Package 'uSDT'

November 5, 2026

Type Package

Title Hierarchical Signal Detection Theory Models for Unconscious Processing

Version 0.2.0

Description Fits hierarchical signal detection theory (SDT) models to paired awareness and influence measures in research on unconscious processing. Estimates mean sensitivities, their latent correlation and influence sensitivity at zero awareness sensitivity, using maximum likelihood with 'lme4' or Bayesian inference with 'brms'. Provides Bayes factors, equivalence tests, reliability estimates and study planning for power, precision and cost.

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 4.1)

Imports grid, parallel, stats, utils, lme4, ggplot2, gtable, rlang, brms, posterior, logspline, callr, cli, withr (>= 3.0.3)

Suggests testthat (>= 3.1.7), knitr, rmarkdown, BH, RcppEigen, cmdstanr

Additional_repositories https://stan-dev.r-universe.dev

SystemRequirements C++17 compiler (for Bayesian model compilation)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

URL https://github.com/RicardoReySaez/uSDT, https://ricardoreysaez.github.io/uSDT/

BugReports https://github.com/RicardoReySaez/uSDT/issues

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Ricardo Rey-Sáez [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6739-2035>),
Francisco Garre-Frutos [aut] (ORCID:
<https://orcid.org/0000-0001-9810-186X>),
Alicia Franco-Martínez [aut] (ORCID:
<https://orcid.org/0000-0002-9710-1240>),
Miguel Vadillo [aut] (ORCID: <https://orcid.org/0000-0001-8421-816X>)

Maintainer Ricardo Rey-Sáez <ricardoreysaez95@gmail.com>

Repository CRAN

Date/Publication 2026-11-05 20:30:02 UTC

Contents

hsdt 2
meyen_split 6
plot.hsdt 7
plot.usdt_bf 10
plot.usdt_priors 11
plot_power 12
sdt_moments 14
usdt_aggregate 16
usdt_bf 17
usdt_boot 21
usdt_data 23
usdt_equivalence 26
usdt_hypotheses 29
usdt_mdes 32
usdt_plan 35
usdt_population 38
usdt_power 41
usdt_priors 44
usdt_reliability 47
vadillo_awareness 49
vadillo_cuing 50

Index **51**

`hsdt` *Fit a hierarchical signal detection theory model*

Description

Models paired awareness and influence sensitivities and criteria, their population means and between-subject variation. Frequentist estimation uses `lme4::glmer()`. Bayesian estimation uses brms and Stan.

Usage

```
hsdt(
  data,
  estimation = c("frequentist", "bayesian"),
  fix_criteria = c("auto", "none"),
  level = 0.95,
  optimizer = "bobyqa",
  priors = usdt_priors(),
  unequal_variances = FALSE,
  latent = c("correlation", "regression"),
  backend = getOption("brms.backend", "rstan"),
  ...
)

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

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

Arguments

`data`	A `usdt_data` object from `usdt_data_tasks()` or `usdt_data_long()`.
`estimation`	Estimation framework: `"frequentist"` (maximum likelihood via Laplace approximation with lme4) or `"bayesian"` (Hamiltonian Monte Carlo with Stan, through brms).
`fix_criteria`	How to handle response criteria. `"auto"` fixes to zero a dichotomized task's criterion when its mean absolute descriptive subject criterion is below 0.02. `"none"` estimates all criteria.
`level`	Confidence level for Wald intervals, or credible level for the posterior intervals of a Bayesian fit (default is 0.95).
`optimizer`	Primary optimizer passed to `lme4::glmerControl()`. Alternative optimizers are automatically evaluated if the default fails to converge or produces a singular fit. Frequentist estimation only.
`priors`	A `usdt_priors` object from `usdt_priors()`. Bayesian estimation only.
`unequal_variances`	Logical. Estimate the standard deviation of the signal distribution of each task instead of fixing it to that of the noise distribution. Bayesian estimation only.
`latent`	How the Bayesian model writes the relation between the two latent sensitivities, which decides where it places the prior on the location of the influence sensitivity: on its mean, by their latent correlation (`"correlation"`, the default), or on the intercept of the latent regression of the influence on the awareness sensitivity (`"regression"`). Both keep the priors on the awareness mean, the standard deviations and the correlation. Bayesian estimation only.
`backend`	Stan interface brms uses for Bayesian estimation: `"rstan"` (available from CRAN) or `"cmdstanr"` (installed from the Stan R-universe together with Cmd-

Stan). As in brms, the default is the `brms.backend` option, or `"rstan"` when it is not set. Bayesian estimation only.

`...` For frequentist estimation, additional arguments passed to `lme4::glmer()`. The model formula, family and data remain managed by the package. For Bayesian estimation, sampling options: `chains` (default 4), `iter` (iterations per chain including warmup, default 6000), `warmup` (default 1000, so each chain keeps 5000 draws), `cores` (chains run at once, default `min(parallel::detectCores() - 2, 4)`, at least 1), `seed`, `control = list(adapt_delta, max_treedepth)` (default 0.95 and 10) and `refresh` (iterations between progress updates of each chain, default 1% of `iter`, and `0` hides the progress bar).

`object` An `hsdt` object.

`x` An `hsdt` object.

Details

Hypotheses evaluated by default:

- **H1:** Mean sensitivity difference between tasks.
- **H2:** Latent correlation of sensitivities across participants.
- **H3:** Latent regression of influence on awareness sensitivity. Its intercept reflects expected influence performance when awareness sensitivity is zero ($d'_{\text{Awareness}} = 0$).

When sample sizes or trial counts are low, variance components can reach singular boundaries. In these cases, the function issues a warning, and parametric bootstrap intervals can be calculated using `usdt_boot()`, or the model can be fitted with `estimation = "bayesian"`.

Value

An object of class `hsdt` containing:

- `$fit`: The underlying fit: a `glmerMod` object from lme4, or a `brmsfit` object from brms (see `brms::prior_summary()` and `brms::pp_check()`).
- `$tests`: Summary table for hypotheses H1, H2, and H3.
- `$design`: Summary of the model specification.
- `$diagnostics`: Convergence diagnostics.

A frequentist fit also contains `$pars`, the parameter estimates and their joint covariance. A Bayesian fit also contains `$priors`. Its posterior draws are those of `$fit`, where `b_dA_Intercept` and `b_dI_Intercept` are the mean d' of the awareness and influence tasks, `sd_subj__dA_Intercept` and `sd_subj__dI_Intercept` their SDs, `cor_subj__dA_Intercept__dI_Intercept` their correlation, `b_cA_Intercept` and `b_cI_Intercept` the mean of an estimated criterion (on the classical scale), and `b_sA_Intercept` and `b_sI_Intercept` the signal SDs under unequal variances.

Bayesian estimation

The Bayesian model places its priors on the mean, standard deviation and correlation of the two sensitivities and on the criteria. See `usdt_priors()`. The slope and intercept of the latent regression are derived from them draw by draw. The hypothesis table reports the posterior mean and standard deviation, a central credible interval and a two-sided posterior p-value, $2\min\{P(\theta > 0 \mid y), P(\theta <$

$0 \mid y)\}$. The p-value falls below `1 - level` exactly when zero lies outside the credible interval, so with the default `level = 0.95` a p-value below .05 means that the 95% credible interval excludes zero.

With `latent = "regression"`, the prior of the influence mean moves to the intercept β_0 of the latent regression, and the influence mean follows as $\mu_I = \beta_0 + \rho\,(\sigma_I/\sigma_A)\,\mu_A$. The intercept tested by H3 then has its specified prior, and the correlation (H2) keeps its own. The difference between the means (H1) has an induced, heavy-tailed prior instead, and because the data fix μ_I, the intercept prior also informs the posterior of ρ. See `usdt_priors()` and `plot.usdt_priors()`. The fit samples the same parameters and reports the same hypotheses either way. Near-zero awareness SDs can produce difficult posterior geometry and divergences. A prior with less weight near zero may help, if substantively justified. Inspect its implications with `plot.usdt_priors()`.

With `unequal_variances = TRUE`, the noise distribution of each task is $N(-d/2, 1)$ and the signal distribution $N(d/2, \sigma_s^2)$, so d is the distance between their means in units of the noise standard deviation, and H1-H3 compare these d. When σ_s differs between the tasks, d no longer measures the same discriminability in both: the measure comparable with an equal-variance d' is $d_a = d\sqrt{2/(1+\sigma_s^2)}$, and the mean d_a of a task is its mean d times that factor, which can be computed from the draws of `b_dA_Intercept` and `b_sA_Intercept` (and their influence counterparts) in `$fit`. The correlation (H2) is the same on either scale.

The model is a brms model of the aggregated counts, with one prior per parameter and task. It is compiled the first time it is used with a given set of priors, which needs a C++ toolchain, and stays in a per-user cache directory for later sessions. Inspect the reported divergences, tree-depth warnings, R-hat and effective sample sizes before interpreting results.

See Also

`usdt_data_tasks()`, `usdt_priors()`, `usdt_tests()`, `usdt_boot()`, `plot.hsdt()`

Examples

```
# Contextual cuing data from Vadillo et al. (2025)
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

m <- hsdt(d)

# Full summary table with SDT parameters and hypothesis tests
summary(m)

# Inspect the model formula (influence criterion omitted by default)
m$design$formula
```

```
## Not run:
# Requires a C++ toolchain and the BH and RcppEigen headers for rstan.
# The first call compiles the model.
mb <- hsdt(d, estimation = "bayesian", cores = 2, seed = 1)
summary(mb)

## End(Not run)
```

`meyen_split` *Dichotomize response times into binary choices*

Description

Splits response times (or other continuous measures) at each subject's overall median, following the preprocessing approach of Meyen et al. (2022). The median is calculated across all trials for each participant without distinguishing between stimulus conditions or other covariates. This produces a binary outcome that allows response times to be mapped onto a Signal Detection Theory sensitivity metric (d').

Usage

```
meyen_split(x, by, signal = c("faster", "slower"), ties = c("noise", "random"))
```

Arguments

`x`	Numeric vector of continuous values, typically response times.
`by`	Vector identifying the subject for each observation in `x`. Medians are computed independently for each participant.
`signal`	Character string specifying which side of the median will be treated as the "signal" response under an SDT framework. Use `"faster"` when the target condition speeds up responses (e.g., facilitatory priming, spatial cueing) or `"slower"` when it slows responses down (e.g., interference, Stroop-like effects).
`ties`	How to handle trials that match the subject's median exactly. `"noise"` assigns them to the noise category (0). `"random"` breaks ties at random, keeping cell proportions as balanced as possible.

Details

With an odd number of trials (n), a dataset cannot be split into two equal halves because the median falls exactly on an observed trial. Setting `ties = "noise"` assigns this middle trial to noise, producing a signal proportion of $(n-1)/(2 \cdot n)$ and slightly shifting the response criterion. In practice, this difference $(1/(2 \cdot n))$ is negligible, but setting `ties = "random"` resolves ties probabilistically to avoid any systematic directional bias.

Value

An integer vector of `0`s (noise response) and `1`s (signal response) matching the length of `x`. Missing values (`NA`) are preserved.

References

Meyen, S., Zerweck, I. A., Amado, C., von Luxburg, U., & Franz, V. H. (2022). Advancing research on unconscious priming: When can scientists claim an indirect task advantage? *Journal of Experimental Psychology: General*, 151(1), 65–81. doi:10.1037/xge0001065

Examples

```
rt   <- c(320, 410, 295, 500, 380, 450)
subj <- rep(c("s1", "s2"), each = 3)
meyen_split(rt, by = subj)
```

---

plot.hsdt *Diagnostic and analytical plots for hierarchical SDT models*

Description

Generates diagnostic visualizations for a fitted hierarchical SDT model, including observed versus latent regression (H3), shrinkage patterns, participant-level caterpillar intervals, and model-implied ROC curves.

Usage

