Package ‘dynConfiR’

May 8, 2025

Type Package

Title Dynamic Models for Confidence and Response Time Distributions

Version 1.0.0

Maintainer Sebastian Hellmann <`sebastian.hellmann@tum.de`>

Description Provides density functions for the joint distribution of choice, response time and confidence for discrete confidence judgments as well as functions for parameter fitting, prediction and simulation for various dynamical models of decision confidence. All models are explained in detail by Hellmann et al. (2023; Preprint available at <`https://osf.io/9jfqr/`>, published version: <doi:10.1037/rev0000411>). Implemented models are the dynaViTE model, dynWEV model, the 2DSD model (Pleskac & Busemeyer, 2010, <doi:10.1037/a0019737>), and various race models. C++ code for dynWEV and 2DSD is based on the 'rtdists' package by Henrik Singmann.

License GPL (>= 3)

URL `https://github.com/SeHellmann/dynConfiR`, `https://sehellmann.github.io/dynConfiR/`

BugReports `https://github.com/SeHellmann/dynConfiR/issues`

Depends R (>= 4.0)

Imports dplyr, magrittr, minqa, parallel, progress, Rcpp, rlang, stats

Suggests covr, ggplot2, Hmisc, knitr, logger, rmarkdown, testthat (>= 3.0.0), tidyr

LinkingTo Rcpp

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.3.1

Author Sebastian Hellmann [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3621-6343>),
Manuel Rausch [aut, fnd] (ORCID:
<https://orcid.org/0000-0002-5805-5544>)

Date/Publication 2025-05-08 17:00:02 UTC

Contents

`dynConfiR-package` *The dynConfiR Package*

Description

Dynamic Models for Confidence and Response Time Distributions

Details

Package:	dynConfiR
Type:	Package
Version:	0.1.0
Date:	2023-06-19

Depends: R (>= 4.0)
License: GPL (>=3)
URL: https://github.com/SeHellmann/dynConfiR

Provides response time and confidence distributions (density/PDF) for following models: dynaViTE,dynWEV, 2DSD, 2DSDT, IRM and PCRM

Author(s)

Sebastian Hellmann

`ConfidenceOrientation` *Confidence and response time data*

Description

A data set containing results from an orientation discrimination experiment with confidence judgments. The data set includes results from 16 participants and 3 sessions. The task was to identify the orientation (horizontal or vertical) of a grid that was briefly visible and then covered by a mask in form of a checkerboard pattern.

Usage

```
data(ConfidenceOrientation)
```

Format

A data frame with 25920 rows and 12 variables:

participant integer values as unique participant identifier

session session identifier ranging from 1 to 3

gender gender of the participant: "w" for female; "m" for male participants

age the age of participants in years

SOA stimulus-onset-asynchrony in ms (i.e. time between stimulus and mask onset)

orientation orientation of the target stimulus (0: vertical, 90: horizontal)

stimulus stimulus identity ("senkrecht": vertical, "waagrecht": horizontal)

response response for the discrimination task (see stimulus column)

correct 0-1 column indicating whether the discrimination response was correct (1) or not (0)

rt response time for the discrimination response in sec

cont_rating confidence rating as registered (continuous values ranging from -1 (unsure) to 1 (sure))

disc_rating confidence rating discretized in 5 steps using equidistant breaks

Source

`https://github.com/SeHellmann/SeqSamplingConfidenceModels`

`d2DSD` *Pleskac and Busemeyer's 2DSD Model for Decision Confidence*

Description

Likelihood function and random number generator for a generalization of the 2DSD Model presented by Pleskac & Busemeyer (2010). It includes following parameters: DDM parameters: `a` (threshold separation), `z` (starting point; relative), `v` (drift rate), `t0` (non-decision time/ response time constant), `d` (differences in speed of response execution), `sv` (inter-trial-variability of drift), `st0` (inter-trial-variability of non-decisional components), `sz` (inter-trial-variability of relative starting point), `s` (diffusion constant).

Usage

```
d2DSD(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5, d = 0,
  sz = 0, sv = 0, st0 = 0, tau = 1, lambda = 0, s = 1,
  simult_conf = FALSE, precision = 6, z_absolute = FALSE,
  stop_on_error = TRUE, stop_on_zero = FALSE)

r2DSD(n, a, v, t0 = 0, z = 0.5, d = 0, sz = 0, sv = 0, st0 = 0,
  tau = 1, lambda = 0, s = 1, delta = 0.01, maxrt = 15,
  simult_conf = FALSE, z_absolute = FALSE, stop_on_error = TRUE)
```

Arguments

`rt`	a vector of RTs. Or for convenience also a `data.frame` with columns `rt` and `response`.
`response`	character vector, indicating the decision, i.e. which boundary was met first. Possible values are `c("upper", "lower")` (possibly abbreviated) and `"upper"` being the default. Alternatively, a numeric vector with values 1=lower and 2=upper or -1=lower and 1=upper, respectively. For convenience, `response` is converted via `as.numeric` also allowing factors. Ignored if the first argument is a `data.frame`.
`th1`	together with `th2`: scalars or numerical vectors giving the lower and upper bound of the interval, in which the accumulator should end at the time of the confidence judgment (i.e. at time `rt+tau`). Only values with `th2>=th1` are accepted.
`th2`	(see `th1`)
`a`	threshold separation. Amount of information that is considered for a decision. Large values indicate a conservative decisional style. Typical range: 0.5 < `a` < 2
`v`	drift rate. Average slope of the information accumulation process. The drift gives information about the speed and direction of the accumulation of information. Large (absolute) values of drift indicate a good performance. If received information supports the response linked to the upper threshold the sign will be positive and vice versa. Typical range: -5 < `v` < 5

`t0`	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution). Typical range: 0.1 < `t0` < 0.5. Default is 0.
`z`	(by default relative) starting point. Indicator of an a priori bias in decision making. When the relative starting point `z` deviates from `0.5`, the amount of information necessary for a decision differs between response alternatives. Default is `0.5` (i.e., no bias).
`d`	differences in speed of response execution (in seconds). Positive values indicate that response execution is faster for responses linked to the upper threshold than for responses linked to the lower threshold. Typical range: -0.1 < `d` < 0.1. Default is 0.
`sz`	inter-trial-variability of starting point. Range of a uniform distribution with mean `z` describing the distribution of actual starting points from specific trials. Values different from 0 can predict fast errors (but can slow computation considerably). Typical range: 0 < `sz` < 0.2. Default is 0. (Given in relative range i.e. bounded by 2*min(z, 1-z))
`sv`	inter-trial-variability of drift rate. Standard deviation of a normal distribution with mean `v` describing the distribution of actual drift rates from specific trials. Values different from 0 can predict slow errors. Typical range: 0 < `sv` < 2. Default is 0.
`st0`	inter-trial-variability of non-decisional components. Range of a uniform distribution with mean `t0 + st0/2` describing the distribution of actual `t0` values across trials. Accounts for response times below `t0`. Reduces skew of predicted RT distributions. Values different from 0 can slow computation considerably. Typical range: 0 < `st0` < 0.2. Default is 0.
`tau`	post-decisional accumulation time. The length of the time period after the decision was made until the confidence judgment is made. Range: `tau`>0. Default: `tau`=1.
`lambda`	power for judgment time in the division of the confidence measure by the judgment time (Default: 0, i.e. no division which is the version of 2DSD proposed by Pleskac and Busemeyer)
`s`	diffusion constant. Standard deviation of the random noise of the diffusion process (i.e., within-trial variability), scales `a`, `v`, `sv`, and `th`'s. Needs to be fixed to a constant in most applications. Default is 1. Note that the default used by Ratcliff and in other applications is often 0.1.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and computations are different, when there is an observable interjudgment time (then `simult_conf` should be `FALSE`).
`precision`	`numerical` scalar value. Precision of calculation. Determines the the stepsize of integration w.r.t. `z` and `t0`. Represents the number of decimals precisely computed on average. Default is 6.
`z_absolute`	logical. Determines whether `z` is treated as absolute start point (`TRUE`) or relative (`FALSE`; default) to `a`.
`stop_on_error`	Should the diffusion functions return 0 if the parameters values are outside the allowed range (= `FALSE`) or produce an error in this case (= `TRUE`).

`stop_on_zero`	Should the computation of densities stop as soon as a density value of 0 occurs. This may save a lot of time if the function is used for a likelihood function. Default: FALSE
`n`	integer. The number of samples generated.
`delta`	numeric. Discretization step size for simulations in the stochastic process
`maxrt`	numeric. Maximum decision time returned. If the simulation of the stochastic process exceeds a decision time of `maxrt`, the `response` will be set to 0 and the `maxrt` will be returned as `rt`.

Details

For confidence: `tau` (post-decisional accumulation time), `lambda` the exponent of judgment time for the division by judgment time in the confidence measure, `th1` and `th2` (lower and upper thresholds for confidence interval).

Note that the parameterization or defaults of non-decision time variability `st0` and diffusion constant `s` differ from what is often found in the literature.

The drift diffusion model (DDM; Ratcliff and McKoon, 2008) is a mathematical model for two-choice discrimination tasks. It is based on the assumption that information is accumulated continuously until one of two decision thresholds is hit. For introduction see Ratcliff and McKoon (2008).

The 2DSD is an extension of the DDM to explain confidence judgments based on the preceding decision. It assumes a post decisional period where the process continues the accumulation of information. At the end of the period a confidence judgment (i.e. a judgment of the probability that the decision was correct) is made based on the state of the process. Here, we use a given interval, given by `th1` and `th2`, assuming that the data is given with discrete judgments and pre-processed, s.t. these discrete ratings are translated to the respective intervals. The 2DSD Model was proposed by Pleskac and Busemeyer (2010).

All functions are fully vectorized across all parameters as well as the response to match the length or `rt` (i.e., the output is always of length equal to `rt`). This allows for trial wise parameters for each model parameter.

For convenience, the function allows that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt` and `response`). Other columns (as well as passing `response` separately argument) will be ignored.

Value

`d2DSD` gives the density/likelihood/probability of the diffusion process producing a decision of `response` at time `rt` and a confidence judgment corresponding to the interval [`th1`, `th2`]. The value will be a numeric vector of the same length as `rt`.

`r2DSD` returns a `data.frame` with three columns and `n` rows. Column names are `rt` (response time), `response` (-1 (lower) or 1 (upper), indicating which bound was hit), and `conf` (the value of the confidence measure; not discretized!).

The distribution parameters (as well as `response`, `tau`, `th1` and `th2`) are recycled to the length of the result. In other words, the functions are completely vectorized for all parameters and even the response boundary.

Note

The parameterization of the non-decisional components, `t0` and `st0`, differs from the parameterization sometimes used in the literature. In the present case `t0` is the lower bound of the uniform distribution of length `st0`, but *not* its midpoint. The parameterization employed here is in line with the functions in the `rtdists` package.

The default diffusion constant `s` is 1 and not 0.1 as in most applications of Roger Ratcliff and others. Usually `s` is not specified as the other parameters: `a`, `v`, and `sv`, may be scaled to produce the same distributions (as is done in the code).

The function code is basically an extension of the `ddiffusion` function from the package `rtdists` for the Ratcliff diffusion model.

Author(s)

For the original `rtdists` package: Underlying C code by Jochen Voss and Andreas Voss. Porting and R wrapping by Matthew Gretton, Andrew Heathcote, Scott Brown, and Henrik Singmann. `qdiffusion` by Henrik Singmann. For the `d2DSD` function the C code was extended by Sebastian Hellmann.

References

Pleskac, T. J., & Busemeyer, J. R. (2010). Two-Stage Dynamic Signal Detection: A Theory of Choice, Decision Time, and Confidence, *Psychological Review*, 117(3), 864-901. doi:10.1037/a0019737

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, 20(4), 873-922.

