

Package: brick (via r-universe)

October 8, 2026

Type Package

Title Building sector model with heterogeneous renovation and construction of the stock

Version 0.14.2

Date 2026-08-31

Description This building stock model represents residential and commercial buildings at customisable regional and temporal resolution. The building stock is quantified in floor area and distinguished by building type (SFH/MFH) and location (rural/urban). In each building category, construction cohorts are tracked explicitly. This allows to characterise buildings specifically for each subset of buildings. The evolution of the building stock follows from the flows of constructed, renovated and demolished buildings and is optimised under cost minimisation with a benefit for heterogeneity in the choice of construction and renovation alternatives. This benefit captures heterogeneity in the preferences of the agents and the building structure.

License LGPL-3

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

Depends madrat, magclass

Imports digest, dplyr, ggplot2, gamstransfer (>= 3.0.1), gms (>= 0.26.0), grDevices, mip, piamutils (>= 0.0.10), pkgload, purrr, quitte, reportbrick (>= 0.12.2), scales, stringr, tidyr, withr, yaml

Suggests covr, devtools, testthat

Encoding UTF-8

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 libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-09-02 05:32:39 UTC

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

RemoteRef HEAD

RemoteSha d0be0ac2a595fe240211b412190e123b226fb1a5

Contents

brick-package	4
.addSpecCostToInput	5
.addTargetsToInput	6
.addTimeStamp	6
.adjustStepSizeAfterIteration	7
.aggregateDim	7
.argsHash	8
.asNumeric	8
.checkStepCondition	9
.checkStoppingCriterion	10
.combineOuterObjective	10
.combineStepSizeParams	11
.computeDescentDirection	12
.computeDeviation	12
.computeOuterObjective	13
.computeStepSizeParams	14
.computeSumSqTarget	15
.createListWithEmptyDf	15
.cropParamsToThist	16
.determineSpecCost	16
.explicitZero	17
.filter	18
.filterLevel	18
.findCfg	19
.findMinimumStepSize	19
.findOriginGdxFile	20
.findRegionMapping	20
.freezeParameters	21
.getMapping	21
.initOptimVar	22
.initOuterObjective	22
.initStepSize	23
.insertBetween	23
.isTRUE	24
.logLikelihood	24
.makeIdentVin	25
.namedLapply	25
.onLoad	26
.overwriteList	26
.pick	27

.printConvergenceMessage	27
.readCalibTarget	28
.readCfg	28
.readOuterObjectiveOptim	29
.refMapName	29
.rejectErrorRun	30
.sumSquare	30
.unique	31
.updateStepSize	32
.updateX	32
.updateXSelect	33
.writeCostIntang	34
.writeStock	34
addAssump	35
aggregateMatching	35
brick.file	36
checkGamsSuccess	36
checkStockBal	37
computeShareRen	38
copyGamsFiles	39
copyHistoryGdx	39
copyInitialGdx	40
createCalibrationTarget	40
createInputData	41
createMatchingData	42
createParameters	42
createRunFolder	43
createSets	44
expandSets	44
findLastRun	45
getBrickMapping	45
getDimMap	46
initModel	46
insertGamsCode	48
insertMatchingCode	49
isSlurmAvailable	49
listToDf	50
loadCalibrationTarget	50
loadMadrataData	51
makeHandle	51
periodFromConfig	52
plotRefDeviation	52
plotSummary	53
readConfig	53
readInput	54
readSymbol	55
reportMif	55
reweightMatchingReferences	56

runCalibration	56
runCalibrationLogit	57
runCalibrationOptim	58
runGams	59
setSlurmConfig	60
startModel	60
toModelResolution	61
Index	62

brick-package	<i>brick: Building sector model with heterogeneous renovation and construction of the stock</i>
---------------	---

Description

This building stock model represents residential and commercial buildings at customisable regional and temporal resolution. The building stock is quantified in floor area and distinguished by building type (SFH/MFH) and location (rural/urban). In each building category, construction cohorts are tracked explicitly. This allows to characterise buildings specifically for each subset of buildings. The evolution of the building stock follows from the flows of constructed, renovated and demolished buildings and is optimised under cost minimisation with a benefit for heterogeneity in the choice of construction and renovation alternatives. This benefit captures heterogeneity in the preferences of the agents and the building structure.

Author(s)

Maintainer: Robin Hasse <robin.hasse@pik-potsdam.de> ([ORCID](#))

Authors:

- Robin Hasse <robin.hasse@pik-potsdam.de> ([ORCID](#))
- Ricarda Rosemann <ricarda.rosemann@pik-potsdam.de> ([ORCID](#))

See Also

Useful links:

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

<code>.addSpecCostToInput</code>	<i>Write the specific costs to the input.gdx</i>
----------------------------------	--

Description

Write the specific costs to the input.gdx

Usage

```
.addSpecCostToInput(  
    m,  
    path,  
    optimVar,  
    xinit,  
    tcalib,  
    dims,  
    vinExists,  
    vinCalib,  
    varName = "x",  
    shiftIntang = TRUE  
)
```

Arguments

<code>m</code>	Gams transfer container with previous input data
<code>path</code>	character, path to this run
<code>optimVar</code>	list of data frames with optimization variables of all flows
<code>xinit</code>	list of data frames with initial specific intangible costs of all flows
<code>tcalib</code>	numeric, calibration time steps
<code>dims</code>	list of characters, dimensions of each flow
<code>vinExists</code>	data frame of vintages that exist for each time period
<code>vinCalib</code>	data frame with vintages that exist in calibration periods
<code>varName</code>	character, optimization variable to calculate specific costs from. Needs to be a column of <code>optimVarCon</code> and <code>optimVarRen</code> , should be either 'x' or 'xA'.
<code>shiftIntang</code>	logical indicating whether intangible costs should be shifted to positive range

.addTargetsToInput	<i>Add the calibration targets to the input.gdx</i>
--------------------	---

Description

Add the calibration targets to the input.gdx

Usage

.addTargetsToInput(mInput, path, p_calibTarget, dims)

Arguments

mInput	gamstransfer container of the input.gdx
path	character, path to output folder of this run
p_calibTarget	list of data frames of calibration targets
dims	list of characters, dimensions of data

Value

gamstransfer container with the modified input.gdx (invisibly)

.addTimeStamp	<i>Add time stamp</i>
---------------	-----------------------

Description

Append current date and time

Usage

.addTimeStamp(x)

Arguments

x	character
---	-----------

Value

character with time stamp suffix

`.adjustStepSizeAfterIteration`

Handle the case that the step size adaptation condition is not satisfied after the predefined number of iterations.

Description

If for any combination the condition is still not satisfied: Set the step size to zero and print a warning if no descent has been detected. Use the last step size of the iteration otherwise.

Usage

```
.adjustStepSizeAfterIteration(totalStep, stepSizeParams)
```

Arguments

`totalStep` data frame with combinations for which the condition is not satisfied
`stepSizeParams` data frame with the parameters of the step size adaptation procedure

Value

data frame with step size and related paramters, updated if applicable.

`.aggregateDim` *Aggregate data across given dimensions by a given function*

Description

Aggregate data across given dimensions by a given function

Usage

```
.aggregateDim(df, agg, func = sum, valueNames = "value")
```

Arguments

`df` data frame with data to be aggregated
`agg` character, columns with the dimensions to be aggregated
`func` function to be used for aggregation
`valueNames` character, names of the columns containing the values to aggregate

Value

data frame with aggregated data

.argHash	<i>Get hash for further arguments</i>
----------	---------------------------------------

Description

This is copied code from madrat to create the hash for additional arguments to ‘madrat::retrieveData’ besides the regionmapping. This way, we can recreate the file name of the tgz file with the input data.

