Not run:
    coal <- calcPrice(gdx, enty = "pecoal")
    gas <- calcPrice(gdx, enty = "pegas")

## End(Not run)
```

calc_CES_marginals *Calculate CES Marginals*

Description

Calculate marginals on the REMIND CES function and combine them to prices.

Usage

```
calc_CES_marginals(gdxName, id = "file")
```

Arguments

gdxName	Vector of paths to .gdx files.
id	If several .gdx files are read, an id column is appended to the result; either file, with the paths of the originating .gdx files, or scenario, with the content of c_expname.

Details

Marginals are calculated analytically

$$\frac{\partial V_i}{\partial V_o} = \xi_i(\theta_i\delta_i)^{\rho_o} {V_o}^{1-\rho_o} {V_i}^{\rho_o-1}$$

and prices by recursively applying the chain rule

$$\pi_i = \frac{\partial V_i}{\partial V_o}\pi_o \quad \forall(i,o) \in CES$$

Value

A data frame with columns pf (production factor), t, regi, marginal, price, and file (path to originating .gdx file).

calc_regionSubset_sums *Calculate Sums for Region Subsets*

Description

Sum up values in data for all sets of regions in regionSubsetList.

Usage

```
calc_regionSubset_sums(data, regionSubsetList)
```

Arguments

data	A MAgPIE object.
regionSubsetList	A list of region subsets to calculate of the form list(subset_name = c(region_A, region_B, ...)

Value

A MAgPIE object.

Author(s)

Michaja Pehl

Examples

```
calc_regionSubset_sums(population_magpie, list(xAM = c('LAM', 'NAM')))
```

---

checkVsCalibData *Render checkVsCalibData*

Description

Render checkVsCalibData

Usage

```
checkVsCalibData(
  gdx,
  outputDir = getwd(),
  outputFile = "Check_vs_CalibData.pdf"
)
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
outputDir	character(1). The directory where the output document and intermediary files are created.
outputFile	character(1). File name (without extension) of the output document to be created.

Value

The value returned by rmarkdown::render().

Author(s)

Falk Benke

Examples

```
## Not run:
# Simple use. Creates PDF:
checkVsCalibData(
  gdx = "path/to/data.gdx",
  outputDir = "path/to/output/directory",
  outputFile = "Check_vs_CalibData_Example.pdf"
)

## End(Not run)
```

`colorScenConf` *take scenario-config*.csv files and produce colorful xlsx file with values from main.gms in first row, identical values in this column in turquoise, and unknown column names in red*

Description

take scenario-config*.csv files and produce colorful xlsx file with values from main.gms in first row, identical values in this column in turquoise, and unknown column names in red

Usage

```
colorScenConf(fileList = "", remindPath = ".", expanddata = FALSE)
```

Arguments

`fileList`	vector containing csv file paths
`remindPath`	path to REMIND directory containing the main.gms
`expanddata`	fill all the data based on copyConfigFrom and main.gms

Value

nothing. For each file in fileList a _colorful.xlsx file is written.

Author(s)

Oliver Richters

Examples

```
## Not run:
   colorScenConf(fileList = c("scenario_config.csv"), remindPath = ".")

## End(Not run)
```

compareCalibrationTargets *Render compareCalibrationTargets*

Description

Render compareCalibrationTargets

Usage

```
compareCalibrationTargets(
  gdxPaths,
  outputDir = getwd(),
  outputFile = "compareCalibrationTargets.html"
)
```

Arguments

gdxPaths	character(n), optionally named. Paths to GDX objects. If the vector has names, those are used to refer to the scenarios in the output file.
outputDir	The directory where the output document and intermediary files are created.
outputFile	File name (without extension) of the output document to be created.

Value

The value returned by `rmarkdown::render()`.

Author(s)

Falk Benke

Examples

```
## Not run:
# Simple use.
checkVsCalibData(
  gdxPaths = c("path/to/fulldata.gdx", "another path/to/fulldata.gdx"),
  outputDir = "path/to/output/directory",
  outputFile = "myComparison.html"
)

## End(Not run)
```

compareScenarios2 *Render CompareScenarios2*

Description

A deprecated function. Please refer to piamPlotComparison::compareScenarios

Usage

```
compareScenarios2(...)
```

Arguments

...	parameters to be passed on to piamPlotComparison::compareScenarios

Value

The value returned by `rmarkdown::render()`.

compareScenConf *take two REMIND scenario-config*.csv files and print the difference, comparing it to a default.cfg*

Description

take two REMIND scenario-config*.csv files and print the difference, comparing it to a default.cfg

Usage

```
compareScenConf(
  fileList = NULL,
  remindPath = "/p/projects/rd3mod/github/repos/remindmodel/remind/develop",
  row.names = 1,
  renamedCols = NULL,
  renamedRows = NULL,
  printit = TRUE,
  expanddata = TRUE
)
```

Arguments

fileList	vector containing one csv file paths or two paths as c(oldfile, newfile) If one, searches same filename in defaultPath. If NULL, user can select
remindPath	path to REMIND directory containing main.gms
row.names	column in csv used for row.names. Use NULL for mapping files
renamedCols	vector with old and new column names such as c("old1" = "new1", "old2" = "new2"))
renamedRows	vector with old and new row names such as c("old3" = "new3", "old3" = "new4", "old5" = "new5")) the "old" name can also remain in the new file, if you generated a variant
printit	boolean switch (default: TRUE) whether function prints its output
expanddata	fill empty cells with default values

Value

list with $allwarnings and $out

Author(s)

Oliver Richters

Examples

```
 ## Not run:
    compareScenConf(fileList = c("scenario_config_old.csv", "scenario_config_new.csv"),
    renamedCols = c("old1" = "new1", "old2" = "new2"),
    renamedRows = c("old3" = "new3", "old4" = "new4"))

## End(Not run)
```

