

Package: remind2 (via r-universe)

October 15, 2026

Type Package

Title The REMIND R package (2nd generation)

Version 2.9.0

Date 2026-10-08

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

License LGPL-3

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

Depends magclass (>= 6.17.2)

Imports assertr, dplyr (>= 1.1.1), gdx2 (>= 0.7.0), GDPuc, ggplot2, gms, lucode2 (>= 0.43.0), madrat (>= 3.13.0), magrittr, mip (>= 0.149.2), openxlsx, piamInterfaces (>= 0.33.0), piamPlotComparison (>= 0.0.10), piamutils, plotly (>= 4.10.4), quitte (>= 0.3132.0), readr, rlang, rmarkdown, tibble, tidyr, tidyselect, withr, digest

Suggests covr, gridExtra, htmltools, kableExtra, knitr, piamValidation, remulator, scales, testthat, tidyverse

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/parallel true

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git libglpk-dev make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-10-08 13:13:53 UTC

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

RemoteRef HEAD

RemoteSha e414aecde1eb0ce334c2424367617d711374ea09

Contents

remind2-package	3
calcRegionSubsetSums	4
checkVsCalibData	5
colorScenConf	6
compareCalibrationTargets	6
compareScenarios2	7
compareScenConf	8
convGDX2CSV_LCOE	9
convGDX2MIF	10
convGDX2MIF_REMIND2MAgPIE	11
createVarListHtml	12
dimSums	13
downloadAuxiliaryFile	14
exoGAINS2025AirPollutants	14
getMifScenPath	15
loadCs2Data	16
loadModeltest	16
modifyInvestmentVariables	17
nashConvergenceReport	18
parseLSTReasons	18
plotNashConvergence	19
readAll	20
readSupplycurveBio	21
reportAirPollutantEmissions	22
reportCapacity	23
reportCapitalStock	24
reportClimate	25
reportCosts	25
reportCrossVariables	26
reportEmi	27
reportEmiForClimateAssessment	28
reportEmployment	29
reportExtraction	30
reportExtraEmissions	31
reportFE	31
reportInvestments	32
reportLCOE	33
reportMacroEconomy	34
reportPE	35
reportPolicyCosts	35
reportPrices	36
reportSDPVariables	37
reportSE	38
reportTax	39
reportTechnology	40
reportTrade	41

<i>remind2-package</i>	3
------------------------	---

switchValuesScenConf	42
testRanges	42
timeCumulate	43
toolRegionSubsets	44
variablesAsList	45

Index	47
--------------	-----------

remind2-package	<i>The REMIND R package (2nd generation)</i>
-----------------	--

Description

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

Author(s)

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

Authors:

- Renato Rodrigues <renato.rodrigues@pik-potsdam.de>
- Lavinia Baumstark
- Falk Benke
- David Bantje
- Jan Philipp Dietrich
- Alois Dirnaichner
- Tabea Dorndorf
- Jakob Duerrwaechter
- Pascal Führlich
- Anastasis Giannousakis
- Robin Hasse
- Jérôme Hilaire
- David Klein
- Johannes Koch
- Katarzyna Kowalczyk
- Fabrice Lécuyer
- Antoine Levesque
- Aman Malik
- Anne Merfort
- Leon Merfort
- Simón Morena-Leiva

- Adrian Odenweller
- Michaja Pehl
- Robert Pietzcker
- Sebastian Rauner
- Oliver Richters
- Marianna Rottoli
- Tonn Rüter <tonn.rueter@pik-potsdam.de>
- Robert Salzwedel
- Christof Schötz
- Felix Schreyer
- Kais Siala
- Björn Sörgel
- Mike Spahr
- Jessica Strefler
- Philipp Verpoort
- Pascal Weigmann
- Simon Krogmann

See Also

Useful links:

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

calcRegionSubsetSums *Calculate Sums for Region Subsets*

Description

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

Usage

```
calcRegionSubsetSums(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

```
calcRegionSubsetSums(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	<i>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</i>
---------------	---

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	<i>Render compareCalibrationTargets</i>
---------------------------	---