```
## S3 method for class 'hsdt'
plot(
  x,
  type = c("regression", "shrinkage", "caterpillar", "roc"),
  subject_id = NULL,
  band = TRUE,
  population_reference = TRUE,
  observed_se = NULL,
  nrow = NULL,
  caption = TRUE,
  ...
)
```

Arguments

`x`	An `hsdt` object fitted by `hsdt()`, frequentist or Bayesian.
`type`	Character string indicating the plot type: • `"regression"`: Compares the observed OLS regression with the latent regression line (H3).

- `"shrinkage"`: Connects observed d' to model-implied d' for each participant across bivariate contours.
- `"caterpillar"`: Compares observed d' and model-implied d' with confidence intervals for each participant alongside zero.
- `"roc"`: Shows model-implied ROC curves with estimated criteria.

`subject_id` Identifier for a specific participant when `type = "roc"`. If omitted, displays population-level curves.

`band` Logical. Show confidence bands around regression lines or ROC curves (default is `TRUE`).

`population_reference` Logical. When plotting an individual ROC curve, add population-average curves as dashed reference lines.

`observed_se` Variance formulation for empirical intervals in `"caterpillar"` plots: `"gg"` (Gourevitch & Galanter, 1967, default) or `"miller"` (Miller, 1996).

`nrow` Number of rows of the two panels when `type = "regression"`: `1` (default) places them side by side as squares, `2` stacks the observed panel above the model one, each as wide as the figure.

`caption` Logical. Show the caption that explains the plot under it (default `TRUE`). The shrinkage plot has none.

`...` Additional arguments passed to underlying plotting methods.

Value

A `ggplot` object. Its underlying data frame is stored in `$data`.

Regression plot (`type = "regression"`)

Compares observed and latent associations across two panels sharing axes, side by side or, with `nrow = 2`, one above the other. The first panel shows observed d' values and an ordinary least-squares line. When awareness task reliability is low, trial-level sampling noise attenuates this observed slope toward zero. The second panel plots the latent regression line (d'_I on d'_A) from H3, correcting for measurement error. The value of each line at $d'_A = 0$ marks the intercept testing for unconscious processing, and both panels display it the same way: an open circle at the point estimate with a vertical line spanning its confidence interval. The observed marker is the least-squares intercept and the latent marker is its measurement-error-corrected counterpart, so the two panels place the same hypothesis side by side. Confidence bands are computed via the delta method or bootstrap replicates when `usdt_boot()` is present.

Shrinkage plot (`type = "shrinkage"`)

Connects each participant's observed d' (from `sdt_moments()` with Hautus correction) to their model-implied d'. The lower the reliability of the measures, the higher the shrinkage of observed estimates toward the group-level mean.

Caterpillar plot (`type = "caterpillar"`)

Plots observed d' and model-implied d' with confidence intervals for every participant. Empirical intervals use normal approximations based on `observed_se`. Model-implied intervals incorporate uncertainty from population means, variance components, and participant random effects.

ROC plot (`type = "roc"`)

Displays model-implied ROC curves for an average participant or a specific individual, with points marking the estimated response criteria.

Bayesian fits

For a fit with `estimation = "bayesian"`, every model-estimated value is a posterior mean and every model band is a central credible interval, computed draw by draw: the latent line and its band, each subject's sensitivity in the shrinkage and caterpillar plots, and the ROC curves and operating points. The latent p-values are the two-sided posterior p-values of the hypothesis table. Under unequal variances the ROC curves bend with the signal standard deviation of each draw.

See Also

`hsdt()`, `sdt_moments()`, `usdt_boot()`

Examples

```
# Contextual cuing data from Vadillo et al. (2025)
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

fit <- hsdt(d)

# 1. Observed vs. latent regression (H3), side by side or stacked
plot(fit, type = "regression")
plot(fit, type = "regression", nrow = 2)

# 2. Bivariate shrinkage toward the group mean
plot(fit, type = "shrinkage")

# 3. Participant-level intervals (observed vs. model-implied)
plot(fit, type = "caterpillar")

# 4. Model-implied ROC curves
```

```
plot(fit, type = "roc")
plot(fit, type = "roc", subject_id = "2001", population_reference = TRUE)
```

`plot.usdt_bf` *Plot the prior and posterior of tested quantities*

Description

Draws, for each hypothesis of a `usdt_bf()` result, the posterior density (solid) and the prior density (dashed) of the tested quantity, which names the x axis of its panel: $\Delta d'$, ρ, or β_0 and β_1 for the intercept and slope of the latent regression. The panel title says whether the hypothesis is the null (H0), as a point is, or the alternative (H1), as an inequality or a region is, and the panel reports the Bayes factor of that hypothesis against the other: BF_{01} for a point, BF_{10} otherwise. For a point hypothesis, a dotted line marks the tested value and two points mark the heights whose ratio is the Savage-Dickey Bayes factor. For a directional or interval hypothesis, the shaded area is the posterior probability of H1. Each panel has a density scale of its own, so the panels share one y axis, labelled Density, drawn on the first panel of each row and without values.

Usage

```
## S3 method for class 'usdt_bf'
plot(x, caption = TRUE, ...)
```

Arguments

`x`	A `usdt_bf` object from `usdt_bf()`.
`caption`	Logical. Show the caption that explains the plot under it (default `TRUE`).
`...`	Ignored.

Details

`usdt_bf()` draws this figure by default. Call `plot()` to draw it again, or to draw some of its rows, e.g. `plot(b[3, ])`.

Value

A `ggplot` object. Its underlying data frame is stored in `$data`.

See Also

`usdt_bf()`

`plot.usdt_priors` *Prior predictive check for the Bayesian hierarchical SDT model*

Description

Draws what the priors imply before seeing any data: the hit and false-alarm rates of a new subject in each task, and the prior of the quantity tested by each hypothesis. Use it to check that the priors allow the data you expect and to see the prior that the model induces on the latent intercept.

Usage

```
## S3 method for class 'usdt_priors'
plot(
  x,
  data = NULL,
  unequal_variances = FALSE,
  latent = c("correlation", "regression"),
  caption = TRUE,
  ...
)
```

Arguments

`x`	A `usdt_priors` object from `usdt_priors()`.
`data`	Optional `usdt_data` object. When given, a criterion fixed by the median split stays fixed, as in `hsdt()`, and the tasks take the data labels. Without it, both criteria are estimated. The caption says which criteria are fixed.
`unequal_variances`	Logical. Simulate the signal standard deviation from its prior instead of fixing it to 1.
`latent`	How the model writes the relation between the two latent sensitivities, as in `hsdt()`: by their means, standard deviations and correlation (`"correlation"`, the default) or by the latent regression of the influence on the awareness one (`"regression"`).
`caption`	Logical. Show the caption that explains the plot under it (default `TRUE`).
`...`	Ignored.

Details

The plot simulates 100,000 draws from the priors in R. No model is fitted and Stan is not needed.

The rates panel shows the hit and false-alarm rates of one new subject drawn from each population draw. Rates piled up at 0 or 1 mean that the priors allow extreme performance.

Each hypothesis panel shows the prior of the quantity it tests. A quantity with a prior of its own is drawn exactly: the correlation (H2), and under the latent regression also the intercept (H3). The others have priors that the model induces through the remaining priors: the intercept under the

correlation parameterization, the difference (H1) under the latent regression. Each of these is drawn as the density that one prior gives it, averaged over 20,000 prior draws of the other parameters (a Rao-Blackwell estimate), the same priors that `usdt_bf()` averages over 100,000 draws. The difference (H1) and the intercept (H3) span four robust standard deviations either side of their median, since an induced prior of either has heavy tails. Dotted lines mark the value of each hypothesis under the null.

Value

A `ggplot` object. Its underlying data frame is stored in `$data`.

See Also

`usdt_priors()`

Examples

```
plot(usdt_priors())

# A narrower prior for the mean influence sensitivity
plot(usdt_priors(dprime = list(awareness = "normal(0, 1)",
                               influence = "normal(0, 0.5)")))

# The latent regression, with a prior on its intercept
plot(usdt_priors(intercept = "normal(0, 0.3)"), latent = "regression")
```

---

plot_power *Plot a population, its power, its smallest effects or a plan*

Description

Draws the objects of the power analysis with the style of the other uSDT plots.

Usage

```
## S3 method for class 'usdt_population'
plot(
  x,
  trials = NULL,
  signal = 0.5,
  dichotomize = c("none", "influence", "awareness", "both"),
  caption = TRUE,
  ...
)

## S3 method for class 'usdt_power'
plot(x, caption = TRUE, ...)
```

```
## S3 method for class 'usdt_mdes'
plot(x, caption = TRUE, ...)

## S3 method for class 'usdt_plan'
plot(x, caption = TRUE, ...)
```

Arguments

x	A usdt_population, usdt_power, usdt_mdes or usdt_plan object.
trials	For a population, the number of trials per subject in each task, as in usdt_power(), to add the distribution of the measured d'.
signal, dichotomize	For a population with trials, as in usdt_power().
caption	Logical. Show the caption that explains the plot under it (default TRUE).
...	Ignored.

Details

- A usdt_population shows, for each scenario, the bivariate normal distribution of the true d' across subjects (ellipses holding 50% and 90% of them) with the latent regression of influence on awareness d' and its intercept at $d'_A = 0$. With trials, a dashed ellipse adds the distribution of the d' that those trials measure, wider by the measurement error, which is what attenuates the observed correlation.
- A usdt_power shows power against the number of subjects, or of trials when the subjects are fixed, one line per hypothesis. A simulation adds its 95% Wilson intervals. A single design shows one point per hypothesis.
- A usdt_mdes shows the power along the values searched, with the smallest detectable effect and the population value marked.
- A usdt_plan with subjects and trials free shows the cost of the cheapest closed-form design for each number of trials (a line for one free task, a map for two), with the chosen design marked. With one value free, it shows the approximate power (or the expected interval width) along it.

Value

A ggplot object. Its underlying data frame is stored in $data.

See Also

usdt_population(), usdt_power(), usdt_mdes(), usdt_plan()

Examples

```
pop <- usdt_population(dprime    = c(awareness = 0.15, influence = 0.30),
                       sd_dprime = c(awareness = 0.35, influence = 0.30),
                       cor_dprime = 0.3, criterion = 0.1)
plot(pop, trials = c(awareness = 64, influence = 200),
```

```
    dichotomize = "influence")

plot(usdt_power(pop, subjects = seq(20, 300, 40),
                trials = c(awareness = 64, influence = 200),
                dichotomize = "influence"))

plot(usdt_mdes(pop, subjects = 80,
               trials = c(awareness = 64, influence = 200),
               hypothesis = "intercept > 0", dichotomize = "influence"))

plot(usdt_plan(pop, hypothesis = "diff = 0", subjects = c(20, 300),
               trials = list(awareness = c(32, 96), influence = 200),
               cost = c(subject = 600, awareness = 4, influence = 1.5),
               dichotomize = "influence"))
```

`sdt_moments` *Signal detection measures for each subject*

Description

Computes empirical hit rates, false-alarm rates, sensitivity (d'), and response criteria for each participant without fitting a model. It can also calculate sampling variances, standard errors, and expected values for d'.

Usage

