

Package: beezdiscounting (via r-universe)

October 17, 2026

Title Behavioral Economic Easy Discounting

Version 0.4.0.9000

Date 2026-08-24

Description Facilitates some of the analyses performed in studies of behavioral economic discounting. The package supports scoring of the 21- and 27-Item Monetary Choice Questionnaires (see Kaplan et al., 2016; <[doi:10.1007/s40614-016-0070-9](https://doi.org/10.1007/s40614-016-0070-9)>), scoring of the 30-Item Probability Discounting Questionnaire (see Madden et al., 2009; <[doi:10.1037/a0016806](https://doi.org/10.1037/a0016806)>), calculating k values (Mazur's simple hyperbolic and exponential) using nonlinear regression, calculating various Area Under the Curve (AUC) measures, plotting regression curves for both fit-to-group and two-stage approaches, checking for unsystematic discounting (Johnson & Bickel, 2008; <[doi:10.1037/1064-1297.16.3.264](https://doi.org/10.1037/1064-1297.16.3.264)>) and scoring of the minute discounting task (see Koffarnus & Bickel, 2014; <[doi:10.1037/a0035973](https://doi.org/10.1037/a0035973)>) using the Qualtrics 5-trial discounting template (see the Qualtrics Minute Discounting User Guide; <[doi:10.13140/RG.2.2.26495.79527](https://doi.org/10.13140/RG.2.2.26495.79527)>), which is also available as a .qsf file in this package. The package also fits mixed-effects (hierarchical) discounting models with a scale-location-truncated beta likelihood that admits indifference points at exactly 0 and 1 (Kim, Kaplan, Koffarnus, and Franck, 2025; <[doi:10.48550/arXiv.2509.13167](https://doi.org/10.48550/arXiv.2509.13167)>). The package also implements the linearized Mazur estimator, random-effects model, and F-test of Hinds et al. (2026; <[doi:10.1016/j.jmp.2026.103006](https://doi.org/10.1016/j.jmp.2026.103006)>).

License GPL (>= 2)

URL <https://brentkaplan.github.io/beezeconomic/>,
<https://github.com/brentkaplan/beezeconomic>

BugReports <https://github.com/brentkaplan/beezeconomic/issues>

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.2.0)

Imports beezdemand, broom, cli, dplyr, generics, ggplot2, gtools,
magrittr, minpack.lm, nlme, psych, purrr, stringr, tibble,
tidyr, TMB (>= 1.9.0)

Suggests brms (>= 2.20.0), knitr, loo, posterior, pwr, rmarkdown,
testthat (>= 3.0.0)

LinkingTo RcppEigen, TMB

VignetteBuilder knitr

SystemRequirements GNU make

Config/testthat/edition 3

LazyData true

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Repository <https://brentkaplan.r-universe.dev>

Date/Publication 2026-09-17 21:39:54 UTC

RemoteUrl <https://github.com/brentkaplan/beeze discounting>

RemoteRef develop

RemoteSha d76271a7c6e17d2ba8badf4e2352ebf85f0e868a

Contents

anova.beeze discounting_linear	4
ans_dd	5
ans_pd	6
augment.beeze discounting_choice	6
augment.beeze discounting_tmb	7
beeze discounting_linear-methods	8
calc_aucs	10
calc_conf_int	11
calc_dd	11
calc_pd	12
calc_r2	13
check_unsystematic	13
coef.beeze discounting_choice	14
coef.beeze discounting_tmb	15
confint.beeze discounting_brms	16
confint.beeze discounting_choice	16
confint.beeze discounting_choice_brms	17
confint.beeze discounting_tmb	18
dd_ip	19
default_dd_choice_priors	20
default_dd_priors	21

find_n_discounting	22
fit_dd	25
fit_dd_brms	26
fit_dd_choice	28
fit_dd_choice_brms	30
fit_dd_linear	32
fit_dd_tmb	33
fitted.beezdiscounting_choice	36
fitted.beezdiscounting_tmb	36
five.fivetrial_dd	37
five.fivetrial_pd	37
fixef.beezdiscounting_choice	38
fixef.beezdiscounting_tmb	38
generate_data_mcq	39
get_dd_comparisons	39
get_dd_param_emms	41
get_lookup_table	42
glance.beezdiscounting_brms	43
glance.beezdiscounting_choice	43
glance.beezdiscounting_choice_brms	44
glance.beezdiscounting_tmb	44
long_to_wide_mcq	45
long_to_wide_mcq_excel	45
mcq_to_choice	46
mcq21	47
mcq27	48
mcq27_to_choice	48
pdq	49
pdq_to_choice	50
plot.beezdiscounting_brms	51
plot.beezdiscounting_choice	52
plot.beezdiscounting_choice_brms	53
plot.beezdiscounting_comparison	54
plot.beezdiscounting_linear	54
plot.beezdiscounting_tmb	56
plot.prop_sc_output	57
plot.prop_ss_output	58
plot.score_mcq_output	59
plot.score_mcq27_output	60
plot.score_pdq_output	61
plot_dd	61
plot_qq.beezdiscounting	62
power_discounting	63
predict.beezdiscounting_brms	66
predict.beezdiscounting_choice	67
predict.beezdiscounting_choice_brms	68
predict.beezdiscounting_tmb	69
print.beezdiscounting_choice	70

print.beezdiscounting_tmb	71
print.summary.beezdiscounting_choice	71
print.summary.beezdiscounting_tmb	72
prop_sc	72
prop_ss	73
ranef.beezdiscounting_choice	74
ranef.beezdiscounting_tmb	74
residuals.beezdiscounting_choice	75
residuals.beezdiscounting_tmb	76
results_dd	77
score_dd	78
score_mcq	78
score_mcq27	79
score_pd	81
score_pdq	81
simulate_dd_choice	83
simulate_dd_ip	84
simulate_dd_linear	87
summarize_mcq	88
summary.beezdiscounting_brms	88
summary.beezdiscounting_choice	89
summary.beezdiscounting_choice_brms	89
summary.beezdiscounting_tmb	90
tidy.beezdiscounting_brms	91
tidy.beezdiscounting_choice	91
tidy.beezdiscounting_choice_brms	92
tidy.beezdiscounting_comparison	93
tidy.beezdiscounting_tmb	94
timing_dd	95
timing_pd	95
VarCorr.beezdiscounting_choice	96
VarCorr.beezdiscounting_tmb	97
wide_to_long_mcq	97
wide_to_long_mcq_excel	98

Index 99

anova.beezdiscounting_linear

F-test for condition means in a linearized Mazur fit (Hinds et al. 2026, Prop. 3.5)

Description

The paper's F-test is exact when the random-effects variance estimate $\hat{g} > 0$ (Prop. 3.5's assumption); when $\hat{g} = 0$ the statistic is still reported, with a warning.

Usage

```
## S3 method for class 'beezdiscounting_linear'
anova(object, hypothesis = NULL, pairwise = FALSE, ...)
```

Arguments

object	A beezdiscounting_linear fit with a factor.
hypothesis	NULL (all condition means equal) or a list of character vectors, each naming levels constrained equal under H0.
pairwise	If TRUE, test every pair of levels (uncorrected p-values); cannot be combined with hypothesis.
...	Unused; present for S3 generic consistency.

Value

Tibble of class beezdiscounting_linear_anova.

See Also

[beezdiscounting_linear-methods](#) for the other S3 methods.

ans_dd	<i>Converts answers from 5.5 trial delay discounting from Qualtrics template</i>
--------	--

Description

Converts answers from 5.5 trial delay discounting from Qualtrics template

Usage

```
ans_dd(df)
```

Arguments

df	A dataframe containing all the columns
----	--

Value

A dataframe with the ResponseId, index, and response (ss or ll).

Examples

```
ans_dd(five.fivetrial_dd)
```

ans_pd	<i>Converts answers from 5.5 trial probability discounting from Qualtrics template</i>
--------	--

Description

Converts answers from 5.5 trial probability discounting from Qualtrics template

Usage

```
ans_pd(df)
```

Arguments

df	A dataframe containing all the columns
----	--

Value

A dataframe with the ResponseId, index, and response (sc or lu).

Examples

```
ans_pd(five.fivetrial_pd)
```

augment.beezdiscounting_choice	<i>Augment a beezdiscounting_choice model</i>
--------------------------------	---

Description

Returns the data used for fitting (or newdata) as a tibble with three diagnostic columns appended:

- `.fitted`: subject-conditional fitted choice probability $P(LL)$.
- `.resid`: raw (response) residual choice - `.fitted`.
- `.std_resid`: Pearson residual `.resid / sqrt(p * (1 - p))`.

Usage

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

Arguments

x	A beezdiscounting_choice object.
newdata	Optional data frame. NULL uses the fitting data.
...	Unused.

Value

A tibble with the same rows as the data plus `.fitted`, `.resid`, and `.std_resid`.

```
augment.beezdiscounting_tmb
```

Augment a beezdiscounting_tmb model

Description

Returns the data used for fitting (or newdata) as a tibble with three diagnostic columns appended:

- `.fitted`: subject-conditional fitted indifference proportion (clamped to $(0, 1)$).
- `.resid`: raw residual $y - \text{.fitted}$ on the response scale.
- `.std_resid`: Pearson (standardized) residual, i.e., `.resid` divided by the per-row response SD. For family = "gaussian" the SD is the constant `sigma_e`; for family = "sltb" it is the delta-method SLT SD $\sqrt{\mu * (1 - \mu) / (\phi + 1)}$.

Usage

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

Arguments

<code>x</code>	A <code>beezdiscounting_tmb</code> object.
<code>newdata</code>	Optional data frame. NULL uses the fitting data. A supplied <code>newdata</code> must use the package's canonical column names (<code>id</code> , <code>x</code> , <code>y</code>), regardless of the <code>id_var</code> / <code>x_var</code> / <code>y_var</code> you gave at fit time.
<code>...</code>	Unused.

Value

A tibble with the same rows as the data plus `.fitted`, `.resid`, and `.std_resid`.

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, verbose = 0)
head(augment(fit))
```

beezdiscounting_linear-methods

Methods for linearized Mazur fits

Description

S3 methods for objects of class `beezdiscounting_linear` returned by `fit_dd_linear()`. All quantities are closed form; nothing is re-optimized.

Usage

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

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

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

## S3 method for class 'beezdiscounting_linear'
tidy(x, effects = c("subject", "population"), ...)

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

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

## S3 method for class 'beezdiscounting_linear'
confint(object, parm = c("population", "subject"), level = NULL, ...)

## S3 method for class 'beezdiscounting_linear'
augment(x, ...)

## S3 method for class 'beezdiscounting_linear'
predict(object, type = "parameters", ...)

## S3 method for class 'beezdiscounting_linear'
nobs(object, ...)

## S3 method for class 'beezdiscounting_linear'
logLik(
  object,
  level = c("population", "subject"),
  scale = c("raw", "transformed"),
  ...)
```

)

Arguments

...	Unused; present for S3 generic consistency.
object, x	A beezdiscounting_linear fit (for print.summary(), the object returned by summary()).
effects	For tidy(): "subject" (default) returns the per-unit table; "population" returns the random-effects MLEs $\mu_{<level>}$, σ^2 and g .
parm	Which interval confint() returns: "population" (default) is an ANOVA-style t-interval on each condition mean, using the between-unit mean square with $N - C$ degrees of freedom; "subject" gives the per-subject t-intervals on $\ln k$. It does not select parameter names.
level	For confint(), the confidence level. NULL (default) means 0.95 for parm = "population", as in the package's other confint() methods, and the fit's conf_level for parm = "subject", so that call reproduces the intervals in tidy() / fit\$subjects. For logLik(), which likelihood to return: "population" (the random-effects MLE, $df = C + 2$) or "subject" (the sum of the per-unit log-likelihoods, $df = 2$ per unit).
type	For predict(): only "parameters" exists, returning the per-subject $\ln k$ and k with their t-intervals (id, condition, logk, logk_lower, logk_upper, k, k_lower, k_upper). There is no newdata prediction; augment() gives the fitted curve.
scale	For logLik(): "raw" (default) is Jacobian-corrected back to the indifference-point scale; "transformed" is on the linearized scale.

Details

confint() uses parm to SELECT WHICH INTERVAL is returned, not to filter parameter names as in [stats::confint\(\)](#). logLik() reports the raw-scale (Jacobian-corrected) log-likelihood by default, which makes it comparable with this package's Gaussian NLS and TMB log-likelihoods.

Value

print() and print.summary() return their input invisibly; summary() returns an object of class summary.beezdiscounting_linear; tidy(), glance(), augment() and predict() return tibbles. augment() adds .fitted and .resid on the raw indifference-point scale and .std_resid, the transformed-scale residual $y_{lin} - \ln k_i$ divided by the model's error standard deviation ($\sqrt{\sigma^2}$) from the random-effects fit, or the pooled within-subject mean square when that component is unavailable; .std_resid is NA for points dropped under boundary = "drop". coef() returns the named vector of condition means μ ; confint() returns a two-column matrix of lower/upper bounds; nobs() returns the number of usable transformed observations; logLik() returns a "logLik" object with df and nobs attributes. Methods needing the random-effects component error when the design was unbalanced and re is NULL.

See Also

[fit_dd_linear\(\)](#), [anova.beezdiscounting_linear\(\)](#), [plot.beezdiscounting_linear\(\)](#)

calc_aucs	<i>Calculate Area-Under-the-Curve (AUC) Metrics for Delay Discounting Data</i>
-----------	--

Description

Calculates three area-under-the-curve (AUC) metrics for delay-discounting data: regular AUC (using raw delays), log10 AUC (using logarithmically scaled delays), and ordinal AUC (using ordinally scaled delays). The three differ only in how the delays are scaled before the area is computed. Metrics are computed separately for each id, so a data frame with several subjects returns one row per subject.

Usage

```
calc_aucs(dat)
```

Arguments

dat	A data frame containing delay discounting data. It must include the following columns: • id: Participant or group identifier. • x: Delay values (e.g., in days). • y: Indifference point values (e.g., subjective value of the delayed reward).
-----	---

Value

A tibble with the following columns:

- id: The participant or group identifier.
- auc_regular: The regular AUC, calculated using the raw delay values.
- auc_log10: The log10 AUC, calculated using logarithmically transformed delay values.
- auc_ord: The ordinal AUC, calculated using ordinally scaled delay values.

Examples

```
# Example data
data <- data.frame(
  id = rep("P1", 6),
  x = c(1, 7, 30, 90, 180, 365),
  y = c(0.8, 0.5, 0.3, 0.2, 0.1, 0.05)
)

# Calculate AUC metrics for a single participant
calc_aucs(data)
```

calc_conf_int	<i>Calculate Confidence Intervals for a Parameter</i>
---------------	---

Description

Computes the lower and upper bounds of a t-based confidence interval for a parameter estimate from its standard error, a specified significance level, and the degrees of freedom from the model.

Usage

```
calc_conf_int(estimate, std_error, model, alpha = 0.05)
```

Arguments

estimate	A numeric value representing the parameter estimate.
std_error	A numeric value representing the standard error of the parameter estimate.
model	A fitted model object that provides the residual degrees of freedom via <code>df.residual()</code> .
alpha	A numeric value representing the significance level. Default is 0.05 (95% confidence interval).

Value

A numeric vector of length two:

- First element: Lower bound of the confidence interval.
- Second element: Upper bound of the confidence interval.

Examples

```
# Example using a linear model
data <- data.frame(x = 1:10, y = c(2.3, 2.1, 3.7, 4.5, 5.1, 6.8, 7.3, 7.9, 9.2, 10.1))
lm_model <- lm(y ~ x, data = data)
calc_conf_int(estimate = 0.5, std_error = 0.1, model = lm_model, alpha = 0.05)
```

calc_dd	<i>Calculate scores, answers, and timing for 5.5 trial delay discounting from Qualtrics template</i>
---------	--

Description

Calculate scores, answers, and timing for 5.5 trial delay discounting from Qualtrics template

Usage

```
calc_dd(df)
```

Arguments

df A dataframe containing all the columns from the template.

Value

A dataframe with k/ed50 values, answers, timing

Examples

```
calc_dd(five.fivetrial_dd)
```

calc_pd	<i>Calculate scores, answers, and timing for 5.5 trial probability discounting from Qualtrics template</i>
---------	--

Description

Calculate scores, answers, and timing for 5.5 trial probability discounting from Qualtrics template

Usage

```
calc_pd(df)
```

Arguments

df A dataframe containing all the columns from the template.

Value

A dataframe with h/ep50 values, answers, timing

Examples

```
calc_pd(five.fivetrial_pd)
```

calc_r2*Calculate R-Squared for a Model*

Description

Calculates the coefficient of determination (R^2) for a fitted model as one minus the ratio of the sum of squared errors (SSE) to the total sum of squares (SST).

Usage

```
calc_r2(model)
```

Arguments

model A fitted model object. The model must have `resid()` and `fitted()` methods to extract residuals and fitted values.

Value

A numeric value representing the R^2 value of the model. Returns NA if the model is NULL.

Examples

```
# Example using a simple linear model
data <- data.frame(x = 1:10, y = c(1, 2, 3, 4, 5, 6, 7, 9, 10, 11))
lm_model <- lm(y ~ x, data = data)
calc_r2(lm_model)
```

check_unsystematic*Check for Unsystematic Data Violations*

Description

Checks a data set against the two Johnson and Bickel (2008) criteria for unsystematic delay-discounting data:

- Criterion 1: Any subsequent value of y exceeds the previous value by more than a specified proportion of the larger later reward (`ll`).
- Criterion 2: The last value of y is not at least a specified proportion less than the first value of y .

Usage

```
check_unsystematic(dat, ll = 1, c1 = 0.2, c2 = 0.1)
```

Arguments

dat	A data frame containing the delay-discounting data. It must have at least two columns: • id: A unique identifier for the data set. • y: The indifference points to be analyzed. • x (optional): Delay values; when present, each subject's points are ordered by delay first.
l1	A numeric value representing the larger later reward. Default is 1.
c1	A numeric value for the threshold proportion for Criterion 1. Default is 0.2.
c2	A numeric value for the threshold proportion for Criterion 2. Default is 0.1.

Details

The criteria are applied separately to each id, so a data frame containing several subjects returns one row per subject. When an x (delay) column is present, each subject's points are ordered by delay before the criteria are applied.

Value

A tibble with the following columns:

- id: The unique identifier for the data set.
- c1_pass: Logical value indicating whether Criterion 1 was passed.
- c2_pass: Logical value indicating whether Criterion 2 was passed.

