Package 'logitr'

July 24, 2024

Title Logit Models w/Preference & WTP Space Utility Parameterizations

Version 1.1.2

Description Fast estimation of multinomial (MNL) and mixed logit (MXL) models in R. Models can be estimated using ``Preference'' space or ``Willingness-to-pay'' (WTP) space utility parameterizations. Weighted models can also be estimated. An option is available to run a parallelized multistart optimization loop with random starting points in each iteration, which is useful for non-convex problems like MXL models or models with WTP space utility parameterizations. The main optimization loop uses the 'nloptr' package to minimize the negative log-likelihood function. Additional functions are available for computing and comparing WTP from both preference space and WTP space models and for predicting expected choices and choice probabilities for sets of alternatives based on an estimated model. Mixed logit models can include uncorrelated or correlated heterogeneity covariances and are estimated using maximum simulated likelihood based on the algorithms in Train (2009) <doi:10.1017/CBO9780511805271>. More details can be found in Helveston (2023) <doi:10.18637/jss.v105.i10>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

Depends R (>= 3.5.0)

Suggests apollo, broom, broom.helpers (>= 1.15.0), dplyr, fastDummies, ggplot2, ggrepel, gmnl, gtsummary (>= 2.0.0), here, kableExtra, knitr, mixl, mlogit, rmarkdown, testthat, texreg, tidyr

Imports generics, MASS, nloptr, parallel, randtoolbox, stats, tibble

URL https://github.com/jhelvy/logitr

BugReports https://github.com/jhelvy/logitr/issues

NeedsCompilation no

Author John Helveston [aut, cre, cph] (<https://orcid.org/0000-0002-2657-9191>), Connor Forsythe [ctb]

Maintainer John Helveston <john.helveston@gmail.com>

Repository CRAN

Date/Publication 2024-07-24 11:20:04 UTC

Contents

`apolloModeChoiceData` *Simulated SP dataset of mode choice (from the apollo package).*

Description

A simulated dataset containing 7,000 mode choices among four alternatives. Data comes from 500 individuals, each with 14 stated stated preference (SP) observations. There are 7,000 choices in total. Each observation contains attributes for the alternatives, availability of alternatives, and characteristics of the individuals.

Usage

```
data(apolloModeChoiceData)
```

Format

Variable	Description
ID	individual identifiers
obsID	identifier for unique choice observation
altID	alternative in each choice observation
qID	Numeric. Consecutive ID of SP choice tasks.
choice	dummy code for choice (1 or 0)
mode	Character describing mode: "air", "rail", "car", "bus"
time	Travel time in minutes.
cost	cost (in GBP) of trip.
access	Access time in minutes.
service	Numeric. Additional services: 1 for no-frills, 2 for wifi, 3 for food.
mode_air	Dummy coefficient for "air" mode.
mode_bus	Dummy coefficient for "bus" mode.
mode_car	Dummy coefficient for "car" mode.
mode_rail	Dummy coefficient for "rail" mode.
service_no_frills	Dummy coefficient for "no-frills" additional service.
service_wifi	Dummy coefficient for "wifi" additional service.
service_food	Dummy coefficient for "food" additional service.
time_car	Travel time (in minutes) for car trip.
time_bus	Travel time (in minutes) for bus trip.
time_air	Travel time (in minutes) for air trip.
time_rail	Travel time (in minutes) for rail trip.
female	Numeric. Sex of individual. 1 for female, 0 for male.
business	Numeric. Purpose of the trip. 1 for business, 0 for other.
income	Numeric. Income (in GBP per annum) of the individual.

Source

Data imported from the apollo package archive

References

Hess, S. & Palma, D. (2019), Apollo: a flexible, powerful and customisable freeware package for choice model estimation and application, Journal of Choice Modelling, Volume 32, September 2019. doi:10.1016/j.jocm.2019.100170

Examples

```
data(apolloModeChoiceData)

head(apolloModeChoiceData)
```

augment.logitr *Glance a* logitr *class object*

Description

Glance a logitr class object

Usage

```
## S3 method for class 'logitr'
augment(x, newdata = NULL, obsID = NULL, type = "prob", ...)
```

Arguments

x	is an object of class logitr.
newdata	a data.frame. Each row is an alternative and each column an attribute corresponding to parameter names in the estimated model. Defaults to NULL, in which case predictions are made on the original data used to estimate the model.
obsID	The name of the column that identifies each set of alternatives in the data. Required if newdata != NULL. Defaults to NULL, in which case the value for obsID from the data in object is used.
type	A character vector defining what to predict: prob for probabilities, outcomes for outcomes. If you want both outputs, use c("prob", "outcome"). Outcomes are predicted randomly according to the predicted probabilities. Defaults to "prob".
...	further arguments.

Value

A tibble of ...

