Package 'hmetad'

August 21, 2026

Title Fit the Meta-D' Model of Confidence Ratings Using 'brms'

Version 0.2.0

Description Implementation of Bayesian regressions over the meta-d' model of psychological data from two alternative forced choice tasks with ordinal confidence ratings. For more information, see Maniscalco & Lau (2012) <doi:10.1016/j.concog.2011.09.021>. The package is a front-end to the 'brms' package, which facilitates a wide range of regression designs, as well as tools for efficiently extracting posterior estimates, plotting, and significance testing.

License GPL (>= 3)

Depends R (>= 4.2.0), brms (>= 2.23.0)

Imports abind (>= 1.4.8), dplyr (>= 1.2.0), glue (>= 1.8.0), posterior (>= 1.6.1), rlang (>= 1.2.0), stats, stringr (>= 1.6.0), tidybayes (>= 3.0.7), tidyr (>= 1.3.2)

Suggests colorspace, knitr, purrr (>= 1.2.1), rmarkdown, testthat (>= 3.0.0), tidyverse (>= 2.0.0)

VignetteBuilder knitr

Encoding UTF-8

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

URL https://metacoglab.github.io/hmetad/, https://github.com/metacoglab/hmetad

BugReports https://github.com/metacoglab/hmetad/issues

NeedsCompilation no

Author Kevin O'Neill [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7401-9802>), Stephen Fleming [aut, cph] (ORCID: <https://orcid.org/0000-0003-0233-4891>)

Maintainer Kevin O'Neill <kevin.o'neill@ucl.ac.uk>

Repository CRAN

Date/Publication 2026-08-21 05:42:50 UTC

Contents

aggregate_metad *Aggregate* data *by* response, confidence, *and other columns*

Description

Counts number of rows in data with unique combinations values in the columns response, confidence, and any other columns in ....

Usage

```
aggregate_metad(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  .name = "N",
  K = NULL,
  pivot_longer = FALSE
)
```

Arguments

data	The data frame to aggregate
...	Grouping columns in data. These columns will be converted to factors.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
.name	The name of the resulting column containing trial counts
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
pivot_longer	If FALSE (default), return aggregated data as a single matrix column. Otherwise, return aggregated data in a tidy long format.

Details

The data frame data must have one column with the name given by .stimulus. Additionally, it must have either:

- Two columns with names given by .response and .confidence
- One column with the name given by .joint_response

Finally, it must also have columns for any additional variables in ....

Value

A tibble with one row per combination of the variables in ..., and another column named by the value of `.response` containing trial counts. For K confidence levels, this will be an $N \times K * 4$ matrix, such that the columns represent (for stimulus S, type 1 response R, and type 2 response C):

$[N_{S=0,R=0,C=K}, \ldots, N_{S=0,R=0,C=1}, N_{S=0,R=1,C=1}, \ldots, N_{S=0,R=1,C=K}, N_{S=1,R=0,C=K}, \ldots, N_{S=1,R=0,C=1}, N_{S=1,R=}$

If `pivot_longer=TRUE`, counts are stored in separate rows rather than as a single matrix column.

Examples

```
# aggregate a dataset without grouping factors
d <- sim_metad()
aggregate_metad(d)

# aggregate a dataset with grouping factors
d2 <- sim_metad_condition()
aggregate_metad(d2, condition)

# can also aggregate ignoring grouping factors
aggregate_metad(d2)

# aggregate data with only `joint_response` column
library(dplyr)
d |>
  ungroup() |>
  mutate(joint_response = joint_response(
    response, confidence,
    max(as.integer(confidence))
  )) |>
  select(-response, -confidence) |>
  aggregate_metad()
```

---

auroc1 *Calculate area under the empirical pseudo-type 1 receiver operating characteristic curve*

Description

Calculate area under the empirical pseudo-type 1 receiver operating characteristic curve

Usage

```
auroc1(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
```

```
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL
)
```

Arguments

data	The data frame to aggregate
...	Grouping columns in data. These columns will be converted to factors.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.

Value

A tibble with columns:

- ...: the grouping columns in data
- auroc1: the area under the pseudo type 1 ROC curve

See Also

roc1(), auroc1_draws(), auroc1_rvars()

Examples

```
# calculate area under the type 1 ROC
auroc1(example_data())

# calculate type 1 ROCs by condition
auroc1(sim_metad_condition(), condition)
```

---

auroc1_draws *Obtain posterior draws of the area under the pseudo type 1 receiver operating characteristic (ROC) curve.*

Description

Given a data frame and a meta-d' model, adds estimates of the area under the type 1 ROC curve. For auroc1_draws and add_auroc1_draws, estimates are returned in a tidy tibble with one row per posterior draw. For auroc1_rvars and add_auroc1_rvars, parameters are returned as posterior::rvars, with one row per row in newdata.

Usage

```
auroc1_draws(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_auroc1_draws(newdata, object, ...)

auroc1_rvars(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_auroc1_rvars(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional parameters passed to tidybayes::epred_draws or tidybayes::epred_rvars
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column

Value

a tibble containing posterior draws of the area under the pseudo type 1 ROC with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `auroc1_draws` and `add_auroc1_draws`, identifiers for the posterior sample
- `auroc1`: the area under the pseudo type 1 ROC curve

See Also

`auroc1()`, `roc1_draws()`, `roc1_rvars()`

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute pseudo-type 1 ROC curve
# equivalent to `auroc1_draws(example_model(), newdata)`
add_auroc1_draws(newdata, example_model())

# use posterior::rvar for additional efficiency
# equivalent to `add_auroc1_draws(newdata, example_model())`
auroc1_rvars(example_model(), newdata)
```

`auroc2` *Calculate the area under empirical type 2 receiver operating characteristic curves*

Description

Calculate the area under empirical type 2 receiver operating characteristic curves

Usage

```
auroc2(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  by_response = TRUE
)
```

Arguments

`data`	The data frame to aggregate
`...`	Grouping columns in `data`. These columns will be converted to factors.
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column
`K`	The number of confidence levels in `data`. If `NULL`, this is estimated from `data` using the maximum value of either the confidence column or joint response column.
`by_response`	If `TRUE` (default), calculate type 2 ROCs conditional on type 1 response.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.response} (if by_response=TRUE): the type 1 response
- auroc2: the area under the type 2 ROC

See Also

roc2(), auroc2_draws(), auroc2_rvars()

Examples

```
# calculate type 2 ROCs by stimulus
auroc2(example_data())

# calculate type 2 ROCs by condition, averaging over type 1 responses
auroc2(sim_metad_condition(), condition, by_response = FALSE)
```

---

auroc2_draws *Obtain posterior draws of the area under the type 2 receiver operating characteristic (ROC) curve.*

Description

Given a data frame and a meta-d' model, adds estimates of AUROC2 (optionally for each type 1 response). For auroc2_draws and add_auroc2_draws, estimates are returned in a tidy tibble with one row per posterior draw. For auroc2_rvars and add_auroc2_rvars, parameters are returned as posterior::rvars, with one row per row in newdata.

Usage

```
auroc2_draws(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  by_response = TRUE
)

add_auroc2_draws(newdata, object, ...)

auroc2_rvars(
  object,
  newdata,
  ...,
```

```
  .response = "response",
  .confidence = "confidence",
  by_response = TRUE
)

add_auroc2_rvars(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional parameters passed to tidybayes::epred_draws
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`by_response`	If `TRUE` (default), compute separate ROCs for each type 1 response. Otherwise, average ROCs across both type 1 responses.

Value

a tibble containing posterior draws of the pseudo type 1 ROC with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `auroc2_draws` and `add_auroc2_draws`, identifiers for the posterior sample
- `{.response}`: the type 1 response for perceived stimulus presence ($R \in \{0, 1\}$)
- `{.confidence}`: the type 2 confidence response ($C \in [1, K]$)
- `p_fa2`: the cumulative probability of an incorrect response ($P(C \geq c \mid R \neq S)$)
- `p_hit2`: the cumulative probability of a correct response ($P(C \geq c \mid R = S)$)

See Also

`auroc2()`, `tidybayes::epred_draws()`, `tidybayes::epred_rvars()`

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute type 2 ROC curve
# equivalent to `add_auroc2_draws(newdata, example_model())`
auroc2_draws(example_model(), newdata)

# use posterior::rvar for additional efficiency
# equivalent to `add_auroc2_rvars(newdata, example_model())`
auroc2_rvars(example_model(), newdata)
```

cor_matrix *Generate a correlation matrix with all off-diagonal values equal to* r

Description

Generate a correlation matrix with all off-diagonal values equal to r

Usage

```
cor_matrix(r, nrow = 2)
```

Arguments

r	The correlation to fill in the matrix off-diagonals
nrow	The number of rows (and columns) of the resulting matrix

Value

An [nrow x nrow] matrix with values along the diagonal equal to 1 and values off of the diagonal equal to r

Examples

```
cor_matrix(0, nrow = 3)

cor_matrix(-.5, nrow = 4)
```

cov_matrix *Generate a covariance matrix.*

Description

Generate a covariance matrix.