Examples

```
data <- tibble::tibble(
  id = c(rep("P1", 6)),
  x = c(1, 7, 30, 90, 180, 365), # delays
  y = c(0.9, 0.5, 0.3, 0.2, 0.1, 0.05) # indifference points
)
check_unsystematic(data, l1 = 1, c1 = 0.2, c2 = 0.1)
```

coef.beezdiscounting_choice

Extract coefficients from a structural choice model

Description

Returns the optimizer's flat named parameter vector: beta_k (one entry per fixed-effect design column, on the log-k scale), log_sigma_u, log_gamma, and beta0 (only when intercept = TRUE). This is the numeric escape hatch consumed by tooling such as car::deltaMethod.

Usage

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

Arguments

object	A beeزdiscounting_choice object.
...	Unused.

Value

Named numeric vector.

```
coef.beeزdiscounting_tmb
```

Extract coefficients from a TMB discounting model

Description

Returns the optimizer's flat named parameter vector: beta_k (one entry per fixed-effect design column, on the log-k scale), log_sigma_u, and the auxiliary parameter (log_phi for family = "sltb", log_sigma_e for family = "gaussian"). This is the numeric escape hatch consumed by tooling such as car::deltaMethod.

Usage

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

Arguments

object	A beeزdiscounting_tmb object.
...	Unused.

Value

Named numeric vector.

```
confint.beezdiscounting_brms
```

Credible intervals for a beezdiscounting_brms model

Description

Equal-tailed quantile credible intervals on the report-space-transformed draws. Columns match `confint.beezdiscounting_tmb()`: `term`, `estimate`, `conf.low`, `conf.high`, `level`.

Usage

```
## S3 method for class 'beezdiscounting_brms'
confint(
  object,
  parm = NULL,
  level = 0.95,
  report_space = c("natural", "log10", "internal", "log"),
  ...
)
```

Arguments

<code>object</code>	A <code>beezdiscounting_brms</code> object.
<code>parm</code>	Optional terms (display, e.g. "k: (Intercept)"/"s", or TMB coefficient names "beta_k"/"log_s").
<code>level</code>	Credible level (default 0.95).
<code>report_space</code>	Reporting scale.
<code>...</code>	Unused.

Value

A tibble.

```
confint.beezdiscounting_choice
```

Confidence intervals for a structural choice discounting model

Description

Wald (Hessian-based) confidence intervals: `estimate +/- z * se` on the internal scale, then back-transformed to `report_space`.

Usage

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

Arguments

object	A beeزdiscounting_choice object.
parm	Optional character vector for filtering. Accepts display names ("k: (Intercept)", "gamma") or raw optimizer names ("beta_k", "log_gamma", "log_sigma_u", "beta0"). NULL returns all coefficients.
level	Confidence level (default 0.95).
report_space	"internal" (default; all coefficients on their estimation scale) or "natural" (exponentiate the beta_k rows so the intercept is k at the reference level, and exponentiate log_gamma so the row is gamma; log_sigma_u and beta0 stay on their internal scales).
...	Unused.

Value

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

confint.beeزdiscounting_choice_brms

Credible intervals for a beeزdiscounting_choice_brms model

Description

Quantile credible intervals on the report-space-transformed draws; columns term, estimate, conf.low, conf.high, level. beta0 stays on the identity scale across report spaces, matching tidy.beeزdiscounting_choice().

Usage

```
## S3 method for class 'beeزdiscounting_choice_brms'
confint(
  object,
  parm = NULL,
  level = 0.95,
  report_space = c("natural", "log10", "internal", "log"),
  ...
)
```

Arguments

object	A beezdiscounting_choice_brms object.
parm	Optional terms: display names ("k:(Intercept)", "k:<level>", "gamma", "beta0") or TMB coefficient names ("beta_k", which selects every log-k design coefficient, or "log_gamma").
level	Credible level (default 0.95).
report_space	Reporting scale.
...	Unused.

Value

A tibble.

confint.beezdiscounting_tmb

Confidence intervals for a TMB discounting model

Description

Wald (Hessian-based) confidence intervals: estimate $\pm z \times se$ on the internal (log-k) scale, then back-transformed to report_space.

Usage

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

Arguments

object	A beezdiscounting_tmb object.
parm	Optional character vector for filtering. Accepts display names ("k:(Intercept)", and "s" for the GM/Rachlin shape parameter) or raw optimizer names ("beta_k", "log_sigma_u", "log_phi", "log_sigma_e", "log_s", and for a 2-RE fit "log_sd_re" / "cor_re"). NULL returns all coefficients.
level	Confidence level (default 0.95).
report_space	"internal" (default; all coefficients on their estimation/log scale) or "natural" (exponentiate beta_k rows so the intercept is k at the reference level and non-intercept terms are multiplicative fold-changes; variance/aux params stay on their internal scale).
...	Unused.

Value

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

Note

For a 2-RE fit the `cor_re` and `log_sd_re` rows are reported on their internal (atanh / $\log$) scales and are NOT back-transformed by `report_space = "natural"` (only `beta_k/log_s` rows are); use `VarCorr()` to obtain the correlation and natural-log SDs.

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, equation = "mazur", family = "sltb",
  random_effects = k ~ 1, verbose = 0)
confint(fit)
confint(fit, parm = "k:(Intercept)", report_space = "natural")
```

dd_ip	<i>Delay Discounting Data</i>
-------	-------------------------------

Description

A dataset containing a set of fake delay discounting responses

Usage

```
dd_ip
```

Format

A data frame with 600 rows and 3 columns:

- id** Subject identifier.
- x** Delay until the larger, later reward (in days).
- y** Indifference point, expressed as a proportion of the larger, later reward (these synthetic values are near, but not strictly within, the unit interval).

default_dd_choice_priors

Default priors for the Bayesian (brms) choice model

Description

$\log\gamma \sim \text{normal}(1, 1)$ (choice sensitivity, typically 1-20) and the optional logit-scale bias $b_0 \sim \text{normal}(0, 1.5)$; $\log k$ defaults as in [default_dd_priors\(\)](#) (autoscaled to the median delay when data is supplied; `delay_var` names the delay column).

Usage

```
default_dd_choice_priors(
  equation = c("mazur", "exponential"),
  intercept = FALSE,
  data = NULL,
  delay_var = "delay",
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  autoscale = !is.null(data)
)
```

Arguments

<code>equation</code>	"mazur" or "exponential".
<code>intercept</code>	Include the b_0 prior.
<code>data</code>	Optional data frame used for $\log k$ autoscaling.
<code>delay_var</code>	Delay column name in data.
<code>factors, factor_interaction, continuous_covariates</code>	Fixed-effect design on $\log k$, as passed to fit_dd_choice_brms() . When the design has non-intercept coefficients (derived through <code>build_fixed_rhs()</code>), a fold-change normal(0, 1) class-level coefficient prior is added; with an intercept-only design it is omitted (it would be unused, and brms warns).
<code>autoscale</code>	Logical; defaults to TRUE when data is supplied.

Value

A brmsprior data frame.

default_dd_priors	<i>Default priors for Bayesian (brms) delay-discounting models</i>
-------------------	--

Description

Returns the default prior table used by `fit_dd_brms()`, so the defaults can be inspected, modified row-wise, and passed back via the fitter's prior argument. The logk location is the principled fix for k's delay-unit dependence: with `autoscale = TRUE` (the default whenever data is supplied) it centers $k * \text{median}(\text{delay}) = 1$ (the delay at which the Mazur curve crosses 0.5) via `normal(-log(median(x)), 2.5)`; the static fallback is `normal(-4.5, 2.5)`. The anchors used are attached as `attr("autoscale_info")`; numeric values are formatted with `format(x, digits = 6, scientific = FALSE)`.

Usage

```
default_dd_priors(
  equation = c("mazur", "exponential", "green-myerson", "rachlin"),
  family = c("beta", "gaussian"),
  data = NULL,
  y_var = "y",
  x_var = "x",
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  autoscale = !is.null(data),
  random_effects = k ~ 1,
  covariance_structure = c("pdSymm", "pdDiag")
)
```

Arguments

<code>equation</code>	Discounting equation (TMB-tier vocabulary).
<code>family</code>	"beta" or "gaussian".
<code>data</code>	Optional data frame used for autoscaling.
<code>y_var, x_var</code>	Column names in data (canonical defaults).
<code>factors, factor_interaction, continuous_covariates</code>	Fixed-effect design on logk, as passed to the fitter. When the design has non-intercept coefficients (derived through <code>build_fixed_rhs()</code> , so single-level dropped factors do not count), a fold-change <code>normal(0, 1)</code> class-level coefficient prior is added; with an intercept-only design it is omitted (it would be unused, and brms warns).
<code>autoscale</code>	Logical; defaults to TRUE when data is supplied.
<code>random_effects</code>	$k \sim 1$ (default) or $k + \phi \sim 1$. The latter re-keys the beta precision: ϕ becomes a predicted distributional parameter, so the scalar <code>gamma(2, 0.1)</code> is replaced by a log-scale intercept prior, a half-t precision-RE SD, and (for "pdSymm") an LKJ correlation prior.

covariance_structure
 (log k, log phi) covariance for k + phi ~ 1: "pdSymm" (default, correlated)
 or "pdDiag" (independent).

Details

Other defaults: $\log s \sim \text{normal}(0, 0.5)$ (two-parameter equations; s is near 1 a priori), $\text{sd}(\log k) \sim \text{student_t}(3, 0, 1)$, $\phi \sim \text{gamma}(2, 0.1)$ (Beta precision; mean 20, far from brms's near-improper $\text{gamma}(0.01, 0.01)$), and $\sigma \sim \text{student_t}(3, 0, 0.25)$ for the Gaussian family (y is a proportion in the unit interval, so $\text{sd}(y) \leq 0.5$).

Value

A brmsprior data frame, with `attr(, "autoscale_info")` when autoscaling was used.

find_n_discounting	<i>Find the smallest sample size reaching a target power (discounting)</i>
--------------------	--

Description

Bisection search over `n_subjects` for the smallest total N whose Monte Carlo power estimate from `power_discounting()` reaches `target_power`. The search accounts for Monte Carlo noise: at each evaluated N , replicates are added in batches (up to `n_sim_max`) until the Wilson interval for power lies wholly above or below the target; if it still straddles the target at the cap, the decision falls back to the point estimate and the result is flagged uncertain. The selected N and its lower neighbor are then re-evaluated with fresh replicates before minimality is claimed.

The returned n is an *estimated minimum under Monte Carlo uncertainty* rather than an exact bound. For grant-quality reporting, rerun `power_discounting()` at the returned n with a large `n_sim` (2000+) and report that estimate with its Monte Carlo confidence interval.

Monotonicity assumption. Bisection presumes that power is non-decreasing in `n_subjects`. Because every evaluated N is judged from its own independent replicates (and a convergence-conditioned denominator), a Monte Carlo fluctuation can make a lower N read "below" when its true power is above the target, so the search may step past a lower crossing that it never revisits. Evaluated N that contradict the assumption (a lower N reading "above" the selected N) demote the status to "uncertain"; N that were never evaluated cannot be checked. Widen `n_sim/n_sim_max` when the reported n matters. When the target is already met at `n_range[1]`, that bound is likewise re-evaluated with fresh replicates before "at_lower_bound" is reported; if the second look does not clear the target the bound is treated as below and the bisection proceeds upward.

Usage

```
find_n_discounting(
  target_power = 0.8,
  effect = list(delta_k = NULL),
  design = list(),
  n_range = c(6, 200),
  n_sim = 200,
```

```

    n_sim_max = 4 * n_sim,
    alpha = 0.05,
    df = NULL,
    seed = NULL,
    equation = c("mazur", "exponential"),
    family = c("sltb", "gaussian"),
    random_effects = k ~ 1,
    multi_start = FALSE,
    verbose = TRUE,
    ...
)

```

Arguments

target_power	Target power in (0, 1).
effect	Named list supplying delta_k: the true condition-2 shift on natural-log k. 0 is allowed (useful for Type I error checks). E.g. delta_k = log(2) means condition 2's k is twice condition 1's.
design	Named list of data-generating settings, merged over the simulator defaults: delays (vector), log_k_pop, sigma_u (subject SD on log k), phi (SLT-beta precision; used when family = "sltb"), and sigma_e (residual SD; used when family = "gaussian").
n_range	Integer bracket c(lower, upper) of total N to search ($2 \leq \text{lower} < \text{upper}$). The search errors (rather than extrapolating) if the target is not reached at upper.
n_sim	Replicates per evaluation batch. Smaller than the power_discounting() default because several N values are evaluated; the adaptive rule adds batches where the verdict is close.
n_sim_max	Maximum replicates per evaluated N (default $4 * n_sim$).
alpha	Nominal two-sided test level.
df	Degrees of freedom for the Wald test's t reference. NULL (default) tracks the evaluated sample size as $n - 2$; a numeric value (or Inf for the asymptotic z-test) is used at every evaluated N.
seed	Optional integer seed; identical seeds give identical results. The caller's RNG state is restored on exit.
equation	Discounting function used for BOTH simulation and refitting (they are always matched): "mazur" (default) or "exponential". The two-parameter Green-Myerson and Rachlin forms are out of scope in this version.
family	Observation family used for both simulation and refitting: "sltb" (default) or "gaussian".
random_effects	Random-effects formula passed to fit_dd_tmb() . The default $k \sim 1$ matches the simulator's data-generating process (a single subject random intercept on log k). The simulator ALWAYS generates only that intercept: any richer formula (e.g. $k + \phi \sim 1$, $k + s \sim 1$) is accepted but produces a deliberately over-specified refit of data with no such variance component. That refit is useful for probing robustness rather than for estimating power under those random effects (out of scope in this version).

multi_start Passed to `fit_dd_tmb()`. Defaults to FALSE for speed; non-convergent replicates are excluded and surfaced rather than biasing the estimate.

verbose Logical; report each evaluation.

... Additional arguments passed to `fit_dd_tmb()`.

Value

An object of class `beezdiscounting_power_n`: a list with

n Estimated smallest total `n_subjects` reaching `target_power`; NA when the confirmation pass contradicted the search (`status = "unresolved"`).

target_power As supplied.

status "confirmed" (selected N re-confirmed above target and N - 1 below), "uncertain" (a decision relied on a point estimate, N - 1 also cleared the target on reconfirmation, or an evaluated lower N read above the target, so the returned N may not be minimal), "unresolved" (the selected N failed its own reconfirmation; n is NA), or "at_lower_bound" (the target was already met at `n_range[1]` on two independent looks; smaller N was not explored; widen `n_range` downward if that matters). These labels describe a heuristic Monte Carlo decision rule (repeated looks at ordinary Wilson intervals across several N) rather than a formal sequential error guarantee.

uncertain Logical; TRUE when any search decision was made on a point estimate rather than a conclusive Wilson interval, or the status is not "confirmed"/"at_lower_bound".

evaluations Tibble of every evaluation: `n_subjects`, `n_sim_total`, `n_used`, `usable_fraction`, `power`, `ci_lower`, `ci_upper`, `decision`. A warning fires when any evaluation had fewer than 95% usable fits.

alpha, **df**, **effect**, **design**, **n_range**, **n_sim**, **n_sim_max**, **seed**, **settings**, **call** Echoed inputs and effective settings.

See Also

`power_discounting()` for the Monte Carlo engine.

Other power-analysis: `power_discounting()`

Examples

```
# Small search for demonstration (use larger n_sim for real planning)
res <- find_n_discounting(
  target_power = 0.8,
  effect = list(delta_k = log(8)),
  n_range = c(6, 20), n_sim = 4, n_sim_max = 4, seed = 1, verbose = FALSE
)
print(res)
```

fit_dd

*Fit Delay-Discounting Model***Description**

Fits a delay-discounting model to a data set with the specified equation and fitting method.

Usage

```
fit_dd(dat, equation, method)
```

Arguments

- | | |
|----------|---|
| dat | A data frame containing delay (x) and indifference point (y) data. For "two stage" methods, the data must include an id column to identify participants. |
| equation | A character string specifying the delay-discounting equation to use. Options include: <ul style="list-style-type: none"> • "mazur" or "hyperbolic": Hyperbolic delay-discounting model ($y = 1/(1 + k \cdot x)$). • "exponential": Exponential delay-discounting model ($y = \exp(-k \cdot x)$). |
| method | A character string specifying the method for fitting the model. Options include: <ul style="list-style-type: none"> • "pooled" or "agg": Fits the model using pooled data. • "mean": Fits the model using the mean of indifference points at each delay. • "ts" or "two stage": Fits the model separately for each participant (requires an id column in dat). |

Value

A list object of class "fit_dd", containing:

- The fitted model(s).
- The original dataset (dat).
- The specified method (method).

Examples

```
data <- data.frame(
  id = rep(1:2, each = 6),
  x = rep(c(1, 7, 30, 90, 180, 365), 2),
  y = c(0.9, 0.5, 0.3, 0.2, 0.1, 0.05, 0.85, 0.55, 0.35, 0.15, 0.1, 0.05)
)
fit_dd(data, equation = "mazur", method = "two stage")
```

fit_dd_brms

*Fit a Bayesian Mixed-Effects Discounting Model via brms***Description**

Fits the four discounting equations of `fit_dd_tmb()` ("mazur", "exponential", "green-myerson", "rachlin") as Bayesian nonlinear mixed-effects models, with the discount rate estimated on the natural-log scale (logk, subject random intercept; $k \sim 1$ by default, or $k + \phi \sim 1$ to add a per-subject precision random effect) and the shape exponent (logs) population-level for the two-parameter equations.

Usage