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Extract a tibble of the model summary statistics
augment(mnl_pref)
```

cars_china *Stated car choice observations by Chinese car buyers*

Description

Data from Helveston et al. (2015) containing 448 stated choice observations from Chinese car buyers and 384 stated choice observations from US car buyers. Conjoint surveys were fielded in 2012 in four major Chinese cities (Beijing, Shanghai, Shenzhen, and Chengdu), online in the US on Amazon Mechanical Turk, and in person at the Pittsburgh Auto show. Participants were asked to select a vehicle from a set of three alternatives. Each participant answered 15 choice questions.

Usage

```
data(cars_china)
```

Format

Variable	Description
id	individual identifiers
obsnum	identifier for unique choice observation
choice	dummy code for choice (1 or 0)
hev	dummy code for HEV vehicle type (1 or 0)
phev10	dummy code for PHEV vehicle type w/10 mile electric driving range (1 or 0)
phev20	dummy code for PHEV vehicle type w/20 mile electric driving range (1 or 0)
phev40	dummy code for PHEV vehicle type w/40 mile electric driving range (1 or 0)
bev75	dummy code for BEV vehicle type w/75 mile electric driving range (1 or 0)
bev100	dummy code for BEV vehicle type w/100 mile electric driving range (1 or 0)
bev150	dummy code for BEV vehicle type w/150 mile electric driving range (1 or 0)
phevFastcharge	dummy code for whether PHEV vehicle had fast charging capability (1 or 0)
bevFastcharge	dummy code for whether BEV vehicle had fast charging capability (1 or 0)
price	price of vehicle ($USD)
opCost	operating cost of vehicle (US cents / mile)
accelTime	0-60 mph acceleration time (seconds)
american	dummy code for whether American brand (1 or 0)
japanese	dummy code for whether Japanese brand (1 or 0)
chinese	dummy code for whether Chinese brand (1 or 0)
skorean	dummy code for whether S. Korean brand (1 or 0)
weights	weights for each individual computed so that the sample age and income demographics matched with thos

Source

Raw data downloaded from this repo

References

Helveston, J. P., Liu, Y., Feit, E. M., Fuchs, E. R. H., Klampfl, E., & Michalek, J. J. (2015). "Will Subsidies Drive Electric Vehicle Adoption? Measuring Consumer Preferences in the U.S. and China." Transportation Research Part A: Policy and Practice, 73, 96–112. doi:10.1016/j.tra.2015.01.002

Examples

```
data(cars_china)

head(cars_china)
```

---

cars_us *Stated car choice observations by US car buyers*

Description

Data from Helveston et al. (2015) containing 448 stated choice observations from Chinese car buyers and 384 stated choice observations from US car buyers. Conjoint surveys were fielded in 2012 in four major Chinese cities (Beijing, Shanghai, Shenzhen, and Chengdu), online in the US on Amazon Mechanical Turk, and in person at the Pittsburgh Auto show. Participants were asked to select a vehicle from a set of three alternatives. Each participant answered 15 choice questions.

Usage

```
data(cars_us)
```

Format

Variable	Description
`id`	individual identifiers
`obsnum`	identifier for unique choice observation
`choice`	dummy code for choice (`1` or `0`)
`hev`	dummy code for HEV vehicle type (`1` or `0`)
`phev10`	dummy code for PHEV vehicle type w/10 mile electric driving range (`1` or `0`)
`phev20`	dummy code for PHEV vehicle type w/20 mile electric driving range (`1` or `0`)
`phev40`	dummy code for PHEV vehicle type w/40 mile electric driving range (`1` or `0`)
`bev75`	dummy code for BEV vehicle type w/75 mile electric driving range (`1` or `0`)
`bev100`	dummy code for BEV vehicle type w/100 mile electric driving range (`1` or `0`)
`bev150`	dummy code for BEV vehicle type w/150 mile electric driving range (`1` or `0`)
`phevFastcharge`	dummy code for whether PHEV vehicle had fast charging capability (`1` or `0`)
`bevFastcharge`	dummy code for whether BEV vehicle had fast charging capability (`1` or `0`)
`price`	price of vehicle ($USD)
`opCost`	operating cost of vehicle (US cents / mile)
`accelTime`	0-60 mph acceleration time (seconds)
`american`	dummy code for whether American brand (`1` or `0`)
`japanese`	dummy code for whether Japanese brand (`1` or `0`)

chinese	dummy code for whether Chinese brand (1 or 0)
skorean	dummy code for whether S. Korean brand (1 or 0)
weights	weights for each individual computed so that the sample age and income demographics matched with thos

Source

Raw data downloaded from this repo

References

Helveston, J. P., Liu, Y., Feit, E. M., Fuchs, E. R. H., Klampfl, E., & Michalek, J. J. (2015). "Will Subsidies Drive Electric Vehicle Adoption? Measuring Consumer Preferences in the U.S. and China." Transportation Research Part A: Policy and Practice, 73, 96–112. doi:10.1016/j.tra.2015.01.002

Examples