Examples

```
# Plot rt distribution ignoring confidence
curve(d2DSD(x, "upper", -Inf, Inf, tau=1, a=2, v=0.4, sz=0.2, sv=0.9), xlim=c(0, 2), lty=2)
curve(d2DSD(x, "lower", -Inf, Inf, tau=1, a=2, v=0.4, sz=0.2, sv=0.9), col="red", lty=2, add=TRUE)
curve(d2DSD(x, "upper", -Inf, Inf, tau=1, a=2, v=0.4),add=TRUE)
curve(d2DSD(x, "lower", -Inf, Inf, tau=1, a=2, v=0.4), col="red", add=TRUE)
# Generate a random sample
dfu <- r2DSD(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1)
# Same RT distribution but upper and lower responses changed
dfl <- r2DSD(50, a=2,v=-0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1)
head(dfu)

d2DSD(dfu, th1=-Inf, th2=Inf, a=2, v=.5)[1:5]
# Scaling diffusion parameters leads do same density values
s <- 2
d2DSD(dfu, th1=-Inf, th2=Inf, a=2*s, v=.5*s, s=2)[1:5]
if (requireNamespace("ggplot2", quietly = TRUE)) {
  require(ggplot2)
  ggplot(dfu, aes(x=rt, y=conf))+
    stat_density_2d(aes(fill = after_stat(density)), geom = "raster", contour = FALSE) +
    facet_wrap(~response)
}
boxplot(conf~response, data=dfu)
```

```
# Restricting to specific confidence region
dfu <- dfu[dfu$conf >0 & dfu$conf <1,]
d2DSD(dfu, th1=0, th2=1, a=2, v=0.5)[1:5]

# If lower confidence threshold is higher than the upper, the function throws an error,
# except when stop_on_error is FALSE
d2DSD(dfu[1:5,], th1=1, th2=0, a=2, v=0.5, stop_on_error = FALSE)
```

`dDDConf` *Drift Diffusion Model with time-dependent confidence*

Description

Likelihood function and random number generator for the Drift Diffusion Model with confidence computed as decision time. It includes following parameters: DDM parameters: `a` (threshold separation), `z` (starting point; relative), `v` (drift rate), `t0` (non-decision time/ response time constant), `d` (differences in speed of response execution), `sv` (inter-trial-variability of drift), `st0` (inter-trial-variability of non-decision components), `sz` (inter-trial-variability of relative starting point), `s` (diffusion constant).

Usage

```
dDDConf(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5, d = 0,
  sz = 0, sv = 0, st0 = 1, s = 1, precision = 3,
  z_absolute = FALSE, stop_on_error = TRUE, stop_on_zero = FALSE,
  st0stepsize = 0.001)

rDDConf(n, a, v, t0 = 0, z = 0.5, d = 0, sz = 0, sv = 0, st0 = 2,
  s = 1, delta = 0.01, maxrt = 15, z_absolute = FALSE,
  stop_on_error = TRUE)
```

Arguments

`rt`	a vector of RTs. Or for convenience also a `data.frame` with columns `rt` and `response`.
`response`	character vector, indicating the decision, i.e. which boundary was met first. Possible values are `c("upper", "lower")` (possibly abbreviated) and `"upper"` being the default. Alternatively, a numeric vector with values 1=lower and 2=upper or -1=lower and 1=upper, respectively. For convenience, `response` is converted via `as.numeric` also allowing factors. Ignored if the first argument is a `data.frame`.
`th1`	together with `th2`: scalars or numerical vectors giving the lower and upper bound of the interval, in which the accumulator should end at the time of the confidence judgment (i.e. at time `rt+tau`). Only values with `th2>=th1` are accepted.
`th2`	(see `th1`)

`a`	threshold separation. Amount of information that is considered for a decision. Large values indicate a conservative decisional style. Typical range: 0.5 < `a` < 2
`v`	drift rate. Average slope of the information accumulation process. The drift gives information about the speed and direction of the accumulation of information. Large (absolute) values of drift indicate a good performance. If received information supports the response linked to the upper threshold the sign will be positive and vice versa. Typical range: -5 < `v` < 5
`t0`	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution). Typical range: 0.1 < `t0` < 0.5. Default is 0.
`z`	(by default relative) starting point. Indicator of an a priori bias in decision making. When the relative starting point `z` deviates from `0.5`, the amount of information necessary for a decision differs between response alternatives. Default is `0.5` (i.e., no bias).
`d`	differences in speed of response execution (in seconds). Positive values indicate that response execution is faster for responses linked to the upper threshold than for responses linked to the lower threshold. Typical range: -0.1 < `d` < 0.1. Default is 0.
`sz`	inter-trial-variability of starting point. Range of a uniform distribution with mean `z` describing the distribution of actual starting points from specific trials. Values different from 0 can predict fast errors (but can slow computation considerably). Typical range: 0 < `sz` < 0.2. Default is 0. (Given in relative range i.e. bounded by 2*min(z, 1-z))
`sv`	inter-trial-variability of drift rate. Standard deviation of a normal distribution with mean `v` describing the distribution of actual drift rates from specific trials. Values different from 0 can predict slow errors. Typical range: 0 < `sv` < 2. Default is 0.
`st0`	inter-trial-variability of non-decisional components. Range of a uniform distribution with mean `t0 + st0/2` describing the distribution of actual `t0` values across trials. Accounts for response times below `t0`. Reduces skew of predicted RT distributions. Values different from 0 can slow computation considerably. Typical range: 0 < `st0` < 0.2. Default is 0.
`s`	diffusion constant. Standard deviation of the random noise of the diffusion process (i.e., within-trial variability), scales `a`, `v`, `sv`, and `th`'s. Needs to be fixed to a constant in most applications. Default is 1. Note that the default used by Ratcliff and in other applications is often 0.1.
`precision`	`numerical` scalar value. Precision of calculation. Corresponds to the stepsize of integration w.r.t. `z`. Default is 1e-5.
`z_absolute`	logical. Determines whether `z` is treated as absolute start point (`TRUE`) or relative (`FALSE`; default) to `a`.
`stop_on_error`	Should the diffusion functions return 0 if the parameters values are outside the allowed range (= `FALSE`) or produce an error in this case (= `TRUE`).
`stop_on_zero`	Should the computation of densities stop as soon as a density value of 0 occurs. This may save a lot of time if the function is used for a likelihood function. Default: `FALSE`

`st0stepsize`	numerical scalar value. Stepsize for integration over `t0`.
`n`	integer. The number of samples generated.
`delta`	numeric. Discretization step size for simulations in the stochastic process
`maxrt`	numeric. Maximum decision time returned. If the simulation of the stochastic process exceeds a decision time of `maxrt`, the `response` will be set to 0 and the `maxrt` will be returned as `rt`.

Details

For the confidence part: `th1` and `th2` (lower and upper thresholds for decision time interval).

Note that the parameterization or defaults of non-decision time variability `st0` and diffusion constant `s` differ from what is often found in the literature.

The Ratcliff diffusion model (Ratcliff and McKoon, 2008) is a mathematical model for two-choice discrimination tasks. It is based on the assumption that information is accumulated continuously until one of two decision thresholds is hit. For introduction see Ratcliff and McKoon (2008).

This model incorporates the idea, that the decision time T is informative for stimulus difficulty and thus confidence is computed as a monotone function of $\frac{1}{\sqrt{T}}$. In this implementation, confidence is the decision time, directly. Here, we use an interval, given by `th1` and `th2`, assuming that the data is given with discrete judgments and pre-processed, s.t. these discrete ratings are translated to the respective intervals.

All functions are fully vectorized across all parameters as well as the response to match the length or `rt` (i.e., the output is always of length equal to `rt`). This allows for trial wise parameters for each model parameter.

For convenience, the function allows that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt` and `response`). Other columns (as well as passing `response` separately argument) will be ignored.

Value

`dDDConf` gives the density/likelihood/probability of the diffusion process producing a decision of `response` at time `rt` and a confidence judgment corresponding to the interval [`th1`, `th2`]. The value will be a numeric vector of the same length as `rt`.

`rDDConf` returns a `data.frame` with three columns and `n` rows. Column names are `rt` (response time), `response` (-1 (lower) or 1 (upper), indicating which bound was hit), `conf` for the decision time (without non-decision time component; not discretized!).

The distribution parameters (as well as `response`, `th1` and `th2`) are recycled to the length of the result. In other words, the functions are completely vectorized for all parameters and even the response boundary.

Note

The parameterization of the non-decisional components, `t0` and `st0`, differs from the parameterization sometimes used in the literature. In the present case `t0` is the lower bound of the uniform distribution of length `st0`, but *not* its midpoint. The parameterization employed here is in line with the functions in the `rtdists` package.

The default diffusion constant `s` is 1 and not 0.1 as in most applications of Roger Ratcliff and others. Usually `s` is not specified as the other parameters: `a`, `v`, and `sv`, may be scaled to produce the same distributions (as is done in the code).

The function code is basically an extension of the `ddiffusion` function from the package `rtdists` for the Ratcliff diffusion model.

Author(s)

For the original `rtdists` package: Underlying C code by Jochen Voss and Andreas Voss. Porting and R wrapping by Matthew Gretton, Andrew Heathcote, Scott Brown, and Henrik Singmann. `qdiffusion` by Henrik Singmann. For the `dDDConf` function the C code was extended by Sebastian Hellmann.

References

Ratcliff, R., & McKoon, G. (2008). The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Computation*, 20(4), 873-922.

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# Plot rt distribution ignoring confidence
curve(dDDConf(x, "upper", 0, Inf, a=2, v=0.4, sz=0.2, sv=0.9), xlim=c(0, 2), lty=2, n=40)
curve(dDDConf(x, "lower", 0, Inf, a=2, v=0.4, sz=0.2, sv=0.9), col="red", lty=2, add=TRUE, n=40)
curve(dDDConf(x, "upper", 0, Inf, a=2, v=0.4),add=TRUE, n=40)
curve(dDDConf(x, "lower", 0, Inf, a=2, v=0.4), col="red", add=TRUE, n=40)
# Generate a random sample
dfu <- rDDConf(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=1, s=1)
# Same RT distribution but upper and lower responses changed
dfl <- rDDConf(50, a=2,v=-0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=1, s=1)
head(dfu)

dDDConf(dfu, th1=0.5, th2=2.5, a=2, v=.5, st0=1)[1:5]
# Scaling diffusion parameters leads do same density values
s <- 2
dDDConf(dfu, th1=0.5, th2=2.5, a=2*s, v=.5*s, s=2, st0=1)[1:5]
if (requireNamespace("ggplot2", quietly = TRUE)) {
  require(ggplot2)
  ggplot(dfu, aes(x=rt, y=conf))+
    stat_density_2d(aes(fill = after_stat(density)), geom = "raster", contour = FALSE) +
    facet_wrap(~response)
}
boxplot(conf~response, data=dfu)

# Restricting to specific confidence region
dfu <- dfu[dfu$conf >0 & dfu$conf <1,]
dDDConf(dfu, th1=0, th2=1, a=2, v=0.5, st0=1)[1:5]

