

# Package: `piamInterfaces` (via `r-universe`)

February 13, 2025

**Type** Package

**Title** Project specific interfaces to REMIND / MAgPIE

**Version** 0.44.4

**Date** 2025-02-13

**Description** Project specific interfaces to REMIND / MAgPIE.

**License** LGPL-3

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

**Imports** dplyr (>= 1.1.1), GDPuc, gms, jsonlite, magclass (>= 6.17.2),  
mip (>= 0.150.0), readxl, quitte (>= 0.3137.1), piamutils (>= 0.0.12), rlang, stringr, tibble, tidyr, tidyselect, yaml

**Suggests** covr, testthat (>= 3.2.3), withr, writexl

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Config/testthat/parallel** true

**Config/testthat/edition** 3

**Config/testthat/start-first** plotIntercomparison,  
generateIIASASubmission, checkSummations

**Config/pak/sysreqs** make libicu-dev libssl-dev libx11-dev zlib1g-dev

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

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

**RemoteRef** HEAD

**RemoteSha** 05a396d77a7b304a8772da75135083fc7e2bccc5

## Contents

<code>piamInterfaces-package</code> . . . . .	2
<code>areUnitsIdentical</code> . . . . .	3
<code>checkFixUnits</code> . . . . .	4
<code>checkIIASASubmission</code> . . . . .	4

checkMissingVars . . . . .	5
checkNGFS . . . . .	6
checkSummations . . . . .	7
checkSummationsRegional . . . . .	8
checkUnitFactor . . . . .	9
checkVarNames . . . . .	10
convertHistoricalData . . . . .	10
extractReferenceYear . . . . .	11
fillMissingSummations . . . . .	12
fillSummationPairs . . . . .	12
fixOnRef . . . . .	13
generateIIASASubmission . . . . .	14
getMapping . . . . .	16
getMappingVariables . . . . .	17
getREMINDTemplateVariables . . . . .	17
getSummations . . . . .	18
getTemplate . . . . .	18
loadIIASATemplate . . . . .	19
mappingNames . . . . .	19
plotIntercomparison . . . . .	20
priceIndicesAdd . . . . .	21
priceIndicesFix . . . . .	22
priceIndicesIIASA . . . . .	22
readMifs . . . . .	23
removePlus . . . . .	23
renameOldInMappings . . . . .	24
renameOldVariables . . . . .	24
setLogFile . . . . .	25
summationsNames . . . . .	26
sumNamesWithFactors . . . . .	26
templateNames . . . . .	27
variableInfo . . . . .	27
writeMapping . . . . .	28

<b>Index</b>	<b>29</b>
--------------	-----------

---

piamInterfaces-package

*piamInterfaces: Project specific interfaces to REMIND / MAgPIE*

---

## Description

Project specific interfaces to REMIND / MAgPIE.

### Author(s)

**Maintainer:** Falk Benke <benke@pik-potsdam.de>

Authors:

- Oliver Richters

### See Also

Useful links:

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

---

areUnitsIdentical      *Check whether units are identical following a specified list*

---

### Description

Check whether units are identical following a specified list

### Usage

```
areUnitsIdentical(vec1, vec2 = NULL)
```

### Arguments

vec1                  units to be checked against vec2, elementwise

vec2                  units to be checked against vec1, elementwise

### Value

boolean

### Author(s)

Oliver Richters

---

checkFixUnits	<i>Check units in IIASA submission by comparing mifdata to a project template</i>
---------------	---

---

**Description**

Check units in IIASA submission by comparing mifdata to a project template

**Usage**

```
checkFixUnits(mifdata, template, logFile = NULL, failOnUnitMismatch = TRUE)
```

**Arguments**

mifdata	quite object or filename of mif file
template	object provided by loadIIASAtemplate() or getMapping() interprets it as a mapping if 'piam_variable' and 'piam_unit' columns exist
logFile	filename of file for logging
failOnUnitMismatch	boolean whether to fail in case of unit mismatches recommended for submission, not used for generating mappings

**Value**

quite object with adapted mif data

**Author(s)**

Oliver Richters

---

checkIIASASubmission	<i>Check IIASA submission by comparing mif data to a template file (xlsx or yaml) provided by IIASA</i>
----------------------	---