```
data(cars_us)

head(cars_us)
```

---

ci *Obtain a confidence interval from coefficient draws*

Description

Returns a data frame with the columns 'mean', 'lower', and 'upper' reflecting the mean and lower and upper bounds of a confidence interval (quantiles) for every column in a data frame of draws

Usage

```
ci(df, level = 0.95)
```

Arguments

df	A data frame of draws with all numeric columns.
level	The sensitivity of the computed confidence interval (CI). Defaults to level = 0.95, reflecting a 95% CI.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Obtain 10,000 draws of parameters from model
coefs <- coef(mnl_pref)
covariance <- vcov(mnl_pref)
coef_draws <- as.data.frame(MASS::mvrnorm(10^4, coefs, covariance))

# Compute a confidence interval
ci(coef_draws, level = 0.95)
```

---

confint.logitr *Extract Model Confidence Interval*

Description

Returns confidence intervals from an object of class `logitr`.

Usage

```
## S3 method for class 'logitr'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	is an object of class `logitr` (a model estimated using the 'logitr()' function).
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
`level`	The confidence level required.
`...`	further arguments.

Value

A data frame of the confidence intervals of model coefficients.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Compute a confidence interval
confint(mnl_pref)
```

`electricity` *Stated preference data for the choice of electricity suppliers (from mlogit package)*

Description

A sample of 2308 households in the United States.

Usage

```
data(electricity)
```

Format

Variable	Description
`id`	individual identifiers
`obsID`	identifier for unique choice observation
`choice`	dummy code for choice (1 or 0)
`alt`	alternative in each choice observation
`pf`	fixed price at a stated cents per kWh, with the price varying over suppliers and experiments, for scenario i=(1, 2, 3,
`cl`	the length of contract that the supplier offered, in years (such as 1 year or 5 years.) During this contract period, the s
`loc`	is the supplier a local company.
`wk`	is the supplier a well-known company.
`tod`	a time-of-day rate under which the price is 11 cents per kWh from 8am to 8pm and 5 cents per kWh from 8pm to 8a
`seas`	a seasonal rate under which the price is 10 cents per kWh in the summer, 8 cents per kWh in the winter, and 6 cents

Source

Kenneth Train's home page

References

Croissant, Y. (2020). Estimation of Random Utility Models in R: The mlogit Package. Journal of Statistical Software, 95(11), 1–41. doi:10.18637/jss.v095.i11

Examples

```
data(electricity)

head(electricity)
```

`fitted.logitr` *Extract Model Fitted Values*

Description

Returns fitted values from an object of class `logitr`.

Usage

```
## S3 method for class 'logitr'
fitted(object, probs = NULL, ...)
```

Arguments

`object`	is an object of class `logitr` (a model estimated using the 'logitr()' function).
`probs`	Predicted probabilities for an object of class `logitr` to use in computing fitted values Defaults to `NULL`.
`...`	further arguments.

Value

A data frame of the `obsID` and the fitted values extracted from `object`.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Extract the fitted values from the model
fitted(mnl_pref)
```

fquantile *Predict probabilities and / or outcomes*

Description

This function is a faster implementation of the "type 7" `quantile()` algorithm and is modified from this gist: https://gist.github.com/sikli/f1775feb9736073cefee97ec81f6b193 It returns sample quantiles corresponding to the given probabilities. The smallest observation corresponds to a probability of 0 and the largest to a probability of 1. For speed, output quantile names are removed as are error handling such as checking if x are factors, or if probs lie outside the `[0,1]` range.

Usage

```
fquantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE)
```

Arguments

`x`	numeric vector whose sample quantiles are wanted. `NA` and `NaN` values are not allowed in numeric vectors unless `na.rm` is `TRUE`.
`probs`	numeric vector of probabilities with values in `[0,1]`. (Values up to `2e-14` outside that range are accepted and moved to the nearby endpoint.)
`na.rm`	logical; if `TRUE`, any `NA` and `NaN`'s are removed from `x` before the quantiles are computed.

Value

A vector of length `length(probs)` is returned;

Examples

```
library(logitr)
```

glance.logitr *Glance a* `logitr` *class object*

Description

Glance a `logitr` class object

Usage

```
## S3 method for class 'logitr'
glance(x, ...)
```

Arguments

x	is an object of class logitr.
...	further arguments.

Value

A tibble of the model summary statistics.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Extract a tibble of the model summary statistics
glance(mnl_pref)
```

---

logitr *The main function for estimating logit models*

Description

Use this function to estimate multinomial (MNL) and mixed logit (MXL) models with "Preference" space or "Willingness-to-pay" (WTP) space utility parameterizations. The function includes an option to run a multistart optimization loop with random starting points in each iteration, which is useful for non-convex problems like MXL models or models with WTP space utility parameterizations. The main optimization loop uses the nloptr() function to minimize the negative log-likelihood function.