```
sdt_moments(
  data,
  subject_col = NULL,
  condition_col = NULL,
  condition_levels = NULL,
  response_col = NULL,
  response_levels = NULL,
  coding = c("deviation", "treatment"),
  correction = c("hautus", "none"),
  variances = FALSE
)
```

Arguments

`data`	A `usdt_data` object or a standard trial-level data frame. When given a `usdt_data` object, the function processes both tasks and includes a `task` column in the output.
`subject_col`, `condition_col`, `response_col`	Column names for subject, condition, and response variables. Only required when `data` is a plain data frame.

`condition_levels`, `response_levels`
: Named vectors mapping condition and response labels, like `c(signal = "old", noise = "new")`. Required for a plain data frame. A `usdt_data` object supplies its own roles and needs neither.

`coding`
: Criterion definition to report: `"deviation"` measures the criterion from the midpoint between the signal and noise distributions, whereas `"treatment"` measures it from the noise distribution. A `usdt_data` object supplies its own coding.

`correction`
: Handling of extreme rates (0 or 1) that make d' infinite. `"hautus"` adds 0.5 to all four cell counts for affected participants. `"none"` leaves infinite values in place.

`variances`
: Logical. If `TRUE`, computes the sampling variance of d' from Gourevitch and Galanter (1967) and Miller (1996), each with its own standard error, as well as the expected value of d' under Miller's distribution.

Details

Edge corrections apply only to participants with extreme rates (0 or 1).

When requested, the sampling variance of d' is estimated using the asymptotic approximation of Gourevitch and Galanter (1967) and the binomial-distribution approach of Miller (1996). See Suero et al. (2017) for a comparison between the two approaches.

Value

A data frame with one row per subject (or per subject and task for `usdt_data` inputs) containing:

- `hit`, `miss`, `fa`, `cr`: Raw response counts.
- `hr`, `far`: Observed hit and false-alarm rates.
- `zhr`, `zfar`: Probit-transformed rates.
- `dprime`, `criterion`: Descriptive SDT estimates.
- `corrected`: Logical flag indicating whether the participant received an edge correction.
- `var_gg`, `se_gg`: Asymptotic variance and standard error from Gourevitch and Galanter (1967), present when `variances = TRUE`.
- `var_miller`, `se_miller`, `expected_dprime`: Moments from Miller (1996), present when `variances = TRUE`.

References

Gourevitch, V., & Galanter, E. (1967). A significance test for one parameter isosensitivity functions. *Psychometrika*, 32(1), 25–33. doi:10.1007/BF02289402

Hautus, M. J. (1995). Corrections for extreme proportions and their biasing effects on estimated values of d'. *Behavior Research Methods, Instruments, & Computers*, 27(1), 46–51. doi:10.3758/BF03203619

Miller, J. (1996). The sampling distribution of d'. *Perception & Psychophysics*, 58(1), 65–72. doi:10.3758/BF03205476

Suero, M., Privado, J., & Botella, J. (2017). Methods to estimate the variance of some indices of the signal detection theory: A simulation study. *Psicologica*, 38(1), 77–109.

See Also

hsdt()

Examples

```
# 1. From a prepared usdt_data object (both tasks at once)
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

head(sdt_moments(d))

# 2. From raw trials with sampling variances and standard errors
head(sdt_moments(vadillo_awareness,
                 subject_col      = "subj",
                 condition_col    = "condition",
                 condition_levels = c(signal = "old", noise = "new"),
                 response_col     = "judged.old",
                 response_levels  = c(signal = 1, noise = 0),
                 variances        = TRUE))
```

---

usdt_aggregate *Count binary responses by group*

Description

Aggregates trial-level binary responses into the number of signal responses (y) and the number of trials (n) for every combination of the grouping columns. A binomial model fitted to these counts has the same likelihood, and therefore the same estimates and posterior, as a Bernoulli model fitted to the trials, and it runs much faster. Use it to prepare custom models fitted with brms or lme4 (see usdt_hypotheses).

Usage

```
usdt_aggregate(data, response, by)
```

Arguments

`data`	A data frame with one row per trial.
`response`	Name of the binary response column, coded `0`/`1` or `FALSE`/`TRUE`.
`by`	Names of the grouping columns. Every predictor of the model, the subject included, must be among them, so that all the trials of a row share their predictors.

Value

A data frame with one row per combination of the `by` columns that occurs in `data`, those columns, and the counts `y` and `n`.

See Also

usdt_hypotheses

Examples

```
# Awareness task of Vadillo et al. (2025): "old" judgements by subject,
# condition and set size
counts <- usdt_aggregate(vadillo_awareness, response = "judged.old",
                         by = c("subj", "condition", "set.size"))
head(counts)

# The counts keep every trial
sum(counts$n) == nrow(vadillo_awareness)
```

`usdt_bf` *Bayes factors for the hypotheses of a Bayesian hierarchical SDT model*

Description

Tests point, directional and interval hypotheses about the quantities of the three uSDT hypotheses in a Bayesian `hsdt()` fit or a custom model fitted with brms: the difference between mean sensitivities (`diff`, H1, reported as $\Delta d'$, influence minus awareness), their correlation (`rho`, H2) and the slope and intercept of the latent regression (`slope`, `intercept`, H3).

Usage

```
usdt_bf(
  fit,
  hypothesis = c("diff = 0", "rho = 0", "intercept = 0"),
  awareness = "d_A",
  influence = "d_I",
  level = NULL,
  plot = TRUE
)
```

```
## S3 method for class 'usdt_bf'
print(x, digits = 3L, width = 80L, ...)
```

Arguments

fit	A Bayesian hsdt object, fitted with estimation = "bayesian", or a brmsfit whose priors meet the conditions in Details.
hypothesis	Character vector of hypotheses. Each one is "<quantity> <op> <value>", where the quantity is diff, rho, slope or intercept and the operator is =, < or > (<= and >= are read as < and >), or "<quantity> in [a, b]" and "<quantity> out [a, b]" for a region and its complement. A point statement is the null hypothesis H0, and H1 its negation. An inequality or a region is the alternative H1, and H0 its complement. The default tests the three point nulls.
awareness, influence	For a brms model, the population-level terms of the awareness and influence sensitivities, as in usdt_tests(). Ignored for an hsdt fit.
level	Credible level of the reported intervals. Defaults to the level of an hsdt fit, and to 0.95 for a brms model.
plot	Logical. Draw, for each hypothesis, a panel with the prior and posterior of the tested quantity (default TRUE). Set plot = FALSE inside loops, reports and other functions, and draw the figure later with plot.usdt_bf().
x	A usdt_bf object.
digits	Number of decimals printed.
width	Width of the printed report, in characters.
...	Ignored.

Value

An object of class usdt_bf: a data frame with one row per hypothesis and the columns hypothesis, test ("Savage-Dickey density ratio test", "Directional test" or "Interval test"), H1, H0, estimate and est.error (posterior mean and SD), conf.low and conf.high (central credible interval), post.prob (posterior probability of H1 for a directional or interval test), log_BF10, BF10, BF01, evidence and favours. It prints one report per hypothesis, with both log Bayes factors. The prior and posterior curves are stored, so plot.usdt_bf() can draw the figure again.

Point hypotheses: the Savage-Dickey density ratio

A point hypothesis $\theta = c$ compares a model in which θ is fixed at c with the fitted model, in which it is free. When the first is nested in the second and keeps the conditional prior of the other parameters, the Bayes factor is the ratio of the posterior and prior densities at c (Dickey & Lientz, 1970, Wagenmakers et al., 2010):

$$\mathrm{BF}_{01} = p(\theta = c \mid y)/p(\theta = c).$$

The posterior density comes from a logspline fit to the posterior draws. A fit is used only if its distribution function stays within 0.05 of that of the draws at every percentile. When neither

logspline algorithm gives such a fit, which happens with the very heavy tails of the intercept and slope when the awareness sensitivities barely vary across subjects, the draws are fitted on the scale $\operatorname{asinh}((x - m)/s)$, with m their median and s their median absolute deviation, and the density is mapped back. Density estimates are least precise far from the posterior mass, so a test value deep in a tail is read on the log scale and with caution.

Directional and interval hypotheses

An inequality or a region is evidence about where the quantity lies, so its Bayes factor is the ratio of posterior to prior odds of H1:

$$\mathrm{BF}_{10} = \frac{P(H_1 \mid y)/P(H_0 \mid y)}{P(H_1)/P(H_0)}.$$

`post.prob` reports $P(H_1 \mid y)$, the share of posterior draws in H1, and the prior probabilities come from the priors described below. When every draw falls on one side, the draws cannot bound the Bayes factor and `log_BF10` is infinite.

The prior of each quantity

- `rho`: the marginal of the `lkj` prior of `usdt_priors()`, exact.
- `diff`: given the mean awareness sensitivity μ_A, the difference is the mean influence sensitivity shifted by μ_A, so its prior density and probabilities are those of the prior of μ_I at $x + \mu_A$, averaged over 100,000 prior draws of μ_A (a Rao-Blackwell estimate). With normal priors on both means this is the normal prior of their difference.
- `intercept`: induced by the other priors. Given the slope β_1 and μ_A, the intercept is μ_I shifted by $\beta_1\mu_A$, so its prior is likewise averaged over 100,000 prior draws of the slope and of μ_A. The prior has no mean or variance, because the slope involves $1/\sigma_A$.
- `slope`: given the ratio σ_I/σ_A, the slope is a rescaled correlation, so its prior is averaged over 100,000 prior draws of that ratio. Its density is infinite at zero, and the slope always has the sign of the correlation, so `slope = 0`, `slope > 0` and `slope < 0` are tested as the same statements about `rho`. The row still summarises the posterior of the slope.

Under the latent regression parameterization the intercept has a prior of its own instead of μ_I, so it is exact like `rho`, and `slope` is as above. The difference is induced:

- `diff`: $\beta_0 + (\beta_1 - 1)\mu_A$, so its prior is that of the intercept, shifted and averaged over 100,000 prior draws of the slope and of μ_A. Like the induced intercept above, it has heavy tails and no mean or variance.

The data fix $\mu_I = \beta_0 + \beta_1\mu_A$, so under the latent regression a narrow intercept prior also moves the posterior of the slope and of `rho`, even though their priors are independent of it: the Bayes factors of `rho` and `slope` then depend on the intercept prior as well.

These averages work for any family of `usdt_priors()`. The prior draws use the random number generator. Call `set.seed()` first for reproducible results.

Custom brms models

For a `brmsfit`, the priors are read from `brms::prior_summary()`, each parameter taking its most specific prior (coefficient, then group, then class), and the quantities get the same priors as above. The model must have:

- a proper prior on each sensitivity term, from the families of `usdt_priors()`, e.g. `prior(normal(0, 1), class = b)`. The flat default of brms is improper, and a Bayes factor needs a proper prior.
- an LKJ(η) prior on the correlations of the group-level term that holds both sensitivities. One correlation of a $K \times K$ LKJ matrix has the marginal of a 2×2 LKJ($\eta - 1 + K/2$), so a block with other terms besides the two sensitivities still has an exact prior for `rho`.
- for `intercept` and `slope`, priors on the two SDs from the families of `usdt_priors()`, truncated at zero as brms does. The brms default, `student_t(3, 0, 2.5)`, qualifies.

The Bayes factor of a point hypothesis depends on the prior of the tested quantity: a wider prior puts less density at the tested value and favours the null. The intercept prior is induced by the means, SDs and correlation, so its Bayes factor depends on all of them. Report results with the priors used, and check them with other reasonable priors.

`evidence` follows the bands of Jeffreys (1961) with the labels of Lee and Wagenmakers (2013): anecdotal (BF $<$ 3), moderate (< 10), strong (< 30), very strong (< 100) and extreme.

References

Dickey, J. M., & Lientz, B. P. (1970). The weighted likelihood ratio, sharp hypotheses about chances, the order of a Markov chain. *The Annals of Mathematical Statistics*, 41(1), 214-226. doi:10.1214/aoms/1177697203

Jeffreys, H. (1961). *Theory of probability* (3rd ed.). Oxford University Press.

Lee, M. D., & Wagenmakers, E.-J. (2013). *Bayesian cognitive modeling: A practical course*. Cambridge University Press.

Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage-Dickey method. *Cognitive Psychology*, 60(3), 158-189. doi:10.1016/j.cogpsych.2009.12.001

See Also

`hsdt()`, `usdt_priors()`, `plot.usdt_bf()`

Examples

```
## Not run:
# Requires a C++ toolchain and the BH and RcppEigen headers for rstan.
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

m <- hsdt(d, estimation = "bayesian", cores = 2, seed = 1)
```

```
# The three point nulls, with their figure
set.seed(1)
usdt_bf(m)