---

convGDX2CSV_LCOE *Read in GDX and write LCOE .csv reporting*

Description

Read in all information from GDX file, calculate the LCOE for pe2se technologies and create the LCOE .csv reporting

Usage

```
convGDX2CSV_LCOE(
  gdx,
  file = NULL,
  scen = "default",
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
file	name of the csv file which will be written
scen	scenario name that is used in the *.mif reporting
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2030,2050)

Author(s)

Felix Schreyer

Examples

```
## Not run: convGDX2CSV_LCOE(gdx,file="REMIND_LCOE_reporting.csv",scenario="default")
```

---

convGDX2MIF *Read in GDX and write *.mif reporting*

Description

Read in all information from GDX file and create the *.mif reporting

Usage

```
convGDX2MIF(
  gdx,
  gdx_ref = NULL,
  file = NULL,
  scenario = "default",
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150),
  gdx_refpolicycost = gdx_ref,
  testthat = FALSE
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
gdx_ref	reference-gdx for < cm_startyear, used for fixing the prices to this scenario
file	name of the mif file which will be written, if no name is provided a magpie object containing all the reporting information is returned
scenario	scenario name that is used in the *.mif reporting
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)
gdx_refpolicycost	reference-gdx for policy costs, a GDX as created by readGDX, or the file name of a gdx
testthat	boolean whether called by tests, turns some messages into warnings

Author(s)

Lavinia Baumstark

Examples

```
## Not run: convGDX2MIF(gdx,gdx_refpolicycost,file="REMIND_generic_default.csv",scenario="default")
```

---

convGDX2MIF_LCOE *Read in GDX and write LCOE .mif reporting*

Description

Read in all information from GDX file, calculate the LCOE for pe2se technologies and create the LCOE .mif reporting

Usage

```
convGDX2MIF_LCOE(
  gdx,
  gdx_ref,
  file = NULL,
  scenario = "default",
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
gdx_ref	a GDX as created by readGDX of the reference run
file	name of the mif file which will be written, if no name is provided a magpie object containing all the reporting information is returned
scenario	scenario name that is used in the *.mif reporting
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2030,2050)

Author(s)

Lavinia Baumstark

Examples

```
## Not run:
convGDX2MIF(gdx, gdx_ref, file = "REMIND_generic_LCOE.csv", scenario = "default")

## End(Not run)
```

convGDX2MIF_REMIND2MAgPIE *Read in GDX and write *.mif short reporting for REMIND-MAgPIE coupling*

Description

Read in information from GDX file and create the *.mif reporting using only reporting functions that calculate the variables that are relevant for the REMIND-MAgPIE coupling

Usage

```
convGDX2MIF_REMIND2MAgPIE(
  gdx,
  file = NULL,
  scenario = "default",
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
file	name of the mif file which will be written. If no name is provided a magpie object containing all the reporting information is returned
scenario	scenario name that is used in the *.mif reporting
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

David Klein

Examples

```
## Not run: convGDX2MIF_REMIND2MAgPIE(gdx,file="REMIND_generic_default.csv",scenario="default")
```

createVarListHtml *Create an HTML Document of a Hierarchical List of Variables*

Description

Creates a hierarchical list from mif data using variablesAsList and writes it as an HTML document that displays the hierarchy via the <details> HTML5-tag.

Usage

```
createVarListHtml(
  x,
  outFileName,
  title = "List of Variables",
  htmlBefore = "",
  usePlus = FALSE,
  details = NULL
)
```

Arguments

x	A character vector of variable names, a quitte object, or a character vectors of paths to mif files.
outFileName	A single string. The path of the output file, preferably ending in `.html`
title	The title displayed at the top of the created HTML.
htmlBefore	`character(1)`. HTML to be put between title and list of variables.
usePlus	`logical(1)`. If `FALSE`, removes `|+|`, `|++|`, ... from variable names.
details	`NULL` or a data frame with a column `name`. The entries of further columns of `details` are put into the INFO nodes.

See Also

variablesAsList

Examples

```
## Not run:
loadModeltest()
createVarListHtml(data, "variables.html")
detailsAR6 <-
  readr::read_delim(
    paste0(
      "https://raw.githubusercontent.com/pik-piam/",
      "piamInterfaces/master/inst/mappings/mapping_AR6.csv"),
    delim = ";",
    col_select = c(r21m42, Definition)
  ) %>%
  rename(name = r21m42)
createVarListHtml(
  data,
  "variablesWithDescription.html",
  title = "Reported REMIND Variables with AR6 Description",
  usePlus = TRUE,
  details = detailsAR6)

## End(Not run)
```

dimSums *dimSums*

Description

Slightly modified version of the dimSums function of the magclass package. It will expand magclass objects of length 0 by adding an element in the dimension over which dimSumsSpecial is applied and set all values to 0 and it will name the spatial dimension "GLO" if summation over this dimension happens.

Usage

```
dimSums(x, dim = 3, na.rm = FALSE)
```

Arguments

x	A MAgPIE-object
dim	The dimensions(s) to sum over. A vector of dimension codes or dimension names. See dimCode for more information
na.rm	logical. Should missing values (including NaN) be omitted from the calculations?

Value

A MAgPIE object with values summed over the specified dimensions

Author(s)

Jan Philipp Dietrich

gdx.copy *function for copying REMIND GDX files*

Description

allows to copy a GDX file from place A to place B using the name with the *.gz ending, regardless if it's zipped or not

Usage

```
gdx.copy(from, to)
```

Arguments

from	path to a zipped or unzipped GDX
to	where the unzipped GDX will go and what name it will have

Author(s)

Anastasis Giannousakis

See Also

file.copy

Examples

```
## Not run: gdx.copy(path_gdx, "config/input.gdx", overwrite=TRUE)
```

---

getMifScenPath *Get Paths to Certain Files in the REMIND Directory*

Description

getMifScenPath: get path to the scenarios' reporting mifs. getMifHistPath: get path to the scenarios' historical.mif. getCfgScenPath: get path to the scenarios' config.Rdata. getCfgDefaultPath: get path to REMIND's default.cfg.