Description

Render compareCalibrationTargets

Usage

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

Arguments

<code>.gdxPaths</code>	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.
<code>outputDir</code>	The directory where the output document and intermediary files are created.
<code>outputFile</code>	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)
```

<code>compareScenarios2</code>	<i>Render CompareScenarios2</i>
--------------------------------	---------------------------------

Description

A deprecated function. Please refer to `piamPlotComparison::compareScenarios`

Usage

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

Arguments

<code>...</code>	parameters to be passed on to <code>piamPlotComparison::compareScenarios</code>
------------------	---

Value

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

compareScenConf	<i>take two REMIND scenario-config*.csv files and print the difference, comparing it to a default.cfg Can also be used for piamInterfaces mapping files with row.names=NULL and expanddata=FALSE.</i>
-----------------	---

Description

take two REMIND scenario-config*.csv files and print the difference, comparing it to a default.cfg
Can also be used for piamInterfaces mapping files with row.names=NULL and expanddata=FALSE.

Usage

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

Arguments

fileList	vector containing one csv file paths or two paths as <code>c(oldfile, newfile)</code> If one, searches same filename in defaultPath. If NULL, user can select
remindPath	path to REMIND directory containing main.gms. Used for extracting the default configuration and find a file for comparison if only one config file is supplied in fileList
row.names	column in csv used for row.names. Defaults to 1. Use NULL for mapping files which either picks the 'variable' column or uses the first two columns.
renamedCols	vector with old and new column names such as <code>c("old1" = "new1", "old2" = "new2")</code>). Use TRUE as vector element to try to guess automatically: <code>cm_switch <-> c_switch</code>
renamedRows	vector with old and new row names such as <code>c("old3" = "new3", "old3" = "new4", "old5" = "new5")</code>) 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. Use FALSE for mapping files

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

<code>.gdx</code>	a GDX as created by readGDX, or the file name of a.gdx
<code>file</code>	name of the csv file which will be written
<code>scen</code>	scenario name that is used in the *.mif reporting
<code>t</code>	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")

## End(Not run)
```

convGDX2MIF	<i>Read in GDX and write *.mif reporting</i>
-------------	--

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,
  .extraData = NULL
)
```

Arguments

<code>gdx</code>	a GDX as created by <code>gdx2::readGDX</code> , or the file name of a gdx
<code>gdx_ref</code>	reference-gdx for < cm_startyear, used for fixing the prices to this scenario
<code>file</code>	name of the mif file which will be written, if no name is provided a magpie object containing all the reporting information is returned
<code>scenario</code>	scenario name that is used in the *.mif reporting
<code>t</code>	temporal resolution of the reporting, default: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>
<code>gdx_refpolicycost</code>	reference-gdx for policy costs, a GDX as created by <code>gdx2::readGDX</code> , or the file name of a gdx
<code>testthat</code>	boolean whether called by tests, turns some messages into warnings
<code>extraData</code>	path to extra data files to be used in the reporting

Author(s)

Lavinia Baumstark

Examples

```
## Not run:
convGDX2MIF(gdx, gdx_refpolicycost, file = "REMIND_generic_default.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),
  extraData = NULL
)
```

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)
extraData	path to extra data files to be used in the reporting

Author(s)

David Klein

Examples

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

createVarListHtml	<i>Create an HTML Document of a Hierarchical List of Variables</i>
-------------------	--

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

<code>x</code>	A MAgPIE-object
<code>dim</code>	The dimensions(s) to sum over. A vector of dimension codes or dimension names. See dimCode for more information
<code>na.rm</code>	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

`downloadAuxiliaryFile` *Helper to download auxiliary file needed for reporting from RSE server*

Description

Helper to download auxiliary file needed for reporting from RSE server

Usage

```
downloadAuxiliaryFile(fname)
```

Arguments

`fname` name of the file to be downloaded from RSE server folder with extra data

Author(s)

Falk Benke

`exoGAINS2025AirPollutants`
Calculate projected air pollutant emissions at the level of GAINS sectors based on REMIND activities, GAINS emission factors, and baseyear (2020) emissions

Description

Calculate projected air pollutant emissions at the level of GAINS sectors based on REMIND activities, GAINS emission factors, and baseyear (2020) emissions

Usage