# Unconscious processing as a directional hypothesis, and a region of
# practical equivalence for the difference, without the figure
b <- usdt_bf(m, c("intercept > 0", "diff in [-0.1, 0.1]"), plot = FALSE)
b
plot(b[1, ])

## End(Not run)
```

`usdt_boot` *Parametric bootstrap intervals for hierarchical SDT models*

Description

Simulates new datasets from a fitted model using `lme4::bootMer()`, refits the model to each replicate, and computes bootstrap confidence intervals for the three core hypotheses (H1, H2, H3). This is especially useful when asymptotic Wald intervals are unreliable due to singular or boundary fits.

Usage

```
usdt_boot(
  object,
  nsim = 1000,
  ncores = 1L,
  max_attempts = 2 * nsim,
  seed = NULL,
  progress = interactive(),
  level = 0.95,
  type = c("perc", "norm", "basic")
)
```

Arguments

`object`	A frequentist `hsdt` object fitted by `hsdt()`. A Bayesian fit already carries the posterior of every quantity.
`nsim`	Target number of successful replicates (at least 500).
`ncores`	Number of CPU cores for parallel processing. Values above 1 create a temporary cluster that works across Windows, macOS, and Linux, and automatically stops when finished.
`max_attempts`	Maximum number of refits to attempt. Defaults to `2 * nsim`.
`seed`	Random seed for reproducibility. With the default `NULL`, no seed is set and the bootstrap continues the current random number stream.

progress	Logical. Display a progress bar during fitting (defaults to TRUE in interactive sessions).
level	Confidence level for intervals (default is 0.95).
type	Type of bootstrap interval: "perc" (percentile), "norm" (normal approximation with bias correction), or "basic" (empirical basic). For correlations, "norm" and "basic" operate on the Fisher-z scale. If the sample correlation lies on the boundary (-1 or 1), these types return NA, whereas "perc" remains available.

Details

Refits are retained if they converge and all four hypothesis estimates are finite, including singular or boundary refits meeting those conditions. Inspect $boot$failures and $boot$retained. For dichotomized data, the bootstrap simulates binary responses, without repeating the median split. It remains conditional on the fitted model and available information.

When an attempted run finishes with fewer than 500 usable replicates, the original Wald intervals are preserved, a warning is issued, and raw attempt diagnostics are stored in $boot.

Point estimates remain identical to the original model fit. Two-sided bootstrap p-values compare the observed test statistic against the centered bootstrap distribution using standard finite-sample adjustment ($(k + 1)/(B + 1)$), ensuring p-values never equal zero.

Value

An updated hsdt object where interval columns in $tests are replaced by bootstrap estimates. A new $boot element contains:

- $t: Matrix of replicates for the three hypotheses.
- $variance: Replicates of sensitivity variances.
- $population: Replicates of average criteria and sensitivities.
- $subjects: Replicates of individual-level parameters.
- Run diagnostics and convergence counts (usable, attempted, retained, failures).

See Also

hsdt(), usdt_tests()

Examples

```
# Contextual cuing data from Vadillo et al. (2025)
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
```

```
)

m <- hsdt(d)


# Run parametric bootstrap with 500 replicates. Refitting this model 500
# times takes several minutes
b <- usdt_boot(m, nsim = 500, seed = 1)

# Inspect updated summary with bootstrap intervals and p-values
summary(b)

# Check fit diagnostics across bootstrap replicates
b$boot[c("usable", "attempted", "retained", "failures")]
```

`usdt_data` *Prepare data for hierarchical SDT models*

Description

Formats awareness and influence task data into a standardized structure for `hsdt()`. Use `usdt_data_tasks()` when tasks are stored in separate data frames, or `usdt_data_long()` when both tasks are kept in a single data frame with a column that identifies each task.

Usage

```
usdt_data_tasks(
  awareness,
  influence,
  subject_col,
  condition_col = NULL,
  condition_levels = NULL,
  response_col = NULL,
  response_levels = NULL,
  successes_col = NULL,
  trials_col = NULL,
  successes_type = c("auto", "counts", "proportions"),
  sdt_cols = NULL,
  dichotomize = c("none", "influence", "awareness", "both"),
  ties = c("noise", "random"),
  coding = c("deviation", "treatment"),
  labels = c(awareness = "Awareness", influence = "Influence")
)

usdt_data_long(
  data,
```

```
  task_col,
  task_levels,
  subject_col,
  condition_col = NULL,
  condition_levels = NULL,
  response_col = NULL,
  response_levels = NULL,
  successes_col = NULL,
  trials_col = NULL,
  successes_type = c("auto", "counts", "proportions"),
  sdt_cols = NULL,
  dichotomize = c("none", "influence", "awareness", "both"),
  ties = c("noise", "random"),
  coding = c("deviation", "treatment"),
  labels = NULL
)

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

Arguments

awareness, influence	Data frames for each task (used in `usdt_data_tasks()`).
subject_col	Name of the column that identifies participants.
condition_col	Name of the column for signal and noise conditions. Not needed when using `sdt_cols`.
condition_levels	Named vector mapping condition labels, like `c(signal = "old", noise = "new")`. Required: which label is the signal and the noise.
response_col	Name of the column with responses. Can be binary choices or continuous values (like response times) to split at the median.
response_levels	Named vector mapping responses, like `c(signal = 1, noise = 0)`. For a task named in `dichotomize` it names the side of the median instead, as `c(signal = "faster", noise = "slower")` or the reverse. Required in both cases.
successes_col, trials_col	Names of columns with pre-calculated counts or proportions of signal responses and total trials. Use these instead of `response_col`.
successes_type	Format of `successes_col`: `"counts"`, `"proportions"`, or `"auto"`.
sdt_cols	Named vector for SDT table columns, like `c(hit = "H", miss = "M", fa = "FA", cr = "CR")`.
dichotomize	Which tasks to split at the median using `meyen_split()`. Use `"none"`, `"awareness"`, `"influence"`, `"both"`, or a list like `list(awareness = FALSE, influence = TRUE)`.
ties	How to handle trials that fall exactly on the median. See `meyen_split()`.

coding	How condition is coded in the model: "deviation" (-0.5, 0.5) or "treatment" (0, 1). See Details.
labels	Optional names for the tasks in printed output.
data	A single data frame with both tasks (used in usdt_data_long()).
task_col	Name of the column that identifies the task in data.
task_levels	Named vector mapping task labels, like c(awareness = "A", influence = "I").
x	A usdt_data object.
...	Ignored.

Details

The functions count responses for each subject and condition, check that the same subjects appear in both tasks, and print a summary table so you can verify the column settings before fitting the model.

Value

An object of class usdt_data. The $agg table contains the counts used by hsdt(), and $meta contains setup details and summaries.

Settings per task

Arguments for columns and levels take either a single value (used for both tasks) or a list with separate settings for each task:

```
condition_col    = list(awareness = "cond", influence = "cue")
condition_levels = list(awareness = c(signal = "old", noise = "new"),
                        influence = c(signal = "congruent",
                                      noise = "incongruent"))
dichotomize      = list(awareness = FALSE, influence = TRUE)
```

This works for all column and level arguments, so you can combine trial-level data in one task with summary tables in the other.

Condition coding

Under "deviation" coding (-0.5 vs. +0.5), the intercept is $-c$, the criterion measured from the point between the two distributions. Under "treatment" coding (0 vs. 1), the intercept is $z(\mathrm{FAR})$, the criterion measured from the noise distribution.

With balanced signal/noise trials and an exact half split, the classical criterion is zero under deviation coding. Ties, odd counts or imbalance can change this. hsdt() fixes a dichotomized task's criterion only when its mean absolute descriptive subject criterion is below 0.02.

See Also

hsdt(), meyen_split(), sdt_moments()

Examples

```
# 1. Tasks in separate data frames
# Awareness task: binary choices (old/new)
# Influence task: response times (split at the median)
d_separate <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)
d_separate

# 2. Tasks combined in a single long data frame
long <- rbind(
  data.frame(task      = "A",
             subj      = vadillo_awareness$subj,
             condition = vadillo_awareness$condition,
             response  = vadillo_awareness$judged.old),
  data.frame(task      = "I",
             subj      = vadillo_cuing$subj,
             condition = vadillo_cuing$condition,
             response  = vadillo_cuing$rt)
)

d_long <- usdt_data_long(
  long,
  task_col         = "task",
  task_levels      = c(awareness = "A", influence = "I"),
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = "response",
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)
d_long
```

`usdt_equivalence` *Equivalence, non-inferiority and minimal-effect tests*

Description

Tests interval, one-sided and point hypotheses about the quantities of the three uSDT hypotheses in a frequentist `hsdt()` fit or a custom model fitted with `lme4::glmer()`: the difference between mean sensitivities (`diff`, H1, reported as $\Delta d'$, influence minus awareness), their correlation (`rho`, H2) and the slope and intercept of the latent regression (`slope`, `intercept`, H3). The hypotheses are written as in `usdt_bf()`, so a frequentist and a Bayesian analysis can state the same bounds.

Usage

```
usdt_equivalence(
  fit,
  hypothesis,
  awareness = "d_A",
  influence = "d_I",
  alpha = 0.05
)

## S3 method for class 'usdt_equivalence'
print(x, digits = 3L, width = 80L, ...)
```

Arguments

`fit`	A frequentist `hsdt` object, with or without the replicates of `usdt_boot()`, or a `glmerMod` from `lme4::glmer()`.
`hypothesis`	Character vector of hypotheses. It has no default: the bounds are the smallest effects of interest, which only the analysis can set. Each one is `"<quantity> <op> <value>"`, where the quantity is `diff`, `rho`, `slope` or `intercept` and the operator is =, < or > (<= and >= are read as < and >), or `"<quantity> in [a, b]"` and `"<quantity> out [a, b]"` for a region and its complement.
`awareness`, `influence`	For a `glmerMod`, the fixed effects of the awareness and influence sensitivities, as in `usdt_tests()`. Defaults match the internal naming of `hsdt()`.
`alpha`	Significance level of every test, between 0 and 0.5 (default 0.05).
`x`	A `usdt_equivalence` object.
`digits`	Number of decimals printed.
`width`	Width of the printed report, in characters.
`...`	Ignored.

Value

An object of class `usdt_equivalence`: a data frame with one row per hypothesis and the columns `hypothesis`, `test` (`"Two-sided test"`, `"One-sided test"`, `"Equivalence test (TOST)"` or `"Minimal-effect test"`), `H1`, `H0`, `estimate`, `se`, `conf.level`, `conf.low` and `conf.high` (the interval that decides the test), `ci_method`, `p.lower` and `p.upper` (the one-sided p-values at the two bounds of a region), `p.value`, `decision`, `status` and `reason`. Rows that cannot be estimated have `status = "not estimable"` and give the `reason`. It prints one report per hypothesis.

The tests

A point statement names the null hypothesis H0. An inequality or a region names the alternative H1, as in `usdt_bf()`. Each test rejects H0 at level `alpha`:

- `q = c`: the two-sided test of H0: $q = c$.
- `q > c` and `q < c`: one-sided tests. With a margin on the unfavourable side of the reference value, `diff > -0.2` is a non-inferiority test and `diff < 0.2` a non-superiority test.
- `q in [a, b]`: an equivalence test by two one-sided tests (TOST, Schuirmann, 1987. Lakens et al., 2018). H0, $q \leq a$ or $q \geq b$, is rejected when both one-sided tests reject their part of it, so `p.value` is the larger of `p.lower` and `p.upper`.
- `q out [a, b]`: a minimal-effect test. H0, $a \leq q \leq b$, is rejected when either one-sided test rejects it at `alpha / 2`, so `p.value` is twice the smaller of `p.lower` and `p.upper`. As the region shrinks to a point it becomes the two-sided test of that point.