---

**Description**

Check IIASA submission by comparing mif data to a template file (xlsx or yaml) provided by IIASA

**Usage**

```
checkIIASASubmission(
  mifdata,
  iiasatemplate,
  logFile = NULL,
  failOnUnitMismatch = TRUE
)
```

**Arguments**

mifdata            quitte object or filename of mif file  
 iiasatemplate    filename of xlsx or yaml file provided by IIASA  
 logFile           filename of file for logging. Set to NULL for stdout, set to FALSE for none.  
 failOnUnitMismatch            boolean whether to fail in case of unit mismatches recommended for submission

**Value**

quitte object with adapted mif data

**Author(s)**

Oliver Richters

**Examples**

```
## Not run:
# Simple use. Generates submission file in output folder:
checkIIASASubmission(
  mifdata = "file.mif",
  iiasatemplate = "template.xlsx",
  logFile = "logFile.txt"
)

## End(Not run)
```

---

checkMissingVars	<i>Check whether all variables required by mapping are present in mif-data for given source</i>
------------------	---

---

**Description**

Check whether all variables required by mapping are present in mifdata for given source

**Usage**

```
checkMissingVars(mifdata, mapping = TRUE, sources = TRUE)
```

**Arguments**

mifdata            data that can be read with as.quitte  
 mapping           name of the project of requested mappings such as c('AR6', 'AR6\_NGFS') or 'mapping.csv'. Use TRUE for all mappings.  
 sources            model abbreviation(s) used in 'source' column. R = REMIND, M = MAgPIE, T = EDGE-T, B = Brick, C = Climate/MAGICC, TRUE = all

**Value**

an invisible vector with missing variables

**Author(s)**

Oliver Richters

---

checkNGFS	<i>Check NGFS submission by comparing mif data to a template file (xlsx or yaml) provided by IIASA</i>
-----------	--

---

**Description**

Check NGFS submission by comparing mif data to a template file (xlsx or yaml) provided by IIASA

**Usage**

```
checkNGFS(mifdata, iiasatemplate, logFile, generatePlots = TRUE)
```

**Arguments**

mifdata	quite object or filename of mif file
iiasatemplate	filename of xlsx or yaml file provided by IIASA
logFile	filename of file for logging. Set to NULL for stdout, set to FALSE for none.
generatePlots	boolean whether to plot failing summations

**Value**

quite object with adapted mif data

**Author(s)**

Oliver Richters

---

checkSummations	<i>Checks for a run if the variables sum up as expected and logs spotted gaps</i>
-----------------	---

---

## Description

Checks for a run if the variables sum up as expected and logs spotted gaps

## Usage

```
checkSummations(
  mifFile,
  outputDirectory = ".",
  template = NULL,
  summationsFile = NULL,
  logFile = NULL,
  logAppend = FALSE,
  generatePlots = FALSE,
  mainReg = "World",
  dataDumpFile = "checkSummations.csv",
  plotprefix = NULL,
  absDiff = 0.001,
  relDiff = 1,
  roundDiff = TRUE,
  csvSeparator = ";"
)
```

## Arguments

mifFile	path to the mif file to apply summation checks to, or quitte object
outputDirectory	path to directory to place logFile and dataDumpFile.
template	mapping to be loaded, used to print the piam_variable corresponding to the data variables
summationsFile	in inst/summations folder that describes the required summation groups if set to 'extractVariableGroups', tries to extract summations from variables with + notation
logFile	file where human-readable summary is saved. If NULL, write to stdout. If FALSE, don't log.
logAppend	boolean whether to append or overwrite logFile
generatePlots	boolean whether pdfs to compare data are generated. Requires outputDirectory.
mainReg	main region for the plot generation
dataDumpFile	file where data.frame with the data analysis is saved. Requires outputDirectory. If NULL, result is returned.

plotprefix	added before filename
absDiff	threshold for absolute difference between parent variable and summation
relDiff	threshold (in percent) for relative difference between parent variable and summation
roundDiff	should the absolute and relative differences in human-readable summary and dataDumpFile be rounded? The returned object always contains unrounded values.
csvSeparator	separator for dataDumpFile, defaults to semicolon