# If lower confidence threshold is higher than the upper, the function throws an error,
```

```
# except when stop_on_error is FALSE
dDDConf(dfu[1:5,], th1=1, th2=0, a=2, v=0.5, stop_on_error = FALSE)
```

`dynaViTE` *Dynamical visibility, time, and evidence model (dynaViTE) and Dynamical weighted evidence and visibility model (dynWEV)*

Description

Likelihood function and random number generator for the dynaViTE and dynWEV model (Hellmann et al., 2023). It includes following parameters from the drift diffusion model: `a` (threshold separation), `z` (starting point; relative), `v` (drift rate), `t0` (non-decision time/response time constant), `d` (differences in speed of response execution), `sv` (inter-trial-variability of drift), `st0` (inter-trial-variability of non-decisional components), `sz` (inter-trial-variability of relative starting point) and `s` (diffusion constant). For the computation of confidence following parameters were added: `tau` (post-decisional accumulation time), `w` (weight on the decision evidence (weight on visibility is (1-w))), `muvis` (mean drift rate of visibility process), `svis` (diffusion constant of visibility process), `sigvis` (variability in drift rate of visibility accumulator), `th1` and `th2` (lower and upper thresholds for confidence interval). `lambda` for dynaViTE only, the exponent of judgment time for the division by judgment time in the confidence measure, and **Note that the parametrization or defaults of non-decision time variability `st0` and diffusion constant `s` differ from what is often found in the literature.**

Likelihood function and random number generator for the dynaViTE and dynWEV model (Hellmann et al., 2023). It includes following parameters from the drift diffusion model: `a` (threshold separation), `z` (starting point; relative), `v` (drift rate), `t0` (non-decision time/response time constant), `d` (differences in speed of response execution), `sv` (inter-trial-variability of drift), `st0` (inter-trial-variability of non-decisional components), `sz` (inter-trial-variability of relative starting point) and `s` (diffusion constant). For the computation of confidence following parameters were added: `tau` (post-decisional accumulation time), `w` (weight on the decision evidence (weight on visibility is (1-w))), `muvis` (mean drift rate of visibility process), `svis` (diffusion constant of visibility process), `sigvis` (variability in drift rate of visibility accumulator), `th1` and `th2` (lower and upper thresholds for confidence interval). `lambda` for dynaViTE only, the exponent of judgment time for the division by judgment time in the confidence measure, and **Note that the parametrization or defaults of non-decision time variability `st0` and diffusion constant `s` differ from what is often found in the literature.**

Usage

```
dWEV(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5, d = 0,
  sz = 0, sv = 0, st0 = 0, tau = 1, w = 0.5, muvis = NULL,
  sigvis = 0, svis = 1, lambda = 0, s = 1, simult_conf = FALSE,
  precision = 6, z_absolute = FALSE, stop_on_error = TRUE,
  stop_on_zero = FALSE)

ddynaViTE(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5,
  d = 0, sz = 0, sv = 0, st0 = 0, tau = 1, w = 0.5, muvis = NULL,
```

```
  sigvis = 0, svis = 1, lambda = 0, s = 1, simult_conf = FALSE,
  precision = 6, z_absolute = FALSE, stop_on_error = TRUE,
  stop_on_zero = FALSE)

rdynaViTE(n, a, v, t0 = 0, z = 0.5, d = 0, sz = 0, sv = 0, st0 = 0,
  tau = 1, w = 0.5, muvis = NULL, sigvis = 0, svis = 1, lambda = 0,
  s = 1, delta = 0.01, maxrt = 15, simult_conf = FALSE,
  z_absolute = FALSE, stop_on_error = TRUE, process_results = FALSE)

dWEV(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5, d = 0,
  sz = 0, sv = 0, st0 = 0, tau = 1, w = 0.5, muvis = NULL,
  sigvis = 0, svis = 1, lambda = 0, s = 1, simult_conf = FALSE,
  precision = 6, z_absolute = FALSE, stop_on_error = TRUE,
  stop_on_zero = FALSE)

rWEV(n, a, v, t0 = 0, z = 0.5, d = 0, sz = 0, sv = 0, st0 = 0,
  tau = 1, w = 0.5, muvis = NULL, sigvis = 0, svis = 1, lambda = 0,
  s = 1, delta = 0.01, maxrt = 15, simult_conf = FALSE,
  z_absolute = FALSE, stop_on_error = TRUE, process_results = FALSE)

ddynaViTE(rt, response = "upper", th1, th2, a, v, t0 = 0, z = 0.5,
  d = 0, sz = 0, sv = 0, st0 = 0, tau = 1, w = 0.5, muvis = NULL,
  sigvis = 0, svis = 1, lambda = 0, s = 1, simult_conf = FALSE,
  precision = 6, z_absolute = FALSE, stop_on_error = TRUE,
  stop_on_zero = FALSE)