```
exoGAINS2025AirPollutants(  
  remind_output,  
  gains_emifacs,  
  baseyear_emis_2020,  
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150),  
  logging = FALSE  
)
```

Arguments

remind_output	a magpie object containing all needed REMIND activities
gains_emifacs	a magpie object containing historic and projected GAINS emission factors for all 35 GAINS sectors
baseyear_emis_2020	a magpie object containing baseyear (2020) emissions for all 35 GAINS sectors
t	temporal resolution of the reporting, default: t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)
logging	Boolean to activate logging

Author(s)

Laurin Köhler-Schindler

Examples

```
## Not run:
exoGAINS2025AirPollutants(remind_output, gains_emifacs, baseyear_emis_2020)

## End(Not run)
```

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.

loadCs2Data	<i>Load compareScenarios Data</i>
-------------	-----------------------------------

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 or NULL.
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	<i>Load Modeltest Results</i>
---------------	-------------------------------

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)
```

modifyInvestmentVariables

Modify Investment Variables

Description

This function transforms investment variables to the normal reporting convention.

Usage

```
modifyInvestmentVariables(x, ref = NULL, startYear = NULL)
```

Arguments

x	a magclass object to be manipulated
ref	an optional magclass object to be used for fixing values before 'startYear'
startYear	years before will be overwritten with values from 'ref'

Details

Years represented by investment variables in the energy system ('vm_deltaCap', 'vm_costInvTeDir' and 'vm_costInvTeAdj') are different from the normal reporting convention. In the current REMIND version, vm_deltacap(t) represents the average of the years t-4..t, while in the general reporting convention it represents the average of t-2.5..t+2.5 (for 5 year time steps).

See also: <https://github.com/remindmodel/remind/pull/1238>.

Author(s)

Falk Benke

nashConvergenceReport *Nash Convergence Report*

Description

Create plots visualizing nash convergence of a given REMIND run

Usage

```
nashConvergenceReport(gdx = "fulldata.gdx", outputDir = getwd())
```

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.

Author(s)

Falk Benke and Renato Rodrigues

parseLSTReasons *Parse per-iteration non-convergence reasons from a GAMS listing file*

Description

Extract, for each Nash iteration, the block of diagnostic text that REMIND writes to the GAMS listing file (full.lst) between the lines "Reasons for non-convergence in this iteration" and "See the indicators below to dig deeper ...". Each block is associated with the iteration number printed just before it (PARAMETER o_iterationNumber).

The listing file can be several hundred MB. When the command line tools grep and sed are available, they are used to scan the file and extract only the relevant lines, which is markedly faster than reading the whole file into R. Otherwise the function falls back to a pure-R implementation. Both paths yield identical results, so the function is portable to systems without these tools (e.g. Windows).

Usage

```
parseLSTReasons(lstFile, timeLimit = 90)
```

Arguments

lstFile	character(1). Path to the GAMS listing file.
timeLimit	numeric(1). Maximum time in seconds allowed for parsing. If exceeded, the function stops and fails gracefully by returning NULL. Default 90.

Value

A data.frame with integer column iteration and character column text, one row per iteration ordered by iteration, or NULL if lstFile does not exist, the time limit is exceeded, or parsing otherwise fails.

Author(s)

Gabriel Abrahão

plotNashConvergence	<i>Creates a REMIND convergence overview</i>
---------------------	--

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)
```

readAll	<i>readAll</i>
---------	----------------

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

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

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)
```

readSupplycurveBio	<i>Read bioenergy supplycurve from GDX</i>
--------------------	--

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 supply-curve correctly. User can provide a functional form using the following syntax: <code>function(param,x)return(param[[1]] + param[[2]] * x ^param[[3]])</code> . This function is the default.
mult_on	Should the multiplication factor (read from the.gdx) be applied on the entire formula ("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](#)

reportAirPollutantEmissions

Read in GDX, GAINS2025 emission factors and baseyear emissions and calculate air pollution emissions, used in convGDX2MIF.R for the reporting