**Author(s)**

Falk Benke, Oliver Richters

---

checkSummationsRegional

*Checks for a run if the regions for selected variables sum up as expected*

---

**Description**

Checks for a run if the regions for selected variables sum up as expected

**Usage**

```
checkSummationsRegional(
  mifFile,
  parentRegion = NULL,
  childRegions = NULL,
  variables = NULL,
  skipUnits = NULL,
  skipBunkers = NULL,
  intensiveUnits = TRUE,
  absDiff = 1e-04,
  relDiff = 0.1
)
```

**Arguments**

mifFile	path to the mif file to apply summation checks to, or quitte object
parentRegion	region to sum up to. Defaults to World or GLO
childRegions	regions that should sum up to parentRegion. Default to all except parentRegion
variables	list of variables to check. Defaults to all in mifFile
skipUnits	units to be skipped. Set to TRUE to get list of units pointing towards their variable being intensive. You can also use c(TRUE, "additionalunit")
skipBunkers	set to TRUE to skip AR6 variables that contain bunkers only at the global level



intensiveUnits intensive units where the global value should not be the sum, but instead lie between the regional values. Set to TRUE to get list of units pointing towards their variable being intensive. You can also use c(TRUE, "additionalunit").

absDiff threshold for absolute difference between parent variable and summation

relDiff threshold (in percent) for relative difference between parent variable and summation

### Author(s)

Falk Benke

### Examples

```
## Not run:
checkSummationsRegional(
  mifFile = "path/to/file",
  childRegions = c("R5ASIA", "R5LAM", "R5MAF", "R5OEC90+EU", "R5REF"),
  parentRegion = "World",
  variables = c("Final Energy|Industry", "Emissions|CO2|Energy|Demand|Industry")
)

## End(Not run)
```

---

checkUnitFactor	<i>Check unit factor in template</i>
-----------------	--------------------------------------

---

### Description

This function checks whether the `piam_factor` in a mapping fits unit and `piam_unit`. It does the following:

1. check whether the units are identical based on `areUnitsIdentical()` and `piam_factor` is 1 or -1.
2. based on `scaleConversion` defined below, check whether manually added factors are satisfied. This works based on regex matching, so for example `1000 TW = 1 PW` is matched by the `c("1000", "T", "P")` line. If the tests fail because of a new unit, you can add them below. If the units are really identical except for spelling, better add them to `areUnitsIdentical.R`

### Usage

```
checkUnitFactor(template, logFile = NULL, failOnUnitMismatch = TRUE)
```

### Arguments

template object provided by `loadIIASAtemplate()`

logFile filename of file for logging

failOnUnitMismatch boolean whether to fail in case of unit mismatches recommended for submission, not used for checking mapping

**Value**

quite object with adapted mif data

**Author(s)**

Oliver Richters

---

checkVarNames	<i>checkVariablesNames checks reporting and mappings on inconsistency in variable names</i>
---------------	---

---

**Description**

Pass a vector of variable names (including the units if withunits=TRUE) or a quite object. Get warnings if inconsistencies are found for the reporting

**Usage**

```
checkVarNames(vars, withunits = TRUE)
```

**Arguments**

vars	vector with variable names (and units such as "PE (EJ)")
withunits	should the var vector contain units in paranthesis?

**Value**

vector with all variables that show issues

**Author(s)**

Oliver Richters

---

convertHistoricalData	<i>Converts data in historical.mif to match project-specific variables and regions so that it can be used for comparison in an intermodel comparison project</i>
-----------------------	--

---

**Description**

Converts data in historical.mif to match project-specific variables and regions so that it can be used for comparison in an intermodel comparison project

**Usage**

```
convertHistoricalData(mif, project, regionMapping = NULL, mainReg = "World")
```

**Arguments**

mif	quitted object with historical data or path to historical.mif
project	name of the project, determines the mapping to be loaded
regionMapping	(optional) csv file with mapping of REMIND regions or ISO to project regions, must contain two columns: One can be called 'REMIND' or 'CountryCode' and contains the regions in the data. The second can be called 'project_region' or 'RegionCode', to which the first is mapped. You can also specify the filename part before the .csv from inst/regionmapping
mainReg	if regionMapping is specified, additional main region that is kept as is

