Package 'beezdemand'

March 3, 2026

Version 0.2.0

Date 2026-02-27

Title Behavioral Economic Easy Demand

Author Brent Kaplan [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3758-6776>),
Shawn Gilroy [ctb]

Maintainer Brent Kaplan <bkaplan.ku@gmail.com>

Description Facilitates many of the analyses performed in studies of behavioral economic demand. The package supports commonly-used options for modeling operant demand including (1) data screening proposed by Stein, Koffarnus, Snider, Quisenberry, & Bickel (2015; <doi:10.1037/pha0000020>), (2) fitting models of demand such as linear (Hursh, Raslear, Bauman, & Black, 1989, <doi:10.1007/978-94-009-2470-3_22>), exponential (Hursh & Silberberg, 2008, <doi:10.1037/0033-295X.115.1.186>) and modified exponential (Koffarnus, Franck, Stein, & Bickel, 2015, <doi:10.1037/pha0000045>), and (3) calculating numerous measures relevant to applied behavioral economists (Intensity, Pmax, Omax). Also supports plotting and comparing data.

Depends R (>= 4.1.0)

Imports nlsr, nlstools, nls2, ggplot2, stats, optimx, broom, lme4, emmeans, minpack.lm, nls.multstart, performance, scales, tibble, lifecycle, dplyr, tidyr, nlme, rlang, utils, TMB

Suggests broom.mixed, GGally, knitr, tidyverse, rmarkdown, purrr, conflicted, devtools, here, readr, patchwork, testthat (>= 3.0.0)

LinkingTo TMB, RcppEigen

License GPL (>= 2)

URL https://brentkaplan.github.io/beezdemand/, https://github.com/brentkaplan/beezdemand

BugReports https://github.com/brentkaplan/beezdemand/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

SystemRequirements GNU make

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-03-03 09:00:19 UTC

Contents

`AIC.beezdemand_hurdle` *AIC for Hurdle Demand Model*

Description

AIC for Hurdle Demand Model

Usage

```
## S3 method for class 'beezdemand_hurdle'
AIC(object, ..., k = 2)
```

Arguments

`object`	A `beezdemand_hurdle` object.
`...`	Additional arguments (unused).
`k`	Penalty per parameter. Default is 2 (standard AIC).

Value

A numeric AIC value.

annotation_logticks2 *annotation_logticks2*

Description

Creates annotation layer

Usage

```
annotation_logticks2(
  base = 10,
  sides = "bl",
  scaled = TRUE,
  short = unit(0.1, "cm"),
  mid = unit(0.2, "cm"),
  long = unit(0.3, "cm"),
  colour = "black",
  size = 0.5,
  linetype = 1,
  alpha = 1,
  data = data.frame(x = NA),
  color = NULL,
  ...
)
```

Arguments

base	base for drawing in scale
sides	sides to draw, by default bottom and left
scaled	true by default
short	short tick settings
mid	mid tick settings
long	long tick settings
colour	default to black colour
size	size for labels
linetype	default linetype
alpha	default alpha level
data	data to include
color	colors to include
...	additional arguments

Details

Inherit and extend layer for use in ggplot draw

Value

ggplot2 layer

Author(s)

Shawn Gilroy shawn.gilroy@temple.edu

`anova.beezdemand_hurdle`

ANOVA Method for Hurdle Demand Models

Description

Compare nested hurdle demand models using likelihood ratio tests.

Usage

```
## S3 method for class 'beezdemand_hurdle'
anova(object, ...)
```

Arguments

`object`	A `beezdemand_hurdle` model.
`...`	Additional `beezdemand_hurdle` models to compare.

Details

All models must be fit to the same data. Models are ordered by degrees of freedom, and sequential likelihood ratio tests are performed.

Value

An object of class `anova.beezdemand` containing:

table Data frame with model comparison statistics

lrt Likelihood ratio test results

Examples

```
data(apt)
fit2 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0"))
fit3 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0", "alpha"))
anova(fit2, fit3)
```

anova.beezdemand_nlme *ANOVA Method for NLME Demand Models*

Description

Compare nested NLME demand models using likelihood ratio tests.

Usage

```
## S3 method for class 'beezdemand_nlme'
anova(object, ...)
```

Arguments

object	A beezdemand_nlme model.
...	Additional beezdemand_nlme models to compare.

Details

For NLME models, this method delegates to nlme::anova.lme() on the underlying model objects when possible.

Value

An object of class anova.beezdemand containing model comparison statistics.

Examples

```
data(ko)
fit1 <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                         id_var = "monkey", equation_form = "zben",
                         random_effects = Q0 ~ 1)
fit2 <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                         id_var = "monkey", equation_form = "zben",
                         random_effects = Q0 + alpha ~ 1)
anova(fit1, fit2)
```

apt *Example alcohol purchase task data*

Description

A dataset containing alcohol purchase task data for a small number of participants

Usage

```
apt
```

Format

Long-form data.frame with columns: id, x, y. Participants were asked how many standard sized alcoholic beverages they would buy at various prices.

apt_full *Full alcohol purchase task dataset*

Description

A larger dataset containing alcohol purchase task data with demographic covariates. Suitable for testing hurdle models and mixed-effects models with covariates.

Usage

```
apt_full
```

Format

A data frame with 18,700 rows and 8 columns:

id Unique participant identifier (1-1100)

gender Participant gender (Male/Female)

age Participant age in years

binges Number of binge drinking episodes

totdrinks Total number of drinks consumed

tothours Total hours spent drinking

x Price point for the purchase task

y Number of drinks participant would purchase at price x

Examples

```
data(apt_full)
# Use a subset for quick demonstration
apt_sub <- apt_full[apt_full$id %in% unique(apt_full$id)[1:20], ]
fit <- fit_demand_hurdle(apt_sub, y_var = "y", x_var = "x", id_var = "id")
summary(fit)
```

`augment.beezdemand_fixed` *Augment a beezdemand_fixed Model with Fitted Values and Residuals*

Description

Returns the original data with fitted values and residuals from individual demand curve fits. This enables easy model diagnostics and visualization with the tidyverse.

Usage

```
## S3 method for class 'beezdemand_fixed'
augment(x, newdata = NULL, ...)
```

Arguments

`x`	An object of class `beezdemand_fixed`.
`newdata`	Optional data frame of new data for prediction. If NULL, uses the original data from the model.
`...`	Additional arguments (currently unused).

Details

For "hs" equation models where fitting is done on the log10 scale, fitted values are back-transformed to the natural scale.

Value

A tibble containing the original data plus:

.fitted Fitted demand values on the response scale

.resid Residuals (observed - fitted)

Examples

```
data(apt)
fit <- fit_demand_fixed(apt, y_var = "y", x_var = "x", id_var = "id")
augmented <- augment(fit)

# Plot residuals by subject
library(ggplot2)
ggplot(augmented, aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.5) +
  facet_wrap(~id) +
  geom_hline(yintercept = 0, linetype = "dashed")
```

`augment.beezdemand_hurdle` *Augment a beezdemand_hurdle Model with Fitted Values and Residuals*

Description

Returns the original data with fitted values, residuals, and predictions from a hurdle demand model. This enables easy model diagnostics and visualization with the tidyverse.

Usage

```
## S3 method for class 'beezdemand_hurdle'
augment(x, newdata = NULL, ...)
```

Arguments

`x`	An object of class `beezdemand_hurdle`.
`newdata`	Optional data frame of new data for prediction. If NULL, uses the original data from the model.
`...`	Additional arguments (currently unused).

Details

For two-part hurdle models:

- `.fitted` gives predicted demand on the natural consumption scale
- `.fitted_prob` gives the predicted probability of positive consumption
- `.resid` is defined only for positive observations as log(y) - .fitted_link
- Observations with zero consumption have `.resid` = `NA` since they are explained by Part I (the zero-probability component), not Part II

Value

A tibble containing the original data plus:

.fitted Fitted demand values (natural scale)

.fitted_link Fitted values on log scale (Part II mean)

.fitted_prob Predicted probability of consumption (1 - P(zero))

.resid Residuals on log scale for positive observations, NA for zeros

.resid_response Residuals on response scale (y - .fitted)

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
augmented <- augment(fit)

# Plot residuals
library(ggplot2)
ggplot(augmented, aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed")
```

`augment.beezdemand_nlme` *Augment a beezdemand_nlme Model with Fitted Values and Residuals*

Description

Returns the original data with fitted values and residuals from a nonlinear mixed-effects demand model. This enables easy model diagnostics and visualization with the tidyverse.

Usage

```
## S3 method for class 'beezdemand_nlme'
augment(x, newdata = NULL, ...)
```

Arguments

`x`	An object of class `beezdemand_nlme`.
`newdata`	Optional data frame of new data for prediction. If NULL, uses the original data from the model.
`...`	Additional arguments (currently unused).

Details

The fitted values and residuals are on the same scale as the response variable used in the model. For `equation_form = "zben"`, this is the LL4-transformed scale. For `equation_form = "simplified"` or `"exponentiated"`, this is the natural consumption scale.

To back-transform predictions to the natural scale for "zben" models, use: `ll4_inv(augmented$.fitted)`

Value

A tibble containing the original data plus:

.fitted Fitted values on the model scale (may be transformed, e.g., LL4)

.resid Residuals on the model scale

.fixed Fitted values from fixed effects only (population-level)

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", factors = "dose", equation_form = "zben")
augmented <- augment(fit)

# Plot residuals
library(ggplot2)
ggplot(augmented, aes(x = .fitted, y = .resid)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed")
```

---

beezdemand_descriptive_methods

S3 Methods for beezdemand_descriptive Objects

Description

Methods for printing, summarizing, and visualizing objects of class `beezdemand_descriptive` created by `get_descriptive_summary()`.

Usage

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

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

## S3 method for class 'beezdemand_descriptive'
plot(x, x_trans = "identity", y_trans = "identity", show_zeros = FALSE, ...)
```

Arguments

x, object	A beezdemand_descriptive object
...	Additional arguments (currently unused)
x_trans	Character string specifying x-axis transformation. Options: "identity" (default), "log10", "log", "sqrt". See scales::transform_log10() etc.
y_trans	Character string specifying y-axis transformation. Options: "identity" (default), "log10", "log", "sqrt", "pseudo_log" (signed log).
show_zeros	Logical indicating whether to show proportion of zeros as labels on the boxplot (default: FALSE)

Details

Print Method:

Displays a compact summary showing the number of subjects and prices analyzed, plus a preview of the statistics table.

Summary Method:

Provides extended information including:

- Data summary (subjects, prices analyzed)
- Distribution of means across prices (min, median, max)
- Proportion of zeros by price (range)
- Missing data summary

Plot Method:

Creates a boxplot showing the distribution of consumption at each price point. Features:

- Red cross markers indicate means
- Boxes show median and quartiles
- Whiskers extend to 1.5 * IQR
- Supports axis transformations (log, sqrt, etc.)
- Uses modern beezdemand styling via theme_apa()

Value

- print() - Returns the object invisibly (called for side effects)
- summary() - Returns a list with extended summary information
- plot() - Returns a ggplot2 object

See Also

get_descriptive_summary()

Examples

```
data(apt, package = "beezdemand")
desc <- get_descriptive_summary(apt)

# Print compact summary
print(desc)

# Extended summary
summary(desc)

# Default boxplot
plot(desc)

# With log-transformed y-axis
plot(desc, y_trans = "log10")

# With pseudo-log y-axis (handles zeros gracefully)
plot(desc, y_trans = "pseudo_log")
```

`beezdemand_empirical_methods`

S3 Methods for beezdemand_empirical Objects

Description

Methods for printing, summarizing, and visualizing objects of class `beezdemand_empirical` created by `get_empirical_measures()`.

Usage

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

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

## S3 method for class 'beezdemand_empirical'
plot(x, type = "histogram", ...)
```

Arguments

`x, object`	A `beezdemand_empirical` object
`...`	Additional arguments passed to plotting functions
`type`	Character string specifying plot type. Options: • "histogram" (default) - Faceted histograms showing distribution of each measure

- "matrix" - Scatterplot matrix showing pairwise relationships between measures

Details

Print Method:
Displays a compact summary showing the number of subjects analyzed and a preview of the empirical measures table.

Summary Method:
Provides extended information including:

- Data summary (subjects, zero consumption patterns, completeness)
- Descriptive statistics for each empirical measure (min, median, mean, max, SD)
- Missing data patterns

Plot Method:
Creates visualizations of empirical measures across subjects.
Histogram type (default):

- Six-panel faceted plot showing distribution of each measure
- Helps identify central tendencies and outliers
- Uses modern beezdemand styling

Matrix type:

- Scatterplot matrix (pairs plot) showing relationships between measures
- Useful for identifying correlated demand metrics
- Lower triangle: scatterplots with smoothed trend lines
- Diagonal: density plots
- Upper triangle: correlation coefficients

Value

- `print()` - Returns the object invisibly (called for side effects)
- `summary()` - Returns a list with extended summary information
- `plot()` - Returns a ggplot2 object

See Also

`get_empirical_measures()`

Examples

```
data(apt, package = "beezdemand")
emp <- get_empirical_measures(apt)

# Print compact summary
print(emp)

# Extended summary
```

```
summary(emp)

# Histogram of measure distributions
plot(emp)

# Scatterplot matrix
plot(emp, type = "matrix")
```

BIC.beezdemand_hurdle *BIC for Hurdle Demand Model*

Description

BIC for Hurdle Demand Model

Usage

```
## S3 method for class 'beezdemand_hurdle'
BIC(object, ...)
```

Arguments

object	A beezdemand_hurdle object.
...	Additional arguments (unused).

Value

A numeric BIC value.

---

calculate_amplitude_persistence *Calculate Amplitude and Persistence*

Description

Calculates Amplitude and Persistence latent factors from demand metrics for various beezdemand model objects.

Usage

```
calculate_amplitude_persistence(
  fit,
  amplitude = c("Intensity", "Q0d", "Q0"),
  persistence = c("BP0", "Pmaxe", "Omaxe", "Alpha"),
  use_inv_alpha = TRUE,
  strict = TRUE,
  min_persistence_components = 2L,
  empirical_y_var = NULL,
  basis_means = NULL,
  basis_sds = NULL,
  ...
)
```

Arguments

`fit`	An object of class `beezdemand_fixed`, `beezdemand_hurdle`, `beezdemand_nlme`, or a `data.frame`.
`amplitude`	Character vector of column names to consider for the Amplitude factor. The function will use the first column found in the data. Default is `c("Intensity", "Q0d", "Q0")`.
`persistence`	Character vector of column names to include in the Persistence factor. Default is `c("BP0", "Pmaxe", "Omaxe", "Alpha")`. All found columns will be used.
`use_inv_alpha`	Logical. If "Alpha" (or a variation) is present in `persistence`, should it be inverted (1/Alpha) before standardization? Default is `TRUE`.
`strict`	Logical. If `TRUE` (default), missing metrics, duplicated `id`s, and other data integrity problems produce errors instead of warnings.
`min_persistence_components`	Integer. Minimum number of non-missing standardized persistence components required to compute `Persistence` for a given `id`. If fewer are available, `Persistence` is set to `NA`. Default is `2`.
`empirical_y_var`	For `beezdemand_nlme` objects, optional column name in `fit$data` to use when computing empirical indices (e.g., `BP0`). This is important when the fitted `y_var` is transformed (e.g., log10). If `NULL`, the method will attempt to choose a sensible default and may error in `strict` mode if ambiguous.
`basis_means`	Optional named numeric vector of means to use for Z-score standardization. Names must match the columns used (e.g., `c(Intensity = 10, BP0 = 5)`). If NULL (default), the sample means are used.
`basis_sds`	Optional named numeric vector of standard deviations to use for Z-score standardization. Names must match the columns used. If NULL (default), the sample SDs are used.
`...`	Additional arguments passed to methods.

Details

This function calculates Amplitude and Persistence by:

1. Extracting the relevant demand metrics from the `fit` object.
2. Resolving requested metric columns (case-insensitive, with limited synonym support for common beezdemand outputs).
3. Inverting Alpha if requested (1/Alpha).
4. Standardizing (Z-scoring) the metrics. If `basis_means` and `basis_sds` are provided, they are used; otherwise, the current sample's statistics are used.
5. Aggregating the Z-scores into the two latent factors.

Amplitude is defined by the variable specified in `amplitude` (typically Intensity/Q0). **Persistence** is defined as the mean of the standardized values of the variables specified in `persistence` (typically Breakpoint, Pmax, Omax, and 1/Alpha).

Value

A data frame with the original ID and calculated Amplitude and Persistence scores, along with the standardized (Z-scored) constituent metrics.

Models

- **beezdemand_fixed**: Extracts metrics from `fit$results`.
- **beezdemand_hurdle**: Extracts metrics from `fit$subject_pars`.
- **beezdemand_nlme**: Calculates subject-specific parameters from fixed and random effects. Parameters `Q0` and `Alpha` are assumed to be on `log10` scale for `zben` and `simplified` equations and are converted to linear scale. Omax and Pmax are calculated empirically from predictions. Breakpoint is calculated empirically from the raw data.

Examples

```
data(apt, package = "beezdemand")
fit <- FitCurves(apt, "hs", k = "share")
calculate_amplitude_persistence(fit)
```

---

calc_group_metrics *Calculate Group-Level Demand Metrics*

Description

Calculates group-level (population) Omax and Pmax from a fitted hurdle demand model.

Usage

```
calc_group_metrics(object)
```

Arguments

object	A fitted beezdemand_hurdle object.

Value

A named list with group-level Pmax, Omax, and Qmax.

See Also

calc_omax_pmax, fit_demand_hurdle

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
calc_group_metrics(fit)
```

---

calc_observed_pmax_omax

Calculate Observed Pmax/Omax Grouped by ID

Description

Calculate Observed Pmax/Omax Grouped by ID

Usage

```
calc_observed_pmax_omax(
  data,
  id_var = "id",
  price_var = "x",
  consumption_var = "y"
)
```

Arguments

data	Data frame with id, price, and consumption columns
id_var	Name of ID column
price_var	Name of price column
consumption_var	Name of consumption column

Value

Data frame with observed pmax/omax for each subject

Examples

```
data(apt, package = "beezdemand")
calc_observed_pmax_omax(apt, id_var = "id", price_var = "x", consumption_var = "y")
```

`calc_omax_pmax` *Calculate Omax and Pmax for Demand Curves*

Description

Calculates the maximum expenditure (Omax) and the price at maximum expenditure (Pmax) for the exponential demand model used in the two-part hurdle model.

Usage

```
calc_omax_pmax(Q0, k, alpha, price_range = NULL)
```

Arguments

`Q0`	Intensity parameter (consumption at price 0).
`k`	Scaling parameter for the exponential decay.
`alpha`	Elasticity parameter (rate of decay).
`price_range`	Numeric vector of length 2 specifying the price range to search for Pmax. Default is `NULL`, which uses an adaptive range based on alpha (approximately 0 to 10/alpha).

Details

For the demand function:

$$Q(p) = Q_0 \cdot \exp(k \cdot (\exp(-\alpha \cdot p) - 1))$$

Expenditure is E(p) = p * Q(p). Omax is the maximum of E(p) and Pmax is the price at which this maximum occurs. These are found numerically.

The search range is automatically adjusted based on alpha to ensure the maximum is found. For small alpha values, Pmax can be quite large.

Value

A named list with:

Pmax Price at maximum expenditure

Omax Maximum expenditure (price * quantity)

Qmax Quantity at Pmax

See Also

calc_group_metrics, fit_demand_hurdle

Examples

```
# Calculate for group-level parameters
calc_omax_pmax(Q0 = 10, k = 2, alpha = 0.5)

# With k >= e (~2.718), a local maximum exists
calc_omax_pmax(Q0 = 10, k = 3, alpha = 0.5)
```

---

cannabisCigarettes *Cannabis/cigarette cross-price responses*

Description

Cross-price style data with cannabis and cigarette context.

Usage

```
cannabisCigarettes
```

Format

A data frame with columns including: id, x, y, commodity, and auxiliary fields (Q1035, CigPrice, CanPrice, variable, value).

Examples

```
# data(cannabisCigarettes)
```

ChangeData *ChangeData*

Description

Changes demand data

Usage

```
ChangeData(
  dat,
  nrepl = 1,
  replnum = 0.01,
  rem0 = FALSE,
  remq0e = FALSE,
  replfree = NULL,
  xcol = "x",
  ycol = "y",
  idcol = "id"
)
```

Arguments

dat	A long form dataframe
nrepl	Number of zeros to replace with replacement value (replnum). Can accept either a number or "all" if all zeros should be replaced. Default is to replace the first zero only
replnum	Value to replace zeros. Default is .01
rem0	If TRUE, removes all 0s in consumption data prior to analysis. Default value is FALSE
remq0e	If TRUE, removes consumption and price where price == 0. Default value is FALSE
replfree	Optionally replaces price == 0 with specified value.
xcol	Column name in dataframe that signifies x values (usually price or the IV)
ycol	Column name in dataframe that signifies y values (usually consumption or the DV)
idcol	Column name in dataframe that signifies identifying id grouping

Details

Change demand data in various ways. Ways include replacing any number of 0 values with a replacement number (or remove them completely), removing price and consumption at free, replacing free with some number. This will soon replace ReplaceZeros and certain arguments in FitCurves.

Value

Long form dataframe resembling the originally provided dataframe

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## Change just the first instance of 0 within each unique value of id with .1
ChangeData(apt, nrepl = 1, replnum = .1)
```

CheckCols *Check Column Names*

Description

Checks to ensure column names are specified

Usage

```
CheckCols(dat, xcol, ycol, idcol, groupcol = NULL)
```

Arguments

dat	Dataframe
xcol	Name of x column
ycol	Name of y column
idcol	Name of id column
groupcol	Name of group column

Details

Check column names

Value

Dataframe

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
dat <- data.frame(price = 1:5, quantity = c(10, 8, 5, 2, 0), subj = rep(1, 5))
CheckCols(dat, xcol = "price", ycol = "quantity", idcol = "subj")
```

`CheckUnsystematic` *Systematic Purchase Task Data Checker*

Description

Applies Stein, Koffarnus, Snider, Quisenberry, & Bickel's (2015) criteria for identification of non-systematic purchase task data.

Usage

```
CheckUnsystematic(dat, deltaq = 0.025, bounce = 0.1, reversals = 0, ncons0 = 2)
```

Arguments

`dat`	Dataframe in long form. Colums are id, x, y.
`deltaq`	Numeric vector of length equal to one. The criterion by which the relative change in quantity purchased will be compared. Relative changes in quantity purchased below this criterion will be flagged. Default value is 0.025.
`bounce`	Numeric vector of length equal to one. The criterion by which the number of price-to-price increases in consumption that exceed 25% of initial consumption at the lowest price, expressed relative to the total number of price increments, will be compared. The relative number of price-to-price increases above this criterion will be flagged. Default value is 0.10.
`reversals`	Numeric vector of length equal to one. The criterion by which the number of reversals from number of consecutive (see ncons0) 0s will be compared. Number of reversals above this criterion will be flagged. Default value is 0.
`ncons0`	Numer of consecutive 0s prior to a positive value is used to flag for a reversal. Value can be either 1 (relatively more conservative) or 2 (default; as recommended by Stein et al., (2015).

Details

This function applies the 3 criteria proposed by Stein et al., (2015) for identification of nonsystematic purchase task data. The three criteria include trend (deltaq), bounce, and reversals from 0. Also reports number of positive consumption values.

Value

Dataframe

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## Using all default values
CheckUnsystematic(apt, deltaq = 0.025, bounce = 0.10, reversals = 0, ncons0 = 2)
## Specifying just 1 zero to flag as reversal
CheckUnsystematic(apt, deltaq = 0.025, bounce = 0.10, reversals = 0, ncons0 = 1)
```

---

check_demand_model *Check Demand Model Diagnostics*

Description

Performs diagnostic checks on fitted demand models, returning information about convergence, boundary conditions, and residual patterns.

