

# Package ‘hmetad’

May 15, 2026

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

**Version** 0.1.2

**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](https://doi.org/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** 8.0.0

**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](mailto:kevin.o'neill@ucl.ac.uk)>

**Repository** CRAN

**Date/Publication** 2026-05-15 20:50:02 UTC

## Contents

aggregate_metad . . . . .	2
cor_matrix . . . . .	4
cov_matrix . . . . .	5
epred_draws_metad . . . . .	5
example_data . . . . .	6
example_model . . . . .	7
fit_metad . . . . .	8
joint_response . . . . .	10
linpred_draws_metad . . . . .	11
mean_confidence_draws . . . . .	12
metac2_parameters . . . . .	14
metacognitive_bias_draws . . . . .	15
metad . . . . .	16
metad_pmf . . . . .	17
normal_lcdf . . . . .	18
predicted_draws_metad . . . . .	19
response_probabilities . . . . .	20
rmatrixnorm . . . . .	21
roc1_draws . . . . .	22
roc2_draws . . . . .	23
sim_metad . . . . .	25
sim_metad_condition . . . . .	26
sim_metad_participant . . . . .	28
sim_metad_participant_condition . . . . .	30
stanvars_metad . . . . .	32
to_signed . . . . .	33
<b>Index</b>	<b>35</b>

---

aggregate_metad	<i>Aggregate data by response, confidence, and other columns</i>
-----------------	--

---

## 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
)

```

**Arguments**

<code>data</code>	The data frame to aggregate
<code>...</code>	Grouping columns in data. These columns will be converted to factors.
<code>.stimulus</code>	The name of "stimulus" column
<code>.response</code>	The name of "response" column
<code>.confidence</code>	The name of "confidence" column
<code>.joint_response</code>	The name of "joint_response" column
<code>.name</code>	The name of the resulting column containing trial counts
<code>K</code>	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.

**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  $C$ ):

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

**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()

```

---

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	<i>Generate a covariance matrix.</i>
------------	--------------------------------------

---

**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  $N = \text{length}(S)$ .

**Examples**

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

---

epred_draws_metad	<i>Obtain posterior draws of joint response probabilities</i>
-------------------	---

---

**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, ...)
add_epred_draws_metad(newdata, object, ...)
epred_rvars_metad(object, newdata, ...)
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 <a href="#">tidybayes::add_epred_draws</a> or <a href="#">tidybayes::add_epred_rvars</a>

**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,  $P(R, C | S)$

**See Also**

[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,
  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 <code>metad(K)</code> .
data	A tibble containing the data to fit the model. <ul style="list-style-type: none"> <li>If <code>aggregate==TRUE</code>, data should have one row per observation with columns <code>stimulus</code>, <code>response</code>, <code>confidence</code>, and any other variables in <code>formula</code></li> <li>If <code>aggregate==FALSE</code>, 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 <code>aggregate_metad()</code>).</li> </ul>
...	Additional parameters passed to the <code>brm</code> function.

aggregate	If TRUE, automatically aggregate data by the variables included in formula using <code>aggregate_metad()</code> . 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 <code>dprime</code> .
metac_absolute	If TRUE, fix the type 2 criterion to be equal to the type 1 criterion. Otherwise, equate the criteria relatively such that $metac/metadprime = c/dprime$ .
stanvars	Additional stanvars to pass to the model code, for example to define an alternative distribution or a custom model prior (see <code>brms::stanvar()</code> ).
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_response	<i>Convert between separate and joint type 1/type 2 responses</i>
----------------	---

---

### 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**

<code>object</code>	The brms model with the metad family
<code>newdata</code>	A data frame from which to generate posterior predictions
<code>...</code>	Additional arguments passed to <code>tidybayes::add_linpred_draws</code> or <code>tidybayes::add_linpred_rvars</code>
<code>pivot_longer</code>	Return the draws in long format? <ul style="list-style-type: none"> <li>• if TRUE, resulting data frame has one row per posterior draw per model parameter</li> <li>• if FALSE (default), resulting data frame has one row per posterior draw</li> </ul>

**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_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,
  ...,
  by_stimulus = TRUE,
  by_response = TRUE,
  by_correct = FALSE
)
```

```
add_mean_confidence_draws(newdata, object, ...)
```

```
mean_confidence_rvars(
  object,
  newdata,
  ...,
  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 <a href="#">tidybayes::epred_draws</a> or <a href="#">tidybayes::epred_rvars</a>
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**

