

Package ‘logitr’

July 6, 2026

Title Logit Models w/Preference & WTP Space Utility Parameterizations

Version 1.2.0

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) <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 7.3.2

VignetteBuilder knitr

Depends R (>= 3.5.0)

Suggests apollo, broom, broom.helpers (>= 1.15.0), dplyr, fastDummies, ggplot2, ggrepel, gmn1, gtsummary (>= 2.0.0), here, kableExtra, knitr, mixl, mlogit, rmarkdown, testthat, texreg, tidyr

Imports generics, MASS, mirai, nloptr, parallel, randtoolbox, Rcpp, RcppParallel, stats, tibble

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

URL <https://github.com/jhelvy/logitr>, <https://jhelyv.github.io/logitr/>

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

NeedsCompilation yes

Author John Helveston [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-2657-9191>),
 Connor Forsythe [ctb]
Maintainer John Helveston <john.helveston@gmail.com>
Repository CRAN
Date/Publication 2026-07-06 15:20:02 UTC

Contents

apolloModeChoiceData	3
augment.logitr	4
cars_china	5
cars_us	6
ci	8
confint.logitr	8
electricity	9
fitted.logitr	10
fquantile	11
glance.logitr	12
logitr	13
logit_probs	18
loglik_draws	20
miscmethods.logitr	21
model.frame.logitr	22
model.matrix.logitr	23
predict.logitr	23
recodeData	26
residuals.logitr	27
runtimes	28
runtimes_draws	29
se	30
se.logitr	30
statusCodes	31
tidy.logitr	31
validate_data	32
vcov.logitr	33
wtp	33
wtp.logitr	34
wtpCompare	35
yogurt	36
Index	38

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 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](https://doi.org/10.1016/j.jocm.2019.100170)

Examples

```
data(apolloModeChoiceData)
```

```
head(apolloModeChoiceData)
```

augment.logitr	<i>Glance a logitr class object</i>
----------------	-------------------------------------

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 those of the US population

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)
```

<i>cars_us</i>	<i>Stated car choice observations by US car buyers</i>
----------------	--

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 those

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	<i>Obtain a confidence interval from coefficient draws</i>
----	--

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	<i>Extract Model Confidence Interval</i>
----------------	--

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 <code>logitr</code> (a model estimated using the <code>'logitr()'</code> 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	<i>Stated preference data for the choice of electricity suppliers (from mlogit package)</i>
-------------	---

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)
```

<code>fitted.logitr</code>	<i>Extract Model Fitted Values</i>
----------------------------	------------------------------------

Description

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

Usage

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

Arguments

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

<code>x</code>	numeric vector whose sample quantiles are wanted. NA and NaN values are not allowed in numeric vectors unless <code>na.rm</code> is TRUE.
<code>probs</code>	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.)
<code>na.rm</code>	logical; if TRUE, any NA and NaN's are removed from <code>x</code> before the quantiles are computed.

Value

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

Examples

```
library(logitr)
```

glance.logitr	<i>Glance a logitr class object</i>
---------------	-------------------------------------

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 = "sobol",
  numDraws = 500,
  numDrawsBatch = NULL,
  numCores = NULL,
  numThreads = NULL,
  vcov = FALSE,
  predict = TRUE,
  backend = "cpp",
  options = list(print_level = 0, xtol_rel = 1e-06, xtol_abs = 1e-06, ftol_rel = 1e-10,
    ftol_abs = 1e-10, maxeval = 1000, algorithm = "NLOPT_LD_LBFGS"),
  price,
  randPrice,
  choice,
```