Usage

```
.argHash(formals, useLabels)
```

Arguments

formals	named list with arguments
useLabels	logical, should be TRUE for now

Details

Ideally, this should be an exported function of madrat rather than copied code.

Value

character with hash

Author(s)

Robin Hasse

.asNumeric	<i>Convert to numeric vector</i>
------------	----------------------------------

Description

Convert factor to numeric vector. This only works for factors with numeric levels.

Usage

```
.asNumeric(x, warn = TRUE)
```

Arguments

x	factor vector
warn	logical, if TRUE, a warning is thrown when trying to convert a non-factor

Value

numeric vector

<code>.checkStepCondition</code>	<i>Check if the Armijo condition holds or if a local minimum of the outer objective has been identified. Return only data combinations for which it does not hold.</i>
----------------------------------	--

Description

Check if the Armijo condition holds or if a local minimum of the outer objective has been identified. Return only data combinations for which it does not hold.

Usage

```
.checkStepCondition(  
  prevStep,  
  stepSizeParams,  
  outerObjective,  
  sensitivityArmijo  
)
```

Arguments

<code>prevStep</code>	data frame with data combinations that did not satisfy the step size condition in the previous step
<code>stepSizeParams</code>	data frame with step size and related parameters
<code>outerObjective</code>	data frame containing the value of the outer objective function. Needs to contain the columns <code>f</code> and <code>fA</code> .
<code>sensitivityArmijo</code>	numeric, parameter of the Armijo condition specifying how strict the condition is

Value

data frame of combinations that satisfy neither the Armijo condition nor the local minimum criterion in the objective function

```
.checkStoppingCriterion
```

Check for the stopping criterion of the iteration

Description

Returns TRUE if and only if both the absolute and the relative stopping criterion are satisfied for all subsets.

Usage

```
.checkStoppingCriterion(
  outerObjective,
  p_calibTarget,
  tcalib,
  threshold,
  zeroFlow = FALSE
)
```

Arguments

outerObjective	data frame containing the value of the outer objective function.
p_calibTarget	list of data frames with historic flows
tcalib	numeric, time periods on which the calibration is executed
threshold	list of numeric, absolute and relative threshold of the stopping criterion
zeroFlow	logical, whether renovation flow data should be filtered for zero flows

Value

logical, whether the stopping criterion is satisfied for all subsets

```
.combineOuterObjective
```

Read in brick results and compute the total outer objective function by combining results from construction and renovation.

Description

Read in brick results and compute the total outer objective function by combining results from construction and renovation.

Usage

```
.combineOuterObjective(  
  m,  
  outerObjective,  
  p_calibTarget,  
  tcalib,  
  dims,  
  varName = "f",  
  agg = NULL  
)
```

Arguments

<code>m</code>	Gams transfer container to read brick results from
<code>outerObjective</code>	data frame to write the outer objective to
<code>p_calibTarget</code>	data frame with historic flows
<code>tcalib</code>	numeric, calibration time steps
<code>dims</code>	named character, dimensions of the flows
<code>varName</code>	character, column name in <code>outerObjective</code> to write the result to. Should be either 'f' or 'fA'.
<code>agg</code>	character, dimension(s) to aggregate the data over

Value

data frame with the outer objective computed from all calibration variables stored in column `f` or `fA`. If applicable, joined with previously computed value for `f`.

`.combineStepSizeParams`

Combine the step size paramters delta and phi-derivative from construction and renovation flows

Description

Combine the step size paramters delta and phi-derivative from construction and renovation flows

Usage

```
.combineStepSizeParams(deviation)
```

Arguments

<code>deviation</code>	named list of data frames with deviation and adjustment term <code>d</code> from all calibration variables
------------------------	--

Value

data frame with step size adaptation parameters computed from all calibration variables

.computeDescentDirection

Compute the descent direction from Brick results

Description

Read in objective function values for slightly shifted optimization variables and compute the gradient of the objective function. Set the adjustment term d to the negative of the gradient.

Usage

```
.computeDescentDirection(
  m,
  dims,
  tcalib,
  flow = c("construction", "renovation", "renovationBS", "renovationHS")
)
```

Arguments

m	Gams transfer container with current Brick results
dims	character, dimensions of the desired result object
tcalib	numeric, time steps to calibrate on
flow	character, either 'construction' or 'renovation'

Value

data frame with the adjustment term d.

.computeDeviation

Compute the deviation between historic data and Brick results as the log ratio. Then compute the adjustment term for the calibration d

Description

Compute the deviation between historic data and Brick results as the log ratio. Then compute the adjustment term for the calibration d

Usage

```
.computeDeviation(
  m,
  target,
  dims,
  tcalib,
  renAllowed,
  vinExists,
  flow = c("construction", "renovation", "renovationBS", "renovationHS"),
  agg = NULL,
  calibResolution = "full"
)
```

Arguments

<code>m</code>	Gams transfer container with current Brick results
<code>target</code>	data frame of historical data
<code>dims</code>	character, dimensions of historic data and Brick results
<code>tcalib</code>	numeric, time steps to calibrate on
<code>renAllowed</code>	data frame with allowed renovation transitions
<code>vinExists</code>	data frame with existing vintages for each time period
<code>flow</code>	character, either 'construction' or 'renovation'
<code>agg</code>	character, dimensions to aggregate Brick results and target data over
<code>calibResolution</code>	character, resolution of the calibration for renovation flows

Value

data frame containing the deviation from the target dev and the adjustment term d.

`.computeOuterObjective`

Evaluate the outer objective function.

Description

In case of the logit calibration, this outer objective function is virtual: We are not actually performing an optimization algorithm on this function, but only use it for the step size adaptation.

Usage

```
.computeOuterObjective(m, gamsVar, target, dims, tcalib, agg = NULL)
```

Arguments

<code>m</code>	Gams transfer container object to read set values from
<code>gamsVar</code>	data frame with brick results for the given variable
<code>target</code>	data frame with historic data
<code>dims</code>	character, dimensions of data
<code>tcalib</code>	numeric, calibration time steps
<code>agg</code>	character, dimension(s) to aggregate the data over

Value

data frame with the outer objective computed from one calibration variable

`.computeStepSizeParams`

Compute the step size adaptation paramters delta and phi-derivative

Description

Compute the step size adaptation paramters delta and phi-derivative

Usage

```
.computeStepSizeParams(deviation)
```

Arguments

<code>deviation</code>	data frame with deviation and adjustment term 'd'
------------------------	---

Value

data frame with step size adaptation parameters delta and phiDeriv.

<code>.computeSumSqTarget</code>	<i>Compute the sum of the squares for a calibration target</i>
----------------------------------	--

Description

Compute the sum of the squares for a calibration target

Usage

```
.computeSumSqTarget(target, zeroFlow = FALSE)
```

Arguments

<code>target</code>	data frame with calibration target data
<code>zeroFlow</code>	logical, whether data should be filtered for the zero flow

Value

data frame with the same dimensions as the outer objective, aggregated by computing the sum of the squares

<code>.createListWithEmptyDf</code>	<i>Create a list of empty data frames with given names</i>
-------------------------------------	--

Description

Create a list of empty data frames with given names

Usage

```
.createListWithEmptyDf(nm)
```

Arguments

<code>nm</code>	character, names of the empty data frames
-----------------	---

Value

list of empty data frames

<code>.cropParamsToThist</code>	<i>crop.gdx parameters to historic periods</i>
---------------------------------	--

Description

remove all records of temporal parameters that are outside of `thist` and make zero values explicit by filling with EPS.