`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)
```

---

<code>metac2_parameters</code>	<i>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.</i>
--------------------------------	--

---

**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- $\Delta$ , an index of metacognitive bias. meta- $\Delta$  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, ..., by_response = TRUE)
```

```
add_metacognitive_bias_draws(newdata, object, ...)
```

```
metacognitive_bias_rvars(object, newdata, ..., 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 <code>tidybayes::epred_draws</code> or <code>tidybayes::epred_rvars</code>
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- $\Delta$  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	<i>brms family for the metad' model</i>
-------	---

---

**Description**

brms family for the metad' model

**Usage**

```
metad(K, distribution = "normal", metac_absolute = TRUE, categorical = 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.

### 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	<i>Generate (log) probability simplex over the joint type 1/type 2 responses</i>
-----------	--

---

### 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 meta-d' 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 meta-d' 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

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

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, ...)
add_predicted_draws_metad(newdata, object, ...)
predicted_rvars_metad(object, newdata, ...)
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 <code>tidybayes::add_predicted_draws</code> or <code>tidybayes::add_predicted_rvars</code>

**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)
```

---

response\_probabilities

*Compute joint response probabilities from aggregated counts*

---

**Description**

Compute joint response probabilities from aggregated counts

**Usage**

```
response_probabilities(counts)
```

**Arguments**

`counts` A vector (or matrix) of counts of joint type 1/type 2 responses as provided by [aggregate\\_metad](#)

**Details**

For response  $R$ , confidence  $C$ , stimulus  $S$ , and number of confidence levels  $K$ , counts should be a vector (or matrix with rows) of the form:

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

Returns a vector (or matrix with rows) of the form:

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

**Value**

A vector (or matrix) of response probabilities  $P(R, C | S)$

**Examples**

```
# Aggregate responses from simulated data
d <- sim_metad() |> aggregate_metad()

# Compute conditional response probabilities
response_probabilities(d$N)

# Also works on matrices
matrix(rep(1, 16), nrow = 2) |> response_probabilities()
```

---

 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_draws	<i>Obtain posterior draws of the pseudo type 1 receiver operating characteristic (ROC) curve.</i>
------------	---

---

## 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, ..., bounds = FALSE)
```

```
add_roc1_draws(newdata, object, ...)
```

```
roc1_rvars(object, newdata, ..., 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 <a href="#">tidybayes::epred_draws</a> or <a href="#">tidybayes::epred_rvars</a>
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

[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\_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, ..., bounds = FALSE)
```

```
add_roc2_draws(newdata, object, ...)
```

```
roc2_rvars(object, newdata, ..., bounds = FALSE)
```

```
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 <a href="#">tidybayes::epred_draws</a>
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 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**

[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  $d_{\text{prime}}$ , response bias  $c$ , metacognitive efficiency  $\log_M$ , 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,
  log_M = 0,
  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**

<code>N_trials</code>	Total number of trials to simulate. Half of these trials will have <code>stimulus=0</code> and half will have <code>stimulus=1</code> .
<code>dprime</code>	The sensitivity of the signal detection agent to simulate
<code>c</code>	The response bias of the signal detection agent to simulate
<code>log_M</code>	The metacognitive efficiency of the agent on the logarithmic scale, where 0 indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers indicate metacognitive hyper-efficiency.
<code>c2_0_diff, c2_1_diff</code>	Distances between confidence thresholds for "0" and "1" responses, such that $\text{meta}_{c2\_0} = \text{meta}_c - \text{cumsum}(c2\_0\_diff)$ and $\text{meta}_{c2\_1} = \text{meta}_c + \text{cumsum}(c2\_1\_diff)$ .
<code>metac_absolute</code>	Determines how to fix the type 1 threshold for modeling confidence ratings. If <code>metac_absolute=TRUE</code> , $\text{meta}_c = c$ . Otherwise, $\text{meta}_c = M * c$ .
<code>summarize</code>	Aggregate the data? <ul style="list-style-type: none"> <li>If <code>FALSE</code>, returns a dataset with one row per observation.</li> <li>If <code>summarize=TRUE</code>, returns an aggregated dataset where <code>n</code> is the number of observations per response, accuracy, and confidence level.</li> </ul>
<code>lcdf, lccdf</code>	The log (complement) cumulative distribution function of the underlying signal distribution. By default, uses a normal( $\pm d_{\text{prime}}/2$ , 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 log\_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),
  log_M = rep(0, 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
log_M	The metacognitive efficiency of the agent on the logarithmic scale, where 0 indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers 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  $d_{\text{prime}}$ , response bias  $c$ , metacognitive efficiency  $\log_M$ , 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,
  mu_dprime = 1,
  sd_dprime = 0.5,
  mu_c = 0,
  sd_c = 0.5,
  mu_log_M = 0,
  sd_log_M = 0.5,
  mu_z_c2_0 = rep(-1, 3),
  sd_z_c2_0 = rep(0.1, 3),
  r_z_c2_0 = diag(3),
  mu_z_c2_1 = rep(-1, 3),
  sd_z_c2_1 = rep(0.1, 3),
  r_z_c2_1 = diag(3),
  metac_absolute = TRUE,
  summarize = FALSE,
  lcdf = normal_lcdf,
  lccdf = normal_lccdf
)
```

