Package ‘mixl’

March 21, 2020

Type Package

Title Simulated Maximum Likelihood Estimation of Mixed Logit Models for Large Datasets

Version 1.1.2

Date 2020-03-20

Description Specification and estimation of multinomial logit models. Large datasets and complex models are supported, with an intuitive syntax. Multinomial Logit Models, Mixed models, random coefficients and Hybrid Choice are all supported. For more information, see Molloy et al. (2019) <doi:10.3929/ethz-b-000334289>.

License GPL (>= 2)

Imports maxLik, numDeriv, randtoolbox, Rcpp (>= 0.12.19), readr, sandwich, stats, stringr (>= 1.3.1)

Suggests knitr, mlogit, rmarkdown, testthat, texreg, xtable

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Author Joseph Molloy [aut, cre]

Maintainer Joseph Molloy <joseph.molloy@ivt.baug.ethz.ch>

Repository CRAN

Date/Publication 2020-03-21 16:30:02 UTC

R topics documented:

mixl-package ................................................................. 2
av_matrix ................................................................. 3
check_inputs ............................................................. 4
compileUtilityFunction .................................................. 4
create_halton_draws ...................................................... 5
estimate ................................................................. 5
mixl-package

Estimate mixed multinomial logit models

Description

Estimate mixed multinomial logit models using (simulated) maximum likelihood estimation. The package supports standard mnl, mixed-logit and hybrid choice. Using compilation to C++, model estimation is significantly faster than in native R code.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Joe Molloy <joseph.molloy@ivt.baug.ethz.ch>.

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

data("Train", package="mlogit")
head(Train, 3)
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
mnl_test <- "

av_matrix

// ASC_B_RND = @ASC_B + draw_2 * @SIGMA_B;

U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60 + @B_change * $change_A;
U_B = @ASC_B + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;

model_spec <- mixl::specify_model(mnl_test, Train)

# only take starting values that are needed
est <- stats::setNames(c(0,0,0,0,0), c("B_price", "B_time", "B_timeB", "B_change", "ASC_B"))

availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities, nDraws = 20)

summary(model)

---

**av_matrix**

Extract the availabilites matrix from the dataset, using column indicies

---

**Description**

Extract the availabilites matrix from the dataset, using column indicies

**Usage**

```r
av_matrix(data, av_cols)
```

**Arguments**

- `data`: The dataset used in the model
- `av_cols`: A vector of the the column indicies of the availabilities for each alternative

**Value**

Matrix of availabilities for alternatives and the number of choice observations

**Examples**

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
Train$avail_A <- sample(2, replace=TRUE, size=nrow(Train))-1
Train$avail_B <- sample(2, replace=TRUE, size=nrow(Train))-1
av_matrix(Train, c('avail_A', 'avail_B'))
```
check_inputs  

**Description**

This function checks the start_values, data, availabilities, draws and fixedparams for validity. If this function runs without error, then the inputs are valid for the maxLikelihood function. These checks are important, because an error in the internal C++ code will cause the Rstudio session to crash. Incidentally, if there is concern of this happening, it is recommended to run the script from the command line, using Rscript.

**Usage**

```r
check_inputs(model_spec, start_values, data, availabilities, draws, fixedparam)
```

**Arguments**

- `model_spec`: The specified Model
- `start_values`: Named vector of proposed start values for the model
- `data`: the dataset on which to estimate
- `availabilities`: The availabilities for the alternatives in the model specification
- `draws`: The matrix of random draws
- `fixedparam`: Named vector of parameters to be fixed

**Value**

Nothing

---

**compileUtilityFunction**

*compileUtilityFunction Deprecated, please see specify_model()*

**Description**

compileUtilityFunction Deprecated, please see specify_model()

**Usage**

```r
compileUtilityFunction(...)```

**Arguments**

- `...`: Parameters to specify_model
create_halton_draws

Create a standard set of Halton draws to use in estimation

Description
Create a standard set of Halton draws to use in estimation

Usage
create_halton_draws(Nindividuals, nDraws, draw_dimensions)

Arguments
- Nindividuals: The number individuals in the dataset
- nDraws: The number of draws needed
- draw_dimensions: the number of draw dimensions needed

Value
Matrix of availabilities for alternatives and the number of choice observations

Examples
create_halton_draws(100, 10, 5)
create_halton_draws(100, 100, 20)

estimate
Runs a maximum likelihood estimation on a mixl choice model

Description
This function performs a maximum likelihood estimation for choice models specified using this package.