rdynaViTE(n, a, v, t0 = 0, z = 0.5, d = 0, sz = 0, sv = 0, st0 = 0,
  tau = 1, w = 0.5, muvis = NULL, sigvis = 0, svis = 1, lambda = 0,
  s = 1, delta = 0.01, maxrt = 15, simult_conf = FALSE,
  z_absolute = FALSE, stop_on_error = TRUE, process_results = FALSE)
```

Arguments

`rt`	a vector of RTs. Or for convenience also a `data.frame` with columns `rt` and `response`.
`response`	character vector, indicating the decision, i.e. which boundary was met first. Possible values are `c("upper", "lower")` (possibly abbreviated) and `"upper"` being the default. Alternatively, a numeric vector with values 1=lower and 2=upper or -1=lower and 1=upper, respectively. For convenience, `response` is converted via `as.numeric` also allowing factors. Ignored if the first argument is a `data.frame`.
`th1`	together with `th2`: scalars or numerical vectors giving the lower and upper bound of the interval of the confidence measure (see Details). Only values with `th2>=th1` are accepted.
`th2`	(see `th1`)
`a`	threshold separation. Amount of information that is considered for a decision. Large values indicate a conservative decisional style. Typical range: 0.5 < `a` < 2

`v`	drift rate of decision process. Average slope of the information accumulation process. The drift gives information about the speed and direction of the accumulation of information. Large (absolute) values of drift indicate a good performance. If received information supports the response linked to the upper threshold the sign will be positive and vice versa. Typical range: -5 < `v` < 5
`t0`	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution). Typical range: 0.1 < `t0` < 0.5. Default is 0.
`z`	(by default relative) starting point of decision process. Indicator of an a priori bias in decision making. When the relative starting point `z` deviates from `0.5`, the amount of information necessary for a decision differs between response alternatives. Default is `0.5` (i.e., no bias).
`d`	differences in speed of response execution (in seconds). Positive values indicate that response execution is faster for responses linked to the upper threshold than for responses linked to the lower threshold. Typical range: -0.1 < `d` < 0.1. Default is 0.
`sz`	inter-trial-variability of starting point. Range of a uniform distribution with mean `z` describing the distribution of actual starting points from specific trials. Values different from 0 can predict fast errors (but can slow computation considerably). Typical range: 0 < `sz` < 0.2. Default is 0. (Given in relative range i.e. bounded by 2*min(z, 1-z))
`sv`	inter-trial-variability of drift rate of decision process. Standard deviation of a normal distribution with mean `v` describing the distribution of actual drift rates from specific trials. Values different from 0 can predict slow errors. Typical range: 0 < `sv` < 2. Default is 0.
`st0`	inter-trial-variability of non-decisional components. Range of a uniform distribution with mean `t0 + st0/2` describing the distribution of actual `t0` values across trials. Accounts for response times below `t0`. Reduces skew of predicted RT distributions. Values different from 0 can slow computation considerably. Typical range: 0 < `st0` < 0.2. Default is 0.
`tau`	post-decisional accumulation time; the length of the time period after the decision was made until the confidence judgment is made. Range: `tau`>0. Default: `tau`=1.
`w`	weight put on the final state of the decision accumulator for confidence computation. 1-w is the weight on the visibility accumulator. Range: 0<`w`<1. Default: `w`=0.5.
`muvis`	mean drift of visibility process. If `NULL` (default), `muvis` will be set to the absolute value of `v`.
`sigvis`	the variability in drift rate of the visibility process (which varies independently from the drift rate in decision process). Range: `sigvis`>=0. Default: `sigvis`=0.
`svis`	diffusion constant of visibility process. Range: `svis`>0. Default: `svis`=1.
`lambda`	power for judgment time in the division of the confidence measure by the judgment time (Default: 0, i.e. no division which is the version of dynWEV proposed by Hellmann et al., 2023)

`s`	diffusion constant of decision process; standard deviation of the random noise of the diffusion process (i.e., within-trial variability), scales other parameters (see Note). Needs to be fixed to a constant in most applications. Default is 1. Note that the default used by Ratcliff and in other applications is often 0.1.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision. If that is the case decision and confidence judgment are assumed to have happened subsequent before the response. Therefore `tau` is included in the response time. If the decision was reported before the confidence report, `simul_conf` should be `FALSE`.
`precision`	numerical scalar value. Precision of calculation. Determines the step size of integration w.r.t. `z` and `t0`. Represents the number of decimals precisely computed on average. Default is 6.
`z_absolute`	logical. Determines whether `z` is treated as absolute start point (`TRUE`) or relative (`FALSE`; default) to `a`.
`stop_on_error`	Should the diffusion functions return 0 if the parameters values are outside the allowed range (= `FALSE`) or produce an error in this case (= `TRUE`).
`stop_on_zero`	Should the computation of densities stop as soon as a density value of 0 occurs. This may save a lot of time if the function is used for a likelihood function. Default: FALSE
`n`	integer. The number of samples generated.
`delta`	numeric. Discretization step size for simulations in the stochastic process
`maxrt`	numeric. Maximum decision time returned. If the simulation of the stochastic process exceeds a decision time of `maxrt`, the `response` will be set to 0 and the `maxrt` will be returned as `rt`.
`process_results`	logical. Whether the output simulations should contain the final state of the decision (and visibility) process as additional column. Default is FALSE, meaning that no additional columns for the final process states are returned.

Details

The function dWEV was renamed to ddynaViTE in version 0.1.0 of the package. It is still here for reasons of backwards compatibility. The function just calls the ddynaViTE function (and produces a deprecation warning).

The dynamical visibility, time, and evidence (dynaViTE) model and the weighted evidence and visibility model (dynWEV) are extensions of the 2DSD model for decision confidence (see `d2DSD`). It assumes that the decision follows a drift diffusion model with two additional assumptions to account for confidence. First, there is a post-decisional period of further evidence accumulation `tau`. Second, another accumulation process accrues information about stimulus reliability (the visibility process) including also evidence about decision irrelevant features. See Hellmann et al. (2023) for more information. The measure for confidence is then a weighted sum of the final state of the decision process X and the visibility process V over a power-function of total accumulation time, i.e. for a decision time T (which is not the response time), the confidence variable is

$$conf = \frac{wX(T+\tau) + (1-w)V(T+\tau)}{(T+\tau)^{\lambda}}.$$

The dynWEV model is a special case of dynaViTE, with the parameter `lambda=0`.

All functions are fully vectorized across all parameters as well as the response to match the length or `rt` (i.e., the output is always of length equal to `rt`). This allows for trial wise parameters for each model parameter.

For convenience, the function allows that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt` and `response`). Other columns (as well as passing `response` separately argument) will be ignored.

The functions dWEV and rWEV were renamed to ddynaViTE and rdynaViTE, respectively in version 0.1.0 of the package. They are still here for reasons of backwards compatibility. The functions just calls their counterpar ddynaViTE and rdynaViTE (and produce a deprecation warning).

The dynamical visibility, time, and evidence (dynaViTE) model and the weighted evidence and visibility model (dynWEV) are extensions of the 2DSD model for decision confidence (see `d2DSD`). It assumes that the decision follows a drift diffusion model with two additional assumptions to account for confidence. First, there is a post-decisional period of further evidence accumulation `tau`. Second, another accumulation process accrues information about stimulus reliability (the visibility process) including also evidence about decision irrelevant features. See Hellmann et al. (2023) for more information. The measure for confidence is then a weighted sum of the final state of the decision process X and the visibility process V over a power-function of total accumulation time, i.e. for a decision time T (which is not the response time), the confidence variable is

$$conf = \frac{wX(T+\tau) + (1-w)V(T+\tau)}{(T+\tau)^{\lambda}}.$$

The dynWEV model is a special case of dynaViTE, with the parameter `lambda=0`.

All functions are fully vectorized across all parameters as well as the response to match the length or `rt` (i.e., the output is always of length equal to `rt`). This allows for trial wise parameters for each model parameter.

For convenience, the function allows that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt` and `response`). Other columns (as well as passing `response` separately argument) will be ignored.

Value

`ddynaViTE` gives the density/likelihood/probability of the diffusion process producing a decision of `response` at time `rt` and a confidence judgment corresponding to the interval [`th1`, `th2`]. The value will be a numeric vector of the same length as `rt`.

`rdynaViTE` returns a `data.frame` with three columns and `n` rows. Column names are `rt` (response time), `response` (-1 (lower) or 1 (upper), indicating which bound was hit), and `conf` (the value of the confidence measure; not discretized!).

The distribution parameters (as well as `response`, `tau`, `th1` and `th2`, `w` and `sig`) are recycled to the length of the result. In other words, the functions are completely vectorized for all parameters and even the response boundary.

`ddynaViTE` gives the density/likelihood/probability of the diffusion process producing a decision of `response` at time `rt` and a confidence judgment corresponding to the interval [`th1`, `th2`]. The value will be a numeric vector of the same length as `rt`.

`rdynaViTE` returns a `data.frame` with three columns and `n` rows. Column names are `rt` (response time), `response` (-1 (lower) or 1 (upper), indicating which bound was hit), and `conf` (the value of the confidence measure; not discretized!).

The distribution parameters (as well as `response`, `tau`, `th1` and `th2`, `w` and `sig`) are recycled to the length of the result. In other words, the functions are completely vectorized for all parameters and even the response boundary.

Note

The parameterization of the non-decisional components, `t0` and `st0`, differs from the parameterization sometimes used in the literature. In the present case `t0` is the lower bound of the uniform distribution of length `st0`, but *not* its midpoint. The parameterization employed here is in line with the functions in the `rtdists` package.

The default diffusion constant `s` is 1 and not 0.1 as in most applications of Roger Ratcliff and others. Usually `s` is not specified as the other parameters: `a`, `v`, `sv`, `muvis`, `sigvis`, and `svis` respectively, may be scaled to produce the same distributions (as is done in the code).

The function code is basically an extension of the `ddiffusion` function from the package `rtdists` for the Ratcliff diffusion model.

The parameterization of the non-decisional components, `t0` and `st0`, differs from the parameterization sometimes used in the literature. In the present case `t0` is the lower bound of the uniform distribution of length `st0`, but *not* its midpoint. The parameterization employed here is in line with the functions in the `rtdists` package.

The default diffusion constant `s` is 1 and not 0.1 as in most applications of Roger Ratcliff and others. Usually `s` is not specified as the other parameters: `a`, `v`, `sv`, `muvis`, `sigvis`, and `svis` respectively, may be scaled to produce the same distributions (as is done in the code).

The function code is basically an extension of the `ddiffusion` function from the package `rtdists` for the Ratcliff diffusion model.

Author(s)

Sebastian Hellmann

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# Plot rt distribution ignoring confidence
curve(ddynaViTE(x, "upper", -Inf, Inf, tau=1, a=2, v=0.4, sz=0.2, sv=0.9), xlim=c(0, 2), lty=2)
curve(ddynaViTE(x, "lower", -Inf, Inf,tau=1, a=2, v=0.4, sz=0.2, sv=0.9))
curve(ddynaViTE(x, "upper", -Inf, Inf,  tau=1, a=2, v=0.4),add=TRUE)
curve(ddynaViTE(x, "lower", -Inf, Inf, tau=1, a=2, v=0.4), col="red", add=TRUE)
```

```
# Generate a random sample
df1 <- rdynaViTE(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1, w=0.9)
# Same RT and response distribution but different confidence distribution
df2 <- rdynaViTE(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1, w=0.1)
head(df1)

# Scaling diffusion parameters leads do same density values
ddynaViTE(df1[1:5,], th1=-Inf, th2=Inf, a=2, v=.5)[1:5]
s <- 2
ddynaViTE(df1[1:5,], th1=-Inf, th2=Inf, a=2*s, v=.5*s, s=2)[1:5]

# Diffusion constant also scales confidence parameters
ddynaViTE(df1[1:5,], th1=0.2, th2=1, a=2, v=.5, sv=0.2, w=0.5, sigvis = 0.2, svis = 1)[1:5]
s <- 2
ddynaViTE(df1[1:5,], th1=0.2*s, th2=1*s, a=2*s, v=.5*s, s=2,
     sv=0.2*s, w=0.5, sigvis=0.2*s, svis=1*s)[1:5]


two_samples <- rbind(cbind(df1, w="high"),
                     cbind(df2, w="low"))
# no difference in RT distributions
boxplot(rt~w+response, data=two_samples)
# but different confidence distributions
boxplot(conf~w+response, data=two_samples)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  require(ggplot2)
  ggplot(two_samples, aes(x=rt, y=conf))+
    stat_density_2d(aes(fill = after_stat(density)), geom = "raster", contour = FALSE) +
    xlim(c(0, 2))+ ylim(c(-1.5, 4))+
    facet_grid(cols=vars(w), rows=vars(response), labeller = "label_both")
}

# Restricting to specific confidence region
df1 <- df1[df1$conf >0 & df1$conf <1,]
ddynaViTE(df1[1:5,], th1=0, th2=1, a=2, v=0.5)[1:5]

# If lower confidence threshold is higher than the upper, the function throws an error,
# except when stop_on_error is FALSE
ddynaViTE(df1[1:5,], th1=1, th2=0, a=2, v=0.5, stop_on_error = FALSE)

# Plot rt distribution ignoring confidence
curve(ddynaViTE(x, "upper", -Inf, Inf, tau=1, a=2, v=0.4,
                    sz=0.2, sv=0.9),
      xlim=c(0, 2), lty=2)
curve(ddynaViTE(x, "lower", -Inf, Inf,tau=1, a=2, v=0.4,
                    sz=0.2, sv=0.9),
     col="red", lty=2, add=TRUE)
curve(ddynaViTE(x, "upper", -Inf, Inf,  tau=1, a=2, v=0.4), add=TRUE)
curve(ddynaViTE(x, "lower", -Inf, Inf, tau=1, a=2, v=0.4), add=TRUE)
# Generate a random sample
df1 <- rdynaViTE(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1, w=0.9)
# Same RT and response distribution but different confidence distribution
df2 <- rdynaViTE(5000, a=2,v=0.5,t0=0,z=0.5,d=0,sz=0,sv=0, st0=0,  tau=1, s=1, w=0.1)
```

```
head(df1)

# Scaling diffusion parameters leads do same density values
ddynaViTE(df1[1:5,], th1=-Inf, th2=Inf, a=2, v=.5)[1:5]
s <- 2
ddynaViTE(df1[1:5,], th1=-Inf, th2=Inf, a=2*s, v=.5*s, s=2)[1:5]

# Diffusion constant also scales confidence parameters
ddynaViTE(df1[1:5,], th1=0.2, th2=1, a=2, v=.5, sv=0.2, w=0.5, sigvis = 0.2, svis = 1)[1:5]
s <- 2
ddynaViTE(df1[1:5,], th1=0.2*s, th2=1*s, a=2*s, v=.5*s, s=2,
     sv=0.2*s, w=0.5, sigvis=0.2*s, svis=1*s)[1:5]


two_samples <- rbind(cbind(df1, w="high"),
                     cbind(df2, w="low"))
# no difference in RT distributions
boxplot(rt~w+response, data=two_samples)
# but different confidence distributions
boxplot(conf~w+response, data=two_samples)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  require(ggplot2)
  ggplot(two_samples, aes(x=rt, y=conf))+
    stat_density_2d(aes(fill = after_stat(density)), geom = "raster", contour = FALSE) +
    xlim(c(0, 2))+ ylim(c(-1.5, 4))+
    facet_grid(cols=vars(w), rows=vars(response), labeller = "label_both")
}

# Restricting to specific confidence region
df1 <- df1[df1$conf >0 & df1$conf <1,]
ddynaViTE(df1[1:5,], th1=0, th2=1, a=2, v=0.5)[1:5]

# If lower confidence threshold is higher than the upper, the function throws an error,
# except when stop_on_error is FALSE
ddynaViTE(df1[1:5,], th1=1, th2=0, a=2, v=0.5, stop_on_error = FALSE)
```

`fitRTConf` *Function for fitting sequential sampling confidence models*

Description

Fits the parameters of different models of response time and confidence, including the 2DSD model (Pleskac & Busemeyer, 2010), dynWEV, DDConf, and various flavors of race models (Hellmann et al., 2023). Which model to fit is specified by the argument `model`. Only a ML method is implemented. See `ddynaViTE`, `d2DSD`, and `dRM` for more information about the parameters and Details for not-fitted parameters.

Usage

```
fitRTConf(data, model = "dynWEV", fixed = list(sym_thetas = FALSE),
  init_grid = NULL, grid_search = TRUE, data_names = list(),
  nRatings = NULL, restr_tau = Inf, precision = 3, logging = FALSE,
  opts = list(), optim_method = "bobyqa", useparallel = FALSE,
  n.cores = NULL, ...)
```

Arguments

`data`	a `data.frame` where each row is one trial, containing following variables (column names can be changed by passing additional arguments of the form `condition="contrast"`):
	• `condition` (not necessary; for different levels of stimulus quality, will be transformed to a factor),
	• `rating` (discrete confidence judgments, should be given as integer vector; otherwise will be transformed to integer),
	• `rt` (giving the reaction times for the decision task),
	• either 2 of the following (see details for more information about the accepted formats):
	  – `stimulus` (encoding the stimulus category in a binary choice task),
	  – `response` (encoding the decision response),
	  – `correct` (encoding whether the decision was correct; values in 0, 1)
	• `sbj` or `participant` (optional; giving the subject ID; only relevant if `logging = TRUE`; if unique the ID is used in saved files with interim results and logging messages; if non-unique or missing and `logging =TRUE`, 999 will be used then)

`model`	character scalar. One of "dynWEV", "2DSD", "IRM", "PCRM", "IRMt", "PCRMt", or "DDConf" for the model to be fit.
`fixed`	list. List with parameter-value pairs for parameters that should not be fitted. See Details.
`init_grid`	data.frame or `NULL`. Grid for the initial parameter search. Each row is one parameter constellation. See details for more information. If `NULL` a default grid will be used.
`grid_search`	logical. If `FALSE`, the grid search before the optimization algorithm is omitted. The fitting is then started with a mean parameter set from the default grid (if `init_grid=NULL`) or directly with the rows from `init_grid`, if not `NULL`. (Default: `TRUE`)
`data_names`	named list (e.g. `c(rating="confidence")`). Alternative possibility of giving other column names for the variables in the data. By default column names are identical to the ones given in the data argument description.
`nRatings`	integer. Number of rating categories. If `NULL`, the maximum of `rating` and `length(unique(rating))` is used. This argument is especially important for data sets where not the whole range of rating categories is realized. If given, ratings has to be given as factor or integer.
`restr_tau`	numerical or `Inf` or `"simult_conf"`. For 2DSD and dynWEV only. Upper bound for tau. Fits will be in the interval (0,`restr_tau`). If FALSE tau will be unbound. For `"simult_conf"`, see the documentation of `d2DSD` and `ddynaViTE`

`precision`	numeric. Precision of calculation for the density functions (see `ddynaViTE` and `dPCRM` for more information).
`logging`	logical. If `TRUE`, a folder 'autosave/fit**model**' is created and messages about the process are printed in a logging file and to console (depending on OS). Additionally intermediate results are saved in a `.RData` file with the participant ID in the name.
`opts`	list. A list for more control options in the optimization routines (depending on the `optim_method`). See details for more information.
`optim_method`	character. Determines which optimization function is used for the parameter estimation. Either `"bobyqa"` (default), `"L-BFGS-B"` or `"Nelder-Mead"`. `"bobyqa"` uses a box-constrained optimization with quadratic interpolation. (See `bobyqa` for more information.) The first two use a box-constraint optimization. For Nelder-Mead a transfinite function rescaling is used (i.e. the constrained arguments are suitably transformed to the whole real line).
`useparallel`	logical. If `TRUE` the grid search in the beginning is done with a parallel back-end, using the `parallel` package.
`n.cores`	integer or `NULL`. Number of cores used for parallelization. If `NULL` (default) the number of available cores -1 is used.
`...`	Possibility of giving alternative variable names in data frame (in the form `condition = "SOA"`, or `response="pressedKey"`).

