Package 'simulateDCE'

July 23, 2025

Title Simulate Data for Discrete Choice Experiments

Version 0.3.0

Description
Supports simulating choice experiment data for given designs. It helps to quickly test different designs against each other and compare the performance of new models. The goal of 'simulateDCE' is to make it easy to simulate choice experiment datasets using designs from 'NGENE', 'idefix' or 'spdesign'. You have to store the design file(s) in a sub-directory and need to specify certain parameters and the utility functions for the data generating process. For more details on choice experiments see Mariel et al. (2021) <doi:10.1007/978-3-030-62669-3>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr (>= 1.1.4), evd, formula.tools, ggplot2, kableExtra, magrittr, mixl, psych, purrr, readr, rmarkdown, stats, utils, stringr, tibble, tictoc, tidyr, future, furrr, qs, data.table

Suggests knitr, testthat (>= 3.0.0), rlang

Config/testthat/edition 3

Depends R (>= 4.1.0)

VignetteBuilder knitr

NeedsCompilation no

Author Julian Sagebiel [aut, cre] (ORCID: <https://orcid.org/0000-0002-0253-6875>)

Maintainer Julian Sagebiel <julian.sagebiel@idiv.de>

Repository CRAN

Date/Publication 2025-07-09 10:30:02 UTC

Contents

`aggregateResults` *Aggregate Simulation Results*

Description

Processes the simulation results to extract summaries, coefficients, and graphs.

Usage

```
aggregateResults(all_designs, fromfolder = NULL)
```

Arguments

`all_designs`	A list of simulation results from sim_choice. Can contain different designs but need to have the common structure returned by simchoice
`fromfolder`	A folder from where to read simulations. If provided, the function will read all .qs files from the folder and process them. The files are usually saved by your earlier work and should be qs files as they are more efficient that rds files.

Value

A list with aggregated results including summary, coefficients, graphs, and power.

`createDataset` *Create a Dataset for Choice Experiment Analysis*

Description

This function takes a design matrix and generates a dataset for use in choice experiments. It handles blocks, replicates the design for the number of respondents, and assigns respondent IDs.

Usage

```
createDataset(design, respondents)
```

Arguments

`design`	A data frame containing the design matrix for the choice experiment. It should include at least the columns `Choice.situation` and optionally `Block`.
`respondents`	The number of respondents to generate data for.

Details

The function performs the following steps:

- Checks if the `Block` column exists in the input design. If absent, it creates a single block.
- Calculates the number of choice sets and blocks, and determines the number of sets per block.
- Replicates the design to account for the specified number of respondents per block.
- Assigns respondent IDs based on the number of respondents and blocks.

Value

A data frame containing the augmented design matrix with additional columns:

ID A unique identifier for each respondent.

Choice.situation The original choice situations, replicated for respondents.

Other columns All original columns in the input `design` are retained.

Examples

```
# Example usage:
design <- data.frame(
  Choice.situation = rep(1:12),
  Attribute1 = rnorm(12),
  Attribute2 = sample(1:3, 12, replace = TRUE)
)
result <- createDataset(design, 10)
```

`extract_b_values` *Title Extracts beta values from an spdesign object*

Description

Title Extracts beta values from an spdesign object

Usage

```
extract_b_values(input_list)
```

Arguments

`input_list`	the list where the parameters are stored. Usually this is `design$utility`

Value

A named list with parameter values which can be used in `sim_all`

Examples

```
d <- system.file("extdata", "CSA", "linear", "BLIeff.RDS", package = "simulateDCE")
extract_b_values(readRDS(d)$utility)
```

readdesign *Creates a dataframe with the design.*

Description

Creates a dataframe with the design.

Usage

```
readdesign(design = designfile, designtype = NULL, destype = NULL)
```

Arguments

design	The path to a design file
designtype	Is it a design created with ngene, spdesign or idefix. use 'ngene', 'spdesign' or 'idefix. Ngene designs should be stored as the standard .ngd output. spdesign should be the spdesign object stored as an RDS file. Idefix objects should also be stored as an RDS file. If designtype is not specified, I try to guess what it is. This is especially helpful if you want to carry out a simulation for both spdesign designs and ngene designs at the same time.
destype	Deprecated. Use designtype instead.

Value

a dataframe

Examples

```
library(simulateDCE)
mydesign <- readdesign(
  system.file("extdata", "agora", "altscf_eff.ngd", package = "simulateDCE"),
  "ngene"
)

print(mydesign)
```

simulate_choices *Simulate choices based on a data.frame with a design and respondents*

Description

Simulate choices based on a data.frame with a design and respondents

Usage

```
simulate_choices(
  data,
  utility,
  setspp,
  bcoeff,
  decisiongroups = c(0, 1),
  manipulations = list(),
  estimate,
  preprocess_function = NULL
)
```

Arguments

data	a dataframe that includes a design repeated for the number of observations
utility	a list with the utility functions, one utility function for each alternatives
setspp	an integer, the number of choice sets per person
bcoeff	List of initial coefficients for the utility function. List content/length can vary based on application. I ideally begins (but does not have to) with b and need be the same as those entered in the utility functions
decisiongroups	A vector showing how decision groups are numerically distributed
manipulations	A variable to alter terms of the utility functions examples may be applying a factor or applying changes to terms selectively for different groups
estimate	If TRUE models will be estimated. If false only a dataset will be simulated. Default is true
preprocess_function	= NULL You can supply a function that reads in external data (e.g. GIS coordinates) that will be merged with the simulated dataset. Make sure the the function outputs a data.frame that has a variable called ID which is used for matching.