### Arguments

<code>N_trials, N_participants</code>	Total number of participants and trials to simulate per participant. Half of these trials will have <code>stimulus=0</code> and half will have <code>stimulus=1</code> .
<code>mu_dprime, sd_dprime</code>	The mean and standard deviation of sensitivities of the signal detection agents to simulate
<code>mu_c, sd_c</code>	The mean and standard deviation of response bias of the signal detection agents to simulate
<code>mu_log_M, sd_log_M</code>	The mean and standard deviation of metacognitive efficiency of the agents on the logarithmic scale, where 0 indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers indicate metacognitive hyper-efficiency.

mu_z_c2_0, mu_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.
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, mu_dprime = 2, mu_log_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  $d_{\text{prime}}$ , response bias  $c$ , metacognitive efficiency  $\log_M$ , 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,
  mu_dprime = rep(1, 2),
  sd_dprime = rep(0.5, 2),
  r_dprime = diag(2),
  mu_c = rep(0, 2),
  sd_c = rep(0.5, 2),
  r_c = diag(2),
  mu_log_M = rep(0, 2),
  sd_log_M = rep(0.5, 2),
  r_log_M = diag(2),
  mu_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),
  mu_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,
  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  $stimulus=1$ .

mu_dprime, sd_dprime, r_dprime	The mean, standard deviation, and within-participant correlations of sensitivities of the signal detection agents to simulate
mu_c, sd_c, r_c	The mean, standard deviation, and within-participant correlations of response bias of the signal detection agents to simulate
mu_log_M, sd_log_M, r_log_M	The mean, standard deviation, and within-participant correlations of metacognitive efficiency of the agents on the logarithmic scale, where $\theta$ indicates optimal metacognitive sensitivity, negative numbers indicate metacognitive inefficiency, and positive numbers indicate metacognitive hyper-efficiency.
mu_z_c2_0, mu_z_c2_1	Mean distance between confidence thresholds for "0" and "1" responses on the log_scale, such that $\text{meta\_c2\_0} = \text{meta\_c} - \text{cumulative\_sum}(\exp(z\_c2\_0))$ and $\text{meta\_c2\_1} = \text{meta\_c} + \text{cumulative\_sum}(\exp(z\_c2\_1))$ .
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 <code>metac_absolute=TRUE</code> , $\text{meta\_c} = c$ . Otherwise, $\text{meta\_c} = M * c$ .
summarize	Aggregate the data? If <code>summarize=FALSE</code> , returns a dataset with one row per observation. If <code>summarize=TRUE</code> , returns an aggregated dataset where <code>n</code> 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 <code>normal(+/- dprime/2, 1)</code> distribution.
lccdf	The log complement cumulative distribution function of the underlying signal distribution. By default, uses a <code>normal(+/- dprime/2, 1)</code> 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	<i>Generate Stan code for the meta-d' model</i>
----------------	---

---

## Description

Generate Stan code for the meta-d' model

## Usage

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

## Arguments

<code>K</code>	The number of confidence levels
<code>distribution</code>	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 <code>&lt;distribution&gt;_lcdf</code> and one for <code>&lt;distribution&gt;_lccdf</code> ).
<code>metac_absolute</code>	Should the type 2 criterion (metac) be fixed to the absolute type 1 criterion (c)? If TRUE, the model will set <code>metac = c</code> . Otherwise, it will set <code>metac = M * c</code> , such that the type 2 criterion is <i>relatively</i> equal to the type 1 criterion (i.e., <code>meta_c/meta_dprime = c/dprime</code> )
<code>categorical</code>	If FALSE (default), use the multinomial likelihood over aggregated data. If TRUE, use the categorical likelihood over individual trials.
<code>logit</code>	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 meta-d' model with  $K$  confidence levels, signal distributed according to the distribution distribution, and where  $\text{metac} = c$  if  $\text{metac\_absolute} == \text{TRUE}$ , and  $\text{metac} = M * c$  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	<i>Convert binary variable <math>x</math> between <math>\{0, 1\}</math> and <math>\{-1, 1\}</math></i>
-----------	--