Details

The fitting involves a first grid search through computation of the likelihood on an initial grid with possible sets of parameters to start the optimization routine. Then the best `nAttempts` parameter sets are chosen for an optimization, which is done with an algorithm, depending on the argument `optim-method`. The Nelder-Mead algorithm uses the R function `optim`. The optimization routine is restarted `nRestarts` times with the starting parameter set equal to the best parameters from the previous routine.

stimulus, response and correct. Two of these columns must be given in data. If all three are given, correct will have no effect (and will be not checked!). stimulus can always be given in numerical format with values -1 and 1. response can always be given as a character vector with "lower" and "upper" as values. Correct must always be given as a 0-1-vector. If the stimulus column is given together with a response column and they both do not match the above format, they need to have the same values/levels (if `factor`). In the case that only stimulus/response is given in any other format together with correct, the unique values will be sorted increasingly and the first value will be encoded as "lower"/-1 and the second as "upper"/+1.

fixed. Parameters that should not be fitted but kept constant. These will be dropped from the initial grid search but will be present in the output, to keep all parameters for prediction in the result. Includes the possibility for symmetric confidence thresholds for both alternative (`sym_thetas`=logical). Other examples are `z =.5`, `sv=0`, `st0=0`, `sz=0`. For race models, the possibility of setting `a='b'` (or vice versa) leads to identical upper bounds on the decision processes, which is the equivalence for `z=.5` in a diffusion process.

Parameters not fitted. The models get developed continuously and not all changes are adopted in the fitting function instantly. Following parameters are currently not included in the fitting routine:

- in race models: `sza`, `szb`, `smu1`, and `smu2` `init_grid`. Each row should be one parameter set to check. The column names should include the parameters of the desired model, which are the following for 2DSD: `a`, `vmin` and `vmax` (will be equidistantly spanned across conditions), `sv`, `z` (as the relative starting point between 0 and `a`), `sz` (also in relative terms), `t0`, `st0`, `theta0` (minimal threshold), `thetamax` (maximal threshold; the others will be equidistantly spanned symmetrically for both decisions), and `tau`. For dynWEV, additionally `w` , `svis`, and `sigvis` are required. For the race models the parameters are: `vmin`, `vmax` (will be equidistantly spanned across conditions), `a` and `b` (decision thresholds), `t0`, `st0`, `theta0` (minimal threshold), `thetamax` (maximal threshold; the others will be equidistantly spanned symmetrically for both decisions), and for time-dependent confidence race models additionally `wrt` and `wint` (as weights compared to `wx=1`).

opts. A list with numerical values. Possible options are listed below (together with the optimization method they are used for).

- `nAttempts` (all) number of best performing initial parameter sets used for optimization; default 5, if `grid_search` is `TRUE`. If `grid_search` is `FALSE` and `init_grid` is `NULL`, then `nAttempts` will be set to 1 (and any input will be ignored). If `grid_search` is `FALSE` and `init_grid` is not `NULL`, the rows of `init_grid` will be used from top to bottom (since no initial grid search is done) with not more than `nAttempts` rows used.
- `nRestarts` (all) number of successive `optim` routines for each of the starting parameter sets; default 5,
- `maxfun` (`'bobyqa'`) maximum number of function evaluations; default: 5000,
- `maxit` (`'Nelder-Mead' and 'L-BFGS-B'`) maximum iterations; default: 2000,
- `reltol` (`'Nelder-Mead'`) relative tolerance; default: 1e-6),
- `factr` (`'L-BFGS-B'`) tolerance in terms of reduction factor of the objective, default: 1e-10)

Value

Gives a one-row data frame with columns for the different parameters as fitted result as well as additional information about the fit (`negLogLik` (for final parameters), `k` (number of parameters), `N` (number of data rows), `BIC`, `AICc` and `AIC`) and the column `fixed`, which includes all information about fixed and not fitted parameters.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

https://nashjc.wordpress.com/2016/11/10/why-optim-is-out-of-date/

https://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf

Examples

```
# We use one of the implemented models, "dynWEV"
# 1. Generate data
# data with positive drift (stimulus = "upper")
data <- rdynaViTE(20, a=2,v=0.5,t0=0.2,z=0.5, sz=0.1,sv=0.1, st0=0,  tau=4, s=1, w=0.3)
data$stimulus <- "upper"
# data with negtive drift (stimulus = "lower") but same intensity
data2 <- rdynaViTE(100, a=2,v=-0.5,t0=0.2,z=0.5,sz=0.1,sv=0.1, st0=0,  tau=4, s=1, w=0.3)
data2$stimulus <- "lower"
data <- rbind(data, data2)
# Transfer response column and add dummy condition column
data$response <- ifelse(data$response==1, "upper", "lower")
data$condition <- 1
# Take some confidence thresholds for discrete ratings
threshs <- c(-Inf, 1, 2, Inf)
data$rating <- as.numeric(cut(data$conf, breaks = threshs, include.lowest = TRUE))
head(data)

# 2. Use fitting function
# Fitting the model with these opts results in a pretty bad fit
# (especially because of omitting the grid_search)

   fitRTConf(data, "dynWEV", fixed=list(sym_thetas=TRUE, z=0.5, st0=0),
            grid_search = FALSE, logging=FALSE,
            opts = list(nAttempts=1, nRestarts=2, maxfun=2000))
```

`fitRTConfModels` *Function for fitting several sequential sampling confidence models in parallel*

Description

This function is a wrapper of the function `fitConfModel` (see there for more information). It calls the function for every possible combination of model and participant/subject in `model` and `data` respectively. Also, see `ddynaViTE`, `d2DSD`, `dDDConf`, and `dRM` for more information about the parameters.

Usage

```
fitRTConfModels(data, models = c("dynaViTE", "2DSD", "PCRMt"),
  nRatings = NULL, fixed = list(sym_thetas = FALSE), restr_tau = Inf,
  grid_search = TRUE, opts = list(), optim_method = "bobyqa",
  logging = FALSE, precision = 3, parallel = TRUE, n.cores = NULL, ...)
```

Arguments

`data`	a `data.frame` where each row is one trial, containing following variables (column names can be changed by passing additional arguments of the form `condition="contrast"`):
	• `condition` (not necessary; for different levels of stimulus quality, will be transformed to a factor),
	• `rating` (discrete confidence judgments, should be given as integer vector; otherwise will be transformed to integer),
	• `rt` (giving the reaction times for the decision task),
	• either 2 of the following (see details for more information about the accepted formats):
	  – `stimulus` (encoding the stimulus category in a binary choice task),
	  – `response` (encoding the decision response),
	  – `correct` (encoding whether the decision was correct; values in 0, 1)
	• `sbj` alternatively `subject` or `participant` (giving the subject ID; the models given in the second argument are fitted for each subject individually. (Furthermore, if `logging = TRUE`, the ID is used in files saved with interim results and logging messages.) The output data frame reused the name of the column in the input (i.e. the output contains a `subject` column, if the input contains `subject` instead of `sbj`).)

`models`	character vector with following possible elements "dynWEV", "2DSD", "IRM", "PCRM", "IRMt", and "PCRMt" for the models to be fit.
`nRatings`	integer. Number of rating categories. If `NULL`, the maximum of `rating` and `length(unique(rating))` is used. This argument is especially important for data sets where not the whole range of rating categories is realized. If given, ratings has to be given as factor or integer.
`fixed`	list. List with parameter value pairs for parameters that should not be fitted. (see Details).
`restr_tau`	numerical or `Inf` or `"simult_conf"`. Used for 2DSD and dynWEV only. Upper bound for tau. Fits will be in the interval (0,`restr_tau`). If `FALSE` tau will be unbound. For `"simult_conf"`, see the documentation of `d2DSD` and `ddynaViTE`
`grid_search`	logical. If `FALSE`, the grid search before the optimization algorithm is omitted. The fitting is then started with a mean parameter set from the default grid. (Default: `TRUE`)
`opts`	list. A list for more control options in the optimization routines (depending on the `optim_method`). See details for more information.
`optim_method`	character. Determines which optimization function is used for the parameter estimation. Either `"bobyqa"` (default), `"L-BFGS-B"` or `"Nelder-Mead"`. `"bobyqa"` uses a box-constrained optimization with quadratic interpolation. (See `bobyqa` for more information.) The first two use a box-constraint optimization. For Nelder-Mead a transfinite function rescaling is used (i.e. the constrained arguments are suitably transformed to the whole real line).
`logging`	logical. If `TRUE`, a folder 'autosave/fit**model**' is created and messages about the process are printed in a logging file and to console (depending on OS). Additionally intermediate results are saved in a `.RData` file with the participant/subject ID in the name.

`precision`	numerical numeric. Precision of calculation for the density functions (see `ddynaViTE` and `dPCRM` for more information).
`parallel`	"models", "single", "both" or `FALSE`. If `FALSE` no parallelization is used in the fitting process. If "models" the fitting process is parallelized over participants and models (i.e. over the calls for fitting functions). If "single" parallelization is used within the fitting processes (over initial grid search and optimization processes for different start points, but see `fitRTConf`). If "both", parallelization is done hierarchical. For small number of models and participants "single" or "both" is preferable. Otherwise, you may use "models".
`n.cores`	integer vector or `NULL`. If `parallel` is "models" or "single", a single integer for the number of cores used for parallelization is required. If `parallel` is "both", two values are required. The first for the number of parallel model-participant combinations and the second for the parallel processes within the fitting procedures (this may be specified to match the `nAttemps`-Value in the `opts` argument. If `NULL` (default) the number of available cores -1 is used. If `NULL` and `parallel` is "both", the cores will be used for model-participant-parallelization, only.
`...`	Possibility of giving alternative variable names in data frame (in the form `condition = "SOA"`, or `response="pressedKey"`).

Details

The fitting involves a first grid search through an initial grid. Then the best `nAttempts` parameter sets are chosen for an optimization, which is done with an algorithm, depending on the argument `optim-method`. The Nelder-Mead algorithm uses the R function `optim`. The optimization routine is restarted `nRestarts` times with the starting parameter set equal to the best parameters from the previous routine.

stimulus, response and correct. Two of these columns must be given in data. If all three are given, correct will have no effect (and will be not checked!). stimulus can always be given in numerical format with values -1 and 1. response can always be given as a character vector with "lower" and "upper" as values. Correct must always be given as a 0-1-vector. If stimulus is given together with response and they both do not match the above format, they need to have the same values/levels (if factor). In the case that only stimulus/response is given in any other format together with correct, the unique values will be sorted increasingly and the first value will be encoded as "lower"/-1 and the second as "upper"/+1.

fixed. Parameters that should not be fitted but kept constant. These will be dropped from the initial grid search but will be present in the output, to keep all parameters for prediction in the result. Includes the possibility for symmetric confidence thresholds for both alternative (`sym_thetas`=logical). Other examples are `z =.5`, `sv=0`, `st0=0`, `sz=0`. For race models, the possibility of setting `a='b'` (or vice versa) leads to identical upper bounds on the decision processes, which is the equivalence for `z=.5` in a diffusion process

opts. A list with numerical values. Possible options are listed below (together with the optimization method they are used for).