Value

a data.frame that includes simulated choices and a design

Examples

```
example_df <- data.frame(
  ID = rep(1:100, each = 4),
  price = rep(c(10, 10, 20, 20), 100),
  quality = rep(c(1, 2, 1, 2), 100)
)

beta <- list(
  bprice   = -0.2,
  bquality =  0.8
)

ut <- list(
```

```
  u1 = list(
    v1 = V.1 ~ bprice * price + bquality * quality,
    v2 = V.2 ~ 0
  )
)
simulate_choices(example_df, ut, setspp = 4, bcoeff = beta, estimate = FALSE)
```

`sim_all` *Is a wrapper for sim_choice executing the simulation over all designs stored in a specific folder update*

Description

Is a wrapper for sim_choice executing the simulation over all designs stored in a specific folder update

Usage

```
sim_all(
  nosim = 2,
  resps,
  designtype = NULL,
  destype = NULL,
  designpath,
  u,
  bcoeff,
  decisiongroups = c(0, 1),
  manipulations = list(),
  estimate = TRUE,
  chunks = 1,
  utility_transform_type = "simple",
  reshape_type = "auto",
  mode = c("parallel", "sequential"),
  preprocess_function = NULL,
  savefile = NULL
)
```

Arguments

`nosim`	Number of runs or simulations. For testing use 2 but once you go serious, use at least 200, for better results use 2000.
`resps`	Number of respondents you want to simulate
`designtype`	Is it a design created with ngene, spdesign or idefix. use 'ngene', 'spdesign' or 'idefix. Ngene designs should be stored as the standard .ngd output. spdesign should be the spdesign object stored as an RDS file. Idefix objects should also be stored as an RDS file. If designtype is not specified, I try to guess what it is.

This is especially helpful if you want to carry out a simulation for both spdesign designs and ngene designs at the same time.

`destype` Deprecated. Use designtype instead.

`designpath` The path to the folder where the designs are stored. For example "c:/myfancydec/Designs"

`u` A list with utility functions. The list can incorporate as many decision rule groups as you want. However, each group must be in a list in this list. If you just use one group (the normal), this group still has to be in a list in the u list. As a convention name beta coefficients starting with a lower case "b"

`bcoeff` List of initial coefficients for the utility function. List content/length can vary based on application. I ideally begins (but does not have to) with b and need be the same as those entered in the utility functions

`decisiongroups` A vector showing how decision groups are numerically distributed

`manipulations` A variable to alter terms of the utility functions examples may be applying a factor or applying changes to terms selectively for different groups

`estimate` If TRUE models will be estimated. If false only a dataset will be simulated. Default is true

`chunks` The number of chunks determines how often results should be stored on disk as a safety measure to not loose simulations if models have already been estimated. For example, if no_sim is 100 and chunks = 2, the data will be saved on disk after 50 and after 100 runs.

`utility_transform_type`
How the utility function you entered is transformed to the utility function required for mixl. You can use the classic way (simple) where parameters have to start with "b" and variables with "alt" or the more flexible (but potentially error prone) way (exact) where parameters and variables are matched exactly what how the are called in the dataset and in the bcoeff list. Default is "simple". In the long run, simple will be deleted, as exact should be downwards compatible.

`reshape_type` Must be "auto", "stats" to use the reshape from the stats package or tidyr to use pivot longer. Default is auto and should not bother you. Only change it once you face an error at this position and you may be lucky that it works then.

`mode` Set to "parallel" if parts should be run in parallel mode

`preprocess_function`
= NULL You can supply a function that reads in external data (e.g. GIS coordinates) that will be merged with the simulated dataset. Make sure the the function outputs a data.frame that has a variable called ID which is used for matching.

`savefile` Indicate a path if you want to store the results after each design simulation locally. This is useful in case you fear that your computer crashes

Value

A list, with all information on the simulation. This list an be easily processed by the user and in the rmarkdown template.