Usage

```
cov_matrix(S, OMEGA)
```

Arguments

S	A vector of standard deviations
OMEGA	A correlation matrix

Value

an $N \times N$ covariance matrix, where N = length(S).

Examples

```
sds <- c(1, 2)
corrs <- matrix(c(1, .5, .5, 1), nrow = 2)
cov_matrix(sds, corrs)
```

`epred_draws_metad` *Obtain posterior draws of joint response probabilities*

Description

Given a data frame and a meta-d' model, adds estimates of joint type 1 and type 2 response probabilities. For `epred_draws_metad` and `add_epred_draws_metad`, estimates are returned in a tidy tibble with one row per posterior draw. For `epred_rvars_metad` and `add_epred_rvars_metad`, parameters are returned as posterior::rvars, with one row per row in `newdata`.

Usage

```
epred_draws_metad(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_epred_draws_metad(newdata, object, ...)

epred_rvars_metad(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_epred_rvars_metad(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional arguments passed to tidybayes::add_epred_draws or tidybayes::add_epred_rvars

`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column

Value

a tibble containing posterior draws of model parameters with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `epred_draws_metad`, identifiers for the posterior sample
- `{.stimulus}`, `{.joint_response}`, `{.response}`, `{.confidence}`: identifiers for the response type
- `.epred`: probability of the type 1 and type 2 response given the stimulus, $P(R, C \mid S)$

See Also

`joint_probabilities()`, `tidybayes::epred_draws()`, `tidybayes::epred_rvars()`

Examples

```
newdata <- tidyr::tibble(.row = 1)

# obtain model predictions
# equivalent to `add_epred_draws_metad(newdata, example_model())`
epred_draws_metad(example_model(), newdata)

# obtain model predictions (`posterior::rvar`)
# equivalent to `add_epred_rvars_metad(newdata, example_model())`
epred_rvars_metad(example_model(), newdata)
```

---

example_data *Simulated data for example model fitting*

Description

A simulated data set of 1000 trials from a two-alternative forced choice task with 4 levels of confidence.

Usage

```
example_data()
```

Value

A tibble of 1000 observations containing the following columns:

- `trial`: the trial number
- `stimulus`: the stimulus presence (`0` or `1`)
- `response`: the simulated type 1 response
- `confidence`: the simulated type 2 response
- `correct`: the accuracy of the simulated type 1 response
- `dprime`, `c`, `meta_dprime`, `M`, `meta_c2_0`, `meta_c2_1`: the parameters of the model used for simulation
- `theta`, `theta_1`, `theta_2`: the joint, type 1, and type 2 response probabilities of the model used for simulation

See Also

`sim_metad()`

Examples

```
# Fit an empty model on the example data
# (remove empty=TRUE to actually fit the model)
fit_metad(N ~ 1, example_data(), empty = TRUE)
```

---

example_model *Example meta-d' model for model post-processing*

Description

A model fit to the simulated data example_data. This model includes one constant set of parameters, with no multilevel structure.

Usage

```
example_model()
```

Value

A `brmsfit` object fitted to simulated data

See Also

`fit_metad()`

Examples

```
# inspect summary of posterior distribution
summary(example_model())

# obtain posterior expectations
epred_draws_metad(example_model(), tidyr::tibble(.row = 1))
```

`fit_metad` *Fit the meta-d' model using* `brms` *package*

Description

This function is a wrapper around `brms::brm()` using a custom family for the meta-d' model.

Usage

```
fit_metad(
  formula,
  data,
  ...,
  aggregate = TRUE,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  distribution = "normal",
  metac_absolute = TRUE,
  allow_negative_values = FALSE,
  stanvars = NULL,
  categorical = FALSE,
  logit = TRUE
)
```

Arguments

`formula`	A model formula for some or all parameters of the `metad` brms family. To display all parameter names for a model with `K` confidence levels, use `metad(K)`.
`data`	A tibble containing the data to fit the model. • If `aggregate==TRUE`, `data` should have one row per observation with columns `stimulus`, `response`, `confidence`, and any other variables in `formula` • If `aggregate==FALSE`, it should be aggregated to have one row per cell of the design matrix, with joint type 1/type 2 response counts in a matrix column (see `aggregate_metad()`).

...	Additional parameters passed to the brm function.
aggregate	If TRUE, automatically aggregate data by the variables included in formula using aggregate_metad(). Otherwise, data should already be aggregated.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum level of either the confidence column or joint response column.
distribution	The noise distribution to use for the signal detection model. By default, uses a normal distribution with a mean parameterized by dprime.
metac_absolute	If TRUE, fix the type 2 criterion to be equal to the type 1 criterion. Otherwise, equate the criteria relatively such that metac/metadprime = c/dprime.
allow_negative_values	If FALSE (default), M-ratio is modeled on the logarithmic scale to prevent negative values. If TRUE, M-ratio is modeled on the identity scale which allows for M <= 0. Note that if allow_negative_values=TRUE, priors for the Intercept should be centered at 1 (not at 0 as on the logarithmic scale).
stanvars	Additional stanvars to pass to the model code, for example to define an alternative distribution or a custom model prior (see brms::stanvar()).
categorical	If FALSE (default), use the multinomial likelihood over aggregated data. If TRUE, use the categorical likelihood over individual trials.
logit	If TRUE (default), use the logit parameterization of the likelihood over the log joint response probabilities. If FALSE, use the standard parameterization of the likelihood over the actual joint response probabilities. In most cases, the logit parameterization should provide more stable numerical computations, but the standard parameterization might be preferable in some settings.

Details

fit_metad(formula, data, ...) is approximately the same as brm(formula, data=aggregate_metad(data, ...), family=metad(...), stanvars=stanvars_metad(...), ...). For some models, it may often be easier to use the more explicit version than using fit_metad.

Value

A brmsfit object containing the fitted model

Examples

```
# check which parameters the model has
metad(3)

# fit a basic model on simulated data
```

```
# (use `empty=true` to bypass fitting, *do not use in real analysis*)
fit_metad(N ~ 1, sim_metad(), empty = TRUE)

# fit a basic model on simulated data
fit_metad(N ~ 1, sim_metad())

# fit a model with condition-level effects
fit_metad(
  bf(
    N ~ condition,
    dprime + c + metac2zero1diff + metac2zero2diff +
      metac2one1diff + metac2one1diff ~ condition
  ),
  data = sim_metad_condition()
)
```

`joint_probabilities` *Calculate empirical joint type 1/type 2 response probabilities*

Description

Given a dataset `data`, determine the probability of each combination of type 1 and type 2 responses, optionally conditional on stimulus.

Usage

```
joint_probabilities(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  by_stimulus = TRUE
)
```

Arguments

`data`	The data frame to aggregate
`...`	Grouping columns in `data`. These columns will be converted to factors.
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column

K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
by_stimulus	If TRUE (default), calculate type 2 response probabilities conditional on stimulus.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.stimulus} (if by_stimulus=TRUE): the stimulus
- {.response}: the type 1 response
- {.confidence}: the type 2 response
- {.joint_response}: the joint type 1/type 2 response
- n: the number of rows in data with the corresponding stimulus (if by_stimulus=TRUE), response, confidence, and joint_response
- p: the proportion of rows in data with the corresponding response (per stimulus if by_stimulus=TRUE)

See Also

epred_draws(), epred_rvars()

Examples

```
# calculate type 2 response probabilities by stimulus
joint_probabilities(example_data())