- `nAttempts` (all) number of best performing initial parameter sets used for optimization; default 5
- `nRestarts` (all) number of successive `optim` routines for each of the starting parameter sets; default 5,

- `maxfun` (`'bobyqa'`) maximum number of function evaluations; default: 5000,
- `maxit` (`'Nelder-Mead'` and `'L-BFGS-B'`) maximum iterations; default: 2000,
- `reltol` (`'Nelder-Mead'`) relative tolerance; default: 1e-6),
- `factr` (`'L-BFGS-B'`) tolerance in terms of reduction factor of the objective, default: 1e-10)

Value

Gives data frame with rows for each model-participant combination and columns for the different parameters as fitted result as well as additional information about the fit (`negLogLik` (for final parameters), `k` (number of parameters), `N` (number of data rows), `BIC`, `AICc` and `AIC`)

Author(s)

Sebastian Hellmann, `<sebastian.hellmann@ku.de>`

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# 1. Generate data from two artificial participants
# Get random drift direction (i.e. stimulus category) and
# stimulus discriminability (two steps: hard, easy)
stimulus <- sample(c(-1, 1), 400, replace=TRUE)
discriminability <- sample(c(1, 2), 400, replace=TRUE)

# generate data for participant 1
data <- rdynaViTE(400, a=2, v=stimulus*discriminability*0.5,
             t0=0.2, z=0.5, sz=0.1, sv=0.1, st0=0,  tau=4, s=1, w=0.3)
# discretize confidence ratings (only 2 steps: unsure vs. sure)
data$rating <- as.numeric(cut(data$conf, breaks = c(-Inf, 1, Inf), include.lowest = TRUE))
data$participant = 1
data$stimulus <- stimulus
data$discriminability <- discriminability
# generate data for participant 2
data2 <- rdynaViTE(400, a=2.5, v=stimulus*discriminability*0.7,
             t0=0.1, z=0.7, sz=0, sv=0.2, st0=0,  tau=2, s=1, w=0.5)
data2$rating <- as.numeric(cut(data$conf, breaks = c(-Inf, 0.3, Inf), include.lowest = TRUE))
data2$participant = 2
data2$stimulus <- stimulus
data2$discriminability <- discriminability

# bind data from participants
data <- rbind(data, data2)
data <- data[data$response!=0, ] # drop not finished decision processes
data <- data[,-3] # drop conf measure (unobservable variable)
head(data)
```

```
# 2. Use fitting function
## Not run:
  # Fitting takes very long to run and uses multiple (6) cores with this
  # call:
  fitRTConfModels(data, models=c("dynWEV", "PCRM"), nRatings = 2,
                  logging=FALSE, parallel="both",
                  n.cores = c(2,3), # fit two participant-model combination in parallel
                  condition="discriminability")# tell which column is "condition"

## End(Not run)
```

LogLikRM — *Log-Likelihood functions for the independent and partially anti-correlated race models of confidence*

Description

Computes the Log-likelihood for given data and parameters in the IRM and PCRM with or without time-scaled confidence measure. It is a wrapped version of the respective densities `dIRM` and `dPCRM`, where one can find more information about the parameters. It restricts the rates of accumulation to be the negative of each other, though (a common assumption in perceptual decision tasks). The function is mainly used inside `fitRTConf` for race models but exported for individual usage in other contexts.

Usage

```
LogLikRM(data, paramDf, model = "IRM", time_scaled = FALSE,
  precision = 6, data_names = list(), ...)
```

Arguments

`data` a dataframe where each row is one trial. Containing following variables:

- condition (not necessary; convertible to integer (e.g. factor); for different levels of stimulus quality),
- rating (convertible to integer (e.g. factor); discrete confidence judgments),
- rt (numeric; giving reaction times for decision task),
- stimulus (values at least convertible to c(1,2), i.e. integer or factor; stimulus category (index of accumulator with higher drift))
- response (values at least convertible to c(1,2); direction of decision; (index of accumulator reaching the boundary first))

`paramDf` a list or data frame with one row. Column names should match the names of RaceModels parameter names (only `mu1` and `mu2` are not used in this context but replaced by the parameter `v`). For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `s` parameters are possible with `s1`,

	`s2`, `s3`,... with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,.... (see Details for the correspondence to the data)
`model`	character scalar. One of "IRM" or "PCRM". ("IRMt" and "PCRMt" will also be accepted. In that case, time_scaled is set to TRUE.)
`time_scaled`	logical. Whether the confidence measure should be scaled by 1/sqrt(rt). Default: TRUE.
`precision`	numerical scalar. Precision of calculation for integration over t0.
`data_names`	list. Possibility of giving alternative column names for the variables in the data. By default column names are identical to the ones given in the data argument description.
`...`	Another possibility of giving alternative variable names in data frame (in the form `condition = "SOA"`).

Details

Note, that the requirements on the format of the columns for the likelihood functions are much stricter, than in `fitRTConf`. This is because the function is very frequently called in the optimization routines of the fitting process and the preprocessing steps are therefore included in the other function.

rating, condition. If integer, values should range from 1 to number of possible ratings/conditions. If factor, the number of levels should be equal to number of possible ratings/conditions. This should be consistent with the parameter vector. The confidence thresholds should be named as `thetaUpper1`, `thetaLower1`,.... (or `theta1`,... for symmetric thresholds), with the number of ratings -1 and the mean drift rates (and possibly the standard deviation in drift rates) should be denoted as `v1`, `v2`,... (and `s1`, `s2`,...) with the number equal to the number of conditions. If only one condition is used `v` will be accepted as well as `v1`.

stimulus, response. stimulus and response should always be given in numerical format with values 1 and 2. Stimulus determines which of two accumulators has positive drift. The other has negative drift with the same absolute value. Response gives the index of the accumulator that reaches the boundary first.

Value

Numeric scalar. The summed Log-likelihood of the data given the parameters in the respective model. If one or more row-wise probabilities is <=0, the function returns -1e+12.

Author(s)

Sebastian Hellmann.

Examples

```
# 1. Generate data from an artificial participants
# Get random index for accumulator with positive
# drift (i.e. stimulus category) and
# stimulus discriminability (two steps: hard, easy)
```

```
stimulus <- sample(c(1, 2), 200, replace=TRUE)
discriminability <- sample(c(1, 2), 200, replace=TRUE)
# generate data for participant 1
data <- rPCRM(200, mu1=ifelse(stimulus==1, 1, -1)*discriminability*0.5,
              mu2=ifelse(stimulus==1, -1, 1)*discriminability*0.5,
             a=2, b=1.8, t0=0.2, st0=0, wx=0.7, wint=0.3, wrt=0)
# discretize confidence ratings (only 2 steps: unsure vs. sure)
data$rating <- as.numeric(cut(data$conf, breaks = c(0, 3, Inf), include.lowest = TRUE))
data$stimulus <- stimulus
data$discriminability <- discriminability
data <- data[data$response!=0, ] # drop not finished decision processes
data <- data[,-c(3,4)] # drop xl and conf measure (unobservable variable)
head(data)

# 2. Define some parameter set in a data.frame
paramDf <- data.frame(a=2,b=2, v1=0.5, v2=1, t0=0.1,st0=0,
                      wx=0.6, wint=0.2, wrt=0.2,
                      theta1=4)

# 3. Compute log likelihood for parameter and data
LogLikRM(data, paramDf, model="PCRMt", condition="discriminability")
# same result
LogLikRM(data, paramDf, model="PCRM", time_scaled=TRUE,condition="discriminability")
# different
LogLikRM(data, paramDf, model="PCRM", condition="discriminability")

# same parameters used for IRM model
LogLikRM(data, paramDf, model="IRMt", condition="discriminability")
```

`LogLikWEV` *Log-Likelihood functions for the dynWEV and 2DSD models of confidence*

Description

Computes the Log-likelihood for given data and parameters in the dynWEV model (Hellmann et al., 2023) and the 2DSD model (Pleskac & Busemeyer, 2010). It is a wrapped version of the respective densities `ddynaViTE` and `d2DSD`, where one can find more information about the parameters (`z` is always given relatively, in the likelihood). The function is mainly used in `fitRTConf` but exported for individual usage in other contexts.

Usage

```
LogLikWEV(data, paramDf, model = "dynaViTE", simult_conf = FALSE,
  precision = 6, stop_on_error = TRUE, data_names = list(), ...)
```

Arguments

`data`	a dataframe where each row is one trial. Containing following variables: • condition (not necessary; convertible to integer (e.g. factor); for different levels of stimulus quality), • rating (convertible to integer (e.g. factor); discrete confidence judgments), • rt (numeric; giving reaction times for decision task), • stimulus (values at least convertible to c(-1,1); stimulus category (direction of evidence accumulation)) • response (characters in `"upper"`, `"lower"` (or convertible to); direction of decision; correct answers are "lower" for stimulus=-1; and "upper" for stimulus=1),
`paramDf`	list or data.frame with one row. Names should match the names of dynaViTE and 2DSD model specific parameter names. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `sv` and/or `s` parameters are possible with `sv1`, `sv2`, `sv3`... (`s1`, `s2`, `s3`,... respectively) with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,... (see Details for the correspondence to the data)
`model`	character scalar. One of "dynWEV" or "2DSD" for the model to fit.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and computations are different, when there is an observable interjudgment time (then `simult_conf` should be `FALSE`).
`precision`	numerical scalar. Precision of calculation for integration over z and t0.
`stop_on_error`	logical. If TRUE an error in the function will be returned in case of invalid parameters. Otherwise, the output will be 0 without error.
`data_names`	list. Possibility of giving alternative column names for the variables in the data. By default column names are identical to the ones given in the data argument description.
`...`	Possibility of giving alternative variable names in data frame (in the form `condition = "SOA"`).

Details

Note, that the requirements on the format of the columns for the likelihood functions are much stricter, than in `fitRTConf`. This is because the function is very frequently calls in the optimization routines of the fitting process and the preprocessing steps are therefore included in that function.

rating, condition. If integer, values should range from 1 to number of possible ratings/conditions. If a factor, the number of levels should be equal to number of possible ratings/conditions. This should be consistent with the parameter vector. The confidence thresholds should be named as `thetaUpper1`, `thetaLower1`,.... (or `theta1`,... for symmetric thresholds), with the number of ratings -1 and the mean drift rates (and possibly the standard deviation in drift rates) should be denoted as `v1`, `v2`,... (and `sv1`, `sv2`,.../`s1`, `s2`, ...) with the number equal to the number of conditions. If only one condition is used `v` will be accepted as well as `v1`.

stimulus, response. stimulus should always be given in numerical format with values -1 and 1. response should always be given as a character vector with `"lower"` and `"upper"` as values. This corresponds to the situation of Ratcliff's diffusion model (Ratcliff, 1978), where stimulus is the sign of the mean drift direction and the response is the `"upper"` or `"lower"` boundary that is first hit by the evidence accumulation. A correct decision is therefore `"lower"`, if stimulus is -1, and `"upper"`, if stimulus is 1.

Value

Numeric scalar. The summed Log-likelihood of the data given the parameters in the respective model. If one or more row-wise probabilities is <=0, the function returns -1e+12.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# 1. Generate data from an artificial participants
# Get random drift direction (i.e. stimulus category) and
# stimulus discriminability (two steps: hard, easy)
stimulus <- sample(c(-1, 1), 200, replace=TRUE)
discriminability <- sample(c(1, 2), 200, replace=TRUE)
# generate data for participant 1
data <- rdynaViTE(200, a=2,v=stimulus*discriminability*0.5,
             t0=0.2,z=0.5, sz=0.1,sv=0.1, st0=0,  tau=4, s=1, w=0.3)
# discretize confidence ratings (only 2 steps: unsure vs. sure)
data$rating <- as.numeric(cut(data$conf, breaks = c(-Inf, 1, Inf), include.lowest = TRUE))
data$stimulus <- stimulus
data$discriminability <- discriminability
data <- data[data$response!=0, ] # drop not finished decision processes
data <- data[,-3] # drop conf measure (unobservable variable)
head(data)

# 2. Define some parameter set in a data.frame
paramDf <- data.frame(a=2.5,v1=0.5, v2=1, t0=0.1,z=0.7,
                      sz=0,sv=0.2, st0=0,  tau=3, w=0.3,
                      theta1=0.8, svis=0.5, sigvis=0.8)

# 3. Compute log likelihood for parameter and data
LogLikWEV(data, paramDf, model="dynWEV", condition="discriminability")
# adding the hypothetical interjudgment time to response times
# results in the same log likelihood as before when simult_conf=TRUE
data$rt <- data$rt + paramDf$tau
LogLikWEV(data, paramDf, model="dynWEV", condition="discriminability", simult_conf=TRUE)
```

```
# the same function for "2DSD" model
paramDf <- data.frame(a=2.5,v1=0.5, v2=1, t0=0.1,z=0.7,
                      sz=0,sv=0.2, st0=0,  tau=3, theta1=0.8)
LogLikWEV(data, paramDf, model="2DSD", condition="discriminability", simult_conf=TRUE)
# this results in the same log likelihood as before
```

`MLE_dirichlet` *Estimate the parameter of a Dirichlet distribution*

Description

The function `MLE_dirichlet` performs a maximum-likelihood estimation of the α parameter of a Dirichlet distribution for a given sample of probability vectors.