Usage

```
check_demand_model(object, ...)

## S3 method for class 'beezdemand_hurdle'
check_demand_model(object, ...)

## S3 method for class 'beezdemand_nlme'
check_demand_model(object, ...)

## S3 method for class 'beezdemand_fixed'
check_demand_model(object, ...)
```

Arguments

`object`	A fitted model object of class `beezdemand_hurdle`, `beezdemand_nlme`, or `beezdemand_fixed`.
`...`	Additional arguments passed to methods.

Details

The function checks for:

- Convergence status and optimization messages
- Parameters at or near boundaries
- Residual patterns (heteroscedasticity, outliers)
- Random effect variance estimates near zero
- Correlation matrices near singularity

Value

An object of class `beezdemand_diagnostics` containing:

convergence List with convergence status and messages

boundary List with boundary condition warnings

residuals Summary statistics for residuals

random_effects Summary of random effects (if applicable)

issues Character vector of identified issues

recommendations Character vector of recommendations

Note

This function is named `check_demand_model()` to avoid potential conflicts with `performance::check_model()` from the performance package.

See Also

`plot_residuals()`, `plot_qq()`

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
diagnostics <- check_demand_model(fit)
print(diagnostics)
```

---

check_systematic_cp *Check Cross-Price Data for Unsystematic Responding*

Description

Modern interface for screening cross-price data with standardized output vocabulary aligned with `check_systematic_demand()`.

Usage

```
check_systematic_cp(
  data,
  trend_threshold = 0.025,
  bounce_threshold_down = 0.1,
  bounce_threshold_up = 0.1,
  bounce_threshold_none = 0.1,
  consecutive_zeros = 2,
  consecutive_nonzeros = 2,
```

```
  expected_down = FALSE,
  x_var = "x",
  y_var = "y",
  id_var = "id"
)
```

Arguments

data	Data frame with columns: id (optional), x (price), y (consumption).
trend_threshold	Numeric. Threshold for trend detection. Default 0.025.
bounce_threshold_down	Numeric. Bounce threshold for upward trends. Default 0.1.
bounce_threshold_up	Numeric. Bounce threshold for downward trends. Default 0.1.
bounce_threshold_none	Numeric. Bounce threshold when no trend. Default 0.1.
consecutive_zeros	Integer. Zeros for reversal detection. Default 2.
consecutive_nonzeros	Integer. Non-zeros for return detection. Default 2.
expected_down	Logical. Suppress reversal detection if TRUE. Default FALSE.
x_var	Character. Name of the price column. Default "x".
y_var	Character. Name of the consumption column. Default "y".
id_var	Character. Name of the subject identifier column. Default "id".

Details

If the data contains an id column (or column specified by id_var), each unique ID is checked separately. Otherwise, the entire dataset is treated as a single pattern.

For cross-price data, the wrapper preserves the legacy meaning of check_unsystematic_cp():

- trend_direction and bounce_direction are taken directly from the legacy function outputs.
- trend_pass is set to NA because cross-price systematicity does not use a separate trend "pass/fail" criterion in the same way as purchase-task screening; instead, trend classification determines which bounce rule applies.
- bounce_stat is reported as the proportion relevant to the legacy bounce rule for the detected trend_direction (or expected_down case), computed from the legacy bounce counts and the number of price steps.

Value

An object of class beezdemand_systematicity with the same structure as check_systematic_demand(), with type = "cp".

Examples

```
data(etm)
check <- check_systematic_cp(etm)
```

`check_systematic_demand` *Check Demand Data for Unsystematic Responding*

Description

Modern interface for screening purchase task data using Stein et al. (2015) criteria. Returns a structured object with standardized output vocabulary that is consistent with `check_systematic_cp()`.

Usage

```
check_systematic_demand(
  data,
  trend_threshold = 0.025,
  bounce_threshold = 0.1,
  max_reversals = 0,
  consecutive_zeros = 2,
  x_var = "x",
  y_var = "y",
  id_var = "id"
)
```

Arguments

`data`	Data frame in long format with columns: `id`, `x` (price), `y` (consumption).
`trend_threshold`	Numeric. Threshold for trend detection (log-log slope). Default `0.025`.
`bounce_threshold`	Numeric. Threshold for bounce proportion. Default `0.10`.
`max_reversals`	Integer. Maximum allowed reversals from zero. Default `0`.
`consecutive_zeros`	Integer. Consecutive zeros required for reversal detection. Default `2` (per Stein et al. 2015).
`x_var`	Character. Name of the price column. Default `"x"`.
`y_var`	Character. Name of the consumption column. Default `"y"`.
`id_var`	Character. Name of the subject identifier column. Default `"id"`.

Details

The `results` tibble contains standardized columns for both demand and cross-price systematicity checks:

id Subject identifier

type "demand" for this function

trend_stat DeltaQ statistic (log-log slope)

trend_threshold Threshold used

trend_direction "down", "up", or "none"

trend_pass Logical: passed trend criterion

bounce_stat Bounce proportion

bounce_threshold Threshold used

bounce_direction "significant" or "none"

bounce_pass Logical: passed bounce criterion

reversals Count of reversals from zero

reversals_pass Logical: passed reversals criterion

returns NA for demand (CP-specific)

n_positive Count of positive values

systematic Logical: passed all criteria

Value

An object of class `beezdemand_systematicity` with components:

results Tibble with one row per subject containing systematicity metrics

type "demand"

call The original function call

n_total Total number of subjects

n_systematic Number of subjects passing all criteria

n_unsystematic Number of subjects failing at least one criterion

Examples

```
data(apt)
check <- check_systematic_demand(apt)
print(check)
summary(check)
tidy(check)
```

`check_unsystematic_cp` *Check for Unsystematic Patterns in Cross-Price Data*

Description

Analyzes whether consumption data shows systematic trends or unsystematic patterns ("bounces") with respect to price. Includes detection of zero-value reversal/return sequences and allows flexible output based on the level of detail requested. See Rzeszutek et al. (in press) for more details.

Usage

```
check_unsystematic_cp(
  data,
  delta_threshold = 0.025,
  bounce_down_threshold = 0.1,
  bounce_up_threshold = 0.1,
  bounce_none_threshold = 0.1,
  rev_zeroes = 2,
  ret_nums = 2,
  expected_down = FALSE,
  verbose = FALSE,
  detailed = FALSE
)
```

Arguments

`data`	A data frame with columns 'x' and 'y', where 'x' is price and 'y' is consumption.
`delta_threshold`	Numeric. Threshold for detecting log-scale trends (default 0.025).
`bounce_down_threshold`	Numeric. Minimum downward bounce proportion to count as significant in upward trends.
`bounce_up_threshold`	Numeric. Minimum upward bounce proportion to count as significant in downward trends.
`bounce_none_threshold`	Numeric. Minimum bounce proportion to count as significant in no-trend cases.
`rev_zeroes`	Integer. Length of zero sequences to detect reversals (default 2).
`ret_nums`	Integer. Length of non-zero sequences to detect returns (default 2).
`expected_down`	Logical. If TRUE, suppress reversal detection.
`verbose`	Logical. If TRUE, print intermediate values (default FALSE).
`detailed`	Logical. If TRUE, return additional columns including all trend/bounce flags.

Value

A data frame of class `cp_unsystematic` with core results:

delta_direction Character: 'down', 'up', or 'none'.

bounce_direction Character: 'up', 'down', 'significant', or 'none'.

bounce_any Logical. TRUE if any bounce pattern detected.

bounce_above Integer. Number of upward changes meeting threshold.

bounce_below Integer. Number of downward changes meeting threshold.

reversals Integer. Detected reversals from 0 to non-0.

returns Integer. Detected returns from non-0 to 0.

If `detailed = TRUE`, returns additional columns:

delta_down Logical. Significant downward trend.

delta_up Logical. Significant upward trend.

delta_none Logical. No significant trend.

bounce_up Logical. Significant bounce up in a downward trend.

bounce_down Logical. Significant bounce down in an upward trend.

bounce_none Logical. Significant bounces in no-trend data.

Examples

```
x_seq <- 10^(seq(-2, 2, length.out = 10))
pattern <- data.frame(x = x_seq, y = c(10, 5, 10, 9, 10, 13, 10, 10, 7, 9))
check_unsystematic_cp(pattern)
```

`coef-methods` *Extract Coefficients from Cross-Price Demand Models*

Description

Methods to extract coefficients from various cross-price demand model objects.

Usage

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

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

## S3 method for class 'cp_model_lmer'
coef(object, fixed_only = FALSE, combine = TRUE, ...)
```

Arguments

object	A cp_model_lm object
...	Additional arguments (not used).
fixed_only	Logical; if TRUE, returns only fixed effects. Default is FALSE.
combine	Logical; if TRUE and fixed_only=FALSE, returns fixed + random effects combined. Default is TRUE.

Value

Named vector of coefficients

A named numeric vector of model coefficients.

Functions

- coef(cp_model_nls): Extract coefficients from a nonlinear cross-price model
- coef(cp_model_lm): Extract coefficients from a linear cross-price model
- coef(cp_model_lmer): Extract coefficients from a mixed-effects cross-price model

---

coef.beezdemand_fixed *Extract Coefficients from Fixed-Effect Demand Fit*

Description

Extract Coefficients from Fixed-Effect Demand Fit

Usage

```
## S3 method for class 'beezdemand_fixed'
coef(object, report_space = c("internal", "natural", "log10"), ...)
```

Arguments

object	A beezdemand_fixed object.
report_space	One of "internal", "natural", or "log10". Default "internal".
...	Unused.

Value

A tibble with columns id, term, estimate, estimate_scale, term_display.

coef.beezdemand_hurdle *Extract Coefficients from Hurdle Demand Model*

Description

Extract Coefficients from Hurdle Demand Model

Usage

```
## S3 method for class 'beezdemand_hurdle'
coef(object, report_space = c("natural", "log10", "internal"), ...)
```

Arguments

object	An object of class beezdemand_hurdle.
report_space	Character. One of "natural" (default), "log10", or "internal". Default is "natural" for consistency with tidy().
...	Additional arguments (currently unused).

Value

Named numeric vector of fixed effect coefficients.

coef.beezdemand_nlme *Extract Coefficients from a beezdemand_nlme Model*

Description

Provides methods to extract fixed effects, random effects, or subject-specific (combined fixed + random) coefficients from a beezdemand_nlme object. This is an S3 method for the generic coef function.

Usage

```
## S3 method for class 'beezdemand_nlme'
coef(
  object,
  type = "combined",
  report_space = c("internal", "natural", "log10"),
  ...
)
```

Arguments

object	A beezdemand_nlme object.
type	Character, type of coefficients to extract. One of: • "fixed": Returns only fixed effects (equivalent to fixef(object)). • "random": Returns only random effects (equivalent to ranef(object)). • "combined" (default): Returns subject-specific coefficients, where each subject's coefficient is the sum of the corresponding fixed effect and that subject's random effect deviation. This is equivalent to what stats::coef() on an nlme object returns.
report_space	Character. One of "internal" (default), "natural", or "log10".
...	Additional arguments passed to the underlying nlme coefficient extraction functions (nlme::fixef(), nlme::ranef(), or stats::coef.nlme()).

Value

Depending on type:

- type="fixed": A named numeric vector of fixed-effect coefficients.
- type="random": A data frame (or list of data frames if multiple levels of grouping) of random effects, as returned by ranef.nlme().
- type="combined": A data frame where rows are subjects (from id_var) and columns are the Q0 and alpha parameters, representing subject-specific estimates (on the log10 scale).

See Also

fixef.beezdemand_nlme, ranef.beezdemand_nlme

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", equation_form = "zben")
coef(fit, type = "fixed")
coef(fit, type = "random")
coef(fit, type = "combined")
```

---

compare_hurdle_models *Compare Nested Hurdle Demand Models*

Description

Performs a likelihood ratio test comparing two nested hurdle demand models. Typically used to test whether adding the random effect on alpha (c_i) significantly improves model fit (3-RE vs 2-RE models).

Usage

```
compare_hurdle_models(model_full, model_reduced)
```

Arguments

model_full	A beezdemand_hurdle object with 3 random effects.
model_reduced	A beezdemand_hurdle object with 2 random effects.

Value

Invisibly returns a list with:

lr_stat Likelihood ratio test statistic

df Degrees of freedom

p_value P-value from chi-squared distribution

model_comparison Data frame with model comparison statistics

See Also

fit_demand_hurdle

Examples

```
data(apt)
fit3 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0", "alpha"))
fit2 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0"))
compare_hurdle_models(fit3, fit2)
```

---

compare_models *Compare Demand Models*

Description

Compare multiple demand models using information criteria and likelihood ratio tests (when applicable). Works with all beezdemand model classes.

Usage

```
compare_models(..., test = c("auto", "lrt", "none"))
```

Arguments

`...`	Two or more model objects of class `beezdemand_hurdle`, `beezdemand_nlme`, or `beezdemand_fixed`.
`test`	Character; type of statistical test. One of: • `"auto"` (default): Use LRT if models are nested and comparable, otherwise IC-only. • `"lrt"`: Force likelihood ratio test (requires nested models from same backend). • `"none"`: Only report information criteria, no p-value.

Details

Models are compared using AIC and BIC. For models from the same statistical backend (e.g., two hurdle models or two NLME models), likelihood ratio tests can be performed if the models are nested.

When comparing models from different backends (e.g., hurdle vs NLME), only information criteria comparisons are possible since the likelihoods are not directly comparable for LRT purposes.

Backend Compatibility:

Backend 1	Backend 2	LRT Possible?
hurdle	hurdle	Yes (if nested)
nlme	nlme	Yes (if nested)
fixed	fixed	No (no likelihood)
hurdle	nlme	No
hurdle	fixed	No
nlme	fixed	No

Value

An object of class `beezdemand_model_comparison` containing:

comparison Data frame with model fit statistics

test_type Type of test performed

lrt_results LRT results if performed (NULL otherwise)

best_model Index of best model by BIC

notes Character vector of notes/warnings

nesting_verified Logical; always FALSE since nesting is not automatically verified. Users must ensure models are properly nested for valid LRT interpretation.

Statistical Notes

The likelihood ratio test (LRT) assumes that:

1. The models are **nested** (the reduced model is a special case of the full model obtained by constraining parameters).

2. Both models are fit to **identical data**.
3. Under the null hypothesis, the LR statistic follows a chi-square distribution with degrees of freedom equal to the difference in the number of parameters.

Important caveat for mixed-effects models: When variance components are tested at the boundary (e.g., testing whether a random effect variance is zero), the standard chi-square distribution is not appropriate. The correct null distribution is a mixture of chi-squares (Stram & Lee, 1994). The p-values reported here use the standard chi-square approximation, which is conservative (p-values are too large) for boundary tests.

This function does **not** automatically verify that models are nested. Users should ensure models are properly nested before interpreting LRT p-values.

References

Stram, D. O., & Lee, J. W. (1994). Variance components testing in the longitudinal mixed effects model. *Biometrics*, 50(4), 1171-1177.

See Also

`compare_hurdle_models()` for the legacy hurdle-specific comparison

Examples

```
data(apt)
fit2 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0"))
fit3 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0", "alpha"))
compare_models(fit2, fit3)
```

---

confint.beezdemand_fixed

Confidence Intervals for Fixed-Effect Demand Model Parameters

---

Description

Computes confidence intervals for Q0, alpha, and k parameters from individual demand curve fits. Uses asymptotic normal approximation based on standard errors when available.

Usage

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

Arguments

object	A beezdemand_fixed object from fit_demand_fixed().
parm	Character vector of parameter names to compute CIs for. Default includes all available parameters.
level	Confidence level (default 0.95).
...	Additional arguments (ignored).

Details

For beezdemand_fixed objects, confidence intervals are computed using the asymptotic normal approximation: estimate +/- z * SE. If standard errors are not available for a parameter, the confidence bounds will be NA.

When the underlying NLS fit objects are available (from detailed = TRUE), this method attempts to use nlstools::confint2() for more accurate profile-based intervals.

Value

A tibble with columns: id, term, estimate, conf.low, conf.high, level.

Examples

```
fit <- fit_demand_fixed(apt, equation = "hs", k = 2)
confint(fit)
confint(fit, level = 0.90)
confint(fit, parm = "Q0")
```

---

confint.beezdemand_hurdle

Confidence Intervals for Hurdle Demand Model Parameters

Description

Computes confidence intervals for fixed effect parameters from a TMB-based hurdle demand model using the asymptotic normal approximation.

Usage

```
## S3 method for class 'beezdemand_hurdle'
confint(
  object,
  parm = NULL,
  level = 0.95,
  report_space = c("internal", "natural"),
  ...
)
```

Arguments

`object`	A `beezdemand_hurdle` object from `fit_demand_hurdle()`.
`parm`	Character vector of parameter names to compute CIs for. Default includes all fixed effect parameters.
`level`	Confidence level (default 0.95).
`report_space`	Character. Reporting space for parameters: • `"internal"`: parameters on internal/fitting scale (log for Q0, alpha) • `"natural"`: back-transformed to natural scale
`...`	Additional arguments (ignored).

Details

Confidence intervals are computed using the asymptotic normal approximation based on standard errors from `TMB::sdreport()`. For parameters estimated on the log scale (Q0, alpha, k), intervals can be back-transformed to the natural scale using `report_space = "natural"`.

The transformation uses:

- For log-scale parameters: exp(estimate +/- z * SE)

Value

A tibble with columns: `term`, `estimate`, `conf.low`, `conf.high`, `level`, `component`, `estimate_scale`.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
confint(fit)
```

`confint.beezdemand_nlme` *Confidence Intervals for Mixed-Effects Demand Model Parameters*

Description

Computes confidence intervals for fixed effect parameters from an NLME-based mixed-effects demand model.

Usage

```
## S3 method for class 'beezdemand_nlme'
confint(object, parm = NULL, level = 0.95, method = c("wald", "profile"), ...)
```

Arguments

object	A beezdemand_nlme object from fit_demand_mixed().
parm	Character vector of parameter names to compute CIs for. Default includes all fixed effect parameters.
level	Confidence level (default 0.95).
method	Character. Method for computing intervals: • "wald": Wald-type intervals using asymptotic normality (default, fast) • "profile": Profile likelihood intervals via nlme::intervals() (slower but more accurate for small samples)
...	Additional arguments passed to nlme::intervals() when method = "profile".