Usage
estimate(
  model_spec, start_values, data, availabilities, draws = NULL, nDraws = NULL, fixedparam = c(),
num_threads = 1,
...
)

Arguments

model_spec The object that contains the loglikelihood function and other variables that help return better error messages. This function is best generated using the `specify_model()` function.

start_values A named vector of start values for the estimation. A warning and error will be given respectively if to many values are included or some are missing.

data A dataframe of the observations. It must include the columns CHOICE and ID, as well as columns for the variables specified in the utility function. The CHOICE variable must be from 1..k, where k is the number of utility functions.

availabilities A 1/0 matrix of availabilities. The dimensions must be `nrows(data) * k`, where there are k utility functions.

draws A numeric matrix of draws for calculating mixed effects. If there no mixed effects, this should be left null. If the model specification included mixed effects, either this or `nDraws` need to be specified.

nDraws The number of draws to use in estimating a mixed model. Only needed if `draws` is left null. Then a matrix of normal halton draws will be generated.

fixedparam (optional) Coefficients which should be fixed to their starting values during estimation.

num_threads The maximum number of parallel cores to use in estimation. The default is 1. This should only be specified on machines with an openMP compiler (linux and some OSXs).

... further arguments. Such as control are passed to the maximisaiton routine in `maxLik`. See `maxLik::maxLik()` for more details.

Details

It is a wrapper for the `maxLik` function in the `maxLik` package. And additional arguments can be passed through to this function if required.

Value

A `mixl` object that contains the results of the estimation.

Examples

data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- "
U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;"
```

model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1,1,1), c("asc", "B_price", "B_time", "B_timeB"))
availabilities <- mixl::generate_default_availabilities(
    Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
print(model)
```

---

**extract_av_cols**

Extract the availabilities matrix from the dataset using a column name prefix

**Description**

Extract the availabilities matrix from the dataset using a column name prefix

**Usage**

```
extract_av_cols(data, prefix)
```

**Arguments**

- **data**: The dataset used in the model
- **prefix**: The prefix of the availability columns, i.e. `avail_`

**Value**

Matrix of availabilities for alternatives and the number of choice observations

**Examples**

```
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
Train$avail_A <- sample(2, replace=TRUE, size=nrow(Train))-1
Train$avail_B <- sample(2, replace=TRUE, size=nrow(Train))-1
extract_av_cols(Train, 'avail_')
```
extract_indiv_data  
Extract the individual level data from the dataset for use in posterior analysis

Description
Extract the individual level data from the dataset for use in posterior analysis

Usage
extract_indiv_data(data, data_cols = NULL)

Arguments

- **data**: The dataset
- **data_cols**: The individual level columns of attributes - Can be null to take aggregate for each column

Value
dataframe of all individual level data for each ID

Examples
```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
# in this case not actually individual data columns
# an ID column is required here
extract_indiv_data(Train, c('comfort_A', 'comfort_B'))
```

generate_default_availabilities  
Generate a ones-matrix of availabilities

Description
Generate a ones-matrix of availabilities

Usage
generate_default_availabilities(data, num_utility_functions)
**Arguments**

- **data**: The dataset used in the model
- **num_utility_functions**: the number of alternatives in the model

**Value**

Ones-matrix of availabilities for alternatives and the number of choice observations

**Examples**

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
generate_default_availabilities(Train, 5)
```

---

**Description**

Calculate the posteriors for a specified and estimated model

**Usage**

`posteriors(model, indiv_data = NULL, code_output_file = NULL)`

**Arguments**

- **model**: The estimated Model
- **indiv_data**: Alternative individual data to use instead of that in the dataset
- **code_output_file**: An (optional) location where the compiled code should be saved (useful for debugging)

**Value**

Dataframe of individual-level posteriors

**Examples**

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)
mnl_test <- "
  ASC_A_RND = @ASC_A + draw_1 * @SIGMA_A1 + draw_7 * @SIGMA_A2;
  ASC_B_RND = @ASC_B + draw_2 * @SIGMA_B;
"`
U_A = ASC_A_RND + @B_price * $price_A / 1000
    + @B_time * $time_A / 60 + @B_change * $change_A;
U_B = ASC_B_RND + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;

#only take starting values that are needed
est <- stats::setNames(c(-1059.69729, -181.27796, -251.78909,
                        -241.18878, -86.77386, -173.09451,
                        291.02618, 142.71793, 332.60909),
                        c("B_price", "B_time", "B_timeB", "B_change",
                        "ASC_A", "ASC_B", "SIGMA_A1", "SIGMA_A2", "SIGMA_B"))

availabilities <- generate_default_availabilities(Train, 2)

model_specification <- specify_model(mnl_test, Train, disable_multicore=T)
model <- estimate(model_specification, est, Train,
                  availabilities = availabilities, nDraws = 1)

posteriors(model)

print.mixl

Prints the output of a model

Description

print() is an S3 method for the mixl class. It creates a model summary and then prints the result

Usage

## S3 method for class 'mixl'
print(x, ...)

Arguments

x         The model to print
...

Options to pass to print

Examples

data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- 
U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;
model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1, 1), c("asc", "B_price", "B_time", "B_timeB"))
availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
summary(model)

print.summary.mixl

Print a model summary

Description

print() is an S3 method for the summary.mixl class, the output of a model plus goodness of fit metrics

Usage

## S3 method for class 'summary.mixl'
print(x, ...)

Arguments

x

The summary to print.

...

Options to pass to print.

Examples

data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- "
  U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
  U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;
"

model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1, 1), c("asc", "B_price", "B_time", "B_timeB"))
availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
summary(model)
Train, model_spec$num_utility_functions
)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
summary(model)

probabilities

Calculate the probabilities for a specified and estimated model. Note that if new data or draws are provided, the model will not be re-estimated

Description

Calculate the probabilities for a specified and estimated model. Note that if new data or draws are provided, the model will not be re-estimated

Usage

probabilities(
  model,
  data = NULL,
  availabilities = NULL,
  draws = NULL,
  nDraws = NULL,
  num_threads = 1
)

Arguments

model The estimated Model
data (Optional) New data to use instead of that in the dataset
availabilities (Optional) New availabilites to use
draws (Optional) Optional new set of random draws to use
nDraws (Optional) Optional new number of random draws to use
num_threads Enable parallel computing where available using this many cores

Value

Dataframe of individual-level posteriors
Examples

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- 
  "U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
  U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;"

model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1, 1, 1), c("asc", "B_price", "B_time", "B_timeB"))
availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
probabilities(model)

# hypothetical scenario where the travel time of option A doubles
Train$time_A = Train$time_A * 2
probabilities(model, Train)
```