# calculate type 2 response probabilities by condition, averaging over stimuli
joint_probabilities(sim_metad_condition(), condition, by_stimulus = FALSE)
```

---

joint_response *Convert between separate and joint type 1/type 2 responses*

Description

Confidence ratings and decisions are collected in one of two ways.

- For separate ratings, there will be a type 1 response ($R \in \{0, 1\}$) and a type 2 response ($C \in [1, K]$).
- For joint ratings, there is instead a combined type 1/type 2 response ($J \in [1, 2K]$), with values in $[1, K]$ indicating a type 1 response of 0 and values in $[K + 1, 2K]$ indicating a type 1 response of 1, with confident responses at the ends of the scale.

joint_response converts separate type 1 and type 2 responses into the joint format

type1_response and type2_response convert the joint response into separate responses.

Usage

```
joint_response(response, confidence, K)

type1_response(joint_response, K)

type2_response(joint_response, K)
```

Arguments

response	A type 1 response (0 or 1)
confidence	A type 2 response/confidence rating (in 1:K)
K	The number of confidence levels
joint_response	A joint type 1/type 2 response

Value

A joint response (for joint_response), type 1 response (for type1_response), or type 2 response (for type2_response)

Examples

```
# convert joint_response to separate responses
joint <- 1:8
K <- 4
type1_response(joint, K)
type2_response(joint, K)

# convert separate responses to a joint response
t1 <- rep(c(0, 1), each = 4)
t2 <- c(4:1, 1:4)
joint_response(t1, t2, K)
```

---

linpred_draws_metad *Obtain posterior draws of meta-d' model parameters*

Description

Given a data frame and a meta-d' model, adds estimates of all model parameters. For linpred_draws_metad and add_linpred_draws_metad, parameters are returned in a tidy tibble with one row per posterior draw. For linpred_rvars_metad and add_linpred_rvars_metad, parameters are returned as posterior::rvars, with one row per row in newdata.

Usage

```
linpred_draws_metad(object, newdata, ..., pivot_longer = FALSE)

add_linpred_draws_metad(newdata, object, ..., pivot_longer = FALSE)

linpred_rvars_metad(object, newdata, ..., pivot_longer = FALSE)

add_linpred_rvars_metad(newdata, object, ..., pivot_longer = FALSE)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional arguments passed to tidybayes::add_linpred_draws or tidybayes::add_linpred_rvars
`pivot_longer`	Return the draws in long format? • if `TRUE`, resulting data frame has one row per posterior draw per model parameter • if `FALSE` (default), resulting data frame has one row per posterior draw

Value

a tibble containing posterior draws of model parameters with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `linpred_draws_metad`, identifiers for the posterior sample
- `.variable`, `.value`: if `pivot_longer=TRUE`, `.variable` identifies different meta-d' model parameters and `.value` stores posterior samples
- `M`, `dprime`, `c`, `meta_dprime`, `meta_c`, `meta_c2_0_<k>`, `meta_c2_1_<k>`: if `pivot_longer=FALSE`, posterior samples of all meta-d' model parameters

See Also

`tidybayes::linpred_draws()`, `tidybayes::linpred_rvars()`

Examples

```
newdata <- tidyr::tibble(.row = 1)

# obtain model parameters (wide format)
# equivalent to `add_linpred_draws_metad(newdata, example_model())`
linpred_draws_metad(example_model(), newdata)

# obtain model parameters (long format)
# equivalent to `add_linpred_draws_metad(newdata, example_model(), pivot_longer = TRUE)`
linpred_draws_metad(example_model(), newdata, pivot_longer = TRUE)

# obtain model parameters (wide format, posterior::rvar)
# equivalent to `add_linpred_rvars_metad(newdata, example_model())`
```

```
linpred_rvars_metad(example_model(), newdata)

# obtain model parameters (long format, posterior::rvar)
# equivalent to `add_linpred_rvars_metad(newdata, example_model(), pivot_longer = TRUE)`
linpred_rvars_metad(example_model(), newdata, pivot_longer = TRUE)
```

`mean_confidence` *Calculate empirical mean confidence*

Description

Given a dataset `data`, determine the mean confidence rating, optionally conditional on stimulus and/or type 1 response.

Usage

```
mean_confidence(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  by_stimulus = TRUE,
  by_response = TRUE,
  by_correct = FALSE
)
```

Arguments

`data`	The data frame to aggregate
`...`	Grouping columns in `data`. These columns will be converted to factors.
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column
`K`	The number of confidence levels in `data`. If `NULL`, this is estimated from `data` using the maximum value of either the confidence column or joint response column.
`by_stimulus`	If `TRUE` (default), calculate mean confidence conditional on stimulus. Ignored if `by_correct=TRUE`.
`by_response`	If `TRUE` (default), calculate mean confidence conditional on type 2 response. Ignored if `by_correct=TRUE`.

by_correct — If FALSE (default), calculate mean confidence conditional on stimulus and/or type 1 response. If TRUE, instead calculate mean confidence conditional on accuracy.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.stimulus}: the stimulus (if by_stimulus=TRUE)
- {.response}: the type 1 response (if by_response=TRUE)
- correct: the accuracy (if by_correct=TRUE)
- mean_confidence: the mean confidence rating

See Also

mean_confidence_draws(), mean_confidence_rvars()

Examples

```
# calculate mean confidence by stimulus and response
mean_confidence(example_data())

# calculate mean confidence by accuracy
mean_confidence(example_data(), by_correct = TRUE)

# calculate mean confidence by condition, averaging over type 1 responses
mean_confidence(sim_metad_condition(), condition, by_response = FALSE)
```

---

mean_confidence_draws *Obtain posterior draws of mean confidence*

Description

Computes posterior mean confidence conditional on stimulus and response ($\mathbb{E}[C \mid S = s, R = r]$), stimulus (averaging over responses, $\mathbb{E}[C \mid S = s]$), response (averaging over stimuli, $\mathbb{E}[C \mid R = r]$), neither (averaging over stimuli and responses, $\mathbb{E}[C]$), or accuracy ($\mathbb{E}[C \mid A = (r = s)]$). For mean_confidence_draws and add_mean_confidence_draws, estimates are returned in a tidy tibble with one row per posterior draw, stimulus, and response. For mean_confidence_rvars and add_mean_confidence_rvars, estimates are returned as posterior::rvars, with one row per row in newdata.

add_mean_confidence_draws is an alias of mean_confidence_draws with argument order swapped.

Usage

```
mean_confidence_draws(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  by_stimulus = TRUE,
  by_response = TRUE,
  by_correct = FALSE
)

add_mean_confidence_draws(newdata, object, ...)

mean_confidence_rvars(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  by_stimulus = TRUE,
  by_response = TRUE,
  by_correct = FALSE
)

add_mean_confidence_rvars(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional arguments to tidybayes::epred_draws or tidybayes::epred_rvars
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`by_stimulus`	If TRUE, predict mean confidence separately by stimulus. Otherwise, predict mean confidence averaging over stimuli. Ignored if `by_correct==TRUE`.
`by_response`	If TRUE, predict mean confidence separately by response Otherwise, predict mean confidence averaging over responses. Ignored if `by_correct==TRUE`.
`by_correct`	If TRUE, predict mean confidence separately for correct and incorrect responses.

Value

a tibble containing posterior draws of mean confidence with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `mean_confidence_draws` and `add_mean_confidence_draws`, identifiers for the posterior sample

- {.stimulus}: indicator for stimulus presence (if by_stimulus==TRUE & by_correct==FALSE)
- {.response}: indicator for type 1 response (if by_response==TRUE & by_correct==FALSE)
- correct: indicator for the accuracy of the type 1 response (if by_correct==TRUE)
- .epred: the predicted mean confidence

See Also

mean_confidence(), tidybayes::epred_draws(), tidybayes::epred_rvars()

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute mean confidence by stimulus and response
# equivalent to `add_mean_confidence_draws(newdata, example_model())`
mean_confidence_draws(example_model(), newdata)

# compute mean confidence by stimulus
# equivalent to `add_mean_confidence_draws(newdata, example_model(), by_response = FALSE)`
mean_confidence_draws(example_model(), newdata, by_response = FALSE)

# compute mean confidence by response
# equivalent to `add_mean_confidence_draws(newdata, example_model(), by_stimulus = FALSE)`
mean_confidence_draws(example_model(), newdata, by_stimulus = FALSE)

# compute mean confidence by accuracy
# equivalent to `add_mean_confidence_draws(newdata, example_model(), by_correct = TRUE)`
mean_confidence_draws(example_model(), newdata, by_correct = TRUE)

# compute mean confidence averaging over stimuli and responses
# equivalent to `add_mean_confidence_draws(newdata, example_model(), ...)`
mean_confidence_draws(example_model(), newdata, by_stimulus = FALSE, by_response = FALSE)

# use `posterior::rvar` for increased efficiency
# equivalent to `add_mean_confidence_rvars(newdata, example_model())`
mean_confidence_rvars(example_model(), newdata)
```

---

metac2_parameters *Obtain a vector of the names of the* K-1 *parameters representing the differences between successive confidence criteria for the meta-d' model with* K *levels of confidence.*

Description

Obtain a vector of the names of the K-1 parameters representing the differences between successive confidence criteria for the meta-d' model with K levels of confidence.