Details

For Wald intervals, confidence bounds are computed as estimate ± z * SE using standard errors from the model summary.

For profile intervals, nlme::intervals() is called on the underlying nlme model object. This method provides more accurate intervals but can be computationally intensive for complex models.

Value

A tibble with columns: term, estimate, conf.low, conf.high, level, component.

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", equation_form = "zben")
confint(fit)
```

---

confint.cp_model_nls *Confidence Intervals for Cross-Price NLS Model Parameters*

Description

Computes confidence intervals for parameters from a nonlinear cross-price demand model using nlstools::confint2().

Usage

```
## S3 method for class 'cp_model_nls'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = c("asymptotic", "profile"),
  ...
)
```

Arguments

`object`	A `cp_model_nls` object from `fit_cp_nls()`.
`parm`	Character vector of parameter names to compute CIs for. Default includes all parameters.
`level`	Confidence level (default 0.95).
`method`	Character. Method for computing intervals passed to `nlstools::confint2()`: • `"asymptotic"` (default): Wald-type asymptotic intervals • `"profile"`: Profile-t confidence intervals
`...`	Additional arguments passed to `nlstools::confint2()`.

Details

This method wraps `nlstools::confint2()` to provide confidence intervals for the log10-parameterized coefficients (`log10_qalone`, `I`, `log10_beta`).

For back-transformed natural-scale confidence intervals, apply the transformation: `10^conf.low` and `10^conf.high` for log10-scale parameters.

Value

A tibble with columns: `term`, `estimate`, `conf.low`, `conf.high`, `level`, `method`.

Examples

```
data(etm)
fit <- fit_cp_nls(etm, equation = "exponentiated")
confint(fit)
```

cp *Example cross-price dataset*

Description

A small illustrative dataset of price (x) and consumption (y) target (target), and group (group).

Usage

```
cp
```

Format

A data frame with N rows and columns:

id unique identifier

x price of the alternative

y consumption/demand

target target (e.g., alone, own, alt)

group e.g., drug or product

cp_posthoc_intercepts *Run pairwise intercept comparisons for cross-price demand model*

Description

This function performs pairwise comparisons of intercepts between groups in a cross-price demand model, but only when a significant interaction is present. The emmeans table showing estimated marginal means for intercepts is always returned.

Usage

```
cp_posthoc_intercepts(object, alpha = 0.05, adjust = "tukey", ...)
```

Arguments

object	A cp_model_lmer object from fit_cp_linear
alpha	Significance level for testing (default: 0.05)
adjust	Method for p-value adjustment; see emmeans::contrast (default: "tukey")
...	Additional arguments passed to emmeans

Value

List containing the emmeans table and optionally pairwise comparisons if interaction is significant

Examples

```
data(etm)
fit <- fit_cp_linear(etm, type = "mixed", group_effects = TRUE)
cp_posthoc_intercepts(fit)
```

`cp_posthoc_slopes` *Run pairwise slope comparisons for cross-price demand model*

Description

This function performs pairwise comparisons of slopes between groups in a cross-price demand model, but only when a significant interaction is present. The emmeans table showing estimated marginal means for slopes is always returned.

Usage

```
cp_posthoc_slopes(object, alpha = 0.05, adjust = "tukey", ...)
```

Arguments

`object`	A cp_model_lmer object from fit_cp_linear
`alpha`	Significance level for testing (default: 0.05)
`adjust`	Method for p-value adjustment; see emmeans::contrast (default: "tukey")
`...`	Additional arguments passed to emmeans

Value

List containing the emmeans table and optionally pairwise comparisons if interaction is significant

Examples

```
data(etm)
fit <- fit_cp_linear(etm, type = "mixed", group_effects = TRUE)
cp_posthoc_slopes(fit)
```

etm *Example Experimental Tobacco Marketplace data*

Description

A dataset containing ETM data for a small number of participants

Usage

```
etm
```

Format

Long-form data.frame with columns: id, x, y, target, group. Participants were asked how many cigarettes, e-cigarettes, combustible, and non-combustible products they would buy at various prices.

extract_coefficients *Extract All Coefficient Types from Cross-Price Demand Models*

Description

A convenience function to extract coefficients from any type of cross-price demand model in a unified format. For mixed effects models, returns a list with different coefficient types.

Usage

```
extract_coefficients(object, ...)
```

Arguments

object	A cross-price demand model object (cp_model_nls, cp_model_lm, or cp_model_lmer)
...	Additional arguments passed to the appropriate coef method

Value

For cp_model_nls and cp_model_lm, returns the model coefficients. For cp_model_lmer, returns a list with fixed, random, and combined coefficients.

Examples

```
data(etm, package = "beezdemand")
fit <- fit_cp_nls(etm, equation = "exponentiated")
extract_coefficients(fit)
```

ExtraF *ExtraF*

Description

Extra Sum of Squares F-test

Usage

```
ExtraF(
  dat,
  equation = "hs",
  groups = NULL,
  verbose = FALSE,
  k,
  compare = "alpha",
  idcol = "id",
  xcol = "x",
  ycol = "y",
  groupcol = NULL,
  start_alpha = 0.001
)
```

Arguments

dat	Long form data frame
equation	"hs"
groups	NULL for all. Character vector matching groups in groupcol
verbose	If TRUE, prints all output including models
k	User-defined k value; if missing will attempt to find shared k and then mean emprirical range (in log units)
compare	Specify whether to compare alpha or Q0. Default is alpha
idcol	The column name that should be treated as dataset identifier
xcol	The column name that should be treated as "x" data
ycol	The column name that should be treated as "y" data
groupcol	The column name that should be treated as the groups
start_alpha	Optional numeric to inform starting value for alpha

Details

One alpha better than individual alphas?

Value

List of results and models

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## Compare two groups using equation by Koffarnus et al., 2015 and a fixed k of 2

apt$group <- NA
apt[apt$id %in% sample(unique(apt$id), length(unique(apt$id))/2), "group"] <- "a"
apt$group[is.na(apt$group)] <- "b"
ExtraF(apt, "koff", k = 2, groupcol = "group")
```

`FitCurves` *FitCurves*

Description

Analyzes purchase task data

Usage

```
FitCurves(
  dat,
  equation,
  k,
  agg = NULL,
  detailed = FALSE,
  xcol = "x",
  ycol = "y",
  idcol = "id",
  groupcol = NULL,
  lobound,
  hibound,
  constrainq0 = NULL,
  startq0 = NULL,
  startalpha = NULL,
  param_space = c("natural", "log10")
)
```

Arguments

`dat`	data frame (long form) of purchase task data.
`equation`	Character vector of length one. Accepts either "hs" for Hursh and Silberberg (2008) or "koff" for Koffarnus, Franck, Stein, and Bickel (2015).

k	A numeric (or character) vector of length one. Reflects the range of consumption in log10 units. If none provided, k will be calculated based on the max/min of the entire sample + .5. If k = "ind", k will be calculated per individual using max/min + .5. If k = "fit", k will be a free parameter on an individual basis. If k = "range", k will be calculated based on the max/min of the entire sample + .5.
agg	Character vector of length one accepts either "Mean" or "Pooled". If not NULL (default), data will be aggregrated appropriately and analyzed in the specified way.
detailed	If TRUE, output will be a 3 element list including (1) dataframe of results, (2) list of model objects, (3) list of individual dataframes used in fitting. Default value is FALSE, which returns only the dataframe of results.
xcol	The column name that should be treated as "x" data
ycol	The column name that should be treated as "y" data
idcol	The column name that should be treated as dataset identifier
groupcol	The column name that should be treated as the groups
lobound	Optional. A named vector of length 2 ("q0", "alpha") or 3 ("q0", "k", "alpha"), the latter length if k = "fit", specifying the lower bounds.
hibound	Optional. A named vector of length 2 ("q0", "alpha") or 3 ("q0", "k", "alpha"), the latter length if k = "fit", specifying the upper bounds.
constrainq0	Optional. A number that will be used to constrain Q0 in the fitting process. Currently experimental and only works with a fixed k value.
startq0	Optional. A number that will be used to start Q0 in the fitting process. Currently experimental.
startalpha	Optional. A number that will be used to start Alpha in the fitting process. Currently experimental.
param_space	Character. One of "natural" (default) or "log10". Specifies whether parameters (Q0, alpha) are estimated in natural space or log10-transformed space.

Details

[Superseded]

`FitCurves()` has been superseded by `fit_demand_fixed()`, which provides a modern S3 interface with standardized methods (`summary()`, `tidy()`, `glance()`, `predict()`). `FitCurves()` will continue to work but is no longer recommended for new code. See `vignette("migration-guide")` for migration instructions.

Value

If detailed == FALSE (default), a dataframe of results. If detailed == TRUE, a 3 element list consisting of (1) dataframe of results, (2) list of model objects, (3) list of individual dataframes used in fitting

Author(s)

Brent Kaplan bkaplan.ku@gmail.com Shawn Gilroy shawn.gilroy@temple.edu

See Also

fit_demand_fixed() for the modern interface

Examples

```
## Analyze using Hursh & Silberberg, 2008 equation with a k fixed to 2
FitCurves(apt[sample(apt$id, 5), ], "hs", k = 2)
```

FitMeanCurves *Fit Pooled/Mean Curves*

Description

Fits curve to pooled or mean data

Usage

```
FitMeanCurves(
  dat,
  equation,
  k,
  remq0e = FALSE,
  replfree = NULL,
  rem0 = FALSE,
  nrepl = NULL,
  replnum = NULL,
  plotcurves = FALSE,
  method = NULL,
  indpoints = TRUE,
  vartext = NULL
)
```

Arguments

dat	data frame (long form) of purchase task data.
equation	Character vector of length one. Accepts either "hs" for Hursh and Silberberg (2008) or "koff" for Koffarnus, Franck, Stein, and Bickel (2015).
k	A numeric vector of length one. Reflects the range of consumption in log10 units. If none provided, k will be calculated based on the max/min of the entire sample. If k = "fit", k will be a free parameter
remq0e	If TRUE, removes consumption and price where price == 0. Default value is FALSE
replfree	Optionally replaces price == 0 with specified value. Note, if fitting using equation == "hs", and 0 is first price, 0 gets replaced by replfree. Default value is .01

rem0	If TRUE, removes all 0s in consumption data prior to analysis. Default value is FALSE.
nrepl	Number of zeros to replace with replacement value (replnum). Can accept either a number or "all" if all zeros should be replaced. Default is to replace the first zero only.
replnum	Value to replace zeros. Default is .01
plotcurves	Boolean whether to create plot. If TRUE, a "plots/" directory is created one level above working directory. Default is FALSE.
method	Character string of length 1. Accepts "Mean" to fit to mean data or "Pooled" to fit to pooled data
indpoints	Boolean whether to plot individual points in gray. Default is TRUE.
vartext	Character vector specifying indices to report on plots. Valid indices include "Q0d", "Alpha", "Q0e", "EV", "Pmaxe", "Omaxe", "Pmaxd", "Omaxd", "K", "Q0se", "Alphase", "R2", "AbsSS"

Details

[Superseded]

`FitMeanCurves()` has been superseded by `fit_demand_fixed()` with the agg parameter. `FitMeanCurves()` will continue to work but is no longer recommended for new code. See `vignette("migration-guide")` for migration instructions.

Value

Data frame

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

See Also

`fit_demand_fixed()` for the modern interface with agg parameter

Examples

```
## Fit aggregated data (mean only) using Hursh & Silberberg, 2008 equation with a k fixed at 2
FitMeanCurves(apt[sample(apt$id, 5), ], "hs", k = 2, method = "Mean")
```

`fit_cp_linear` *Fit a Linear Cross-Price Demand Model*

Description

Fit a Linear Cross-Price Demand Model

Usage

```
fit_cp_linear(
  data,
  type = c("fixed", "mixed"),
  x_var = "x",
  y_var = "y",
  id_var = "id",
  group_var = "group",
  target_var = "target",
  filter_target = TRUE,
  target_level = "alt",
  formula = NULL,
  log10x = FALSE,
  group_effects = FALSE,
  random_slope = FALSE,
  return_all = TRUE,
  ...
)

fit_cp_linear.default(
  data,
  formula = NULL,
  log10x = FALSE,
  return_all = FALSE,
  ...
)

fit_cp_linear.mixed(
  data,
  formula = NULL,
  log10x = FALSE,
  return_all = FALSE,
  ...
)
```

Arguments

`data`	A data frame containing columns for price, consumption, and optionally target indicator, subject identifier, and group.

type	The type of model: "fixed" for standard linear or "mixed" for mixed effects.
x_var	Character string; name of the price column. Default is "x". Renamed to "x" internally.
y_var	Character string; name of the consumption column. Default is "y". Renamed to "y" internally.
id_var	Character string; name of the subject identifier column. Default is "id". Required for mixed models; renamed to "id" internally.
group_var	Character string; name of the group column. Default is "group". Renamed to "group" internally when present.
target_var	Character string; name of the target indicator column. Default is "target". Renamed to "target" internally when present.
filter_target	Logical; if TRUE (default), filters to rows where the target column equals target_level.
target_level	Character string; value of the target column to retain when filter_target = TRUE. Default is "alt".
formula	Optional formula override. If NULL, a formula will be constructed based on other parameters. If non-NULL and any *_var argument differs from its default, an error is thrown because the formula references canonical column names that no longer exist before renaming; rename columns before calling, or omit the formula argument.
log10x	Logical; if TRUE and formula is NULL, uses log10(x) instead of x in the formula. Default is FALSE.
group_effects	Logical or character; if TRUE, includes group as a factor with interactions. Can also be "intercept" for group intercepts only or "interaction" for full interactions. Default is FALSE.
random_slope	Logical; for mixed models, if TRUE, includes random slopes for x. Default is FALSE.
return_all	Logical; if TRUE, returns additional model metadata.
...	Additional arguments passed to underlying modeling functions.

Value

Fitted linear model.

Examples

```
data(etm)
## Fixed-effects linear cross-price model
fit_fixed <- fit_cp_linear(etm, type = "fixed", group_effects = TRUE)
summary(fit_fixed)

## Mixed-effects linear cross-price model
fit_mixed <- fit_cp_linear(etm, type = "mixed", group_effects = TRUE)
summary(fit_mixed)
```

`fit_cp_nls` *Fit cross-price demand with NLS (+ robust fallbacks)*

Description

Fits a **cross-price demand** curve using log10-parameterization for numerical stability. The optimizer estimates parameters on the log10 scale where applicable, ensuring positive constraints are naturally satisfied.

Equation forms:

- **Exponentiated** (default):
$$y = Q_{alone} \cdot 10^{I \cdot \exp(-\beta \cdot x)}$$
- **Exponential** (fits on log10 response scale):
$$\log_{10}(y) = \log_{10}(Q_{alone}) + I \cdot \exp(-\beta \cdot x)$$
- **Additive** (level on y):
$$y = Q_{alone} + I \cdot \exp(-\beta \cdot x)$$

where x is the alternative product price (or "cross" price) and y is consumption of the target good.

Optimizer parameters (log10 parameterization):

- `log10_qalone`: $\log_{10}(Q_{alone})$ - baseline consumption when the alternative is effectively absent.
- `I`: cross-price **interaction intensity**; sign and magnitude reflect substitution/complementarity. Unconstrained (can be negative for substitutes).
- `log10_beta`: $\log_{10}(\beta)$ - rate at which cross-price influence decays as x increases.

Natural-scale values are recovered as $Q_{alone} = 10^{log10\_qalone}$ and $\beta = 10^{log10\_beta}$.

The function first attempts a multi-start nonlinear least squares fit (`nls.multstart`). If that fails—or if explicit `start_values` are provided—it falls back to `minpack.lm::nlsLM`. Optionally, it will make a final attempt with `nlsr::wrapnlsr`. Returns either the fitted model or a structured object with metadata for downstream methods.

Usage

```
fit_cp_nls(
  data,
  equation = c("exponentiated", "exponential", "additive"),
  x_var = "x",
  y_var = "y",
  start_values = NULL,
  start_vals = lifecycle::deprecated(),
  iter = 100,
  bounds = NULL,
  fallback_to_nlsr = TRUE,
  return_all = TRUE
)
```

Arguments

data	A data frame with columns for price and consumption. Additional columns are ignored. Input is validated internally.
equation	Character string; model family, one of c("exponentiated", "exponential", "additive"). Default is "exponentiated".
x_var	Character string; name of the price column in data. Default is "x". The column is renamed to "x" internally; see validate_cp_data().
y_var	Character string; name of the consumption column in data. Default is "y". The column is renamed to "y" internally.
start_values	Optional **named list** of initial values for parameters log10_qalone, I, and log10_beta. If NULL, the function derives plausible ranges from the data and uses multi-start search.
start_vals	**[Deprecated]** Use start_values instead.
iter	Integer; number of random starts for nls.multstart (default 100).
bounds	Deprecated. Log10-parameterized parameters are naturally unbounded. This argument is ignored but retained for backwards compatibility.
fallback_to_nlsr	Logical; if TRUE (default), try nlsr::wrapnlsr when both multi-start NLS and nlsLM fail.
return_all	Logical; if TRUE (default), return a list containing the model and useful metadata. If FALSE, return the bare fitted model object.

Details

Start values. When start_values is missing, the function: (1) estimates a reasonable range for log10_qalone from the observed y, (2) estimates log10_beta from the price range, and (3) launches a multi-start grid in nls.multstart.

Zero handling for exponential equation. Since the exponential equation fits on the $\log_{10}(y)$ scale, observations with $y \leq 0$ are automatically removed with a warning. Use the exponentiated or additive forms if you need to retain zero consumption values.

Fitting pipeline (short-circuiting):

1. nls.multstart::nls_multstart() with random starts.
2. If that fails (or if start_values provided): minpack.lm::nlsLM() using start_values (user or internally estimated).
3. If that fails and fallback_to_nlsr = TRUE: nlsr::wrapnlsr().

The returned object has class "cp_model_nls" (when return_all = TRUE) with components: model, method (the algorithm used), equation, start_vals, nlsLM_fit, nlsr_fit, and the data used. This is convenient for custom print/summary/plot methods.

Value

If return_all = TRUE (default): a list of class "cp_model_nls":

- model: the fitted object from the successful backend.

- method: one of "nls_multstart", "nlsLM", or "wrapnlsr".
- equation: the model family used.
- start_vals: named list of starting values (final used; kept for backward compatibility).
- nlsLM_fit, nlsr_fit: fits from later stages (if attempted).
- data: the 2-column data frame actually fit.

If return_all = FALSE: the fitted model object from the successful backend.

Convergence & warnings

- Check convergence codes and residual diagnostics from the underlying fit.
- Poor scaling or extreme y dispersion can make parameters weakly identified.
- For "exponential", the model fits on the $\log_{10}(y)$ scale internally.

See Also

check_unsystematic_cp for pre-fit data screening, validate_cp_data for input validation.

Examples

```
## --- Example: Real data (E-Cigarettes, id = 1) ---
dat <- structure(list(
  id = c(1, 1, 1, 1, 1, 1),
  x = c(2, 4, 8, 16, 32, 64),
  y = c(3, 5, 5, 16, 17, 13),
  target = c("alt", "alt", "alt", "alt", "alt", "alt"),
  group = c("E-Cigarettes", "E-Cigarettes", "E-Cigarettes",
            "E-Cigarettes", "E-Cigarettes", "E-Cigarettes")
), row.names = c(NA, -6L), class = c("tbl_df", "tbl", "data.frame"))

## Fit the default (exponentiated) cross-price form
fit_ecig <- fit_cp_nls(dat, equation = "exponentiated", return_all = TRUE)
summary(fit_ecig)         # model summary
fit_ecig$method           # backend actually used (e.g., "nls_multstart")
coef(fit_ecig$model)      # parameter estimates: log10_qalone, I, log10_beta
```

---

fit_demand_fixed *Fit Fixed-Effect Demand Curves*

Description

Modern interface for fitting individual demand curves via nonlinear least squares. Returns a structured S3 object with standard methods including summary(), tidy(), and glance().

Usage

```
fit_demand_fixed(
  data,
 equation = c("hs", "koff", "simplified", "linear", "exponential", "exponentiated"),
  k = 2,
  agg = NULL,
  x_var = "x",
  y_var = "y",
  id_var = "id",
  param_space = c("natural", "log10"),
  ...
)
```

Arguments

data	Data frame in long format with columns: id, x (price), y (consumption).
equation	Character. Equation type: "hs" (Hursh & Silberberg, 2008), "koff" (Koffarnus et al., 2015), "simplified" (Rzeszutek et al., 2025; simplified exponential with normalized decay, no k parameter), or "linear". The modern aliases "exponential" (equivalent to "hs") and "exponentiated" (equivalent to "koff") are also accepted. Default "hs".
k	Scaling constant. Numeric value (fixed), "ind" (individual), "fit" (free parameter), or "range" (data-driven). Default 2.
agg	Character. Aggregation method: "Mean", "Pooled", or NULL for individual fits. Default NULL.
x_var	Character. Name of the price column. Default "x".
y_var	Character. Name of the consumption column. Default "y".
id_var	Character. Name of the subject identifier column. Default "id".
param_space	Character. Parameterization used for fitting. One of: • "natural": fit Q0, alpha (and k if k = "fit") on their natural scale • "log10": fit log10(Q0), log10(alpha) (and log10(k) if k = "fit")
...	Additional arguments passed to the underlying FitCurves() engine.