Usage

```
logitr(
  data,
  outcome,
  obsID,
  pars,
  scalePar = NULL,
  randPars = NULL,
  randScale = NULL,
  modelSpace = NULL,
  weights = NULL,
```

```
  panelID = NULL,
  clusterID = NULL,
  robust = FALSE,
  correlation = FALSE,
  startValBounds = c(-1, 1),
  startVals = NULL,
  numMultiStarts = 1,
  useAnalyticGrad = TRUE,
  scaleInputs = TRUE,
  standardDraws = NULL,
  drawType = "halton",
  numDraws = 50,
  numCores = NULL,
  vcov = FALSE,
  predict = TRUE,
 options = list(print_level = 0, xtol_rel = 1e-06, xtol_abs = 1e-06, ftol_rel = 1e-06,
    ftol_abs = 1e-06, maxeval = 1000, algorithm = "NLOPT_LD_LBFGS"),
  price,
  randPrice,
  choice,
  parNames,
  choiceName,
  obsIDName,
  priceName,
  weightsName,
  clusterName,
  cluster
)
```

Arguments

`data`	The data, formatted as a `data.frame` object.
`outcome`	The name of the column that identifies the outcome variable, which should be coded with a `1` for `TRUE` and `0` for `FALSE`.
`obsID`	The name of the column that identifies each observation.
`pars`	The names of the parameters to be estimated in the model. Must be the same as the column names in the `data` argument. For WTP space models, do not include the `scalePar` variable in `pars`.
`scalePar`	The name of the column that identifies the scale variable, which is typically "price" for WTP space models, but could be any continuous variable, such as "time". Defaults to `NULL`.
`randPars`	A named vector whose names are the random parameters and values the distribution: `'n'` for normal, `'ln'` for log-normal, or `'cn'` for zero-censored normal. Defaults to `NULL`.
`randScale`	The random distribution for the scale parameter: `'n'` for normal, `'ln'` for log-normal, or `'cn'` for zero-censored normal. Only used for WTP space MXL models. Defaults to `NULL`.

`modelSpace` This argument is no longer needed as of v0.7.0. The model space is now determined based on the `scalePar` argument: if `NULL` (the default), the model will be in the preference space, otherwise it will be in the WTP space. Defaults to `NULL`.

`weights` The name of the column that identifies the weights to be used in model estimation. Defaults to `NULL`.

`panelID` The name of the column that identifies the individual (for panel data where multiple observations are recorded for each individual). Defaults to `NULL`.

`clusterID` The name of the column that identifies the cluster groups to be used in model estimation. Defaults to `NULL`.

`robust` Determines whether or not a robust covariance matrix is estimated. Defaults to `FALSE`. Specification of a `clusterID` or `weights` will override the user setting and set this to 'TRUE' (a warning will be displayed in this case). Replicates the functionality of Stata's cmcmmixlogit.

`correlation` Set to `TRUE` to account for correlation across random parameters (correlated heterogeneity). Defaults to `FALSE`.

`startValBounds` sets the `lower` and `upper` bounds for the starting parameter values for each optimization run, which are generated by `runif(n, lower, upper)`. Defaults to `c(-1, 1)`.

`startVals` is vector of values to be used as starting values for the optimization. Only used for the first run if `numMultiStarts > 1`. Defaults to `NULL`.

`numMultiStarts` is the number of times to run the optimization loop, each time starting from a different random starting point for each parameter between `startValBounds`. Recommended for non-convex models, such as WTP space models and mixed logit models. Defaults to `1`.

`useAnalyticGrad`
Set to `FALSE` to use numerically approximated gradients instead of analytic gradients during estimation. For now, using the analytic gradient is faster for MNL models but slower for MXL models. Defaults to `TRUE`.

`scaleInputs` By default each variable in `data` is scaled to be between 0 and 1 before running the optimization routine because it usually helps with stability, especially if some of the variables have very large or very small values (e.g. `> 10^3` or `< 10^-3`). Set to `FALSE` to turn this feature off. Defaults to `TRUE`.

`standardDraws` By default, a new set of standard normal draws are generated during each call to `logitr` (the same draws are used during each multistart iteration). The user can override those draws by providing a matrix of standard normal draws if desired. Defaults to `NULL`.

`drawType` Specify the draw type as a character: `"halton"` (the default) or `"sobol"` (recommended for models with more than 5 random parameters).

`numDraws` The number of Halton draws to use for MXL models for the maximum simulated likelihood. Defaults to `50`.