Usage

```
MLE_dirichlet(probs, alpha0 = rep(1, ncol(probs)))
```

Arguments

`probs`	a matrix with N rows representing observations of probability vectors and K columns representing the classes. Therefore, values of each row should sum to 1.
`alpha0`	vector of K=ncol(probs) values as starting parameter for the optimization. Values have to be greater 0.

Details

The density of the Dirichlet distribution for $\alpha = (\alpha_1, ..., \alpha_K)$ and $\alpha_i > 0 \forall i = 1, ..., K$ is given by

$$f(p|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1} K p_i^{\alpha_i - 1},$$

if $0 \leq p_i \leq 1 \forall i = 1, ..., K$ and $\sum_{i=1}^{K} p_i)1$, and $f(p|\alpha) = 0$, else.

The function optimizes the log-likelihood of a sample of probability vectors given in `probs` using the function `optim` and a Nelder-Mead algorithm.

Value

Returns a numeric vector of length K=ncol(probs) representing the α of the Dirichlet distribution.

Author(s)

Sebastian Hellmann.

Examples

```
probs <- matrix(c(0.2, 0.4, 0.2, 0.4, 0, 0.2, 0.4, 0.4, 0.6, 0.2, 0.2,
                0.4, 0.4, 0.2, 0.2, 0.4, 0.8, 0.4), ncol=3)
MLE_dirichlet(probs)
```

`PDFtoQuantiles` *Get Quantiles from vectors of PDF or CDF values*

Description

`CDFtoQuantiles` computes quantiles for a given CDF. `PDFtoQuantiles` computes the quantiles for given PDF values within groups of other variables, if available.

Usage

```
PDFtoQuantiles(pdf_df, p = c(0.1, 0.3, 0.5, 0.7, 0.9), agg_over = NULL,
  scaled = FALSE)

CDFtoQuantiles(cdf, x = NULL, p)
```

Arguments

`pdf_df` dataframe. Should have at least two columns:

- `rt` (for reaction times) or `x` for the support values of the pdf
- `dens` or `pdf` for the pdf values
- All other columns will be used as grouping factors, for which separate quantiles will be returned.

`p` numeric vector. Probabilities for returned quantiles. Default: c(.1, .3, .5, .7, .9).

`agg_over` character. Names of columns in `pdf_df` to aggregate over (using the mean of densities, which is valid only, if groups occur with equal probabilities) before computing the quantiles.

`scaled` logical. Indicating whether the pdf values are from a proper probability distribution. Non-scaled pdfs will scaled to 1. If `scaled` is TRUE, this may cause problems with high probabilities. In any case we strongly recommend to cover the most probability mass with the values in the support vector.

`cdf` numeric. A increasing vector of the same length as `x` giving the CDF for respective x-Values. Dataframe inputs are accepted. If a column `x` is available there, this will be used as support values.

`x` numeric. A increasing vector of same length as `cdf`. Can also be specified as column of `cdf`.

Details

For a reasonable accuracy the number of steps in the support column (`rt`/`x`) should be high, i.e. the distance between values small. We recommend, to ensure that the support vector in the input to be equidistant, i.e. the difference between consecutive support values should be constant, though this is not required. If both column names `x` and `rt` are present in `pdf_df`, `rt` will be preferred. Attention should be given to the columns of `pdf_df` other than `rt`/`x` and `dens`/`pdf`.

The column for the pdf may be scaled to integrate to 1 but do not have to.

Quantile computation in the `dynConfiR` package:

As argument `pdf_df`, the outputs of `predictRT` and `predictRTModels` from the `dynConfiR` package can be used. In the context of confidence models grouping factors often used are conditions, correct/incorrect answers and confidence ratings.

Value

`PDFtoQuantiles` returns a `tibble` with columns p and q indicating probabilities and respective quantiles. Furthermore, the output has grouping columns identical to the additional columns in the input (without `rt`/`x`, `dens` and `densscaled`), but without the ones in the `agg_over` argument. `CDFtoQuantiles` returns only a `data.frame` with columns `p` and `q`.

Author(s)

Sebastian Hellmann.

Examples

```
## Demonstrate PDFtoQuantiles
pred <- expand.grid(model = c("dynWEV", "PCRMt"),
                    rt =  seq(0, 15, length.out=1200),
                    condition = c(1,2,3),
                    rating = c(1,2))
pred$dens <- dchisq(pred$rt, 3) # pdf may also be used as column name
head(pred)
res <- PDFtoQuantiles(pred, p=c(0.3, 0.5, 0.7))
head(res)
nrow(res) #= 3(quantiles)*2(models)*3(conditions)*2(rating)
# Compare to true quantiles of Chi-square distribution
qchisq(p=c(0.3, 0.5, 0.7), 3)
res$q[1:3]


res2 <- PDFtoQuantiles(pred, p=c(0.3, 0.5, 0.7), agg_over = "model")
nrow(res2) #=18 because res aggregated over models


  pred$pdf <- dchisq(pred$rt, 3)
  head(pred)
  # following call throws a warning, because both columns pdf and dens are present
  PDFtoQuantiles(pred, p=c(0.3, 0.5, 0.7), agg_over = "model")
```

```
  pred2 <- data.frame(rt=seq(0, 7, length.out=100))
  pred2$dens <- dchisq(pred2$rt, 5)
 # following call throws a warning, because density is assumed to be scaled (scaled=TRUE), i.e.
  # integrate to 1, but the .95 quantile is not reached in the rt column
  PDFtoQuantiles(pred2, p=c(0.3, 0.5, 0.95), scaled=TRUE) # Gives a warning


## Demonstrate CDFtoQuantiles
X <- seq(-2, 2, length.out=300)
pdf_values <- pnorm(X)
CDFtoQuantiles(pdf_values, X, p=c(0.2, 0.5, 0.8))
qnorm(c(0.2, 0.5, 0.8))
```

`predictDDConf` *Prediction of Confidence Rating and Reaction Time Distribution in the drift diffusion confidence model*

Description

`predictDDConf_Conf` predicts the categorical response distribution of decision and confidence ratings, `predictDDConf_RT` computes the RT distribution (density) in the drift diffusion confidence model (Hellmann et al., 2023), given specific parameter constellations. See `dDDConf` for more information about the model and parameters.

Usage

```
predictDDConf_Conf(paramDf, maxrt = 15, subdivisions = 100L,
  stop.on.error = FALSE, .progress = TRUE)

predictDDConf_RT(paramDf, maxrt = 9, subdivisions = 100L, minrt = NULL,
  scaled = FALSE, DistConf = NULL, .progress = TRUE)
```

Arguments

`paramDf`	a list or data frame with one row. Column names should match the names of DDConf model parameter names. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `sv` and/or `s` parameters are possible with `sv1`, `sv2`, `sv3`... (`s1`, `s2`, `s3`,... respectively) with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....
`maxrt`	numeric. The maximum RT for the integration/density computation. Default: 15 (for `predictDDConf_Conf` (integration)), 9 (for `predictDDConf_RT`).
`subdivisions`	`integer` (default: 100). For `predictDDConf_Conf` it is used as argument for the inner integral routine. For `predictDDConf_RT` it is the number of points for which the density is computed.

`stop.on.error`	logical. Argument directly passed on to integrate. Default is `FALSE`, since the densities invoked may lead to slow convergence of the integrals (which are still quite accurate) which causes R to throw an error.
`.progress`	logical. If `TRUE` (default) a progress bar is drawn to the console.
`minrt`	numeric or `NULL`(default). The minimum rt for the density computation.
`scaled`	logical. For `predictDDConf_RT`. Whether the computed density should be scaled to integrate to one (additional column `densscaled`). Otherwise the output contains only the defective density (i.e. its integral is equal to the probability of a response and not 1). If `TRUE`, the argument `DistConf` should be given, if available. Default: `FALSE`.
`DistConf`	`NULL` or `data.frame`. A `data.frame` or `matrix` with column names, giving the distribution of response and rating choices for different conditions and stimulus categories in the form of the output of `predictDDConf_Conf`. It is only necessary, if `scaled=TRUE`, because these probabilities are used for scaling. If `scaled=TRUE` and `DistConf=NULL`, it will be computed with the function `predictDDConf_Conf`, which takes some time and the function will throw a message. Default: `NULL`

Details

The function `predictDDConf_Conf` consists merely of an integration of the response time density, `dDDConf`, over the response time in a reasonable interval (0 to `maxrt`). The function `predictDDConf_RT` wraps these density functions to a parameter set input and a `data.frame` output. For the argument `paramDf`, the output of the fitting function `fitRTConf` with the DDConf model may be used.

Value

`predictDDConf_Conf` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `p`, `info`, `err`. `p` is the predicted probability of a response and `rating`, given the stimulus category and condition. `info` and `err` refer to the respective outputs of the integration routine used for the computation. `predictDDConf_RT` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `rt` and `dens` (and `densscaled`, if `scaled=TRUE`).

