# Package ‘mixl’

February 7, 2024

**Type**  Package

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

**Version**  1.3.4

**Date**  2024-02-06

**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. (2021) <https://www.research-collection.ethz.ch/handle/20.500.11850/477416>.

**License**  GPL (>= 2)

**URL**  https://github.com/joemolloy/fast-mixed-mnl

**BugReports**  https://github.com/joemolloy/fast-mixed-mnl/issues

**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**  7.3.1

**NeedsCompilation**  yes

**Author**  Joseph Molloy [aut, cre]

**Maintainer**  Joseph Molloy <joe.m34@gmail.com>

**Repository**  CRAN

**Date/Publication**  2024-02-07 12:10:02 UTC
R topics documented:

mixl-package ................................................................. 2
av_matrix ................................................................. 3
check_draw_inputs ......................................................... 4
check_inputs ............................................................. 4
compileUtilityFunction .................................................. 5
create_halton_draws ....................................................... 6
estimate ................................................................. 6
extract_av_cols .......................................................... 8
extract_indiv_data ....................................................... 9
generate_default_availabilities ..................................... 9
posteriors ............................................................... 10
print.mixl .............................................................. 11
print.summary.mixl ...................................................... 12
probabilities ............................................................ 13
specify_model ........................................................... 14
summary.mixl ............................................................ 16
summary_tex ............................................................ 17
utilities ................................................................. 17
vcov.mixl .............................................................. 18

Index 20

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@iwt.baug.ethz.ch>.

References

### Examples

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

mnl_test <-
  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_RND + @B_price * $price_B / 1000 + @B_timeB * $time_B / 60;
```

```r
model_spec <- mixl::specify_model(mnl_test, Train)
#only take starting values that are needed
est <- stats::setNames(c(0,0,0,0,0,0), c("B_price", "B_time", "B_timeB", "B_change", "ASC_B", "SIGMA_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

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_draw_inputs

Description

Check the inputs to the draw function

Usage

check_draw_inputs(draws, nDraws, draw_dimensions, Nindividuals)

Arguments

draws The specified Model
nDraws Named vector of proposed start values for the model
draw_dimensions the dataset on which to estimate
Nindividuals The availabilities for the alternatives in the model specification

Value

A list consisting of the checked draws and Ndraws, both computed if required)

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,
  weights
)
```

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
- `weights`: The weights vector

Value

Nothing

---

**compileUtilityFunction**

`compileUtilityFunction Deprecated, please see specify_model()`

----

Description

compileUtilityFunction Deprecated, please see `specify_model()`

Usage

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

Arguments

```r
... 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,
  nDraws,
  fixedparam = c(),
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 too 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 are 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 OS Xs).

**weights** (optional) A vector of weights (vector length must equal the number of observations).

... further arguments such as control are passed to the maximisation 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

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

```r
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**

```r
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

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

posteriors Calculate the posteriors for a specified and estimated model

Description

Calculate the posteriors for a specified and estimated model

Usage

posteriors(model, indiv_data, 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

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;"
\[
U_A = ASC_A_{\text{RND}} + \beta_{\text{price}} \cdot \$\text{price}_A / 1000 \\
+ \beta_{\text{time}} \cdot \$\text{time}_A / 60 + \beta_{\text{change}} \cdot \$\text{change}_A; \\
U_B = ASC_B_{\text{RND}} + \beta_{\text{price}} \cdot \$\text{price}_B / 1000 + \beta_{\text{timeB}} \cdot \$\text{time}_B / 60;
\]

# only take starting values that are needed
est <- stats::setNames(c(-1059.69729, -181.27796, -251.78909, 
-241.18878, -36.77386, -173.09451, 
291.02618, 142.71793, 332.60909), 
  c("\text{B\_price}", "\text{B\_time}", "\text{B\_timeB}", "\text{B\_change}", 
    "\text{ASC\_A}", "\text{ASC\_B}", "\text{SIGMA\_A1}", "\text{SIGMA\_A2}", "\text{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)
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)

model2 <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
print(model2)

print.summary.mixl

---

**Description**

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

**Usage**

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

**Arguments**

- `x` The summary to print.
- `...` Options to pass to print.

**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)

model2 <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)

print(model2)
```
Train, model_spec$num_utility_functions

model2 <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
print(model2)

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 availabilities 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 loglikelihood 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 = T,
  ...
)
```
**specify_model**

**Arguments**

utility_script The utility script to be compiled

dataset An (optional) dataframe to check if 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. Default is true.

model_name A name for the model, which will be used for saving. Defaults to `mnl_model`

disable_multicore Deprecated and not used. Multicore is now autodetected

... Further parameters to pass to `sourceCpp`

**Value**

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

**See Also**

`browseVignettes("mnl")`

**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)
summary.mixl

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)

model2 <- mixl::estimate(model_spec, est, Train, availabilities = availabilities)
print(model2)
```
### summary_tex

Return tex formatted output of a model summary. If an `output_file` parameter is provided, save the object to that location.

**Description**

Return tex formatted output of a model summary. If an `output_file` parameter is provided, save the object to that location.

**Usage**

```r
summary_tex(model_summary, output_file)
```

**Arguments**

- `model_summary`: A summary of an estimated Model
- `output_file`: Where to save the tex representation

**Value**

Formatted texreg object containing the latex table suitable for a research paper. See `createTexreg`.

---

### utilities

Return the the utilities for a set of coefficients.

**Description**

Return the the utilities for a set of coefficients.

**Usage**

```r
utilities(model_spec, beta, data, availabilities, draws, nDraws)
```

**Arguments**

- `model_spec`: The generated model_spec.
- `beta`: The coefficients to use in the model when estimating the utilities.
- `data`: The dataframe of observations.
- `availabilities`: The availabilities of each alternative.
- `draws`: For mixed models, a matrix of draws. If none is provided, one is created.
- `nDraws`: The number of draws to use or generated.

**Value**

Dataframe of utilities for each observation.
Examples

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

est <- stats::setNames(c(1,1,1,1), c("B_price", "B_time", "B_timeB", "B_change"))

availabilities <- mixl::generate_default_availabilities(Train, 2)

Nindividuals <- length(unique(Train$ID))

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

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

utilities_matrix = mixl::utilities(model_spec, est, Train, availabilities, NULL)

utilities_matrix

vcov.mixl

Calculates the Variance-Covariance Matrix of the mixl summary

Description

vcov() is an S3 method for the summary.mixl class, giving the Variance-Covariance Matrix

Usage

## S3 method for class 'mixl'
vcov(object, eigentol = 1e-12, ...)

Arguments

object The summary to print.
eigentol The tolerance value.
... 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, 1,1,1), c("asc", "B_price", "B_time", "B_timeB"))
availabilitys <- mixl::generate_default_availablesabilities(Train, model_spec$num_utility_functions)

model2 <- mixl::estimate(model_spec, est, Train, availables = availabilitys)
print(model2)
Index

* Multinomial logit MNL GPU
  mixl-package, 2

av_matrix, 3

check_draw_inputs, 4
check_inputs, 4
compileUtilityFunction, 5
create_halton_draws, 6
createTexreg, 17

estimate, 6
extract_av_cols, 8
extract_indiv_data, 9

generate_default_availabilities, 9

maxLik::maxLik(), 7
mixl (mixl-package), 2
mixl-package, 2

posterior, 10
print(), 11, 12
print.mixl, 11
print.summary.mixl, 12
probabilities, 13

sourceCpp, 15
specify_model, 14
specify_model(), 5, 7
summary(), 16
summary.mixl, 16
summary_tex, 17

utilities, 17

vcov(), 18
vcov.mixl, 18