**Author(s)**

Falk Benke

**Examples**

```
## Not run:
data <- convertHistoricalData(
  mif = "path/to/historical.mif",
  project = "NAVIGATE",
  regionMapping = "path/to/region_mapping_NAVIGATE.csv"
)

## End(Not run)
```

---

extractReferenceYear *Extract reference year for price indices from unit*

---

**Description**

Extract reference year for price indices from unit

**Usage**

```
extractReferenceYear(unit, var = NULL)
```

**Arguments**

unit	vector or string of units such as 'Index (2020 = 1)'
var	optional string of variable name to facilitate debugging

**Value**

vector or string of reference years such as '2020'

**Author(s)**

Oliver Richters

---

fillMissingSummations *Recursively calculate additional variables based on given summations and add them to the given mif file*

---

**Description**

Recursively calculate additional variables based on given summations and add them to the given mif file

**Usage**

```
fillMissingSummations(mifFile, summationsFile, iteration = 1, logFile = NULL)
```

**Arguments**

mifFile	path to mif file or a quitte object
summationsFile	in inst/summations folder that describes the required summation groups, or path to summations file
iteration	keeps track of number of recursive calls, leave to default
logFile	path to logFile. if NULL, write to stdout, if FALSE don't write

**Author(s)**

Falk Benke, Renato Rodrigues

---

fillSummationPairs *add missing variable values if the value can be obtained from two other reported results.*

---

**Description**

add missing variable values if the value can be obtained from two other reported results.

**Usage**

```
fillSummationPairs(mifFile, summationsFile)
```

**Arguments**

mifFile	path to mif file or a quitte object
summationsFile	in inst/summations folder that describes the required summation groups, or path to summations file

**Author(s)**

Renato Rodrigues

---

fixOnRef	<i>Checks for a run if it is correctly fixed on the reference run for <math>t &lt; \text{startyear}</math></i>
----------	--

---

### Description

Checks for a run if it is correctly fixed on the reference run for  $t < \text{startyear}$

### Usage

```
fixOnRef(  
  data,  
  refscen,  
  startyear,  
  ret = "boolean",  
  failfile = NULL,  
  relDiff = 1e-12  
)
```

### Arguments

data	quite object or mif file
refscen	scenario name of reference scenario, or file or quite object with reference data
startyear	first time step for which scenarios and reference scenario are expected to differ
ret	"boolean": just return TRUE/FALSE if check was successful "fails": data frame with mismatches between scenario and reference data "fixed": quite object with data correctly fixed on reference data "TRUE_or_fixed": TRUE if check was successful, fixed object otherwise
failfile	csv file to which mismatches are written to
relDiff	threshold for acceptable relative difference

### Value

see parameter 'ret'

### Author(s)

Oliver Richters

---

```
generateIIASASubmission
```

```
generateIIASASubmission
```

---

## Description

Generates an IIASA submission from REMIND or MAGPIE runs by applying a project-specific mapping. The script starts from 'mifs' which can be a directory with mif files, a vector of files or a quitte object. In outputDirectory/outputFilename, you will get the data in a joint xlsx or mif file.

## Usage

```
generateIIASASubmission(
  mifs = ".",
  mapping = NULL,
  model = NULL,
  removeFromScen = NULL,
  addToScen = NULL,
  dropRegi = "auto",
  outputDirectory = "output",
  outputFilename = "submission.xlsx",
  logFile = if (is.null(outputFilename)) NULL else paste0(gsub("\\.[a-zA-Z]+$",
  "_log.txt", outputFilename)),
  iiasatemplate = NULL,
  generatePlots = FALSE,
  timesteps = c(seq(2005, 2060, 5), seq(2070, 2100, 10)),
  checkSummation = TRUE,
  mappingFile = NULL,
  naAction = "na.omit"
)
```