Each test is decided by the position of a confidence interval, and the row reports that interval: of level `1 - 2 * alpha` for the one-sided and equivalence tests, and `1 - alpha` for the two-sided and minimal-effect tests, which split `alpha` between two tails. With the default `alpha = 0.05`, an equivalence test reports a 90% interval, and H0 is rejected exactly when that interval lies inside the region.

The correlation

The bounds of `rho` are given, and its estimate and interval reported, on the correlation scale. The test runs on the Fisher-z scale, where the sampling distribution of the estimate is closer to normal:

$$z = \frac{\operatorname{atanh}(\hat{\rho}) - \operatorname{atanh}(c)}{SE/(1-\hat{\rho}^2)}.$$

Because `atanh()` is increasing, $\rho > c$ and $\operatorname{atanh}(\rho) > \operatorname{atanh}(c)$ are the same statement, so the p-value and the decision are those of the hypothesis about `rho`, and the interval, transformed back, stays inside (-1, 1).

Every p-value comes from the interval reported with it. `usdt_tests()` instead tests a null association on the covariance that `rho` and the slope share, so its p-values for `rho = 0` and `slope = 0` can differ slightly from these.

The intercept and the slope

Both divide by the variance of the awareness sensitivity. When the awareness sensitivity barely varies across subjects, their standard errors are large and their Wald intervals inaccurate, so their equivalence tests need many subjects to succeed, and the bootstrap of `usdt_boot()` gives more accurate intervals.

Boundary fits and the bootstrap

The correlation, slope and intercept need the full covariance matrix of the fit, which is unavailable when the sensitivity covariance is on the boundary (a variance estimated as zero, or a correlation of -1 or 1). Their rows are then marked `"not estimable"`, while the difference, which only needs the fixed effects, is still tested. After `usdt_boot()`, every test uses the bootstrap replicates and the interval type chosen there. The p-value at a bound is the share of the bootstrap distribution beyond

it, counted as $(k+1)/(B+1)$ as in `usdt_boot()`, so the decision agrees with the reported interval up to that correction.

References

Lakens, D., Scheel, A. M., & Isager, P. M. (2018). Equivalence testing for psychological research: A tutorial. *Advances in Methods and Practices in Psychological Science*, 1(2), 259-269. doi:10.1177/2515245918770963

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics*, 15(6), 657-680. doi:10.1007/BF01068419

See Also

`usdt_tests()`, `usdt_boot()`, `usdt_bf()`

Examples

```
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

m <- hsdt(d)

# Mean sensitivities equivalent within 0.2 d' units, and an unconscious
# effect at zero awareness smaller than 0.2
usdt_equivalence(m, c("diff in [-0.2, 0.2]", "intercept < 0.2"))

# The bounds of the correlation are correlations
usdt_equivalence(m, "rho in [-0.3, 0.3]")
```

`usdt_hypotheses` *Test the three core hypotheses of a hierarchical SDT model*

Description

Evaluates the difference between average task sensitivities (H1), their correlation across subjects (H2), and the regression of influence sensitivity on awareness sensitivity (H3). `usdt_tests()` computes all three together, while individual functions compute them separately.

Usage

```
usdt_tests(fit, awareness = "d_A", influence = "d_I", level = 0.95)

sensitivity_diff(fit, awareness = "d_A", influence = "d_I", level = 0.95)

latent_cor(fit, awareness = "d_A", influence = "d_I", level = 0.95)

latent_regression(fit, awareness = "d_A", influence = "d_I", level = 0.95)
```

Arguments

`fit`	A fitted model: an `hsdt` object, frequentist or Bayesian, a `glmerMod` from `lme4::glmer()` or a `brmsfit` from `brms::brm()`.
`awareness, influence`	Character strings naming the sensitivity terms in the model. Defaults match the internal naming of `hsdt()`. For custom models, both terms must be fixed effects and share a common random-effects grouping by subject.
`level`	Confidence level for intervals (default is 0.95).

Details

These tests run automatically inside `hsdt()` and appear in its summary. Calling them directly is especially useful when fitting custom models with `lme4::glmer()` or `brms::brm()`, allowing you to test these hypotheses while controlling for additional covariates (e.g., set size, experimental groups).

Value

A data frame with columns `term`, `estimate`, `se`, `statistic`, `p.value`, `conf.low`, `conf.high`, and `ci_method`. Columns `status` and `reason` flag unsupported estimates (e.g., singular fits). `usdt_tests()` includes an extra `hypothesis` column (`H1`, `H2`, `H3`).

For a Bayesian `hsdt` fit the columns are `term`, `estimate` (posterior mean), `est.error` (posterior standard deviation), `conf.low` and `conf.high` (central credible interval), `ci_method`, `p.value` (two-sided posterior p-value, below `1 - level` exactly when the interval excludes zero), `rhat` and `ess_bulk`.

The three hypotheses

- **H1 (Mean difference):** Tests whether average sensitivity differs between the awareness and influence tasks.
- **H2 (Correlation):** Tests the correlation between task sensitivities across participants using a Fisher-z transformed interval.
- **H3 (Latent regression):** Regresses influence sensitivity onto awareness sensitivity. The intercept represents expected influence performance when awareness sensitivity is zero ($d'_{\text{Awareness}} = 0$), testing for unconscious processing.

With positive sensitivity variances, H2 and the H3 slope share the same zero-covariance null. The analytical frequentist output uses the same covariance test for both. The correlation interval uses Fisher-z.

Custom models with covariates

To adjust tests for additional factors, specify the model directly using `lme4::glmer()` or `brms::brm()`. As long as the two sensitivity terms are included as fixed effects and correlated across subjects via random slopes, `usdt_tests()` will compute the latent tests conditional on those covariates. A brms model is summarised from its posterior, like a Bayesian `hsdt()` fit, and keeps the term names that lme4 would give it. Aggregating the trials with `usdt_aggregate()` gives either model the same estimates and makes it much faster to fit.

See Also

`hsdt()`, `usdt_boot()`, `usdt_aggregate()`

Examples

```
# 1. Standard model via hsdt()
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)

m <- hsdt(d)

# All three tests at once
usdt_tests(m)

# Or one test at a time
sensitivity_diff(m)   # H1
latent_cor(m)         # H2
latent_regression(m)  # H3

# 2. Custom model with covariates via glmer()
# Controlling for display set size across both tasks
trials <- rbind(
  data.frame(vadillo_awareness[c("subj", "condition", "set.size")],
             task = "A", resp = vadillo_awareness$judged.old),
  data.frame(vadillo_cuing[c("subj", "condition", "set.size")],
             task = "I", resp = meyen_split(vadillo_cuing$rt,
                                            by = vadillo_cuing$subj))
)

# Deviation contrasts (-0.5 vs 0.5). `awareness` flags the awareness task
trials <- within(trials, {
  cond   <- ifelse(condition == "old", 0.5, -0.5)
```

```
  size   <- ifelse(set.size == "set size 16", 0.5, -0.5)
  awareness <- as.integer(task == "A")
})

# Counting responses per subject, task and condition keeps the likelihood
# and makes the fit much faster than one row per trial
counts <- usdt_aggregate(trials, response = "resp",
                         by = c("subj", "task", "cond", "size", "awareness"))

# Standard glmer formula: influence criterion is omitted (fixed at 0
# by the median split). Random effects estimate the awareness criterion
# and correlated task sensitivities across subjects.
fit <- lme4::glmer(
  cbind(y, n - y) ~ 0 + awareness + task:size + task:cond +
    (0 + awareness | subj) + (0 + task:cond | subj),
  data = counts, family = binomial("probit"),
  control = lme4::glmerControl(optimizer = "bobyqa")
)

# Check the names lme4 assigned to the sensitivity terms
names(lme4::fixef(fit))

# Evaluate hypotheses conditional on set size
usdt_tests(fit, awareness = "taskA:cond", influence = "taskI:cond")

## Not run:
# 3. The same model fitted with brms. Compiling needs a C++ toolchain
# and the BH and RcppEigen headers for rstan.
fit_b <- brms::brm(
  y | trials(n) ~ 0 + awareness + task:size + task:cond +
    (0 + awareness | subj) + (0 + task:cond | subj),
  data = counts, family = binomial("probit"),
  prior = brms::prior(normal(0, 1), class = b),
  chains = 2, cores = 2, seed = 1, refresh = 0
)

# The same term names. The tests now summarise the posterior
usdt_tests(fit_b, awareness = "taskA:cond", influence = "taskI:cond")

# With a proper prior on the sensitivities, also Bayes factors
set.seed(1)
usdt_bf(fit_b, awareness = "taskA:cond", influence = "taskI:cond")

## End(Not run)
```

`usdt_mdes` *Smallest effect a uSDT study detects*

Description

Finds, for each hypothesis, the smallest effect that a design detects with the given power: the population value of the tested quantity, moved away from H0 with the other values of the population fixed, at which the approximate power of `usdt_power()` reaches it. For an equivalence test (`in`), it finds the narrowest equivalence region, around the same centre, that the design can establish.

Usage

```
usdt_mdes(
  population,
  subjects,
  trials,
  hypothesis = c("diff = 0", "rho = 0", "intercept = 0"),
  power = 0.8,
  alpha = 0.05,
  signal = 0.5,
  dichotomize = c("none", "influence", "awareness", "both"),
  vary = NULL,
  range = NULL
)

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

Arguments

`population`	A `usdt_population` object, or a `usdt_plan` from `usdt_plan()`, whose designs, hypotheses and settings are then used.
`subjects`	Number of subjects. Several values give one design each.
`trials`	Number of trials per subject in each task, signal and noise trials together: one value for both tasks, or one for each as `c(awareness = , influence = )` or `list(awareness = , influence = )`. Several values give one design each. A value shared by both tasks keeps them equal, and values given for each task are crossed.
`hypothesis`	Character vector of hypotheses, written as in `usdt_equivalence()`: `"<quantity> <op> <value>"` with the quantity `diff`, `rho`, `slope` or `intercept` and the operator `=`, `<` or `>`, or `"<quantity> in [a, b]"` and `"<quantity> out [a, b]"`. The default tests the three point nulls that `hsdt()` reports.
`power`	Power to reach (default 0.80).
`alpha`	Significance level of every test, between 0 and 0.5 (default 0.05).
`signal`	Proportion of signal trials in each task (default 0.5), one value for both tasks or one for each. Each task must have a whole number of signal and of noise trials.
`dichotomize`	Which tasks are split at the median, as in `usdt_data_tasks()`: `"none"`, `"influence"`, `"awareness"`, `"both"`, or one logical value for each task. The split fixes the criterion of a task at zero, so its criterion values in `population` are not used.

vary	Optional name of the population value to move instead of the tested quantity: "mu_A", "mu_I", "sigma_A", "sigma_I", "rho", "intercept", "criterion_A", "criterion_I", "sd_criterion_A" or "sd_criterion_I". Needs range.
range	With vary, the interval of values to search, c(from, to). The result is the value nearest to from that reaches the power.
x	A usdt_mdes object.
digits	Number of decimals printed.
...	Ignored.

Details

Moving a quantity keeps the other values of the population fixed as its parameterization says (see usdt_population()): the difference moves the influence mean with the awareness mean fixed. The intercept moves the influence mean with the means and SDs and the correlation fixed. The correlation (and the slope, through it) keeps both means under the correlation parameterization and the intercept under the latent regression. The search runs from the value of H0 (or the nearest bound) outwards, in the direction of H1, and the power along the way comes from the same draws of the sample mean and covariance as in usdt_power(), so it can level off below the target when fits on the boundary are frequent. The result is then NA.

Value

An object of class usdt_mdes: a data frame with one row per scenario, design and hypothesis, and the columns scenario, subjects, trials_awareness, trials_influence, hypothesis, varies (the value moved), population (its value in the population), power_population (the power there) and mdes (the value at which the power is reached, or NA if it is not reached in the range searched). For an equivalence test, varies is "margin" and mdes the half-width of the narrowest region. The attribute curve holds the power over the values searched.

See Also

usdt_power(), usdt_plan()

Examples