Usage

```
.cropParamsToThist(gdx, thist)
```

Arguments

<code>gdx</code>	character, file path to.gdx
<code>thist</code>	numeric vector of historic periods

<code>.determineSpecCost</code>	<i>Assemble specific costs from initial specific costs and the optimization variable</i>
---------------------------------	--

Description

Assemble specific costs from initial specific costs and the optimization variable

Usage

```
.determineSpecCost(
  optimVar,
  xinit,
  dims,
  tcalib,
  flow = c("construction", "renovation", "renovationBS", "renovationHS"),
  varName = "x",
  vinExists = NULL,
  vinCalib = NULL,
  shiftIntang = TRUE
)
```

Arguments

optimVar	data frame with the optimization variable 'x' and optionally 'xA'
xinit	data frame with initial specific intangible costs
dims	character, dimensions of the optimization variable
tcalib	numeric, calibration time steps
flow	character, type of flow, either construction or renovation
varName	character, optimization variable to calculate specific costs from. Should be either 'x' or 'xA'.
vinExists	data frame of vintages that exist for each time period
vinCalib	data frame with vintages that exist in calibration periods
shiftIntang	logical indicating whether intangible costs should be shifted to positive range

Value

data frame with specific intangible costs for the given flow

.explicitZero	<i>Make zeros explicit</i>
---------------	----------------------------

Description

Replace zeros in value column with the gams special value EPS. This way, the value is saved in.gdx files by gams.

Usage

```
.explicitZero(x, value = "value")
```

Arguments

x	data.frame
value	character, name of value column

Value

data.frame with explicit zeros in value column

Author(s)

Robin Hasse

<code>.filter</code>	<i>Filter data frame rows with reference data frames</i>
----------------------	--

Description

Performs successive semi_joins with passed data frames by all common columns. Useful to filter allowed renovations or vintages that exist.

Usage

```
.filter(x, ...)
```

Arguments

<code>x</code>	data.frame
<code>...</code>	data frames that only have columns existing also in <code>x</code>

Value

filtered data frame

Author(s)

Robin Hasse

<code>.filterLevel</code>	<i>filter rows with specified entry in column</i>
---------------------------	---

Description

used to select a specific level or scenario from a data frame with alternative values.

Usage

```
.filterLevel(df, lvl, switchName = "", lvlCol = "level")
```

Arguments

<code>df</code>	data.frame
<code>lvl</code>	value used to select rows
<code>switchName</code>	character, corresponding BRICK switch name (only used for more informative error message)
<code>lvlCol</code>	character, colname containing <code>lvl</code>

Value

data.frame with selected rows without `lvlCol`

<code>.findCfg</code>	<i>Find config file path</i>
-----------------------	------------------------------

Description

Search for config file in multiple steps. The file is found if * config is a full file path already * config is the name of a file in the config folder * there is exactly one file in configFolder matching the pattern passed via config

Usage

```
.findCfg(config, configFolder, isFinalCfg)
```

Arguments

<code>config</code>	character, config file, either a path to a yaml file or the name of the file in 'inst/config/'
<code>configFolder</code>	character, directory to search for configs. If NULL, the BRICK-internal config folder is used.
<code>isFinalCfg</code>	logical, is this the final config and not an intermediate?

Value

file path to config

<code>.findMinimumStepSize</code>	<i>Determine the so far minimum step size. If the outerObjective is increasing again after a decrease, set the step size to the value yielding the current minimum.</i>
-----------------------------------	---

Description

Determine the so far minimum step size. If the outerObjective is increasing again after a decrease, set the step size to the value yielding the current minimum.

Usage

```
.findMinimumStepSize(totalStep, stepSizeParams, outerObjective)
```

Arguments

<code>totalStep</code>	data frame with combinations to perform the adjustment of the step size on
<code>stepSizeParams</code>	data frame with the parameters of the step size adjustment algorithm
<code>outerObjective</code>	data frame containing the value of the outer objective function.

Value

data frame with the updated step size where applicable and all relevant step size parameters

`.findOriginGdxFile` *find origin.gdx file*

Description

Depending on what is passed via `originGDX`, this function returns the file path if it exists, looks for recognised file names in the given directory if it exists or looks for the latest run with the given name in the `outputFolder`.

Usage

```
.findOriginGdxFile(originGdx, outputFolder)
```

Arguments

<code>originGdx</code>	character, file path to run or.gdx file used as historical or scenario name
<code>outputFolder</code>	directory of output folder, only required if <code>originGDX</code> is a scenario name

Value

file path to origin.gdx file

`.findRegionMapping` *Find region mapping*

Description

Find region mapping

Usage

```
.findRegionMapping(regionmapping)
```

Arguments

<code>regionmapping</code>	character. If NULL, country regions are used. It can otherwise be a path to a mapping file (length one) or a vector of length two with the file name and the location (where) of a madrat regional mapping.
----------------------------	---

Value

data frame with region mapping

.freezeParameters	<i>Freeze Parameters</i>
-------------------	--------------------------

Description

Select parameters and freeze them to the value in a selected period

Usage

```
.freezeParameters(m, params, period)
```

Arguments

m	gamstransfer Container with parameters
params	character vector of parameter names
period	numeric, period that all values are freed to; has to be an element of ttot

Value

gams Container

Author(s)

Robin Hasse

.getMapping	<i>Get BRICK mapping</i>
-------------	--------------------------

Description

Get BRICK mapping

Usage

```
.getMapping(mapping)
```

Arguments

mapping	character or list of characters, name of BRICK mapping. If "dim" or "agg" is passed, all corresponding dimension mappings are returned as named list.
---------	---

Value

data frame or list of data frames

Author(s)

Robin Hasse

<code>.initOptimVar</code>	<i>Initialize the data frame for the optimization variables</i>
----------------------------	---

Description

Initialize the data frame for the optimization variables

Usage

```
.initOptimVar(mInput, tcalib, dims, renAllowed, vinExists)
```

Arguments

<code>mInput</code>	Gamstransfer container with the input data
<code>tcalib</code>	numeric, calibration time periods
<code>dims</code>	character, dimensions to initialize the data with
<code>renAllowed</code>	data frame with allowed renovation transitions
<code>vinExists</code>	data frame with allowed vintage and time period combinations

Value

data frame of desired dimension with $x = 0$ (column to contain the optimization variable) and $x_A = 0$.