**specify_model**

_validate the utility functions against the dataset and generate the optimised log likelihood function_

**Description**

This function takes a utility function description, and generates a optimised C++ version of the utility function which can be called from R. If the data_names are provided, then the variables in the function are checked against those provided. If an output_file is provided, the C++ code is saved there. See the user guide vignette for how to write valid utility scripts. There is some minimal specific syntax required.

**Usage**

```r
specify_model(
  utility_script,
  dataset = NULL,
  output_file = NULL,
  compile = TRUE,
  model_name = "mixl_model",
  disable_multicore = F
)
```
### Arguments

- **utility_script**: The utility script to be compiled
- **dataset**: An (optional) dataframe to check if the all the variables are present
- **output_file**: An (optional) location where the compiled code should be saved (useful for debugging)
- **compile**: If compile is false, then the code will not be compiled, but just validated and saved if an output file is specified
- **model_name**: A name for the model, which will be used for saving. Defaults to `mixl_model`
- **disable_multicore**: True to disable openMP parallelism if openMP isn’t installed (default on OSX - see the user guide for how to enable it)

### Value

An object which contains the loglikelihood function, and information from the compile process

### See Also

- `browseVignettes("mixl")`

### Examples

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- 
U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;

model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1, 1), c("asc", "B-price", "B-time", "B-timeB"))
availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
print(model)
```
Create a model summary

Description

`summary()` is an S3 method for the class `mixl`, which adds metrics of goodness of fit.

Usage

```r
## S3 method for class 'mixl'
summary(object, ...)  
```

Arguments

- `object`: The `mixl` output to summarize.
- `...`: Options to pass to `summarize` (currently).

Value

A summary object for a `mixl` model.

Examples

```r
data("Train", package="mlogit")
Train$ID <- Train$id
Train$CHOICE <- as.numeric(Train$choice)

mnl_test <- "
U_A = @B_price * $price_A / 1000 + @B_time * $time_A / 60;
U_B = @asc + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;
"

model_spec <- mixl::specify_model(mnl_test, Train, disable_multicore=T)

# only take starting values that are needed
est <- stats::setNames(c(1, 1, 1, 1), c("asc", "B_price", "B_time", "B_timeB"))
availabilities <- mixl::generate_default_availabilities(Train, model_spec$num_utility_functions)

model <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
summary(model)
```
summary_tex

Return tex formatted output of a model summary

Description

Return tex formatted output of a model summary

Usage

summary_tex(model_summary)

Arguments

model_summary  A summary of an estimated Model

Value

Formatted text output suitable for a research paper.
Index

*Topic package
    mixl-package, 2

av_matrix, 3
check_inputs, 4
compileUtilityFunction, 4
create_halton_draws, 5
estimate, 5
extract_av_cols, 7
extract_indiv_data, 8

generate_default_availabilities, 8
maxLik::maxLik(), 6
mixl(mixl-package), 2
mixl-package, 2

posteriors, 9
print(), 10, 11
print.mixl, 10
print.summary.mixl, 11
probabilities, 12

specify_model, 13
specify_model(), 4, 6
summary(), 15
summary.mixl, 15
summary_tex, 16