Usage

```
getMifScenPath(outputDirs, mustWork = FALSE)

getMifHistPath(outputDirs, mustWork = FALSE)

getCfgScenPath(outputDirs, mustWork = FALSE)

getCfgDefaultPath(remindDir = ".", mustWork = FALSE)
```

Arguments

outputDirs	A character vector of paths to output folders of REMIND runs.
mustWork	logical(1). Throw error if file not available?
remindDir	A single string. The path to the remind directory.

getRunsMIFGDX	*Copy fulldata.gdx and mif files from a suite of runs into one folder. This function creates the folder "./data/" in your working directory if such folder does not exist. It will furthermore create a subfolder with the experiment name where the gdx and mif-files will be copied to. Care: If this subfolder is already existing, old files in this subfolder will be overwritten!*

Description

Copy fulldata.gdx and mif files from a suite of runs into one folder. This function creates the folder "./data/" in your working directory if such folder does not exist. It will furthermore create a subfolder with the experiment name where the gdx and mif-files will be copied to. Care: If this subfolder is already existing, old files in this subfolder will be overwritten!

Usage

```
getRunsMIFGDX(output.folder, experiment)
```

Arguments

output.folder	a vector with the paths to REMIND output folders of the desired runs
experiment	name of this experiment (suite of runs)

Value

no return, function creates data folder and copies output files into it

Author(s)

Felix Schreyer

Examples

```
## Not run: copyMIFGDX(c("Z:/Modeling/REMIND/output/BAU_2019-12-03_14.30.56/",
                        "Z:/Modeling/REMIND/output/NDC_2019-12-03_15.20.53/"),
                      experiment = "NewFeature_03_12_19")
## End(Not run)
```

get_total_efficiencies *Get CES Total Efficiencies*

Description

Computes total efficiency $\alpha = \xi\,(\theta\delta)^{\rho}$ from `pm_cesdata`.

Usage

```
get_total_efficiencies(gdxName)
```

Arguments

`gdxName`	Path to `.gdx` file

Value

A `data.frame` with columns `t`, `regi`, `pf` (production factor) and the `alpha` value.

Author(s)

Michaja Pehl

loadCs2Data *Load compareScenarios Data*

Description

Load data from mif files into R-objects as used in compareScenarios().

Usage

```
loadCs2Data(
  mifScen,
  mifHist,
  cfgScen = NULL,
  cfgDefault = NULL,
  envir = globalenv()
)
```

Arguments

mifScen	character(n), optionally named. Paths to scenario mifs. If the vector has names, those are used to refer to the scenarios in the output file.
mifHist	character(1). Path to historical mif.
cfgScen, cfgDefault	See section "YAML Parameters" in compareScenarios().
envir	environment. The environment where the loaded data is put into.

Examples

```
## Not run:
loadCs2Data(
  c("path/to/Base.mif", "path/to/NDC.mif"),
  "path/to/historical.mif")

## End(Not run)
```

---

loadModeltest *Load Modeltest Results*

Description

The newest model tests are collected from the cluster and copied into a temporary folder (by default). Then the compareScenarios() data loading procedure is used to load this data into the users environment.

Usage

```
loadModeltest(
  envir = globalenv(),
  namePattern = "^SSP2-.*-AMT$",
  folder = tempdir()
)
```

Arguments

envir	environment. The environment where the loaded data is put into.
namePattern	character(1). A regular expression to filter the modeltest run names.
folder	character(1). A folder to copy the modeltest data to.

Examples

```
## Not run:
loadModeltest()

ssp1 <- new.env()
ssp2eu <- new.env()
loadModeltest(ssp1, "^SSP1-AMT-")
loadModeltest(ssp2eu, "^SSP2-.*-AMT$")
ssp1$data
ssp2eu$data

## End(Not run)
```

`nashAnalysis` *Nash Analysis*

Description

Create plots visualizing nash convergence of a given REMIND run

Usage

```
nashAnalysis(
  gdx = "fulldata.gdx",
  outputDir = getwd(),
  outputFile = "Nash Analysis.html"
)
```

Arguments

`gdx`	a GDX object as created by readGDX, or the path to a gdx
`outputDir`	`character(1)`. The directory where the output document and intermediary files are created.
`outputFile`	`character(1)`. File name (without extension) of the output document to be created.

Value

The value returned by `rmarkdown::render()`.

Author(s)

Falk Benke

plotLCOE *Read in LCOE mif and write LCOE_plots.pdf*

Description

Read in all information from LCOE mif file and create the LCOE_plots.pdf

Usage

```
plotLCOE(
  LCOEfile,
  gdx,
  y = c(2015, 2020, 2030, 2040, 2050, 2060),
  reg = "all_regi",
  fileName = "LCOE_plots.pdf"
)
```

Arguments

LCOEfile	a path to the LCOE reporting csv file of the scenario to be plotted
gdx	gdx file of run
y	time span for the data in line plots, default: y=c(2015, 2020,2030,2040,2050,2060)
reg	region(s) in focus, reg ="all_regi" shows all regions if the mifs contain different regions
fileName	name of the pdf, default = "LCOE_plots.pdf"

Author(s)

Felix Schreyer

plotNashConvergence *Creates a REMIND convergence overview*

Description

Creates a REMIND convergence overview

Usage

```
plotNashConvergence(gdx)
```

Arguments

gdx	GDX file

Author(s)

Renato Rodrigues, Falk Benke

Examples

```
  ## Not run:
    plotNashConvergence(gdx="fulldata.gdx")

## End(Not run)
```

`read.reportEntry` *Read entry in file of report format*

Description

This function reads one entry of a reporting file (a file in the model intercomparison file format *.mif) into a MAgPIE object. This function can be used by readAll() to read in the data for more than one output folder

Usage

```
read.reportEntry(outputdir, entry, type = NULL)
```

Arguments

`outputdir`	output folder which contains the reporting
`entry`	entry of the reporting that you want to read in
`type`	type of the reporting that you want to read in

Author(s)

Lavinia Baumstark, David Klein

Examples

```
## Not run:
read.reportEntry("output/SSP2-ref", entry = "Emi|Kyoto Gases (Mt CO2eq/yr)")
  remind2:::readAll(outputdirs, read.reportEntry, entry = "Emi|Kyoto Gases (Mt CO2eq/yr)",
                    asList = FALSE)

## End(Not run)
```

readAll *readAll*

Description

(This function was copied from magpie::read_all to remove the magpie dependency.) Function to read in input from multiple gdx files.