## Arguments

mifs	path to mif files or directories with mif files of a REMIND run, or quitte object
mapping	mapping names such as c("AR6", "AR6_NGFS") or a vector of mapping file names. If NULL, the user is asked. Multiple mappings are concatenated.
model	name of model as registered with IIASA
removeFromScen	regular expression to be removed from scenario name (optional). Example: '_d50d95'
addToScen	string to be added as prefix to scenario name (optional)
dropRegi	regions to be dropped from output. Default is "auto" which drops aggregate regions for REMIND EU21. Set to NULL for none. Set c("auto", "World") for dropping EU21 aggregate plus World
outputDirectory	path to directory for the generated submission (default: output). If NULL, no files are written and logFile and outputFilename have no effect.

outputFilename	filename of the generated submission. Must be mif or xlsx file. If NULL, submission data is returned. If outputDirectory is set to NULL, this parameter has no effect.
logFile	path to the logfile with warnings as passed to generateMappingfile, checkIIASASubmission (default: outputDirectory/submission_log.txt). Set to FALSE for none. If outputDirectory is set to NULL, this parameter has no effect.
iiasatemplate	optional filename of xlsx or yaml file provided by IIASA used to delete superfluous variables and adapt units
generatePlots	boolean, whether to generate plots of failing summation checks. Needs outputDirectory not NULL.
timesteps	timesteps that are accepted in final submission
checkSummation	either TRUE to identify summation files from mapping, or filename, or FALSE
mappingFile	has no effect and is only kept for backwards-compatibility
naAction	a function which indicates what should happen when the data contain NA values.

## Details

To provide the mapping, two options exist:

- If you want to generate the mapping from one or more mappings from the inst/mappings folder, set `mapping = c("AR6", "AR6_NGFS")` or so.
- Alternatively, you can provide a path or a vector of paths to mapping files. If you provide your own mapping files, make sure they follow the standard format (see `getTemplate` for more information)
- It is also possible, to mix both options, e.g. `c("AR6", "/path/to/mapping_file.csv")`

In any case, multiple mapping files will be concatenated.

`iiasatemplate` is a xlsx or yaml file provided by IIASA with the variable + unit definitions that are accepted in the database. The function `'priceIndicesIIASA'` will be called to calculate price indices that are missing or with the wrong base year. `'checkIIASASubmission'` will be called to remove all variables that are not accepted in the database.

For all elements of the parameter `mapping` that contain a summation file in `inst/summations`, the function `'checkSummations'` is called to verify variable summation checks.

To alter the data, you can use those parameters: `model`, `addToScen`, `removeFromScen` and `timesteps`.

For a broader overview of the submission process, consult [https://github.com/remindmodel/remind/blob/develop/tutorials/13\\_](https://github.com/remindmodel/remind/blob/develop/tutorials/13_)

## Author(s)

Falk Benke, Oliver Richters

## Examples

```
## Not run:
# Simple use. Generates submission file in output folder:
generateIIASASubmission(
  mifs = "/path/to/REMIMD/mifs",
```

```
    model = "REMIND-MAgPIE 2.1-4.2",
    mapping = "NAVIGATE"
)

## End(Not run)
```

---

getMapping

*getMapping*

---

## Description

Retrieves latest mapping for a given project. Mappings must contain the columns "variable", "unit", "piam\_variable", "piam\_unit", "piam\_factor". Mappings are csv files with semicolon as a separator and no quotation marks around fields, see main README.Rd file

## Usage

```
getMapping(project = NULL, requiredColsOnly = FALSE)
```

## Arguments

`project` name of requested mapping, or file name pointing to a mapping

`requiredColsOnly` whether only the mandatory 5 columns are return set to TRUE if you want to concatenate mappings

## Author(s)

Falk Benke, Oliver Richters

## Examples

```
## Not run:
getMapping("ECEMF")
getMapping("/path/to/mapping/file")

## End(Not run)
```



---

getMappingVariables     *Retrieves all variables allocated to source potentially used in mappings to project variables*

---

**Description**

Retrieves all variables allocated to source potentially used in mappings to project variables