<code>.initOuterObjective</code>	<i>Initialize the data frame for the objective function value</i>
----------------------------------	---

Description

Initialize the data frame for the objective function value

Usage

```
.initOuterObjective(mInput)
```

Arguments

<code>mInput</code>	Gamstransfer container with the input data
---------------------	--

Value

data frame with required dimensions for the outer objective

<code>.initStepSize</code>	<i>Compute the initial step size of the step size iteration algorithm</i>
----------------------------	---

Description

Compute the initial step size of the step size iteration algorithm

Usage

```
.initStepSize(i, stepSizeParams, outerObjective, stepSizeInit)
```

Arguments

<code>i</code>	numeric, iteration number of overall calibration procedure
<code>stepSizeParams</code>	data frame with parameters for the step size adaptation
<code>outerObjective</code>	data frame with previous and current values of outer objective function; needs to contain the columns <code>fPrev</code> and <code>f</code>
<code>stepSizeInit</code>	numeric, lower bound for the initial step size

Value

data frame with the initial step size and initial values for the minimum outer objective and corresponding step size

<code>.insertBetween</code>	<i>Insert element between each element of a vector</i>
-----------------------------	--

Description

Insert element between each element of a vector

Usage

```
.insertBetween(x, insert)
```

Arguments

<code>x</code>	vector
<code>insert</code>	element to insert

Value

vector of length $\text{length}(x) * 2 - 1$

Author(s)

Robin Hasse

<code>.isTRUE</code>	<i>Test if vector elements are TRUE</i>
----------------------	---

Description

Perform `isTRUE` on each vector element

Usage

```
.isTRUE(x)
```

Arguments

<code>x</code>	vector
----------------	--------

Value

logical vector

Author(s)

Robin Hasse

<code>.logLikelihood</code>	<i>Calculate log-Likelihood of historical values</i>
-----------------------------	--

Description

Calculate log-Likelihood of historical values

Usage

```
.logLikelihood(res, target)
```

Arguments

<code>res</code>	numeric, result values
<code>target</code>	numeric, historic values

Value

numeric, negative of the log likelihood

<code>.makeIdentVin</code>	<i>Make vintage characteristic identical</i>
----------------------------	--

Description

fill value column with average across vintages

Usage

```
.makeIdentVin(x)
```

Arguments

<code>x</code>	data.frame with brick parameter, requires vin and value columns
----------------	---

Value

data.frame with identical structure but averaged values

<code>.namedLapply</code>	<i>Lapply with output named by given list</i>
---------------------------	---

Description

Lapply with output named by given list

Usage

```
.namedLapply(nm, ...)
```

Arguments

<code>nm</code>	list of characters, lapply input list and name of output
<code>...</code>	further inputs to lapply

Value

named list

<code>.onLoad</code>	<i>Check if gamstransfer is available</i>
----------------------	---

Description

Just a check to get a helpful error if gamstransfer is missing. Once the package is available on GitHub, it should be listed as a dependency and this file can be removed. The check is disabled during continuous integration (CI) i.a. GitHub actions and during `devtools:check` which is called by `lucode2::buildLibrary`.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

<code>libname</code>	not used
<code>pkgname</code>	not used

Author(s)

Robin Hasse

<code>.overwriteList</code>	<i>Overwrite list with another list</i>
-----------------------------	---

Description

Overwrite a named list with another named list. The result corresponds to the overwritten list unless a value is overwritten by the overwriting list.

Usage

```
.overwriteList(x, y, isFinalCfg, defaultCfgPath)
```

Arguments

<code>x</code>	named list, provides the structure and default values that can be overwritten by <code>y</code> .
<code>y</code>	named list, overwrites <code>x</code> wherever it has values different from <code>NULL</code> . Cannot have keys that are not specified in <code>x</code> .
<code>isFinalCfg</code>	logical, is this the final config and not an intermediate?
<code>defaultCfgPath</code>	character, path to default config. Only used for more helpful error message.

Details

This function is called recursively until the default config is reached. Therefor, the list structure will always correspond to the default config. No config based directly or indirectly on the default can have list keys that the default config doesn't have.

Value

named list with the structure of x that is (partly) overwritten by y.

<code>.pick</code>	<i>Pick lines from data.frame by identifiers</i>
--------------------	--

Description

Pick lines from data.frame by identifiers

Usage

```
.pick(x, ..., .colsMustExist = TRUE)
```

Arguments

x	data.frame
...	<column> = <selectionElement>
.colsMustExist	logical, if TRUE, an error is thrown for missing columns

Value

data.frame with filtered rows without identifier columns

<code>.printConvergenceMessage</code>	<i>Print message that the calibration converged.</i>
---------------------------------------	--

Description

Print message that the calibration converged.

Usage

```
.printConvergenceMessage(iteration, threshold)
```

Arguments

iteration	numeric, number of current iteration
threshold	list of numeric, absolute and relative threshold of the stopping criterion

<code>.readCalibTarget</code>	<i>Read calibration targets from input folder</i>
-------------------------------	---

Description

Read calibration targets from input folder

Usage

```
.readCalibTarget(dims, mInput)
```

Arguments

<code>dims</code>	named character, dimensions of each variable of the calibration
<code>mInput</code>	gamstransfer container with the input.gdx

Value

list of data frames with calibration targets for each variable

<code>.readCfg</code>	<i>Read config file</i>
-----------------------	-------------------------

Description

Read yaml file from given path, check minimum requirement (title exists) and save the file path as a attribute.

Usage

```
.readCfg(file)
```

Arguments

<code>file</code>	character, path to config file
-------------------	--------------------------------

Value

named list with config parameters

`.readOuterObjectiveOptim`*Read the outer objective from a gams calibration run.*

Description

Only applicable for optimization calibration.

Usage

```
.readOuterObjectiveOptim(m, outerObjective, varName = "f")
```

Arguments

<code>m</code>	Gams transfer container to read brick results from
<code>outerObjective</code>	data frame to write the outer objective to
<code>varName</code>	character, column name in <code>outerObjective</code> to write the result to. Should be either 'f' or 'fA'.

Value

data frame with outer objective read from Gams results stored in column `f` or `fA`. If applicable, joined with previously computed value for `f`.

`.refMapName`*Create reference mapping name*

Description

Create reference mapping name

Usage

```
.refMapName(ref, file = FALSE)
```

Arguments

<code>ref</code>	character, reference name
<code>file</code>	file extension appended unless set to FALSE

Value

character with reference mapping name

Author(s)

Robin Hasse

.rejectErrorRun	<i>Reject step sizes with unsuccessful Gams run</i>
-----------------	---

Description

If Gams was not successful, set the outer objective to infinity so that it cannot be accepted by the step size adaptation. Copy relevant files to trace the error (gdx, main.log and main.lst)

Usage

```
.rejectErrorRun(  
  path,  
  outerObjective,  
  gamsSuccess,  
  iteration,  
  iterationStepSize,  
  varName = "fA",  
 .gdxName = "calibrationA.gdx"  
)
```

Arguments

- path character, run path where all files, in particular Gams files, are stored
- outerObjective data frame with data on outer objective
- gamsSuccess data frame containing whether Gams was successful for each subset
- iteration numeric, current calibration iteration
- iterationStepSize numeric, current iteration of step size adaptation
- varName character, column of outer objective to be modified
- gdxName character, name of the gdx file to be copied.

Value

data frame, outer objective with modifications in the column given by varName

.sumSquare	<i>Calculate sum of squared differences between results and historic values</i>
------------	---

Description

Calculate sum of squared differences between results and historic values

Usage

```
.sumSquare(res, target = 0)
```

Arguments

res	numeric, result values
target	numeric, historic values

Value

numeric, sum of squared differences

.unique	<i>Extract elements that are unique to gams</i>
---------	---

Description

Wrapper around unique that also drops elements that differ in cases but are identical to gams as it is case-insensitive.