Usage

```
metac2_parameters(K, response = "both")
```

Arguments

K	The number of confidence levels
response	If "both", list confidence criteria parameters for both "0" and "1" responses. If "zero" or "0", list only confidence criteria for the "0" response. If "one" or "1", list only confidence criteria for the "1" response.

Examples

```
# list confidence criteria parameters for K=3 confidence levels
metac2_parameters(K = 3)

# list parameters for "0" responses
metac2_parameters(K = 3, response = "zero")

# useful for setting model priors
set_prior("normal(0, 1)", class = "b", dpar = metac2_parameters(K = 4))
```

---

metacognitive_bias_draws

Obtain posterior draws of an index of metacognitive bias

Description

Computes meta-Δ, an index of metacognitive bias. meta-Δ is the distance between `meta_c` and the average of the the confidence criteria `meta_c2_0` and `meta_c2_1`. For `metacognitive_bias_draws` and `add_metacognitive_bias_draws`, parameters are returned in a tidy tibble with one row per posterior draw and per response. For `metacognitive_bias_rvars` and `add_metacognitive_bias_rvars`, parameters are returned as posterior::rvars, with one row per row in `newdata` and per response.

Usage

```
metacognitive_bias_draws(
  object,
  newdata,
  ...,
  .response = "response",
  by_response = TRUE
)

add_metacognitive_bias_draws(newdata, object, ...)

metacognitive_bias_rvars(
```

```
  object,
  newdata,
  ...,
  .response = "response",
  by_response = TRUE
)

add_metacognitive_bias_rvars(newdata, object, ...)
```

Arguments

object	The brms model with the metad family
newdata	A data frame from which to generate posterior predictions
...	Additional parameters passed to tidybayes::epred_draws or tidybayes::epred_rvars
.response	The name of "response" column
by_response	If TRUE, compute metacognitive bias separately for the two type 1 responses. If FALSE, compute an un-weighted average of the two measures.

Value

a tibble containing posterior draws of meta-Δ with the following columns:

- .row: the row of newdata
- .chain, .iteration, .draw: for metacognitive_bias_draws and add_metacognitive_bias_draws, identifiers for the posterior sample
- {.response}: the type 1 response for perceived stimulus presence
- metacognitive_bias: the distance between meta_c and the average of the confidence criteria meta_c2_{response}.

See Also

tidybayes::linpred_draws(), tidybayes::linpred_rvars()

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute metacognitive bias
# equivalent to `add_metacognitive_bias_draws(newdata, example_model())`
metacognitive_bias_draws(example_model(), newdata)

# use `posterior::rvar` for increased efficiency
# equivalent to `add_metacognitive_bias_rvars(newdata, example_model())`
metacognitive_bias_rvars(example_model(), newdata)

# average over the two type 1 responses
metacognitive_bias_rvars(example_model(), newdata, by_response = FALSE)
```

`metad` `brms` *family for the metad' model*

Description

`brms` family for the metad' model

Usage

```
metad(
  K,
  distribution = "normal",
  metac_absolute = TRUE,
  categorical = FALSE,
  allow_negative_values = FALSE
)
```

Arguments

`K`	The number of confidence levels
`distribution`	The noise distribution to use for the signal detection model
`metac_absolute`	If `TRUE`, fix the type 2 criterion to be equal to the type 1 criterion. Otherwise, equate the criteria relatively such that $$\frac{\text{meta-}c}{\text{meta-}d'} = \frac{c}{d'}$$
`categorical`	If `FALSE` (default), use the multinomial likelihood over aggregated data. If `TRUE`, use the categorical likelihood over individual trials.
`allow_negative_values`	If `FALSE` (default), M-ratio is modeled on the logarithmic scale to prevent negative values. If `TRUE`, M-ratio is modeled on the identity scale which allows for `M <= 0`.

Value

A `brms` family for the metad' model with K confidence levels

Examples

```
# create a family using the normal distribution and 3 levels of confidence
metad(3)

# create a family with meta_c = M * c
metad(3, metac_absolute = FALSE)

# create a family with an alternative distribution
# note: cumulative distribution functions must be defined
# in R and in Stan using [brms::stanvar()]
metad(4, distribution = "gumbel_min")
```

`metad_pmf` *Generate (log) probability simplex over the joint type 1/type 2 responses*

Description

Generate (log) probability simplex over the joint type 1/type 2 responses

Usage

```
metad_pmf(
  stimulus,
  dprime,
  c,
  meta_dprime,
  meta_c,
  meta_c2_0,
  meta_c2_1,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf,
  log = FALSE
)
```

Arguments

`stimulus`	the stimulus (0 or 1)
`dprime`	the type 1 sensitivity
`c`	the type 1 response criterion
`meta_dprime`	the type 2 sensitivity
`meta_c`	the type 1 criterion for generating confidence ratings
`meta_c2_0`	the type 2 response criteria for `"0"` responses, indexed by increasing confidence levels
`meta_c2_1`	the type 2 response criteria for `"1"` responses, indexed by increasing confidence levels
`lcdf`	The log cumulative distribution function for the underlying distribution in the metad' model. By default, uses the normal distribution with a standard deviation of 1.
`lccdf`	The log complement cumulative distribution function for the underlying distribution in the metad' model. By default, uses the normal distribution with a standard deviation of 1.
`log`	if TRUE, return log probabilities instead of probabilities

Value

A probability simplex

$$\left[P(R=0,C=K|S=0),\ldots,P(R=0,C=1|S=0),P(R=0,C=1|S=1),\ldots,P(R=1,C=1|S=1)\right]$$

for response R and confidence C given stimulus S, as defined by the meta-d' model.

Examples

```
metad_pmf(
  stimulus = 0, dprime = 2, c = .5, meta_dprime = 1, meta_c = .5,
  meta_c2_0 = c(0, -.5), meta_c2_1 = c(1, 1.5)
)
```

`normal_lcdf` *Normal cumulative distribution functions*

Description

Normal cumulative distribution functions

Usage

```
normal_lcdf(x, mu)

normal_lccdf(x, mu)
```

Arguments

`x`	The quantile to evaluate the l(c)cdf at
`mu`	The mean of the normal distribution

Value

$log(P(X < x))$ (for `normal_lcdf`) or $log(P(X > x))$ (for `normal_lccdf`) where X is sampled from a normal distribution with mean `mu` and standard deviation of 1

Examples

```
normal_lcdf(0, mu = 1)
normal_lccdf(0, mu = 1)
```

predicted_draws_metad *Obtain posterior predictions of joint responses*

Description

Given a data frame and a meta-d' model, adds predictions of joint type 1 and type 2 responses For `predicted_draws_metad` and `add_predicted_draws_metad`, predictions are returned in a tidy tibble with one row per posterior draw. For `predicted_rvars_metad` and `add_predicted_rvars_metad`, parameters are returned as posterior::rvars, with one row per row in `newdata`.

Usage

```
predicted_draws_metad(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_predicted_draws_metad(newdata, object, ...)

predicted_rvars_metad(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response"
)

add_predicted_rvars_metad(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional arguments passed to tidybayes::add_predicted_draws or tidybayes::add_predicted_rvars
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column

Value

a tibble containing posterior draws of model parameters with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `predicted_draws_metad`, identifiers for the posterior sample
- `{.stimulus}`, `{.joint_response}`, `{.response}`, `{.confidence}`: identifiers for the response type
- `.prediction`: predicted type 1 and type 2 responses given the stimulus

See Also

`tidybayes::predicted_draws()`, `tidybayes::predicted_rvars()`

Examples

```
newdata <- aggregate_metad(example_data())

# obtain model predictions
# equivalent to `add_predicted_draws_metad(newdata, example_model())`
predicted_draws_metad(example_model(), newdata)

# obtain model predictions (posterior::rvar)
# equivalent to `add_predicted_rvars_metad(newdata, example_model())`
predicted_rvars_metad(example_model(), newdata)
```