Details

This function is a modern wrapper around the legacy FitCurves() function. It provides the same fitting capabilities but returns a structured S3 object with standardized methods for model interrogation.

Value

An object of class beezdemand_fixed with components:

results Data frame of fitted parameters for each subject

fits List of model fit objects (if detailed = TRUE internally)

predictions List of prediction data frames

data_used List of data frames used for each fit

call The original function call

equation The equation form used

k_spec Description of k specification

agg Aggregation method used

n_total Total number of subjects/fits attempted

n_success Number of successful fits

n_fail Number of failed fits

Examples

```
data(apt)
fit <- fit_demand_fixed(apt, equation = "hs", k = 2)
print(fit)
summary(fit)
tidy(fit)
glance(fit)
```

`fit_demand_hurdle` *Fit Two-Part Mixed Effects Hurdle Demand Model*

Description

Fits a two-part hurdle model for demand data using TMB (Template Model Builder). Part I models the probability of zero consumption using logistic regression. Part II models log-consumption given positive response using a nonlinear mixed effects model.

Usage

```
fit_demand_hurdle(
  data,
  y_var,
  x_var,
  id_var,
  random_effects = c("zeros", "q0", "alpha"),
  epsilon = 0.001,
  start_values = NULL,
  tmb_control = list(max_iter = 200, eval_max = 1000, trace = 0),
  verbose = 1,
  part2 = c("zhao_exponential", "exponential", "simplified_exponential"),
  ...
)
```

Arguments

`data`	A data frame containing the demand data.
`y_var`	Character string specifying the column name for consumption values.
`x_var`	Character string specifying the column name for price.
`id_var`	Character string specifying the column name for subject IDs.
`random_effects`	Character vector specifying which random effects to include. Options are `"zeros"` (a_i for Part I), `"q0"` (b_i for intensity), and `"alpha"` (c_i for elasticity). Default is `c("zeros", "q0", "alpha")` for the full 3-random-effect model. Use `c("zeros", "q0")` for the simplified 2-random-effect model (fixed alpha across subjects).
`epsilon`	Small constant added to price before log transformation in Part I. Used to handle zero prices: `log(price + epsilon)`. Default is 0.001.
`start_values`	Optional named list of starting values for optimization. If `NULL` (default), sensible defaults are used.
`tmb_control`	List of control parameters for TMB optimization: **max_iter** Maximum number of optimization iterations (default 200) **eval_max** Maximum number of function evaluations (default 1000) **trace** Print optimization trace: 0 = none, 1 = some (default 0)
`verbose`	Integer controlling output verbosity: 0 = silent, 1 = progress messages, 2 = detailed optimization trace. Default is 1.
`part2`	Character string selecting the Part II mean function. Options are `"zhao_exponential"` (default; no Q0 normalization in the exponent), `"exponential"` (HS-standardized; Q0 inside the exponent), and `"simplified_exponential"` (SND/log-linear; no `k` parameter).
`...`	Additional arguments (reserved for future use).

Details

The model structure is:

Part I (Binary - probability of zero consumption):

$$logit(\pi_{ij}) = \beta_0 + \beta_1 \cdot \log(price + \epsilon) + a_i$$

Part II (Continuous - log consumption given positive):

With 3 random effects (`random_effects = c("zeros", "q0", "alpha")`):

$$\log(Q_{ij}) = (\log Q_0 + b_i) + k \cdot (\exp(-\alpha_i \cdot price) - 1) + \epsilon_{ij}$$

where $\alpha_i = \exp(\log(\alpha) + c_i)$ and $k = \exp(\log(k))$.

With 2 random effects (`random_effects = c("zeros", "q0")`):

$$\log(Q_{ij}) = (\log Q_0 + b_i) + k \cdot (\exp(-\alpha \cdot price) - 1) + \epsilon_{ij}$$

where $\alpha = \exp(\log(\alpha))$ and $k = \exp(\log(k))$.

Random effects follow a multivariate normal distribution with unstructured covariance matrix. Use `compare_hurdle_models` for likelihood ratio tests comparing nested models.

Value

An object of class `beezdemand_hurdle` containing:

model List with coefficients, se, variance_components, correlations

random_effects Matrix of empirical Bayes random effect estimates

subject_pars Data frame of subject-specific parameters including Q0, alpha, breakpoint, Pmax, Omax

tmb_obj TMB objective function object

opt Optimization result from `nlminb`

sdr TMB sdreport object

call The matched call

data Original data used for fitting

param_info List with y_var, x_var, id_var, n_subjects, n_obs, etc.

converged Logical indicating convergence

loglik Log-likelihood at convergence

AIC, BIC Information criteria

error_message Error message if fitting failed, NULL otherwise

Parameterization and comparability

The TMB backend estimates positive-constrained parameters on the natural-log scale: $\log(Q_0)$, $\log(\alpha)$, and $\log(k)$. Reporting methods (`summary()`, `tidy()`, `coef()`) can back-transform to the natural scale or present parameters on the $\log_{10}$ scale.

To compare α estimates with models fit in $\log_{10}$ space, use:

$$\log_{10}(\alpha) = \log(\alpha)/\log(10).$$

See Also

`summary.beezdemand_hurdle`, `predict.beezdemand_hurdle`, `plot.beezdemand_hurdle`, `compare_hurdle_models`, `simulate_hurdle_data`

Examples

```
# Load example data
data(apt)

# Fit full model with 3 random effects
fit3 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0", "alpha"))

# Fit simplified model with 2 random effects (fixed alpha)
fit2 <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id",
                          random_effects = c("zeros", "q0"))

# View results
```

```
summary(fit3)

# Compare models with likelihood ratio test
compare_hurdle_models(fit3, fit2)
```

`fit_demand_mixed` *Fit Nonlinear Mixed-Effects Demand Model*

Description

Fits a nonlinear mixed-effects model for behavioral economic demand data. The function allows Q0 and alpha parameters to vary by specified factors and supports different demand equation forms.

Usage

```
fit_demand_mixed(
  data,
  y_var,
  x_var,
  id_var,
  factors = NULL,
  factor_interaction = FALSE,
  equation_form = c("zben", "simplified", "exponentiated"),
  param_space = c("log10", "natural"),
  k = NULL,
  custom_model_formula = NULL,
  fixed_rhs = NULL,
  continuous_covariates = NULL,
  start_value_method = c("heuristic", "pooled_nls"),
  random_effects = Q0 + alpha ~ 1,
  covariance_structure = c("pdDiag", "pdSymm"),
  start_values = NULL,
  collapse_levels = NULL,
 nlme_control = list(msMaxIter = 200, niterEM = 100, maxIter = 200, pnlsTol = 0.001,
  tolerance = 1e-06, apVar = TRUE, minScale = 1e-09, opt = "nlminb", msVerbose = FALSE),
 method = "ML",
  ...
)
```

Arguments

`data`	A data frame.
`y_var`	Character string, the name of the dependent variable column. For `equation_form = "zben"`, this should be log-transformed consumption (e.g., "y_ll4"). For `equation_form = "simplified"` or `"exponentiated"`, this should be raw, untransformed consumption (e.g., "y").

`x_var` Character string, the name of the independent variable column (e.g., "x", price).

`id_var` Character string, the name of the subject/group identifier column for random effects.

`factors` A character vector of factor names (up to two) by which Q0 and alpha are expected to vary (e.g., `c("dose", "treatment")`).

`factor_interaction`
Logical. If `TRUE` and two factors are provided, their interaction term is included in the fixed effects for Q0 and alpha. Defaults to `FALSE` (additive effects).

`equation_form` Character string specifying the demand equation form. Options are:

- `"zben"` (default): Assumes `y_var` is log-transformed. Equation: `y_var ~ Q0 * exp(-(10^alpha / Q0) * (10^Q0) * x_var)`. Model parameter `Q0` represents `log10(True Max Consumption)`. Model parameter `alpha` represents `log10(True Alpha Sensitivity)`.
- `"simplified"`: Assumes `y_var` is raw (untransformed) consumption. Equation: `y_var ~ (10^Q0) * exp(-(10^alpha) * (10^Q0) * x_var)`. Model parameter `Q0` represents `log10(True Max Consumption)`. Model parameter `alpha` represents `log10(True Alpha Sensitivity)`.
- `"exponentiated"`: Koffarnus et al. (2015) exponentiated equation. Assumes `y_var` is raw (untransformed) consumption. Requires the `k` parameter. Equation (log10 param_space): `y_var ~ (10^Q0) * 10^(k * (exp(-(10^alpha) * (10^Q0) * x_var) - 1))`. Equation (natural param_space): `y_var ~ Q0 * 10^(k * (exp(-alpha * Q0 * x_var) - 1))`.

`param_space` Character. Parameterization used for fitting core demand parameters. One of:

- `"log10"`: treat `Q0` and `alpha` as `log10(Q0)` and `log10(alpha)` (default)
- `"natural"`: treat `Q0` and `alpha` as natural-scale parameters

Notes:

- For `equation_form = "zben"`, only `"log10"` is currently supported.
- For `equation_form = "simplified"` or `"exponentiated"`, both `"log10"` and `"natural"` are supported.

`k` Numeric. Range parameter (in log10 units) used with `equation_form = "exponentiated"`. If `NULL` (default), k is calculated from the data range: `log10(max(y)) - log10(min(y)) + 0.5`. Ignored for other equation forms.

`custom_model_formula`
An optional custom nonlinear model formula (nlme format). If provided, this overrides `equation_form`. The user is responsible for ensuring the `y_var` scale matches the formula and that starting values are appropriate. The formula should use parameters named `Q0` and `alpha`.

`fixed_rhs` Optional one-sided formula or character string specifying the right-hand side (RHS) for the fixed-effects linear models of `Q0` and `alpha`. When provided, this RHS is used for both parameters and overrides `factors`, `factor_interaction`, and `continuous_covariates` for building the fixed-effects design matrix. Example: `"~ 1 + drug * dose + session"`.

`continuous_covariates`
: Optional character vector of continuous (numeric) predictor names to be included additively in the fixed-effects RHS when `fixed_rhs` is `NULL`. These variables are not coerced to factors and are stored for downstream functions (e.g., plotting) to condition on.

`start_value_method`
: Character, method to generate starting values if `start_values` is NULL. Options: "heuristic" (default, uses data-driven heuristics) or "pooled_nls" (fits a simpler pooled NLS model first; falls back to heuristic if NLS fails).

`random_effects`
: A formula or a list of formulas for the random effects structure. Default `nlme::pdDiag(Q0 + alpha ~ 1)`.

`covariance_structure`
: Character, covariance structure for random effects. Options: `"pdDiag"` (default) or `"pdSymm"`

`start_values`
: Optional named list of starting values for fixed effects. If `NULL`, defaults are estimated based on `equation_form` and `y_var` scale.

`collapse_levels`
: Optional named list specifying factor level collapsing separately for Q0 and alpha parameters. Structure:

```
list(
  Q0 = list(factor_name = list(new_level = c(old_levels), ...)),
  alpha = list(factor_name = list(new_level = c(old_levels), ...))
)
```

Either `Q0` or `alpha` (or both) can be omitted to use original factor levels for that parameter. This allows different collapsing schemes for each parameter. Ignored if `fixed_rhs` is provided.

`nlme_control`
: Control parameters for `nlme::nlme()`.

`method`
: Fitting method for `nlme::nlme()` ("ML" or "REML"). Default "ML".

`...`
: Additional arguments passed to `nlme::nlme()`.

Value

An object of class `beezdemand_nlme`.

Examples

```
# Basic mixed-effects demand fit with apt data
# Transform consumption using LL4 for the zben equation
apt_ll4 <- apt |> dplyr::mutate(y_ll4 = ll4(y))

fit <- fit_demand_mixed(
  data = apt_ll4,
  y_var = "y_ll4",
  x_var = "x",
  id_var = "id",
  equation_form = "zben"
)
```

```
print(fit)
summary(fit)
```

`fixed-demand` *Fixed-Effect Demand Curve Fitting*

Description

Modern wrapper for fitting individual demand curves via nonlinear least squares. Returns a structured S3 object with standard methods.

`fixef.beezdemand_nlme` *Extract Fixed Effects from a beezdemand_nlme Model*

Description

S3 method for `fixef` for objects of class `beezdemand_nlme`. Extracts the fixed-effect coefficients from the fitted `nlme` model.

Usage

```
## S3 method for class 'beezdemand_nlme'
fixef(object, ...)
```

Arguments

`object`	A `beezdemand_nlme` object.
`...`	Additional arguments passed to `nlme::fixef()`.

Value

A named numeric vector of fixed-effect coefficients.

See Also

`coef.beezdemand_nlme`, `ranef.beezdemand_nlme`

fixef.cp_model_lmer *Extract Fixed Effects from Mixed-Effects Cross-Price Model*

Description

Extract Fixed Effects from Mixed-Effects Cross-Price Model

Usage

```
## S3 method for class 'cp_model_lmer'
fixef(object, ...)
```

Arguments

object	A cp_model_lmer object
...	Additional arguments passed to fixef

Value

Named vector of fixed effects

GetAnalyticPmax *Get pmax*

Description

...

Usage

```
GetAnalyticPmax(Alpha, K, Q0)
```

Arguments

Alpha	alpha parameter
K	k parameter (> lower limit)
Q0	Q0

Details

...

Value

Numeric

Author(s)

Shawn Gilroy sgilroy1@lsu.edu

Examples

```
GetAnalyticPmax(Alpha = 0.001, K = 3, Q0 = 10)
```

`GetAnalyticPmaxFallback` *Analytic Pmax Fallback*

Description

Fallback method for Analytic Pmax

Usage

```
GetAnalyticPmaxFallback(K_, A_, Q0_)
```

Arguments

`K_`	k parameter
`A_`	alpha parameter
`Q0_`	q0 parameter

Details

Derivative-based optimization strategy

Value

numeric

Author(s)

Shawn Gilroy sgilroy1@lsu.edu

Examples

```
GetAnalyticPmaxFallback(K_ = 1, A_ = 0.001, Q0_ = 10)
```

`GetDescriptives` *Get Purchase Task Descriptive Summary*

Description

Calculates descriptive statistics from purchase task data.

Usage

```
GetDescriptives(
  dat,
  bwplot = FALSE,
  outdir = "../plots/",
  device = "png",
  filename = "bwplot"
)
```

Arguments

`dat`	Dataframe (long form)
`bwplot`	Boolean. If TRUE, a ggplot2 box and whisker plot is saved. Default is FALSE.
`outdir`	Character. Directory where plot will be saved. Be sure to include trailing '/'. Default location is one level up in "../plots/".
`device`	Character. Type of file. Default is "png". Can be "pdf".
`filename`	Character. Specify filename. Defualt is "bwplot".

Details

[Superseded]

`GetDescriptives()` has been superseded by `get_descriptive_summary()`, which provides a modern S3 interface with standardized methods (`print()`, `summary()`, `plot()`). `GetDescriptives()` will continue to work but is no longer recommended for new code.

Provides the following descriptive statistics from purchase task data at each price: mean consumption, median consumption, standard deviation of consumption, proportion of 0 values, number of NAs, minimum consumption, and maximum consumption.

Value

Dataframe with descriptive statistics

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

See Also

`get_descriptive_summary()` for the modern interface

Examples

```
GetDescriptives(apt)
```

`GetEmpirical` *GetEmpirical*

Description

Calculates empirical measures for purchase task data

Usage

```
GetEmpirical(dat, xcol = "x", ycol = "y", idcol = "id")
```

Arguments

`dat`	data frame (long form) of purchase task data.
`xcol`	The column name that should be treated as "x" data
`ycol`	The column name that should be treated as "y" data
`idcol`	The column name that should be treated as dataset identifier

Details

[Superseded]

`GetEmpirical()` has been superseded by `get_empirical_measures()`, which provides a modern S3 interface with standardized methods (`print()`, `summary()`, `plot()`). `GetEmpirical()` will continue to work but is no longer recommended for new code.

Will calculate and return the following empirical measures: Intensity, BP0, BP1, Omax, and Pmax

Value

Data frame of empirical measures

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

See Also

`get_empirical_measures()` for the modern interface

Examples

```
## Obtain empirical measures
GetEmpirical(apt)
```

GetK *Get K*

Description

Calculates a k value by looking for the max/min consumption across entire dataset and adds .5 to that range

Usage

```
GetK(dat, mnrange = TRUE)
```

Arguments

dat	Dataframe (long form)
mnrange	Boolean for whether k should be calculated based on the mean range + .5

Details

[Superseded]

`GetK()` has been superseded by `get_k()`, which provides explicit parameters for the adjustment value and optional verbose output for better transparency. `GetK()` will continue to work but is no longer recommended for new code.

Will look for maximum/minimum greater zero

Value

Numeric

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

See Also

`get_k()` for the modern interface

Examples

```
GetK(apt)
```

GetSharedK *Get Shared K*

Description

Finds shared k among selected datasets using global regression

Usage

```
GetSharedK(dat, equation, sharecol = "group")
```

Arguments

dat	Dataframe (longform)
equation	Character vector. Accepts either "hs" or "koff"
sharecol	Character for column to find shared k. Default to "group" but can loop based on id.

Details

Uses global regression to fit a shared k among datasets. Assumes the dataset is in its final form. Used within FitCurves

Value

Numeric value of shared k

Author(s)

Brent Kaplan bkaplan.ku@gmail.com Shawn P Gilroy shawn.gilroy@temple.edu

Examples

```
## Find a shared k value across datasets indicated by id

GetSharedK(apt, "hs", sharecol = "id")
```

`GetValsForSim` *Get Values for SimulateDemand*

Description

Gets values used in SimulateDemand

Usage

```
GetValsForSim(dat)
```

Arguments

`dat`	Dataframe (long form)

Details

Gets values used in SimulateDemand

Value

List of 3: setaparams, sdindex, x

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
GetValsForSim(apt)
```

`get_demand_comparisons`
Get Pairwise Comparisons for Demand Parameters

Description

Conducts pairwise comparisons for Q0 and/or alpha parameters from a `beezdemand_nlme` model across levels of specified factors. Comparisons are performed on the log10 scale of the parameters. Results include estimates of differences (on log10 scale) and optionally, ratios (on the natural scale by applying 10^difference).

Usage

```
get_demand_comparisons(
  fit_obj,
  params_to_compare = c("Q0", "alpha"),
  compare_specs = NULL,
  contrast_type = "pairwise",
  contrast_by = NULL,
  adjust = "tukey",
  at = NULL,
  ci_level = 0.95,
  report_ratios = TRUE,
  ...
)
```

Arguments

fit_obj	A beezdemand_nlme object.
params_to_compare	Character vector: "Q0", "alpha", or c("Q0", "alpha"). Default c("Q0", "alpha").
compare_specs	A formula specifying the factors whose levels are to be included in the EMM calculation prior to contrasting. This defines the "cells" of your design for EMMs. E.g., ~ factor1 (EMMs for levels of factor1, averaging over others), ~ factor1 * factor2 (EMMs for all cells of factor1 x factor2). If NULL, it defaults to an interaction of all factors in fit_obj$param_info$factors.
contrast_type	Character string specifying the type of contrast (passed to method in emmeans::contrast). Commonly "pairwise", "revpairwise", "eff", "consec", "poly". Default "pairwise".
contrast_by	Optional character vector of factor names to condition the contrasts by (passed to by in emmeans::contrast). If NULL (default), contrasts are performed over the primary terms implied by compare_specs and contrast_type. Example: If compare_specs = ~ dose * drug, contrast_type = "pairwise", and contrast_by = "dose", this will perform pairwise comparisons of drug levels within each level of dose. **Note:** If the original fit_obj model is additive for the factors involved (i.e., no interaction term was fitted), specifying contrast_by will result in identical contrast estimates across the levels of the contrast_by variable(s). In such cases, consider analyzing main effects directly (e.g., compare_specs = ~drug, contrast_by = NULL).
adjust	P-value adjustment method. Default "tukey".
at	Optional named list for emmeans::ref_grid().
ci_level	Confidence level. Default 0.95.
report_ratios	Logical. If TRUE, reports contrasts as ratios. Default TRUE.
...	Additional arguments passed to emmeans::emmeans() or emmeans::contrast().

Value

A list named by parameter. Each element contains:

emmeans Tibble of EMMs (log10 scale) with CIs.

contrasts_log10
Tibble of comparisons (log10 differences) with CIs and p-values.

contrasts_ratio
(If report_ratios=TRUE and successful) Tibble of comparisons as ratios (natural scale), with CIs for ratios.

S3 class beezdemand_comparison is assigned.

Examples