Usage

```
.unique(x)
```

Arguments

x	a vector or a data frame
---	--------------------------

Value

an object of the same kind but possibly fewer elements. Vectors can be shorter and data frames can have fewer rows

Author(s)

Robin Hasse

.updateStepSize	<i>Update the step size for the selected combinations</i>
-----------------	---

Description

Update the step size for the selected combinations

Usage

```
.updateStepSize(
  totalStep,
  stepSizeParams,
  outerObjective,
  stepReduction,
  lastIteration
)
```

Arguments

totalStep	data frame with combinations to perform the adjustment of the step size on
stepSizeParams	data frame with the parameters of the step size adjustment algorithm
outerObjective	data frame containing the value of the outer objective function.
stepReduction	numeric, factor applied to stepSize to reduce the step size. Should be < 1.
lastIteration	logical, if this is the last iteration of the step size adaptation

Value

data frame with the current step size and all related parameters. The step size and the minimum outer objective and minimum step size are updated for selected combinations only.

.updateX	<i>Update the optimization variable 'x'</i>
----------	---

Description

Adjust 'x' by adding the adjustment term 'd' multiplied by the step size

Usage

```
.updateX(optimVar, deviation, stepSizeParams, dims, nameTo = "x")
```

Arguments

<code>optimVar</code>	data frame with the optimization variable 'x' and optionally 'xA'
<code>deviation</code>	data frame with deviation and adjustment term 'd'
<code>stepSizeParams</code>	data frame with parameters of step size adaptation, including step size 'stepSize'
<code>dims</code>	character, dimensions of the optimization variable
<code>nameTo</code>	character, optimization variable to write the result to

Value

data frame with the updated optimization variable x or xA

<code>.updateXSelect</code>	<i>Update the optimization variable 'x' for selected combinations only</i>
-----------------------------	--

Description

Update the optimization variable 'x' for selected combinations only

Usage

```
.updateXSelect(totalStep, optimVar, deviation, stepSizeParams, dims)
```

Arguments

<code>totalStep</code>	data frame with combinations to perform the adjustment of 'x' on
<code>optimVar</code>	data frame with the optimization variables
<code>deviation</code>	data frame with the deviation from historic data and the adjustment parameter 'd'
<code>stepSizeParams</code>	data frame with the parameters of the step size adaptation procedure
<code>dims</code>	character, dimensions of the optimization variable

Value

data frame with the optimization variable which was updated for the desired combinations

.writeCostIntang	<i>Write the intangible costs to a .csv file</i>
------------------	--

Description

Write the intangible costs to a .csv file

Usage

```
.writeCostIntang(
  file,
  optimVar,
  xinit,
  dims,
  tcalib,
  flow = c("construction", "renovation", "renovationBS", "renovationHS"),
  vinExists = NULL,
  vinCalib = NULL
)
```

Arguments

file	character, path to file to which results should be written
optimVar	data frame with optimization variable, needs to contain column 'x'
xinit	data frame with initial intangible specific costs
dims	character, dimensions of the optimization variable
tcalib	numeric, calibration time steps
flow	character, specifies flow considered. Must be construction or renovation
vinExists	data frame with existing vintage and time step combinations
vinCalib	data frame with vintage and time step combinations that exist during calibration

.writeStock	<i>Write the stock of the calibration result to a csv file</i>
-------------	--

Description

Write the stock of the calibration result to a csv file

Usage

```
.writeStock(m, path)
```

Arguments

m	Gams container with the calibration output.gdx
path	character, path to the calibration run

addAssump	<i>Add assumed intangible costs</i>
-----------	-------------------------------------

Description

Add assumed intangible costs

Usage

```
addAssump(df, assumpFile, vinDimMap = NULL, key = NULL)
```

Arguments

df	data.frame for the cost of construction or renovation
assumpFile	character, file path to assumption file
vinDimMap	data frame with mapping for vintages to start and end years.
key	character, renovation asset, either "BS" (building shell) or "HS" heating system.

Value

data frame with added intangible cost

Author(s)

Robin Hasse

aggregateMatching	<i>Aggregate matching run results for calibration</i>
-------------------	---

Description

Temporary function that prepares historic stock and flows for calibration

Usage

```
aggregateMatching(path, config, overwrite = FALSE)
```

Arguments

path	character, path to calibration run
config	named list, configuration of calibration run
overwrite	logical, should existing data be overwritten?

Details

This functionality will migrate to mredgebuidlings but this infrastructure requires more time to be developed.

Author(s)

Robin Hasse

brick.file	<i>Find the full file names of files in BRICK</i>
------------	---

Description

This is meant to work with the installed package BRICK but also when loading the package via devtools::load_all("path/to/brick")

Usage

```
brick.file(..., mustWork = TRUE)
```

Arguments

...	character vectors, specifying subdirectory and files within brick
mustWork	if TRUE, an error is given if there are no matching files

Value

A character vector of positive length, containing the file paths that matched ... in BRICK.

Author(s)

Robin Hasse

checkGamsSuccess	<i>Check whether Gams finished successfully</i>
------------------	---

Description

Check which file is the.gdx file written by Gams from main.log. Read model and solver status from this.gdx file and print.

Usage

```
checkGamsSuccess(path, runType = "scenario", silent = FALSE)
```

Arguments

path	character, path to search for gams output files
runType	character, type of this run
silent	logical, whether no messages should be printed.

Details

Stop with error message if: - main.log does not exist - the.gdx file does not exist - more than one.gdx file was written.

Value

logical, whether GAMS was successful for all subsets

Author(s)

Ricarda Rosemann

checkStockBal	<i>Check if stock balance is fulfilled</i>
---------------	--

Description

Only works for sequential renovation right now.

Usage

```
checkStockBal(path)
```

Arguments

path	character, directory of a brick run
------	-------------------------------------

Value

named list with deviations of each balance equation

Author(s)

Robin Hasse

computeShareRen	<i>Compute the share of heating systems to be renovated</i>
-----------------	---

Description

Calculate share of buildings that need to be renovated or demolished between given time steps assuming a Weibull distribution of the technology life time.

Usage

```
computeShareRen(
  params,
  m,
  cutOffShare = 1,
  standingLifeTime = FALSE,
  timePeriods = NULL,
  vintages = NULL
)
```

Arguments

params	data frame with parameters of the Weibull distribution
m	Gams container with at least ttotIn, ttotOut, tinit and p_dt
cutOffShare	numeric, renovation share at which we cut off the distribution function by jumping to 1
standingLifeTime	logical, whether standing lifetime needs to be considered
timePeriods	data frame with time periods for which the calculations should be done
vintages	data frame with vintages and their starting and end year

Details

If standingLifeTime = TRUE, a standing stock is considered. For most technologies we assume a constant inflow between the start of the respective vintage cohort and initial time. For the more recent technology of heat pumps (ehp1), we assume that the inflow decreases quadratically with the age of the system going back from initial time to the start of the vintage cohort, but at most 25 years. For the rather outdated technologies solids (sobo) and liquids (libo) boilers, we assume that the inflow increases quadratically with the age of the system going back from initial time to the start of the vintage cohort, but at most 25 (sobo) or 20 (libo) years.

Value

Data frame of shares to be renovated in each time period

copyGamsFiles	<i>Copy gams scripts to output folder</i>
---------------	---

Description

Copy gams scripts to output folder

Usage

```
copyGamsFiles(path, overwrite = FALSE)
```

Arguments

path	character vector with folders to write input data into
overwrite	logical, should existing input.gdx be overwritten?