---

**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)
```

# Index

add\_epred\_draws\_metad  
    (epred\_draws\_metad), 5  
add\_epred\_rvars\_metad  
    (epred\_draws\_metad), 5  
add\_linpred\_draws\_metad  
    (linpred\_draws\_metad), 11  
add\_linpred\_rvars\_metad  
    (linpred\_draws\_metad), 11  
add\_mean\_confidence\_draws  
    (mean\_confidence\_draws), 12  
add\_mean\_confidence\_rvars  
    (mean\_confidence\_draws), 12  
add\_metacognitive\_bias\_draws  
    (metacognitive\_bias\_draws), 15  
add\_metacognitive\_bias\_rvars  
    (metacognitive\_bias\_draws), 15  
add\_predicted\_draws\_metad  
    (predicted\_draws\_metad), 19  
add\_predicted\_rvars\_metad  
    (predicted\_draws\_metad), 19  
add\_roc1\_draws (roc1\_draws), 22  
add\_roc1\_rvars (roc1\_draws), 22  
add\_roc2\_draws (roc2\_draws), 23  
add\_roc2\_rvars (roc2\_draws), 23  
aggregate\_metad, 2, 20  
aggregate\_metad(), 8, 9  
  
brms::brm(), 8  
brms::stanvar, 33  
brms::stanvar(), 9  
  
cor\_matrix, 4  
cov\_matrix, 5  
  
epred\_draws\_metad, 5  
epred\_rvars\_metad (epred\_draws\_metad), 5  
example\_data, 6, 7  
example\_model, 7  
  
fit\_metad, 8  
  
fit\_metad(), 7  
  
joint\_response, 10  
  
linpred\_draws\_metad, 11  
linpred\_rvars\_metad  
    (linpred\_draws\_metad), 11  
  
mean\_confidence\_draws, 12  
mean\_confidence\_rvars  
    (mean\_confidence\_draws), 12  
metac2\_parameters, 14  
metacognitive\_bias\_draws, 15  
metacognitive\_bias\_rvars  
    (metacognitive\_bias\_draws), 15  
metad, 16  
metad\_pmf, 17  
  
normal\_lccdf (normal\_lcdf), 18  
normal\_lcdf, 18  
  
posterior::rvar, 5, 11, 12, 15, 19, 22  
predicted\_draws\_metad, 19  
predicted\_rvars\_metad  
    (predicted\_draws\_metad), 19  
  
response\_probabilities, 20  
rmatrixnorm, 21  
roc1\_draws, 22  
roc1\_rvars (roc1\_draws), 22  
roc2\_draws, 23  
roc2\_rvars (roc2\_draws), 23  
  
sim\_metad, 25  
sim\_metad(), 7  
sim\_metad\_condition, 26  
sim\_metad\_participant, 28  
sim\_metad\_participant\_condition, 30  
stanvars\_metad, 32  
  
tidybayes::add\_epred\_draws, 6

tidybayes::add\_epred\_rvars, [6](#)  
tidybayes::add\_linpred\_draws, [11](#)  
tidybayes::add\_linpred\_rvars, [11](#)  
tidybayes::add\_predicted\_draws, [19](#)  
tidybayes::add\_predicted\_rvars, [19](#)  
tidybayes::epred\_draws, [13](#), [15](#), [22](#), [24](#)  
tidybayes::epred\_draws(), [6](#), [14](#), [23](#), [24](#)  
tidybayes::epred\_rvars, [13](#), [15](#), [22](#)  
tidybayes::epred\_rvars(), [6](#), [14](#), [23](#), [24](#)  
tidybayes::linpred\_draws(), [12](#), [16](#)  
tidybayes::linpred\_rvars(), [12](#), [16](#)  
tidybayes::predicted\_draws(), [20](#)  
tidybayes::predicted\_rvars(), [20](#)  
to\_signed, [33](#)  
to\_unsigned (to\_signed), [33](#)  
type1\_response (joint\_response), [10](#)  
type2\_response (joint\_response), [10](#)