```

    parNames,
    choiceName,
    obsIDName,
    priceName,
    weightsName,
    clusterName,
    cluster
  )

```

Arguments

<code>data</code>	The data, formatted as a <code>data.frame</code> object.
<code>outcome</code>	The name of the column that identifies the outcome variable, which should be coded with a 1 for TRUE and 0 for FALSE.
<code>obsID</code>	The name of the column that identifies each observation.
<code>pars</code>	The names of the parameters to be estimated in the model. Must be the same as the column names in the <code>data</code> argument. For WTP space models, do not include the <code>scalePar</code> variable in <code>pars</code> .
<code>scalePar</code>	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.
<code>randPars</code>	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.
<code>randScale</code>	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.
<code>modelSpace</code>	This argument is no longer needed as of v0.7.0. The model space is now determined based on the <code>scalePar</code> argument: if NULL (the default), the model will be in the preference space, otherwise it will be in the WTP space. Defaults to NULL.
<code>weights</code>	The name of the column that identifies the weights to be used in model estimation. Defaults to NULL.
<code>panelID</code>	The name of the column that identifies the individual (for panel data where multiple observations are recorded for each individual). Defaults to NULL.
<code>clusterID</code>	The name of the column that identifies the cluster groups to be used in model estimation. Defaults to NULL.
<code>robust</code>	Determines whether or not a robust covariance matrix is estimated. Defaults to FALSE. Specification of a <code>clusterID</code> or <code>weights</code> will override the user setting and set this to 'TRUE' (a warning will be displayed in this case). Replicates the functionality of Stata's <code>cmcmmlxlogit</code> .
<code>correlation</code>	Set to TRUE to account for correlation across random parameters (correlated heterogeneity). Defaults to FALSE.
<code>startValBounds</code>	sets the lower and upper bounds for the starting parameter values for each optimization run, which are generated by <code>runif(n, lower, upper)</code> . Defaults to <code>c(-1, 1)</code> .

<code>startVals</code>	is vector of values to be used as starting values for the optimization. Only used for the first run if <code>numMultiStarts > 1</code> . Defaults to <code>NULL</code> .
<code>numMultiStarts</code>	is the number of times to run the optimization loop, each time starting from a different random starting point for each parameter between <code>startValBounds</code> . Recommended for non-convex models, such as WTP space models and mixed logit models. Defaults to 1.
<code>useAnalyticGrad</code>	Set to <code>FALSE</code> 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 <code>TRUE</code> .
<code>scaleInputs</code>	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 <code>FALSE</code> to turn this feature off. Defaults to <code>TRUE</code> .
<code>standardDraws</code>	By default, a new set of standard normal draws are generated during each call to <code>logitr</code> (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 <code>NULL</code> .
<code>drawType</code>	Specify the draw type as a character: "sobol" (the default), "halton", or "mlhs" (Modified Latin Hypercube Sampling). Sobol and Halton draws are deterministic; MLHS draws are randomized and so are controlled by <code>set.seed()</code> . For models with more than 5 random parameters, "sobol" or "mlhs" are recommended over "halton", which becomes correlated in higher dimensions.
<code>numDraws</code>	The number of draws to use for MXL models for the maximum simulated likelihood. Defaults to 500. Draw counts that are too low (e.g. the 40-50 draws that many packages default to) produce noticeably unstable simulated log-likelihoods, especially in models with many random parameters; the compiled backend makes larger draw counts cheap enough to be the default.
<code>numDrawsBatch</code>	The number of draws to process at a time when evaluating the simulated log-likelihood of MXL models with <code>backend = "cpu"</code> . Batching (or "streaming") the draws keeps peak memory bounded by the batch size rather than by <code>numDraws</code> , which is what makes large draw counts feasible on the R backend. Defaults to <code>NULL</code> , in which case <code>logitr</code> automatically streams only when the draws would otherwise exceed an internal memory budget (so typical models are unaffected). Set to an integer to force a specific batch size, or to a value $\geq$ <code>numDraws</code> to disable streaming. Only used by the "cpu" backend for MXL models; the default "cpp" backend is memory-flat in the number of draws and ignores this argument.
<code>numCores</code>	The number of cores to use for parallel processing of the multistart. Set to 1 to serially run the multistart. Defaults to <code>NULL</code> , in which case the number of cores is set to <code>parallel::detectCores() - 1</code> . Max cores allowed is capped at <code>parallel::detectCores()</code> .
<code>numThreads</code>	The number of threads to use for parallel evaluation of the MXL simulated log-likelihood with <code>backend = "cpp"</code> . The draws are processed in parallel across threads. Defaults to <code>NULL</code> , in which case a single thread is used when running a