`numCores` The number of cores to use for parallel processing of the multistart. Set to `1` to serially run the multistart. Defaults to `NULL`, in which case the number of cores is set to `parallel::detectCores() - 1`. Max cores allowed is capped at `parallel::detectCores()`.

vcov	Set to TRUE to evaluate and include the variance-covariance matrix and coefficient standard errors in the returned object. Defaults to FALSE.
predict	If FALSE, predicted probabilities, fitted values, and residuals are not included in the returned object. Defaults to TRUE.
options	A list of options for controlling the nloptr() optimization. Run nloptr::nloptr.print.options() for details.
price	No longer used as of v0.7.0 - if provided, this is passed to the scalePar argument and a warning is displayed.
randPrice	No longer used as of v0.7.0 - if provided, this is passed to the randScale argument and a warning is displayed.
choice	No longer used as of v0.4.0 - if provided, this is passed to the outcome argument and a warning is displayed.
parNames	No longer used as of v0.2.3 - if provided, this is passed to the pars argument and a warning is displayed.
choiceName	No longer used as of v0.2.3 - if provided, this is passed to the outcome argument and a warning is displayed.
obsIDName	No longer used as of v0.2.3 - if provided, this is passed to the obsID argument and a warning is displayed.
priceName	No longer used as of v0.2.3 - if provided, this is passed to the scalePar argument and a warning is displayed.
weightsName	No longer used as of v0.2.3 - if provided, this is passed to the weights argument and a warning is displayed.
clusterName	No longer used as of v0.2.3 - if provided, this is passed to the clusterID argument and a warning is displayed.
cluster	No longer used as of v0.2.3 - if provided, this is passed to the clusterID argument and a warning is displayed.

Details

The the options argument is used to control the detailed behavior of the optimization and must be passed as a list, e.g. options = list(...). Below are a list of the default options, but other options can be included. Run nloptr::nloptr.print.options() for more details.

Argument	Description	Default
xtol_rel	The relative x tolerance for the nloptr optimization loop.	1.0e-6
xtol_abs	The absolute x tolerance for the nloptr optimization loop.	1.0e-6
ftol_rel	The relative f tolerance for the nloptr optimization loop.	1.0e-6
ftol_abs	The absolute f tolerance for the nloptr optimization loop.	1.0e-6
maxeval	The maximum number of function evaluations for the nloptr optimization loop.	1000
algorithm	The optimization algorithm that nloptr uses.	"NLOPT_LD_LBFGS"
print_level	The print level of the nloptr optimization loop.	0

Value

The function returns a list object containing the following objects.

Value	Description
coefficients	The model coefficients at convergence.
logLik	The log-likelihood value at convergence.
nullLogLik	The null log-likelihood value (if all coefficients are 0).
gradient	The gradient of the log-likelihood at convergence.
hessian	The hessian of the log-likelihood at convergence.
probabilities	Predicted probabilities. Not returned if predict = FALSE.
fitted.values	Fitted values. Not returned if predict = FALSE.
residuals	Residuals. Not returned if predict = FALSE.
startVals	The starting values used.
multistartNumber	The multistart run number for this model.
multistartSummary	A summary of the log-likelihood values for each multistart run (if more than one multistart was used).
time	The user, system, and elapsed time to run the optimization.
iterations	The number of iterations until convergence.
message	A more informative message with the status of the optimization result.
status	An integer value with the status of the optimization (positive values are successes). Use statusCodes
call	The matched call to logitr().
inputs	A list of the original inputs to logitr().
data	A list of the original data provided to logitr() broken up into components used during model estima
numObs	The number of observations.
numParams	The number of model parameters.
freq	The frequency counts of each alternative.
modelType	The model type, 'mnl' for multinomial logit or 'mxl' for mixed logit.
weightsUsed	TRUE or FALSE for whether weights were used in the model.
numClusters	The number of clusters.
parSetup	A summary of the distributional assumptions on each model parameter ("f"="fixed", "n"="normal dis
parIDs	A list identifying the indices of each parameter in coefficients by a variety of types.
scaleFactors	A vector of the scaling factors used to scale each coefficient during estimation.
standardDraws	The draws used during maximum simulated likelihood (for MXL models).
options	A list of options for controlling the nloptr() optimization. Run nloptr::nloptr.print.options()

Examples

```
# For more detailed examples, visit
# https://jhelvy.github.io/logitr/articles/

library(logitr)

# Estimate a MNL model in the Preference space
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)
```

```
# Estimate a MNL model in the WTP space, using a 5-run multistart
mnl_wtp <- logitr(
  data          = yogurt,
  outcome       = "choice",
  obsID         = "obsID",
  pars          = c("feat", "brand"),
  scalePar      = "price",
  numMultiStarts = 5
)

# Estimate a MXL model in the Preference space with "feat"
# following a normal distribution
# Panel structure is accounted for in this example using "panelID"
mxl_pref <- logitr(
  data     = yogurt,
  outcome  = "choice",
  obsID    = "obsID",
  panelID  = "id",
  pars     = c("price", "feat", "brand"),
  randPars = c(feat = "n")
)
```

`logit_probs` *Compute logit fraction for sets of alternatives given coefficient draws*

Description

Returns a data frame of the predicted probabilities (with a confidence interval) for a data frame of alternatives given coefficient draws. WARNING: Most of the time you probably want to use `predict()` instead of this function. Where `logit_probs()` is useful is if you estimate a model with an interaction parameter to see differences between groups. In those cases, you can obtain draws of the estimated parameters and then use the draws to predict probabilities for each group after summing together the appropriate columns of the draws for each group. Also note that this function is only useful for multinomial logit models and is not appropriate for mixed logit models.