```
fit_dd_brms(
  data,
  y_var = "y",
  x_var = "x",
  id_var = "id",
  equation = c("mazur", "exponential", "green-myerson", "rachlin"),
  family = c("beta", "gaussian"),
  boundary = c("squeeze", "zoib", "error"),
  random_effects = k ~ 1,
  covariance_structure = c("pdSymm", "pdDiag"),
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  ll = NULL,
  response_scale = c("proportion", "percent", "amount"),
  prior = NULL,
  autoscale_priors = TRUE,
  chains = 4,
  iter = 2000,
  warmup = floor(iter/2),
  thin = 1,
  cores = getOption("mc.cores", 1L),
  seed = NA,
  backend = getOption("brms.backend", "rstan"),
  control = list(adapt_delta = 0.95),
  init = c("prior_center", "tmb", "random", "0"),
  sample_prior = "no",
  loo = TRUE,
  file = NULL,
  file_refit = getOption("brms.file_refit", "on_change"),
  verbose = 1,
  ...
)
```

Arguments

<code>data</code>	Long-format data frame (one row per subject-delay).
<code>y_var, x_var, id_var</code>	Column names (canonical defaults), as in <code>fit_dd_tmb()</code> .
<code>equation</code>	Discounting equation.
<code>family</code>	"beta" or "gaussian"; "sltb" errors with guidance.
<code>boundary</code>	Boundary handling for the beta family (see Details).
<code>random_effects</code>	$k \sim 1$ (single random intercept on log k , the default) or $k + \phi \sim 1$ (adds a per-subject precision random effect; beta family only). The latter mirrors <code>fit_dd_tmb(random_effects = k + phi ~ 1)</code> : ϕ becomes a predicted distributional parameter on the log link carrying a subject random intercept.
<code>covariance_structure</code>	For $k + \phi \sim 1$, the (log k , log ϕ) covariance: "pdSymm" (default; correlated, the TMB parity choice) or "pdDiag" (independent). Ignored for $k \sim 1$.
<code>factors, factor_interaction, continuous_covariates</code>	Fixed-effect design on log k , as in <code>fit_dd_tmb()</code> .
<code>ll, response_scale</code>	Response coercion, as in <code>fit_dd_tmb()</code> .
<code>prior</code>	Optional brmsprior; user rows override the defaults.
<code>autoscale_priors</code>	Anchor the log k prior to the median delay (see <code>default_dd_priors()</code>).
<code>chains, iter, warmup, thin, cores, seed, backend, control, sample_prior</code>	MCMC settings passed to <code>brms::brm()</code> .
<code>init</code>	"prior_center" (default), "tmb" (a quiet <code>fit_dd_tmb()</code> pre-fit supplies the centers, with prior_center fallback on failure; the beta family maps to the TMB sltb pre-fit), "random", or " $\emptyset$ "; or a list/function passed through to <code>brms::brm()</code> .
<code>loo</code>	Compute and store <code>brms::loo()</code> at fit time.
<code>file, file_refit</code>	Passed to <code>brms::brm()</code> for fit caching.
<code>verbose</code>	0 (silent), 1 (messages), 2 (full Stan output).
<code>...</code>	Passed through to <code>brms::brm()</code> .

Details

`family = "beta"` (default) uses `Beta(link = "identity")` with the mean squished into $(1e-6, 1 - 1e-6)$, the closest brms analog of the TMB SLT-beta (`family = "sltb"` has no brms equivalent and errors with this pointer). Boundary observations (y exactly 0 or 1) are handled per boundary: "squeeze" (default) applies the Smithson-Verkuilen transform $y^* = (y (N - 1) + 0.5) / N$ to all responses (message reports the boundary count); "zoib" switches to `zero_one_inflated_beta` (statistically more honest, but it changes the estimand: k then describes interior responses only); "error" refuses to fit. `family = "gaussian"` matches `fit_dd_tmb(family = "gaussian")` wherever the TMB template's μ clamp into $[1e-6, 1 - 1e-6]$ does not bind (everywhere except extreme decay underflow).

Value

An object of class `beezdiscounting_brms: model` (posterior medians/SDs on the estimation scale under TMB names `beta_k/log_s`, plus `variance_components`), `brmsfit`, `subject_pars` (`id`, `k`, `k_lower`, `k_upper`, plus `phi/phi_lower/phi_upper` for $k + \phi \sim 1$), `converged` ($R_{\text{hat}} < 1.01$, no divergences, bulk ESS ≥ 400), `mcmc_info`, `loo`, `data`, `param_info`, `priors`, `autoscale_info`.

See Also

`fit_dd_tmb()`; `default_dd_priors()`.

`fit_dd_choice`

Fit a trial-level SS-vs-LL choice model (binomial GLMM) via TMB

Description

Fits a trial-level binary choice model from two complementary perspectives. The structural model estimates the discount rate k directly from choices via the scale-invariant relative-value comparison $\text{logit } P(\text{LL}) = \beta_0 + \gamma * ((11/\text{ss}) * D(k, \text{delay}) - 1)$, $k = \exp(X \beta_k + \sigma_u u)$. The descriptive model (Young 2018) forgoes a discount function and instead characterises choices via separate magnitude and delay sensitivities with optional correlated per-subject random slopes.

Usage

```
fit_dd_choice(
  data,
  mode = c("structural", "descriptive"),
  id_var = "id",
  ss_var = "ss_amount",
  ll_var = "ll_amount",
  delay_var = "delay",
  choice_var = "choice",
  equation = c("mazur", "exponential"),
  intercept = FALSE,
  predictors = NULL,
  random_slopes = TRUE,
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  start_values = NULL,
  tmb_control = list(iter_max = 1000, eval_max = 2000),
  multi_start = TRUE,
  verbose = 1,
  ...
)
```

Arguments

<code>data</code>	Trial-level data frame (see the <code>*_var</code> args).
<code>mode</code>	"structural" (default) estimates the discount rate k directly from choices via a discount function (Mazur or exponential); shares the IP family's k /emmeans contract. "descriptive" fits the Young (2018) correlated random-slope logistic model with separate magnitude and delay sensitivity predictors; the inferential targets are VarCorr / ranef, not emmeans.
<code>id_var, ss_var, ll_var, delay_var, choice_var</code>	Column names.
<code>equation</code>	"mazur" or "exponential" (structural mode only).
<code>intercept</code>	Logical; include the choice-bias β_0 (default FALSE). Structural only (the descriptive design carries its own intercept policy via predictors).
<code>predictors</code>	Descriptive (mode = "descriptive") only: a one-sided formula for the fixed-effect design on the logit of choosing the larger-later reward, or NULL for Young's (2018) two scale-invariant predictors ($\sim 0 + \log(\text{ll_amount} / \text{ss_amount}) + \log(\text{delay} + 1)$). Ignored when mode = "structural".
<code>random_slopes</code>	Descriptive only: logical; TRUE (default) fits two correlated per-subject random slopes on Young's two predictors ($\log(\text{ll}/\text{ss})$ and $\log(\text{delay} + 1)$), giving a full bivariate random-effect covariance; FALSE fits a pooled fixed-effect logistic model with no random effects. Ignored when mode = "structural".
<code>factors, factor_interaction, continuous_covariates</code>	Between-subject design on $\log k$ (same semantics as <code>fit_dd_tmb()</code>). Structural only.
<code>start_values, tmb_control, multi_start, verbose, ...</code>	As in <code>fit_dd_tmb()</code> .

Details

The **structural** model (mode = "structural") parameterises choices through a classical discount function (Mazur hyperbolic or exponential) and shares the IP family's k /emmeans contract: k estimates and estimated marginal means are accessible via `get_dd_param_emms()` and `get_dd_comparisons()`. The **descriptive** model (mode = "descriptive") follows Young (2018): it regresses binary choice on $\log(\text{ll_amount} / \text{ss_amount})$ (magnitude sensitivity) and $\log(\text{delay} + 1)$ (delay sensitivity) with no assumed discount function. When `random_slopes = TRUE` (default) each subject receives correlated random slopes on these two predictors; the primary inferential targets are the random-effect covariance (via `nlme::VarCorr()`) and per-subject slopes (via `nlme::ranef()`), not emmeans.

Value

An object of class `beezdiscounting_choice`.

See Also

`simulate_dd_choice()` for data generation; `nlme::VarCorr()` and `nlme::ranef()` for descriptive-mode random-effect output; `get_dd_param_emms()` and `get_dd_comparisons()` for structural-mode emmeans.

Examples

```
# Structural model: estimate a discount rate k from binary choices
sim_s <- simulate_dd_choice(n_subjects = 30, mode = "structural", seed = 1)
fit_s <- fit_dd_choice(sim_s, mode = "structural", equation = "mazur")
summary(fit_s)

# Descriptive (Young 2018) model: correlated per-subject magnitude/delay slopes
sim_d <- simulate_dd_choice(n_subjects = 30, mode = "descriptive", seed = 1)
fit_d <- fit_dd_choice(sim_d, mode = "descriptive")
summary(fit_d)
nlme::VarCorr(fit_d)
```

fit_dd_choice_brms	<i>Fit a Bayesian Structural Choice Discounting Model via brms</i>
--------------------	--

Description

The TMB structural likelihood of `fit_dd_choice()`: $\text{logit } P(LL) = [b_0] + \gamma * ((ll/ss) * D(k, \text{delay}) - 1)$ with choice sensitivity $\gamma = \exp(\text{loggamma})$ and the discount rate $k = \exp(\text{logk})$ carrying the subject random intercept. With `bernoulli("logit")` the brms nonlinear formula IS the logit, so the likelihood matches TMB exactly.

Usage

```
fit_dd_choice_brms(
  data,
  mode = c("structural", "descriptive"),
  id_var = "id",
  ss_var = "ss_amount",
  ll_var = "ll_amount",
  delay_var = "delay",
  choice_var = "choice",
  equation = c("mazur", "exponential"),
  intercept = FALSE,
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  prior = NULL,
  autoscale_priors = TRUE,
  chains = 4,
  iter = 2000,
  warmup = floor(iter/2),
  thin = 1,
  cores = getOption("mc.cores", 1L),
  seed = NA,
  backend = getOption("brms.backend", "rstan"),
```

```

control = list(adapt_delta = 0.95),
init = c("prior_center", "tmb", "random", "0"),
sample_prior = "no",
loo = TRUE,
file = NULL,
file_refit = getOption("brms.file_refit", "on_change"),
verbose = 1,
...
)

```

Arguments

<code>data</code>	Long-format trial-level data (one row per choice).
<code>mode</code>	"structural" (only mode in v1).
<code>id_var, ss_var, ll_var, delay_var, choice_var</code>	Column names, as in <code>fit_dd_choice()</code> .
<code>equation</code>	"mazur" or "exponential".
<code>intercept</code>	Include the logit-scale bias term b_0 .
<code>factors, factor_interaction, continuous_covariates</code>	Between-subject fixed-effect design on log k (same semantics as <code>fit_dd_choice()</code>); γ and b_0 stay population-level.
<code>prior</code>	Optional <code>brmsprior</code> ; user rows override the defaults.
<code>autoscale_priors</code>	Anchor the log k prior to the median delay (see <code>default_dd_priors()</code>).
<code>chains, iter, warmup, thin, cores, seed, backend, control, sample_prior</code>	MCMC settings passed to <code>brms::brm()</code> .
<code>init</code>	"prior_center" (default), "tmb" (a quiet <code>fit_dd_tmb()</code> pre-fit supplies the centers, with <code>prior_center</code> fallback on failure; the beta family maps to the TMB <code>sltb</code> pre-fit), "random", or "0"; or a list/function passed through to <code>brms::brm()</code> .
<code>loo</code>	Compute and store <code>brms::loo()</code> at fit time.
<code>file, file_refit</code>	Passed to <code>brms::brm()</code> for fit caching.
<code>verbose</code>	0 (silent), 1 (messages), 2 (full Stan output).
<code>...</code>	Passed through to <code>brms::brm()</code> .

Details

v1 implements `mode = "structural"` only: the descriptive (Young 2018) model is a plain logistic GLMM expressible directly with `brms::brm()`.

Value

An object of class `beezdiscounting_choice_brms`. Coefficients are posterior medians on the estimation scale under the TMB names (one β_k per design column, $\log \gamma$, and β_0 when `intercept = TRUE`).

See Also

[fit_dd_choice\(\)](#); [default_dd_choice_priors\(\)](#); [get_dd_param_emms\(\)](#) and [get_dd_comparisons\(\)](#) for draws-based marginal means and contrasts of k .

fit_dd_linear

*Fit the linearized Mazur hyperbola (Hinds et al., 2026)***Description**

Linearizes $D = 1/(1 + k \cdot t)$ to $\ln(1/D - 1) - \ln(t) = \ln(k) + e$ so that $\ln(k)$ has a closed-form per-subject estimator (the geometric mean of $(1/D - 1)/t$), a one-way random-effects model on the transformed scale has closed-form MLEs, and condition means can be compared with an exact F-test ([anova.beezdiscounting_linear\(\)](#)); the test is exact when the random-effects variance estimate $\hat{g} > 0$, and when $\hat{g} = 0$ the statistic is still reported, with a warning. No numerical optimization is involved.

Usage

```
fit_dd_linear(
  data,
  y_var = "y",
  x_var = "x",
  id_var = "id",
  factors = NULL,
  response_scale = c("proportion", "percent", "amount"),
  ll = NULL,
  boundary = c("clamp", "drop", "error"),
  eps = NULL,
  conf_level = 0.95
)
```

Arguments

<code>data</code>	Long data frame with subject id, delay, and indifference point columns.
<code>y_var, x_var, id_var</code>	Column names for indifference point, delay, subject.
<code>factors</code>	Optional name of ONE categorical column giving the experimental condition. Must be between-subject: every id appears under exactly one level; frames with an id under several levels are rejected because the exact F-test assumes independent units per condition.
<code>response_scale, ll</code>	As in fit_dd_tmb() .
<code>boundary</code>	How to treat y in $\{0, 1\}$: "clamp" (default; exact 0 becomes <code>eps</code> , exact 1 becomes $1 - \text{eps}$, nothing else moves), "drop" (per-subject estimates only unless the design stays balanced), "error".

eps	Where exact 0 / 1 are placed under boundary = "clamp" (eps and 1 - eps); default 1/(2 * ll) if ll is given, else 0.005.
conf_level	Confidence level for per-subject ln(k) intervals (a single number strictly between 0 and 1); also the default level of confint(parm = "subject").

Details

Indifference points at exactly 0 or 1 are undefined under the transform; by default they are moved to eps and 1 - eps respectively (boundary = "clamp"). Only exact 0 and 1 are touched — interior points, however close to a bound, are used as observed. This is a package decision; the paper does not address boundary values. Use boundary = "drop" or "error" when you would rather not impute them.

Value

An object of class `beezdiscounting_linear`: a list with subjects (per-unit tibble of ln k estimates, intervals and log-likelihoods), re (closed-form random-effects MLEs mu, sigma2, g, or NULL if the design is unbalanced), design (factor name and levels), transform (boundary bookkeeping; the counts refer to the retained units), data (the long frame: id, x, y, the factor, and the derived columns condition, unit, y_lin, d_used, log_jac; factors may not name one of the derived columns, and other user columns are not carried), conf_level, and call.

References

Hinds, D., Tegge, A. N., Stein, J. S., LaConte, S. M., McClure, S. M., & Ferreira, M. A. R. (2026). To linearize or not to linearize: That is the Mazur delay discounting question. *Journal of Mathematical Psychology*, 130, 103006. doi:10.1016/j.jmp.2026.103006

See Also

[beezdiscounting_linear-methods](#) for the S3 surface, [plot.beezdiscounting_linear\(\)](#), [anova.beezdiscounting_linear](#), [simulate_dd_linear\(\)](#), and [vignette\("linearized-mazur"\)](#).

Examples

```
fit <- fit_dd_linear(dd_ip)
fit
```

fit_dd_tmb

Fit an indifference-point mixed-effects discounting model via TMB

Description

Fits a 1-parameter discounting model (Mazur hyperbolic or exponential) with a random intercept on log k, between-subject fixed effects, and either an SLT-beta or Gaussian observation family, using Template Model Builder for exact AD + Laplace approximation.

Usage

```
fit_dd_tmb(
  data,
  y_var = "y",
  x_var = "x",
  id_var = "id",
  equation = c("mazur", "exponential", "green-myerson", "rachlin"),
  family = c("sltb", "gaussian"),
  random_effects = k ~ 1,
  factors = NULL,
  factor_interaction = FALSE,
  continuous_covariates = NULL,
  ll = NULL,
  response_scale = c("proportion", "percent", "amount"),
  start_values = NULL,
  tmb_control = list(iter_max = 1000, eval_max = 2000),
  multi_start = TRUE,
  verbose = 1,
  covariance_structure = c("pdSymm", "pdDiag"),
  ...
)
```

Arguments

<code>data</code>	Long data frame with subject id, delay, and indifference proportion columns.
<code>y_var, x_var, id_var</code>	Column names (defaults "y", "x", "id").
<code>equation</code>	One of "mazur", "exponential", "green-myerson", or "rachlin". The two 2-parameter (hyperboloid) forms add a single population nonlinearity exponent s (estimated on the log scale) and reduce to "mazur" at $s = 1$.
<code>family</code>	Observation family: "sltb" (default) or "gaussian".
<code>random_effects</code>	RE formula: $k \sim 1$ (single random intercept on log k), $k + \phi \sim 1$ (a joint 2-D random intercept on (log k , log ϕ), SLT-beta only), or $k + s \sim 1$ (a joint 2-D random intercept on (log k , log s), Green-Myerson and Rachlin only).
<code>factors</code>	Character vector of between-subject factor names.
<code>factor_interaction</code>	Logical; include a pairwise factor interaction.
<code>continuous_covariates</code>	Character vector of covariate names.
<code>ll</code>	Optional larger-later reward for amount-scale coercion.
<code>response_scale</code>	One of "proportion", "percent", "amount".
<code>start_values</code>	Optional named list overriding defaults.
<code>tmb_control</code>	Optimizer control list.
<code>multi_start</code>	Logical; if TRUE (default), run the 3-set guarded multi-start.
<code>verbose</code>	Integer verbosity (0 silent, 1 progress, 2 debug).

covariance_structure Covariance for a 2-D random effect ($k + \text{phi} \sim 1$ or $k + s \sim 1$): "pdSymm" (default; correlated random intercepts) or "pdDiag" (independent, correlation fixed at 0). Ignored for $k \sim 1$.

... Reserved.

Value

An object of class `beezdiscounting_tmb` with components:

call The matched call.

opt Normalized optimizer result (par, objective, convergence, message).

model List of coefficients, se, and variance_components.

sdr TMB sdreport object (or NULL if SE computation failed).

hessian_pd Logical positive-definiteness of the Hessian.

param_info Model metadata (equation, family, dimensions, factor spec, parsed random effects).

formula_details Fixed-effect design (X, rhs, contrasts).

subject_pars Data frame of subject-level parameters. For a 1-RE fit ($k \sim 1$) the columns are `id`, `u_i`, `k`; for a phi-target 2-RE fit ($k + \text{phi} \sim 1$) they are `id`, `re_k`, `re_phi`, `k`, `phi`; for an s-target 2-RE fit ($k + s \sim 1$, GM/Rachlin) they are `id`, `re_k`, `re_s`, `k`, `s` where `s` is soft-clamped toward (0.05, 20).

loglik, AIC, BIC Fit statistics.

converged, se_available Convergence / SE-availability flags.

opt_warnings Character vector of optimizer warnings.

data The single filtered model frame (`id/x/y` + retained design columns), row-aligned with the design matrix.

data_all The validated frame before complete-casing.

coercion_info Scale-coercion/clamping audit list.

References

Young, M. E. (2017). Discounting: A practical guide to multilevel analysis of indifference data. *Journal of the Experimental Analysis of Behavior*, 108(1), 97-112. doi:10.1002/jeab.265

Kim, M., Koffarnus, M. N., & Franck, C. T. (2024). Thinking inside the bounds: Improved error distributions for indifference point data analysis and simulation via beta regression using common discounting functions. *arXiv preprint arXiv:2404.18000*.

Kim, M., Kaplan, B. A., Koffarnus, M. N., & Franck, C. T. (2025). Scale-location-truncated beta regression: Expanding beta regression to accommodate 0 and 1. *arXiv preprint arXiv:2509.13167*.

Examples

```
# Small two-subject long-format indifference-point data frame.
dd <- data.frame(
  id = rep(c("s1", "s2"), each = 5),
  x = rep(c(7, 30, 180, 365, 730), times = 2),
```

```

y = c(0.95, 0.80, 0.45, 0.30, 0.15,
      0.90, 0.70, 0.40, 0.25, 0.10)
)
fit <- fit_dd_tmb(dd, equation = "mazur", family = "sltb", verbose = 0)
exp(fit$model$coefficients[["beta_k"]]) # population k

```

```
fitted.beezdiscounting_choice
```

Fitted values for a beezdiscounting_choice fit

Description

Fitted values for a beezdiscounting_choice fit

Usage

```
## S3 method for class 'beezdiscounting_choice'
fitted(object, level = c("subject", "population"), ...)
```

Arguments

object	A beezdiscounting_choice object.
level	"subject" (default) or "population".
...	Unused.

Value

Numeric vector of fitted choice probabilities, length nobs(object).