	parallel multistart (<code>numMultiStarts > 1</code> , to avoid oversubscribing cores), otherwise all but one of the available cores are used. Set to 1 to disable threading. Only used by the "cpp" backend. Note that threading uses a parallel reduction, so results are not bit-identical across runs (they differ at the level of floating-point rounding, far below the optimization tolerance). Set <code>numThreads = 1</code> (or <code>backend = "cpu"</code>) if you need exactly reproducible results.
<code>vcov</code>	Set to <code>TRUE</code> to evaluate and include the variance-covariance matrix and coefficient standard errors in the returned object. Defaults to <code>FALSE</code> .
<code>predict</code>	If <code>FALSE</code> , predicted probabilities, fitted values, and residuals are not included in the returned object. Defaults to <code>TRUE</code> .
<code>backend</code>	The computational backend used to evaluate the log-likelihood and gradient during estimation. For mixed logit (MXL) models the default is "cpp", a compiled C++ implementation that is typically about 4 times faster (and faster still with multiple threads, see <code>numThreads</code>) while producing the same results as the native R backend to floating-point precision. Set <code>backend = "cpu"</code> to use logitr's native R implementation instead (for example if you want exactly bit-reproducible results, since the "cpp" backend's threaded reduction is not bit-identical across runs). Multinomial logit (MNL) models always use the R implementation regardless of this argument, as they are already fast.
<code>options</code>	A list of options for controlling the <code>nloptr()</code> optimization. Run <code>nloptr::nloptr.print.options()</code> for details.
<code>price</code>	No longer used as of v0.7.0 - if provided, this is passed to the <code>scalePar</code> argument and a warning is displayed.
<code>randPrice</code>	No longer used as of v0.7.0 - if provided, this is passed to the <code>randScale</code> argument and a warning is displayed.
<code>choice</code>	No longer used as of v0.4.0 - if provided, this is passed to the <code>outcome</code> argument and a warning is displayed.
<code>parNames</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>pars</code> argument and a warning is displayed.
<code>choiceName</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>outcome</code> argument and a warning is displayed.
<code>obsIDName</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>obsID</code> argument and a warning is displayed.
<code>priceName</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>scalePar</code> argument and a warning is displayed.
<code>weightsName</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>weights</code> argument and a warning is displayed.
<code>clusterName</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>clusterID</code> argument and a warning is displayed.
<code>cluster</code>	No longer used as of v0.2.3 - if provided, this is passed to the <code>clusterID</code> 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 `nlopitr::nlopitr.print.options()` for more details.

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

Value

The function returns a list object containing the following objects.

Value	Description
<code>coefficients</code>	The model coefficients at convergence.
<code>loglik</code>	The log-likelihood value at convergence.
<code>nullLogLik</code>	The null log-likelihood value (if all coefficients are 0).
<code>gradient</code>	The gradient of the log-likelihood at convergence.
<code>hessian</code>	The hessian of the log-likelihood at convergence.
<code>probabilities</code>	Predicted probabilities. Not returned if <code>predict = FALSE</code> .
<code>fitted.values</code>	Fitted values. Not returned if <code>predict = FALSE</code> .
<code>residuals</code>	Residuals. Not returned if <code>predict = FALSE</code> .
<code>startVals</code>	The starting values used.
<code>multistartNumber</code>	The multistart run number for this model.
<code>multistartSummary</code>	A summary of the log-likelihood values for each multistart run (if more than one multistart was used).
<code>time</code>	The user, system, and elapsed time to run the optimization.
<code>iterations</code>	The number of iterations until convergence.
<code>message</code>	A more informative message with the status of the optimization result.
<code>status</code>	An integer value with the status of the optimization (positive values are successes). Use statusCodes
<code>call</code>	The matched call to <code>logitr()</code> .
<code>inputs</code>	A list of the original inputs to <code>logitr()</code> .
<code>data</code>	A list of the original data provided to <code>logitr()</code> broken up into components used during model estimation.
<code>numObs</code>	The number of observations.
<code>numParams</code>	The number of model parameters.
<code>freq</code>	The frequency counts of each alternative.
<code>modelType</code>	The model type, <code>'mnl'</code> for multinomial logit or <code>'mxl'</code> for mixed logit.
<code>weightsUsed</code>	TRUE or FALSE for whether weights were used in the model.
<code>numClusters</code>	The number of clusters.
<code>parSetup</code>	A summary of the distributional assumptions on each model parameter (<code>"f"="fixed"</code> , <code>"n"="normal distribution"</code>).
<code>parIDs</code>	A list identifying the indices of each parameter in <code>coefficients</code> by a variety of types.
<code>scaleFactors</code>	A vector of the scaling factors used to scale each coefficient during estimation.
<code>standardDraws</code>	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"),
  numDraws  = 50 # fewer draws than the default to keep the example fast
)
```