Usage

```
logit_probs(object, coef_draws, newdata, obsID = NULL, level = 0.95)
```

Arguments

`object`	is an object of class `logitr` (a model estimated using the 'logitr()' function).
`coef_draws`	A data frame of coefficients draws.
`newdata`	A data frame of sets of alternatives for which to compute logit probabilities. Each row is an alternative.
`obsID`	The name of the column in `newdata` that identifies each set of alternatives. Defaults to `NULL`, in which case it assumes the newdata are all one choice scenario.
`level`	The sensitivity of the computed confidence interval (CI). Defaults to `level = 0.95`, reflecting a 95% CI.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Create a set of alternatives for which to simulate probabilities
# (Columns are attributes, rows are alternatives)
data <- data.frame(
  altID       = c(1, 2, 3, 4),
  obsID       = c(1, 1, 1, 1),
  price       = c(8, 6, 7, 10),
  feat    = c(0, 1, 0, 0),
  brand   = c('dannon', 'hiland', 'weight', 'yoplait')
)

# Obtain 10,000 draws of parameters from model
coefs <- coef(mnl_pref)
covariance <- vcov(mnl_pref)
coef_draws <- as.data.frame(MASS::mvrnorm(10^4, coefs, covariance))

# Compute the probabilities
sim <- logit_probs(
  mnl_pref,
  coef_draws = coef_draws,
  newdata = data,
  obsID = 'obsID',
  level = 0.95
)
```

---

miscmethods.logitr *Methods for logitr objects*

Description

Miscellaneous methods for `logitr` class objects.

Usage

```
## S3 method for class 'logitr'
logLik(object, ...)

## S3 method for class 'logitr'
```

```
terms(x, ...)

## S3 method for class 'logitr'
coef(object, ...)

## S3 method for class 'summary.logitr'
coef(object, ...)

## S3 method for class 'logitr'
summary(object, ...)

## S3 method for class 'logitr'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'summary.logitr'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)

## S3 method for class 'logitr_wtp'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  width = getOption("width"),
  ...
)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
...	further arguments.
x	is an object of class logitr.
digits	the number of digits for printing, defaults to 3.
width	the width of the printing.

model.frame.logitr *Extracting the Model Frame from a Formula or Fit*

Description

Returns a data.frame with the variables needed to use formula and any ... arguments.

Usage

```
## S3 method for class 'logitr'
model.frame(formula, ...)
```

Arguments

formula	a model formula or terms object or an R object.
...	further arguments.

Value

A data.frame with the variables needed to use formula and any ... arguments.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Get the model.frame data frame
model.frame(mnl_pref)
```

model.matrix.logitr *Construct Design Matrices*

Description

Creates a design (or model) matrix, e.g., by expanding factors to a set of dummy variables (depending on the contrasts) and expanding interactions similarly.

Usage

```
## S3 method for class 'logitr'
model.matrix(object, ...)
```

Arguments

object	an object of an appropriate class. For the default method, a model formula or a terms object.
...	further arguments.

Value

A design matrix

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Get the model.matrix design matrix
model.matrix(mnl_pref)
```

---

predict.logitr *Predict probabilities and / or outcomes*

Description

This method is used for computing predicted probabilities and / or outcomes for either the data used for model estimation or a new data set consisting of a single or multiple sets of alternatives.