**Usage**

```
getMappingVariables(project = TRUE, sources = TRUE, addunit = TRUE)
```

**Arguments**

project	name of the project of requested mappings such as c('AR6', 'AR6_NGFS') or 'mapping.csv'. Use TRUE for all mappings.
sources	model abbreviation(s) used in 'source' column. R = REMIND, M = MAgPIE, T = EDGE-T, B = Brick, C = Climate/MAGICC, TRUE = all
addunit	boolean whether units should be appended as 'Variable (Unit)'

**Author(s)**

Falk Benke, Oliver Richters

**Examples**

```
getMappingVariables("AR6", "RT")
```

---

getREMINDTemplateVariables  
*legacy function to be used by remind2*

---

**Description**

legacy function to be used by remind2

**Usage**

```
getREMINDTemplateVariables(project)
```

**Arguments**

project	name of the project of requested mapping
---------	--

---

getSummations                      *Retrieves latest summation group file for a given project*

---

**Description**

Retrieves latest summation group file for a given project

**Usage**

```
getSummations(project = NULL, format = "dataframe")
```

**Arguments**

project	name of the project of requested summation group, or summation group file-name which can be a csv file of a xlsx file where the first sheet containing a 'variable' column is expected to have a 'components' column as well. If project is a <a href="https://files.ece.iiasa.ac.at/*">https://files.ece.iiasa.ac.at/*</a> xlsx URL, it will be automatically downloaded.
format	either "dataframe" or "list", the latter ignores the factor column

**Author(s)**

Oliver Richters

---

getTemplate                      *for backwards compatibility*

---

**Description**

for backwards compatibility

**Usage**

```
getTemplate(project = NULL)
```

**Arguments**

project	name of requested mapping, or file name pointing to a mapping
---------	---

---

loadIIASATemplate	<i>Loads IIASA template (xlsx or yaml)</i>
-------------------	--

---

**Description**

Loads IIASA template (xlsx or yaml)

**Usage**

```
loadIIASATemplate(iiasatemplate)
```

**Arguments**

iiasatemplate filename of xlsx or yaml file provided by IIASA

**Author(s)**

Oliver Richters

**Examples**

```
## Not run:  
# Simple use. Generates submission file in output folder:  
loadIIASATemplate(  
  iiasatemplate <- "template.xlsx"  
)  
  
## End(Not run)
```

---

mappingNames	<i>Retrieves mapping file names</i>
--------------	-------------------------------------

---

**Description**

Retrieves mapping file names

**Usage**

```
mappingNames(project = NULL)
```

**Arguments**

project name of the project of requested mapping. If not specified, all existing mappings will be returned

**Value**

mapping file(s)

**Author(s)**

Oliver Richters

---

plotIntercomparison *Model intercomparison plots: area plots based on summation groups, line plots for further variables. Creates a PDF for each model and scenario in the outputDirectory*

---

**Description**

Model intercomparison plots: area plots based on summation groups, line plots for further variables. Creates a PDF for each model and scenario in the outputDirectory

**Usage**

```
plotIntercomparison(
  mifFile,
  outputDirectory = "output",
  summationsFile = "AR6",
  renameModels = NULL,
  lineplotVariables = TRUE,
  areaplotVariables = TRUE,
  interactive = FALSE,
  mainReg = "World",
  plotby = c("model", "scenario"),
  diffTo = NULL,
  yearsBarPlot = c(2030, 2050),
  postfix = format(Sys.time(), "%Y-%m-%d_%H.%M.%S")
)
```

**Arguments**

mifFile path to the mif or xlsx file to apply summation checks to, or quitte object

outputDirectory path to directory to place one PDF for each model and scenario

summationsFile in inst/summations folder that describes the required summation groups. set to "extractVariableGroups" to extract it automatically from data based on |+| notation

renameModels vector with oldname = newname

lineplotVariables vector with variable names for lineplots or filenames of files containing a 'variable' column (or both)

areaplotVariables	vector with variable names for areaplot or filenames of files containing a 'variable' column. Only those available in the summationsFile can be plotted.
interactive	allows to select various settings interactively: subset of c("variable", "model", "scenario", "region", "period", "plotby", "diffto", "yearsBarPlot") or set to TRUE to select all of them
mainReg	region name of main region to be passed to mip
plotby	whether you would like to have everything plotted by scenario, model and/or onefile. set to NULL to be asked.
diffto	if specified, the difference to this scenario is calculated and plotted
yearsBarPlot	years for which bar plots are to be made.
postfix	to the filename, defaults to something like "_2024-09-05_12.47.28"