```
fitted.beezdiscounting_tmb
```

Fitted values for a beezdiscounting_tmb fit

Description

Fitted values for a beezdiscounting_tmb fit

Usage

```
## S3 method for class 'beezdiscounting_tmb'
fitted(object, level = c("subject", "population"), ...)
```

Arguments

object	A beezdiscounting_tmb object.
level	"subject" (default) or "population".
...	Unused.

Value

Numeric vector of fitted indifference proportions, length nobs(object).

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, verbose = 0)
head(fitted(fit))
```

five.fivetrial_dd	<i>Example Qualtrics output from the 5.5 trial delay discounting template.</i>
-------------------	--

Description

An example dataset containing four participants' data (two typical discounting patterns and two patterns suggesting potential misattention to the task).

Usage

```
five.fivetrial_dd
```

Format

Example Qualtrics output

five.fivetrial_pd	<i>Example Qualtrics output from the 5.5 trial probability discounting template.</i>
-------------------	--

Description

An example dataset containing four participants' data.

Usage

```
five.fivetrial_pd
```

Format

Example Qualtrics output

```
fixef.beezdiscounting_choice
```

Extract fixed effects from a structural choice model

Description

Extract fixed effects from a structural choice model

Usage

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

Arguments

object	A beezdiscounting_choice object.
...	Unused.

Value

Named numeric vector (identical to coef()).

```
fixef.beezdiscounting_tmb
```

Extract fixed effects from a TMB discounting model

Description

Extract fixed effects from a TMB discounting model

Usage

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

Arguments

object	A beezdiscounting_tmb object.
...	Unused.

Value

Named numeric vector (identical to coef()).

generate_data_mcq	<i>Generate fake MCQ data</i>
-------------------	-------------------------------

Description

Generate fake MCQ data

Usage

```
generate_data_mcq(n_ids = 100, n_items = 27, seed = 1234, prop_na = 0)
```

Arguments

n_ids	Number of subjectids
n_items	Number of trials
seed	Random seed; identical seeds give identical results. The caller's RNG state is restored on exit.
prop_na	Proportion of NAs in the entire data set

Value

Dataframe of subjectid, questionid, and response

Examples

```
generate_data_mcq(n_ids = 2, n_items = 27, prop_na = .01)
```

get_dd_comparisons	<i>Factor-level comparisons of the discount rate k</i>
--------------------	--

Description

Computes factor-level contrasts of the discount rate k from a fitted `beezdiscounting_tmb` model (or a structural `beezdiscounting_choice` fit). k is linear in the fixed-effect coefficients on the natural-log scale ($\log k = X \beta_k$), so each contrast is a linear combination of β_k with a Wald standard error from the β_k block of `TMB::sdreport()`'s fixed-effect covariance. Contrasts are **reported on the log10 scale** and, optionally, as multiplicative ratios ($\text{ratio} = \exp(\text{est_log})$). The returned container mirrors the `beezdemand` and `beezdemand_comparison` shape, so `tidy.beezdiscounting_comparison()` gives a flat, cross-backend frame. Bayesian fits (`beezdiscounting_brms`, `beezdiscounting_choice_brms`) compute per-draw contrasts over the same reference grid: posterior medians with quantile credible intervals and `post.prob` in place of adjusted p-values (no multiplicity adjustment; the joint posterior already encodes contrast dependence).

Usage

```
get_dd_comparisons(
  fit,
  compare_specs = NULL,
  contrast_type = c("pairwise", "trt.vs.ctrl"),
  contrast_by = NULL,
  adjust = "holm",
  at = NULL,
  ci_level = 0.95,
  report_ratios = TRUE,
  ...
)
```

Arguments

<code>fit</code>	A <code>beezdiscounting_tmb</code> , <code>structural beezdiscounting_choice</code> , <code>beezdiscounting_brms</code> , or <code>beezdiscounting_choice_brms</code> object.
<code>compare_specs</code>	Optional one-sided formula naming the factor subset to contrast (e.g. <code>~ condition</code>). Omitted fitted factors are marginalized over with equal weights across the full crossing of their levels. If <code>NULL</code> (default), all fitted factors are retained. Unknown names abort.
<code>contrast_type</code>	Character. <code>"pairwise"</code> (all <code>choose(n, 2)</code> pairs, factor-level order) or <code>"trt.vs.ctrl"</code> (each level vs. the first/reference level).
<code>contrast_by</code>	Optional <code>NULL</code> (default) or character vector of factor name(s) within <code>compare_specs</code> to condition the contrasts on. Within each observed combination of by-level(s), pairwise (or <code>trt.vs.ctrl</code>) contrasts are computed over the remaining (non-by) factors, with p-value adjustment applied per by-cell . A <code>contrast_by</code> factor absent from <code>compare_specs</code> aborts.
<code>adjust</code>	Character. P-value adjustment method; must be one of <code>stats::p.adjust.methods</code> (default <code>"holm"</code>). <code>emmeans</code> -only methods such as <code>"tukey"/"sidak"</code> are rejected (the TMB backend uses an asymptotic z + <code>stats::p.adjust()</code>).
<code>at</code>	Named list specifying factor levels and/or continuous-covariate values to condition on, as in <code>get_dd_param_emms()</code> .
<code>ci_level</code>	Numeric. Confidence level for intervals (default 0.95).
<code>report_ratios</code>	Logical. If <code>TRUE</code> (default), include a <code>contrasts_ratio</code> block (multiplicative ratios).
<code>...</code>	Additional arguments (reserved; <code>factors_in_emm</code> is accepted as a lower-level alternative to <code>compare_specs</code>).

Value

A `beezdiscounting_comparison` object: a list with a single element `k`, itself a list with `emmeans` (the EMM tibble from `get_dd_param_emms()`), `contrasts_log10` (columns `contrast`, `estimate`, `std.error`, `statistic`, `df`, `conf.low`, `conf.high`, `p.value`), and (if `report_ratios`) `contrasts_ratio` (columns `contrast`, `ratio`, `conf.low`, `conf.high`, `p.value`). When `contrast_by` is active, the contrast tables gain leading by-column(s) before `contrast`. Attributes `backend`, `adjustment_method`, `compare_specs_used`, `contrast_type_used`, and `contrast_by_used` describe the call.

See Also

`tidy.beezdiscounting_comparison()` for the cross-backend frame.

Examples

```
sim <- simulate_dd_ip(
  n_subjects = 30, n_conditions = 2, delta_k = c(0, log(3)), seed = 1
)
fit <- fit_dd_tmb(sim, factors = "condition", verbose = 0)
get_dd_comparisons(fit, contrast_type = "pairwise")
```

<code>get_dd_param_emms</code>	<i>Estimated marginal means of the discount rate k</i>
--------------------------------	--

Description

Computes estimated marginal means (EMMs) of the discount rate k from a fitted `beezezdiscounting_tmb` model (or a structural `beezezdiscounting_choice` fit, which shares the $\log k$ design). EMMs are computed on the $\log k$ scale (linear in the fixed-effect coefficients) using the averaging-matrix reference grid, then back-transformed with `exp()` so that $k = \exp(k_{\log})$. Standard errors use the `beta_k` block of `TMB::sdreport()`'s fixed-effect covariance; intervals are Wald on the log scale and exponentiated. Bayesian fits (`beezezdiscounting_brms`, `beezezdiscounting_choice_brms`) use the same reference grid with draws-based summaries: posterior medians and equal-tailed quantile intervals, no delta method.

Usage

```
get_dd_param_emms(fit, factors_in_emm = NULL, at = NULL, ci_level = 0.95, ...)
```

Arguments

<code>fit</code>	A <code>beezezdiscounting_tmb</code> , structural <code>beezezdiscounting_choice</code> , <code>beezezdiscounting_brms</code> , or <code>beezezdiscounting_choice_brms</code> object.
<code>factors_in_emm</code>	Character vector of factors to retain in the EMM reference grid. A strict subset marginalizes the omitted factors with equal weights across the full crossing of their levels (emmeans' default <code>weights = "equal"</code>); <code>NULL</code> (default) retains all fitted factors; <code>character(0)</code> marginalizes everything to a single grand-mean cell.
<code>at</code>	Named list specifying factor levels and/or continuous-covariate values for conditional EMMs (one numeric per covariate; multiple values warn and use the first). <code>at</code> on an omitted factor restricts the level set averaged over.
<code>ci_level</code>	Numeric confidence level for intervals (default 0.95).
<code>...</code>	Additional arguments (currently unused).

Value

A tibble with columns `level`, `k`, `k_log`, `std.error`, `conf.low`, `conf.high`. `k_log` is the marginal mean on the log k scale; $k = \exp(k_log)$; `std.error` is the SE of `k_log`; the intervals are on the k (natural) scale.

Examples

```
sim <- simulate_dd_ip(
  n_subjects = 30, n_conditions = 2, delta_k = c(0, log(3)), seed = 1
)
fit <- fit_dd_tmb(sim, factors = "condition", verbose = 0)
get_dd_param_emms(fit)
```

get_lookup_table	<i>Get an internal instrument design table</i>
------------------	--

Description

Get an internal instrument design table

Usage

```
get_lookup_table(items = NULL, instrument = NULL)
```

Arguments

<code>items</code>	Deprecated. Number of MCQ items (27 or 21), kept as a back-compatible alias for <code>instrument = "mcq27" / "mcq21"</code> (including positional calls like <code>get_lookup_table(21)</code>). Use <code>instrument</code> in new code; <code>items</code> cannot address the PDQ.
<code>instrument</code>	Instrument key: <code>"mcq27"</code> (Kirby, Petry, & Bickel, 1999), <code>"mcq21"</code> (Kirby & Marakovic, 1996), or <code>"pdq"</code> (Madden, Petry, & Johnson, 2009). Supply <code>items</code> or <code>instrument</code> , not both. The default (neither) returns the 27-item MCQ design.

Value

Data frame with the complete item design. MCQ tables: `questionid`, `magnitude`, `kindiff`, `k_rank`, `ss_amount`, `ll_amount`, `delay` (days). PDQ table: `questionid`, `block`, `h_rank`, `overall_rank` (the item's position in the pooled overall ladder used by the `overall_h` extension in `score_pdq()`), `sc_amount`, `lu_amount`, `prob`, `theta` (odds against), `hindiff`.

Examples

```
get_lookup_table()
get_lookup_table(items = 21)
get_lookup_table(instrument = "pdq")
```

```
glance.beezdiscounting_brms
```

Glance at a beezdiscounting_brms model

Description

The dd glance columns with backend = "brms" and logLik/AIC/BIC = NA_real_ by design, plus the Bayesian comparison and diagnostic currency (elpd_loo, p_loo, looic, rhat_max, ess_bulk_min, num_divergent).

Usage

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

Arguments

x A beezdiscounting_brms object.
... Unused.

Value

A one-row tibble.

```
glance.beezdiscounting_choice
```

Glance at a beezdiscounting_choice model

Description

Returns a one-row tibble of model-level summary statistics suitable for use with broom::glance() workflows and multi-model comparison tables.

Usage

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

Arguments

x A beezdiscounting_choice object.
... Unused.

Value

A one-row tibble with columns model_class, backend, mode, equation, nobs, n_subjects, n_random_effects, converged, logLik, AIC, BIC.

```
glance.beezdiscounting_choice_brms
```

Glance at a beezdiscounting_choice_brms model

Description

Glance at a beezdiscounting_choice_brms model

Usage

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

Arguments

x	A beezdiscounting_choice_brms object.
...	Unused.

Value

A one-row tibble.

```
glance.beezdiscounting_tmb
```

Glance at a beezdiscounting_tmb model

Description

Returns a one-row tibble of model-level summary statistics suitable for use with broom::glance() workflows and multi-model comparison tables.

Usage

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

Arguments

x	A beezdiscounting_tmb object.
...	Unused.

Value

A one-row tibble with columns model_class, backend, equation, family, nobs, n_subjects, n_random_effects, converged, logLik, AIC, BIC.

long_to_wide_mcq	<i>Reshape MCQ data long to wide</i>
------------------	--------------------------------------

Description

Reshape MCQ data long to wide

Usage

```
long_to_wide_mcq(dat, q_col = "questionid", ans_col = "response")
```

Arguments

dat	Long format MCQ
q_col	Name of the question column (default is "questionid")
ans_col	Name of the answer column (default is "response")

Value

Wide format data frame

Examples

```
long_to_wide_mcq(mcq27)
```

long_to_wide_mcq_excel	<i>Reshape MCQ data from long to wide (as used in the 21- and 27-Item Monetary Choice Questionnaire Automated Scorer)</i>
------------------------	---

Description

Reshape MCQ data from long to wide (as used in the 21- and 27-Item Monetary Choice Questionnaire Automated Scorer)

Usage

```
long_to_wide_mcq_excel(dat, subj_col = "subjectid", ans_col = "response")
```

Arguments

dat	Long format MCQ data
subj_col	Character column name of subject ids
ans_col	Character column name of responses

Value

Wide format MCQ data that can be used in the Excel Automated Scorers

Examples

```
long_to_wide_mcq_excel(generate_data_mcq(2))
```

mcq_to_choice	<i>Convert 27- or 21-item MCQ responses to a trial-level choice frame</i>
---------------	---

Description

Reshapes long-form Monetary Choice Questionnaire (MCQ) responses into the per-trial smaller-sooner versus larger-later choice frame consumed by `fit_dd_choice()`, joining each questionid to the canonical item design (immediate amount, delayed amount, delay) bundled in the lookup table (see `get_lookup_table()`; Kirby, Petry, & Bickel (1999) for the 27-item MCQ, or Kirby & Maraković (1996, Table 1) for the 21-item MCQ).

Usage

```
mcq_to_choice(  
  responses,  
  items = 27,  
  id_var = "subjectid",  
  question_var = "questionid",  
  response_var = "response"  
)
```

Arguments

- responses Long-form data frame with one row per MCQ item per subject, holding the columns named by id_var, question_var, and response_var. response is 0 for the smaller-immediate reward (SIR/SS) and 1 for the larger-delayed reward (LDR/LL), the same coding `fit_dd_choice()` expects, so no recoding is applied.
- items Number of MCQ items: 27 (Kirby, Petry, & Bickel, 1999) or 21 (Kirby & Maraković, 1996). Default is 27.
- id_var, question_var, response_var Column names in responses for the subject id, MCQ question id, and the binary choice. Defaults match the bundled mcq27 dataset ("subjectid", "questionid", "response").

Details

Unknown or non-coercible question ids raise an error rather than silently producing unmatched rows. Question ids 1-21 are valid in both the 21- and 27-item designs, so passing the wrong `items` does not error; it silently returns the wrong amounts/delays for those ids. Make sure `items` matches the instrument actually administered. Ragged input is allowed (subjects need not have all items) and NA responses are preserved (they are complete-cased by `fit_dd_choice()`). For the strict scorer see `score_mcq()`.

Value

A [tibble](#) with columns `id` (character), `ss_amount`, `ll_amount`, `delay` (days), and `choice` (0/1, 1 = chose LL), in the input row order. Ready to pass to `fit_dd_choice()`.

See Also

[fit_dd_choice\(\)](#), [score_mcq\(\)](#), [get_lookup_table\(\)](#), [mcq27_to_choice\(\)](#)

Examples

```
ch <- mcq_to_choice(mcq27, items = 27)
head(ch)
# feeds directly into the structural choice model (requires TMB):
# fit_dd_choice(ch, mode = "structural", equation = "mazur")
```

mcq21

Example 21-item MCQ data

Description

A dataset containing two illustrative participants' 21-item MCQ (Kirby & Marakovic, 1996) responses: subject 1 responds like the median subject in Kirby & Marakovic (1996) (switching between the eighth and ninth ladder items, overall k of about 0.007) and subject 2 always chooses the larger delayed reward. These responses are constructed for illustration only, not drawn from an empirical sample.

Usage

```
mcq21
```

Format

Long-form data.frame with columns: `subjectid`, `questionid`, `response`.

Source

Constructed in `data-raw/mcq21_data.R`.

mcq27	<i>Example 27-item MCQ data</i>
-------	---------------------------------

Description

A dataset containing two participants' data (same data as in the paper by Kaplan et al., 2016)

Usage

mcq27

Format

Long-form data.frame with columns: subjectid, questionid, response.

mcq27_to_choice	<i>Convert 27-item MCQ responses to a trial-level choice frame</i>
-----------------	--

Description

Reshapes long-form Monetary Choice Questionnaire (MCQ) responses into the per-trial smaller-sooner versus larger-later choice frame consumed by `fit_dd_choice()`, joining each questionid to the canonical item design (immediate amount, delayed amount, delay) bundled in the lookup table (see `get_lookup_table()`; Kirby, Petry, & Bickel (1999) for the 27-item MCQ, or Kirby & Maraković (1996, Table 1) for the 21-item MCQ).

Usage

```
mcq27_to_choice(  
  responses,  
  id_var = "subjectid",  
  question_var = "questionid",  
  response_var = "response"  
)
```

Arguments

- responses Long-form data frame with one row per MCQ item per subject, holding the columns named by id_var, question_var, and response_var. response is 0 for the smaller-immediate reward (SIR/SS) and 1 for the larger-delayed reward (LDR/LL), the same coding `fit_dd_choice()` expects, so no recoding is applied.
- id_var, question_var, response_var Column names in responses for the subject id, MCQ question id (1-27), and the binary choice. Defaults match the bundled mcq27 dataset ("subjectid", "questionid", "response").