```
data(ko, package = "beezdemand")
ko$y_ll4 <- ll4(ko$y, lambda = 4)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
  id_var = "monkey", factors = "dose", equation_form = "zben")
get_demand_comparisons(fit)
```

---

get_demand_param_emms *Get Estimated Marginal Means for Demand Parameters*

Description

Calculates Estimated Marginal Means (EMMs) for Q0 and alpha parameters from a beezdemand_nlme model for all combinations of specified factor levels. Reports parameters on both their estimation scale (log10) and their natural, back-transformed scale. Optionally includes Essential Value (EV).

Usage

```
get_demand_param_emms(
  fit_obj,
  factors_in_emm = NULL,
  at = NULL,
  ci_level = 0.95,
  include_ev = FALSE,
  ...
)
```

Arguments

fit_obj A beezdemand_nlme object.

factors_in_emm Character vector of factor names to compute EMMs over. Defaults to all factors present in the fit_obj.

at Optional named list specifying levels of conditioning variables for emmeans::ref_grid().

ci_level Confidence level for the EMMs (default 0.95).

include_ev Logical. If TRUE, calculates and includes Essential Value (EV) derived from alpha, along with its confidence interval (calculated by back-transforming the CI of alpha_param_log10). Default FALSE.

... Additional arguments passed to emmeans::emmeans().

Value

A tibble containing:

`Factor levels` Columns for each factor in `factors_in_emm`.

`Q0_param_log10`, `alpha_param_log10`
EMMs for the model parameters (log10 scale) with their respective confidence intervals (LCL_Q0_param, UCL_Q0_param, etc.).

`Q0_natural`, `alpha_natural`
EMMs back-transformed to the natural scale (10^param) with their respective confidence intervals (LCL_Q0_natural, UCL_Q0_natural, etc.).

`EV`, `LCL_EV`, `UCL_EV`
(If `include_ev=TRUE`) Essential Value and its CI.

Examples

```
data(ko, package = "beezdemand")
ko$y_ll4 <- ll4(ko$y, lambda = 4)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
  id_var = "monkey", factors = "dose", equation_form = "zben")
get_demand_param_emms(fit)
```

---

get_demand_param_trends

Get Trends (Slopes) of Demand Parameters with respect to Continuous Covariates

---

Description

Computes the trend (slope) of `Q0` and/or `alpha` with respect to one or more continuous covariates using `emmeans::emtrends()` on a fitted `beezdemand_nlme` model. Trends are computed on the parameter estimation scale (log10), consistent with how parameters are modeled.

Usage

```
get_demand_param_trends(
  fit_obj,
  params = c("Q0", "alpha"),
  covariates,
  specs = ~1,
  at = NULL,
  ci_level = 0.95,
  ...
)
```

Arguments

fit_obj	A beezdemand_nlme object from fit_demand_mixed().
params	Character vector of parameters to analyze: any of "Q0", "alpha". Default c("Q0", "alpha").
covariates	Character vector of continuous covariate names for which to compute trends.
specs	A formula specifying the factors over which to produce trends (e.g., ~ drug for trends by drug; ~ 1 for overall). Default ~ 1.
at	Optional named list to condition variables (factors or continuous) when computing trends (passed through to emmeans::ref_grid).
ci_level	Confidence level for intervals. Default 0.95.
...	Additional args passed to emmeans::emtrends().

Value

A tibble combining trends for each requested parameter and covariate, including columns for grouping factors (from specs), parameter, covariate, trend (slope on log10 scale), and its CI (lower.CL, upper.CL).

Examples

```
data(ko)
ko$dose_num <- as.numeric(as.character(ko$dose))
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", factors = "drug",
                        equation_form = "zben")
trends <- get_demand_param_trends(fit, covariates = "dose_num",
                                  specs = ~ drug)
```

---

get_descriptive_summary

Calculate Descriptive Statistics by Price

Description

Calculates summary statistics for consumption data at each price point, including measures of central tendency (mean, median), variability (SD), range (min, max), and data quality (proportion of zeros, missing values).

This is the modern replacement for GetDescriptives(), returning a structured S3 object with dedicated methods for printing, summarizing, and visualizing.

Usage

```
get_descriptive_summary(data, x_var = "x", y_var = "y", id_var = "id")
```

Arguments

`data`	A data frame in long format with columns for subject ID, price, and consumption
`x_var`	Character string specifying the column name for price (default: "x")
`y_var`	Character string specifying the column name for consumption (default: "y")
`id_var`	Character string specifying the column name for subject ID (default: "id")

Details

For each unique price in the dataset, the function calculates:

- **Mean** - Average consumption across subjects (rounded to 2 decimals)
- **Median** - Median consumption (rounded to 2 decimals)
- **SD** - Standard deviation (rounded to 2 decimals)
- **PropZeros** - Proportion of subjects with zero consumption (0-1)
- **NAs** - Count of missing values
- **Min** - Minimum consumption value (rounded to 2 decimals)
- **Max** - Maximum consumption value (rounded to 2 decimals)

Value

An S3 object of class `beezdemand_descriptive` containing:

- **statistics** - Data frame with 8 columns (Price, Mean, Median, SD, PropZeros, NAs, Min, Max) and one row per unique price
- **call** - The matched call
- **data_summary** - List with n_subjects, n_prices, and prices vector

See Also

- `GetDescriptives()` - Legacy function (superseded)
- `plot.beezdemand_descriptive()` - Visualization method
- `summary.beezdemand_descriptive()` - Extended summary

Examples

```
data(apt, package = "beezdemand")

# Calculate descriptive statistics
desc <- get_descriptive_summary(apt)
print(desc)

# View statistics table
desc$statistics

# Create visualization
plot(desc)
```

```
# Extended summary with distribution info
summary(desc)
```

`get_empirical_measures` *Calculate Empirical Demand Measures*

Description

Calculates empirical (model-free) measures of demand from purchase task data. These metrics characterize consumption patterns without fitting a demand curve model.

This is the modern replacement for `GetEmpirical()`, returning a structured S3 object with dedicated methods for printing, summarizing, and visualizing.

Usage

```
get_empirical_measures(data, x_var = "x", y_var = "y", id_var = "id")
```

Arguments

`data`	A data frame in long format with columns for subject ID, price, and consumption
`x_var`	Character string specifying the column name for price (default: "x")
`y_var`	Character string specifying the column name for consumption (default: "y")
`id_var`	Character string specifying the column name for subject ID (default: "id")

Details

Empirical Measures:

- **Intensity** - The consumption value at the lowest price point. Reflects unrestricted demand or preferred level of consumption.
- **BP0** (Breakpoint 0) - The first price at which consumption drops to zero. If consumption never reaches zero, BP0 = NA. Indicates the price threshold where the commodity becomes unaffordable or undesirable.
- **BP1** (Breakpoint 1) - The highest price at which consumption is still non-zero. If all consumption is zero, BP1 = NA. Represents the upper limit of the commodity's value to the consumer.
- **Omaxe** (Empirical Omax) - The maximum observed expenditure across all prices (max of price × consumption). Represents peak spending on the commodity.
- **Pmaxe** (Empirical Pmax) - The price at which maximum expenditure occurs. If maximum expenditure is zero, Pmaxe = 0. If multiple prices have the same maximum expenditure, the highest price is returned.

Value

An S3 object of class `beezdemand_empirical` containing:

- **measures** - Data frame with one row per subject and columns:
  - *id* - Subject identifier
  - *Intensity* - Consumption at lowest price (demand intensity)
  - *BP0* - Breakpoint 0: first price where consumption = 0
  - *BP1* - Breakpoint 1: last price with non-zero consumption
  - *Omaxe* - Empirical Omax: maximum total expenditure (price × consumption)
  - *Pmaxe* - Empirical Pmax: price at which maximum expenditure occurs
- **call** - The matched call
- **data_summary** - List with n_subjects, has_zeros, and complete_cases

Note

- Data must not contain duplicate prices within a subject (will error)
- Missing values (NA) in consumption are preserved in calculations where applicable
- Breakpoints require at least one zero consumption value to be meaningful

See Also

- `GetEmpirical()` - Legacy function (superseded)
- `plot.beezdemand_empirical()` - Visualization method
- `summary.beezdemand_empirical()` - Extended summary

Examples

```
data(apt, package = "beezdemand")

# Calculate empirical measures
emp <- get_empirical_measures(apt)
print(emp)

# View measures table
emp$measures

# Extended summary with distribution info
summary(emp)

# Visualize distribution of measures
plot(emp)  # histogram by default
plot(emp, type = "matrix")  # scatterplot matrix
```

`get_hurdle_param_summary` *Get Hurdle Model Parameter Summary*

Description

Provides summary statistics for subject-level demand parameters from a hurdle demand model. This is analogous to EMMs but based on empirical Bayes estimates of subject-specific parameters.

Usage

```
get_hurdle_param_summary(fit_obj, ci_level = 0.95)
```

Arguments

`fit_obj`	A `beezdemand_hurdle` object.
`ci_level`	Confidence level for intervals (default 0.95).

Value

A data frame with summary statistics for each parameter:

parameter Parameter name

mean Mean across subjects

sd Standard deviation across subjects

median Median across subjects

lcl Lower confidence limit (based on percentiles)

ucl Upper confidence limit (based on percentiles)

min Minimum value

max Maximum value

See Also

`fit_demand_hurdle`, `get_subject_pars`

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
get_hurdle_param_summary(fit)
```

get_individual_coefficients *Calculate Individual-Level Predicted Coefficients from beezdemand_nlme Model*

Description

This function extracts and combines fixed and random effects to calculate individual-level predicted coefficients for all parameter-factor combinations from a beezdemand_nlme model object. It automatically detects the factor structure and calculates coefficients for each individual and factor level.

Usage

```
get_individual_coefficients(
  fit_obj,
  params = c("Q0", "alpha"),
  format = c("wide", "long")
)
```

Arguments

`fit_obj`	A `beezdemand_nlme` object returned by `fit_demand_mixed()`.
`params`	Character vector specifying which parameters to calculate. Options are "Q0", "alpha", or c("Q0", "alpha"). Default is c("Q0", "alpha").
`format`	Character, output format. "wide" returns one row per individual with separate columns for each parameter-factor combination. "long" returns one row per individual-parameter-factor combination. Default is "wide".

Details

Individual-level coefficients represent the predicted parameter values for each subject in the study. For models with factors, these coefficients combine:

1. The baseline intercept effect (fixed + random)
2. The factor-specific effect (fixed + random) for each factor level

This is equivalent to manually calculating: `coefficient = intercept_fixed + intercept_random + factor_fixed + factor_random`

The function automatically handles:

- Models with or without factors
- Any number of factor levels
- Missing random effects (defaults to 0)
- Complex factor structures with multiple factors

For models without factors, only intercept coefficients are calculated. For models with factors, both intercept and factor-level coefficients are provided.

Value

A data frame with individual-level predicted coefficients.

- In "wide" format: rows are individuals, columns are parameter-factor combinations
- In "long" format: columns are id, parameter, condition, coefficient_value

Column naming convention for wide format:

- `estimated_\{param\}_intercept`: Baseline/reference level coefficient
- `estimated_\{param\}_\{factor\}\{level\}`: Factor level-specific coefficient

All coefficients are on the log10 scale (same as model estimation scale). To convert to natural scale, use `10^coefficient`.

See Also

`fit_demand_mixed` for fitting the original model `coef.beezdemand_nlme` for extracting model coefficients `get_demand_param_emms` for estimated marginal means

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", factors = "drug",
                        equation_form = "zben")
individual_coefs <- get_individual_coefficients(fit)
head(individual_coefs)
```

---

get_k *Calculate K Scaling Parameter for Demand Curve Fitting*

Description

Calculates the k scaling parameter used in demand curve equations to normalize consumption across different units or ranges. The k value is derived from the logarithmic range of consumption values.

This is the modern replacement for `GetK()`, with explicit parameters for the adjustment value and optional verbose output.

Usage

```
get_k(
  data,
  use_means = TRUE,
  adjustment = 0.5,
  x_var = "x",
  y_var = "y",
  verbose = FALSE
)
```

Arguments

`data`	A data frame in long format with columns for price and consumption
`use_means`	Logical indicating whether to calculate k from mean consumption by price (TRUE, default) or from individual consumption values (FALSE)
`adjustment`	Numeric adjustment added to the log range (default: 0.5). This value ensures k is slightly larger than the observed range.
`x_var`	Character string specifying the column name for price (default: "x")
`y_var`	Character string specifying the column name for consumption (default: "y")
`verbose`	Logical indicating whether to print calculation details (default: FALSE)

Details

The k parameter is calculated as:

$$k = \log_{10}(\text{max}) - \log_{10}(\text{min}) + \text{adjustment}$$

where max and min are the maximum and minimum non-zero consumption values.

Use in Demand Equations:

The k parameter appears in several demand curve equations:

- **Hursh & Silberberg (2008)**: Scales the exponential term
- **Koffarnus et al. (2015)**: Normalizes the exponentiated model
- Ensures numerical stability during model fitting

Calculation Modes:

- **use_means = TRUE** (default): Calculates k from mean consumption at each price point. Recommended when data has multiple subjects, as it reduces influence of individual outliers.
- **use_means = FALSE**: Calculates k from the full range of individual consumption values. May be preferable for single-subject data or when individual variability is theoretically important.

Value

A single numeric value representing the k scaling parameter

Note

- Only non-zero consumption values are used (zero values are excluded)
- Missing values (NA) are automatically removed via `na.rm = TRUE`
- The default adjustment of 0.5 is conventional but can be modified

See Also

- `GetK()` - Legacy function (superseded)
- `FitCurves()` - Uses k parameter in demand curve fitting

Examples

```
data(apt, package = "beezdemand")

# Calculate k using default settings (mean range + 0.5)
k_val <- get_k(apt)

# Calculate k from individual values
k_ind <- get_k(apt, use_means = FALSE)

# Calculate with custom adjustment
k_custom <- get_k(apt, adjustment = 1.0)

# Show calculation details
k_verbose <- get_k(apt, verbose = TRUE)
```

---

get_observed_demand_param_emms

Get Estimated Marginal Means for Observed Factor Combinations

---

Description

This function is a wrapper around `get_demand_param_emms`. It first calls `get_demand_param_emms` to calculate Estimated Marginal Means (EMMs) for Q0 and alpha parameters over all combinations of the specified factor levels. It then filters these results to return EMMs only for the combinations of factor levels that were actually present in the original dataset used to fit the `beezdemand_nlme` model.

Usage

```
get_observed_demand_param_emms(
  fit_obj,
  factors_in_emm = NULL,
  at = NULL,
  ci_level = 0.95,
  include_ev = FALSE,
  ...
)
```

Arguments

`fit_obj`	A `beezdemand_nlme` object returned by `fit_demand_mixed()`.
`factors_in_emm`	Character vector of factor names to compute EMMs over. Defaults to all factors present in the `fit_obj`. These factors define the grid over which EMMs are initially calculated and then filtered.
`at`	Optional named list specifying levels of conditioning variables for `emmeans::ref_grid()`. Passed to `get_demand_param_emms`.

ci_level	Confidence level for the EMMs (default 0.95). Passed to get_demand_param_emms.
include_ev	Logical. If TRUE, calculates and includes Essential Value (EV) derived from alpha. Passed to get_demand_param_emms. Default FALSE.
...	Additional arguments passed to get_demand_param_emms and subsequently to emmeans::emmeans().

Value

A tibble similar to the output of get_demand_param_emms, but filtered to include only rows corresponding to factor level combinations that were observed in the original fit_obj$data. Contains:

Factor levels	Columns for each factor in factors_in_emm.
Q0_param_log10, alpha_param_log10	EMMs for model parameters (log10 scale) and CIs.
Q0_natural, alpha_natural	EMMs back-transformed to natural scale and CIs.
EV, LCL_EV, UCL_EV	(If include_ev=TRUE) Essential Value and its CI.

See Also

get_demand_param_emms

Examples

```
data(ko, package = "beezdemand")
ko$y_ll4 <- ll4(ko$y, lambda = 4)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
  id_var = "monkey", factors = "dose", equation_form = "zben")
get_observed_demand_param_emms(fit)
```

---

get_subject_pars *Get Subject-Specific Parameters*

Description

Convenience function to extract subject-specific demand parameters from a fitted hurdle demand model. Equivalent to accessing object$subject_pars.

Usage

```
get_subject_pars(object)
```

Arguments

object	A fitted beezdemand_hurdle object.

Value

Data frame with subject-specific parameters including:

id Subject identifier

a_i Random effect for Part I (zeros)

b_i Random effect for Part II (Q0)

c_i Random effect for alpha (3-RE model only)

Q0 Subject-specific intensity (consumption at price 0)

alpha Subject-specific elasticity

breakpoint Price where P(quit) = 0.5

Pmax Price at maximum expenditure

Omax Maximum expenditure

See Also

fit_demand_hurdle

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
pars <- get_subject_pars(fit)
head(pars)
```

---

glance.beezdemand_fixed

Glance Method for beezdemand_fixed

Description

Glance Method for beezdemand_fixed

Usage

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

Arguments

x	A beezdemand_fixed object
...	Additional arguments (ignored)

Value

A one-row tibble of model statistics

glance.beezdemand_hurdle

Glance at a beezdemand_hurdle Model

Description

Returns a one-row tibble of model-level statistics.

Usage

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

Arguments

x	An object of class beezdemand_hurdle.
...	Additional arguments (currently unused).

Value

A one-row tibble with columns:

model_class "beezdemand_hurdle"

backend "TMB"

nobs Number of observations

n_subjects Number of subjects

n_random_effects Number of random effects

converged Convergence status

logLik Log-likelihood

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

glance.beezdemand_nlme

Glance method for beezdemand_nlme

Description

Glance method for beezdemand_nlme

Usage

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

Arguments

x	A beezdemand_nlme object
...	Additional arguments (ignored)

Value

A one-row tibble of model statistics with columns:

- model_class: "beezdemand_nlme"
- backend: "nlme"
- equation_form: The equation form used
- nobs: Number of observations
- n_subjects: Number of subjects
- converged: Convergence status
- logLik, AIC, BIC: Model fit statistics
- sigma: Residual standard error

---

glance.beezdemand_systematicity
Glance Method for beezdemand_systematicity

Description

Glance Method for beezdemand_systematicity

Usage

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

Arguments

x	A beezdemand_systematicity object
...	Additional arguments (ignored)

Value

A one-row tibble of overall statistics

glance.cp_model_lm *Get model summaries from a linear cross-price model*

Description

Get model summaries from a linear cross-price model

Usage

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

Arguments

x	A cp_model_lm object.
...	Additional arguments (unused).

Value

A tibble with model summary statistics.

glance.cp_model_lmer *Get model summaries from a mixed-effects cross-price model*

Description

Get model summaries from a mixed-effects cross-price model

Usage

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

Arguments

x	A cp_model_lmer object.
...	Additional arguments passed to broom.mixed::glance.

Value

A tibble with model summary statistics.

`glance.cp_model_nls` *Get model summaries from a cross-price model*

Description

This function extracts model summary statistics from a cross-price demand model into a single-row data frame, following the conventions of the broom package. It returns goodness-of-fit measures and other model information.

Usage

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

Arguments

`x`	A model object from fit_cp_nls or fit_cp_linear
`...`	Additional arguments (unused)

Value

A one-row data frame with model summary statistics:

`r.squared`	R-squared value indicating model fit
`aic`	Akaike Information Criterion
`bic`	Bayesian Information Criterion
`equation`	The equation type used in the model
`method`	The method used to fit the model
`transform`	The transformation applied to the data, if any

Examples

```
data(etm)
fit <- fit_cp_nls(etm, equation = "exponentiated")
glance(fit)
```

ko — *Example nonhuman demand data with drug and dose*

Description

Data from Ko et al. (2002)

Usage

```
ko
```

Format

A tibble with 135 rows and 6 columns:

monkey unique identifier

x fixed-ratio requirement

y consumption

y_ll4 consumption transformed via ll4

drug drug

dose dose

lambertW — *Lambert W*

Description

Ben Bolker's port of Lambert W from GNU Scientific Library (GPLV3)

Usage

```
lambertW(z, b = 0, maxiter = 10, eps = .Machine$double.eps, min.imag = 1e-09)
```

Arguments

z	input value
b	branch, set to principal by default
maxiter	Halley iteration count
eps	error precision
min.imag	minimum for imaginary solution

Details

Ben Bolker's port of Lambert W from GNU Scientific Library

Value

numeric

Author(s)

Benjamin Bolker (port)

Examples

```
## Principal branch: W(1) ~ 0.5671
lambertW(1)

## Verify: W(z) * exp(W(z)) == z
w <- lambertW(2)
w * exp(w)
```

---

ll4 *Log-Logistic Transformation (LL4-like)*

Description

Applies a log-logistic like transformation, specifically `log_base(x^lambda + 1) / lambda`. This transformation is useful for compressing data that spans several orders of magnitude while handling zero values gracefully (as `x=0` yields `0`). It's a variation related to the Box-Cox transformation or a generalized logarithm.