Description

Read in GDX, GAINS2025 emission factors and baseyear emissions and calculate air pollution emissions, used in convGDX2MIF.R for the reporting

Usage

```
reportAirPollutantEmissions(
 .gdx,
  output = NULL,
  regionSubsetList = NULL,
  t = c(seq(2005, 2060, 5), seq(2070, 2110, 10), 2130, 2150),
  extraData = NULL,
  addTmpTotal = FALSE
)
```

Arguments

gdx	a GDX as created by <code>gdx2::readGDX</code> , 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: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>
extraData	path to extra data files to be used in the reporting
addTmpTotal	choose if temporary totals are computed (default = FALSE)

Author(s)

Laurin Köhler-Schindler

Examples

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

## End(Not run)
```

reportCapacity	<i>Read in GDX and calculate capacities, used in convGDX2MIF.R for the reporting</i>
----------------	--

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),
  gdx_ref = gdx
)
```

Arguments

gdx	a GDX object as created by gdx2::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)
gdx_ref	a GDX object as created by gdx2::readGDX, or the path to a gdx of the reference run. It is used to guarantee consistency before 'cm_startyear' for capacity variables using time averaging.

Value

MAGPIE object - contains the capacity variables

Author(s)

Lavinia Baumstark, Christoph Bertram, Fabrice Lecuyer

See Also

[convGDX2MIF](#)

Examples

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

## End(Not run)
```

reportCapitalStock	<i>Read in GDX and calculate capital stocks, used in convGDX2MIF.R for the reporting</i>
--------------------	--

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),
  gdx_ref = NULL
)
```

Arguments

gdx	a GDX object as created by gdx2::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)
gdx_ref	a GDX object as created by gdx2::readGDX, or the path to a gdx of the reference run. It is used to guarantee consistency before 'cm_startyear' for capacity variables using time averaging.

Value

MAGPIE object - contains the capital stock variables

Author(s)

Lavinia Baumstark; Michaja Pehl

See Also

[convGDX2MIF](#)

Examples

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

## End(Not run)
```

reportClimate	<i>Read in GDX and extract climate assessment variables</i>
---------------	---

Description

Read climate assessment variables from GDX file

Usage

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

Arguments

gdx	A GDX as created by <code>gdx2::readGDX</code> , or the file name of a gdx
output	Existing magclass object containing data generated by other <code>report*.R</code> functions

Author(s)

Tonn Rüter

Examples

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

## End(Not run)
```

reportCosts	<i>Read in GDX and calculate costs, used in convGDX2MIF.R for the reporting</i>
-------------	---

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

<code>gdx</code>	a GDX object as created by <code>gdx2::readGDX</code> , or the path to a <code>gdx</code>
<code>output</code>	a magpie object containing all needed variables generated by other <code>report*.R</code> functions
<code>regionSubsetList</code>	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
<code>t</code>	temporal resolution of the reporting, default: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>

Value

MAGPIE object - contains the cost variables

Author(s)

David Klein

See Also

[convGDX2MIF](#)

Examples

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

## End(Not run)
```

<code>reportCrossVariables</code>	<i>Read in GDX and calculate variables that need variables produced by other <code>report*.R</code> functions, used in <code>convGDX2MIF.R</code> for the reporting</i>
-----------------------------------	---

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),
  extraData = NULL
)
```

Arguments

gdx	a GDX as created by gdx2::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)
extraData	path to extra data files to be used in the reporting

Author(s)

Lavinia Baumstark, Falk Benke

Examples

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

## End(Not run)
```

reportEmi	<i>Read in GDX and calculate emissions, used in convGDX2MIF.R for the reporting</i>
-----------	---

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),
  extraData = NULL
)
```

Arguments

gdx	a GDX as created by gdx2::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)
extraData	path to extra data files to be used in the reporting

Author(s)

Felix Schreyer, Anne Merfort

Examples

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

## End(Not run)
```

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\CA\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\CA\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),
  extraData = NULL
)
```

Arguments

gdx	a GDX as created by <code>gdx2::readGDX</code> , 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: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>
extraData	path to extra data files to be used in the reporting