Details

Unknown or non-coercible question ids raise an error rather than silently producing unmatched rows. Question ids 1-21 are valid in both the 21- and 27-item designs, so passing the wrong `items` does not error; it silently returns the wrong amounts/delays for those ids. Make sure `items` matches the instrument actually administered. Ragged input is allowed (subjects need not have all items) and NA responses are preserved (they are complete-cased by `fit_dd_choice()`). For the strict scorer see `score_mcq()`.

Value

A [tibble](#) with columns `id` (character), `ss_amount`, `ll_amount`, `delay` (days), and `choice` (0/1, 1 = chose LL), in the input row order. Ready to pass to `fit_dd_choice()`.

See Also

[fit_dd_choice\(\)](#), [score_mcq27\(\)](#), [get_lookup_table\(\)](#), [mcq_to_choice\(\)](#)

Examples

```
ch <- mcq27_to_choice(mcq27)
head(ch)
# feeds directly into the structural choice model (requires TMB):
# fit_dd_choice(ch, mode = "structural", equation = "mazur")
```

pdq

Example 30-item PDQ data

Description

A dataset containing two illustrative participants' Probability Discounting Questionnaire (Madden, Petry, & Johnson, 2009) responses: subject 1 chooses the guaranteed reward through the fifth h rank of every block and the risky reward after (block h values of about 1.22), and subject 2 always chooses the risky reward. These responses are constructed for illustration only, not drawn from an empirical sample.

Usage

```
pdq
```

Format

Long-form `data.frame` with columns: `subjectid`, `questionid`, `response` (0 = guaranteed, 1 = risky).

Source

Constructed in `data-raw/pdq_data.R`.

pdq_to_choice

*Convert PDQ responses to a trial-level choice frame***Description**

Reshapes long-form Probability Discounting Questionnaire (PDQ) responses into a per-trial certain-versus-risky choice frame, joining each questionid to the canonical item design (certain amount, probabilistic amount, probability, and odds against $\theta = (1 - p)/p$) from Madden, Petry, & Johnson (2009; see [get_lookup_table\(\)](#)).

Usage

```
pdq_to_choice(
  responses,
  id_var = "subjectid",
  question_var = "questionid",
  response_var = "response"
)
```

Arguments

responses Long-form data frame with one row per PDQ item per subject, holding the columns named by **id_var**, **question_var**, and **response_var**. **response** is 0 for the smaller guaranteed reward and 1 for the larger risky reward.

id_var, question_var, response_var Column names in **responses** for the subject id, PDQ question id (1-30), and the binary choice. Defaults match the bundled pdq dataset ("subjectid", "questionid", "response").

Details

Unknown or non-coercible question ids raise an error rather than silently producing unmatched rows. Ragged input is allowed (subjects need not have all items) and NA responses are preserved. For the strict scorer see [score_pdq\(\)](#).

Value

A [tibble](#) with columns **id** (character), **sc_amount**, **lu_amount**, **prob**, **theta** (odds against winning), and **choice** (0/1, 1 = chose the risky reward), in the input row order.

See Also

[score_pdq\(\)](#), [get_lookup_table\(\)](#), [mcq_to_choice\(\)](#)

Examples

```
ch <- pdq_to_choice(pdq)
head(ch)
```

plot.beezdiscounting_brms

Plot a Bayesian (brms) indifference-point discounting model

Description

Visualize a fitted `fit_dd_brms()` model. `type = "population"` (default) draws the posterior-median discount curve with a credible band for the population conditional mean (random effects set to zero; a credible band rather than a posterior-predictive band) over the observed (boundary-squeezed) indifference points; `type = "individual"` adds per-subject median curves; `type = "parameters"` shows the posterior subject-k caterpillar; `type = "resid"` plots standardized residuals against fitted values.

Usage

```
## S3 method for class 'beezdiscounting_brms'
plot(
  x,
  type = c("population", "individual", "parameters", "resid"),
  ids = NULL,
  at = NULL,
  n_points = 200,
  x_trans = c("log10", "linear"),
  show_observed = TRUE,
  ...
)
```

Arguments

<code>x</code>	A <code>beezdiscounting_brms</code> object.
<code>type</code>	One of "population", "individual", "parameters", "resid".
<code>ids</code>	Optional subset of subject ids for <code>type = "individual"</code> .
<code>at</code>	Optional named list conditioning the population curve on factor levels / covariate values.
<code>n_points</code>	Number of delay points in the curve grid.
<code>x_trans</code>	Delay-axis scale: "log10" (default) or "linear".
<code>show_observed</code>	Overlay the observed indifference points.
<code>...</code>	Unused.

Value

A `ggplot2::ggplot` object.

plot.beezdiscounting_choice

Plot a choice-based discounting model

Description

Visualize a fitted `fit_dd_choice()` model. For a *structural* fit, `type = "population"` (default) draws the implied discount curve from the estimated discount rate, `type = "individual"` adds the per-subject curves, and `type = "parameters"` shows the subject-k distribution. For both structural and *descriptive* (Young 2018) fits, `type = "calibration"` plots the observed choice proportion against the fitted $P(\text{choose LL})$, binned by fitted probability (a likelihood-geometry-agnostic goodness-of-fit view). Descriptive fits have no structural discount rate, so the curve/parameter types are unavailable.

Usage

```
## S3 method for class 'beeزdiscounting_choice'
plot(
  x,
  type = NULL,
  ids = NULL,
  at = NULL,
  n_points = 200,
  x_trans = c("log10", "linear"),
  show_observed = TRUE,
  ...
)
```

Arguments

<code>x</code>	A <code>beeزdiscounting_choice</code> object.
<code>type</code>	One of "population", "individual", "calibration", "parameters". Defaults to "population" (structural) or "calibration" (descriptive).
<code>ids</code>	Optional subset of subject ids for <code>type = "individual"</code> .
<code>at</code>	Optional named list conditioning the curve on factor levels / covariate values.
<code>n_points</code>	Number of delay points in the curve grid.
<code>x_trans</code>	Delay-axis scale: "log10" (default) or "linear".
<code>show_observed</code>	Unused for the implied discount curve (kept for a consistent signature across tiers).
<code>...</code>	Unused.

Value

A `ggplot2::ggplot` object.

Examples

```
ch <- simulate_dd_choice(n_subjects = 12, seed = 1)
fit <- fit_dd_choice(ch, equation = "mazur")
plot(fit)
plot(fit, type = "calibration")
```

```
plot.beezdiscounting_choice_brms
```

Plot a Bayesian (brms) choice-based discounting model

Description

Visualize a fitted `fit_dd_choice_brms()` model. `type = "population"` (default) draws the posterior implied discount curve with a credible band; `type = "individual"` adds per-subject (posterior-median) curves; `type = "parameters"` shows the posterior subject-k caterpillar; and `type = "calibration"` plots observed choice proportion against fitted $P(\text{choose LL})$, binned by fitted probability.

Usage

```
## S3 method for class 'beeZdiscounting_choice_brms'
plot(
  x,
  type = c("population", "individual", "calibration", "parameters"),
  ids = NULL,
  at = NULL,
  n_points = 200,
  x_trans = c("log10", "linear"),
  show_observed = TRUE,
  ...
)
```

Arguments

<code>x</code>	A <code>beeZdiscounting_choice_brms</code> object.
<code>type</code>	One of "population", "individual", "calibration", "parameters".
<code>ids</code>	Optional subset of subject ids for <code>type = "individual"</code> .
<code>at</code>	Optional named list conditioning the population curve on factor levels / covariate values.
<code>n_points</code>	Number of delay points in the curve grid.
<code>x_trans</code>	Delay-axis scale: "log10" (default) or "linear".
<code>show_observed</code>	Unused for the implied discount curve.
<code>...</code>	Unused.

Value

A `ggplot2::ggplot` object.

plot.beezdiscounting_comparison

Plot group differences in discount rate

Description

Forest plot of the pairwise (or treatment-vs-control) contrasts from `get_dd_comparisons()`. `type = "ratio"` (default) shows each k ratio on a log axis with a reference line at 1; `type = "difference"` shows the difference in $\log_{10} k$ with a reference line at 0. A contrast is flagged when its interval excludes the null (a backend-agnostic rule that applies to both the frequentist (TMB) and Bayesian (brms) backends).

Usage

```
## S3 method for class 'beezdiscounting_comparison'
plot(x, type = c("ratio", "difference"), ...)
```

Arguments

<code>x</code>	A <code>beezdiscounting_comparison</code> object.
<code>type</code>	"ratio" (default) or "difference".
<code>...</code>	Unused.

Value

A `ggplot2::ggplot` object.

Examples

```
sim <- simulate_dd_ip(n_subjects = 30, n_conditions = 2, delta_k = c(0, 0.5), seed = 1)
fit <- fit_dd_tmb(sim, equation = "mazur", factors = "condition")
cmp <- get_dd_comparisons(fit)
plot(cmp)
```

plot.beezdiscounting_linear

Plot a linearized Mazur discounting fit

Description

Visualize a fitted `fit_dd_linear()` model. `type = "population"` (default) draws the hyperbola implied by each condition's geometric-mean discount rate $\exp(\mu)$ over the observed indifference points (one curve per condition when the fit has a factor); `type = "individual"` adds each subject's own closed-form hyperbola $1 / (1 + k_i x)$ as thin lines; `type = "transformed"` shows the linearization itself, $\ln(1/D - 1)$ against $\ln(\text{delay})$ per subject with a slope-1 line at that subject's $\ln k$; `type = "parameters"` shows the subject $\ln k$ estimates with their t-intervals (by condition, with the condition mean μ and its interval from `confint()` overlaid, when the fit has a factor; otherwise ordered as a caterpillar), or the same quantities as k on a log10 axis with `k_scale = "log10"`; `type = "resid"` plots the standardized residual on the transformed scale against $\ln(\text{delay})$.

Usage

```
## S3 method for class 'beezdiscounting_linear'
plot(
  x,
  type = c("population", "individual", "transformed", "parameters", "resid"),
  ids = NULL,
  n_points = 200,
  x_trans = c("log10", "linear"),
  show_observed = TRUE,
  k_scale = c("ln", "log10"),
  ...
)
```

Arguments

<code>x</code>	A <code>beezdiscounting_linear</code> object.
<code>type</code>	One of "population", "individual", "transformed", "parameters", "resid".
<code>ids</code>	Optional subset of subject ids for <code>type = "individual"</code> and <code>type = "transformed"</code> .
<code>n_points</code>	Number of delay points in the curve grid.
<code>x_trans</code>	Delay-axis scale: "log10" (default) or "linear".
<code>show_observed</code>	Overlay the observed indifference points.
<code>k_scale</code>	For <code>type = "parameters"</code> : "ln" (default) draws $\ln k$ on a linear axis; "log10" draws k on a log10 axis as the other tiers do.
<code>...</code>	Unused.

Details

The method shares its core arguments (`type`, `ids`, `n_points`, `x_trans`, `show_observed`) with `plot.beezdiscounting_tmb()`; it adds `k_scale` and omits `at` (a linearized fit has no covariates or reference grid, so there is nothing to condition on). `x_trans`, `n_points` and `show_observed` apply to the "population" and "individual" curves only; `ids` applies to "individual" (default: every subject) and "transformed" (default: the first 12 subjects, with a message when the fit has more).

Unlike the mixed-model tiers there is no shrinkage: every subject's curve is its own closed-form estimate, and the per-subject pictures do not need the random-effects component. "population" requires it (the condition means μ); when the design was unbalanced and `re` is `NULL`, "population"

errors, "individual" draws the subject curves only, and "parameters" omits the condition means with a message.

Two pictures deliberately differ from the mixed-model tiers because this model is fitted on the linearized scale. "parameters" defaults to $\ln k$ (the scale of `coef()`, `confint()` and `anova()`'s effect size); `k_scale = "log10"` gives the sibling tiers' picture, k on a $\log_{10}$ axis. "resid" plots the transformed-scale residual against $\ln(\text{delay})$, the model's regressor with its slope fixed at 1, so a trend across delay may indicate delay-dependent lack of fit; the sibling tiers plot residuals against fitted values instead. `k_scale` affects only "parameters".

The "transformed" abscissa is $\ln(\text{delay})$ by construction (a $\log_{10}$ axis would break the slope-1 reference), so `x_trans` is ignored there. Indifference points at exactly 0 or 1 are shown at their observed value on the raw scale, and at their boundary-handled value `d_used` on the transformed scale; points dropped under `boundary = "drop"` do not appear on the transformed scale. "resid" uses `augment()`'s `.std_resid`, the transformed-scale residual divided by the model's error standard deviation.

Value

A `ggplot2::ggplot` object.

See Also

[fit_dd_linear\(\)](#), [beezdiscounting_linear-methods](#), [vignette\("linearized-mazur"\)](#).

Examples

```
sim <- simulate_dd_linear(
  n_subjects = 10, delays = c(7, 30, 90, 365),
  mu = c(A = -6, B = -4.5), sigma2 = 2, g = 8, seed = 1
)
fit <- fit_dd_linear(sim, factors = "condition")
plot(fit)
plot(fit, type = "individual")
plot(fit, type = "transformed", ids = c("A_1", "B_1"))
plot(fit, type = "parameters")
```

plot.beezdiscounting_tmb

Plot a mixed-effects discounting model

Description

Visualize a fitted `fit_dd_tmb()` model. `type = "population"` draws the population (random-effects-at-zero) discount curve over the observed indifference points; `type = "individual"` adds the per-subject curves (the shrinkage picture). For a fit with factors the population curve is drawn once per factor-level combination; use `at` to condition on specific levels or covariate values. `type = "parameters"` shows the distribution of the subject-specific discount rate k (log scale); `type = "resid"` plots standardized (Pearson) residuals against fitted values.

Usage

```
## S3 method for class 'beezdiscounting_tmb'
plot(
  x,
  type = c("population", "individual", "parameters", "resid"),
  ids = NULL,
  at = NULL,
  n_points = 200,
  x_trans = c("log10", "linear"),
  show_observed = TRUE,
  ...
)
```

Arguments

x	A beezdiscounting_tmb object.
type	One of "population", "individual", "parameters", "resid".
ids	Optional subset of subject ids for type = "individual".
at	Optional named list conditioning the population curve on factor levels / covariate values (e.g. list(group = "A")), passed to the same reference-grid machinery as get_dd_comparisons() .
n_points	Number of delay points in the curve grid.
x_trans	Delay-axis scale: "log10" (default) or "linear".
show_observed	Overlay the observed indifference points.
...	Unused.

Value

A [ggplot2::ggplot](#) object.

Examples

```
sim <- simulate_dd_ip(n_subjects = 12, seed = 1)
fit <- fit_dd_tmb(sim, equation = "mazur")
plot(fit)
plot(fit, type = "individual")
```

plot.prop_sc_output	<i>Plot Proportion of Guaranteed Choices by h Rank</i>
---------------------	--

Description

Plot Proportion of Guaranteed Choices by h Rank

Usage

```
## S3 method for class 'prop_sc_output'
plot(
  x,
  ...,
  pt_shape = 21,
  pt_fill = "white",
  pt_size = 3,
  title = "Proportion of guaranteed choices by h rank",
  xlab = "h value rank",
  ylab = "Proportion of guaranteed choices"
)
```

Arguments

x	Output from the prop_sc function
...	Additional arguments passed to ggplot2::geom_point()
pt_shape	Shape of the points in the plot. Default is 21.
pt_fill	Fill color of the points in the plot. Default is "white".
pt_size	Size of the points in the plot. Default is 3.
title	Title of the plot.
xlab	Label for the x-axis. Default is "h value rank".
ylab	Label for the y-axis.

Value

A ggplot object.

Examples

```
plot(prop_sc(pdq))
```

plot.prop_ss_output	<i>Plot Proportion of SIR/SS Choices by k Value</i>
---------------------	---

Description

Plots the proportion of SIR/SS choices by k value using the output of the prop_ss function.

Usage

```
## S3 method for class 'prop_ss_output'
plot(
  x,
  ...,
  pt_shape = 21,
  pt_fill = "white",
  pt_size = 3,
  title = "Proportion of SIR/SS choices by k value",
  xlab = "k value rank",
  ylab = "Proportion of SS choices"
)
```

Arguments

x	Output from the prop_ss function
...	Additional arguments passed to ggplot2::geom_point()
pt_shape	Shape of the points in the plot. Default is 21.
pt_fill	Fill color of the points in the plot. Default is "white".
pt_size	Size of the points in the plot. Default is 3.
title	Title of the plot. Default is "Proportion of SIR/SS choices by k value".
xlab	Label for the x-axis. Default is "k value rank".
ylab	Label for the y-axis. Default is "Proportion of SS choices".

Value

A ggplot object.

Examples

```
plot(prop_ss(mcq27))
```

plot.score_mcq_output *Plot MCQ Scores*

Description

Boxplots the MCQ scores for different metrics (small_k, medium_k, large_k, geomean_k, overall_k). Log-transformed k values (trans = "log"/"ln") are detected and the y-axis label adjusted accordingly.

Usage

```
## S3 method for class 'score_mcq_output'
plot(x, ..., xlab = "Metric", alpha = 0.3)
```

Arguments

x	A data frame returned by the score_mcq function.
...	Additional arguments passed to methods.
xlab	Label for the x-axis. Default is "Metric".
alpha	Transparency of the points in the plot. Default is 0.3.

Value

A ggplot object showing the boxplot of MCQ scores.

Examples

```
dat21 <- data.frame(subjectid = 1:5, questionid = rep(1:21, 5), response = 1)
plot(score_mcq(dat21, items = 21))
```

```
plot.score_mcq27_output
```

Plot MCQ-27 Scores

Description

Boxplots the MCQ-27 scores for different metrics (small_k, medium_k, large_k, geomean_k, overall_k). Log-transformed k values (trans = "log"/"ln") are detected and the y-axis label adjusted accordingly.

Usage

```
## S3 method for class 'score_mcq27_output'
plot(x, ..., xlab = "Metric", alpha = 0.3)
```

Arguments

x	A data frame returned by the score_mcq27 function.
...	Additional arguments passed to methods.
xlab	Label for the x-axis. Default is "Metric".
alpha	Transparency of the points in the plot. Default is 0.3.

Value

A ggplot object showing the boxplot of MCQ-27 scores.

Examples

```
plot(score_mcq27(mcq27))
```

plot.score_pdq_output *Plot PDQ Scores*

Description

Boxplot of PDQ h metrics (block1_h, block2_h, block3_h, mean_h, geomean_h, and the pooled overall_h extension), handling the same log transformations as [plot.score_mcq_output\(\)](#).

Usage

```
## S3 method for class 'score_pdq_output'
plot(x, ..., xlab = "Metric", alpha = 0.3)
```

Arguments

x	A data frame returned by the score_pdq function.
...	Additional arguments passed to methods.
xlab	Label for the x-axis. Default is "Metric".
alpha	Transparency of the points in the plot. Default is 0.3.