Usage

```
readAll(pathToGdx, func, asList = TRUE, ...)
```

Arguments

pathToGdx	A vector or list. Can contain either the filenames of gdx files or GDX lists as created by readGDX. If it is named, the names will also be used for the output)
func	The output function that should be executed on the gdx files. E.g. emissions
asList	If TRUE, the output will be a list of length(gdx). If FALSE, read_all tries to store everything in one magpie object.
...	Additional arguments passed to func.

Value

A list of magpie objects (as.list=TRUE) or one magpie object (as.list=FALSE) with the output returned by func for all the gdx files

Author(s)

Markus Bonsch

Examples

```
## Not run:
gdxPaths <- c(baseline = "fulldata1.gdx", policy = "fulldata2.gdx")
croparea <- read_all(gdxPaths, func = croparea, level = "glo", crop_aggr = TRUE, asList = TRUE)

## End(Not run)
```

readAllReportingMIFinFolder *Reads in all valid MIF reporting files in a folder*

Description

read in all valid MIF reporting files in a folder and return data as a quitte-object. If no mif exists it searches for CSV

Usage

```
readAllReportingMIFinFolder(dir, RData = FALSE, verbose = TRUE)
```

Arguments

dir	character. Directory containing the MIF files.
RData	logical. If true data is saved in Rdata format to save read-in time.
verbose	logical. If true (un)helpful information will be display.

Value

quitte object

Author(s)

Anselm Schultes, Jerome Hilaire, Lavinia Baumstark, David Klein

Examples

```
  ## Not run:
    qd <- readAllReportingMIFinFolder('/my/magic/remind/run/')

## End(Not run)
```

readConsumption *Read Consumption from GDX file*

Description

Read Consumption data from a GDX file into a magpie object.

Usage

```
readConsumption(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up'

Author(s)

Jonas Hoersch

Examples

```
## Not run: cons <- readConsumption(gdx)
```

---

readCurrentAccount *Read Current Account from GDX file*

Description

Read Current Account data from a GDX file into a magpie object.

Usage

```
readCurrentAccount(gdx)
```

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)

Author(s)

Jonas Hoersch

Examples

```
## Not run: readCurrentAccount(gdx)
```

readEmissions *Read Emissions from GDX file*

Description

Read emission data from a GDX file into a magpie object. summs automatically over all kinds of enty and vm_emiengregi and vm_emiengregi

Usage

```
readEmissions(gdx, emiengregi, eminegregi)
```

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)
emiengregi	enty that is read in from vm_emiengregi, vector is possible e.g. c("co2","n2o"). If you do not want to add an enty from vm_emiengregi use emiengregi=NULL
eminegregi	enty that is read in from vm_eminegregi, vector is possible e.g. c("co2","co2cement"). If you do not want to add an enty from vm_emiengregi use emiengregi=NULL

Author(s)

Lavinia Baumstark

Examples

```
## Not run: emi <- readEmissions(gdx,emiengregi=c("n2o","co2"),eminegregi=NULL)
```

readEnergyInvestments *Read Energy Investments from GDX file*

Description

Read Energy Investments data from a GDX file into a MagPIE object.

Usage

```
readEnergyInvestments(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up')

Value

Energy Investments data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: invests <- readEnergyInvestments(gdx)
```

`readFE` *Read final energy from GDX file*

Description

Read final energy data from a GDX file into a magpie object.

Usage

```
readFE(gdx)
```

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)

Author(s)

Lavinia Baumstark

Examples

```
## Not run: fe <- readFE(gdx)
```

readFuelex *Read Fuelex from GDX file*

Description

Read Fuelex data for the specified enty (energy type) from a GDX file into a magpie object.

Usage

```
readFuelex(gdx, enty = 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)
enty	one of the energy types in REMIND: "pecoal", "pegas", "peoil", "pebiolc", "peur", etc.

Value

Fuelex data as MAgPIE object

Author(s)

David Klein, Lavinia Baumstark

Examples

```
  ## Not run:
    coal <- readFuelex(gdx, enty = "pecoal")
    gas <- readFuelex(gdx, enty = "pegas")

## End(Not run)
```

readFuelSupplyCosts *Read Fuel supply costs from GDX file*

Description

Read Fuel supply costs data from a GDX file into a magpie object.

Usage

```
readFuelSupplyCosts(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up')

Value

Fuel supply costs data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: readFuelSupplyCosts(gdx)
```

---

readGDPMER *Read GDP|MER from GDX file*

Description

Read GDP|MER data from a GDX file into a magpie object.

Usage

```
readGDPMER(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up'

Author(s)

Jonas Hoersch

Examples

```
## Not run: gdp <- readGDPMER(gdx)
```

readInvestmentsNonESM *Read Investments|Non-ESM from GDX file*

Description

Read Investments|Non-ESM data from a GDX file into a magpie object.

Usage

```
readInvestmentsNonESM(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up')

Value

Investments|Non-ESM data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: readInvestmentsNonESM(gdx)
```

readNonEnergyAbatementCosts *Read non-Energy Abatement costs from GDX file*

Description

Read non-Energy Abatement costs data from a GDX file into a magpie object.

Usage

```
readNonEnergyAbatementCosts(gdx)
```

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)

Value

non-Energy Abatement costs data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: readNonEnergyAbatementCosts(gdx)
```

---

readOandMcosts *Read OandM costs from GDX file*

Description

Read OandM costs data from a GDX file into a magpie object.