Author(s)

Robin Hasse

copyHistoryGdx	<i>Copy history.gdx to output folder</i>
----------------	--

Description

Copy history.gdx to output folder

Usage

```
copyHistoryGdx(  
  path,  
  outputFolder = NULL,  
  config,  
  overwrite = FALSE,  
  thistOnly = TRUE  
)
```

Arguments

path	character vector with folders to write input data into
outputFolder	directory of output folder
config	named list with run configuration
overwrite	logical, should existing input.gdx be overwritten?
thistOnly	logical, crop temporal parameters to historic periods

Author(s)

Robin Hasse

copyInitialGdx	<i>Copy initial.gdx to output folder</i>
----------------	--

Description

Copy initial.gdx to output folder

Usage

```
copyInitialGdx(path, startingPoint, overwrite = FALSE)
```

Arguments

path	character vector with folders to write input data into
startingPoint	character with path to the starting point or the respective directory
overwrite	logical, should existing input.gdx be overwritten?

Author(s)

Robin Hasse

createCalibrationTarget	<i>Create calibration targets</i>
-------------------------	-----------------------------------

Description

read results from matching run and aggregate them spatially and temporally to the resolution of calibration runs

Usage

```
createCalibrationTarget(  
  path,  
  calibConfig,  
  outDir = NULL,  
  digits = 4,  
  rmCorrectionRun = FALSE  
)
```

Arguments

path	character, directory of matching folder
calibConfig	character, calibration config
outDir	character, directory to save calibration target files in
digits	integer indicating the number of decimal places
rmCorrectionRun	logical, if TRUE, the run folder of the renovation correction is eventually removed

Details

The temporal aggregation of renovation flows is non-trivial: It is generally possible that buildings can change states multiple time within one aggregated time step but they can't be tracked through renovations individually. Currently, we first assume the mean of renovations across all years of an aggregated time step. In a second step, we perform a renovation correction run with BRICK that minimises deviation from these mean flows and the stock subject to the stock balance and building shell and heating system life times. The correction assures consistency, mostly changing the zero flow that depends most on the temporal resolution. This renovation correction run is performed in a sub folder of the matching run.

Value

named list with aggregated and corrected stock and flows

Author(s)

Robin Hasse

createInputData	<i>Create input data</i>
-----------------	--------------------------

Description

Create a complete set of input data for the gams optimisation.

Usage

```
createInputData(path, config, overwrite = FALSE)
```

Arguments

path	character vector with folders to write input data into
config	named list with run configuration
overwrite	logical, should existing input.gdx be overwritten?

Details

This function reads static input data according to the input data revision in the config and creates all required sets and parameters for the gams optimisation depending on the switches in the config.

Author(s)

Robin Hasse

createMatchingData	<i>Create data for reference matching</i>
--------------------	---

Description

This function also performs auto gams code generation to load the reference sets and parameters

Usage

```
createMatchingData(path, config, overwrite = FALSE)
```

Arguments

path	character vector with folders to run the model in
config	run configurations
overwrite	logical, should existing data be overwritten?

Author(s)

Robin Hasse

createParameters	<i>Create parameters</i>
------------------	--------------------------

Description

Add all parameters to gams container based on config

Usage

```
createParameters(m, config, inputDir)
```

Arguments

m	gams Container, central object to store all data for input.gdx
config	named list with run configuration
inputDir	directory of input folder

Value

gams Container with parameters added

Author(s)

Robin Hasse

createRunFolder	<i>Create new run folder</i>
-----------------	------------------------------

Description

Create a folder for the model to run in and copy required gams files there.

Usage

```
createRunFolder(  
  path,  
  config = NULL,  
  overwrite = FALSE,  
  recursive = FALSE,  
  showWarnings = TRUE  
)
```

Arguments

path	character vector, containing the path(s) to the folder(s) to be created.
config	list with run configuration
overwrite	logical, should existing folders be overwritten?
recursive	logical, should elements of the path other than the last be created?
showWarnings	logical, should a warning on not created paths be shown?

Author(s)

Robin Hasse

createSets

Create sets

Description

Add all sets to gams container based on config

Usage

```
createSets(m, config)
```

Arguments

m	gams Container, central object to store all data for input.gdx
config	named list with run configuration

Value

gams Container with sets added

Author(s)

Robin Hasse

expandSets

Expand set values to data frame

Description

Create a data frame that has a column for each set and the full crossing of all set entries as rows

Usage

```
expandSets(..., .m = NULL)
```

Arguments

...	gams sets
.m	gams Container with sets referenced in ...

Value

data.frame with full crossing of set entries

Author(s)

Robin Hasse

findLastRun	<i>Determine the latest path of a run</i>
-------------	---

Description

Search the given output folder for the run with the most recent time stamp and return the path to this run. If the output folder contains only one run, return its path, also if it does not contain a time stamp.

Usage

```
findLastRun(outputFolder)
```

Arguments

outputFolder	character, output folder to search in
--------------	---------------------------------------

Author(s)

Ricarda Rosemann

getBrickMapping	<i>Retrieve mapping file from BRICK</i>
-----------------	---

Description

Retrieve mapping file from BRICK

Usage

```
getBrickMapping(  
  name,  
  type = "sectoral",  
  error.missing = TRUE,  
  returnPathOnly = FALSE  
)
```

Arguments

name	character, file name of the mapping file
type	character, Mapping type (e.g. "regional", or "sectoral")
error.missing	logical, return error if the file does not exist
returnPathOnly	logical, file name of the mapping file

Author(s)

Robin Hasse

getDimMap	<i>Get Brick dimension mapping</i>
-----------	------------------------------------

Description

Get Brick dimension mapping

Usage

```
getDimMap(dim, granularity = NULL)
```

Arguments

dim	character, dimension name
granularity	character, name of BRICK granularity

Value

data.frame with dimension mapping

Author(s)

Robin Hasse

initModel	<i>Initialize the model:</i>
-----------	------------------------------

Description

Preparations of a model run, send the model to SLURM if desired

Usage

```
initModel(
  config = NULL,
  path = NULL,
  configFolder = NULL,
  outputFolder = "output",
  references = NULL,
  restart = FALSE,
  runReporting = TRUE,
  sendToSlurm = NULL,
  slurmQOS = NULL,
  tasksPerNode = NULL,
  timeLimit = NULL,
  tasks32 = FALSE
)
```

Arguments

config	run configurations
path	character vector with folders to run the model in
configFolder	character, directory to search for configs. If NULL, the BRICK-internal config folder is used.
outputFolder	directory of output folder
references	named character vector of matching references
restart	logical or character vector of elements to be restarted. If FALSE (default), then no restart is initiated. If TRUE, then the run in the given path or the latest run is restarted with default settings. Allowed elements of the character vector are: • "copyGams" to recopy the Gams scripts (necessary if changes were made in Gams code) • "createInput" to recreate the input data • "createMatching" to recreate the matching data • "reweightMatching" to update the matching reference selection and weighting • "useAsStart" to use the run from which we restart as the starting point • "none" (or any other string) to do none of the above
runReporting	logical, whether to run the reporting, i.e. write the mif
sendToSlurm	boolean whether or not the run should be started via SLURM
slurmQOS	character, slurm QOS to be used
tasksPerNode	numeric, number of tasks per node to be requested
timeLimit	character, time limit of the slurm job given in the format hh:mm:ss
tasks32	boolean whether or not the SLURM run should be with 32 tasks

Details

This function creates the run folder with the necessary config and gams files. It then either calls the function to start the model directly or passes the model to SLURM.

Value

path (invisible)

Author(s)

Ricarda Rosemann

insertGamsCode	<i>Insert code into gams scripts</i>
----------------	--------------------------------------

Description

Lines between marker comments are replaced by given lines. The respective script file gets overwritten.