```
pop <- usdt_population(dprime    = c(awareness = 0.15, influence = 0.30),
                       sd_dprime = c(awareness = 0.35, influence = 0.30),
                       cor_dprime = 0.3, criterion = 0.1)

usdt_mdes(pop, subjects = 80, trials = c(awareness = 64, influence = 200),
          dichotomize = "influence")

# The narrowest equivalence margin for the difference
usdt_mdes(update(pop, dprime = c(awareness = 0.15, influence = 0.15)),
          subjects = 80, trials = c(awareness = 64, influence = 200),
          hypothesis = "diff in [-0.2, 0.2]", dichotomize = "influence")

# The SD of the awareness d' the intercept test needs
usdt_mdes(pop, subjects = 80, trials = c(awareness = 64, influence = 200),
```

```
hypothesis = "intercept > 0", vary = "sigma_A",
range = c(0.05, 0.6), dichotomize = "influence")
```

`usdt_plan` *Plan the subjects and trials of a uSDT study*

Description

Finds the design, the number of subjects and of trials per task, that reaches a given power for the tests of the uSDT hypotheses at the lowest cost, the design with the highest power within a budget, or the design that estimates a quantity with a given precision. What is free is given as a range and what is fixed as a single number, so the same function finds the subjects for given trials, the trials for given subjects, or both.

Usage

```
usdt_plan(
  population,
  hypothesis = c("diff = 0", "rho = 0", "intercept = 0"),
  power = 0.8,
  subjects = c(10, 1000),
  trials,
  cost = NULL,
  budget = NULL,
  precision = NULL,
  combine = c("each", "all"),
  alpha = 0.05,
  signal = 0.5,
  dichotomize = c("none", "influence", "awareness", "both")
)

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

Arguments

`population`	A `usdt_population` object from `usdt_population()`. With several scenarios, one plan for each.
`hypothesis`	Character vector of hypotheses, written as in `usdt_equivalence()`: `"<quantity> <op> <value>"` with the quantity `diff`, `rho`, `slope` or `intercept` and the operator =, < or >, or `"<quantity> in [a, b]"` and `"<quantity> out [a, b]"`. The default tests the three point nulls that `hsdt()` reports.
`power`	Power to reach (default 0.80).
`subjects`	Number of subjects: one number fixes it, and a range `c(min, max)` lets the plan choose it (default `c(10, 1000)`).

trials	Number of trials per subject in each task, signal and noise trials together: one number fixes it, and a range c(min, max) lets the plan choose it. One value or range for both tasks, which keeps them equal, or one for each as c(awareness = , influence =) or list(awareness = , influence =).
cost	Cost of a design, which the plan minimises: a named vector with the cost of each subject (a session, recruitment) and of each awareness and influence trial, in any unit (missing names cost 0), so that a design costs subjects * (subject + awareness * trials_awareness + influence * trials_influence), or a function of subjects, awareness and influence that returns the cost of each design. Needed when more than one of the subjects and the trials is free. With only one free, the plan minimises it.
budget	Largest cost allowed. With a budget, the plan finds the design with the highest power that costs no more, and power is not used.
precision	Named vector of the widths that the confidence intervals of usdt_tests() (level 1 - alpha) should not exceed, by quantity: diff, rho, intercept or slope, e.g. c(intercept = 0.2). With it, the plan finds the cheapest design whose expected interval is that narrow, and hypothesis and power are not used.
combine	"each" plans a design for each hypothesis. "all" plans one design in which every hypothesis rejects H0 in the same study with the given power.
alpha	Significance level of every test, between 0 and 0.5 (default 0.05).
signal	Proportion of signal trials in each task (default 0.5), one value for both tasks or one for each. Each task must have a whole number of signal and of noise trials.
dichotomize	Which tasks are split at the median, as in usdt_data_tasks(): "none", "influence", "awareness", "both", or one logical value for each task. The split fixes the criterion of a task at zero, so its criterion values in population are not used.
x	A usdt_plan object.
digits	Number of decimals printed.
...	Ignored.

Value

An object of class usdt_plan: a list with $design, a data frame with one row per scenario and hypothesis (or quantity, for a precision target) and the columns scenario, hypothesis, subjects, trials_awareness, trials_influence, power (approximate, as in usdt_power()), estimable, width (for a precision target), cost and limit (the free argument whose limit the design reaches, if any). $frontier records the smallest number of subjects that reaches the target with each number of trials, from the closed form. $checked holds the designs the search checked with the draws of the sample mean and covariance, with the number of subjects and cost of each. $settings stores the planning options. It prints one report per design.

How the design is found

The search follows the two approximations of usdt_power(). With each number of trials in the ranges, on a grid of at most 400 values per task, the closed form gives the smallest number of subjects that reaches the power, and with it the cost of each design. The cheapest designs are then checked with the draws of the sample mean and covariance, which see the fits on the boundary, and

the number of subjects of each is raised or lowered until those draws reach the power. The search stops when the next design costs more in the closed form than the best design checked so far. A design whose subjects or trials reach the limit of their range, or come within 2% of it, is marked: the range may be too narrow.

The design chosen this way rests on the approximation. Simulate it with `usdt_power(plan, method = "simulation")` before relying on it, as the approximation is somewhat optimistic for H2 and H3 when the awareness task is unreliable.

Costs

Many designs reach the same power, trading subjects for trials, so the choice among them depends on what each costs. With linear costs and the difference between means (H1), the cheapest number of trials of each task is close to $\sqrt{(c_s/c_t)(a_t/\tau^2)}$, where c_s is the cost of a subject, c_t that of a trial of the task, a_t the error variance of a single trial (about 2π near $d' = 0$) and τ^2 the variance of the difference between the two d' across subjects, whatever the power.

See Also

`usdt_power()`, `usdt_mdes()`, `usdt_population()`

Examples

```
pop <- usdt_population(dprime    = c(awareness = 0.15, influence = 0.30),
                       sd_dprime = c(awareness = 0.35, influence = 0.30),
                       cor_dprime = 0.3, criterion = 0.1)

# Subjects for given trials
usdt_plan(pop, hypothesis = "diff = 0",
          trials = c(awareness = 64, influence = 200),
          dichotomize = "influence")

# Subjects and trials together, with the session time of each in seconds
plan <- usdt_plan(pop, hypothesis = "diff = 0", power = 0.9,
                  subjects = c(20, 300),
                  trials = list(awareness = c(32, 160),
                                influence = c(100, 300)),
                  cost = c(subject = 600, awareness = 4, influence = 1.5),
                  dichotomize = "influence")
plan

# The most powerful design within 30 hours of sessions
usdt_plan(pop, hypothesis = "rho = 0", subjects = c(20, 400),
          trials = list(awareness = c(32, 256), influence = c(100, 800)),
          cost = c(subject = 600, awareness = 4, influence = 1.5),
          budget = 30 * 3600, dichotomize = "influence")

# The intercept within an interval 0.2 wide
usdt_plan(pop, precision = c(intercept = 0.2),
          trials = c(awareness = 64, influence = 200),
          dichotomize = "influence")
```

```
# The intercept, whose search checks more designs (several seconds)
usdt_plan(pop, hypothesis = "intercept > 0", power = 0.9,
          subjects = c(20, 300),
          trials = list(awareness = c(32, 160), influence = c(100, 300)),
          cost = c(subject = 600, awareness = 4, influence = 1.5),
          dichotomize = "influence")

# Check the chosen design by simulation
usdt_power(plan, method = "simulation", nsim = 50, seed = 1)
```

`usdt_population` *Population values for planning a uSDT study*

Description

Describes the population a planned study samples from: how awareness and influence sensitivity (d') vary and covary across subjects, and how the response criteria vary. The arguments take the names of `usdt_priors()`, with numbers instead of priors, and the population can be written in either parameterization of `hsdt()`.

Usage

```
usdt_population(
  dprime = NULL,
  sd_dprime = NULL,
  cor_dprime = NULL,
  criterion = 0,
  sd_criterion = 0.3,
  cor_criterion = 0,
  sd_signal = 1,
  intercept = NULL,
  from = NULL
)

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

## S3 method for class 'usdt_population'
update(object, ...)
```

Arguments

`dprime`	Mean d' of each task. Under the latent regression parameterization (when `intercept` is given), the mean of the awareness task only.
`sd_dprime`	Between-subject standard deviation of d' in each task.

cor_dprime	Correlation between awareness and influence d' across subjects.
criterion	Mean criterion c of each task, on the classical scale where positive values are conservative.
sd_criterion	Between-subject standard deviation of the criterion in each task. Zero gives every subject the same criterion.
cor_criterion	Correlation between the two criteria across subjects.
sd_signal	Standard deviation of the signal distribution of each task, relative to the noise distribution. The default, 1, is the equal-variance model that hsdt() fits.
intercept	Intercept of the latent regression of influence on awareness d'. Giving it selects the latent regression parameterization, in which it replaces the influence mean.
from	A frequentist hsdt() fit, such as a pilot study, whose estimates give every value not set in the call.
x	A usdt_population object.
...	For update(), the arguments of usdt_population() to change. A task argument may name one task only, which keeps the value of the other.
object	A usdt_population object.

Value

An object of class usdt_population: a data frame with one row per scenario and the columns scenario, mu_A, mu_I, sigma_A, sigma_I, rho, the tested quantities diff, intercept and slope, and criterion_A, criterion_I, sd_criterion_A, sd_criterion_I, cor_criterion, sd_signal_A and sd_signal_I. Its latent attribute is "correlation" or "regression".

Values for each task

dprime, sd_dprime, criterion, sd_criterion and sd_signal take one value for both tasks or one for each, as c(awareness = , influence =) or list(awareness = , influence =). The correlations and the intercept take a single value.

The two parameterizations

The correlation parameterization gives the bivariate normal distribution of d': its means μ_A and μ_I, standard deviations σ_A and σ_I and correlation ρ. The latent regression parameterization gives the intercept β_0 instead of μ_I, which then follows as $\mu_I = \beta_0 + \rho\,(\sigma_I/\sigma_A)\,\mu_A$, as in hsdt(latent = "regression"). Both describe the same distribution, and the printed population reports the tested quantities of either: the difference $\mu_I - \mu_A$ (H1), the correlation (H2) and the intercept and slope of the latent regression (H3).

The parameterization decides what stays fixed when a value changes, in a scenario or in update(). Under the correlation parameterization a new correlation keeps both means and moves the intercept (unless $\mu_A = 0$). Under the latent regression it keeps the intercept and moves the influence mean, and with it the difference.

Criteria

The criteria matter only through the precision with which each subject's trials measure d'. A task split at the median has its criterion fixed at zero by the split, whatever its value here. The criterion correlation applies when both criteria are estimated.

Scenarios

Several values in an argument give one scenario for each, and several arguments with several values give one scenario for each combination. A value shared by both tasks keeps them equal in every scenario: `sd_dprime = c(0.2, 0.3)` gives two scenarios, while `sd_dprime = list(awareness = c(0.2, 0.3), influence = c(0.2, 0.3))` gives four.

Values from a fitted model

With `from`, the estimates of a frequentist `hsdt()` fit give the means, standard deviations and correlations, and the arguments set in the call replace them, so `usdt_population(from = m, cor_dprime = 0.3)` keeps every estimate of `m` except the correlation. Setting `intercept` switches to the latent regression parameterization and drops the estimated influence mean. A criterion fixed at zero by the median split is zero with no variation.

