Package ‘logitr’

June 17, 2022

Title Logit Models w/Preference & WTP Space Utility Parameterizations
Version 0.7.0
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) "Discrete Choice Methods with Simulation, 2nd Edition" <doi:10.1017/CBO9780511805271>.

License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
VignetteBuilder knitr
Depends R (>= 3.5.0)
Suggests dplyr, fastDummies, ggplot2, ggrepel, here, kableExtra, knitr, rmarkdown, testthat, tidyr
Imports nloptr, parallel, stats, randtoolbox, MASS
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>
**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)
apolloModeChoiceData

Format

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

Source

Data imported from the apollo package archive

References


Examples

data(apolloModeChoiceData)
head(apolloModeChoiceData)
### 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**

```r
data(cars_china)
```

**Format**

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

**Source**

Raw data downloaded from [this repo](#)
References

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

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

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 those
carried in the general car-buying population.

Source
Raw data downloaded from this repo

References

Examples
data(cars_us)
head(cars_us)

| 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>individual identifiers</td>
</tr>
<tr>
<td>obsID</td>
<td>identifier for unique choice observation</td>
</tr>
<tr>
<td>choice</td>
<td>dummy code for choice (1 or 0)</td>
</tr>
<tr>
<td>alt</td>
<td>alternative in each choice observation</td>
</tr>
<tr>
<td>pf</td>
<td>fixed price at a stated cents per kWh, with the price varying over suppliers and experiments, for scenario ( i = 1, 2, 3, 4 )</td>
</tr>
<tr>
<td>cl</td>
<td>the length of contract that the supplier offered, in years (such as 1 year or 5 years.) During this contract period, the supplier could not cancel the contract. In the second contract period, the supplier could cancel the contract without penalty. For the third contract period, if the supplier did not renew the contract or contracted in a different manner, the contract was automatically renewed. This is recorded as a contract length of 0.</td>
</tr>
<tr>
<td>loc</td>
<td>is the supplier a local company.</td>
</tr>
<tr>
<td>wk</td>
<td>is the supplier a well-known company.</td>
</tr>
<tr>
<td>tod</td>
<td>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 8am.</td>
</tr>
<tr>
<td>seas</td>
<td>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 per kWh in the rest of the year.</td>
</tr>
</tbody>
</table>
fitted.logitr

Source

Kenneth Train’s home page

References


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](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)
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**

```r
logitr(
  data, outcome, obsID, pars,
  scalePar = NULL, randPars = NULL, randScale = NULL,
  modelSpace = NULL, weights = NULL, panelID = NULL, clusterID = NULL,
  robust = FALSE, correlation = FALSE,
  startParBounds = c(-1, 1), startVals = NULL,
  numMultiStarts = 1, useAnalyticGrad = TRUE, scaleInputs = TRUE,
  standardDraws = NULL, drawType = "halton", numDraws = 50,
  numCores = NULL, vcov = FALSE, predict = TRUE,
  price, randPrice, choice, parNames, choiceName, obsIDName,
  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"),
```
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 cmcmmxlogit.
correlation Set to TRUE to account for correlation across random parameters (correlated heterogeneity). Defaults to FALSE.
startParBounds sets the lower and upper bounds for the starting parameters 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 startParBounds. 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.

cscaleInputs
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 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.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>xtol_rel</td>
<td>The relative x tolerance for the nloptr optimization loop.</td>
<td>1.0e-6</td>
</tr>
<tr>
<td>xtol_abs</td>
<td>The absolute x tolerance for the nloptr optimization loop.</td>
<td>1.0e-6</td>
</tr>
<tr>
<td>ftol_rel</td>
<td>The relative f tolerance for the nloptr optimization loop.</td>
<td>1.0e-6</td>
</tr>
<tr>
<td>ftol_abs</td>
<td>The absolute f tolerance for the nloptr optimization loop.</td>
<td>1.0e-6</td>
</tr>
<tr>
<td>maxeval</td>
<td>The maximum number of function evaluations for the nloptr optimization loop.</td>
<td>1000</td>
</tr>
<tr>
<td>algorithm</td>
<td>The optimization algorithm that nloptr uses.</td>
<td>&quot;NLOPT_LD_LBFGS&quot;</td>
</tr>
<tr>
<td>print_level</td>
<td>The print level of the nloptr optimization loop.</td>
<td>0</td>
</tr>
</tbody>
</table>

Value

The function returns a list object containing the following objects.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coefficients</td>
<td>The model coefficients at convergence.</td>
</tr>
<tr>
<td>logLik</td>
<td>The log-likelihood value at convergence.</td>
</tr>
<tr>
<td>nullLogLik</td>
<td>The null log-likelihood value (if all coefficients are 0).</td>
</tr>
<tr>
<td>gradient</td>
<td>The gradient of the log-likelihood at convergence.</td>
</tr>
<tr>
<td>hessian</td>
<td>The hessian of the log-likelihood at convergence.</td>
</tr>
<tr>
<td>probabilities</td>
<td>Predicted probabilities. Not returned if predict = FALSE.</td>
</tr>
<tr>
<td>fitted.values</td>
<td>Fitted values. Not returned if predict = FALSE.</td>
</tr>
<tr>
<td>residuals</td>
<td>Residuals. Not returned if predict = FALSE.</td>
</tr>
<tr>
<td>startPars</td>
<td>The starting values used.</td>
</tr>
<tr>
<td>multistartNumber</td>
<td>The multistart run number for this model.</td>
</tr>
<tr>
<td>multistartSummary</td>
<td>A summary of the log-likelihood values for each multistart run (if more than one multistart was used).</td>
</tr>
<tr>
<td>time</td>
<td>The user, system, and elapsed time to run the optimization.</td>
</tr>
<tr>
<td>iterations</td>
<td>The number of iterations until convergence.</td>
</tr>
<tr>
<td>message</td>
<td>A more informative message with the status of the optimization result.</td>
</tr>
<tr>
<td>status</td>
<td>An integer value with the status of the optimization (positive values are successes). Use statusCodes()</td>
</tr>
<tr>
<td>call</td>
<td>The matched call to logitr().</td>
</tr>
<tr>
<td>inputs</td>
<td>A list of the original inputs to logitr().</td>
</tr>
<tr>
<td>data</td>
<td>A list of the original data provided to logitr() broken up into components used during model estimation.</td>
</tr>
<tr>
<td>numObs</td>
<td>The number of observations.</td>
</tr>
<tr>
<td>numParams</td>
<td>The number of model parameters.</td>
</tr>
<tr>
<td>freq</td>
<td>The frequency counts of each alternative.</td>
</tr>
</tbody>
</table>
**miscmethods.logitr**