---

rmatrixnorm *Sample from a matrix-normal distribution*

Description

Sample from a matrix-normal distribution

Usage

```
rmatrixnorm(mu, L_sigma_rows, L_sigma_cols)
```

Arguments

`mu`	a matrix of means
`L_sigma_rows`	the Cholesky-decomposed covariance matrix for the rows
`L_sigma_cols`	the Cholesky-decomposed covariance matrix for the columns

Value

A single sample from a matrix-normal distribution with mean `mu` (a matrix), row-wise covariances `sigma_rows`, and column-wise covariances `sigma_cols`, where `L_sigma_rows` and `L_sigma_cols` are the Cholesky-decomposed covariance matrices

Examples

```
mu <- matrix(rep(0, 8), nrow = 4)
sd_rows <- rep(1, 4)
sd_cols <- rep(1, 2)
r_rows <- cor_matrix(.25, 4)
r_cols <- cor_matrix(.75, 2)
L_sigma_rows <- chol(cov_matrix(sd_rows, r_rows))
L_sigma_cols <- chol(cov_matrix(sd_cols, r_cols))
rmatrixnorm(mu, L_sigma_rows, L_sigma_cols)
```

---

roc1 *Calculate empirical pseudo-type 1 receiver operating characteristic curves*

Description

Given a dataset `data`, determine the cumulative probability of each combination of type 1 and type 2 responses conditional on stimulus.

Usage

```
roc1(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  bounds = FALSE
)
```

Arguments

`data`	The data frame to aggregate
`...`	Grouping columns in `data`. These columns will be converted to factors.
`.stimulus`	The name of "stimulus" column
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column

K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
bounds	If TRUE, include the endpoints of the ROC at $(0, 0)$ and $(1, 1)$. Otherwise, the endpoints are excluded.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.response}: the type 1 response
- {.confidence}: the type 2 response
- {.joint_response}: the joint type 1/type 2 response
- n_0: the number of rows in data with stimulus=0 and the corresponding joint response
- n_1: the number of rows in data with stimulus=1 and the corresponding joint response
- p_0: where stimulus=0, the proportion of rows in data with joint response equal to .joint_response
- p_1: where stimulus=1, the proportion of rows in data with joint response equal to .joint_response
- p_fa: where stimulus=0, the proportion of rows in data with joint response greater than .joint_response
- p_hit: where stimulus=1, the proportion of rows in data with joint response greater than .joint_response

See Also

roc1_draws(), roc1_rvars()

Examples

```
# calculate type 1 ROCs
roc1(example_data())

# calculate type 1 ROCs by condition
roc1(sim_metad_condition(), condition)
```

---

roc1_draws *Obtain posterior draws of the pseudo type 1 receiver operating characteristic (ROC) curve.*

Description

Given a data frame and a meta-d' model, adds estimates of the cumulative probability over joint_responses. For roc1_draws and add_roc1_draws, estimates are returned in a tidy tibble with one row per posterior draw and per joint response. For roc1_rvars and add_roc1_rvars, parameters are returned as posterior::rvars, with one row per row in newdata and per joint response.

Usage

```
roc1_draws(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  bounds = FALSE
)

add_roc1_draws(newdata, object, ...)

roc1_rvars(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  bounds = FALSE
)

add_roc1_rvars(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional parameters passed to tidybayes::epred_draws or tidybayes::epred_rvars
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column
`.joint_response`	The name of "joint_response" column
`bounds`	If `TRUE`, include the endpoints of the ROC at $(0, 0)$ and $(1, 1)$. Otherwise, the endpoints are excluded.

Value

a tibble containing posterior draws of the pseudo type 1 ROC with the following columns:

- `.row`: the row of `newdata`
- `.chain`, `.iteration`, `.draw`: for `roc1_draws` and `add_roc1_draws`, identifiers for the posterior sample
- `{.joint_response}`: the combined type 1 / type 2 response ($J \in [1, 2K]$) for K confidence levels)

- `{.response}`: the type 1 response for perceived stimulus presence ($R \in \{0, 1\}$)
- `{.confidence}`: the type 2 confidence response ($C \in [1, K]$)
- `p_fa`: the cumulative probability of a 'present'/'old' response for `stimulus==0` ($P(J \geq j \mid S = 0)$)
- `p_hit`: the cumulative probability of a 'present'/'old' response for `stimulus==1` ($P(J \geq j \mid S = 1)$)

See Also

`roc1()`, `tidybayes::epred_draws()`, `tidybayes::epred_rvars()`

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute pseudo-type 1 ROC curve
# equivalent to ``
roc1_draws(example_model(), newdata)
add_roc1_draws(newdata, example_model())

# use posterior::rvar for additional efficiency
# equivalent to `add_roc1_draws(newdata, example_model())`
roc1_rvars(example_model(), newdata)

# include the ROC bounds
# equivalent to `add_roc1_draws(newdata, example_model(), bounds = TRUE)`
roc1_draws(example_model(), newdata, bounds = TRUE)
```

---

roc2 *Calculate empirical type 2 receiver operating characteristic curve*

Description

Given a dataset `data`, determine the cumulative probability of each type 2 responses conditional on accuracy, optionally conditional on type 1 response.

Usage

```
roc2(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  bounds = FALSE,
  by_response = TRUE
)
```

Arguments

data	The data frame to aggregate
...	Grouping columns in data. These columns will be converted to factors.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
bounds	If TRUE, include the endpoints of the ROC at $(0, 0)$ and $(1, 1)$. Otherwise, the endpoints are excluded.
by_response	If TRUE (default), calculate type 2 ROCs conditional on type 1 response.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.response} (if by_response=TRUE): the type 1 response
- {.confidence}: the type 2 response
- n_0: the number of rows in data with stimulus=0 and the corresponding joint_response
- n_1: the number of rows in data with stimulus=1 and the corresponding joint_response
- p_0: for incorrect trials, the proportion of rows in data with confidence equal to confidence
- p_1: for correct trials the proportion of rows in data with confidence equal to confidence
- p_fa2: for incorrect trials, the proportion of rows in data with confidence greater than confidence
- p_hit2: for correct trials, the proportion of rows in data with confidence greater than confidence

See Also

roc2_draws(), roc2_rvars()

Examples

```
# calculate type 2 ROCs by stimulus
roc2(example_data())

# calculate type 2 ROCs by condition, averaging over type 1 responses
roc2(sim_metad_condition(), condition, by_response = FALSE)
```

roc2_draws *Obtain posterior draws of the response-specific type 2 receiver operating characteristic (ROC) curves.*

Description

Given a data frame and a meta-d' model, adds estimates of the cumulative probability over confidence for each type 1 response. For `roc2_draws` and `add_roc2_draws`, estimates are returned in a tidy tibble with one row per posterior draw and per joint response. For `roc2_rvars` and `add_roc2_rvars`, parameters are returned as `posterior::rvars`, with one row per row in `newdata` and per joint response.

Usage

```
roc2_draws(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  bounds = FALSE,
  by_response = TRUE
)

add_roc2_draws(newdata, object, ...)

roc2_rvars(
  object,
  newdata,
  ...,
  .response = "response",
  .confidence = "confidence",
  bounds = FALSE,
  by_response = TRUE
)

add_roc2_rvars(newdata, object, ...)
```

Arguments

`object`	The `brms` model with the `metad` family
`newdata`	A data frame from which to generate posterior predictions
`...`	Additional parameters passed to tidybayes::epred_draws
`.response`	The name of "response" column
`.confidence`	The name of "confidence" column

bounds	If TRUE, include the endpoints of the ROC at $(0, 0)$ and $(1, 1)$. Otherwise, the endpoints are excluded.
by_response	If TRUE (default), compute separate ROCs for each type 1 response. Otherwise, average ROCs across both type 1 responses.