Examples

```
library(rlang)
designpath <- system.file("extdata", "SE_DRIVE", package = "simulateDCE")
resps <- 120 # number of respondents
nosim <- 2 # number of simulations to run (about 500 is minimum)

decisiongroups <- c(0, 0.7, 1)

# pass beta coefficients as a list
bcoeff <- list(
  b.preis = -0.01,
  b.lade = -0.07,
  b.warte = 0.02
)

manipulations <- list(
  alt1.x2 = expr(alt1.x2 / 10),
  alt1.x3 = expr(alt1.x3 / 10),
  alt2.x2 = expr(alt2.x2 / 10),
  alt2.x3 = expr(alt2.x3 / 10)
)


# place your utility functions here
ul <- list(
  u1 =

    list(
      v1 = V.1 ~ b.preis * alt1.x1 + b.lade * alt1.x2 + b.warte * alt1.x3,
      v2 = V.2 ~ b.preis * alt2.x1 + b.lade * alt2.x2 + b.warte * alt2.x3
    ),
  u2 = list(
    v1 = V.1 ~ b.preis * alt1.x1,
    v2 = V.2 ~ b.preis * alt2.x1
  )
)


sedrive <- sim_all(
  nosim = nosim,
  resps = resps,
  designpath = designpath,
  u = ul,
  bcoeff = bcoeff,
  decisiongroups = decisiongroups,
  manipulations = manipulations,
  utility_transform_type = "exact",
  mode = "sequential",
  estimate=FALSE
)
```

`sim_choice` *Simulate and estimate choices*

Description

Simulate and estimate choices

Usage

```
sim_choice(
  designfile,
  no_sim = 10,
  respondents = 330,
  u,
  designtype = NULL,
  destype = NULL,
  bcoeff,
  decisiongroups = c(0, 1),
  manipulations = list(),
  estimate,
  chunks = 1,
  utility_transform_type = "simple",
  mode = c("parallel", "sequential"),
  preprocess_function = NULL,
  savefile = NULL
)
```

Arguments

`designfile`	path to a file containing a design.
`no_sim`	Number of runs i.e. how often do you want the simulation to be repeated
`respondents`	Number of respondents. How many respondents do you want to simulate in each run.
`u`	A list with utility functions. The list can incorporate as many decision rule groups as you want. However, each group must be in a list in this list. If you just use one group (the normal), this group still has to be in a list in the u list. As a convention name beta coefficients starting with a lower case "b"
`designtype`	Is it a design created with ngene, spdesign or idefix. use 'ngene', 'spdesign' or 'idefix. Ngene designs should be stored as the standard .ngd output. spdesign should be the spdesign object stored as an RDS file. Idefix objects should also be stored as an RDS file. If designtype is not specified, I try to guess what it is. This is especially helpful if you want to carry out a simulation for both spdesign designs and ngene designs at the same time.
`destype`	Deprecated. Use designtype instead.

bcoeff — List of initial coefficients for the utility function. List content/length can vary based on application. I ideally begins (but does not have to) with b and need be the same as those entered in the utility functions

decisiongroups — A vector showing how decision groups are numerically distributed

manipulations — A variable to alter terms of the utility functions examples may be applying a factor or applying changes to terms selectively for different groups

estimate — If TRUE models will be estimated. If false only a dataset will be simulated. Default is true

chunks — The number of chunks determines how often results should be stored on disk as a safety measure to not loose simulations if models have already been estimated. For example, if no_sim is 100 and chunks = 2, the data will be saved on disk after 50 and after 100 runs.

utility_transform_type — How the utility function you entered is transformed to the utility function required for mixl. You can use the classic way (simple) where parameters have to start with "b" and variables with "alt" or the more flexible (but potentially error prone) way (exact) where parameters and variables are matched exactly what how the are called in the dataset and in the bcoeff list. Default is "simple". In the long run, simple will be deleted, as exact should be downwards compatible.

mode — Set to "parallel" if parts should be run in parallel mode

preprocess_function — = NULL You can supply a function that reads in external data (e.g. GIS coordinates) that will be merged with the simulated dataset. Make sure the the function outputs a data.frame that has a variable called ID which is used for matching.

savefile — Indicate a path if you want to store the results after each design simulation locally. This is useful in case you fear that your computer crashes

Value

a list with all information on the run

Examples

```
bcoeff <- list(
  basc = -1.2,
  basc2 = -1.4,
  baction = 0.1,
  badvisory = 0.4,
  bpartnertest = 0.3,
  bcomp = 0.02
)
ul <- list(
  u1 =
    list(
     #' # model specification ----------------------------------------------
v1 <- V.1 ~ basc +
  baction      * alt1.b +
  badvisory    * alt1.c +
```

```
  bpartnertest * alt1.d +
  bcomp        * alt1.p,

v2 <- V.2 ~ basc2 +
  baction      * alt2.b +
  badvisory    * alt2.c +
  bpartnertest * alt2.d +
  bcomp        * alt2.p,
 v3 <- V.3 ~ 0
    )
)

sim_choice(
  designfile = system.file("extdata", "agora", "altscf_eff.ngd", package = "simulateDCE"),
  no_sim = 2,
  respondents = 144,
  u = ul,
  bcoeff = bcoeff,
  estimate = FALSE
)
```

Index