- **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 distribution", "ln"="log-normal distribution").

- **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**

```r
# 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")
)
```

---

**miscmethods.logitr**  *Methods for logitr objects*
Description

Miscellaneous methods for logitr class objects.

Usage

```r
## 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.
predict.logitr

x is an object of class logitr.
digits the number of digits for printing, defaults to 3.
width the width of the printing.

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,
  ci = NULL,
  numDrawsCI = 10^3,
  ...
)

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.
ci If a confidence interval (CI) for the predicted probabilities is desired, set ci to a number between 0 and 1 to define the CI sensitivity. For example, ci = 0.95 will return a 95% CI. Defaults to NULL, in which case no CI is computed.
numDrawsCI The number of draws to use in simulating uncertainty for the computed CI. Defaults to 10^3.

... 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 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

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

residuals.logitr Extract Model Residuals

Description

Returns model residuals from an object of class `logitr`.

Usage

```r
## 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

```r
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/1vY1BdJd4xCV43UwJ33XXpO3Ys8xWkuxx?usp=sharing

### Usage

```r
data(runtimes)
```

### Format

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>package</td>
<td>Package name.</td>
</tr>
<tr>
<td>time_sec</td>
<td>The estimation time in seconds.</td>
</tr>
<tr>
<td>numDraws</td>
<td>The number of random draws used during estimation.</td>
</tr>
</tbody>
</table>

### 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).
...

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()

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**

\[ \text{wtp(object, scalePar)} \]

**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 should be price to obtain WTP estimates. Uncertainty is handled via simulation.

**Value**

A data frame of the WTP estimates.

**Examples**

```r
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

**Description**

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

**Usage**

```r
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**

```r
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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>individual identifiers</td>
</tr>
<tr>
<td>obsID</td>
<td>identifier for unique choice observation</td>
</tr>
<tr>
<td>alt</td>
<td>alternative in each choice observation</td>
</tr>
<tr>
<td>choice</td>
<td>dummy code for choice (1 or 0)</td>
</tr>
<tr>
<td>price</td>
<td>price of yogurt</td>
</tr>
<tr>
<td>feat</td>
<td>dummy for whether a newspaper advertisement was shown to the customer (1 or 0)</td>
</tr>
<tr>
<td>brand</td>
<td>yogurt brand: &quot;yoplait&quot;, &quot;dannon&quot;, &quot;hiland&quot;, or &quot;weight&quot; (for weight watcher)</td>
</tr>
</tbody>
</table>

#### Source

Raw data downloaded from the package mlogit v0.3-0 by Yves Croissant [archive](http://archive.yvescroissant.com/mlogit/v0.3-0.html)
References


Examples

data(yogurt)

head(yogurt)
Index

* codes
  statusCodes, 20
* datasets
  apolloModeChoiceData, 2
cars_china, 4
cars_us, 5
electricity, 6
runtimes, 18
yogurt, 24
* fitted_values
  fitted.logitr, 7
* fitted
  fitted.logitr, 7
* logit
  fitted.logitr, 7
  logitr, 9
  predict.logitr, 15
  residuals.logitr, 17
  statusCodes, 20
  wtp, 21
  wtp.logitr, 22
  wtpCompare, 23
* logitr
  fitted.logitr, 7
  logitr, 9
* mixed
  logitr, 9
* mnl
  logitr, 9
* mxl
  logitr, 9
* nloptr
  statusCodes, 20
* predict
  predict.logitr, 15
* probabilities
  predict.logitr, 15
* residuals
  residuals.logitr, 17
* resid
  residuals.logitr, 17
  runtimes, 18
* status
  statusCodes, 20
* willingness-to-pay
  logitr, 9
* wtp
  logitr, 9
  wtp, 21
  wtp.logitr, 22
  wtpCompare, 23
apolloModeChoiceData, 2
cars_china, 4
cars_us, 5
coef.logitr (miscmethods.logitr), 13
coef.summary.logitr (miscmethods.logitr), 13
electricity, 6
fitted.logitr, 7
fquantile, 8
logitr, 9
logLik.logitr (miscmethods.logitr), 13
miscmethods.logitr, 13
predict.logitr, 15
print.logitr (miscmethods.logitr), 13
print.logitr_wtp (miscmethods.logitr), 13
print.summary.logitr (miscmethods.logitr), 13
recodeData, 16
residuals.logitr, 17
runtimes, 18
se, 19
INDEX

se.logitr, 19
statusCodes, 20
statusCodes(), 12
summary.logitr (miscmethods.logitr), 13

terms.logitr (miscmethods.logitr), 13

vcov.logitr, 20

wtp, 21
wtp.logitr, 22
wtpCompare, 23

yogurt, 24