Value

a tibble containing posterior draws of the pseudo type 1 ROC with the following columns:

- .row: the row of newdata
- .chain, .iteration, .draw: for roc2_draws and add_roc2_draws, identifiers for the posterior sample
- {.response}: the type 1 response for perceived stimulus presence ($R \in \{0, 1\}$)
- {.confidence}: the type 2 confidence response ($C \in [1, K]$)
- p_fa2: the cumulative probability of an incorrect response ($P(C \geq c \mid R \neq S)$)
- p_hit2: the cumulative probability of a correct response ($P(C \geq c \mid R = S)$)

See Also

roc2(), tidybayes::epred_draws(), tidybayes::epred_rvars()

Examples

```
newdata <- tidyr::tibble(.row = 1)

# compute type 2 ROC curve
# equivalent to `add_roc2_draws(newdata, example_model())`
roc2_draws(example_model(), newdata)

# use posterior::rvar for additional efficiency
# equivalent to `add_roc2_rvars(newdata, example_model())`
roc2_rvars(example_model(), newdata)

# include the ROC bounds
# equivalent to `roc2_draws(newdata, example_model(), bounds = TRUE)`
roc2_draws(example_model(), newdata, bounds = TRUE)
```

---

sim_metad *Simulate from the meta-d' model*

Description

Generate a simulated dataset from the meta-d' model with sensitivity dprime, response bias c, metacognitive efficiency M, and distances between confidence thresholds c2_0_diff and c2_1_diff (for the two responses).

Usage

```
sim_metad(
  N_trials = 100,
  dprime = 1,
  c = 0,
  M = 1,
  c2_0_diff = rep(0.5, 3),
  c2_1_diff = rep(0.5, 3),
  metac_absolute = TRUE,
  summarize = FALSE,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf
)
```

Arguments

N_trials	Total number of trials to simulate. Half of these trials will have stimulus=0 and half will have stimulus=1.
dprime	The sensitivity of the signal detection agent to simulate
c	The response bias of the signal detection agent to simulate
M	The metacognitive efficiency of the agent, where negative values indicate below chance metacognitive sensitivity, 0 indicates an absence of metacognitive sensitivity, values between 0 and 1 indicate metacognitive inefficiency, 1indicates optimal metacognitive indicate metacognitive hyper-efficiency.
c2_0_diff, c2_1_diff	Distances between confidence thresholds for "0" and "1" responses, such that meta_c2_0 = meta_c - cumsum(c2_0_diff) and meta_c2_1 = meta_c + cumsum(c2_1_diff).
metac_absolute	Determines how to fix the type 1 threshold for modeling confidence ratings. If metac_absolute=TRUE, meta_c = c. Otherwise, meta_c = M * c.
summarize	Aggregate the data? • If summarize=FALSE, returns a dataset with one row per observation. • If summarize=TRUE, returns an aggregated dataset where n is the number of observations per response, accuracy, and confidence level.
lcdf, lccdf	The log (complement) cumulative distribution function of the underlying signal distribution. By default, uses a normal(+/-dprime/2, 1) distribution.

Value

A simulated dataset of type 1 responses and confidence ratings, with columns:

- trial: the simulated trial number
- stimulus: the value of the stimulus on each trial (either 0 or 1)
- response: the simulated type 1 response (either 0 or 1)
- correct: whether stimulus==response (either 0 or 1)
- confidence: the simulated type 2 response (from 1 to length(c2_0_diff)+1)

- dprime:theta_2: the simulated agent's parameter values

If summarize=TRUE, the trial column is replaced with an n column indicating the number of simulated type 1/type 2 responses for each possible value.

Examples

```
sim_metad(N_trials = 10)
sim_metad(N_trials = 10000, summarize = TRUE)
sim_metad(N_trials = 10, c2_0_diff = 1, c2_1_diff = 1)
```

---

sim_metad_condition *Simulate from the meta-d' model across separate conditions*

Description

Generate a simulated dataset across separate conditions from the meta-d' model with sensitivity dprime, response bias c, metacognitive efficiency M, and distances between confidence thresholds c2_0_diff and c2_1_diff (for the two responses).

Usage

```
sim_metad_condition(
  N_trials = 100,
  dprime = rep(1, 2),
  c = rep(0, 2),
  M = rep(1, 2),
  c2_0_diff = list(rep(0.5, 3), rep(0.5, 3)),
  c2_1_diff = list(rep(0.5, 3), rep(0.5, 3)),
  metac_absolute = TRUE,
  summarize = FALSE,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf
)
```

Arguments

N_trials	Total number of trials to simulate. Half of these trials will have stimulus=0 and half will have stimulus=1.
dprime	The sensitivity of the signal detection agent to simulate
c	The response bias of the signal detection agent to simulate
M	The metacognitive efficiency of the agent, where negative values indicate below chance metacognitive sensitivity, 0 indicates an absence of metacognitive sensitivity, values between 0 and 1 indicate metacognitive inefficiency, 1indicates optimal metacognitive indicate metacognitive hyper-efficiency.

c2_0_diff, c2_1_diff
: Distances between confidence thresholds for "0" and "1" responses, such that meta_c2_0 = meta_c - cumsum(c2_0_diff) and meta_c2_1 = meta_c + cumsum(c2_1_diff).

metac_absolute
: Determines how to fix the type 1 threshold for modeling confidence ratings. If metac_absolute=TRUE, meta_c = c. Otherwise, meta_c = M * c.

summarize
: Aggregate the data? If summarize=FALSE, returns a dataset with one row per observation. If summarize=TRUE, returns an aggregated dataset where n is the number of observations per response, accuracy, and confidence level.

lcdf, lccdf
: The log (complement) cumulative distribution function of the underlying signal distribution. By default, uses a normal(+/-dprime/2, 1) distribution.

Value

A simulated dataset of type 1 responses and confidence ratings, with columns:

- trial: the simulated trial number
- condition: the simulated condition number
- stimulus: the value of the stimulus on each trial (either 0 or 1)
- response: the simulated type 1 response (either 0 or 1)
- correct: whether stimulus==response (either 0 or 1)
- confidence: the simulated type 2 response (from 1 to length(c2_0_diff)+1)
- dprime:theta_2: the simulated agent's parameter values

If summarize=TRUE, the trial column is replaced with an n column indicating the number of simulated type 1/type 2 responses for each possible value.

Examples

```
sim_metad_condition(N_trials = 10)
sim_metad_condition(N_trials = 10000, summarize = TRUE)
sim_metad_condition(N_trials = 10, c2_0_diff = list(1, .5), c2_1_diff = list(1, .5))
```

---

sim_metad_participant *Simulate from the hierarchical meta-d' model*

Description

Generate a simulated dataset across participants from the meta-d' model with sensitivity dprime, response bias c, metacognitive efficiency M, and distances between confidence thresholds c2_0_diff and c2_1_diff (for the two responses).

Usage

```
sim_metad_participant(
  N_participants = 100,
  N_trials = 100,
  mean_dprime = 1,
  sd_dprime = 0.5,
  mean_c = 0,
  sd_c = 0.5,
  mean_M = 0,
  sd_M = 0.5,
  mean_z_c2_0 = rep(-1, 3),
  sd_z_c2_0 = rep(0.1, 3),
  r_z_c2_0 = diag(3),
  mean_z_c2_1 = rep(-1, 3),
  sd_z_c2_1 = rep(0.1, 3),
  r_z_c2_1 = diag(3),
  metac_absolute = TRUE,
  allow_negative_values = FALSE,
  summarize = FALSE,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf
)
```

Arguments

`N_trials`, `N_participants`
: Total number of participants and trials to simulate per participant. Half of these trials will have `stimulus=0` and half will have `stimulus=1`.

`mean_dprime`, `sd_dprime`
: The mean and standard deviation of sensitivities of the signal detection agents to simulate

`mean_c`, `sd_c`
: The mean and standard deviation of response bias of the signal detection agents to simulate

`mean_M`, `sd_M`
: The mean and standard deviation of metacognitive efficiency of the agents. If `allow_negative_values==TRUE` (default), M-ratio is simulated on the logarithmic scale, where `0` indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers indicate metacognitive hyper-efficiency. Otherwise, M-ratio is modeled on its natural scale to allow negative values.

`mean_z_c2_0`, `mean_z_c2_1`
: Mean distance between confidence thresholds for `"0"` and `"1"` responses on the log_scale, such that `meta_c2_0 = meta_c - cumulative_sum(exp(z_c2_0))` and `meta_c2_1 = meta_c + cumulative_sum(exp(z_c2_1))`.