Usage

```
readOandMcosts(gdx, field = "l")
```

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)
field	one of 'l', 'm', 's', 'lo', 'up')

Value

OandM costs data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: readOandMcosts(gdx)
```

readPopulation *Read Population from GDX file*

Description

Read Population data from a GDX file into a magpie object.

Usage

```
readPopulation(gdx)
```

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)

Author(s)

Lavinia Baumstark

Examples

```
## Not run: gdp <- readPopulation(gdx)
```

readPVP *Read PVP ("Present value price") from GDX file*

Description

Read PVP data of a certain commodity from a GDX file into a magpie object.

Usage

```
readPVP(gdx, level = "glo", enty = "good")
```

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)
level	spartiel resolution "glo" = global or "reg" = regional
enty	one of the traded commodities in REMIND: "good", "pecoal", "pegas", "peoil", "pebiolc", "peur", "perm"

Value

PVP data as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
  ## Not run:
    coal <- readPVP(gdx, enty = "pecoal")
    gas <- readPVP(gdx, enty = "pegas")

## End(Not run)
```

`readReportingMIF` *Reads in a valid MIF (or CSV) reporting file*

Description

read in a valid MIF (or CSV) reporting file and return data as a quitte-object

Usage

```
readReportingMIF(pathToMIF, RData = FALSE, verbose = TRUE)
```

Arguments

`pathToMIF`	character. Path to a valid MIF reporting file.
`RData`	logical. if true data is saved in Rdata format to save read-in time
`verbose`	logical. If true (un)helpful information will be display.

Value

quitte object

Author(s)

Anselm Schultes, Jerome Hilaire, Lavinia Baumstark, David Klein

Examples

```
  ## Not run:
    qd <- readReportingMIF('/my/magic/remind/run/REMIND_generic.mif')
    qd <- readReportingMIF('/my/magic/remind/run/REMIND_generic.csv')

## End(Not run)
```

readSupplycurveBio *Read bioenergy supplycurve from GDX*

Description

Read coefficients for bioenergy prices from gdx and calculate bioenergy supplycurve. Also read the actual demand for bioenergy from gdx.

Usage

```
readSupplycurveBio(
  outputdirs,
  userfun = function(param, x) {
     return(param[[1]] + param[[2]] * x)
 },
  mult_on = "all"
)
```

Arguments

outputdirs	Vector providing the folders that contain the gdx files
userfun	Function that was used to fit the supplycurve. Is needed to calculate the supplycurve correctly. User can provide a functional form using the following syntax: `function(param,x)return(param[[1]] + param[[2]] * x ^param[[3]])`. This function is the default.
mult_on	Should the multiplication factor (read from the gdx) be applied on the entire forumla ("all"), or on the slope only ("slope"). Use "all" for REMIND 2.0 (this is the default) and "slope" for REMIND 1.7

Author(s)

David Klein

See Also

emulator, calc_supplycurve

readTimeStepWeight *Read Time Step Weight from GDX file*

Description

Read Time Step Weight from a GDX file into a magpie object.

Usage

```
readTimeStepWeight(gdx)
```

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)

Value

Time Step Weight as MAgPIE object

Author(s)

Jonas Hoersch

Examples

```
## Not run: readTimeStepWeight(gdx)
```

readTrade *Read Trade from GDX file*

Description

Read Exports or Imports of a commodity from a GDX file into a magpie object.

Usage

```
readTrade(gdx, type, variable)
```

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)
type	one of "Imports" or "Exports"
variable	one of the traded commodities in REMIND: good, pecoal, pegas, peoil, pebiolc, peur, perm

Value

Trade data as MAgPIE object

Author(s)

Niklas Roming

Examples

```
  ## Not run:
    coal <- readTrade(gdx, "Exports", "pecoal") # Coal exports
    gas <- readTrade(gdx, "Imports", "pegas") # Gas exports

## End(Not run)
```

---

reportCapacity *Read in GDX and calculate capacities, used in convGDX2MIF.R for the reporting*

Description

Read in capacity information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportCapacity(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the capacity variables

Author(s)

Lavinia Baumstark, Christoph Bertram

See Also

convGDX2MIF

Examples

```
## Not run:
reportCapacity(gdx)

## End(Not run)
```

`reportCapitalStock` *Read in GDX and calculate capital stocks, used in convGDX2MIF.R for the reporting*

Description

Read in capital stock information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportCapitalStock(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	a GDX object as created by readGDX, or the path to a gdx
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the capital stock variables

Author(s)

Lavinia Baumstark; Michaja Pehl

See Also

convGDX2MIF

Examples

```
## Not run: reportCapitalStock(gdx)
```

`reportClimate` *Read in GDX and extract climate assessment variables*

Description

Read climate assessment variables from GDX file

Usage

```
reportClimate(gdx, output)
```

Arguments

`gdx`	A GDX as created by readGDX, or the file name of a gdx
`output`	Existing magclass object containing data generated by other report*.R functions

Author(s)

Tonn Rüter

Examples

```
## Not run:
output <- reportMacroEconomy(gdx, NULL)
reportClimate(gdx, output)

## End(Not run)
```

`reportCosts` *Read in GDX and calculate costs, used in convGDX2MIF.R for the reporting*

Description

Read in cost information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportCosts(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
output	a magpie object containing all needed variables generated by other report*.R functions
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the cost variables

Author(s)

David Klein

See Also

convGDX2MIF

Examples

```
## Not run: reportCosts(gdx)
```

---

reportCrossVariables *Read in GDX and calculate variables that need variables produced by other report*.R functions, used in convGDX2MIF.R for the reporting*

Description

Read in GDX and calculate variables that need variables produced by other report*.R functions

Usage

```
reportCrossVariables(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
output	a magpie object containing all needed variables generated by other report*.R functions
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Lavinia Baumstark, Falk Benke

Examples