Author(s)

Gabriel Abrahao, Tonn Rüter

Examples

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

## End(Not run)
```

reportEmployment	<i>Computes the employment values (jobs) across different sectors</i>
------------------	---

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 <code>calcD-spvShare</code> 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)
```

reportExtraction	<i>Compute the reporting values of the extraction sector</i>
------------------	--

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)
```

reportExtraEmissions	<i>Report Extra Emissions</i>
----------------------	-------------------------------

Description

Calculate additional emissions based on CEDS 2020 emissions and REMIND/EDGE-Transport Data

Usage

```
reportExtraEmissions(mif, extraData, gdx)
```

Arguments

mif	a mif file with reported variables from remind2 and EDGE-Transport
extraData	path to extra data files to be used in the reporting
gdx	a GDX as created by readGDX, or the file name of a gdx, needed to define region subsets

Author(s)

Gabriel Abrahão, Falk Benke

reportFE	<i>Read in GDX and calculate final energy, used in convGDX2MIF.R for the reporting</i>
----------	--

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 `gdx2::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)
```

reportInvestments	<i>Report investments</i>
-------------------	---------------------------

Description

Report investments

Usage

```
reportInvestments(
  gdx,
  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

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 before 'cm_startyear' for investment variables using time averaging.

Value

magclass object

See Also[convGDX2MIF](#)**Examples**

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

## End(Not run)
```

reportLCOE	<i>Read in GDX and calculate LCOE reporting used in convGDX2MIF_LCOE.</i>
------------	---

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 <code>gdx2::readGDX</code> , 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 "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

Examples

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

## End(Not run)
```

reportMacroEconomy	<i>Read in GDX and calculate macro economy values, used in convGDX2MIF.R for the reporting</i>
--------------------	--

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)
```

reportPE	<i>Read in GDX and calculate primary energy, used in convGDX2MIF.R for the reporting</i>
----------	--

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 gdx2::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, Fabrice Lécuyer

Examples

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

## End(Not run)
```

reportPolicyCosts	<i>Read in GDX and calculate policy costs, used in convGDX2MIF.R for the reporting</i>
-------------------	--

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

<code>gdx</code>	a GDX as created by <code>gdx2::readGDX</code> , or the file name of a <code>gdx</code>
<code>gdx_ref</code>	a reference GDX as created by <code>gdx2::readGDX</code> , or the file name of a <code>gdx</code>
<code>regionSubsetList</code>	a list containing regions to create report variables region aggregations. If NULL (default value) only the global region aggregation "GLO" will be created.
<code>t</code>	temporal resolution of the reporting, default: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>

Author(s)

Lavinia Baumstark

Examples

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

## End(Not run)
```

<code>reportPrices</code>	<i>Read in GDX and calculate prices, used in convGDX2MIF.R for the reporting</i>
---------------------------	--

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,
 .extraData = NULL
)
```

Arguments

gdx	a GDX object as created by <code>gdx2::readGDX</code> , 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: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>
gdx_ref	a GDX object as created by <code>gdx2::readGDX</code> , or the path to a gdx of the reference run. It is used to guarantee consistency for Moving Avg prices before <code>cm_startyear</code>
extraData	path to extra data files to be used in the reporting (might be needed when <code>reportEmi</code> is executed from within this function)

Value

MAGPIE object - contains the price variables

Author(s)

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

See Also

[convGDX2MIF](#)

Examples

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

## End(Not run)
```

reportSDPVariables	<i>Add SDP variables to mif, used in convGDX2MIF.R for the reporting</i>
--------------------	--

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 `gdx2::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)

## End(Not run)
```

reportSE	<i>Read in GDX and calculate secondary energy, used in convGDX2MIF.R for the reporting</i>
----------	--

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 `gdx2::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	<i>Read in GDX and calculate tax, used in convGDX2MIF.R for the reporting</i>
-----------	---

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 `gdx2::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)

## End(Not run)
```

reportTechnology	<i>Read in GDX and calculate technology information, used in convGDX2MIF.R for the reporting</i>
------------------	--