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

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

```

loglik_draws	<i>Data frame of mixed logit log-likelihoods as the number of draws grows</i>
--------------	---

Description

This data frame contains the converged log-likelihoods of the same preference space mixed logit model used in the [runtimes](#) and [runtimes_draws](#) benchmarks, estimated repeatedly at increasing numbers of random draws. At each draw count the model is estimated 10 times using randomized MLHS draws with a different seed each time, with every run starting from the same converged reference solution, so the across-seed spread in log-likelihood at a given draw count reflects simulation error: the noise in the simulated log-likelihood surface shifts the converged value, and at low draw counts it can distort the surface enough that runs converge to different local optima entirely. It is used to show that low draw counts produce unstable simulated log-likelihoods, which stabilize as the number of draws grows. It is generated by the script `data-row/runtimes.R` in the package repository.

Usage

```
data(loglik_draws)
```

Format

Variable	Description
numDraws	The number of random draws used during estimation.
seed	The random seed used to generate the MLHS draws.
logLik	The converged log-likelihood of the model.
time_sec	The estimation time in seconds (on a 10-core Apple M-series Mac).

Source

The `data-row/runtimes.R` script in the [logitr repository](#).

Examples

```

data(loglik_draws)

head(loglik_draws)

```

miscmethods.logitr	<i>Methods for logitr objects</i>
--------------------	-----------------------------------

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"),
  ...
)
```

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

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	<i>Extracting the Model Frame from a Formula or Fit</i>
--------------------	---

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	<i>Construct Design Matrices</i>
---------------------	----------------------------------

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	<i>Predict probabilities and / or outcomes</i>
----------------	--

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 <code>logitr</code> (a model estimated using the <code>'logitr()'</code> function).
newdata	a <code>data.frame</code> . Each row is an alternative and each column an attribute corresponding to parameter names in the estimated model. Defaults to <code>NULL</code> , 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 <code>newdata != NULL</code> . Defaults to <code>NULL</code> , in which case the value for <code>obsID</code> from the data in <code>object</code> is used.
type	A character vector defining what to predict: <code>prob</code> for probabilities, <code>outcomes</code> for outcomes. If you want both outputs, use <code>c("prob", "outcome")</code> . Outcomes are predicted randomly according to the predicted probabilities. Defaults to <code>"prob"</code> .
returnData	If <code>TRUE</code> the data is also returned, otherwise only the predicted values (<code>"prob"</code> and / or <code>"outcome"</code>) are returned. Defaults to <code>FALSE</code> .
interval	Type of interval calculation: <code>"none"</code> (default) or <code>"confidence"</code> . Future versions will include <code>"prediction"</code> 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 <code>scalePar</code> variable in <code>pars</code> .
scalePar	The name of the column that identifies the scale variable, which is typically <code>"price"</code> for WTP space models, but could be any continuous variable, such as <code>"time"</code> . Defaults to <code>NULL</code> .