```
## Not run: reportCrossVariables(gdx)
```

---

reportDIETER *Reporting for the coupled DIETER Model*

Description

The DIETER data is appended to a REMIND_generic_.mif file and the new mif is saved in different location, i.e sub directory "DIETER".

Usage

```
reportDIETER(dieterDatafile = "report_DIETER.gdx", outputDir = ".")
```

Arguments

dieterDatafile	full path with name of dieter gdx file.
outputDir	path to the output folder, default is current folder.

Author(s)

Chen Gong, Pratik Agrawal

`reportEmi` *Read in GDX and calculate emissions, used in convGDX2MIF.R for the reporting*

Description

Read in GDX and calculate emissions, used in convGDX2MIF.R for the reporting

Usage

```
reportEmi(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	a GDX as created by readGDX, or the file name of a gdx
`output`	a magpie object containing all needed variables generated by other report*.R functions
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Felix Schreyer

Examples

```
## Not run:
reportEmi(gdx)

## End(Not run)
```

`reportEmiAirPol` *Read in GDX and calculate air pollution emissions, used in convGDX2MIF.R for the reporting*

Description

Read in air pollution emission information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportEmiAirPol(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	a GDX object as created by readGDX, or the path to a gdx
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the emission variables

Author(s)

Antoine Levesque, Jerome Hilaire

See Also

`convGDX2MIF`

Examples

```
## Not run: reportEmiAirPol(gdx)
```

reportEmiForClimateAssessment

Reports emissions & air pollutant values from GDX for climate assessment in between Nash iterations before some of the energy system variables are defined. The report contains only a subset of reported emissions, with the main difference being that Emi|CO2|Energy and Industrial Processes is calculated from the difference between total and LUC emissions. Only global values from this function should be used, as it also skips the subtraction of certain non-regional sources, such as bunkers, from the regional information

Description

Reports emissions & air pollutant values from GDX for climate assessment in between Nash iterations before some of the energy system variables are defined. The report contains only a subset of reported emissions, with the main difference being that Emi|CO2|Energy and Industrial Processes is calculated from the difference between total and LUC emissions. Only global values from this function should be used, as it also skips the subtraction of certain non-regional sources, such as bunkers, from the regional information

Usage

```
reportEmiForClimateAssessment(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
output	a magpie object containing all needed variables generated by other report*.R functions
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Gabriel Abrahao, Tonn Rüter

Examples

```
## Not run:
reportEmiForClimateAssessment(gdx)

## End(Not run)
```

`reportEmployment` *Computes the employment values (jobs) across different sectors*

Description

This function returns a magpie object containing the reporting the jobs for different technologies

Usage

```
reportEmployment(gdx, improvements, multiplier, subtype, shareManf, decline)
```

Arguments

`gdx`	A gdx file output from a REMIND run
`improvements`	Either "None", "CEEW", "Dias", "Rutovitz_aus", "Solar_found" or "All". Use "All" for all improvements.
`multiplier`	controls how the regional multiplier for non-oecd countries changes with time.
`subtype`	Subtype for how shares of solar rooftop, wind offshore, and small hydro are assumed in the future. Options "current", "irena", and "expert". See calcDspvShare for more information.
`shareManf`	Either "current" or "local". Current implies current shares of world manufacture remain same until 2050, current means that in 2050 all countries manufacture required components locally.
`decline`	How should the employment factors change over time? "capcosts" means according to capital costs. "static" means it doesn't change

Value

A magpie object

Author(s)

Aman Malik

Examples

```
## Not run:
reportEmployment(gdx, improvements = "All", multiplier = "own",
subtype = "expert", shareManf = "local", decline = "capcosts")

## End(Not run)
```

`reportEnergyInvestment` *Read in GDX and calculate prices, used in convGDX2MIF.R for the reporting*

Description

Read in price information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportEnergyInvestment(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	a GDX object as created by readGDX, or the path to a gdx
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the price variables

Author(s)

Anastasis Giannousaki

See Also

convGDX2MIF

Examples

```
## Not run:
reportEnergyInvestment(gdx)

## End(Not run)
```

`reportExtraction` *Compute the reporting values of the extraction sector*

Description

This function returns a magpie object containing the reporting values of the extraction sector. Values include quantities extracted, average and supply costs for coal, oil, gas and uranium.

Usage

```
reportExtraction(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	A gdx object
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

A magpie object

Author(s)

Jerome Hilaire, Lavinia Baumstark

Examples

```
## Not run:
reportExtraction(gdx)

## End(Not run)
```

reportFE *Read in GDX and calculate final energy, used in convGDX2MIF.R for the reporting*

Description

Read in final energy information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportFE(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Renato Rodrigues, Felix Schreyer

Examples

```
## Not run: reportFE(gdx)
```

reportLCOE *Read in GDX and calculate LCOE reporting used in convGDX2MIF_LCOE.*

Description

This function provides a post-processing calculation of LCOE (Levelized Cost of Energy) for energy conversion technologies in REMIND. It includes most technologies that generate secondary energy and the distribution technologies which convert secondary energy to final energy. This script calculates two different types of LCOE: average LCOE (standing system) and marginal LCOE (new plant). The average LCOE reflect the total cost incurred by the technology deployment in a specific time step divided by its energy output. The marginal LCOE estimate the per-unit lifetime cost of the output if the model added another capacity of that technology in the respective time step. The marginal LCOE are calculate in two versions: 1) with fuel prices and co2 taxes of the time step for which the LCOE are calculated (time step prices), or 2) with an intertemporal weighted-average of fuel price and co2 taxes over the lifetime of the plant (intertemporal prices).