**Author(s)**

Oliver Richters

**Examples**

```
## Not run:
plotIntercomparison(quitte::quitte_example_dataAR6,
                    lineplotVariables = c("Temperature|Global Mean", "Population"))

## End(Not run)
```

---

priceIndicesAdd	<i>Add price index</i>
-----------------	------------------------

---

**Description**

Add price index

**Usage**

```
priceIndicesAdd(mifdata, priceIndices, scenBase = NULL, referenceYear = 2020)
```

**Arguments**

mifdata	file or data that can be converted into quitte object
priceIndices	vector of missing price index variable names
scenBase	optional scenario name of baseline scenario used to calculate index
referenceYear	in which index = 1

**Author(s)**

Oliver Richters

---

priceIndicesFix	<i>Fixes price indices with wrong reference year</i>
-----------------	--

---

**Description**

Fixes price indices with wrong reference year

**Usage**

```
priceIndicesFix(mifdata, priceIndices, referenceYear = 2020)
```

**Arguments**

mifdata	file or data that can be converted into quitte object
priceIndices	vector of missing price index variable names
referenceYear	in which index = 1

**Author(s)**

Oliver Richters

---

priceIndicesIIASA	<i>Add PriceIndex variables requested in iiasatemplate but missing in data, if PriceI is present in data. Extracts reference year automatically from unit</i>
-------------------	---

---

**Description**

Add PriceIndex variables requested in iiasatemplate but missing in data, if PriceI is present in data. Extracts reference year automatically from unit

**Usage**

```
priceIndicesIIASA(mifdata, iiasatemplate, scenBase = NULL)
```

**Arguments**

mifdata	file or data that can be converted into quitte object
iiasatemplate	filename of xlsx or yaml file provided by IIASA
scenBase	optional scenario name of baseline scenario used to calculate index

**Author(s)**

Oliver Richters

---

readMifs	<i>Pass a character vector containing filenames and directories. Returns data from all files and all '.mif' files in the directories.</i>
----------	---

---

**Description**

Pass a character vector containing filenames and directories. Returns data from all files and all '.mif' files in the directories.

**Usage**

```
readMifs(...)
```

**Arguments**

... path to mif files or directories with mif files of a REMIND run or quitte object

**Author(s)**

Falk Benke, Oliver Richters

**Examples**

```
## Not run:  
# Simple use. Generates submission file in output folder:  
readMifs(  
  mifs = "/path/to/REMIMD/mifs",  
)  
  
## End(Not run)
```

---

removePlus	<i>Remove  + ,  ++  etc. from variable names</i>
------------	--

---

**Description**

Remove |+|, |++| etc. from variable names

**Usage**

```
removePlus(x)
```

**Arguments**

x vector with variable names

**Value**

variable names without any plus notation

**Author(s)**

Oliver Richters

**Examples**

```
#' removePlus(c("FE|+|CDR", "FE|CDR|DACCS"))
```

---

```
renameOldInMappings    renameOldInMappings
```

---

**Description**

Loads "renamed\_piam\_variables.csv" and replaces all instances of old\_name in the piam\_variable columns of all mapping files Handle with care, better commit your changes before using it Best run this in your piamInterfaces folder: Rscript -e 'devtools::load\_all(); renameOldInMappings()'

**Usage**

```
renameOldInMappings(folder = ".")
```

**Arguments**

folder                    piamInterfaces folder

**Author(s)**

Oliver Richters

---

```
renameOldVariables    add variables that are missing based on a list of formulas
```

---

**Description**

add variables that are missing based on a list of formulas

**Usage**

```
renameOldVariables(mifdata, variables, logFile = NULL)
```



**Arguments**

mifdata	quite object or filename of mif file
variables	the list of requested variables
logFile	filename of file for logging