Usage

```
## S3 method for class 'logitr'
predict(
  object,
  newdata = NULL,
  obsID = NULL,
  type = "prob",
  returnData = FALSE,
  interval = "none",
  level = 0.95,
```

```
  numDrawsCI = 10^4,
  pars = NULL,
  scalePar = NULL,
  randPars = NULL,
  randScale = NULL,
  ci,
  ...
)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
newdata	a data.frame. Each row is an alternative and each column an attribute corresponding to parameter names in the estimated model. Defaults to NULL, in which case predictions are made on the original data used to estimate the model.
obsID	The name of the column that identifies each set of alternatives in the data. Required if newdata != NULL. Defaults to NULL, in which case the value for obsID from the data in object is used.
type	A character vector defining what to predict: prob for probabilities, outcomes for outcomes. If you want both outputs, use c("prob", "outcome"). Outcomes are predicted randomly according to the predicted probabilities. Defaults to "prob".
returnData	If TRUE the data is also returned, otherwise only the predicted values ("prob" and / or "outcome") are returned. Defaults to FALSE.
interval	Type of interval calculation: "none" (default) or "confidence". Future versions will include "prediction" intervals as well.
level	Tolerance / confidence interval. Defaults to 0.95.
numDrawsCI	The number of draws to use in simulating uncertainty for the computed CI. Defaults to 10^4.
pars	The names of the parameters to be estimated in the model. Must be the same as the column names in the data argument. For WTP space models, do not include the scalePar variable in pars.
scalePar	The name of the column that identifies the scale variable, which is typically "price" for WTP space models, but could be any continuous variable, such as "time". Defaults to NULL.
randPars	A named vector whose names are the random parameters and values the distribution: 'n' for normal, 'ln' for log-normal, or 'cn' for zero-censored normal. Defaults to NULL.
randScale	The random distribution for the scale parameter: 'n' for normal, 'ln' for log-normal, or 'cn' for zero-censored normal. Only used for WTP space MXL models. Defaults to NULL.
ci	No longer used as of v1.1.0 - if provided, this is passed to the level argument, interval is set to "confidence", and a warning is displayed.
...	further arguments.

Value

A data frame of predicted probabilities and / or outcomes.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Predict probabilities and / or outcomes

# Predict probabilities for each alternative in the model data
probs <- predict(mnl_pref)
head(probs)

# Create a set of alternatives for which to make predictions.
# Each row is an alternative and each column an attribute.
data <- subset(
    yogurt, obsID %in% c(42, 13),
    select = c('obsID', 'alt', 'price', 'feat', 'brand'))
data

# Predict probabilities using the estimated model
predict(mnl_pref, newdata = data, obsID = "obsID")

# Predict probabilities and include a 95% confidence interval
predict(
  mnl_pref,
  newdata = data,
  obsID = "obsID",
  interval = "confidence",
  level = 0.95
)

# Predict outcomes
predict(mnl_pref, newdata = data, obsID = "obsID", type = "outcome")

# Predict outcomes and probabilities
predict(mnl_pref, newdata = data, obsID = "obsID", type = c("prob", "outcome"))
```

`recodeData` *Returns a list of the design matrix* `X` *and updated* `pars` *and* `randPars` *to include any dummy-coded categorical or interaction variables.*

Description

Recodes the data and returns a list of the encoded design matrix (X) as well as two vectors (pars and randPars) with discrete (categorical) variables and interaction variables added to X, pars, and randPars.

Usage

```
recodeData(data, pars, randPars)
```

Arguments

data	The data, formatted as a data.frame object.
pars	The names of the parameters to be estimated in the model. Must be the same as the column names in the data argument. For WTP space models, do not include price in pars - it should instead be defined by the scalePar argument.
randPars	A named vector whose names are the random parameters and values the distribution: 'n' for normal or 'ln' for log-normal. Defaults to NULL.

Value

A list of the design matrix (X) and two vectors (pars and randPars) with discrete (categorical) variables and interaction variables added.

Examples

```
library(logitr)

data(yogurt)

# Recode the yogurt data
result <- recodeData(
    data = yogurt,
    pars = c("price", "feat", "brand", "price*brand"),
    randPars = c(feat = "n", brand = "n")
)

result$formula
result$pars
result$randPars
head(result$X)
```

---

residuals.logitr *Extract Model Residuals*

Description

Returns model residuals from an object of class logitr.

Usage

```
## S3 method for class 'logitr'
residuals(object, fitted = NULL, ...)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
fitted	Fitted values for an object of class logitr to use in computing residuals. Defaults to NULL.
...	further arguments.

Value

A data frame of the obsID and the residuals (response minus fitted values) extracted from object.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Extract the residuals from the model
residuals(mnl_pref)
```

---

runtimes *Data frame of run times for logitr benchmark*

Description

This data frame contains the run times for a benchmark comparing the relative computation time to estimate a preference space mixed logit model using the following R packages: logitr, mixl, mlogit, gmnl, and apollo. The run times are exported from the Google colab notebook here: https://colab.research.google.com/drive/1vYlBdJd4xCV43UwJ33XXpO3Ys8xWkuxx?usp=sharing

Usage

```
data(runtimes)
```

Format

Variable	Description
package	Package name.
time_sec	The estimation time in seconds.
numDraws	The number of random draws used during estimation.

Source

This Google colab notebook

Examples

```
data(runtimes)

head(runtimes)
```

---

se *Extract standard errors*

Description

Extract standard errors

Usage

```
se(object, ...)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
...	further arguments.

se.logitr *Extract standard errors*

Description

Extract standard errors

Usage

```
## S3 method for class 'logitr'
se(object, ...)
```

Arguments

`object`	is an object of class `logitr` (a model estimated using the 'logitr()' function).
`...`	further arguments.

statusCodes *View a description the nloptr status codes*

Description

Prints a description of the status codes from the nloptr optimization routine.

Usage

```
statusCodes()
```

Value

No return value; prints a summary of the `nloptr` status codes to the console.

Examples

```
statusCodes()
```

`tidy.logitr` *Tidy a* `logitr` *class object*

Description

Tidy a `logitr` class object

Usage

```
## S3 method for class 'logitr'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	is an object of class `logitr`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
`conf.level`	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Unused, included for generic consistency only.