Usage

```
insertGamsCode(path, script, section, insert, keepComments = TRUE)
```

Arguments

path	character, description
script	character, name of gams script
section	character, section name
insert	character vector, code line that are inserted
keepComments	logical, if TRUE, lines starting with !! are kept and the inserted lines follow below

Details

Before

```
!! AUTOCODE.<section>
<old code>
!! AUTOCODE_end.<section>
```

After

```
!! AUTOCODE.<section>
<insert>
!! AUTOCODE_end.<section>
```

Author(s)

Robin Hasse

insertMatchingCode	<i>Insert reference-dependent code for matching into gams scripts</i>
--------------------	---

Description

Insert reference-dependent code for matching into gams scripts

Usage

```
insertMatchingCode(path)
```

Arguments

path	character vector with folders to run the model in
------	---

Author(s)

Robin Hasse

isSlurmAvailable	<i>Is Slurm Available</i>
------------------	---------------------------

Description

Checks whether slurm is available so that jobs can be submitted, e.g., via sbatch . . .

Usage

```
isSlurmAvailable()
```

Value

logical(1). Whether slurm is available.

listToDf	<i>Convert named list to data.frame</i>
----------	---

Description

Convert named list to data.frame

Usage

```
listToDf(x, split = NULL)
```

Arguments

x	named list with keys as column names
split	character, split names by this pattern and assume same value for each separated item

Value

data.frame with column for each key

Author(s)

Robin Hasse

loadCalibrationTarget	<i>Load calibration targets from matching results</i>
-----------------------	---

Description

Load calibration targets from matching results

Usage

```
loadCalibrationTarget(config)
```

Arguments

config	named list with run configuration
--------	-----------------------------------

Value

directory of input folder

Author(s)

Ricarda Rosemann

loadMadratData	<i>Load Input data from mredgebuidings</i>
----------------	--

Description

Load Input data from mredgebuidings

Usage

```
loadMadratData(config)
```

Arguments

config	named list with run configuration
--------	-----------------------------------

Value

directory of input folder

Author(s)

Robin Hasse

makeHandle	<i>Named list to handle string</i>
------------	------------------------------------

Description

Helper function to turn a named list into a string appended to the gams command line call

Usage

```
makeHandle(lst, type = c("gams", "model"))
```

Arguments

lst	named list of flags
type	character, type of flags (either "gams" or "model")

Details

Gams parameters are lower case flags and follow one minus '-'. Model switches are upper case flags and follow two minuses '--'.

Value

character of flags

Author(s)

Robin Hasse

periodFromConfig	<i>get Period from config</i>
------------------	-------------------------------

Description

extract different periods from a given BRICK config

Usage

periodFromConfig(config, periodType)

Arguments

config	named list with run configuration
periodType	type of period(s)

Value

numeric vector with periods

Author(s)

Robin Hasse

plotRefDeviation	<i>Plot heat map of reference deviation</i>
------------------	---

Description

Plot heat map of reference deviation

Usage

plotRefDeviation(path, dropZeroWeightRefs = TRUE, dropUnmappedRefVars = TRUE)

Arguments

path	character, directory of output folder
dropZeroWeightRefs	logical, if TRUE, references with zero weight are not shown
dropUnmappedRefVars	logical, if TRUE, reference variables that are not mapped to BRICK variables are not shown

Author(s)

Robin Hasse

plotSummary*Plot Summary of a run*

Description

Plot an overview of the stock and flows

Usage

```
plotSummary(path, facet = "typ", showHistStock = FALSE, splitRen = FALSE)
```

Arguments

path	character, path to the run
facet	character, dimension to resolve as facets
showHistStock	logical, show given historic next to the modeled stock
splitRen	logical, plot renovation with identical replacement semi-transparent

Author(s)

Robin Hasse

readConfig*Read config file*

Description

Read config file in yaml format

Usage

```
readConfig(config = NULL, configFolder = NULL, readDirect = FALSE)
```

Arguments

config	character, config file, either a path to a yaml file or the name of the file in configFolder
configFolder	character, directory to search for configs. If NULL, the BRICK-internal config folder is used.
readDirect	logical, specify whether config is a valid path to a config file that should be read directly.

Details

If no argument is given, the default config is used.

Value

named list with run config

Author(s)

Robin Hasse

readInput

Read madrat input files from input folder

Description

Read madrat input files from input folder

Usage

```
readInput(filename, dims, inputDir = NULL)
```

Arguments

filename	character, filename
dims	character vector, column names
inputDir	character, directory of input folder

Value

data.frame with data of the given input file

Author(s)

Robin Hasse

readSymbol	<i>Read symbol from gams container</i>
------------	--

Description

Read symbol from gams container

Usage

```
readSymbol(x, symbol = NULL, selectArea = TRUE, stringAsFactor = TRUE)
```

Arguments

x	gams Container, Parameter, Variable or Set or path to.gdx
symbol	character, name of gams object if x is a Container else NULL
selectArea	logical, select area quantity and remove this dimension
stringAsFactor	logical, keep default factors from gams

Author(s)

Robin Hasse

reportMif	<i>Create mif file for model run</i>
-----------	--------------------------------------

Description

The mif file contains reporting variables for the given model run.

Usage

```
reportMif(path, file = NULL, tmp1 = NULL)
```

Arguments

path	character, path to the run
file	character, path of mif file. If NULL, default file name in run folder will be used. If FALSE, no file is written and the mif data is returned instead.
tmp1	character, BRICK reporting template. There has to be a brickSets mapping named with the same suffix: brickSets_<tmp1>.yaml. If NULL, the config setting is used.

Author(s)

Robin Hasse

`reweightMatchingReferences`*Update matching reference selection and weights*

Description

Read reference configuration from existing run und update the set of selected references and the reference weights in the reference.gdx

Usage

```
reweightMatchingReferences(path)
```

Arguments

path character, path to run

Value

gamstransfer container with updates reference.gdx content

Author(s)

Robin Hasse

`runCalibration`*Run the brick calibration*

Description

Two methods for the Brick calibration are available: The logit calibration and the optimization calibration.

Usage

```
runCalibration(  
  path,  
  parameters,  
  tcalib,  
  gamsOptions = NULL,  
  switches = NULL,  
  fileName = "main.gms",  
  gamsCall = "gams"  
)
```

Arguments

path	character vector with folders to run gams in
parameters	named list of calibration parameters
tcalib	numeric, time periods to calibrate on
gamsOptions	named list of GAMS options
switches	named list of model switches
fileName	character vector with gams file names
gamsCall	system command to call gams

Details

The general procedure is as follows: We aim to minimize the deviation between historic values and Brick results. An intangible component of the specific construction and renovation costs is iteratively adjusted to achieve this. Thus the intangible costs serve as the optimization variable x and a functional evaluation of the deviation represents the objective function f .

Brick is run iteratively and from each run, an adjustment term d is computed. The optimization variable is adjusted according to $x = x + \text{stepSize} * d$, where the step size is determined according to the Armijo step size adaptation algorithm, supplemented with a heuristic that picks the step size resulting in the (locally) minimum outer objective.

The implementation is centered around the following objects:

- The optimization variable x and its variant(s) required in the step size adaptation are gathered in the data frame `optimVarCon` for construction and `optimVarRen` for renovation.
- The objective function value f and its variant(s) are gathered in the data frame `outerObjective`
- The adjustment term d and related variables if applicable are gathered in the data frame `deviationCon` for construction and `deviationRen` for renovation.
- The step size `stepSize` and related parameters required for the step size adaptation algorithm are gathered in the data frame `stepSizeParams`.