**Value**

quite object with adapted mif data

**Author(s)**

Oliver Richters

---

setLogFile	<i>Generate valid path to logFile and make sure the outputDirectory exists. If logFile is just a file name without any further path info, put logFile in outputDirectory</i>
------------	--

---

**Description**

Generate valid path to logFile and make sure the outputDirectory exists. If logFile is just a file name without any further path info, put logFile in outputDirectory

**Usage**

```
setLogFile(outputDirectory = NULL, logFile = NULL)
```

**Arguments**

outputDirectory	path to directory to place generated files
logFile	path or name for log file

**Author(s)**

Oliver Richters

---

summationsNames	<i>Retrieves summation group file names</i>
-----------------	---

---

**Description**

Retrieves summation group file names

**Usage**

```
summationsNames(project = NULL)
```

**Arguments**

project	name of the project of requested summation file. If not specified, all existing summation files will be returned
---------	--

**Value**

summation file name(s)

**Author(s)**

Oliver Richters

---

sumNamesWithFactors	<i>From mappingData, return the piam_variable sum as a string for a given exportname</i>
---------------------	--

---

**Description**

From mappingData, return the piam\_variable sum as a string for a given exportname

**Usage**

```
sumNamesWithFactors(mappingData, exportname)
```

**Arguments**

mappingData	mapping data as obtained by getMapping()
exportname	name to be matched in 'variable' column

**Author(s)**

Oliver Richters

---

templateNames	<i>for backwards compatibility</i>
---------------	------------------------------------

---

**Description**

for backwards compatibility

**Usage**

```
templateNames(project = NULL)
```

**Arguments**

project	name of the project of requested mapping. If not specified, all existing mappings will be returned
---------	--

---

variableInfo	<i>Provide information on variable, its mappings and summation groups</i>
--------------	---

---

**Description**

Provide information on variable, its mappings and summation groups

**Usage**

```
variableInfo(varname, mif = NULL, mapping = NULL)
```

**Arguments**

varname	string with variable name
mif	filename of miffile
mapping	vector of mapping shortcuts (AR6, NAVIGATE) or mapping filenames. NULL means all

**Value**

prints human-readable summary to the user

**Author(s)**

Oliver Richters

**Examples**

```
# Simple use. prints human-readable summary to the reader on Emi|CO2:
variableInfo(
  "Emi|CO2"
)
```

---

<code>writeMapping</code>	<i>writeMapping</i>
---------------------------	---------------------

---

**Description**

write data.frame obtained from `getMapping()` to mapping file.

**Usage**

```
writeMapping(mapping, filename)
```

**Arguments**

<code>mapping</code>	mapping data.frame
<code>filename</code>	filename.

**Author(s)**

Oliver Richters

# Index

areUnitsIdentical, [3](#)

checkFixUnits, [4](#)  
checkIIASASubmission, [4](#)  
checkMissingVars, [5](#)  
checkNGFS, [6](#)  
checkSummations, [7](#)  
checkSummationsRegional, [8](#)  
checkUnitFactor, [9](#)  
checkVarNames, [10](#)  
convertHistoricalData, [10](#)

extractReferenceYear, [11](#)

fillMissingSummations, [12](#)  
fillSummationPairs, [12](#)  
fixOnRef, [13](#)

generateIIASASubmission, [14](#)  
getMapping, [16](#)  
getMappingVariables, [17](#)  
getREMINDTemplateVariables, [17](#)  
getSummations, [18](#)  
getTemplate, [18](#)

loadIIASATemplate, [19](#)

mappingNames, [19](#)

piamInterfaces  
    (piamInterfaces-package), [2](#)  
piamInterfaces-package, [2](#)  
plotIntercomparison, [20](#)  
priceIndicesAdd, [21](#)  
priceIndicesFix, [22](#)  
priceIndicesIIASA, [22](#)

readMifs, [23](#)  
removePlus, [23](#)  
renameOldInMappings, [24](#)  
renameOldVariables, [24](#)

setLogFile, [25](#)  
summationsNames, [26](#)  
sumNamesWithFactors, [26](#)

templateNames, [27](#)

variableInfo, [27](#)

writeMapping, [28](#)