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	<i>Read in GDX and calculate trade, used in convGDX2MIF.R for the reporting</i>
-------------	---

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 <code>gdx2::readGDX</code> , 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: <code>t=c(seq(2005,2060,5),seq(2070,2110,10),2130,2150)</code>

Value

MAGPIE object - contains the price variables

Author(s)

Lavinia Baumstark, Christoph Bertram, Anselm Schultes

See Also[convGDX2MIF](#)

Examples

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

## End(Not run)
```

switchValuesScenConf	<i>searches all scenario config files of REMIND directory for a switch and prints the occurrences</i>
----------------------	---

Description

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

Usage

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

Arguments

switchname	string with switch that is searched in scenario config files. If NULL or "", 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

testRanges	<i>Test Ranges on Variables in magpie Objects</i>
------------	---

Description

Test Ranges on Variables in magpie Objects

Usage

```
testRanges(
  data,
  tests,
  reaction = c("message", "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 'message' 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, datacol = 3))

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

testRanges(data, tests)
```

timeCumulate

timeCumulate

Description

calculate cumulated values over time for a magpie object

Usage

```
timeCumulate(x, includeEndYear = FALSE)
```

Arguments

x a mapgie object

includeEndYear boolean indicating logic for summing up when set to TRUE, the value for 2010 is the sum of the values from 2005 - 2010, otherwise the value for 2010 is the sum of 2005 - 2009

Author(s)

Falk Benke

toolRegionSubsets	<i>toolRegionSubsets Returns a list of parent regions that are equal to a child mapping union of region mappings.</i>
-------------------	---

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 gdx2::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)

## End(Not run)
```

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

calc_supplycurve, [21](#)
calcRegionSubsetSums, [4](#)
checkVsCalibData, [5](#)
colorScenConf, [6](#)
compareCalibrationTargets, [6](#)
compareScenarios, [16](#)
compareScenarios2, [7](#)
compareScenConf, [8](#)
convGDX2CSV_LCOE, [9](#)
convGDX2MIF, [10](#), [23](#), [24](#), [26](#), [33](#), [37](#), [38](#), [41](#)
convGDX2MIF_REMIND2MagPIE, [11](#)
createVarListHtml, [12](#), [46](#)

dimCode, [13](#)
dimSums, [13](#)
downloadAuxiliaryFile, [14](#)

emulator, [21](#)
exoGAINS2025AirPollutants, [14](#)

getCfgDefaultPath (getMifScenPath), [15](#)
getCfgScenPath (getMifScenPath), [15](#)
getMifHistPath (getMifScenPath), [15](#)
getMifScenPath, [15](#)

loadCs2Data, [16](#)
loadModeltest, [16](#)

MagPIE, [4](#)
magpie, [43](#)
modifyInvestmentVariables, [17](#)

nashConvergenceReport, [18](#)

parseLSTReasons, [18](#)
plotNashConvergence, [19](#)

readAll, [20](#)
readSupplycurveBio, [21](#)
remind2 (remind2-package), [3](#)
remind2-package, [3](#)

reportAirPollutantEmissions, [22](#)
reportCapacity, [23](#)
reportCapitalStock, [24](#)
reportClimate, [25](#)
reportCosts, [25](#)
reportCrossVariables, [26](#)
reportEmi, [27](#)
reportEmiForClimateAssessment, [28](#)
reportEmployment, [29](#)
reportExtraction, [30](#)
reportExtraEmissions, [31](#)
reportFE, [31](#)
reportInvestments, [32](#)
reportLCOE, [33](#)
reportMacroEconomy, [34](#)
reportPE, [35](#)
reportPolicyCosts, [35](#)
reportPrices, [36](#)
reportSDPVariables, [37](#)
reportSE, [38](#)
reportTax, [39](#)
reportTechnology, [40](#)
reportTrade, [41](#)
rmarkdown::render(), [5](#), [7](#), [8](#)

switchValuesScenConf, [42](#)

testRanges, [42](#)
timeCumulate, [43](#)
toolRegionSubsets, [44](#)

variablesAsList, [12](#), [45](#)