Author(s)

Ricarda Rosemann

runCalibrationLogit *Call the logit brick calibration*

Description

The adjustment term is computed as if Brick were a pure logit model: $d = \log(\langle \text{brick result} \rangle / \langle \text{historic value} \rangle)$.

Usage

```
runCalibrationLogit(
  path,
  parameters,
  tcalib,
  dims,
  gamsOptions = NULL,
  switches = NULL,
  fileName = "main.gms",
  gamsCall = "gams"
)
```

Arguments

path	character vector with folders to run gams in
parameters	named list of calibration parameters
tcalib	numeric, time periods to calibrate on
dims	list of characters with dimensions of stock and flow variables
gamsOptions	named list of GAMS options
switches	named list of model switches
fileName	character vector with gams file names
gamsCall	system command to call gams

Details

Here the armijo step size algorithm may not be applicable; if this is the case, the first local minimum objective is used.

runCalibrationOptim	<i>Call the optimization brick calibration</i>
---------------------	--

Description

This implements a gradient descent method to perform the minimization of the deviation. The adjustment term is computed as the negative of the gradient of the objective function.

Usage

```
runCalibrationOptim(
  path,
  parameters,
  tcalib,
  dims,
  gamsOptions = NULL,
  switches = NULL,
```

```

    fileName = "main.gms",
    gamsCall = "gams"
)

```

Arguments

path	character vector with folders to run gams in
parameters	named list of calibration parameters
tcalib	numeric, time periods to calibrate on
dims	list of characters with dimensions of stock and flow variables
gamsOptions	named list of GAMS options
switches	named list of model switches
fileName	character vector with gams file names
gamsCall	system command to call gams

runGams	<i>Run gams optimisation</i>
---------	------------------------------

Description

Run the gams model in a given directory.

Usage

```

runGams(
  path,
  gamsOptions = NULL,
  switches = NULL,
  fileName = "main.gms",
  gamsCall = "gams"
)

```

Arguments

path	character vector with folders to run gams in
gamsOptions	named list of GAMS options
switches	named list of model switches
fileName	character vector with gams file names
gamsCall	system command to call gams

Details

The file ‘input.gdx’ has to exist in the given directory.

Author(s)

Robin Hasse

setSlurmConfig	<i>Set the SLURM configuration</i>
----------------	------------------------------------

Description

Based on user input, construct the SLURM configuration string.

Usage

```
setSlurmConfig(slurmQOS, tasksPerNode = 16, tasks32 = FALSE, timeLimit = NULL)
```

Arguments

slurmQOS	string, name of the desired QOS (Quality of Service)
tasksPerNode	numeric, number of tasks per node to be requested
tasks32	boolean, specify whether a node with 32 tasks should be requested
timeLimit	character, time limit of the slurm job given in the format hh:mm:ss

Details

Check if SLURM configuration is admissible.

Value

string with SLURM configuration

Author(s)

Ricarda Rosemann

startModel	<i>Start the model</i>
------------	------------------------

Description

Run the model with given configuration.

Usage

```
startModel(path, runReporting = TRUE)
```

Arguments

path	character vector with folders to run the model in
runReporting	logical, whether to run the reporting, i.e. write the mif

Details

This function creates a run folder with necessary gams files if missing. It then computes the input data and finally runs the optimisation.

Author(s)

Robin Hasse

toModelResolution	<i>interpolate and filter to get model resolution</i>
-------------------	---

Description

Missing periods are interpolated linearly and extrapolated constantly, additional periods are removed and all other dimensions are filtered to the elements defined in the model sets.

Usage

```
toModelResolution(x, m, value = "value", unfilteredDims = NULL)
```

Arguments

x	data.frame with temporal dimension
m	gams Container with sets as known dimensions
value	character, name of value column
unfilteredDims	character, dimensions to be neither filtered nor extrapolated

Value

data.frame with temporal resolution according to model

Author(s)

Robin Hasse

Index

- [.addSpecCostToInput, 5](#)
- [.addTargetsToInput, 6](#)
- [.addTimeStamp, 6](#)
- [.adjustStepSizeAfterIteration, 7](#)
- [.aggregateDim, 7](#)
- [.argsHash, 8](#)
- [.asNumeric, 8](#)
- [.checkStepCondition, 9](#)
- [.checkStoppingCriterion, 10](#)
- [.combineOuterObjective, 10](#)
- [.combineStepSizeParams, 11](#)
- [.computeDescentDirection, 12](#)
- [.computeDeviation, 12](#)
- [.computeOuterObjective, 13](#)
- [.computeStepSizeParams, 14](#)
- [.computeSumSqTarget, 15](#)
- [.createListWithEmptyDf, 15](#)
- [.cropParamsToThist, 16](#)
- [.determineSpecCost, 16](#)
- [.explicitZero, 17](#)
- [.filter, 18](#)
- [.filterLevel, 18](#)
- [.findCfg, 19](#)
- [.findMinimumStepSize, 19](#)
- [.findOriginGdxFile, 20](#)
- [.findRegionMapping, 20](#)
- [.freezeParameters, 21](#)
- [.getMapping, 21](#)
- [.initOptimVar, 22](#)
- [.initOuterObjective, 22](#)
- [.initStepSize, 23](#)
- [.insertBetween, 23](#)
- [.isTRUE, 24](#)
- [.logLikelihood, 24](#)
- [.makeIdentVin, 25](#)
- [.namedLapply, 25](#)
- [.onLoad, 26](#)
- [.overwriteList, 26](#)
- [.pick, 27](#)
- [.printConvergenceMessage, 27](#)
- [.readCalibTarget, 28](#)
- [.readCfg, 28](#)
- [.readOuterObjectiveOptim, 29](#)
- [.refMapName, 29](#)
- [.rejectErrorRun, 30](#)
- [.sumSquare, 30](#)
- [.unique, 31](#)
- [.updateStepSize, 32](#)
- [.updateX, 32](#)
- [.updateXSelect, 33](#)
- [.writeCostIntang, 34](#)
- [.writeStock, 34](#)
- [addAssump, 35](#)
- [aggregateMatching, 35](#)
- [brick \(brick-package\), 4](#)
- [brick-package, 4](#)
- [brick.file, 36](#)
- [checkGamsSuccess, 36](#)
- [checkStockBal, 37](#)
- [computeShareRen, 38](#)
- [copyGamsFiles, 39](#)
- [copyHistoryGdx, 39](#)
- [copyInitialGdx, 40](#)
- [createCalibrationTarget, 40](#)
- [createInputData, 41](#)
- [createMatchingData, 42](#)
- [createParameters, 42](#)
- [createRunFolder, 43](#)
- [createSets, 44](#)
- [expandSets, 44](#)
- [findLastRun, 45](#)
- [getBrickMapping, 45](#)
- [getDimMap, 46](#)

initModel, [46](#)
insertGamsCode, [48](#)
insertMatchingCode, [49](#)
isSlurmAvailable, [49](#)

listToDf, [50](#)
loadCalibrationTarget, [50](#)
loadMadratData, [51](#)

makeHandle, [51](#)

periodFromConfig, [52](#)
plotRefDeviation, [52](#)
plotSummary, [53](#)

readConfig, [53](#)
readInput, [54](#)
readSymbol, [55](#)
reportMif, [55](#)
reweightMatchingReferences, [56](#)
runCalibration, [56](#)
runCalibrationLogit, [57](#)
runCalibrationOptim, [58](#)
runGams, [59](#)

setSlurmConfig, [60](#)
startModel, [60](#)

toModelResolution, [61](#)