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	<i>Returns a list of the design matrix X and updated pars and randPars to include any dummy-coded categorical or interaction variables.</i>
------------	---

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	<i>Extract Model Residuals</i>
------------------	--------------------------------

Description

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

Usage

```

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

```

Arguments

<code>object</code>	is an object of class <code>logitr</code> (a model estimated using the <code>'logitr()'</code> function).
<code>fitted</code>	Fitted values for an object of class <code>logitr</code> to use in computing residuals. Defaults to <code>NULL</code> .
<code>...</code>	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, gmn1, and apollo. Packages that support parallel estimation (logitr, mixl, and apollo) are benchmarked at multiple core counts, indicated in parentheses in the package name (e.g. "logitr (1 cores)"). The benchmark is generated by the script data-raw/runtimes.R in the package repository. The shipped results were produced on a 10-core Apple M-series Mac, so the absolute times (and the core counts benchmarked) are specific to that machine.

Usage

```
data(runtimes)
```

Format

Variable	Description
package	Package name (with the number of cores in parentheses for packages benchmarked in parallel).
time_sec	The estimation time in seconds.
numDraws	The number of random draws used during estimation.
version	The installed version of the package used for the benchmark.

Source

The data-raw/runtimes.R script in the [logitr repository](#).

Examples

```
data(runtimes)
```

```
head(runtimes)
```

runtimes_draws	<i>Data frame of logitr run times as the number of draws grows</i>
----------------	--

Description

This data frame contains full-fit estimation times for a preference space mixed logit model estimated with logitr as the number of random draws grows to 10,000, comparing the three logitr backends: the native R "cpu" path, the compiled "cpp" backend on a single core, and the compiled backend on all cores. Unlike [runtimes](#), this is a logitr-only benchmark used to show how logitr scales to draw counts the other packages cannot reach in tractable time. It is generated by the script `data-raw/runtimes.R` in the package repository, run on the same 10-core Apple M-series Mac as the [runtimes](#) benchmark, so the absolute times are specific to that machine.

Usage

```
data(runtimes_draws)
```

Format

Variable	Description
<code>config</code>	The logitr backend used: "cpu", "cpp (1 core)", or "cpp (10 cores)" (all cores on the benchmark machine).
<code>time_sec</code>	The estimation time in seconds.
<code>numDraws</code>	The number of random draws used during estimation.

Source

The `data-raw/runtimes.R` script in the [logitr repository](#).

Examples

```
data(runtimes_draws)

head(runtimes_draws)
```

se	<i>Extract standard errors</i>
----	--------------------------------

Description

Extract standard errors

Usage

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

Arguments

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

se.logitr	<i>Extract standard errors</i>
-----------	--------------------------------

Description

Extract standard errors

Usage

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

Arguments

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

statusCodes	<i>View a description the nloptr status codes</i>
-------------	---

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	<i>Tidy a logitr class object</i>
-------------	-----------------------------------

Description

Tidy a logitr class object

Usage

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

Arguments

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

 validate_data

Validate data formatting for logitr models

Description

This function checks that data is properly formatted for use with logitr() and provides detailed diagnostic information about potential issues.

Usage

```
validate_data(
  data,
  outcome,
  obsID,
  pars = NULL,
  scalePar = NULL,
  panelID = NULL
)
```

Arguments

data	The data frame to validate
outcome	The name of the column that identifies the outcome variable
obsID	The name of the column that identifies each observation
pars	Optional. The names of parameters to check (for additional validation)
scalePar	Optional. The name of the scale parameter column (for WTP models)
panelID	Optional. The name of the panel ID column (for panel data)

Value

An object of class 'logitr_validation' containing validation results

Examples

```
library(logitr)

# Validate the yogurt dataset
validate_data(yogurt, outcome = "choice", obsID = "obsID")

# Validate with parameters specified
validate_data(yogurt, outcome = "choice", obsID = "obsID",
              pars = c("price", "feat", "brand"))
```

vcov.logitr	<i>Calculate the variance-covariance matrix</i>
-------------	---

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	<i>Get WTP estimates a preference space model</i>
-----	---

Description

Returns the computed WTP from a preference space model.