Value

A ggplot object showing the boxplot of PDQ scores.

Examples

```
plot(score_pdq(pdq))
```

plot_dd *Plot Delay-Discounting Model*

Description

Plots the delay-discounting data with the fitted model overlaid.

Usage

```
plot_dd(
  fit_dd_object,
  xlabel = "Delay",
  ylabel = "Indifference Point",
  title = "",
  logx = TRUE
)
```

Arguments

fit_dd_object	A fitted delay-discounting model object of class "fit_dd", created by the fit_dd() function.
xlabel	A character string specifying the label for the x-axis. Default is "Delay".
ylabel	A character string specifying the label for the y-axis. Default is "Indifference Point".
title	A character string specifying the plot title. Default is "".
logx	Logical. If TRUE, the x-axis is log-transformed. Default is TRUE.

Value

A ggplot object representing the fitted model and data.

Examples

```
data <- data.frame(
  id = rep(1:2, each = 6),
  x = rep(c(1, 7, 30, 90, 180, 365), 2),
  y = c(0.9, 0.5, 0.3, 0.2, 0.1, 0.05, 0.85, 0.55, 0.35, 0.15, 0.1, 0.05)
)
fit <- fit_dd(data, equation = "mazur", method = "mean")
plot_dd(fit)
```

```
plot_qq.beezdiscounting
```

Random-effect normal QQ plots for discounting models

Description

QQ plot of the estimated subject random-effect deviates against a normal reference (the standard check on the Gaussian random-effects assumption). Methods exist for the TMB indifference-point ([fit_dd_tmb\(\)](#)) and choice ([fit_dd_choice\(\)](#)) models. Bayesian fits are intentionally excluded; use `brms::pp_check()` and MCMC diagnostics there instead. The plotted deviates are shrunken (empirical-Bayes) estimates, so the normal reference is approximate.

Usage

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

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

Arguments

object	A fitted beeزdiscounting_tmb or beeزdiscounting_choice model.
which	Optional character vector selecting random-effect terms to show.
...	Unused.

Value

A `ggplot2::ggplot` object.

power_discounting	Monte Carlo power analysis for between-subject discounting designs
-------------------	--

Description

Estimates statistical power to detect a between-subject difference in the discount rate (a shift δ_k on natural-log k) by simulation: each replicate (1) simulates a two-condition between-subject indifference-point dataset with `simulate_dd_ip()` under assumed population parameters plus the effect, (2) refits it with `fit_dd_tmb()`, and (3) tests the log- k condition contrast with a Wald test at level α , referred to a t distribution with df degrees of freedom (see the `df` argument). Power is the proportion of *usable* fits (converged, positive-definite Hessian, finite standard error) that reject.

Because the power estimate is a proportion from finitely many replicates, it is reported with a Wilson score confidence interval (`power_mc_ci`). Both a p -value verdict ($p < \alpha$) and a confidence-interval verdict (Wald CI excludes 0) are recorded per replicate; they use the same standard error and reference distribution, so they coincide by construction, and both rates are returned.

This mirrors `beezdemand::power_demand()`. The design asymmetry is intentional: this package's simulator models a *between-subject* condition (subjects are split across conditions round-robin), while `beezdemand`'s models a within-subject condition.

Usage

```
power_discounting(
  n_subjects,
  effect = list(delta_k = NULL),
  design = list(),
  n_sim = 500,
  alpha = 0.05,
  df = NULL,
  seed = NULL,
  equation = c("mazur", "exponential"),
  family = c("sltb", "gaussian"),
  random_effects = k ~ 1,
  multi_start = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

<code>n_subjects</code>	Total number of simulated subjects per replicate, split across the two conditions round-robin (even numbers give equal groups).
-------------------------	---

effect	Named list supplying delta_k: the true condition-2 shift on natural-log k. 0 is allowed (useful for Type I error checks). E.g. delta_k = log(2) means condition 2's k is twice condition 1's.
design	Named list of data-generating settings, merged over the simulator defaults: delays (vector), log_k_pop, sigma_u (subject SD on log k), phi (SLT-beta precision; used when family = "sltb"), and sigma_e (residual SD; used when family = "gaussian").
n_sim	Number of Monte Carlo replicates. 500 (default) is suitable for interactive exploration; use 2000+ for grant-quality precision (see vignette("power-analysis")).
alpha	Nominal two-sided test level.
df	Degrees of freedom for the Wald test's t reference distribution. NULL (default) uses n_subjects - 2, the two-sample df of the between-subject design. This is an <i>empirically calibrated</i> small-sample correction rather than a model-derived df (the TMB fit has no exact t sampling theory); it passes the package's Type I calibration battery, while the asymptotic z-test (df = Inf) is anticonservative at study-relevant sample sizes.
seed	Optional integer seed; identical seeds give identical results. The caller's RNG state is restored on exit.
equation	Discounting function used for BOTH simulation and refitting (they are always matched): "mazur" (default) or "exponential". The two-parameter Green-Myerson and Rachlin forms are out of scope in this version.
family	Observation family used for both simulation and refitting: "sltb" (default) or "gaussian".
random_effects	Random-effects formula passed to <code>fit_dd_tmb()</code> . The default $k \sim 1$ matches the simulator's data-generating process (a single subject random intercept on log k). The simulator ALWAYS generates only that intercept: any richer formula (e.g. $k + \phi \sim 1$, $k + s \sim 1$) is accepted but produces a deliberately over-specified refit of data with no such variance component. That refit is useful for probing robustness rather than for estimating power under those random effects (out of scope in this version).
multi_start	Passed to <code>fit_dd_tmb()</code> . Defaults to FALSE for speed; non-convergent replicates are excluded and surfaced rather than biasing the estimate.
verbose	Logical; show a progress bar.
...	Additional arguments passed to <code>fit_dd_tmb()</code> (e.g. <code>tmb_control</code>).

Details

A replicate whose fit fails (non-convergence, non-positive-definite Hessian, unusable standard error, or an error) is excluded from the power denominator and reported through the `n_*` counts and `$replicates$status`. It is never counted as "no effect detected", which would bias power in an unpredictable direction. A warning is issued when fewer than 95% of replicates are usable.

The v1 scope is a single fixed-effect delta_k under the package's existing between-subject simulator. Effects on s or phi, power for derived measures (ED50, AUC), and arbitrary designs are out of scope; see vignette("power-analysis").

Value

An object of class `beezdiscounting_power`: a list with

power Estimated power: proportion of usable replicates whose Wald CI excludes 0 (equal to `hit_rate_ci`). NA if no replicate was usable.

power_mc_ci Wilson 95% confidence interval on power, reflecting Monte Carlo uncertainty from `n_used` replicates.

hit_rate_p Proportion of usable replicates with $p < \alpha$.

hit_rate_ci Proportion of usable replicates whose Wald CI excludes 0 (the same decision rule as `hit_rate_p`, since both use the same SE and t reference; both reported).

n_sim Total replicates attempted.

n_converged Replicates whose fit converged.

n_hessian_pd Replicates with a positive-definite Hessian.

n_used Replicates entering the power denominator (converged, positive-definite Hessian, finite SE).

alpha Nominal test level.

df Degrees of freedom of the t reference distribution actually used (`n_subjects - 2` unless overridden).

effect The validated effect specification (name and delta).

target_term The tested coefficient ("k:conditionC2").

design The merged design list actually used.

n_subjects As supplied.

replicates Tibble with one row per replicate: `sim`, `status` ("ok", "nonconverged", "hessian_not_pd", "se_unusable", "error"), `converged`, `hessian_pd`, `estimate`, `se`, `statistic`, `p_value`, `ci_lower`, `ci_upper`, `hit_p`, `hit_ci`, and `message` (error text, if any). Estimates are on the natural-log-k scale of the simulated `delta_k`.

seed As supplied.

settings List of equation, family, `multi_start`, and the deparsed random-effects formula.

call The matched call.

See Also

[find_n_discounting\(\)](#) to search for the smallest adequate sample size; [fit_dd_tmb\(\)](#) for the model being refit; [simulate_dd_ip\(\)](#) for the data-generating process.

Other power-analysis: [find_n_discounting\(\)](#)

Examples

```
# Quick exploratory run (use n_sim >= 500 for real planning)
res <- power_discounting(
  n_subjects = 8,
  effect = list(delta_k = log(2)),
  n_sim = 4, seed = 1, verbose = FALSE
)
```

```
print(res)
```

```
predict.beezdiscounting_brms
```

Predict from a beezdiscounting_brms model

Description

type = "response" summarizes brms::posterior_epred() draws (the indifference-proportion mean) to the posterior median with equal-tailed intervals; level = "subject" conditions on the subject random effects, "population" sets them to zero. type = "parameters" returns the per-subject k summaries.

Usage

```
## S3 method for class 'beezdiscounting_brms'
predict(
  object,
  newdata = NULL,
  type = c("response", "parameters"),
  level = "subject",
  probs = c(0.025, 0.975),
  draws = FALSE,
  ...
)
```

Arguments

object	A beezdiscounting_brms object.
newdata	Optional data frame (defaults to the model frame). The Rachlin guard columns (xzero/xsafe) are derived automatically.
type	"response" or "parameters".
level	"subject" (default) or "population".
probs	Interval probabilities.
draws	If TRUE, return the epred draws matrix.
...	Unused.

Value

A tibble (or draws matrix).

predict.beezdiscounting_choice

Predict from a structural choice discounting model

Description

Predict from a structural choice discounting model

Usage

```
## S3 method for class 'beeZdiscounting_choice'
predict(
  object,
  newdata = NULL,
  type = c("prob", "parameters"),
  level = "subject",
  ...
)
```

Arguments

object	A beeZdiscounting_choice object.
newdata	Optional data frame. NULL uses the fitting data. A supplied newdata must use the package's canonical column names (id, ss_amount, ll_amount, delay, plus any factor/covariate columns).
type	"prob" (default; fitted choice probabilities $P(LL)$ on $(0, 1)$) or "parameters" (the per-subject parameter tibble).
level	For type = "prob": "subject" (default; conditions on each subject's estimated random intercept, requires the id column) or "population" (random effects set to zero, i.e., the population-mean curve; no id column needed).
...	Unused.

Value

- type = "parameters": the subject-parameter tibble (one row per subject).
- type = "prob": newdata as a tibble plus a .prob column of fitted choice probabilities.

```
predict.beezdiscounting_choice_brms
```

Predict P(LL) from a beezdiscounting_choice_brms model

Description

Predict P(LL) from a beezdiscounting_choice_brms model

Usage

```
## S3 method for class 'beezdiscounting_choice_brms'
predict(
  object,
  newdata = NULL,
  type = c("response", "parameters"),
  level = "subject",
  probs = c(0.025, 0.975),
  draws = FALSE,
  ...
)
```

Arguments

object	A beezdiscounting_choice_brms object.
newdata	Optional trial-level data (needs rel and delay, or ss/ll columns from which rel is derived).
type	"response" (P(choosing the larger-later)) or "parameters" (per-subject k summaries).
level	"subject" or "population".
probs	Interval probabilities.
draws	Return the draws matrix instead.
...	Unused.

Value

A tibble (or draws matrix).

predict.beezdiscounting_tmb

Predict from a TMB mixed-effects discounting model

Description

Predict from a TMB mixed-effects discounting model

Usage

```
## S3 method for class 'beeZdiscounting_tmb'
predict(
  object,
  newdata = NULL,
  type = c("response", "parameters"),
  level = "subject",
  ...
)
```

Arguments

object	A beeZdiscounting_tmb object.
newdata	Optional data frame. NULL uses the fitting data. A supplied newdata must use the package's canonical column names (id, x, y), regardless of the id_var/x_var/y_var you gave at fit time.
type	"response" (fitted indifference proportions on $(0,1)$) or "parameters" (the per-subject parameter tibble).
level	For type = "response": "subject" (default; conditions on each subject's estimated random intercept, requires the id column) and/or "population" (random effects set to zero, i.e., the population-mean curve; no id column needed). Pass c("population", "subject") for both columns side-by-side. A numeric nlme-style level is rejected with an error. For an s-target 2-RE fit ($k + s \sim 1$), the subject level uses each subject's estimated s_i ; the population level (random effects zero) uses the soft-clamped population $s = .dd_soft_clamp_s_log(log_s)$, matching the kernel near the $(0.05, 20)$ bounds (equal to $\exp(log_s)$ in the interior).
...	Unused.

Value

- type = "parameters": the subject-parameter tibble (one row per subject).
- type = "response", level = "subject" only: newdata as a tibble plus a .fitted column.
- type = "response", "population" requested: newdata plus predict.fixed (and predict.id when "subject" is also requested), matching the nlme::predict.lme(level = 0:1) column-name convention so nlme-based plotting code runs unchanged.

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, equation = "mazur", family = "sltb",
  random_effects = k ~ 1, verbose = 0)

# Subject-conditional fitted values
head(predict(fit, type = "response"))

# Population-mean curve at specific delays (no id column needed)
nd <- data.frame(x = c(7, 30, 180, 365))
predict(fit, newdata = nd, level = "population")

# Per-subject parameters
head(predict(fit, type = "parameters"))
```

```
print.beezdiscounting_choice
      Print a structural choice discounting fit
```

Description

Print a structural choice discounting fit

Usage

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

Arguments

x	A beezdiscounting_choice object.
...	Unused.

Value

Invisibly returns x.

```
print.beezdiscounting_tmb
```

Print a TMB mixed-effects discounting fit

Description

Print a TMB mixed-effects discounting fit

Usage

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

Arguments

x	A beezdiscounting_tmb object.
...	Unused.

Value

Invisibly returns x.

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)  
fit <- fit_dd_tmb(dd, equation = "mazur", family = "sltb",  
                 random_effects = k ~ 1, verbose = 0)  
print(fit)
```

```
print.summary.beezdiscounting_choice
```

Print a structural choice discounting model summary

Description

Print a structural choice discounting model summary

Usage

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

Arguments

x	A summary.beezdiscounting_choice object.
digits	Number of significant digits for rounding.
...	Unused.

Value

Invisibly returns x.

```
print.summary.beezdiscounting_tmb
```

Print a TMB discounting model summary

Description

Print a TMB discounting model summary

Usage

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

Arguments

x	A summary.beezdiscounting_tmb object.
digits	Number of significant digits for rounding.
...	Unused.

Value

Invisibly returns x.

```
prop_sc
```

Calculate proportion of guaranteed choices at each PDQ h rank

Description

The PDQ analog of `prop_ss()`: pools all subjects' responses and reports, for each of the 10 h ranks (one item per block per rank), the proportion choosing the smaller guaranteed reward. 1 - prop_sc is the risky-choice proportion at that rank.

Usage

```
prop_sc(dat)
```

Arguments

`dat` Dataframe (longform) with subjectid, questionid (1-30), and response (0 for the guaranteed reward and 1 for the risky reward)

Value

Dataframe with proportion of guaranteed choices at each h rank

Examples

```
prop_sc(pdq)
```

prop_ss	<i>Calculate proportion of SIR/SS responses at each k value</i>
---------	---

Description

Calculate proportion of SIR/SS responses at each k value

Usage

```
prop_ss(dat, items = 27)
```

Arguments

`dat` Dataframe (longform) with subjectid, questionid, and response (0 for SIR/SS and 1 for LDR/LL)

`items` Number of MCQ items (27 or 21)

Details

`items` must match the instrument actually administered. Question ids 1-21 are valid in both the 21- and 27-item designs, so passing the wrong `items` does not error; it silently pools responses into the wrong k-rank rows. If the observed question ids do not exactly match the requested design, `prop_ss()` warns.

Value

Dataframe with proportion of SIR/SS responses at each k rank

Examples

```
prop_ss(mcq27)
dat21 <- data.frame(subjectid = 1, questionid = 1:21, response = 1)
prop_ss(dat21, items = 21)
```

```
ranef.beezdiscounting_choice
```

Extract subject-level random effects from a choice model

Description

Structural: id, the standardized random-intercept deviate u_i (such that $\log k_i = X \beta + \sigma_u * u_i$), and the resolved per-subject discount rate k . Descriptive (Young 2018): id plus the per-subject magnitude and delay random slopes (b_{mag} , b_{delay}); pooled (`random_slopes = FALSE`) returns an id-only frame.

Usage

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

Arguments

<code>object</code>	A beezdiscounting_choice object.
<code>...</code>	Unused.

Value

Data frame: structural id/ u_i / k ; descriptive id/ b_{mag} / b_{delay} (or id-only when pooled).

```
ranef.beezdiscounting_tmb
```

Extract subject-level random effects from a TMB discounting model

Description

Extract subject-level random effects from a TMB discounting model

Usage

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

Arguments

<code>object</code>	A beezdiscounting_tmb object.
<code>...</code>	Unused.

Value

Data frame keyed by id. For a 1-RE fit ($k \sim 1$): the standardized random-intercept deviate u_i (such that $\log k_i = X \beta + \sigma_u * u_i$) and the resolved per-subject discount rate k (no ϕ or s column; both are population-level). For a ϕ -target 2-RE fit ($k + \phi \sim 1$): (re_k , re_ϕ) offsets plus per-subject k and ϕ . For an s -target 2-RE fit ($k + s \sim 1$, GM/Rachlin only): (re_k , re_s) offsets plus per-subject k and s (soft-clamped toward $(0.05, 20)$).

```
residuals.beezdiscounting_choice
```

Residuals for a beezdiscounting_choice fit

Description

Residuals for a beezdiscounting_choice fit

Usage

```
## S3 method for class 'beezdiscounting_choice'
residuals(
  object,
  type = c("response", "pearson"),
  level = c("subject", "population"),
  ...
)
```

Arguments

<code>object</code>	A beezdiscounting_choice object.
<code>type</code>	"response" (default; raw choice - p) or "pearson" (divided by $\sqrt{p * (1 - p)}$).
<code>level</code>	"subject" (default) or "population".
<code>...</code>	Unused.

Value

Numeric vector of residuals, length `nobs(object)`.

```
residuals.beezdiscounting_tmb
```

Residuals for a beezdiscounting_tmb fit

Description

Residuals for a beezdiscounting_tmb fit

Usage

```
## S3 method for class 'beezdiscounting_tmb'
residuals(
  object,
  type = c("response", "pearson"),
  level = c("subject", "population"),
  ...
)
```

Arguments

object	A beezdiscounting_tmb object.
type	"response" (default; raw y - fitted) or "pearson" (divided by the per-row response SD; equivalent to .std_resid in augment()).
level	"subject" (default) or "population".
...	Unused.

Value

Numeric vector of residuals, length nobs(object).

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, verbose = 0)
head(residuals(fit))
head(residuals(fit, type = "pearson"))
```

results_dd

*Extract Results from Delay-Discounting Model***Description**

Extracts parameter estimates, fit statistics, and confidence intervals from a fitted delay-discounting model.