Usage

```
reportLCOE(gdx, output.type = "both")
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
output.type	string to determine which output shall be produced. Can be either "average" (returns only average LCOE), "marginal" (returns only marginal LCOE), "both" (returns marginal and average LCOE) and and "marginal detail" (returns table to trace back how marginal LCOE are calculated).

Value

MAgPIE object - LCOE calculated by model post-processing. Two types a) standing system LCOE b) new plant LCOE.

Author(s)

Felix Schreyer, Robert Pietzcker, Lavinia Baumstark

See Also

convGDX2MIF_LCOE

Examples

```
## Not run: reportLCOE(gdx)
```

reportMacroEconomy *Read in GDX and calculate macro economy values, used in convGDX2MIF.R for the reporting*

Description

Read in macro economy information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportMacroEconomy(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Lavinia Baumstark, Anselm Schultes

Examples

```
## Not run:
reportMacroEconomy(gdx)

## End(Not run)
```

reportMOFEX *Calculate and report relevant fossil fuel cost and quantity variables from MOFEX standalone model.*

Description

Calculate and report relevant fossil fuel cost and quantity variables from MOFEX standalone model.

Usage

```
reportMOFEX(gdx, gdx_ref = NULL, file = NULL, scenario = "default")
```

Arguments

gdx	the filepath to a gdx
gdx_ref	reference-gdx for policy costs, a GDX as created by readGDX, or the file name of a gdx
file	path to output .mif file
scenario	scenario name that is used in the *.mif reporting

Value

MAgPIE object - contains the output from a MOFEX standalone run

Author(s)

Steve Bi

See Also

convGDX2MIF reportPE reportCosts

Examples

```
## Not run:
reportMOFEX(gdx)

## End(Not run)
```

---

reportPE *Read in GDX and calculate primary energy, used in convGDX2MIF.R for the reporting*

Description

Read in primary energy information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportPE(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Lavinia Baumstark

Examples

```
## Not run: reportPE(gdx)
```

---

reportPolicyCosts *Read in GDX and calculate policy costs, used in convGDX2MIF.R for the reporting*

Description

Read in GDX and calculate policy costs functions

Usage

```
reportPolicyCosts(
  gdx,
  gdx_ref,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
gdx_ref	a reference GDX as created by readGDX, or the file name of a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Lavinia Baumstark

Examples

```
## Not run: reportPolicyCosts(gdx)
```

`reportPrices` *Read in GDX and calculate prices, used in convGDX2MIF.R for the reporting*

Description

Read in price information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportPrices(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150),
  gdx_ref = NULL
)
```

Arguments

`gdx`	a GDX object as created by readGDX, or the path to a gdx
`output`	a magpie object containing all needed variables generated by other report*.R functions
`regionSubsetList`	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
`t`	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)
`gdx_ref`	a GDX object as created by readGDX, or the path to a gdx of the reference run. It is used to guarantee consistency for Moving Avg prices before cm_startyear

Value

MAgPIE object - contains the price variables

Author(s)

Alois Dirnaichner, Felix Schreyer, David Klein, Renato Rodrigues, Falk Benke

See Also

convGDX2MIF

Examples

```
## Not run: reportPrices(gdx)
```

`reportSDPVariables` *Add SDP variables to mif, used in convGDX2MIF.R for the reporting*

Description

Add SDP variables to mif, used in convGDX2MIF.R for the reporting

Usage

```
reportSDPVariables(
  gdx,
  output = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

`gdx`	a GDX as created by readGDX, or the file name of a gdx
`output`	a magpie object containing all needed variables generated by other report*.R functions
`t`	temporal resolution of the reporting

Value

MAgPIE object - contains the SDP

Author(s)

Sebastian Rauner

See Also

`convGDX2MIF`

Examples

```
## Not run: reportSDPVariables(gdx, output, t)
```

reportSE *Read in GDX and calculate secondary energy, used in convGDX2MIF.R for the reporting*

Description

Read in secondary energy information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportSE(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Gunnar Luderer, Lavinia Baumstark, Felix Schreyer, Falk Benke

Examples

```
## Not run:
reportSE(gdx)

## End(Not run)
```

reportTax *Read in GDX and calculate tax, used in convGDX2MIF.R for the reporting*

Description

Read in tax information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportTax(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX as created by readGDX, or the file name of a gdx
output	a magpie object containing all needed variables generated by other report*.R functions
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Author(s)

Renato Rodrigues, Lavinia Baumstark, Christoph Bertram

Examples

```
## Not run: reportTax(gdx)
```

---

reportTechnology *Read in GDX and calculate technology information, used in convGDX2MIF.R for the reporting*

Description

Read in technology information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportTechnology(
  gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
output	a magpie object containing all needed variables generated by other report*.R functions
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the technology variables

Author(s)

Michaja Pehl, Lavinia Baumstark

See Also

convGDX2MIF

Examples

```
## Not run:
reportTechnology(gdx)

## End(Not run)
```

---

reportTrade *Read in GDX and calculate trade, used in convGDX2MIF.R for the reporting*

Description

Read in trade information from GDX file, information used in convGDX2MIF.R for the reporting

Usage

```
reportTrade(
  gdx,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150)
)
```

Arguments

gdx	a GDX object as created by readGDX, or the path to a gdx
regionSubsetList	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)

Value

MAgPIE object - contains the price variables

Author(s)

Lavinia Baumstark, Christoph Bertram, Anselm Schultes

See Also

convGDX2MIF

Examples

```
## Not run: reportTrade(gdx)
```

---

runEmployment *Computes the employment values (jobs) across different sectors*

Description

This function returns a magpie object containing the reporting the jobs for different technologies

Usage

```
runEmployment(pathToMIF, improvements, multiplier, subtype, shareManf, decline)
```

Arguments

pathToMIF	A mif file putput from a REMIND run
improvements	Either "None", "CEEW", "Dias", "Rutovitz_aus","Solar_found" or "All". Use "All" for all improvements.
multiplier	controls how the regional multiplier for non-oecd countries changes with time.
subtype	Subtype for how shares of solar rooftop, wind offshore, and small hydro are assumed in the future. Options "current", "irena", and "expert". See calcDspvShare for more information.