Usage

```
ll4(x, lambda = 4, base = 10)
```

Arguments

`x`	A numeric vector or scalar of non-negative values to be transformed.
`lambda`	A positive numeric scalar, the lambda parameter of the transformation. Controls the curvature. Default is `4`.
`base`	A positive numeric scalar, the base of the logarithm. Default is `10`.

Value

A numeric vector or scalar of the transformed values. Returns `NaN` for `x < 0` if `lambda` results in non-real numbers (e.g., even root of negative). However, the intended domain is `x >= 0`.

Examples

```
ll4(0)
ll4(1)
ll4(10)
ll4(100)
ll4(c(0, 1, 10, 100, 1000))

# Using a different lambda or base
ll4(10, lambda = 2)
ll4(10, base = exp(1)) # Natural log base
```

---

ll4_inv *Inverse Log-Logistic Transformation (Inverse LL4-like)*

Description

Applies the inverse of the `ll4` transformation. Given `y = ll4(x)`, this function calculates `x = (base^(y * lambda) - 1)^(1/lambda)`.

Usage

```
ll4_inv(y, lambda = 4, base = 10)
```

Arguments

`y`	A numeric vector or scalar of transformed values (output from `ll4`).
`lambda`	A positive numeric scalar, the lambda parameter used in the original `ll4` transformation. Must match the one used for the forward transform. Default is `4`.
`base`	A positive numeric scalar, the base of the logarithm used in the original `ll4` transformation. Must match. Default is `10`.

Value

A numeric vector or scalar of the original, untransformed values. May return `NaN` if `(base^(y * lambda) - 1)` is negative and `1/lambda` implies an even root (e.g., if `lambda` is 2 or 4).

Examples

```
original_values <- c(0, 1, 10, 100, 1000)
transformed_values <- ll4(original_values)
back_transformed_values <- ll4_inv(transformed_values)
print(data.frame(original_values, transformed_values, back_transformed_values))
all.equal(original_values, back_transformed_values) # Should be TRUE or very close

# Example with negative y (log-transformed value)
# If y_ll4 = -0.5 (meaning original value was between 0 and 1 for log10)
ll4_inv(-0.5, lambda = 4, base = 10) # (10^(-0.5*4) - 1)^(1/4) = (0.01 - 1)^(1/4) -> NaN
# The ll4_inv function as provided will return NaN here.
# A more robust version for demand might floor at 0 if NaN occurs.
```

logLik.beezdemand_hurdle *Extract Log-Likelihood from Hurdle Demand Model*

Description

Extracts the log-likelihood from a fitted hurdle demand model. Useful for likelihood ratio tests comparing nested models.

Usage

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

Arguments

object	An object of class beezdemand_hurdle.
...	Additional arguments (currently unused).

Value

An object of class logLik with the log-likelihood value and attributes for degrees of freedom and number of observations.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
logLik(fit)
```

lowNicClean *Low-nicotine cigarette purchase task*

Description

Long-form purchase task data across nicotine conditions.

Usage

```
lowNicClean
```

Format

A data frame with 5 columns: id, condition, x, y, commodity.

Examples

```
# data(lowNicClean)
```

`ongoingETM` *Experimental Tobacco Marketplace (ETM) data*

Description

ETM data across price points with product quantities.

Usage

```
ongoingETM
```

Format

A data frame with 6 columns: id, x, AdjCig, FixCig, ECig, flavor.

Examples

```
# data(ongoingETM)
```

`palette_beezdemand` *beezdemand Color Palette*

Description

beezdemand Color Palette

Usage

```
palette_beezdemand()
```

Value

Named vector of brand colors.

Examples

```
palette_beezdemand()
```

pivot_demand_data *Reshape Demand Data Between Wide and Long Formats*

Description

Converts demand data between wide format (one row per subject, prices as columns) and long format (one row per observation with id, x, y columns). This is a convenience wrapper around tidyr::pivot_longer() and tidyr::pivot_wider() tailored for behavioral economic purchase task data.

Usage

```
pivot_demand_data(
  data,
  format = c("long", "wide"),
  id_var = "id",
  x_var = "x",
  y_var = "y",
  price_cols = NULL,
  x_values = NULL,
  drop_na = TRUE
)
```

Arguments

data	A data frame or tibble to reshape.
format	Character. Direction of reshaping: "long" (wide to long) or "wide" (long to wide). Default "long".
id_var	Character. Name of the subject/series identifier column. Default "id".
x_var	Character. Name of the price column in long-format data. Used when format = "wide". Default "x".
y_var	Character. Name of the consumption column in long-format data. Used when format = "wide". Default "y".
price_cols	Character vector of column names in wide-format data that represent prices. Used when format = "long". Default NULL, which auto-detects: columns whose names parse as numeric become price columns, and remaining columns (besides id_var) are preserved as identifiers. If no column names parse as numeric, all non-id_var columns are treated as price columns (and x_values must be supplied).
x_values	Numeric vector of actual price values corresponding to each price column in wide-format data. Used when format = "long". Must be the same length as the number of price columns. Default NULL, which parses prices from column names.
drop_na	Logical. When format = "long", drop rows where consumption (y) is NA after pivoting? Default TRUE. A warning is issued when rows are dropped.

Details

Wide to Long (`format = "long"`):

The function determines which columns are price columns by:

1. If `price_cols` is provided, those columns are used directly.
2. If `price_cols` is `NULL`, column names are tested with `as.numeric()`. Columns whose names successfully parse as numbers (e.g., `"0"`, `"0.5"`, `"10"`) are treated as price columns. Remaining non-`id_var` columns are preserved as extra identifiers.
3. If no column names parse as numeric, all non-`id_var` columns become price columns and `x_values` must be supplied.

Actual numeric price values come from `x_values` if supplied, or from parsing column names. If column names cannot be parsed and `x_values` is not supplied, an error is raised.

Long to Wide (`format = "wide"`):

Pivots long data so that each unique value in `x_var` becomes a column, with values from `y_var`. All columns except `x_var` and `y_var` are used as identifiers.

Value

A tibble. When `format = "long"`: columns `id`, any extra identifier columns, `x` (numeric price), and `y` (numeric consumption). When `format = "wide"`: one row per subject with prices as column names.

Examples

```
# --- Wide to long ---
# Columns named as prices (auto-parsed)
wide_num <- data.frame(
  id = 1:3,
  "0" = c(10, 8, 12), "0.5" = c(9, 7, 11), "1" = c(8, 6, 9),
  check.names = FALSE
)
pivot_demand_data(wide_num, format = "long")

# Columns with non-numeric names require x_values
wide_named <- data.frame(id = 1:2, price_1 = c(10, 8), price_2 = c(5, 4))
pivot_demand_data(wide_named, format = "long",
                  x_values = c(0, 0.5))

# --- Long to wide ---
data(apt, package = "beezdemand")
wide_apt <- pivot_demand_data(apt, format = "wide")
head(wide_apt)
```

plot-theme *beezdemand Plot Theme and Color Palette*

Description

Central plotting style API for beezdemand, with a modern default based on the codedbx "Refined Contemporary" brand and an APA alternative.

plot.beezdemand_fixed *Plot Method for beezdemand_fixed*

Description

Plot Method for beezdemand_fixed

Usage

```
## S3 method for class 'beezdemand_fixed'
plot(
  x,
  type = c("demand", "population", "individual", "both"),
  ids = NULL,
  style = c("modern", "apa"),
  show_observed = TRUE,
  show_pred = NULL,
  x_trans = c("log10", "log", "linear", "pseudo_log"),
  y_trans = NULL,
  free_trans = 0.01,
  x_limits = NULL,
  y_limits = NULL,
  n_points = 200,
  x_lab = NULL,
  y_lab = NULL,
  xlab = NULL,
  ylab = NULL,
  facet = NULL,
  observed_point_alpha = 0.5,
  observed_point_size = 1.8,
  pop_line_alpha = 0.9,
  pop_line_size = 1,
  ind_line_alpha = 0.35,
  ind_line_size = 0.7,
  subtitle = NULL,
  ...
)
```

Arguments

x	A beezdemand_fixed object.
type	Plot type: "demand", "population", "individual", or "both".
ids	Optional vector of subject IDs to plot. Defaults to all subjects.
style	Plot styling, passed to theme_beezdemand().
show_observed	Logical; if TRUE, overlay observed data points where possible.
show_pred	Which prediction layers to plot: "population", "individual", or "both".
x_trans	X-axis transform: "log", "log10", "linear", or "pseudo_log".
y_trans	Y-axis transform: "log", "log10", "linear", or "pseudo_log".
free_trans	Value used to display free (x = 0) on log scales. Use NULL to drop x <= 0 values instead.
x_limits	Optional numeric vector of length 2 for x-axis limits.
y_limits	Optional numeric vector of length 2 for y-axis limits.
n_points	Number of points to use for prediction curves when thinning.
x_lab	Optional x-axis label.
y_lab	Optional y-axis label.
xlab	Deprecated alias for x_lab.
ylab	Deprecated alias for y_lab.
facet	Faceting specification (TRUE for ~id or a formula).
observed_point_alpha	Alpha for observed points.
observed_point_size	Size for observed points.
pop_line_alpha	Alpha for population curve.
pop_line_size	Line size for population curve.
ind_line_alpha	Alpha for individual curves.
ind_line_size	Line size for individual curves.
subtitle	Optional subtitle for the plot.
...	Additional arguments (currently unused).

Value

A ggplot2 object.

plot.beezdemand_hurdle

Plot Demand Curves from Hurdle Demand Model

Description

Creates visualizations of fitted demand curves from a hurdle demand model.

Usage

```
## S3 method for class 'beezdemand_hurdle'
plot(
  x,
  type = c("demand", "population", "probability", "parameters", "individual", "both"),
  ids = NULL,
  subjects = NULL,
  parameters = c("Q0", "alpha", "breakpoint", "Pmax", "Omax"),
  prices = NULL,
  show_population = TRUE,
  show_pred = NULL,
  show_observed = TRUE,
  x_trans = c("log10", "log", "linear", "pseudo_log"),
  y_trans = NULL,
  free_trans = 0.01,
  facet = NULL,
  x_limits = NULL,
  y_limits = NULL,
  x_lab = NULL,
  y_lab = NULL,
  xlab = NULL,
  ylab = NULL,
  style = c("modern", "apa"),
  observed_point_alpha = 0.5,
  observed_point_size = 1.8,
  pop_line_alpha = 0.9,
  pop_line_size = 1,
  ind_line_alpha = 0.35,
  ind_line_size = 0.7,
  ...
)
```

Arguments

x	An object of class beezdemand_hurdle.
type	Character string specifying the plot type:
	"demand" Predicted demand curves (default)

	`"population"` Alias for `"demand"` `"probability"` Probability of zero consumption `"parameters"` Distribution of subject-specific parameters `"individual"` Individual demand curves for selected subjects
`ids`	Optional vector of subject IDs to plot (alias of `subjects`).
`subjects`	Character or numeric vector of subject IDs to plot for `type = "individual"`. If `NULL`, plots first 9 subjects.
`parameters`	Character vector specifying which parameters to plot when `type = "parameters"`. Options are: `"Q0"`, `"alpha"`, `"breakpoint"`, `"Pmax"`, `"Omax"`. Default is all five.
`prices`	Numeric vector of prices for plotting. If `NULL`, uses a sequence from 0 to max observed price.
`show_population`	Logical; if `TRUE`, overlay population-level curve on individual plots. Default is `TRUE`.
`show_pred`	Which prediction layers to plot: "population", "individual", or "both".
`show_observed`	Logical; if `TRUE`, overlay observed data points.
`x_trans`	Character. Transformation for x-axis. Default "log".
`y_trans`	Character. Transformation for y-axis. Default "log".
`free_trans`	Value used to display free (x = 0) on log scales. Use NULL to drop x <= 0 values instead.
`facet`	Faceting specification (TRUE for `~id` or a formula).
`x_limits`	Optional numeric vector of length 2 for x-axis limits.
`y_limits`	Optional numeric vector of length 2 for y-axis limits.
`x_lab`	Optional x-axis label.
`y_lab`	Optional y-axis label.
`xlab`	Deprecated alias for `x_lab`.
`ylab`	Deprecated alias for `y_lab`.
`style`	Plot styling, passed to `theme_beezdemand()`.
`observed_point_alpha`	Alpha for observed points.
`observed_point_size`	Size for observed points.
`pop_line_alpha`	Alpha for population curve.
`pop_line_size`	Line size for population curve.
`ind_line_alpha`	Alpha for individual curves.
`ind_line_size`	Line size for individual curves.
`...`	Additional arguments (currently unused).

Value

A ggplot2 object.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")

# Plot mean demand curve
plot(fit)

# Plot parameter distributions
plot(fit, type = "parameters")
```

`plot.beezdemand_nlme` *Plot Method for beezdemand_nlme Objects*

Description

Creates a ggplot2 visualization of a fitted `beezdemand_nlme` model, showing observed data points and/or model prediction lines.

Usage

```
## S3 method for class 'beezdemand_nlme'
plot(
  x,
  type = c("demand", "population", "individual", "both"),
  ids = NULL,
  show_observed = TRUE,
  observed_point_alpha = 0.6,
  show_pred = "population",
  n_points = 200,
  inv_fun = identity,
  facet = NULL,
  at = NULL,
  color_by = NULL,
  linetype_by = NULL,
  shape_by = NULL,
  x_trans = c("log10", "log", "linear", "pseudo_log"),
  y_trans = NULL,
  free_trans = 0.01,
  x_limits = NULL,
  y_limits = NULL,
  style = c("modern", "apa"),
  title = NULL,
  subtitle = NULL,
  x_lab = NULL,
  y_lab = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  observed_point_size = 2,
  pop_line_size = 1,
  ind_line_size = 0.6,
  pop_line_alpha = 0.9,
  ind_line_alpha = 0.3,
  ...
)
```

Arguments

x	A beezdemand_nlme object.
type	Plot type: "demand", "population", "individual", or "both".
ids	Optional vector of subject IDs to plot.
show_observed	Logical. If TRUE, plots the original data points. Default TRUE.
observed_point_alpha	Alpha for observed points. Default 0.6.
show_pred	Which prediction layers to plot: "population", "individual", or "both".
n_points	Integer. Number of points for prediction lines. Default 100.
inv_fun	Optional function to inverse-transform y-axis and predictions. Default identity.
facet	Optional faceting formula (e.g., ~ dose).
at	Optional named list giving values for continuous covariates used in the fixed-effects RHS. When building prediction grids for population- or individual- level lines, these values will be used. If not provided, the function will default to the median of each continuous covariate found in the original model data. Factor variables are always handled as grids (population) or observed combinations (individual) as before.
color_by	Optional character string: name of a factor to color lines and/or points by. Must be a column in x$data.
linetype_by	Optional character string: name of a factor for linetypes of population prediction lines if individual lines are also shown (otherwise applies to the shown lines). Must be a model factor in x$param_info$factors.
shape_by	Optional character string: name of a factor for shapes of observed points. Must be a column in x$data.
x_trans	Character. Transformation for x-axis. Default "log".
y_trans	Character. Transformation for y-axis. Default "log".
free_trans	Value used to display free (x = 0) on log scales. Use NULL to drop x <= 0 values instead.
x_limits	Optional numeric vector of length 2 for x-axis limits.
y_limits	Optional numeric vector of length 2 for y-axis limits.
style	Plot styling, passed to theme_beezdemand().
title	Optional plot title.

subtitle	Optional subtitle for the plot.
x_lab	Optional x-axis label.
y_lab	Optional y-axis label.
xlab	Deprecated alias for x_lab.
ylab	Deprecated alias for y_lab.
observed_point_size	Size for observed points. Default 2.
pop_line_size	Size for population prediction lines. Default 1.
ind_line_size	Size for individual prediction lines. Default 0.6.
pop_line_alpha	Alpha for population prediction lines. Default 0.9.
ind_line_alpha	Alpha for individual prediction lines. Default 0.3.
...	Additional arguments (currently unused).

Value

A ggplot2 object.

---

plot.cp_model_lm *Plot Method for Linear Cross-Price Demand Models*

Description

Creates a ggplot2 visualization of a fitted linear cross-price demand model (of class cp_model_lm). The plot overlays a prediction line over the observed data points. Axis transformations (e.g., "log10") can be applied. If the model includes group effects, separate lines will be drawn for each group.

Usage

```
## S3 method for class 'cp_model_lm'
plot(
  x,
  data = NULL,
  inv_fun = identity,
  n_points = 100,
  title = NULL,
  xlab = "Price",
  ylab = "Consumption",
  x_trans = "identity",
  y_trans = "identity",
  point_size = 3,
  ...
)
```

Arguments

x	A cp_model_lm object (as returned by fit_cp_linear(type = "fixed", ...)).
data	Optional data frame containing columns x and y to plot. If not provided, the function uses object$data if available.
inv_fun	Optional function to inverse-transform predictions. Default is identity. Not typically used for linear models but included for API consistency.
n_points	Number of points to create in the prediction grid. Default is 100.
title	Optional title for the plot; default is NULL.
xlab	Label for the x-axis. Default is "Price".
ylab	Label for the y-axis. Default is "Consumption".
x_trans	Transformation for the x-axis; one of "identity", "log10", or "pseudo_log". Default is "identity".
y_trans	Transformation for the y-axis; one of "identity", "log10", or "pseudo_log". Default is "identity".
point_size	Size of the data points in the plot. Default is 3.
...	Additional arguments passed to the generic predict method.

Value

A ggplot2 object displaying the fitted model predictions and observed data.

---

plot.cp_model_lmer *Plot Method for Mixed-Effects Cross-Price Demand Models*

Description

Creates a ggplot2 visualization of a fitted mixed-effects cross-price demand model (of class cp_model_lmer). This function allows you to plot:

Usage

```
## S3 method for class 'cp_model_lmer'
plot(
  x,
  data = NULL,
  inv_fun = identity,
  n_points = 100,
  title = NULL,
  xlab = "Price",
  ylab = "Consumption",
  x_trans = "identity",
  y_trans = "identity",
  point_size = 3,
  pred_type = c("fixed", "random", "all"),
  ...
)
```

Arguments

x	A cp_model_lmer object (as returned by fit_cp_linear(type = "mixed", ...)).
data	Optional data frame containing columns x and y to be plotted. If not provided, object$data is used.
inv_fun	Optional function to inverse-transform predictions. Default is identity. Not typically used for linear models but included for API consistency.
n_points	Number of points to use in creating the prediction grid. Default is 100.
title	Optional title for the plot; default is NULL.
xlab	Label for the x-axis. Default is "Price".
ylab	Label for the y-axis. Default is "Consumption".
x_trans	Transformation for the x-axis; one of "identity", "log10", or "pseudo_log". Default is "identity".
y_trans	Transformation for the y-axis; one of "identity", "log10", or "pseudo_log". Default is "identity".
point_size	Size of the observed data points. Default is 3.
pred_type	Character string specifying which prediction components to plot: "fixed" Plot only the fixed-effects (population) prediction. "random" Plot only the subject-specific predictions. "all" Plot both the fixed-effects and the subject-specific predictions. The default is "fixed".
...	Additional arguments passed to predict.cp_model_lmer.

Details

"fixed" Only the population-level (fixed-effects) prediction.

"random" Only the subject-specific predictions.

"all" Both: the fixed-effects and the subject-specific predictions.

If the model includes group effects, separate lines will be drawn for each group.

Value

A ggplot2 object displaying the observed data points along with the prediction curves.

`plot.cp_model_nls` *Plot a Cross-Price Demand Model (Nonlinear)*

Description

Plot a Cross-Price Demand Model (Nonlinear)

Usage

```
## S3 method for class 'cp_model_nls'
plot(
  x,
  data = NULL,
  inv_fun = identity,
  n_points = 100,
  title = NULL,
  xlab = "Price",
  ylab = "Consumption",
  x_trans = "identity",
  y_trans = "identity",
  point_size = 3,
  inverse_fun = deprecated(),
  ...
)
```

Arguments

`x`	A cross-price model object from fit_cp_nls with return_all=TRUE.
`data`	Optional data frame with x and y; if NULL, uses object$data.
`inv_fun`	Optional function to inverse-transform predictions. Default is `identity`.
`n_points`	Number of points used for prediction curve.
`title`	Optional plot title.
`xlab`	X-axis label.
`ylab`	Y-axis label.
`x_trans`	Transformation for x-axis: "identity", "log10", or "pseudo_log".
`y_trans`	Transformation for y-axis: "identity", "log10", or "pseudo_log".
`point_size`	Size of data points.
`inverse_fun`	**[Deprecated]** Use `inv_fun` instead.
`...`	Additional arguments (passed to predict).

Value

A ggplot2 object.