Usage

```
results_dd(fit_dd_object)
```

Arguments

`fit_dd_object` A fitted delay-discounting model object of class "fit_dd", created by the `fit_dd()` function.

Value

A tibble containing the following columns:

- `id`: The participant or group ID (if applicable).
- `term`: The model parameter (e.g., `k`).
- `estimate`: The estimated value of the parameter.
- `std.error`: The standard error of the parameter estimate.
- `statistic`: The t-statistic for the parameter estimate.
- `p.value`: The p-value for the parameter estimate.
- `conf_low`: The lower bound of the 95% confidence interval.
- `conf_high`: The upper bound of the 95% confidence interval.
- `R2`: The coefficient of determination (R^2).

Examples

```
data <- data.frame(
  id = rep(1:2, each = 6),
  x = rep(c(1, 7, 30, 90, 180, 365), 2),
  y = c(0.9, 0.5, 0.3, 0.2, 0.1, 0.05, 0.85, 0.55, 0.35, 0.15, 0.1, 0.05)
)
fit <- fit_dd(data, equation = "mazur", method = "two stage")
results_dd(fit)
```

score_dd	<i>Score 5.5 trial delay discounting from Qualtrics template</i>
----------	--

Description

Score 5.5 trial delay discounting from Qualtrics template

Usage

```
score_dd(df)
```

Arguments

df A dataframe containing all the columns

Details

Currently assumes the attending questions are present and labeled "Attend-LL" and "Attend-SS"

Value

A dataframe with id, indexes, response, k value, and effective delay 50.

Examples

```
score_dd(five.fivetrial_dd)
```

score_mcq	<i>Score an MCQ (27- or 21-item)</i>
-----------	--------------------------------------

Description

Score an MCQ (27- or 21-item)

Usage

```
score_mcq(
  dat = dat,
  items = 27,
  impute_method = "none",
  round = 6,
  random = FALSE,
  trans = "none",
  return_data = FALSE,
  verbose = FALSE
)
```

Arguments

<code>dat</code>	Dataframe (longform) with <code>subjectid</code> , <code>questionid</code> , and <code>response</code> (0 for SIR/SS and 1 for LDR/LL)
<code>items</code>	Number of MCQ items: 27 (Kirby, Petry, & Bickel, 1999) or 21 (Kirby & Marakovic, 1996). Default is 27.
<code>impute_method</code>	One of: "none", "ggm", "GGM", "inn", "INN"
<code>round</code>	Numeric specifying number of decimal places (passed to <code>base::round()</code>)
<code>random</code>	Boolean whether to insert a random draw (0 or 1) for NAs. Default is FALSE
<code>trans</code>	Transformation to apply to k values: "none", "log", or "ln". Default is "none"
<code>return_data</code>	Boolean whether to return the original data and new imputed responses. Default is FALSE.
<code>verbose</code>	Boolean whether to print subject and question ids pertaining to missing data. Default is FALSE.

Details

Each subject's data must satisfy a strict contract: exactly one row per canonical question id (`items` of them; no duplicates, no unknown ids, none missing) and responses coded 0, 1, or NA (numeric, logical, or character/factor values that coerce to 0/1). Malformed input errors rather than silently mis-scoring. Contrast with `mcq_to_choice()`'s lenient, ragged contract, which accepts partial per-subject coverage.

Value

If `return_data = FALSE` (default), a summary data frame with one row per subject. If `return_data = TRUE`, a list with `results` (that summary data frame) and `data` (the input data plus a `newresponse` column reflecting any imputation).

Examples

```
score_mcq(mcq27, items = 27)
dat21 <- data.frame(subjectid = 1, questionid = 1:21, response = 1)
score_mcq(dat21, items = 21)
```

score_mcq27

Score 27-item MCQ

Description

Score 27-item MCQ

Usage

```
score_mcq27(
  dat = dat,
  impute_method = "none",
  round = 6,
  random = FALSE,
  trans = "none",
  return_data = FALSE,
  verbose = FALSE
)
```

Arguments

<code>dat</code>	Dataframe (longform) with subjectid, questionid, and response (0 for SIR/SS and 1 for LDR/LL)
<code>impute_method</code>	One of: "none", "ggm", "GGM", "inn", "INN"
<code>round</code>	Numeric specifying number of decimal places (passed to <code>base::round()</code>)
<code>random</code>	Boolean whether to insert a random draw (0 or 1) for NAs. Default is FALSE
<code>trans</code>	Transformation to apply to k values: "none", "log", or "ln". Default is "none"
<code>return_data</code>	Boolean whether to return the original data and new imputed responses. Default is FALSE.
<code>verbose</code>	Boolean whether to print subject and question ids pertaining to missing data. Default is FALSE.

Details

The subject's data must satisfy a strict contract: exactly one row per canonical question id (27 of them; no duplicates, no unknown ids, none missing) and responses coded 0, 1, or NA (numeric, logical, or character/factor values that coerce to 0/1). Malformed input errors rather than silently mis-scoring. Contrast with `mcq27_to_choice()`'s lenient, ragged contract, which accepts partial per-subject coverage.

Value

If `return_data = FALSE` (default), a summary data frame with one row per subject. If `return_data = TRUE`, a list with `results` (that summary data frame) and `data` (the input data plus a new response column reflecting any imputation).

Examples

```
score_mcq27(mcq27)
```

score_pd	<i>Score 5.5 trial probability discounting from Qualtrics template</i>
----------	--

Description

Score 5.5 trial probability discounting from Qualtrics template

Usage

```
score_pd(df)
```

Arguments

df A dataframe containing all the columns

Details

Currently assumes the attending questions are present and labeled "Attend-LL" and "Attend-SS"

Value

A dataframe with id, indexes, response, h value, and effective probability 50.

Examples

```
score_pd(five.fivetrial_pd)
```

score_pdq	<i>Score the 30-item Probability Discounting Questionnaire (PDQ)</i>
-----------	--

Description

Scores the Madden, Petry, & Johnson (2009) PDQ: three 10-item blocks (block 1: \$20 for sure vs. a chance of \$80; block 2: \$40 vs. \$100; block 3: \$40 vs. \$60), each an ascending ladder of h values at indifference under the hyperbolic odds model $V = A / (1 + h * \theta)$, $\theta = (1 - p) / p$ (Rachlin, Raineri, & Cross, 1991). Each block is scored independently by the same consistency-maximization algorithm as the MCQ scorers; this implementation reproduces the Gray et al. (2016) scoring syntax lookup tables for every possible response pattern.

Usage

```
score_pdq(
  dat = dat,
  impute_method = "none",
  round = 6,
  random = FALSE,
  trans = "none",
  return_data = FALSE,
  verbose = FALSE
)
```

Arguments

<code>dat</code>	Dataframe (longform) with subjectid, questionid (1-30), and response (0 for the smaller guaranteed reward, 1 for the larger risky reward)
<code>impute_method</code>	One of: "none", "ggm", "GGM", "inn", "INN"
<code>round</code>	Numeric specifying number of decimal places (passed to <code>base::round()</code>)
<code>random</code>	Boolean whether to insert a random draw (0 or 1) for NAs. Default is FALSE
<code>trans</code>	Transformation to apply to h values: "none", "log", or "ln". Default is "none"
<code>return_data</code>	Boolean whether to return the original data and new imputed responses. Default is FALSE.
<code>verbose</code>	Boolean whether to print subject and question ids pertaining to missing data. Default is FALSE.

Details

Each subject's data must satisfy a strict contract: exactly one row per canonical question id (30 of them; no duplicates, no unknown ids, none missing) and responses coded 0, 1, or NA (numeric, logical, or character/factor values that coerce to 0/1). Malformed input errors rather than silently mis-scoring. Contrast with [pdq_to_choice\(\)](#)'s lenient, ragged contract.

The published scoring (Madden et al., 2009; Gray et al., 2016) has no overall 30-item ladder; blocks are scored separately, and `mean_h` is Gray et al.'s recommended composite. `overall_h` and `overall_consistency` are a beezdiscounting extension: all 30 items are pooled into a single ascending ladder (exact-rational h order, ties broken by question id; four item pairs tie exactly) and scored by the same consistency-maximization algorithm with the repeat-last edge.

INN imputation groups items sharing an h rank (one item per block). Ladders with remaining NA responses score NA; Gray et al. recommend excluding subjects below 80% consistency on any block.

Value

If `return_data = FALSE` (default), a summary data frame with one row per subject: pooled `overall_h` (see Details), per-block h (`block1_h`, `block2_h`, `block3_h`), their arithmetic mean (`mean_h`; Gray et al.'s recommended composite) and geometric mean (`geomean_h`), pooled and per-block consistency plus their mean (`composite_consistency`), and pooled plus per-block proportions of risky choices (`block*_proportion` is Gray et al.'s risky choice ratio). If `return_data = TRUE`, a list with results and data (the input plus a `newresponse` column reflecting any imputation).

References

- Madden, G. J., Petry, N. M., & Johnson, P. S. (2009). Pathological gamblers discount probabilistic rewards less steeply than matched controls. *Experimental and Clinical Psychopharmacology*, 17(5), 283-290. doi:10.1037/a0016806
- Gray, J. C., Amlung, M. T., Palmer, A. A., & MacKillop, J. (2016). Syntax for calculation of discounting indices from the monetary choice questionnaire and probability discounting questionnaire. *Journal of the Experimental Analysis of Behavior*, 106(2), 156-163. doi:10.1002/jeab.221

Examples

```
score_pdq(pdq)
```

simulate_dd_choice	<i>Simulate trial-level SS-vs-LL choice data</i>
--------------------	--

Description

Generates binary choices from either the structural or descriptive choice model. When mode = "structural" (default) choices are drawn from the structural model: $P(LL) = \text{plogis}(\beta_0 + \gamma * ((11/ss) * D(k, \text{delay}) - 1))$, $k_i = \exp(\log_k\text{pop} + \sigma_u * u_i)$. When mode = "descriptive" choices follow Young's (2018) correlated random-slope logistic model: each subject receives per-subject slopes drawn from a bivariate normal with mean theta and covariance Sigma, and $P(LL) = \text{plogis}((\theta_1 + b_i[1]) * \log(11/ss) + (\theta_2 + b_i[2]) * \log(\text{delay} + 1))$. The descriptive branch is the known-truth generator used by recovery tests.

Usage

```
simulate_dd_choice(
  n_subjects = 50,
  ss_amount = c(40, 55, 31, 14, 47, 25, 78, 40, 11, 67),
  ll_amount = c(65, 75, 85, 25, 60, 30, 80, 65, 30, 75),
  delay = c(25, 61, 14, 19, 160, 7, 4, 62, 7, 119),
  mode = c("structural", "descriptive"),
  log_k_pop = log(0.02),
  sigma_u = 0.5,
  gamma = 4,
  beta0 = 0,
  equation = c("mazur", "exponential"),
  theta = c(1.5, -0.4),
  re_sd = c(0.5, 0.3),
  re_cor = -0.2,
  return_truth = FALSE,
  seed = NULL
)
```

Arguments

n_subjects	Integer number of subjects.
ss_amount, ll_amount, delay	Numeric trial-design vectors (same length); each subject is presented all trials. Defaults are a built-in grid.
mode	"structural" (default) or "descriptive". Selects the generative model.
log_k_pop, sigma_u, gamma, beta0	Structural truth (used only when mode = "structural").
equation	"mazur" or "exponential" (structural mode only).
theta	Length-2 numeric vector of population-level slopes for the descriptive model: theta[1] on log(ll/ss), theta[2] on log(delay + 1).
re_sd	Length-2 numeric vector of random-effect standard deviations (descriptive mode only).
re_cor	Correlation between the two random slopes (descriptive mode only).
return_truth	Logical. If TRUE and mode = "descriptive", the returned tibble carries two attributes: subject_slopes (n_subjects x 2 matrix of realized b_i values) and Sigma (the 2x2 covariance matrix).
seed	Optional integer seed for reproducibility. The caller's RNG state is restored on exit.

Value

A [tibble](#) with columns id, ss_amount, ll_amount, delay, choice (0/1, 1 = LL chosen). When mode = "descriptive" and return_truth = TRUE, the tibble also carries subject_slopes and Sigma attributes.

Examples

```
# Structural (default)
sim <- simulate_dd_choice(n_subjects = 20, seed = 1)
head(sim)

# Descriptive (Young 2018 correlated random-slope model)
sim_desc <- simulate_dd_choice(n_subjects = 20, mode = "descriptive",
                              theta = c(1.5, -0.4), seed = 42)
head(sim_desc)
```

Description

Generates long-format indifference-point data from the mixed-effects discounting model used by `fit_dd_tmb()`. Each subject i has a random discount rate $\log k_i = \log k_{\text{pop}} + \delta_k[\text{condition}] + u_i$, $u_i \sim N(0, \sigma_u^2)$. The mean indifference proportion at delay x is the discounting function μ (Mazur hyperbola, exponential, Green-Myerson, or Rachlin), and observed y is drawn from the scale-location-truncated beta ($\text{family} = \text{"sltb"}$) via the inverse-CDF on the truncated beta, or from a clamped Gaussian ($\text{family} = \text{"gaussian"}$).

Usage

```
simulate_dd_ip(
  n_subjects = 60,
  delays = c(7, 30, 180, 365, 730, 1460, 2920),
  log_k_pop = log(0.01),
  sigma_u = 0.6,
  phi = 10,
  sigma_e = 0.1,
  family = c("sltb", "gaussian"),
  equation = c("mazur", "exponential", "green-myerson", "rachlin"),
  s = 1,
  n_conditions = 1,
  delta_k = NULL,
  seed = NULL,
  sigma_phi = 0,
  rho_kphi = 0,
  attach_truth = FALSE,
  sigma_s = 0,
  rho_ks = 0
)
```

Arguments

<code>n_subjects</code>	Integer; number of subjects.
<code>delays</code>	Numeric vector of delays (days) each subject is observed at.
<code>log_k_pop</code>	Numeric; population intercept on $\log k$.
<code>sigma_u</code>	Numeric; SD of the subject random intercept on $\log k$.
<code>phi</code>	Numeric; SLT-beta precision ($\text{family} = \text{"sltb"}$).
<code>sigma_e</code>	Numeric; residual SD on y ($\text{family} = \text{"gaussian"}$).
<code>family</code>	One of <code>"sltb"</code> (default) or <code>"gaussian"</code> .
<code>equation</code>	One of <code>"mazur"</code> (default), <code>"exponential"</code> , <code>"green-myerson"</code> , or <code>"rachlin"</code> . The two 2-parameter forms use the nonlinearity exponent s and reduce to <code>"mazur"</code> at $s = 1$.
<code>s</code>	Numeric nonlinearity exponent for the 2-parameter equations (Green-Myerson / Rachlin); ignored by <code>"mazur"</code> / <code>"exponential"</code> .
<code>n_conditions</code>	Integer; number of between-subject condition levels. When > 1 , a condition factor is added and subjects are split across levels.

delta_k	Numeric vector of length n_conditions; per-condition shift on log k (the first element is typically 0 for the reference level). Required (non-NULL) when n_conditions > 1.
seed	Optional integer seed; identical seeds give identical results. The caller's RNG state is restored on exit.
sigma_phi	Numeric; SD of the subject random intercept on log phi (SLT-beta precision). Default 0 (no subject-random phi, family = "sltb" only). When > 0, the (log k, log phi) pair is drawn jointly from $N(0, \text{Sigma})$ where Sigma is parameterized by sigma_u, sigma_phi, and rho_kphi. Requires family = "sltb".
rho_kphi	Numeric in [-1, 1]; correlation between subject random intercepts on log k and log phi. Ignored when sigma_phi = 0.
attach_truth	Logical; when TRUE, a phi column is always appended (equal to the constant phi when sigma_phi = 0, or subject-specific when sigma_phi > 0); when sigma_s > 0, an s column is also appended with the subject-specific curvature exponent. Default FALSE.
sigma_s	Numeric; SD of the subject random intercept on log s (GM/Rachlin curvature exponent). Default 0 (no subject-random s). When > 0, the (log k, log s) pair is drawn jointly from $N(0, \text{Sigma})$ where Sigma is parameterized by sigma_u, sigma_s, and rho_ks. Requires equation = "green-myerson" or "rachlin". Mutually exclusive with sigma_phi > 0 (q = 2 only).
rho_ks	Numeric in [-1, 1]; correlation between subject random intercepts on log k and log s. Ignored when sigma_s = 0.

Details

The SLT draw uses the same constants as the C++ template and the verified reference density: $s_slt = 1.0000001$, $l = 1e-8$, with $a = \mu * \phi$, $b = (1 - \mu) * \phi$, and $y = (q\beta(U, a, b) - 1) * s_slt$ where $U \sim \text{Uniform}(p\beta(1, a, b), p\beta(1/s_slt + 1, a, b))$.

Value

A [tibble](#) with columns id (factor), condition (factor; only when n_conditions > 1), x (delay), and y (indifference proportion in [0, 1]). When attach_truth = TRUE, an additional phi column is appended with the subject-specific SLT-beta precision (equal to the constant phi when sigma_phi = 0). When attach_truth = TRUE and sigma_s > 0, an additional s column is appended with the subject-specific curvature exponent.

Examples

```
# A small SLT-beta mixed-effects discounting dataset (id, x, y)
sim <- simulate_dd_ip(n_subjects = 8, seed = 1)
head(sim)

# Two between-subject conditions with a log-k shift on the second level
sim2 <- simulate_dd_ip(
  n_subjects = 8, n_conditions = 2, delta_k = c(0, log(3)), seed = 2
)
```

simulate_dd_linear	<i>Simulate indifference points from the linearized Mazur random-effects model</i>
--------------------	--

Description

Draws $\theta_{ic} \sim N(\mu_c, g \sigma^2 / T)$, $y_{ijc} \sim N(\theta_{ic}, \sigma^2)$ and returns $D = 1 / (1 + \exp(y) t)$ (Hinds et al., 2026, Sec. 4.1), which is always strictly inside (0, 1).

Usage

```
simulate_dd_linear(
  n_subjects,
  delays,
  mu,
  sigma2,
  g,
  seed = NULL,
  attach_truth = FALSE
)
```

Arguments

n_subjects	Integer, scalar (recycled) or one per condition.
delays	Positive numeric vector of delays ($T = \text{length}(\text{delays})$).
mu	Numeric vector of population mean $\ln k$ per condition; names become condition labels.
sigma2	Measurement-error variance on the transformed scale.
g	Random-effect variance multiplier: $\text{Var}(\theta) = g * \sigma^2 / T$.
seed	Optional seed; the global RNG state is restored afterwards.
attach_truth	If TRUE, attach attr(, "truth").