`sd_z_c2_0`, `sd_z_c2_1`
: SD of log distances between confidence thresholds for `"0"` and `"1"` responses on the log_scale.

r_z_c2_0, r_z_c2_1
: Correlation of log distances between confidence thresholds for "0" and "1" responses on the log_scale.

metac_absolute
: Determines how to fix the type 1 threshold for modeling confidence ratings. If metac_absolute=TRUE, meta_c = c. Otherwise, meta_c = M * c.

allow_negative_values
: If allow_negative_values=FALSE (default), M-ratio is simulated using a normal distribution on the logarithmic scale to prohibit negative values. If allow_negative_values=TRUE, then M-ratio is simulated on its natural scale, allowing negative values.

summarize
: Aggregate the data? If summarize=FALSE, returns a dataset with one row per observation. If summarize=TRUE, returns an aggregated dataset where n is the number of observations per response, accuracy, and confidence level.

lcdf
: The log cumulative distribution function of the underlying signal distribution. By default, uses a normal(+/-dprime/2, 1) distribution.

lccdf
: The log complement cumulative distribution function of the underlying signal distribution. By default, uses a normal(+/-dprime/2, 1) distribution.

Value

A simulated dataset of type 1 responses and confidence ratings, with columns:

- trial: the simulated trial number
- participant: the simulated participant number
- stimulus: the value of the stimulus on each trial (either 0 or 1)
- response: the simulated type 1 response (either 0 or 1)
- correct: whether stimulus==response (either 0 or 1)
- confidence: the simulated type 2 response (from 1 to length(c2_0_diff)+1)
- dprime:theta_2: the simulated agent's parameter values

If summarize=TRUE, the trial column is replaced with an n column indicating the number of simulated type 1/type 2 responses for each possible value.

Examples

```
sim_metad_participant(N_participants = 10, N_trials = 10)
sim_metad_participant(N_participants = 25, mean_dprime = 2, mean_M = -1)
```

---

sim_metad_participant_condition

Simulate from the hierarchical meta-d' model across within-participant conditions

---

Description

Generate a simulated dataset across participants and conditions from the meta-d' model with sensitivity dprime, response bias c, metacognitive efficiency M, and distances between confidence thresholds c2_0_diff and c2_1_diff (for the two responses).

Usage

```
sim_metad_participant_condition(
  N_participants = 100,
  N_trials = 100,
  mean_dprime = rep(1, 2),
  sd_dprime = rep(0.5, 2),
  r_dprime = diag(2),
  mean_c = rep(0, 2),
  sd_c = rep(0.5, 2),
  r_c = diag(2),
  mean_M = rep(0, 2),
  sd_M = rep(0.5, 2),
  r_M = diag(2),
  mean_z_c2_0 = matrix(rep(-1, 6), nrow = 3, ncol = 2),
  sd_z_c2_0_condition = rep(0.1, 2),
  r_z_c2_0_condition = diag(2),
  sd_z_c2_0_confidence = rep(0.1, 3),
  r_z_c2_0_confidence = diag(3),
  mean_z_c2_1 = matrix(rep(-1, 6), nrow = 3, ncol = 2),
  sd_z_c2_1_condition = rep(0.1, 2),
  r_z_c2_1_condition = diag(2),
  sd_z_c2_1_confidence = rep(0.1, 3),
  r_z_c2_1_confidence = diag(3),
  metac_absolute = TRUE,
  allow_negative_values = FALSE,
  summarize = FALSE,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf
)
```

Arguments

`N_trials`, `N_participants`
: Total number of participants and trials to simulate per participant. Half of these trials will have `stimulus=0` and half will have `stimulus=1`.

`mean_dprime`, `sd_dprime`, `r_dprime`
: The mean, standard deviation, and within-participant correlations of sensitivities of the signal detection agents to simulate

`mean_c`, `sd_c`, `r_c`
: The mean, standard deviation, and within-participant correlations of response bias of the signal detection agents to simulate

`mean_M`, `sd_M`, `r_M`
: The mean, standard deviation, and within-participant correlations of metacognitive efficiency of the agents. If `allow_negative_values==TRUE` (default), M-ratio is simulated on the logarithmic scale, where `0` indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers indicate metacognitive hyper-efficiency. Otherwise, M-ratio is modeled on its natural scale to allow negative values.

`mean_z_c2_0`, `mean_z_c2_1`
: Mean distance between confidence thresholds for `"0"` and `"1"` responses on the log_scale, such that `meta_c2_0 = meta_c - cumulative_sum(exp(z_c2_0))` and `meta_c2_1 = meta_c + cumulative_sum(exp(z_c2_1))`.

`sd_z_c2_0_condition`, `sd_z_c2_1_condition`
: SD of log distances across conditions between confidence thresholds for `"0"` and `"1"` responses on the log_scale.

`r_z_c2_0_condition`, `r_z_c2_1_condition`
: Correlation across conditions of log distances between confidence thresholds for `"0"` and `"1"` responses on the log_scale.

`sd_z_c2_0_confidence`, `sd_z_c2_1_confidence`
: SD of log distances across confidence levels between confidence thresholds for `"0"` and `"1"` responses on the log_scale.

`r_z_c2_0_confidence`, `r_z_c2_1_confidence`
: Correlation across confidence levels of log distances between confidence thresholds for `"0"` and `"1"` responses on the log_scale.

`metac_absolute`
: Determines how to fix the type 1 threshold for modeling confidence ratings. If `metac_absolute=TRUE`, `meta_c = c`. Otherwise, `meta_c = M * c`.

`allow_negative_values`
: If `allow_negative_values=FALSE` (default), M-ratio is simulated using a normal distribution on the logarithmic scale to prohibit negative values. If `allow_negative_values=TRUE`, then M-ratio is simulated on its natural scale, allowing negative values.

`summarize`
: Aggregate the data? If `summarize=FALSE`, returns a dataset with one row per observation. If `summarize=TRUE`, returns an aggregated dataset where `n` is the number of observations per response, accuracy, and confidence level.

`lcdf`
: The log cumulative distribution function of the underlying signal distribution. By default, uses a `normal(+/- dprime/2, 1)` distribution.

`lccdf`
: The log complement cumulative distribution function of the underlying signal distribution. By default, uses a `normal(+/- dprime/2, 1)` distribution.

Value

A simulated dataset of type 1 responses and confidence ratings, with columns:

- `trial`: the simulated trial number
- `stimulus`: the value of the stimulus on each trial (either `0` or `1`)
- `response`: the simulated type 1 response (either `0` or `1`)
- `correct`: whether `stimulus==response` (either `0` or `1`)
- `confidence`: the simulated type 2 response (from `1` to `length(c2_0_diff)+1`)
- `dprime:theta_2`: the simulated agent's parameter values If `summarize=TRUE`, the `trial` column is replaced with an `n` column indicating the number of simulated type 1/type 2 responses for each possible value.

Examples

```
sim_metad_participant_condition(10, 10)
```

`stanvars_metad` *Generate Stan code for the meta-d' model*

Description

Generate Stan code for the meta-d' model

Usage

```
stanvars_metad(
  K,
  distribution = "normal",
  metac_absolute = TRUE,
  categorical = FALSE,
  logit = TRUE
)
```

Arguments

`K`	The number of confidence levels
`distribution`	The noise distribution to use. Should be a parameter-free distribution, i.e., one that is mean-centered without additional variance/shape parameters. If the distribution is not already available in stan, you must additionally provide two functions to Stan (one for `<distribution>_lcdf` and one for `<distribution>_lccdf`).
`metac_absolute`	Should the type 2 criterion (metac) be fixed to the absolute type 1 criterion (c)? If `TRUE`, the model will set `metac = c`. Otherwise, it will set `metac = M * c`, such that the type 2 criterion is *relatively* equal to the type 1 criterion (i.e., `meta_c/meta_dprime = c/dprime`)
`categorical`	If `FALSE` (default), use the multinomial likelihood over aggregated data. If `TRUE`, use the categorical likelihood over individual trials.
`logit`	If `TRUE` (default), use the logit parameterization of the likelihood over the log joint response probabilities. If `FALSE`, use the standard parameterization of the likelihood over the actual joint response probabilities. In most cases, the logit parameterization should provide more stable numerical computations, but the standard parameterization might be preferable in some settings.

Value

A brms::stanvar object containing Stan code defining the likelihood for the metad' model with `K` confidence levels, signal distributed according to the distribution `distribution`, and where `metac = c` if `metac_absolute==TRUE`, and `metac = M*c` otherwise.