See Also

`usdt_priors()`, `hsdt()`

Examples

```
# The correlation parameterization: means, SDs and correlation of d'
pop <- usdt_population(dprime    = c(awareness = 0.15, influence = 0.30),
                       sd_dprime = c(awareness = 0.35, influence = 0.30),
                       cor_dprime = 0.3)
pop

# The latent regression parameterization: the intercept replaces the
# influence mean
usdt_population(dprime = 0.15, intercept = 0.25,
                sd_dprime = c(awareness = 0.35, influence = 0.30),
                cor_dprime = 0.3)

# Three scenarios for the correlation
update(pop, cor_dprime = c(0.1, 0.3, 0.5))

# The estimates of a pilot study, with a weaker correlation
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)
usdt_population(from = hsdt(d), cor_dprime = 0.3)
```

usdt_power *Power of a planned uSDT study*

Description

Computes the power of the tests of the three uSDT hypotheses, the difference between mean sensitivities (H1), their correlation (H2) and the latent regression (H3), for a study that samples a given number of subjects, each with a given number of trials per task, from a population described by `usdt_population()`.

Usage

```
usdt_power(
  population,
  subjects,
  trials,
  hypothesis = c("diff = 0", "rho = 0", "intercept = 0"),
  alpha = 0.05,
  signal = 0.5,
  dichotomize = c("none", "influence", "awareness", "both"),
  combine = c("each", "all"),
  method = c("approximate", "simulation"),
  nsim = 1000,
  seed = NULL,
  ncores = 1L,
  progress = interactive()
)

## S3 method for class 'usdt_power'
print(x, digits = 3L, ...)

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

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

Arguments

`population`	A `usdt_population` object from `usdt_population()`. With several scenarios, the power of each. It can also be a `usdt_plan` from `usdt_plan()`, whose designs, hypotheses and settings are then used, so that `usdt_power(plan, method = "simulation")` checks a plan. `subjects` and `trials` are then left out.
`subjects`	Number of subjects. Several values give one design each.
`trials`	Number of trials per subject in each task, signal and noise trials together: one value for both tasks, or one for each as `c(awareness = , influence = )` or

	`list(awareness = , influence = )`. Several values give one design each. A value shared by both tasks keeps them equal, and values given for each task are crossed.
`hypothesis`	Character vector of hypotheses, written as in `usdt_equivalence()`: `"<quantity> <op> <value>"` with the quantity `diff`, `rho`, `slope` or `intercept` and the operator `=`, `<` or `>`, or `"<quantity> in [a, b]"` and `"<quantity> out [a, b]"`. The default tests the three point nulls that `hsdt()` reports.
`alpha`	Significance level of every test, between 0 and 0.5 (default 0.05).
`signal`	Proportion of signal trials in each task (default 0.5), one value for both tasks or one for each. Each task must have a whole number of signal and of noise trials.
`dichotomize`	Which tasks are split at the median, as in `usdt_data_tasks()`: `"none"`, `"influence"`, `"awareness"`, `"both"`, or one logical value for each task. The split fixes the criterion of a task at zero, so its criterion values in `population` are not used.
`combine`	`"each"` reports the power of each hypothesis. `"all"` also reports the probability that every one of them rejects H0 in the same study.
`method`	`"approximate"` (the default) computes power from the sampling distribution of the estimates, in seconds. `"simulation"` simulates whole studies and analyses each one with `hsdt()`, which takes minutes. See Details.
`nsim`	Number of simulated studies for each scenario and design (default 1000). Simulation only.
`seed`	Random seed of the simulated studies. With the default `NULL`, the seed is drawn from the current random number stream. Simulation only.
`ncores`	Number of CPU cores. Values above 1 fit the simulated studies in parallel on a temporary cluster, which works on Windows, macOS and Linux and stops when the run ends. Simulation only.
`progress`	Logical. Show the expected duration and a progress bar (default `TRUE` in interactive sessions). Simulation only.
`x`	A `usdt_power` object.
`digits`	Number of decimals printed.
`...`	Ignored.
`object`	A `usdt_power` object from a simulation.

Value

An object of class `usdt_power`: a data frame with one row per scenario, design and hypothesis, and the columns `scenario`, `subjects`, `trials_awareness`, `trials_influence`, `hypothesis`, `test`, `H1`, `H0`, `true` (the population value of the tested quantity), `se` (its expected standard error), `power`, `estimable` (the share of studies in which the analysis can estimate it), `in_null` (whether the population lies in H0, so that `power` is the type I error rate), and `reliability_awareness` and `reliability_influence`. A simulation adds `mc_se`, the Monte Carlo standard error of `power`, and `power.low` and `power.high`, its 95% Wilson interval. With `combine = "all"`, a row per scenario and design whose `hypothesis` is `"all"` gives the joint power. It prints one report for a single design and a table for several.

For a simulation, `summary()` reports the precision of each quantity across the simulated studies that estimate it: the mean estimate, its bias, the empirical standard error (the SD of the estimates),

the mean standard error the fits report, the mean width of the intervals and their coverage of the population value.

The approximation

Each subject's estimated d' is treated as its true d' plus a measurement error. The variance of that error, v, is the one `usdt_reliability()` reports, averaged over the subjects of the population: it falls with the number of trials and grows with extreme rates. The observed d' of the subjects then have covariance $S = \Sigma + \mathrm{diag}(v_A, v_I)$, where Σ is the covariance of the population, and the reliability of each task is $\sigma^2/(\sigma^2 + v)$. Trials act on the tests only through v, and subjects through their number N.

With N subjects, the sample mean of their d' is normal with covariance S/N and their sample covariance has a Wishart distribution. `usdt_power()` draws both 20,000 times, subtracts the error variance to estimate Σ, and in each draw runs every test as the fitted model would, with the standard errors it would compute at those estimates. Power is the share of draws that reject H0, with a Monte Carlo error of at most 0.004. The draws are the same for every design, so power changes smoothly with the design, and they do not change the random number stream of the session.

A draw whose estimated covariance of the sensitivities is not positive definite (a variance of zero or a correlation of -1 or 1) is a fit on the boundary. There, the fitted model reports the difference but marks the correlation, slope and intercept as not estimable, so the draw counts as a failure to reject them, and `estimable` reports how often that happens. With an awareness task of low reliability R_A, this probability is close to $\Phi(-R_A\sqrt{N/2})$, and more awareness trials lower it faster than more subjects.

The tests

Each hypothesis is decided as the analysis would decide it: `"rho = 0"` and `"slope = 0"` by the test on the covariance of `usdt_tests()`, and every other statement by the tests of `usdt_equivalence()`, with the correlation on the Fisher-z scale. `se` is the standard error at the population values. When the population value lies in H0, `power` is the type I error rate, and the report says so.

Limits of the approximation

The approximation assumes the equal-variance model that `hsdt()` fits, so every `sd_signal` of `population` must be 1, and it assumes the estimated standard deviations of the criteria stay away from zero. It follows the fitted model closely when the reliability of both tasks is moderate or high. When the awareness task is unreliable, boundary fits are more frequent in the fitted model than here, and the power of H2 and H3 is somewhat optimistic.

Simulation

With `method = "simulation"`, each study draws its subjects from the population and their trials from their d' and criteria. A task split at the median draws one latent score per trial and is split with `meyen_split()`, as response times would be. Any other task draws binomial counts. The study is then analysed by `hsdt()` with the confidence level `1 - alpha`, and each hypothesis is decided as in the approximation. A fit that fails, or that cannot estimate the quantity, counts as a failure to reject. Each study has its own random number stream, derived from `seed`, so a run gives the same studies whatever `ncores`, and every design uses the same streams, which makes the differences between designs less noisy than their Monte Carlo errors suggest. The simulation also accepts a

population with unequal variances (`sd_signal` other than 1), whose studies are then analysed with the equal-variance model. `se` and the reliabilities are then `NA`.

A simulation is slow, about a second per study with 100 subjects, so it is best kept for checking a design chosen with the approximation, with enough studies to bound the Monte Carlo error: 1000 studies give an error of at most 0.016.

See Also

`usdt_population()`, `usdt_tests()`, `usdt_equivalence()`, `usdt_reliability()`

Examples

```
pop <- usdt_population(dprime    = c(awareness = 0.15, influence = 0.30),
                       sd_dprime = c(awareness = 0.35, influence = 0.30),
                       cor_dprime = 0.3, criterion = 0.1)

# 80 subjects, 64 awareness trials and 200 influence trials split at the
# median
usdt_power(pop, subjects = 80, trials = c(awareness = 64, influence = 200),
           dichotomize = "influence")

# Several numbers of subjects, and one-sided and equivalence tests
usdt_power(pop, subjects = c(40, 80, 160),
           trials = c(awareness = 64, influence = 200),
           hypothesis = c("diff > 0", "intercept > 0", "rho in [-0.1, 0.1]"),
           dichotomize = "influence")

# The same design under weaker correlations
usdt_power(update(pop, cor_dprime = c(0.1, 0.2)), subjects = 80,
           trials = c(awareness = 64,
                      influence = 200), dichotomize = "influence")


# Check the design by simulating and analysing 50 studies (under a
# minute). A real check needs more, e.g. nsim = 1000 with ncores = 2.
sims <- usdt_power(pop, subjects = 80,
                   trials = c(awareness = 64, influence = 200),
                   dichotomize = "influence", method = "simulation",
                   nsim = 50, seed = 1)
sims

# Bias, empirical and model standard errors, interval width and coverage
summary(sims)
```

`usdt_priors` *Priors for the Bayesian hierarchical SDT model*

Description

Specifies the prior distributions of the Bayesian model. Each prior is written as a string in Stan notation, `"family(arguments)"`, with the families and parameterisations of Stan, e.g. `"normal(0, 1)"`, `"student_t(4, 0, 0.5)"` or `"gamma(2, 4)"`. Each argument takes the families that cover the values its parameter can take:

- means of d' and of the criterion can take any real value, so their prior must be defined on the whole real line: `normal`, `std_normal`, `student_t`, `cauchy`, `double_exponential`, `logistic`, `gumbel`, `skew_normal`, `exp_mod_normal` or `skew_double_exponential`.
- standard deviations take any of those, truncated at zero (with location 0, a `student_t` is a half-Student-t), or a family on the positive values: `lognormal`, `exponential`, `gamma`, `inv_gamma`, `weibull`, `frechet`, `rayleigh`, `chi_square`, `inv_chi_square`, `scaled_inv_chi_square`, `loglogistic`, or `pareto_type_2` with `mu <= 0`.
- correlations take `"lkj(eta)"`, the LKJ prior on a correlation matrix and the only correlation prior brms accepts. On a 2 x 2 matrix the correlation follows a beta(eta, eta) distribution stretched from $(0, 1)$ to $(-1, 1)$: `lkj(1)` is uniform, and larger values of eta concentrate it around zero.

Usage

```
usdt_priors(
  dprime = "normal(0, 1)",
  sd_dprime = "student_t(4, 0, 0.5)",
  cor_dprime = "lkj(1)",
  criterion = "normal(0, 0.5)",
  sd_criterion = "student_t(4, 0, 0.5)",
  cor_criterion = "lkj(1)",
  sd_signal = "lognormal(0, 0.5)",
  intercept = "normal(0, 1)"
)

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

Arguments

`dprime`	Prior of the mean d' of each task.
`sd_dprime`	Prior of the between-subject standard deviation of d' in each task.
`cor_dprime`	Prior of the correlation between awareness and influence d' across subjects.
`criterion`	Prior of the mean criterion c of each task, on the classical scale where positive values are conservative.
`sd_criterion`	Prior of the between-subject standard deviation of the criterion in each task.
`cor_criterion`	Prior of the correlation between the two criteria. It is used only when both criteria are estimated.
`sd_signal`	Prior of the standard deviation of the signal distribution, relative to the noise distribution. It is used only under unequal variances.

`intercept`	Prior of the intercept of the latent regression of influence on awareness d'. It is used only by the latent regression parameterization, where it replaces the prior of the influence mean d'.
`x`	A `usdt_priors` object.
`...`	Ignored.

Details

`dprime`, `sd_dprime`, `criterion`, `sd_criterion` and `sd_signal` take one prior for both tasks or one per task as `list(awareness = ..., influence = ...)`. The correlations take a single prior.