PlotCurve *Plot Curve*

Description

Creates a single plot object

Usage

```
PlotCurve(adf, dfrow, newdats, yscale = "log", style = c("modern", "apa"))
```

Arguments

adf	Data frame (long form) of purchase task data.
dfrow	A row of results from FitCurves
newdats	A newdat dataframe from FitCurves
yscale	Scaling of y axis. Default is "log". Can also take "linear"
style	Plot styling, passed to theme_beezdemand().

Details

Creates individual demand curves

Value

ggplot2 graphical object

Author(s)

Shawn Gilroy shawn.gilroy@temple.edu

Examples

```
## Creates a single plot from elements of an object created by FitCurves
if (interactive()) {
  fc <- FitCurves(apt, "hs", k = 2, detailed = TRUE)
  PlotCurve(fc$adfs[[1]], fc$dfres[1, ], fc$newdats[[1]])
}
```

PlotCurves *Plot Curves*

Description

Creates plots

Usage

```
PlotCurves(dat, outdir = NULL, device = "png", ending = NULL, ask = TRUE, ...)
```

Arguments

dat	FitCurves object with 4 elements (dfres, newdats, adfs, fits)
outdir	Directory where plots are saved
device	Type of file. Default is "png". Can be "pdf"
ending	Optional. Can specify to only plot through a certain number of datasets
ask	Can view plots one by one. If TRUE, plots will not save
...	Pass arguments to PlotCurve (for example yscale = c("log", "linear"))

Details

Creates and saves plots of individual demand curves

Value

Nothing

Author(s)

Brent Kaplan bkaplan.ku@gmail.com, Shawn Gilroy shawn.gilroy@temple.edu

Examples

```
## Interactively view plots from output from FitCurves
if (interactive()) {
  fc <- FitCurves(apt, "hs", k = 2, detailed = TRUE)
  PlotCurves(fc, ask = TRUE)
}
```

plot_qq *Plot Random Effects Q-Q*

Description

Creates Q-Q plots for random effects to assess normality assumptions.

Usage

```
plot_qq(object, which = NULL, ...)

## S3 method for class 'beezdemand_hurdle'
plot_qq(object, which = NULL, ...)

## S3 method for class 'beezdemand_nlme'
plot_qq(object, which = NULL, ...)
```

Arguments

object	A fitted model object with random effects (beezdemand_hurdle or beezdemand_nlme).
which	Character vector; which random effects to plot. Default is all.
...	Additional arguments (ignored).

Value

A ggplot2 object.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
plot_qq(fit)
```

plot_residuals *Plot Residual Diagnostics*

Description

Creates diagnostic plots for model residuals including residuals vs fitted, scale-location, and histogram of residuals.

Usage

```
plot_residuals(object, type = c("all", "fitted", "histogram", "qq"), ...)
```

Arguments

object	A fitted model object.
type	Character; type of residual plot. One of: • "fitted": Residuals vs fitted values • "histogram": Histogram of residuals • "qq": Q-Q plot of residuals • "all": All plots combined (default)
...	Additional arguments passed to plotting functions.

Value

A ggplot2 object or list of ggplot2 objects.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
plot_residuals(fit)
```

---

plot_subject *Plot Demand Curve for a Single Subject*

Description

Creates a demand curve plot for a single subject with optional observed data and population reference curve.

Usage

```
plot_subject(
  object,
  subject_id,
  prices = NULL,
  show_data = TRUE,
  show_population = TRUE,
  style = c("modern", "apa")
)
```

Arguments

object	An object of class beezdemand_hurdle.
subject_id	The ID of the subject to plot.
prices	Numeric vector of prices for plotting. If NULL, uses a sequence from 0 to max observed price.

show_data	Logical; if TRUE, overlay observed data points. Default is TRUE.
show_population	Logical; if TRUE, show population curve. Default is TRUE.
style	Plot styling, passed to theme_beezdemand().

Value

A ggplot2 object.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
plot_subject(fit, subject_id = "19")
```

---

predict.beezdemand_fixed

Predict Method for beezdemand_fixed

Description

Predict Method for beezdemand_fixed

Usage

```
## S3 method for class 'beezdemand_fixed'
predict(
  object,
  newdata = NULL,
  type = c("response", "link"),
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  ...
)
```

Arguments

object	A beezdemand_fixed object.
newdata	A data frame containing a price column matching the fitted object's x_var. If NULL, uses the unique observed prices when available.
type	One of "response" (default) or "link".
se.fit	Logical; if TRUE, includes a .se.fit column (currently NA because vcov is not available from legacy fixed fits).

interval	One of "none" (default) or "confidence". When requested, .lower/.upper are returned as NA because vcov is unavailable.
level	Confidence level when interval = "confidence". Currently used only for validation.
...	Unused.

Value

A tibble containing the original newdata columns, plus .fitted and, when requested, .se.fit and .lower/.upper. If newdata does not include an id column, predictions are returned for all subjects (cross product of newdata × subjects) unless k is subject-specific (k = "ind").

`predict.beezdemand_hurdle`

Predict Method for Hurdle Demand Models

Description

Returns predictions from a fitted hurdle demand model.

Usage

```
## S3 method for class 'beezdemand_hurdle'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "parameters", "probability", "demand"),
  prices = NULL,
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  ...
)
```

Arguments

object	An object of class beezdemand_hurdle.
newdata	Optional data frame containing a price column matching the fitted object's x_var. If newdata includes the id column, subject-specific predictions are returned; otherwise population predictions are returned. If newdata is NULL, returns predictions for all subjects across a price grid.
type	One of: "response" Predicted consumption (part II) "link" Predicted log-consumption (linear predictor of part II) "probability" Predicted probability of zero consumption (part I)

	"demand" Predicted expected consumption = (1 - P0) * response
	"parameters" Subject-specific parameters (no .fitted column)
prices	Optional numeric vector of prices used only when newdata = NULL.
se.fit	Logical; if TRUE, includes a .se.fit column (delta-method via sdreport when available).
interval	One of "none" (default) or "confidence".
level	Confidence level when interval = "confidence".
...	Unused.

Value

For type = "parameters", a tibble of subject-level parameters. Otherwise, a tibble containing the newdata columns plus .fitted and helper columns predicted_log_consumption, predicted_consumption, prob_zero, and expected_consumption. When requested, .se.fit and .lower/.upper are included.

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")

# Get subject-specific parameters
pars <- predict(fit, type = "parameters")

# Predict demand at specific prices
demand <- predict(fit, type = "demand", prices = c(0, 0.5, 1, 2, 5))
```

---

predict.beezdemand_nlme

Predict Method for beezdemand_nlme Objects

Description

Generates point predictions from a fitted beezdemand_nlme model. Predictions can be made at the population level (fixed effects only) or group/subject level (fixed + random effects). The output scale depends on the equation_form used during model fitting and whether inv_fun is applied.

Usage

```
## S3 method for class 'beezdemand_nlme'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "population", "individual"),
```

```
  level = 0,
  inv_fun = identity,
  se.fit = FALSE,
  interval = c("none", "confidence"),
  interval_level = 0.95,
  ...
)
```

Arguments

object	A beezdemand_nlme object.
newdata	Optional data frame for which to make predictions. Must contain x_var and all factors specified in the original model. If group-level predictions are desired (level=1), the id_var column from the original fit must also be present in newdata and its levels should correspond to those in the original data for meaningful random effect application. If NULL, predictions are made for the data used in fitting the model.
type	One of "response" (default), "link", "population", or "individual". "population" and "individual" are aliases that set level to 0 or 1, respectively.
level	Integer, prediction level for nlme::predict.nlme(): • 0: Population predictions (based on fixed effects only). • 1 (or higher, up to number of grouping levels in model): Group-specific predictions (fixed effects + random effects for the specified id_var level). Default is 0.
inv_fun	Optional function to inverse-transform the predictions. Example: If y_var was log10-transformed during fitting and equation_form like "zben" produces predictions on that log10 scale, inv_fun = function(x) 10^x would convert predictions back to the original consumption scale. If equation_form was "simplified" (which models raw Y), inv_fun might be identity or not needed if predictions are already on the desired scale.
se.fit	Logical; if TRUE, includes a .se.fit column (currently NA because standard errors are not implemented for beezdemand_nlme predictions).
interval	One of "none" (default) or "confidence". When requested, .lower/.upper are returned as NA.
interval_level	Confidence level when interval = "confidence". Currently used only for validation.
...	Additional arguments passed to nlme::predict.nlme().

Value

A tibble containing the original newdata columns plus .fitted. When requested, .se.fit and .lower/.upper are included (currently NA).

See Also

predict.nlme

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", equation_form = "zben")
# Population-level predictions
preds <- predict(fit, level = 0)

# Subject-level predictions
preds_subj <- predict(fit, level = 1)
```

`predict.cp_model_lm` *Predict method for cp_model_lm objects.*

Description

Predict method for cp_model_lm objects.

Usage

```
## S3 method for class 'cp_model_lm'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	A cp_model_lm object.
`newdata`	Data frame containing new x values.
`...`	Additional arguments.

Value

Data frame with predictions.

`predict.cp_model_lmer` *Predict from a Mixed-Effects Cross-Price Demand Model*

Description

Generates predictions from a mixed-effects cross-price demand model (of class `cp_model_lmer`). The function supports two modes:

Usage

```
## S3 method for class 'cp_model_lmer'
predict(object, newdata = NULL, pred_type = c("fixed", "random"), ...)
```

Arguments

object	A cp_model_lmer object (as returned by fit_cp_linear(type = "mixed", ...)).
newdata	A data frame containing at least an x column. For pred_type = "random", an id column is required. If absent, the function extracts unique ids from object$data and expands the grid accordingly. If no ids are available, a default id of 1 is used (with a warning).
pred_type	Character string specifying the type of prediction: either "fixed" (population-level) or "random" (subject-specific). The default is "fixed".
...	Additional arguments passed to the underlying predict function.

Details

"fixed" Returns predictions based solely on the fixed-effects component (using re.form = NA).

"random" Returns subject-specific predictions (fixed plus random effects) (using re.form = NULL).

Value

A data frame containing all columns of newdata plus a column y_pred with the corresponding predictions.

Examples

```
data(etm)
fit <- fit_cp_linear(etm, type = "mixed")
new_prices <- data.frame(x = c(2, 4, 8, 16, 32, 64))
predict(fit, newdata = new_prices, pred_type = "fixed")
```

---

predict.cp_model_nls *Predict from a Cross-Price Demand Model (Nonlinear)*

Description

Predict from a Cross-Price Demand Model (Nonlinear)

Usage

```
## S3 method for class 'cp_model_nls'
predict(
  object,
  newdata = NULL,
  inv_fun = identity,
  inverse_fun = deprecated(),
  ...
)
```

Arguments

object	A cross-price model object from fit_cp_nls with return_all=TRUE.
newdata	A data frame containing an 'x' column.
inv_fun	Optional inverse transformation function. Default is identity.
inverse_fun	**[Deprecated]** Use inv_fun instead.
...	Additional arguments.

Value

A data frame with x values and predicted y values.

---

print.anova.beezdemand_hurdle

Print Method for ANOVA Comparisons

Description

Print Method for ANOVA Comparisons

Usage

```
## S3 method for class 'anova.beezdemand_hurdle'
print(x, digits = 4, ...)
```

Arguments

x	An anova.beezdemand object.
digits	Number of significant digits to print.
...	Additional arguments (ignored).

Value

Invisibly returns the input object x.

print.beezdemand_comparison

Print method for beezdemand_comparison objects

Description

Print method for beezdemand_comparison objects

Usage

```
## S3 method for class 'beezdemand_comparison'
print(x, digits = 3, ...)
```

Arguments

x	A beezdemand_comparison object.
digits	Number of significant digits to display for estimates.
...	Additional arguments (unused).

Value

Invisibly returns the input object x.

print.beezdemand_diagnostics

Print Method for Model Diagnostics

Description

Print Method for Model Diagnostics

Usage

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

Arguments

x	A beezdemand_diagnostics object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object x.

`print.beezdemand_fixed`

Print Method for beezdemand_fixed

Description

Print Method for beezdemand_fixed

Usage

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

Arguments

`x`	A beezdemand_fixed object
`...`	Additional arguments (ignored)

Value

Invisibly returns the input object `x`.

`print.beezdemand_hurdle`

Print Method for Hurdle Demand Model

Description

Print Method for Hurdle Demand Model

Usage

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

Arguments

`x`	An object of class `beezdemand_hurdle`.
`...`	Additional arguments (currently unused).

Value

Invisibly returns the input object `x`.

print.beezdemand_model_comparison *Print Method for Model Comparison*

Description

Print Method for Model Comparison

Usage

```
## S3 method for class 'beezdemand_model_comparison'
print(x, digits = 4, ...)
```

Arguments

x	A beezdemand_model_comparison object.
digits	Number of significant digits to print.
...	Additional arguments (ignored).

Value

Invisibly returns the input object x.

print.beezdemand_nlme *Print Method for beezdemand_nlme Objects*

Description

Provides a concise summary of a beezdemand_nlme object, typically displaying the call, model specifications, and key results from the nlme fit if successful.

Usage

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

Arguments

x	An object of class beezdemand_nlme.
digits	Minimal number of significant digits, see print.default.
...	Additional arguments passed to print.nlme if the model exists.

Value

Invisibly returns the original object x.

Examples

```
data(ko)
fit <- fit_demand_mixed(ko, y_var = "y_ll4", x_var = "x",
                        id_var = "monkey", equation_form = "zben")
print(fit)
```

`print.beezdemand_summary`

Print Method for beezdemand Summary Objects

Description

Fallback print method for summary objects inheriting from `beezdemand_summary`. Specific summary classes should implement their own `print.summary.*` methods for detailed output; this provides a minimal fallback.

Usage

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

Arguments

`x`	A summary object with class including `beezdemand_summary`.
`...`	Additional arguments (unused).

Value

Invisibly returns `x`.

`print.beezdemand_systematicity`

Print Method for beezdemand_systematicity

Description

Print Method for beezdemand_systematicity

Usage

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

Arguments

x	A beezdemand_systematicity object
...	Additional arguments (ignored)

Value

Invisibly returns the input object x.

---

print.cp_posthoc *Print method for cp_posthoc objects*

Description

Print method for cp_posthoc objects

Usage

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

Arguments

x	A cp_posthoc object
...	Additional arguments passed to print

Value

Invisibly returns the input object x.

---

print.summary.beezdemand_fixed *Print Method for summary.beezdemand_fixed*

Description

Print Method for summary.beezdemand_fixed

Usage

```
## S3 method for class 'summary.beezdemand_fixed'
print(x, digits = 4, n = 20, ...)
```

Arguments

x	A summary.beezdemand_fixed object
digits	Number of significant digits to print
n	Number of subjects (ids) to print (default 20)
...	Additional arguments (ignored)

Value

Invisibly returns the input object x.

---

print.summary.beezdemand_hurdle

Print Summary of Hurdle Demand Model

Description

Print Summary of Hurdle Demand Model

Usage

```
## S3 method for class 'summary.beezdemand_hurdle'
print(x, digits = 4, n = Inf, ...)
```

Arguments

x	An object of class summary.beezdemand_hurdle.
digits	Number of significant digits to print. Default is 4.
n	Number of rows to print for any tables (unused for this class).
...	Additional arguments passed to print.

Value

Invisibly returns the input object x.

print.summary.beezdemand_nlme

Print method for summary.beezdemand_nlme

Description

Print method for summary.beezdemand_nlme

Usage

```
## S3 method for class 'summary.beezdemand_nlme'
print(x, digits = 4, n = Inf, ...)
```

Arguments

x	A summary.beezdemand_nlme object
digits	Number of significant digits to print
n	Number of rows to print for any tables (unused for this class).
...	Additional arguments (ignored)

Value

Invisibly returns the input object x.

print.summary.beezdemand_systematicity

Print Method for summary.beezdemand_systematicity

Description

Print Method for summary.beezdemand_systematicity

Usage

```
## S3 method for class 'summary.beezdemand_systematicity'
print(x, n = 20, ...)
```

Arguments

x	A summary.beezdemand_systematicity object
n	Number of IDs to display (default 20)
...	Additional arguments (ignored)

Value

Invisibly returns the input object x.

`print.summary.cp_model_lm` *Print method for summary.cp_model_lm objects.*

Description

Print method for summary.cp_model_lm objects.

Usage

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

Arguments

`x`	A `summary.*` object
`...`	Unused

Value

Invisibly returns the input object `x`.

`print.summary.cp_model_lmer` *Print method for summary.cp_model_lmer objects.*

Description

Print method for summary.cp_model_lmer objects.

Usage

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

Arguments

`x`	A `summary.*` object
`...`	Unused

Value

Invisibly returns the input object `x`.

print.summary.cp_model_nls

Print method for summary.cp_model_nls objects

Description

Print method for summary.cp_model_nls objects

Usage

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

Arguments

x	A summary.* object
...	Unused

Value

Invisibly returns the input object x.

print.summary.cp_unsystematic

Print Method for Cross-Price Unsystematic Summary

Description

Print Method for summary.cp_unsystematic

Usage

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

Arguments

x	A summary.cp_unsystematic object.
...	Additional arguments (ignored).

Value

Invisibly returns the input object x.

print_mc_summary *Print Monte Carlo Simulation Results*

Description

Prints a formatted summary of Monte Carlo simulation results.

Usage

```
print_mc_summary(mc_results, digits = 3)
```

Arguments

mc_results	Output from run_hurdle_monte_carlo.
digits	Number of digits to display. Default is 3.

Value

Invisibly returns the input mc_results object.

Examples

```
mc_results <- run_hurdle_monte_carlo(n_sim = 50, n_subjects = 100, seed = 123)
print_mc_summary(mc_results)
```

pseudo_ll4_trans *Create a Pseudo-Log LL4 Transformation Object for ggplot2*

Description

Generates a scales::trans object using the ll4 transformation. This transformation object can be passed to the trans argument of ggplot2::scale_x_continuous or ggplot2::scale_y_continuous. It's designed for non-negative data and handles zero values gracefully. The "pseudo" aspect is conceptual, similar to pseudo_log_trans in that it handles a range including zero, but the transformation is ll4.

Usage

```
pseudo_ll4_trans(lambda = 4)
```

Arguments

lambda	A positive numeric scalar, the lambda parameter for the ll4 transformation. Default is 4.

Value

A trans object (from the scales package).

See Also

ll4, ll4_inv, trans_new

Examples

```
if (require(ggplot2) && require(scales)) {
  set.seed(123)
  df <- data.frame(
    x_vals = c(0, 0.01, 0.1, 1, 10, 100, 1000, NA), # Include 0 and NA
    y_vals = c(0, 10, 50, 100, 500, 1000, 2000, 50)
  )

  # Using pseudo_ll4_trans for the y-axis
  ggplot(df, aes(x = x_vals, y = y_vals)) +
    geom_point() +
    scale_y_continuous(trans = pseudo_ll4_trans(lambda = 4),
                       name = "Y-Values (Pseudo-LL4 Scale)") +
    ggtitle("Y-Axis with Pseudo-LL4 Transformation")

  # Using pseudo_ll4_trans for the x-axis
  ggplot(df, aes(x = x_vals, y = y_vals)) +
    geom_point() +
    scale_x_continuous(trans = pseudo_ll4_trans(lambda = 2), # Different lambda
                       name = "X-Values (Pseudo-LL4 Scale)") +
    ggtitle("X-Axis with Pseudo-LL4 Transformation")
}
```

---

ranef.beezdemand_nlme *Extract Random Effects from a beezdemand_nlme Model*

Description

S3 method for ranef for objects of class beezdemand_nlme. Extracts the random effects from the fitted nlme model.

Usage

```
## S3 method for class 'beezdemand_nlme'
ranef(object, ...)
```

Arguments

object	A beezdemand_nlme object.
...	Additional arguments passed to nlme::ranef().

Value

A data frame (or list of data frames if multiple levels of grouping) of random effects, as returned by `ranef.nlme()`.

See Also

`coef.beezdemand_nlme`, `fixef.beezdemand_nlme`

---

ranef.cp_model_lmer *Extract Random Effects from Mixed-Effects Cross-Price Model*

Description

Extract Random Effects from Mixed-Effects Cross-Price Model

Usage

```
## S3 method for class 'cp_model_lmer'
ranef(object, ...)
```

Arguments

`object`	A cp_model_lmer object
`...`	Additional arguments passed to ranef

Value

List of random effects

RecodeOutliers *Recode Outliers*

Description

Recodes outliers

Usage

```
RecodeOutliers(df, outval = 3.29, unitshigher = 0)
```

Arguments

`df`	A dataframe of consumption values
`outval`	Values greater/less than or equal to this number (specified in standard deviations) will be recoded. Default is 3.29SD as specified by Tabachnick and Fidell (2013)
`unitshigher`	Outliers identified by outval will be coded to a certain number of units higher/lower than the greatest nonoutlier value. Default is 0 units.

Details

Recodes outliers using Tabachnick and Fidell's (2013) criteria. A variable is standardized and values that are greater/less than or equal to a specified outlier value (specified in standard deviations; default 3.29SD) are recoded to a certain number of units (default 0) higher/lower than the greatest nonoutlier value. Disregards 'NA' values.