Examples

```
# create stancode for the meta-d' model
# using the normal distribution and 3 levels of confidence
stanvars_metad(3)

# create stancode for the meta-d' model with meta_c = M * c
stanvars_metad(3, metac_absolute = FALSE)

# create stancode for the meta-d' model with
# an alternative distribution
# note: cumulative distribution functions must be defined
# in R and in Stan using [brms::stanvar()]
stanvars_metad(4, distribution = "gumbel_min")
```

---

to_signed *Convert binary variable x between $\{0, 1\}$ and $\{-1, 1\}$*

Description

- `to_signed(x)` converts $x \in \{0, 1\}$ to $x' \in \{-1, 1\}$
- `to_unsigned(x)` converts $x \in \{-1, 1\}$ to $x' \in \{0, 1\}$

Usage

```
to_signed(x)

to_unsigned(x)
```

Arguments

`x`	A binary variable

Value

A signed (for `to_signed`) or unsigned (for `to_unsigned`) version of `x`

Examples

```
# should return `1`
to_signed(0)

# should return `1`
to_signed(1)

# should return `0`
to_unsigned(-1)
```

```
# should return `1`
to_unsigned(1)

# `to_signed` also works with objects `R` interprets as `0` or `1`
to_signed(10)

# `to_unsigned` also works with any signed integer
to_unsigned(-10)

# neither function works with factors
try(to_signed(factor(1)))
tryCatch(to_unsigned(factor(1)), warning = function(w) w)
```

`type1_draws` *Calculate posterior draws of type 1 response probabilities*

Description

Given a data frame and a meta-d' model, adds estimates of type 1 response probabilities (i.e., $P(R = r|S = s)$ or $P(R = r)$ for type 1 response R and stimulus S). For `type1_draws_metad` and `add_type1_draws_metad`, estimates are returned in a tidy tibble with one row per posterior draw. For `type1_rvars_metad` and `add_type1_rvars_metad`, parameters are returned as posterior::rvars, with one row per row in `newdata`.

Usage

```
type1_draws(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  by_stimulus = TRUE
)

add_type1_draws(newdata, object, ...)

type1_rvars(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  by_stimulus = TRUE
```

```
)

add_type1_rvars(newdata, object, ...)
```

Arguments

object	The brms model with the metad family
newdata	A data frame from which to generate posterior predictions
...	Additional arguments passed to tidybayes::add_epred_draws or tidybayes::add_epred_rvars
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
by_stimulus	If TRUE (default), calculate conditional type 1 response probabilities $P(R = r \mid S = s)$. Otherwise, calculate unconditional response probabilities $P(R = r)$ as an unweighted average over stimuli.

Value

a tibble containing posterior draws of model parameters with the following columns:

- .row: the row of newdata
- .chain, .iteration, .draw: for epred_draws_metad, identifiers for the posterior sample
- {.stimulus}, {.response}: identifiers for the response type
- .epred: probability of the type 1 response (optionally given the stimulus)

See Also

type1_probabilities()

Examples

```
newdata <- tidyr::tibble(.row = 1)

# obtain model predictions
# equivalent to `add_type1_draws(newdata, example_model())`
type1_draws(example_model(), newdata)

# obtain model predictions (`posterior::rvar`)
# equivalent to `add_type1_rvars(newdata, example_model(), by_stimulus = FALSE)`
type1_rvars(example_model(), newdata, by_stimulus = FALSE)
```

type1_probabilities *Calculate empirical type 1 response probabilities*

Description

Given a dataset data, determine the probability of each type 1 response, optionally conditional on stimulus.

Usage

```
type1_probabilities(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  by_stimulus = TRUE
)
```

Arguments

data	The data frame to aggregate
...	Grouping columns in data. These columns will be converted to factors.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
by_stimulus	If TRUE (default), calculate conditional type 1 response probabilities $P(R = r\|S = s)$. Otherwise, calculate unconditional response probabilities $P(R = r)$ as an unweighted average over stimuli.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.stimulus} (if by_stimulus=TRUE): the stimulus
- {.response}: the type 1 response
- n: the number of rows in data with the corresponding stimulus (if by_stimulus=TRUE) and response
- p: the proportion of rows in data with the corresponding response (per stimulus if by_stimulus=TRUE)

See Also

type1_draws()

Examples

```
# calculate response probabilities by stimulus
type1_probabilities(example_data())

# calculate response probabilities by condition, averaging over stimuli
type1_probabilities(sim_metad_condition(), condition, by_stimulus = FALSE)
```

---

type2_draws *Calculate posterior draws of type 2 response probabilities*

Description

Given a data frame and a meta-d' model, adds estimates of type 2 response probabilities (i.e., $P(C = c|S = s, R = r)$, $P(C = c|S = s)$, $P(C = c|R = r)$ or $P(C = c)$ for stimulus S), type 1 response R, and type 2 response C. For type2_draws_metad and add_type2_draws_metad, estimates are returned in a tidy tibble with one row per posterior draw. For type2_rvars_metad and add_type2_rvars_metad, parameters are returned as posterior::rvars, with one row per row in newdata.

Usage

```
type2_draws(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  by_stimulus = TRUE,
  by_response = TRUE
)

add_type2_draws(newdata, object, ...)

type2_rvars(
  object,
  newdata,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  by_stimulus = TRUE,
  by_response = TRUE
```

```
)

add_type2_rvars(newdata, object, ...)
```

Arguments

object	The brms model with the metad family
newdata	A data frame from which to generate posterior predictions
...	Additional arguments passed to tidybayes::add_epred_draws or tidybayes::add_epred_rvars
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
by_stimulus	If TRUE (default), calculate type 2 response probabilities separately by stimulus. Otherwise, calculate unconditional type 2 response probabilities as an unweighted average over stimuli.
by_response	If TRUE (default), calculate type 2 response probabilities separately by type 1 response. Otherwise, calculate unconditional type 2 response probabilities as an unweighted average over type 1 responses.

Value

a tibble containing posterior draws of model parameters with the following columns:

- .row: the row of newdata
- .chain, .iteration, .draw: for epred_draws_metad, identifiers for the posterior sample
- {.stimulus}, {.response}, {.confidence}: identifiers for the response type
- .epred: probability of the type 1 and type 2 response given the stimulus, $P(R, C \mid S)$

See Also

type2_probabilities()

Examples

```
newdata <- tidyr::tibble(.row = 1)

# obtain model predictions
# equivalent to `add_type2_draws(newdata, example_model())`
type2_draws(example_model(), newdata)

# obtain model predictions (`posterior::rvar`)
# equivalent to `add_type2_rvars(newdata, example_model(), by_stimulus = FALSE)`
type2_rvars(example_model(), newdata, by_stimulus = FALSE)
```

type2_probabilities *Calculate empirical type 2 response probabilities*

Description

Given a dataset data, determine the probability of each type 2 response, optionally conditional on stimulus and/or type 1 response.

Usage

```
type2_probabilities(
  data,
  ...,
  .stimulus = "stimulus",
  .response = "response",
  .confidence = "confidence",
  .joint_response = "joint_response",
  K = NULL,
  by_stimulus = TRUE,
  by_response = TRUE
)
```

Arguments

data	The data frame to aggregate
...	Grouping columns in data. These columns will be converted to factors.
.stimulus	The name of "stimulus" column
.response	The name of "response" column
.confidence	The name of "confidence" column
.joint_response	The name of "joint_response" column
K	The number of confidence levels in data. If NULL, this is estimated from data using the maximum value of either the confidence column or joint response column.
by_stimulus	If TRUE (default), calculate type 2 response probabilities conditional on stimulus.
by_response	If TRUE (default), calculate type 2 response probabilities conditional on type 1 response.

Value

A tibble with columns:

- ...: the grouping columns in data
- {.stimulus} (if by_stimulus=TRUE): the stimulus
- {.response} (if by_response=TRUE): the type 1 response

- {.confidence}: the type 2 response
- {.joint_response} (if by_response=TRUE): the joint type 1/type 2 response
- n: the number of rows in data with the corresponding stimulus (if by_stimulus=TRUE), response (if by_response=TRUE), and confidence
- p: the proportion of rows in data with the corresponding response (per stimulus if by_stimulus=TRUE and per response if by_response=TRUE)

See Also

type2_draws()

Examples

```
# calculate type 2 response probabilities by stimulus
type2_probabilities(example_data())

# calculate type 2 response probabilities by condition, averaging over stimuli
type2_probabilities(sim_metad_condition(), condition, by_stimulus = FALSE)
```

Index