shareManf	Either "current" or "local". Current implies current shares of world manufacture remain same until 2050, current means that in 2050 all countries manufacture required components locally.
decline	How should the employment factors change over time? "capcosts" means according to capital costs. "static" means it doesn't change

Value

A magpie object

Author(s)

Aman Malik

Examples

```
## Not run:

runEmployment(pathToMIF, improvements = "All", multiplier = "own", subtype = "expert",
shareManf = "local", decline = "capcosts")

## End(Not run)
```

---

switchValuesScenConf *searches all scenario config files of REMIND directory for a switch and prints the occurences*

Description

searches all scenario config files of REMIND directory for a switch and prints the occurences

Usage

```
switchValuesScenConf(switchname = NULL, directory = ".")
```

Arguments

switchname	string with switch that is searched in scenario config files. If NULL, prints all switches with only one value in main.gms and scenario config files
directory	path to REMIND directory

Value

values used in default and all scenario configs if switchname = NULL, then list of all unique values

Author(s)

Oliver Richters

test_ranges *Test Ranges on Variables in magpie Objects*

Description

Test Ranges on Variables in magpie Objects

Usage

```
test_ranges(
  data,
  tests,
  reaction = c("warning", "stop"),
  report.missing = FALSE
)
```

Arguments

data	A magpie object to test.
tests	A list of tests to perform, where each tests consists of: - A regular expression to match variables names in data (mandatory first item). - A named entry low to test the lower bound. Can be set to NULL or omitted. - A named entry up to test the upper bound. Can be set to NULL or omitted. - An optional entry ignore.case which can be set to FALSE if the regular expression should be matched case-sensitive.
reaction	A character string, either 'warning' or 'stop', to either warn or throw an error if variables exceed the ranges.
report.missing	If set to TRUE, will message about regular expressions from tests not matching any variables in data.

Author(s)

Michaja Pehl

Examples

```
require(dplyr)
require(tidyr)

(data <- bind_rows(
  expand_grid(variable = 'Foo Share (%)',
              value = c(-1, 0, 42, 100, 101)),
  expand_grid(variable = 'bar share (percent)',
              value = c(-1, 0, 42, 100, 101))) %>%
  group_by(variable) %>%
  mutate(year = 2000 + (1:n())) %>%
  ungroup() %>%
```

```
  select(year, variable, value) %>%
  as.magpie(spatial = 0, temporal = 1, data = 3))

tests <- list(list("Share.*\\((%|Percent)\\)$", low = 0, up = 100))

test_ranges(data, tests)
```

`toolRegionSubsets` *toolRegionSubsets Returns a list of parent regions that are equal to a child mapping union of region mappings.*

Description

toolRegionSubsets Returns a list of parent regions that are equal to a child mapping union of region mappings.

Usage

```
toolRegionSubsets(
  gdx = NULL,
  map = NULL,
  parentMapping = NULL,
  singleMatches = FALSE,
  removeDuplicates = TRUE,
  regionIndex = NULL,
  countryIndex = NULL
)
```

Arguments

`gdx`	a GDX as created by readGDX, or the file name of a gdx. Gdx file containing child mapping regions list.
`map`	alternative to gdx file. You can also provide a child mapping csv file.
`parentMapping`	parent mapping or csv file (if NULL it uses the "regionmappingH12.csv").
`singleMatches`	if true, includes single element matches in the return list. Default: FALSE
`removeDuplicates`	if true, removes from the return list identic matches (a region with the same name and countries for both parent and child mappings). Default: TRUE
`regionIndex`	Name or index of the region column to be used in the parent mapping (if not set the column with less unique elements from the last two columns will be used).
`countryIndex`	Name or index of the country column to be used in parent mapping (if not set the column with more unique elements from the last two columns will be used).

Value

return: returns a list of parent regions that are equal to a child mapping union of region mappings.

Author(s)

Renato Rodrigues

Examples

```
## Not run: toolRegionSubsets(gdx)
```

`variablesAsList` *Variable Names as Hierarchical List*

Description

Take a character vector of variables names with hierarchical structure indicated by `|` in the name and convert it into a hierarchical structure of named lists.

Usage

```
variablesAsList(
  x,
  entry = c("NULL", "name", "INFO"),
  usePlus = FALSE,
  details = NULL
)
```

Arguments

`x`	A character vector of variable names, a quitte object, or a character vectors of paths to mif files.
`entry`	A string determining the entries of all leafs and the nodes which represent variables in `vars`. `"NULL"` puts `NULL` into the leafs of the resulting list. `"name"` places the full name of an existing variable as entry `nm` in the respective node. `"INFO"` puts a list of further information about this node in each node with existing variables, including `details` if specified.
`usePlus`	`logical(1)`. If `FALSE`, removes `|+|`, `|++|`, ... from variable names.
`details`	`NULL` or a data frame with a column `name`. The entries of further columns of `details` are put into the INFO nodes.

Value

A hierarchical named list.

Author(s)

Christof Schoetz

See Also

createVarListHtml for creating an HTML-file of such a list.

Examples

```
vars <- c(
  "Emi|GHG|CO2", "Emi|GHG|CH4", "Emi|NOX",
  "FE", "FE|Buildings", "FE|Industry", "FE|Transport")
v <- variablesAsList(vars)
## Not run: View(v)
v <- variablesAsList(vars, entry = "name")
## Not run: mip::showLinePlots(data, v$Emi$GHG$nm)
details <- data.frame(name = vars, nr = seq_along(vars))
v <- variablesAsList(vars, entry = "INFO", details = details)
## Not run: View(v)

## Not run:
loadModeltest()
v <- variablesAsList(data, entry = "INFO")
View(v)
# Include structure induced by |+|,|++|, ...:
vp <- variablesAsList(data, entry = "INFO", usePlus = TRUE)
View(vp)

## End(Not run)

## Not run: View(variablesAsList("path/to/scenario.mif"))
```

Index