The defaults are centred at zero and weakly informative on the probit scale: `normal(0, 1)` for the mean d', `normal(0, 0.5)` for the mean criterion, `student_t(4, 0, 0.5)` for every between-subject standard deviation and a uniform `lkj(1)` for every correlation. With them, the hit and false-alarm rates of a new subject spread over $(0, 1)$ without piling up at either end. See `plot.usdt_priors()`. With the same prior for both means, the prior of the difference between them (H1) is symmetric around zero, and every `lkj` prior is centred at zero (H2).

The model describes the two sensitivities by their means, standard deviations and correlation, so the slope and intercept of the latent regression (H3) have no priors of their own: their priors follow from those of the means, standard deviations and correlation. With the defaults, the intercept prior is symmetric around zero but has heavy tails. It does not change when both standard deviation priors are rescaled by the same factor, because the slope depends on their ratio.

The latent regression parameterization moves the prior of the influence mean to the intercept β_0 of the latent regression, so that both the intercept (H3) and the correlation (H2) have the priors you give them. It keeps the priors of the awareness mean, of both standard deviations and of the correlation, and the influence mean follows from them as $\mu_I = \beta_0 + \rho\,(\sigma_I/\sigma_A)\,\mu_A$. `dprime` then applies to the awareness task only. The intercept takes `normal(0, 1)` by default. Inspect its implications with `plot.usdt_priors()`. Two things follow:

- the difference between the means (H1) has an induced prior, $\beta_0 + (\rho\sigma_I/\sigma_A - 1)\mu_A$, with the heavy tails the intercept has under the correlation parameterization.
- the intercept and the correlation are independent a priori but not a posteriori. The data fix the influence mean, the sum of β_0 and $\rho(\sigma_I/\sigma_A)\mu_A$, so a narrow intercept prior also narrows the posterior of ρ, mostly when σ_A is small. The Bayes factor of a correlation hypothesis then depends on the intercept prior too.

Value

An object of class `usdt_priors`: a data frame with one row per prior, giving the parameter, the task, the family, its arguments (a list column) and the prior as a string.

Examples

```
# Default priors
usdt_priors()

# A narrower prior for the mean influence sensitivity
usdt_priors(dprime = list(awareness = "normal(0, 1)",
                          influence = "normal(0, 0.5)"))
```

```
# A correlation prior that concentrates near zero
usdt_priors(cor_dprime = "lkj(2)")

# Standard deviations of d' expected around 0.3
usdt_priors(sd_dprime = "student_t(4, 0.3, 0.2)")

# Any family that covers the positive values, e.g. a gamma distribution
usdt_priors(sd_dprime = "gamma(2, 4)")
```

`usdt_reliability` *Reliability of awareness and influence task measures*

Description

Estimates the reliability of sensitivity (d') by separating true variance across participants from sampling noise caused by finite trial counts. Values close to 1 indicate that the measure reliably separates participants, whereas values near 0 indicate that observed differences are mostly measurement error.

Usage

```
usdt_reliability(object)

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

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

Arguments

`object`	An `hsdt` object from `hsdt()`, frequentist or Bayesian. A frequentist fit may also contain bootstrap results from `usdt_boot()`.
`...`	Ignored.
`x`	A `usdt_reliability` object.

Details

For participant i in task j, reliability is defined as:

$$\frac{\tau_j^2}{\tau_j^2 + v_{ij}}$$

where τ_j^2 is the true variance in sensitivity across participants from the model's random effects, and v_{ij} is the squared standard error of d' for that participant. This error variance reflects how precisely their trials determine sensitivity, accounting for trial count, performance level on the probit curve, and uncertainty in the criterion.

This variance is calculated using the large-sample Fisher information formula from Gourevitch and Galanter (1967). Unlike `sdt_moments()`, which evaluates that formula at raw empirical proportions (`var_gg`), `usdt_reliability()` evaluates it at the response probabilities predicted by the fitted hierarchical model.

Overall task reliability averages v_{ij} across participants, representing the expected proportion of true variance for a participant drawn at random from the sample.

When `object` includes bootstrap replicates from `usdt_boot()`, confidence intervals for reliability are computed automatically across all retained samples.

For a Bayesian fit, reliability is computed draw by draw: each draw gives τ_j^2 and, at that draw's subject sensitivities and criteria, each v_{ij}. The tables report posterior means, posterior standard deviations and central credible intervals. Under unequal variances the information of the signal trials is scaled by the signal standard deviation σ_s, so that $v_{ij} = \sigma_s^2/w_{\text{signal}} + 1/w_{\text{noise}}$ with an estimated criterion, which reduces to the Gourevitch and Galanter formula when $\sigma_s = 1$.

Value

An object of class `usdt_reliability` containing:

- `$tasks`: Overall reliability and variance components for each task.
- `$subjects`: Participant-level d', error variances, and individual reliabilities.
- Bootstrap intervals for both components when available in `object`.

For a Bayesian fit, both tables hold posterior means and add the columns `sd`, `conf.low` and `conf.high`, and `$posterior$tasks` keeps the draws of each task's reliability.

References

Gourevitch, V., & Galanter, E. (1967). A significance test for one parameter isosensitivity functions. *Psychometrika*, 32(1), 25–33. doi:10.1007/BF02289402

See Also

`hsdt()`, `usdt_boot()`, `sdt_moments()`

Examples

```
# Contextual cuing data from Vadillo et al. (2025)
d <- usdt_data_tasks(
  awareness = vadillo_awareness,
  influence = vadillo_cuing,
  subject_col      = "subj",
  condition_col    = "condition",
  condition_levels = c(signal = "old", noise = "new"),
  response_col     = list(awareness = "judged.old", influence = "rt"),
  response_levels  = list(awareness = c(signal = 1, noise = 0),
                          influence = c(signal = "faster",
                                        noise = "slower")),
  dichotomize      = list(awareness = FALSE, influence = TRUE)
)
```

```
r <- usdt_reliability(hsdt(d))
r

# Participant-level estimates (one row per subject and task)
head(r$subjects)
```

`vadillo_awareness` *Awareness data from a probabilistic cuing experiment*

Description

Trial-level awareness data from Experiment 2 of Vadillo et al. (2025). The object reproduces the source CSV file without filtering or recoding and can be used as the `awareness` input to `usdt_data_tasks()`.

Usage

```
vadillo_awareness
```

Format

A data frame with 6,656 rows, 104 participants, and 10 variables:

`subj` Participant identifier.

`file` Original participant file name.

`trial` Trial number.

`pattId` Pattern identifier.

`judgm` Awareness judgement on the original response scale.

`judged.old` Whether the pattern was judged old (1) or new (0).

`condition` Whether the pattern was old or new.

`offset` Spatial-offset condition.

`color` Colour condition.

`set.size` Set-size condition.

Source

Vadillo, M. A., Malejka, S., & Shanks, D. R. (2025). Mapping the reliability multiverse of contextual cuing. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. doi:10.1037/xlm0001410. Data retrieved from `https://osf.io/jp3gx/`.

Examples

```
data(vadillo_awareness)
str(vadillo_awareness)
```

vadillo_cuing *Cuing data from a probabilistic cuing experiment*

Description

Trial-level influence-task data from Experiment 2 of Vadillo et al. (2025). The object reproduces the source CSV file without filtering or recoding and can be used as the `influence` input to `usdt_data_tasks()`.

Usage

```
vadillo_cuing
```

Format

A data frame with 39,936 rows, 104 participants, and 13 variables:

`experiment` Experiment label.

`subj` Participant identifier.

`file` Original participant file name.

`trial` Trial number.

`block` Block number.

`epoch` Epoch number.

`pattId` Pattern identifier.

`acc` Response accuracy (1 = correct, 0 = incorrect).

`rt` Response time in milliseconds.

`condition` Whether the pattern was old or new.

`offset` Spatial-offset condition.

`color` Colour condition.

`set.size` Set-size condition.

Source

Vadillo, M. A., Malejka, S., & Shanks, D. R. (2025). Mapping the reliability multiverse of contextual cuing. *Journal of Experimental Psychology: Learning, Memory, and Cognition*. doi:10.1037/xlm0001410. Data retrieved from `https://osf.io/jp3gx/`.

Examples

```
data(vadillo_cuing)
str(vadillo_cuing)
```

Index

∗ **datasets**
vadillo_awareness, 49
vadillo_cuing, 50

brms::brm(), *30, 31*
brms::pp_check(), *4*
brms::prior_summary(), *4*, *19*

hsdt, 2
hsdt(), *7*, *9*, *11*, *16, 17*, *20–23*, *25*, *27*, *30, 31*, *33*, *35*, *38–40*, *42, 43*, *47, 48*

latent_cor *(*usdt_hypotheses*)*, 29
latent_regression *(*usdt_hypotheses*)*, 29
lme4::bootMer(), *21*
lme4::glmer(), *2*, *4*, *27*, *30, 31*
lme4::glmerControl(), *3*

meyen_split, 6
meyen_split(), *24, 25*, *43*

plot.hsdt, 7
plot.hsdt(), *5*
plot.usdt_bf, 10
plot.usdt_bf(), *18*, *20*
plot.usdt_mdes *(*plot_power*)*, 12
plot.usdt_plan *(*plot_power*)*, 12
plot.usdt_population *(*plot_power*)*, 12
plot.usdt_power *(*plot_power*)*, 12
plot.usdt_priors, 11
plot.usdt_priors(), *5*, *46*
plot_power, 12
print.hsdt *(*hsdt*)*, 2
print.summary.usdt_power *(*usdt_power*)*, 41
print.usdt_bf *(*usdt_bf*)*, 17
print.usdt_data *(*usdt_data*)*, 23
print.usdt_equivalence *(*usdt_equivalence*)*, 26
print.usdt_mdes *(*usdt_mdes*)*, 32
print.usdt_plan *(*usdt_plan*)*, 35
print.usdt_population *(*usdt_population*)*, 38
print.usdt_power *(*usdt_power*)*, 41
print.usdt_priors *(*usdt_priors*)*, 44
print.usdt_reliability *(*usdt_reliability*)*, 47

sdt_moments, 14
sdt_moments(), *8, 9*, *25*, *48*
sensitivity_diff *(*usdt_hypotheses*)*, 29
set.seed(), *19*
summary.hsdt *(*hsdt*)*, 2
summary.usdt_power *(*usdt_power*)*, 41
summary.usdt_reliability *(*usdt_reliability*)*, 47

update(), *39*
update.usdt_population *(*usdt_population*)*, 38
usdt_aggregate, 16
usdt_aggregate(), *31*
usdt_bf, 17
usdt_bf(), *10*, *12*, *27–29*
usdt_boot, 21
usdt_boot(), *4, 5*, *8, 9*, *27–29*, *31*, *47, 48*
usdt_data, 23
usdt_data_long *(*usdt_data*)*, 23
usdt_data_long(), *3*
usdt_data_tasks *(*usdt_data*)*, 23
usdt_data_tasks(), *3*, *5*, *33*, *36*, *42*, *49, 50*
usdt_equivalence, 26
usdt_equivalence(), *33*, *35*, *42–44*
usdt_hypotheses, *16, 17*, 29
usdt_mdes, 32
usdt_mdes(), *13*, *37*
usdt_plan, 35
usdt_plan(), *13*, *33, 34*, *41*
usdt_population, 38
usdt_population(), *13*, *34, 35*, *37*, *41*, *44*
usdt_power, 41

usdt_power(), *13*, *33*, *34*, *36*, *37*
usdt_priors, 44
usdt_priors(), *3–5*, *11*, *12*, *19*, *20*, *38*, *40*
usdt_reliability, 47
usdt_reliability(), *43*, *44*
usdt_tests *(*usdt_hypotheses*)*, 29
usdt_tests(), *5*, *18*, *22*, *27–29*, *43*, *44*

vadillo_awareness, 49
vadillo_cuing, 50