Usage

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

Arguments

object	is an object of class <code>logitr</code> (a model estimated using the <code>'logitr()'</code> 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 <code>logitr</code> (a model estimated using the <code>'logitr()'</code> 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 <code>logitr()</code> function.
model_wtp	The output of a "willingness to pay space" model estimated using the <code>logitr()</code> 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](https://doi.org/10.1080/07350015.1994.10524547)

Examples

```
data(yogurt)
```

```
head(yogurt)
```

Index

- * **codes**
 - statusCodes, 31
- * **confint**
 - confint.logitr, 8
- * **datasets**
 - apolloModeChoiceData, 3
 - cars_china, 5
 - cars_us, 6
 - electricity, 9
 - loglik_draws, 20
 - runtimes, 28
 - runtimes_draws, 29
 - yogurt, 36
- * **fitted.values**
 - fitted.logitr, 10
- * **fitted**
 - fitted.logitr, 10
- * **logitr**
 - confint.logitr, 8
 - fitted.logitr, 10
 - logitr, 13
 - model.frame.logitr, 22
 - model.matrix.logitr, 23
 - predict.logitr, 23
 - residuals.logitr, 27
 - statusCodes, 31
 - wtp, 33
 - wtp.logitr, 34
 - wtpCompare, 35
- * **logit**
 - logitr, 13
- * **mixed**
 - logitr, 13
- * **nnl**
 - logitr, 13
- * **model.frame**
 - model.frame.logitr, 22
- * **model.matrix**
 - model.matrix.logitr, 23
- * **mxl**
 - logitr, 13
- * **nloptr**
 - statusCodes, 31
- * **predict**
 - predict.logitr, 23
- * **probabilities**
 - predict.logitr, 23
- * **residuals**
 - residuals.logitr, 27
- * **resid**
 - residuals.logitr, 27
- * **status**
 - statusCodes, 31
- * **willingness-to-pay**
 - logitr, 13
- * **wtp**
 - logitr, 13
 - wtp, 33
 - wtp.logitr, 34
 - wtpCompare, 35
- apolloModeChoiceData, 3
- augment.logitr, 4
- cars_china, 5
- cars_us, 6
- ci, 8
- coef.logitr (miscmethods.logitr), 21
- coef.summary.logitr (miscmethods.logitr), 21
- confint.logitr, 8
- electricity, 9
- fitted.logitr, 10
- fquantile, 11
- glance.logitr, 12
- logit_probs, 18

logitr, [13](#)
logLik.logitr(miscmethods.logitr), [21](#)
loglik_draws, [20](#)

miscmethods.logitr, [21](#)
model.frame.logitr, [22](#)
model.matrix.logitr, [23](#)

predict.logitr, [23](#)
print.logitr(miscmethods.logitr), [21](#)
print.logitr_validation
 (miscmethods.logitr), [21](#)
print.logitr_wtp(miscmethods.logitr),
 [21](#)
print.summary.logitr
 (miscmethods.logitr), [21](#)

recodeData, [26](#)
residuals.logitr, [27](#)
runtimes, [20](#), [28](#), [29](#)
runtimes_draws, [20](#), [29](#)

se, [30](#)
se.logitr, [30](#)
statusCodes, [31](#)
statusCodes(), [17](#)
summary.logitr(miscmethods.logitr), [21](#)

terms.logitr(miscmethods.logitr), [21](#)
tibble::tibble(), [31](#)
tidy.logitr, [31](#)

validate_data, [32](#)
vcov.logitr, [33](#)

wtp, [33](#)
wtp.logitr, [34](#)
wtpCompare, [35](#)

yogurt, [36](#)