Package: brick (via r-universe)

August 30, 2024

Type Package

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

Version 0.5.3

Date 2024-08-30

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 of 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 dplyr, ggplot2, gamstransfer (>= 3.0.1), gms (>= 0.26.0), piamutils (>= 0.0.10), pkgload, purrr, quitte, reportbrick (>= 0.5.0), scales, tidyr, withr, yaml

Suggests covr, devtools, testthat

Encoding UTF-8

RoxygenNote 7.3.2

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

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

RemoteRef HEAD

RemoteSha 3a1ce4aeb2db5dc1677ebe0307823b178bfad6ea

Contents

`brick-package` *brick: Building sector model with heterogeneous renovation and construction of the stock*

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 of 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:

- Ricarda Rosemann <ricarda.rosemann@pik-potsdam.de> (ORCID)

See Also

Useful links:

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

`.cropParamsToThist` *crop gdx parameters to historic periods*

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

`gdx`	character, file path to gdx
`thist`	numeric vector of historic periods

.filterLevel *filter rows with specified entry in column*

Description

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

Usage

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

Arguments

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

Value

data.frame with selected rows without lvlCol

.findCfg *Find config file path*

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

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

Value

file path to config

.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

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

Value

file path to origin gdx file

.onLoad *Check if gamstransfer is available*

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 contiunous integration (CI) i.a. GitHub actions and during `devtools:check` which is called by `lucode2::buildLibrary`.

Usage

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

Arguments

`libname`	not used
`pkgname`	not used

Author(s)

Robin Hasse

.overwriteList *Overwrite list with another list*

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

x	named list, provides the structure and default values that can be overwritten by y.
y	named list, overwrites x wherever it has values different from NULL. Cannot have keys that are not specified in x.
isFinalCfg	logical, is this the final config and not an intermediate?
defaultCfgPath	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.

.readCfg *Read config file*

Description

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

Usage

```
.readCfg(file)
```

Arguments

file	character, path to config file

Value

named list with config parameters

`addAssump` *Add assumed intangible costs*

Description

Add assumed intangible costs

Usage

```
addAssump(df, assumpFile)
```

Arguments

`df`	data.frame for the cost of construction or renovation
`assumpFile`	character, file path to assumption file

Value

data frame with added intangible cost

Author(s)

Robin Hasse

`aggregateMatching` *Aggregate matching run results for calibration*

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` *Find the full file names of files in BRICK*

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` *Check whether Gams finished successfully*

Description

Check which output file was written and derive state of Gams run

Usage

```
checkGamsSuccess(path)
```

Arguments

path	character, path to search for gams output files

Details

If output.gdx was written, give success message. If abort.gdx was written or no output file exists, stop with error message.

Author(s)

Ricarda Rosemann

---

copyGamsFiles *Copy gams scripts to output folder*

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 *Copy history gdx to output folder*

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 *Copy initial gdx to output folder*

Description

Copy initial gdx to output folder

Usage

```
copyInitialGdx(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?

Author(s)

Robin Hasse

createInputData *Create input data*

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 *Create data for reference matching*

Description

Create data for reference matching

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 *Create parameters*

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 *Create new run folder*

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
config	list with run configuration
overwrite	logical; Should exiting folders be overwritten?
recursive	logical; Should exiting folders be overwritten?
showWarnings	logical; Should exiting folders be overwritten?

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 *Determine the latest path of a run*

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 *Retrieve mapping file from BRICK*

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

guessColnames *guess column names based on column values*

Description

guess column names based on column values

Usage

```
guessColnames(x, m)
```

Arguments

x	data.frame with unknown column names
m	gams Conatiner with sets

Value

data.frame with guessed column names

Author(s)

Robin Hasse

`initModel` *Initialize the model:*

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,
  sendToSlurm = NULL,
  slurmQOS = NULL,
  tasksPerNode = 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 input data, • `"createMatching"` to either recreate the matching data or reaggregate the matching

- "none" (or any other string) to do none of the above

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
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

---

isSlurmAvailable *Is Slurm Avaiable*

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 *Convert nested named list to long data.frame*

Description

Convert nested named list to long data.frame

Usage

```
listToDf(x, n = 1)
```

Arguments

x	named list with identical depth on each branch
n	Integer, number of nested function calls, leave default

Value

data.frame with column for each level of the named list

Author(s)

Robin Hasse

loadMadratData *Load Input data from mredgebuildings*

Description

Load Input data from mredgebuildings

Usage

```
loadMadratData(config)
```

Arguments

config	named list with run configuration

Value

directory of input folder

Author(s)

Robin Hasse

makeHandle *Named list to handle string*

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 *get Period from config*

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 *Plot heat map of reference deviation*

Description

Plot heat map of reference deviation

Usage

```
plotRefDeviation(path)
```

Arguments

path	character; directory of output folder

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 *Read symbol from gams container*

Description

Read symbol from gams container

Usage

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

Arguments

x	gams Container, Parameter, Variable or Set
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 *Create mif file for model run*

Description

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

Usage

```
reportMif(path, file = NULL, tmpl = 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.
tmpl	character, BRICK reporting template. There has to be a brickSets mapping named with the same suffix: brickSets_<tmpl>.yaml. If NNULL, the config setting is used.

Author(s)

Robin Hasse

runGams *Run gams optimisation*

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 *Set the SLURM configuration*

Description

Based on user input, construct the SLURM configuration string.

Usage

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

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

Details

Check if SLURM configuration is admissible.

Value

string with SLURM configuration

Author(s)

Ricarda Rosemann

startModel *Start the model*

Description

Run the model with given configuration.

Usage

```
startModel(path)
```

Arguments

path	character vector with folders to run the model in

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 *interpolate and filter to get model resolution*

Description

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

Usage

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

Arguments

x	data.frame with temporal dimension
m	gams Container with sets as known dimensions
value	character, name of value column

Value

data.frame with temporal resolution according to model

Author(s)

Robin Hasse

Index