Value

A tidy `tibble::tibble()` summarizing component-level information about the model

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Extract a tibble of the model coefficients
tidy(mnl_pref)

# Extract a tibble of the model coefficients with confidence intervals
tidy(mnl_pref, conf.int = TRUE)
```

vcov.logitr *Calculate the variance-covariance matrix*

Description

Returns the variance-covariance matrix of the main parameters of a fitted model object.

Usage

```
## S3 method for class 'logitr'
vcov(object, ...)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
...	further arguments.

wtp *Get WTP estimates a preference space model*

Description

Returns the computed WTP from a preference space model.

Usage

```
wtp(object, scalePar)
```

Arguments

object	is an object of class logitr (a model estimated using the 'logitr()' function).
scalePar	The name of the column that identifies the scale variable, which is typically "price" for WTP space models, but could be any continuous variable, such as "time".

Details

Willingness to pay is computed by dividing the estimated parameters of a utility model in the "preference" space by the scale parameter, which is should be price to obtain WTP estimates. Uncertainty is handled via simulation.

Value

A data frame of the WTP estimates.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Compute the WTP implied from the preference space model
wtp(mnl_pref, scalePar = "price")
```

`wtp.logitr` *Get WTP estimates a preference space model*

Description

Returns the computed WTP from a preference space model.

Usage

```
## S3 method for class 'logitr'
wtp(object, scalePar)
```

Arguments

`object`	is an object of class `logitr` (a model estimated using the 'logitr()' function).
`scalePar`	The name of the column that identifies the scale variable, which is typically "price" for WTP space models, but could be any continuous variable, such as "time".

Details

Willingness to pay is computed by dividing the estimated parameters of a utility model in the "preference" space by the scale parameter, which is should be price to obtain WTP estimates. Uncertainty is handled via simulation.

Value

A data frame of the WTP estimates.

Examples

```
library(logitr)

# Estimate a preference space model
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Compute the WTP implied from the preference space model
wtp(mnl_pref, scalePar = "price")
```

`wtpCompare` *Compare WTP from preference and WTP space models*

Description

Returns a comparison of the WTP between a preference space and WTP space model.

Usage

```
wtpCompare(model_pref, model_wtp, scalePar)
```

Arguments

`model_pref`	The output of a "preference space" model estimated using the `logitr()` function.
`model_wtp`	The output of a "willingness to pay space" model estimated using the `logitr()` function.
`scalePar`	The name of the column that identifies the scale variable, which is typically "price" for WTP space models, but could be any continuous variable, such as "time".

Details

Willingness to pay (WTP) is first computed from the preference space model by dividing the estimated parameters by the scale parameter (typically "price" to obtain WTP estimates). Then those estimates are compared against the WTP values directly estimated from the "WTP" space model. Uncertainty is handled via simulation.

Value

A data frame comparing the WTP estimates from preference space and WTP space models.

Examples

```
library(logitr)

# Estimate a MNL model in the Preference space
mnl_pref <- logitr(
  data    = yogurt,
  outcome = "choice",
  obsID   = "obsID",
  pars    = c("price", "feat", "brand")
)

# Compute the WTP implied from the preference space model
wtp_mnl_pref <- wtp(mnl_pref, scalePar = "price")

# Estimate a MNL model in the WTP Space, using the computed WTP values
# from the preference space model as starting points
mnl_wtp <- logitr(
  data      = yogurt,
  outcome   = "choice",
  obsID     = "obsID",
  pars      = c("feat", "brand"),
  scalePar  = "price",
  startVals = wtp_mnl_pref$Estimate
)

# Compare the WTP between the two spaces
wtpCompare(mnl_pref, mnl_wtp, scalePar = "price")
```

---

yogurt *Choice observations of yogurt purchases by 100 households*

Description

Data from Jain et al. (1994) containing 2,412 choice observations from a series of yogurt purchases by a panel of 100 households in Springfield, Missouri, over a roughly two-year period. The data were collected by optical scanners and contain information about the price, brand, and a "feature" variable, which identifies whether a newspaper advertisement was shown to the customer. There are four brands of yogurt: Yoplait, Dannon, Weight Watchers, and Hiland, with market shares of 34%, 40%, 23% and 3%, respectively.

Usage

```
data(yogurt)
```

Format

Variable	Description
id	individual identifiers

`obsID`	identifier for unique choice observation
`alt`	alternative in each choice observation
`choice`	dummy code for choice (1 or 0)
`price`	price of yogurt
`feat`	dummy for whether a newspaper advertisement was shown to the customer (`1` or `0`)
`brand`	yogurt brand: `"yoplait"`, `"dannon"`, `"hiland"`, or `"weight"` (for weight watcher)

Source

Raw data downloaded from the package mlogit v0.3-0 by Yves Croissant archive

References

Dipak C. Jain, Naufel J. Vilcassim & Pradeep K. Chintagunta (1994) A Random-Coefficients Logit Brand-Choice Model Applied to Panel Data, Journal of Business & Economic Statistics, 12:3, 317-328, doi:10.1080/07350015.1994.10524547

Examples

```
data(yogurt)

head(yogurt)
```

Index