Value

Invisibly, a dataframe with original and recoded (if any) values

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## If any outliers are detected, they would be coded as 1 unit higher

emp <- GetEmpirical(apt)
RecodeOutliers(emp[, c(2:6)], unitshigher = 1)
```

ReplaceZeros *Replace Zeros*

Description

Replaces 0 values

Usage

```
ReplaceZeros(dat, nrepl = 1, replnum = 0.01)
```

Arguments

dat	Dataframe (long form)
nrepl	Number of zeros to replace with replacement value (replnum). Can accept either a number or "all" if all zeros should be replaced. Default is to replace the first zero only.
replnum	Value to replace zeros. Default is .01

Details

Replaces specified number of 0s with replacement value.

Value

Dataframe (long form)

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## Replace all zeros with .01
ReplaceZeros(apt, nrepl = "all", replnum = .01)
```

`run_hurdle_monte_carlo` *Run Monte Carlo Simulation Study for Hurdle Demand Model*

Description

Runs a Monte Carlo simulation study to assess model performance, including bias, standard error estimates, and confidence interval coverage.

Usage

```
run_hurdle_monte_carlo(
  n_sim = 100,
  n_subjects = 100,
  true_params = NULL,
  n_random_effects = 2,
  prices = seq(0, 11, by = 0.5),
  stop_at_zero = TRUE,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

`n_sim`	Number of simulated datasets. Default is 100.
`n_subjects`	Number of subjects per dataset. Default is 100.
`true_params`	Named list of true parameter values. If NULL, defaults are used from `simulate_hurdle_data`.
`n_random_effects`	Number of random effects (2 or 3). Default is 2.
`prices`	Numeric vector of prices. Default is seq(0, 11, by = 0.5).
`stop_at_zero`	Logical; if TRUE in simulation, subjects stop after first zero. Default is TRUE.
`verbose`	Logical; print progress. Default is TRUE.
`seed`	Random seed for reproducibility.

Value

A list with:

estimates Data frame of parameter estimates from each simulation

true_params True parameter values used

summary Summary statistics including bias, SE ratio, and coverage

n_converged Number of simulations that converged

n_sim Total number of simulations attempted

See Also

simulate_hurdle_data, fit_demand_hurdle

Examples

```
# Run small simulation study (for demonstration)
mc_results <- run_hurdle_monte_carlo(n_sim = 10, n_subjects = 50, seed = 123)

# View summary
print(mc_results$summary)

# Check convergence rate
cat("Convergence rate:", mc_results$n_converged / mc_results$n_sim, "\n")
```

---

scale_color_beezdemand

beezdemand Color Scale (Discrete)

Description

beezdemand Color Scale (Discrete)

Usage

```
scale_color_beezdemand(...)
```

Arguments

...	Additional arguments passed to ggplot2::scale_color_manual.

Value

A ggplot2 discrete color scale.

Examples

```
library(ggplot2)
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() +
  scale_color_beezdemand()
```

`scale_fill_beezdemand` *beezdemand Fill Scale (Discrete)*

Description

beezdemand Fill Scale (Discrete)

Usage

```
scale_fill_beezdemand(...)
```

Arguments

`...`	Additional arguments passed to `ggplot2::scale_fill_manual`.

Value

A ggplot2 discrete fill scale.

Examples

```
library(ggplot2)
ggplot(iris, aes(Species, Sepal.Length, fill = Species)) +
  geom_boxplot() +
  scale_fill_beezdemand()
```

`scale_ll4` *Create an LL4-like Scale for ggplot2 Axes*

Description

This function generates a `ggplot2` continuous scale that applies the `ll4` transformation (and its inverse `ll4_inv`) to an axis. This is useful for visualizing data spanning multiple orders of magnitude while handling zeros.

Usage

```
scale_ll4(..., lambda = 4)
```

Arguments

...	Arguments passed on to ggplot2::scale_y_continuous or ggplot2::scale_x_continuous (e.g., name, breaks, labels).
lambda	A positive numeric scalar, the lambda parameter for the ll4 transformation. Default is 4.

Value

A ggplot2 scale object.

See Also

ll4, ll4_inv, trans_new

Examples

```
if (require(ggplot2) && require(scales)) {
  set.seed(123)
  df <- data.frame(
    x = 1:100,
   y_raw = c(0, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, sample(1:2000, 90, replace = TRUE))
  )

  # Plot with y-axis on LL4 scale
  ggplot(df, aes(x = x, y = y_raw)) +
    geom_point() +
    scale_ll4(name = "Y-axis (LL4 Scale)", lambda = 4) +
    ggtitle("Data with LL4 Transformed Y-Axis")

  # Can also be used for x-axis by replacing scale_y_continuous in its definition
  # Or by creating a scale_x_ll4 variant.
}
```

SimulateDemand *Simulate Demand Data*

Description

Simulate demand data

Usage

```
SimulateDemand(nruns = 10, setparams, sdindex, x, outdir = NULL, fn = NULL)
```

Arguments

nruns	Number of runs. Default value is 10
setparams	A 6x1 matrix (or 6 element vector) containing (in order) mean log10alpha, sd log10alpha, mean log10q0, sd log10q0, k, sd of consumption values across all prices
sdindex	A vector of n length of sd consumption values for n prices
x	A vector of n prices
outdir	Optional. Directory to save results. Must end with a "/"
fn	Optional. Filename of saved RData object

Details

Generates and saves simulated datasets in the manner specified in Koffarnus, Franck, Stein, & Bickel (2015).

Value

Invisibly a list consisting of: rounded consumption values, unrounded consumption values, simulation parameters, and inState and outState of seeds.

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
## set values
setparams <- vector(length = 4)
setparams <- c(-2.5547, .702521, 1.239893, .320221, 3.096, 1.438231)
names(setparams) <- c("alphalm", "alphalsd", "q0lm", "q0lsd", "k", "yvalssd")
sdindex <- c(2.1978, 1.9243, 1.5804, 1.2465, 0.8104, 0.1751, 0.0380, 0.0270)
x <- c(.1, 1, 3, 10, 30, 100, 300, 1000)
set.seed(1234)
sim <- SimulateDemand(nruns = 1, setparams = setparams, sdindex = sdindex, x = x)
sim
```

---

simulate_hurdle_data *Simulate Data from Two-Part Mixed Effects Hurdle Demand Model*

Description

Generates simulated demand data from the two-part hurdle model. Useful for Monte Carlo simulation studies, power analyses, and model validation.

Usage

```
simulate_hurdle_data(
  n_subjects = 100,
  prices = seq(0, 11, by = 0.5),
  beta0 = -2,
  beta1 = 1,
  log_q0 = log(10),
  logQ0 = deprecated(),
  k = 2,
  alpha = 0.5,
  sigma_a = 1,
  sigma_b = 0.5,
  sigma_c = 0.1,
  rho_ab = 0.3,
  rho_ac = 0,
  rho_bc = 0,
  sigma_e = 0.3,
  epsilon = 0.001,
  n_random_effects = 2,
  stop_at_zero = TRUE,
  seed = NULL
)
```

Arguments

n_subjects	Number of subjects to simulate. Default is 100.
prices	Numeric vector of prices at which to simulate consumption. Default is seq(0, 11, by = 0.5).
beta0	Intercept for Part I (logistic). Default is -2.
beta1	Slope for Part I (on log(price + epsilon)). Default is 1.
log_q0	Log of intensity parameter (Q0). Default is log(10), meaning Q0 = 10. To specify Q0 directly, use log_q0 = log(your_Q0).
logQ0	**[Deprecated]** Use log_q0 instead.
k	Scaling parameter for demand decay. Default is 2.
alpha	Elasticity parameter controlling rate of demand decay. Default is 0.5.
sigma_a	Standard deviation of random intercept for Part I. Default is 1.
sigma_b	Standard deviation of random intercept for Part II. Default is 0.5.
sigma_c	Standard deviation of random slope for alpha (only used if n_random_effects = 3). Default is 0.1.
rho_ab	Correlation between a_i and b_i. Default is 0.3.
rho_ac	Correlation between a_i and c_i. Default is 0.
rho_bc	Correlation between b_i and c_i. Default is 0.
sigma_e	Residual standard deviation. Default is 0.3.
epsilon	Small constant for log(price + epsilon). Default is 0.001.

`n_random_effects`
: Number of random effects (2 or 3). Default is 2.

`stop_at_zero`
: Logical; if TRUE, stop generating observations for a subject once zero consumption is observed. This means subjects will have varying numbers of observations. Set to FALSE to generate all prices for all subjects. Default is TRUE.

`seed`
: Optional random seed for reproducibility.

Details

The simulation follows Zhao et al. (2016):

Part I (Zero vs Positive):

$$logit(P(Y = 0)) = \beta_0 + \beta_1 \cdot \log(price + \epsilon) + a_i$$

Part II (Positive Consumption):

$$\log(Y|Y > 0) = (\log Q_0 + b_i) + k \cdot (\exp(-(\alpha + c_i) \cdot price) - 1) + \epsilon$$

Random effects (a_i, b_i) or (a_i, b_i, c_i) are drawn from a multivariate normal distribution with the specified variances and correlations.

Value

A data frame with columns:

id Subject identifier

x Price value

y Simulated consumption (may include zeros)

delta Indicator for zero consumption (1 = zero, 0 = positive)

a_i Subject-specific random effect for Part I

b_i Subject-specific random effect for Part II

c_i Subject-specific random effect for alpha (if n_random_effects = 3)

See Also

`fit_demand_hurdle`, `run_hurdle_monte_carlo`

Examples

```
# Simulate with default parameters (2 RE model)
sim_data <- simulate_hurdle_data(n_subjects = 100, seed = 123)
head(sim_data)

# Simulate with custom prices
apt_prices <- c(0, 0.25, 0.5, 1, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 9, 10)
sim_apt <- simulate_hurdle_data(n_subjects = 100, prices = apt_prices, seed = 123)

# Simulate with custom parameters (Q0 = 15, alpha = 0.1)
```

```
sim_custom <- simulate_hurdle_data(
  n_subjects = 100,
  log_q0 = log(15),
  alpha = 0.1,
  seed = 123
)

# Simulate 3 RE model
sim_3re <- simulate_hurdle_data(
  n_subjects = 100,
  n_random_effects = 3,
  sigma_c = 0.1,
  seed = 456
)
```

`summary.beezdemand_fixed` *Summary Method for beezdemand_fixed*

Description

Summary Method for beezdemand_fixed

Usage

```
## S3 method for class 'beezdemand_fixed'
summary(object, report_space = c("natural", "log10"), ...)
```

Arguments

`object`	A beezdemand_fixed object
`report_space`	Character. Reporting space for core parameters. One of: • `"natural"`: report natural-scale parameters (default) • `"log10"`: report `log10()`-scale parameters when defined
`...`	Additional arguments (ignored)

Value

A summary.beezdemand_fixed object (inherits from beezdemand_summary)

`summary.beezdemand_hurdle` *Summarize a Hurdle Demand Model Fit*

Description

Provides a summary of a fitted hurdle demand model, including fixed effects, variance components, correlations, and fit statistics.

Usage

```
## S3 method for class 'beezdemand_hurdle'
summary(object, report_space = c("natural", "log10", "internal"), ...)
```

Arguments

`object`	An object of class `beezdemand_hurdle` from `fit_demand_hurdle`.
`report_space`	Character. Reporting space for core demand parameters. One of: • `"internal"`: report internal/fitting parameters (default internal naming) • `"natural"`: report natural-scale parameters when a natural mapping exists • `"log10"`: report `log10()`-scale parameters when a mapping exists
`...`	Additional arguments (currently unused).

Value

An object of class `summary.beezdemand_hurdle` (also inherits from `beezdemand_summary`) containing:

call The original function call

model_class "beezdemand_hurdle"

backend "TMB"

coefficients Tibble of fixed effects with estimates, SEs, z-values, p-values

coefficients_matrix Matrix form for printing (legacy compatibility)

variance_components Matrix of variance/covariance estimates

correlations Matrix of correlation estimates

n_subjects Number of subjects

nobs Number of observations

converged Logical indicating convergence

logLik Log-likelihood at convergence

AIC Akaike Information Criterion

BIC Bayesian Information Criterion

group_metrics Group-level Pmax and Omax

individual_metrics Summary of individual-level parameters

notes Character vector of warnings/notes

Examples

```
data(apt)
fit <- fit_demand_hurdle(apt, y_var = "y", x_var = "x", id_var = "id")
summary(fit)
```

`summary.beezdemand_nlme` *Summary method for beezdemand_nlme*

Description

Returns a structured summary object containing model coefficients, fit statistics, and random effects information.

Usage

```
## S3 method for class 'beezdemand_nlme'
summary(object, report_space = c("natural", "log10"), ...)
```

Arguments

`object`	A beezdemand_nlme object
`report_space`	Character. Reporting space for core parameters. One of `"natural"` or `"log10"` (default depends on `param_space` used for fitting).
`...`	Additional arguments (passed to summary.nlme)

Value

A `summary.beezdemand_nlme` object (inherits from `beezdemand_summary`) with fields including:

- `call`: The original function call
- `model_class`: "beezdemand_nlme"
- `backend`: "nlme"
- `equation_form`: The equation form used ("zben" or "simplified")
- `coefficients`: Tibble of fixed effects with std.error, statistic, p.value
- `random_effects`: VarCorr output for random effects
- `logLik`, `AIC`, `BIC`: Model fit statistics

summary.beezdemand_systematicity *Summary Method for beezdemand_systematicity*

Description

Summary Method for beezdemand_systematicity

Usage

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

Arguments

object	A beezdemand_systematicity object
...	Additional arguments (ignored)

Value

A summary.beezdemand_systematicity object

summary.cp_model_lm *Summary method for cp_model_lm objects.*

Description

Summary method for cp_model_lm objects.

Usage

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

Arguments

object	A cp_model_lm object.
...	Additional arguments.

Value

A list summarizing the linear model.

summary.cp_model_lmer *Summary method for cp_model_lmer objects.*

Description

Summary method for cp_model_lmer objects.

Usage

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

Arguments

object	A cp_model_lmer object.
...	Additional arguments.

Value

A list summarizing the mixed-effects model.

summary.cp_model_nls *Summarize a Cross-Price Demand Model (Nonlinear)*

Description

Summarize a Cross-Price Demand Model (Nonlinear)

Usage

```
## S3 method for class 'cp_model_nls'
summary(object, inv_fun = identity, inverse_fun = deprecated(), ...)
```

Arguments

object	A cross-price model object from fit_cp_nls with return_all=TRUE.
inv_fun	Optional function to inverse-transform predictions (e.g., ll4_inv). Default is identity.
inverse_fun	**[Deprecated]** Use inv_fun instead.
...	Additional arguments (unused).

Value

A list containing model summary information.

`summary.cp_unsystematic` *Summarize Cross-Price Unsystematic Data Check Results*

Description

Summarizes systematic and unsystematic patterns from multiple calls to `check_unsystematic_cp()`. This includes overall proportions, trend and bounce direction counts, and optionally summaries by subject or group.

Usage

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

Arguments

`object`	A data frame containing results from multiple `check_unsystematic_cp()` calls, with at minimum the columns 'delta_direction', 'bounce_direction', and 'bounce_any'. Columns 'id', 'group', 'reversals', and 'returns' are optional but allow extended summaries.
`...`	Additional arguments (currently unused)

Value

A list of class `summary.cp_unsystematic` with the following elements:

total_patterns Number of total patterns examined.

systematic_count Count of systematic patterns (no bounce).

unsystematic_count Count of unsystematic patterns (bounce detected).

systematic_percent Proportion of systematic patterns.

unsystematic_percent Proportion of unsystematic patterns.

trend_counts Breakdown of trend directions.

bounce_counts Breakdown of bounce directions.

reversal_summary (Optional) Summary of zero-reversal patterns, if present in input.

return_summary (Optional) Summary of zero-return patterns, if present in input.

group_summary (Optional) Summary stats by 'group'.

problem_ids (Optional) Top IDs with unsystematic patterns.

systematic-wrappers *Systematicity Check Wrappers*

Description

Modern wrappers for systematicity checks with unified output vocabulary.

theme_apa *APA Theme*

Description

APA theme for ggplot

Usage

```
theme_apa(plot.box = FALSE)
```

Arguments

plot.box	Boolean for a box around the plot

Details

Theme for ggplot graphics that closely align with APA formatting

Value

ggplot theme

Author(s)

Brent Kaplan bkaplan.ku@gmail.com

Examples

```
p <- ggplot2::ggplot(apt, ggplot2::aes(x = x, y = y)) +
  ggplot2::geom_point()
p + theme_apa()
```

theme_beezdemand *beezdemand Plot Theme*

Description

beezdemand Plot Theme

Usage

```
theme_beezdemand(
  style = c("modern", "apa"),
  base_size = 11,
  base_family = "sans"
)
```

Arguments

style	Character. One of "modern" or "apa".
base_size	Base font size (default: 11).
base_family	Base font family (default: "sans").

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() +
  theme_beezdemand()
```

tidy.beezdemand_fixed *Tidy Method for beezdemand_fixed*

Description

Tidy Method for beezdemand_fixed

Usage

```
## S3 method for class 'beezdemand_fixed'
tidy(x, report_space = c("natural", "log10"), ...)
```

Arguments

x	A beezdemand_fixed object
report_space	Character. Reporting space for core parameters. One of "natural" (default) or "log10".
...	Additional arguments (ignored)

Value

A tibble of model coefficients with columns: id, term, estimate, std.error, statistic, p.value, component

---

tidy.beezdemand_hurdle

Tidy a beezdemand_hurdle Model

Description

Returns a tibble of model coefficients in tidy format.

Usage

```
## S3 method for class 'beezdemand_hurdle'
tidy(x, report_space = c("natural", "log10", "internal"), ...)
```

Arguments

x	An object of class beezdemand_hurdle.
report_space	Character. Reporting space for core demand parameters. One of "internal", "natural", or "log10".
...	Additional arguments (currently unused).

Value

A tibble with columns:

term Parameter name

estimate Point estimate

std.error Standard error

statistic z-value

p.value P-value

component One of "zero_probability", "consumption", "variance", or "fixed"

`tidy.beezdemand_nlme` *Tidy method for beezdemand_nlme*

Description

Tidy method for beezdemand_nlme

Usage

```
## S3 method for class 'beezdemand_nlme'
tidy(
  x,
  effects = c("fixed", "ran_pars"),
  report_space = c("natural", "log10"),
  ...
)
```

Arguments

`x`	A beezdemand_nlme object
`effects`	Which effects to include: "fixed" (default), "ran_pars", or both
`report_space`	Character. Reporting space for core parameters. One of `"natural"` or `"log10"` (default depends on `param_space` used for fitting).
`...`	Additional arguments (ignored)

Value

A tibble of model coefficients with columns:

- `term`: Parameter name
- `estimate`: Point estimate
- `std.error`: Standard error
- `statistic`: t-value
- `p.value`: P-value
- `component`: "fixed" or "variance"

tidy.beezdemand_systematicity *Tidy Method for beezdemand_systematicity*

Description

Tidy Method for beezdemand_systematicity

Usage

```
## S3 method for class 'beezdemand_systematicity'
tidy(x, ...)
```

Arguments

x	A beezdemand_systematicity object
...	Additional arguments (ignored)

Value

The per-subject results tibble

tidy.cp_model_lm *Extract coefficients from a linear cross-price model in tidy format*

Description

Extract coefficients from a linear cross-price model in tidy format

Usage

```
## S3 method for class 'cp_model_lm'
tidy(x, ...)
```

Arguments

x	A cp_model_lm object.
...	Additional arguments (unused).

Value

A tibble with columns: term, estimate, std.error, statistic, p.value.

tidy.cp_model_lmer *Extract coefficients from a mixed-effects cross-price model in tidy format*

Description

Extract coefficients from a mixed-effects cross-price model in tidy format

Usage

```
## S3 method for class 'cp_model_lmer'
tidy(x, effects = c("fixed", "random", "ran_pars"), ...)
```

Arguments

x	A cp_model_lmer object.
effects	Which effects to return: "fixed" (default), "random", or "ran_pars".
...	Additional arguments passed to broom.mixed::tidy.

Value

A tibble with tidy coefficient information.

tidy.cp_model_nls *Convert a cross-price model to a tidy data frame of coefficients*

Description

This function extracts model coefficients from a cross-price demand model into a tidy data frame format, following the conventions of the broom package. It handles cases where model fitting failed gracefully, returning an empty data frame with the expected structure.

Usage

```
## S3 method for class 'cp_model_nls'
tidy(x, ...)
```

Arguments

x	A model object from fit_cp_nls or fit_cp_linear
...	Additional arguments (unused)

Value

A data frame with one row per coefficient, containing columns:

`term`	The name of the model parameter
`estimate`	The estimated coefficient value
`std.error`	The standard error of the coefficient
`statistic`	The t-statistic for the coefficient
`p.value`	The p-value for the coefficient

Examples

```
data(etm)
fit <- fit_cp_nls(etm, equation = "exponentiated")
tidy(fit)
```

Index