Value

Long data.frame(id, condition, x, y).

Examples

```
s <- simulate_dd_linear(30, c(7, 30, 180, 365), mu = c(A = -6, B = -5),
  sigma2 = 2, g = 10, seed = 1)
anova(fit_dd_linear(s, factors = "condition"))
```

summarize_mcq	<i>Provide a summary of the results from the MCQ output table.</i>
---------------	--

Description

Provide a summary of the results from the MCQ output table.

Usage

```
summarize_mcq(res, na.rm = TRUE)
```

Arguments

res	Dataframe with MCQ results (output from score_mcq() or score_mcq27())
na.rm	Boolean whether to remove NAs from the calculation

Value

Dataframe with summary statistics

Examples

```
summarize_mcq(score_mcq27(mcq27))
```

summary.beezdiscounting_brms	<i>Summarize a beezdiscounting_brms model</i>
------------------------------	---

Description

Summarize a beezdiscounting_brms model

Usage

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

Arguments

object	A beezdiscounting_brms object.
report_space	Reporting scale for the coefficient table.
...	Unused.

Value

A summary.beezdiscounting_brms list.

```
summary.beezdiscounting_choice
```

Summarize a structural choice discounting fit

Description

Summarize a structural choice discounting fit

Usage

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

Arguments

object	A beeZdiscounting_choice object.
report_space	Scale for the fixed-effect (beta_k) estimates and standard errors in the coefficient table: "natural" (default; k via exp()), "log10", "internal" or "log". statistic and p.value are always computed on the estimation scale regardless of report_space.
...	Unused.

Value

An object of class "summary.beeZdiscounting_choice" with components: call, model_class, backend, mode, equation, coefficients, variance_components, n_subjects, nobs, converged, logLik, AIC, BIC, notes.

```
summary.beeZdiscounting_choice_brms
```

Summarize a beeZdiscounting_choice_brms model

Description

Summarize a beeZdiscounting_choice_brms model

Usage

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

Arguments

object	A beeZdiscounting_choice_brms object.
report_space	Reporting scale.
...	Unused.

Value

A summary.beezdiscounting_choice_brms list.

```
summary.beezdiscounting_tmb
```

Summarize a TMB mixed-effects discounting fit

Description

Summarize a TMB mixed-effects discounting fit

Usage

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

Arguments

object	A beeZdiscounting_tmb object.
report_space	Scale for the fixed-effect (beta_k) estimates and standard errors in the coefficient table: "natural" (default; k on the natural scale via exp()), "log10" (log10-k), "internal" or "log" (log-k, the estimation scale; the two coincide for beta_k). statistic and p.value are always computed on the estimation (log-k) scale regardless of report_space.
...	Unused.

Value

An object of class "summary.beeZdiscounting_tmb" with components: call, model_class, backend, equation, family, coefficients, variance_components, n_subjects, nobS, converged, logLik, AIC, BIC, notes.

Examples

```
dd <- simulate_dd_ip(n_subjects = 20, seed = 1)
fit <- fit_dd_tmb(dd, equation = "mazur", family = "sltb",
  random_effects = k ~ 1, verbose = 0)
summary(fit)
summary(fit, report_space = "log10")
```

tidy.beezdiscounting_brms

Tidy a beezdiscounting_brms model

Description

The exact 8-column dd coefficient contract (term, estimate, std.error, statistic, p.value, component, estimate_scale, term_display). Estimates are posterior medians and std.error posterior SDs of the report-space-transformed draws (exact; no delta method); statistic/p.value are NA; use confint() for interval summaries. The shape row (s, two-parameter equations) carries component = "shape"; variance rows (effects = "ran_pars") mirror the TMB reporting convention (log10-scale k RE SD; natural phi/sigma).

Usage

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

Arguments

x	A beezdiscounting_brms object.
effects	"fixed", "ran_pars", or both (default).
report_space	"natural" (default), "log10", "internal", or "log".
...	Unused.

Value

A tibble.

tidy.beezdiscounting_choice

Tidy a beezdiscounting_choice model into a coefficient tibble

Description

Returns fixed-effect rows (the log-k coefficients), the shape rows (the choice-sensitivity gamma and, when present, the choice-bias beta0), and the variance-component row, following the broom coefficient-table contract.

Usage

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

Arguments

x	A beezdiscounting_choice object.
effects	Character vector: "fixed" (log-k fixed-effect rows + the shape rows gamma/beta0), "ran_pars" (the RE SD), or both (default).
report_space	Reporting scale for the fixed-effect estimate/std.error: "natural" (default), "log10", "internal", or "log".
...	Unused.

Details

estimate and std.error are reported on the report_space scale for the fixed-effect (beta_k) rows and for gamma (which is transformed EXPLICITLY since the param-space transformer keys only on k/s/phi). beta0 is on the identity (logit-intercept) scale and is NEVER transformed across report spaces. statistic and p.value are always computed on the estimation (internal) scale. Wald statistics are not recomputed after back-transforming (broom convention).

Value

A tibble with exactly 8 columns in this order: term, estimate, std.error, statistic, p.value, component, estimate_scale, term_display. Fixed-effect rows carry component == "fixed"; gamma/beta0 carry component == "shape"; variance rows carry component == "variance".

tidy.beezdiscounting_choice_brms

Tidy a beezdiscounting_choice_brms model

Description

The 8-column dd contract; gamma (choice sensitivity) and k rows are report-space transformed per draw; the optional beta0 bias stays on the logit scale.

Usage

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

Arguments

x	A beezdiscounting_choice_brms object.
effects	"fixed", "ran_pars", or both.
report_space	Reporting scale.
...	Unused.

Value

A tibble.

tidy.beezdiscounting_comparison

Tidy a discounting comparison into a flat contrasts frame

Description

`generics::tidy()` method for `beezdiscounting_comparison` objects (returned by `get_dd_comparisons()`). Produces a flat long tibble whose **column names and order** match the `beezdemand` and `beezdemand_comparison` tidier, so downstream consumers can bind rows from both backends into a single frame. Note that the contrast label dialect differs: `beezdiscounting` emits native factor=level labels (e.g. "condition=C1 - condition=C2"), while `beezdemand` emits bare level labels (e.g. "C1 - C2"). Fully uniform emmeans-style contrast labels across backends are a future cross-backend item. The nested object keeps the native dialect (see `get_dd_comparisons()`).

Usage

```
## S3 method for class 'beezdiscounting_comparison'
tidy(x, exponentiate = FALSE, ...)
```

Arguments

x	A beezdiscounting_comparison object.
exponentiate	Logical. If TRUE, return base-invariant ratios ($\text{estimate} = 10^{\text{estimate}}$, CIs back-transformed); std.error becomes NA following broom's convention for exponentiated fits. Default FALSE (log10-scale contrasts).
...	Unused.

Value

A tibble with columns `param`, `contrast`, `estimate`, `std.error`, `statistic`, `df`, `conf.low`, `conf.high`, `p.value` (with leading `by-column(s)` inserted before `param` when `contrast_by` is active). Estimates and CIs are on the log10 scale (or ratios when `exponentiate = TRUE`). `statistic` is an asymptotic z ($df = Inf$). On the brms backend, `statistic/df/p.value` are NA (posterior summaries), the intervals are quantile credible intervals, and an additional `post.prob` column reports the posterior probability of direction.

`tidy.beezdiscounting_tmb`

Tidy a beezdiscounting_tmb model into a coefficient tibble

Description

Returns fixed-effect rows (the log-k coefficients) and/or variance-component rows, following the broom coefficient-table contract.

Usage

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

Arguments

<code>x</code>	A <code>beezdiscounting_tmb</code> object.
<code>effects</code>	Character vector: "fixed" (log-k fixed-effect rows), "ran_pars" (the RE SD and the auxiliary precision/scale parameter), or both (default).
<code>report_space</code>	Reporting scale for fixed-effect estimate/std.error: "natural" (default), "log10", "internal", or "log" ("internal" and "log" coincide for the log-k beta_k rows).
<code>...</code>	Unused.

Details

`estimate` and `std.error` are reported on the `report_space` scale for the fixed-effect (`beta_k`) rows. `statistic` and `p.value` are always computed on the estimation (log-k) scale. Wald statistics are not recomputed after back-transforming (broom convention; see the `summary()` note for details). Variance-component rows carry NA for `statistic` and `p.value` and are not affected by `report_space`.

Value

A tibble with exactly 8 columns in this order: term, estimate, std.error, statistic, p.value, component, estimate_scale, term_display. Fixed-effect rows carry component == "fixed"; the shape-parameter row (2-parameter equations only) carries component == "shape"; variance rows carry component == "variance".

timing_dd	<i>Extract timing metrics from 5.5 trial delay discounting from Qualtrics template</i>
-----------	--

Description

Extract timing metrics from 5.5 trial delay discounting from Qualtrics template

Usage

```
timing_dd(df)
```

Arguments

df A dataframe containing all the columns

Details

Currently assumes the attending questions are present and labeled "Attend-LL" and "Attend-SS"

Value

A dataframe with ResponseId, indexes, values and timing

Examples

```
timing_dd(five.fivetrial_dd)
```

timing_pd	<i>Extract timing metrics from 5.5 trial probability discounting from Qualtrics template</i>
-----------	--

Description

Extract timing metrics from 5.5 trial probability discounting from Qualtrics template

Usage

```
timing_pd(df)
```

Arguments

df A dataframe containing all the columns

Details

Currently assumes the attending questions are present and labeled "Attend-LL" and "Attend-SS"

Value

A dataframe with ResponseId, indexes, values and timing

Examples

```
timing_pd(five.fivetrial_pd)
```

```
VarCorr.beezdiscounting_choice
```

Random-effect (co)variances for a beezdiscounting_choice fit

Description

Descriptive (Young 2018): the 2-RE slope SDs and their correlation (the per-subject magnitude/delay sensitivities); pooled (random_slopes = FALSE) returns a 0-row frame. Structural: the log-k random-intercept SD.

Usage

```
## S3 method for class 'beezdiscounting_choice'
VarCorr(x, sigma = 1, ...)
```

Arguments

x A beezdiscounting_choice object.
 sigma Ignored (present for nlme::VarCorr() generic compatibility).
 ... Unused.

Value

A data frame of variance components. Descriptive: Group, Term, Variance, StdDev, Corr (one row per random slope; 0 rows when pooled). Structural: Group, Term, StdDev, Scale.

VarCorr.beezdiscounting_tmb

Random-effect covariance for a TMB discounting model

Description

For a 2-RE fit returns the 2x2 covariance with StdDev and the joint correlation (a structural 0 for pdDiag; the first row's Corr is NA by convention). The Term column reflects the fitted pair: (k, phi) for a phi-target fit ($k + \phi \sim 1$) or (k, s) for an s-target fit ($k + s \sim 1$, GM/Rachlin only). For a 1-RE fit ($k \sim 1$) returns the single random-intercept SD on the log-k scale.

Usage

```
## S3 method for class 'beeZdiscounting_tmb'
VarCorr(x, sigma = 1, ...)
```

Arguments

x	A beeZdiscounting_tmb fit.
sigma	Ignored (present for the nlme::VarCorr() generic).
...	Unused.

Value

A data frame with Group, Term, Variance, StdDev, and (2-RE) Corr.

wide_to_long_mcq *Reshape MCQ data wide to long*

Description

Reshape MCQ data wide to long

Usage

```
wide_to_long_mcq(dat, items = 27)
```

Arguments

dat	Wide format MCQ assuming subject id is in column 1
items	Number of MCQ questions

Value

Long format data frame

Examples

```
wide_to_long_mcq(long_to_wide_mcq(mcq27))
```

```
wide_to_long_mcq_excel
```

Reshape MCQ data from wide (as used in the 21- and 27-Item Monetary Choice Questionnaire Automated Scorer) to long

Description

Reshape MCQ data from wide (as used in the 21- and 27-Item Monetary Choice Questionnaire Automated Scorer) to long

Usage

```
wide_to_long_mcq_excel(dat)
```

Arguments

dat Wide format MCQ data as used in the Excel Automated Scorers

Value

Long format data frame

Examples

```
wide_to_long_mcq_excel(long_to_wide_mcq_excel(generate_data_mcq(2)))
```

Index

- * **datasets**
 - dd_ip, 19
 - five.fivetrial_dd, 37
 - five.fivetrial_pd, 37
 - mcq21, 47
 - mcq27, 48
 - pdq, 49
- * **power-analysis**
 - find_n_discounting, 22
 - power_discounting, 63
- anova.beezdiscounting_linear, 4
- anova.beezdiscounting_linear(), 9, 32, 33
- ans_dd, 5
- ans_pd, 6
- augment.beezdiscounting_choice, 6
- augment.beezdiscounting_linear
 - (beezdiscounting_linear-methods), 8
- augment.beezdiscounting_tmb, 7
- beezdiscounting_linear-methods, 5, 8, 33, 56
- brms::brm(), 27, 31
- brms::loo(), 27, 31
- calc_aucs, 10
- calc_conf_int, 11
- calc_dd, 11
- calc_pd, 12
- calc_r2, 13
- check_unsystematic, 13
- coef.beezdiscounting_choice, 14
- coef.beezdiscounting_linear
 - (beezdiscounting_linear-methods), 8
- coef.beezdiscounting_tmb, 15
- confint(), 55
- confint.beezdiscounting_brms, 16
- confint.beezdiscounting_choice, 16
- confint.beezdiscounting_choice_brms, 17
- confint.beezdiscounting_linear
 - (beezdiscounting_linear-methods), 8
- confint.beezdiscounting_tmb, 18
- dd_ip, 19
- default_dd_choice_priors, 20
- default_dd_choice_priors(), 32
- default_dd_priors, 21
- default_dd_priors(), 20, 27, 28, 31
- find_n_discounting, 22
- find_n_discounting(), 65
- fit_dd, 25
- fit_dd_brms, 26
- fit_dd_brms(), 51
- fit_dd_choice, 28
- fit_dd_choice(), 30–32, 46–49, 52, 62
- fit_dd_choice_brms, 30
- fit_dd_choice_brms(), 20, 53
- fit_dd_linear, 32
- fit_dd_linear(), 8, 9, 55, 56
- fit_dd_tmb, 33
- fit_dd_tmb(), 23, 24, 26–29, 31, 32, 56, 62–65, 85
- fitted.beezdiscounting_choice, 36
- fitted.beezdiscounting_tmb, 36
- five.fivetrial_dd, 37
- five.fivetrial_pd, 37
- fixef.beezdiscounting_choice, 38
- fixef.beezdiscounting_tmb, 38
- generate_data_mcq, 39
- generics::tidy(), 93
- get_dd_comparisons, 39
- get_dd_comparisons(), 29, 32, 54, 57, 93
- get_dd_param_emms, 41

get_dd_param_emms(), 29, 32, 40
 get_lookup_table, 42
 get_lookup_table(), 46–50
 ggplot2::ggplot, 51–54, 56, 57, 63
 glance.beezdiscounting_brms, 43
 glance.beezdiscounting_choice, 43
 glance.beezdiscounting_choice_brms, 44
 glance.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8
 glance.beezdiscounting_tmb, 44

 logLik.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8
 long_to_wide_mcq, 45
 long_to_wide_mcq_excel, 45

 mcq21, 47
 mcq27, 48
 mcq27_to_choice, 48
 mcq27_to_choice(), 47, 80
 mcq_to_choice, 46
 mcq_to_choice(), 49, 50, 79

 nlme::ranef(), 29
 nlme::VarCorr(), 29
 nobs.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8

 pdq, 49
 pdq_to_choice, 50
 pdq_to_choice(), 82
 plot.beezdiscounting_brms, 51
 plot.beezdiscounting_choice, 52
 plot.beezdiscounting_choice_brms, 53
 plot.beezdiscounting_comparison, 54
 plot.beezdiscounting_linear, 54
 plot.beezdiscounting_linear(), 9, 33
 plot.beezdiscounting_tmb, 56
 plot.beezdiscounting_tmb(), 55
 plot.prop_sc_output, 57
 plot.prop_ss_output, 58
 plot.score_mcq27_output, 60
 plot.score_mcq_output, 59
 plot.score_mcq_output(), 61
 plot.score_pdq_output, 61
 plot_dd, 61

 plot_qq.beezdiscounting, 62
 plot_qq.beezdiscounting_choice
 (plot_qq.beezdiscounting), 62
 plot_qq.beezdiscounting_tmb
 (plot_qq.beezdiscounting), 62
 power_discounting, 63
 power_discounting(), 22–24
 predict.beezdiscounting_brms, 66
 predict.beezdiscounting_choice, 67
 predict.beezdiscounting_choice_brms,
 68
 predict.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8
 predict.beezdiscounting_tmb, 69
 print.beezdiscounting_choice, 70
 print.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8
 print.beezdiscounting_tmb, 71
 print.summary.beezdiscounting_choice,
 71
 print.summary.beezdiscounting_linear
 (beezdiscounting_linear-methods),
 8
 print.summary.beezdiscounting_tmb, 72
 prop_sc, 72
 prop_ss, 73
 prop_ss(), 72

 ranef.beezdiscounting_choice, 74
 ranef.beezdiscounting_tmb, 74
 residuals.beezdiscounting_choice, 75
 residuals.beezdiscounting_tmb, 76
 results_dd, 77

 score_dd, 78
 score_mcq, 78
 score_mcq(), 47, 49
 score_mcq27, 79
 score_mcq27(), 49
 score_pdq, 81
 score_pdq, 81
 score_pdq(), 42, 50
 simulate_dd_choice, 83
 simulate_dd_choice(), 29
 simulate_dd_ip, 84
 simulate_dd_ip(), 63, 65
 simulate_dd_linear, 87

`simulate_dd_linear()`, 33
`stats::confint()`, 9
`summarize_mcq`, 88
`summary.beezdiscounting_brms`, 88
`summary.beezdiscounting_choice`, 89
`summary.beezdiscounting_choice_brms`,
89
`summary.beezdiscounting_linear`
(`beeزdiscounting_linear-methods`),
8
`summary.beeزdiscounting_tmb`, 90

`tibble`, 47, 49, 50, 84, 86
`tidy.beeزdiscounting_brms`, 91
`tidy.beeزdiscounting_choice`, 91
`tidy.beeزdiscounting_choice_brms`, 92
`tidy.beeزdiscounting_comparison`, 93
`tidy.beeزdiscounting_comparison()`, 39,
41
`tidy.beeزdiscounting_linear`
(`beeزdiscounting_linear-methods`),
8
`tidy.beeزdiscounting_tmb`, 94
`timing_dd`, 95
`timing_pd`, 95

`VarCorr.beeزdiscounting_choice`, 96
`VarCorr.beeزdiscounting_tmb`, 97

`wide_to_long_mcq`, 97
`wide_to_long_mcq_excel`, 98