Note

Different parameters for different conditions are only allowed for drift rate `v`, drift rate variability `sv`, and process variability `s`. Otherwise, `s` is not required in `paramDf` but set to 1 by default. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=2,v1=0.5, v2=1, t0=0.1,z=0.55,
                      sz=0,sv=0.2, st0=0, theta1=0.8)

# 2. Predict discrete Choice x Confidence distribution:
preds_Conf <- predictDDConf_Conf(paramDf,  maxrt = 15)
head(preds_Conf)

# 3. Compute RT density
preds_RT <- predictDDConf_RT(paramDf, maxrt=4, subdivisions=200) #(scaled=FALSE)
# same output with scaled density column:
preds_RT <- predictDDConf_RT(paramDf, maxrt=4, subdivisions=200,
                              scaled=TRUE, DistConf = preds_Conf)
head(preds_RT)


  # Example of visualization
  library(ggplot2)
  preds_Conf$rating <- factor(preds_Conf$rating, labels=c("unsure", "sure"))
  preds_RT$rating <- factor(preds_RT$rating, labels=c("unsure", "sure"))
  ggplot(preds_Conf, aes(x=interaction(rating, response), y=p))+
    geom_bar(stat="identity")+
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")
  ggplot(preds_RT, aes(x=rt, color=interaction(rating, response), y=dens))+
    geom_line(stat="identity")+
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")+
    theme(legend.position = "bottom")
  ggplot(aggregate(densscaled~rt+correct+rating+condition, preds_RT, mean),
         aes(x=rt, color=rating, y=densscaled))+
    geom_line(stat="identity")+
    facet_grid(cols=vars(condition), rows=vars(correct), labeller = "label_both")+
    theme(legend.position = "bottom")

# Use PDFtoQuantiles to get predicted RT quantiles
head(PDFtoQuantiles(preds_RT, scaled = FALSE))
```

`predictRM` *Prediction of Confidence Rating and Reaction Time Distribution in race models of confidence*

Description

`predictRM_Conf` predicts the categorical response distribution of decision and confidence ratings, `predictRM_RT` computes the RT distribution (density) in the independent and partially anti-correlated race models (Hellmann et al., 2023), given specific parameter constellations. See RaceModels for more information about the models and parameters.

Usage

```
predictRM_Conf(paramDf, model = "IRM", time_scaled = FALSE, maxrt = 15,
  subdivisions = 100L, stop.on.error = FALSE, .progress = TRUE)

predictRM_RT(paramDf, model = "IRM", time_scaled = FALSE, maxrt = 9,
  subdivisions = 100L, minrt = NULL, scaled = FALSE, DistConf = NULL,
  .progress = TRUE)
```

Arguments

`paramDf`	a list or data frame with one row. Column names should match the names of RaceModels parameter names (only `mu1` and `mu2` are not used in this context but replaced by the parameter `v`). For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `s` parameters are possible with `s1`, `s2`, `s3`,... with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....
`model`	character scalar. One of "IRM" or "PCRM". ("IRMt" and "PCRMt" will also be accepted. In that case, time_scaled is set to TRUE.)
`time_scaled`	logical. Whether the confidence measure should be scaled by 1/sqrt(rt). Default: FALSE. (It is set to TRUE, if model is "IRMt" or "PCRMt")
`maxrt`	numeric. The maximum RT for the integration/density computation. Default: 15 (for `predictRM_Conf` (integration)), 9 (for `predictRM_RT`).
`subdivisions`	`integer` (default: 100). For `predictRM_Conf` it is used as argument for the inner integral routine. For `predictRM_RT` it is the number of points for which the density is computed.
`stop.on.error`	logical. Argument directly passed on to integrate. Default is FALSE, since the densities invoked may lead to slow convergence of the integrals (which are still quite accurate) which causes R to throw an error.
`.progress`	logical. If TRUE (default) a progress bar is drawn to the console.
`minrt`	numeric or NULL(default). The minimum rt for the density computation.
`scaled`	logical. For `predictRM_RT`. Whether the computed density should be scaled to integrate to one (additional column `densscaled`). Otherwise the output contains only the defective density (i.e. its integral is equal to the probability of a response and not 1). If `TRUE`, the argument `DistConf` should be given, if available. Default: `FALSE`.
`DistConf`	`NULL` or `data.frame`. A `data.frame` or `matrix` with column names, giving the distribution of response and rating choices for different conditions and stimulus categories in the form of the output of `predictRM_Conf`. It is only necessary, if `scaled=TRUE`, because these probabilities are used for scaling. If `scaled=TRUE` and `DistConf=NULL`, it will be computed with the function `predictRM_Conf`, which takes some time and the function will throw a message. Default: `NULL`

Details

The function `predictRM_Conf` consists merely of an integration of the response time density, `dIRM` and `dPCRM`, over the response time in a reasonable interval (0 to `maxrt`). The function `predictRM_RT` wraps these density functions to a parameter set input and a data.frame output. For the argument `paramDf`, the output of the fitting function `fitRTConf` with the respective model may be used.

The drift rate parameters differ from those used in `dIRM`/`dPCRM` since in many perceptual decision experiments the drift on one accumulator is assumed to be the negative of the other. The drift rate of the correct accumulator is `v` (`v1`, `v2`, ... respectively) in `paramDf`.

Value

`predictRM_Conf` returns a `data.frame`/`tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `p`, `info`, `err`. `p` is the predicted probability of a response and `rating`, given the stimulus category and condition. `info` and `err` refer to the respective outputs of the integration routine used for the computation. `predictRM_RT` returns a `data.frame`/`tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `rt` and `dens` (and `densscaled`, if `scaled=TRUE`).

Note

Different parameters for different conditions are only allowed for drift rate, `v`, and process variability `s`. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# Examples for "PCRM" model (equivalent applicable for "IRM" model)
# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=2,b=2, v1=0.5, v2=1, t0=0.1,st0=0,
                      wx=0.6, wint=0.2, wrt=0.2,
                      theta1=4)

# 2. Predict discrete Choice x Confidence distribution:
preds_Conf <- predictRM_Conf(paramDf, "PCRM", time_scaled=TRUE)
# equivalent:
preds_Conf <- predictRM_Conf(paramDf, "PCRMt")
head(preds_Conf)

# 3. Compute RT density
preds_RT <- predictRM_RT(paramDf, "PCRMt", maxrt=7, subdivisions=50)
# same output with scaled density column:
preds_RT <- predictRM_RT(paramDf, "PCRMt", maxrt=7, subdivisions=50,
                         scaled=TRUE, DistConf = preds_Conf)
head(preds_RT)

  # produces a warning, if scaled=TRUE and DistConf missing
```

```
preds_RT <- predictRM_RT(paramDf, "PCRMt", maxrt=7, subdivisions=50,
                         scaled=TRUE)



# Example of visualization
library(ggplot2)
preds_Conf$rating <- factor(preds_Conf$rating, labels=c("unsure", "sure"))
preds_RT$rating <- factor(preds_RT$rating, labels=c("unsure", "sure"))
ggplot(preds_Conf, aes(x=interaction(rating, response), y=p))+
  geom_bar(stat="identity")+
  facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")
ggplot(preds_RT, aes(x=rt, color=interaction(rating, response), y=dens))+
  geom_line(stat="identity")+
  facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")+
  theme(legend.position = "bottom")
ggplot(aggregate(densscaled~rt+correct+rating+condition, preds_RT, mean),
       aes(x=rt, color=rating, y=densscaled))+
  geom_line(stat="identity")+
  facet_grid(cols=vars(condition), rows=vars(correct), labeller = "label_both")+
  theme(legend.position = "bottom")


# Use PDFtoQuantiles to get predicted RT quantiles
# (produces warning because of few rt steps (--> inaccurate calculations))
PDFtoQuantiles(preds_RT, scaled = FALSE)
```

`predictRTConf` *Prediction of confidence rating and response time distribution for sequential sampling confidence models*

Description

`predictConf` predicts the categorical response distribution of decision and confidence ratings, `predictRT` computes the predicted RT distribution (density) for the sequential sampling confidence model specified by the argument `model`, given specific parameter constellations. This function calls the respective functions for diffusion based models (dynWEV and 2DSD: `predictWEV`) and race models (IRM, PCRM, IRMt, and PCRMt: `predictRM`).

Usage

```
predictConf(paramDf, model = NULL, maxrt = 15, subdivisions = 100L,
  simult_conf = FALSE, stop.on.error = FALSE, .progress = TRUE)

predictRT(paramDf, model = NULL, maxrt = 9, subdivisions = 100L,
  minrt = NULL, simult_conf = FALSE, scaled = FALSE, DistConf = NULL,
  .progress = TRUE)
```

Arguments

`paramDf`	a list or dataframe with one row. Column names should match the names of the respective model parameters. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `s` parameters are possible with `s1`, `s2`, `s3`... with equally many steps as for drift rates (same for `sv` parameter in dynWEV and 2DSD). Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....
`model`	character scalar. One of "2DSD", "dynWEV", "IRM", "PCRM", "IRMt", or "PCRMt".
`maxrt`	numeric. The maximum RT for the integration/density computation. Default: 15 (for `predictConf` (integration)), 9 (for `predictRT`).
`subdivisions`	`integer` (default: 100). For `predictConf` it is used as argument for the inner integral routine. For `predictRT` it is the number of points for which the density is computed.
`simult_conf`	logical, only relevant for dynWEV and 2DSD. Whether in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and computations are different, when there is an observable interjudgment time (then `simult_conf` should be FALSE).
`stop.on.error`	logical. Argument directly passed on to integrate. Default is FALSE, since the densities invoked may lead to slow convergence of the integrals (which are still quite accurate) which causes R to throw an error.
`.progress`	logical. If TRUE (default) a progress bar is drawn to the console.
`minrt`	numeric or NULL(default). The minimum rt for the density computation.
`scaled`	logical. For `predictRT`. Whether the computed density should be scaled to integrate to one (additional column `densscaled`). Otherwise the output contains only the defective density (i.e. its integral is equal to the probability of a response and not 1). If `TRUE`, the argument `DistConf` should be given, if available. Default: `FALSE`.
`DistConf`	`NULL` or `data.frame`. A `data.frame` or `matrix` with column names, giving the distribution of response and rating choices for different conditions and stimulus categories in the form of the output of `predictConf`. It is only necessary, if `scaled=TRUE`, because these probabilities are used for scaling. If `scaled=TRUE` and `DistConf=NULL`, it will be computed with the function `predictRM_Conf`, which takes some time and the function will throw a message. Default: `NULL`

Details

The function `predictConf` consists merely of an integration of the reaction time density of the given model, `{d*model*}`, over the response time in a reasonable interval (0 to `maxrt`). The function `predictRT` wraps these density functions to a parameter set input and a data.frame output. For the argument `paramDf`, the output of the fitting function `fitRTConf` with the respective model may be used.

Value

`predictConf` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `p`, `info`, `err`. `p` is the predicted probability of a response and `rating`, given the stimulus category and condition. `info` and `err` refer to the respective outputs of the integration routine used for the computation. `predictRT` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `rt` and `dens` (and `densscaled`, if `scaled=TRUE`).

Note

Different parameters for different conditions are only allowed for drift rate, `v`, drift rate variability, `sv` (in dynWEV and 2DSD), and process variability `s`. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

Examples

```
# Examples for "dynWEV" model (equivalent applicable for
# all other models (with different parameters!))

# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=1.5,v1=0.2, v2=1, t0=0.1,z=0.52,
                      sz=0.3,sv=0.4, st0=0,  tau=3, w=0.5,
                      theta1=1, svis=0.5, sigvis=0.8)

# 2. Predict discrete Choice x Confidence distribution:
preds_Conf <- predictConf(paramDf, "dynWEV", maxrt = 25, simult_conf=TRUE)
head(preds_Conf)


# 3. Compute RT density
preds_RT <- predictRT(paramDf, "dynWEV") #(scaled=FALSE)
# same output with default rt-grid and without scaled density column:
preds_RT <- predictRT(paramDf, "dynWEV", maxrt=5, subdivisions=200,
                      minrt=paramDf$tau+paramDf$t0, simult_conf = TRUE,
                      scaled=TRUE, DistConf = preds_Conf)
head(preds_RT)

  # produces a warning, if scaled=TRUE and DistConf missing
  preds_RT <- predictRT(paramDf, "dynWEV",
                           scaled=TRUE)



  # Example of visualization
  library(ggplot2)
  preds_Conf$rating <- factor(preds_Conf$rating, labels=c("unsure", "sure"))
  preds_RT$rating <- factor(preds_RT$rating, labels=c("unsure", "sure"))
  ggplot(preds_Conf, aes(x=interaction(rating, response), y=p))+
```

```
    geom_bar(stat="identity")+
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")
  ggplot(preds_RT, aes(x=rt, color=interaction(rating, response), y=densscaled))+
    geom_line(stat="identity")+
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")+
    theme(legend.position = "bottom")+ ggtitle("Scaled Densities")
  ggplot(aggregate(dens~rt+correct+rating+condition, preds_RT, mean),
         aes(x=rt, color=rating, y=dens))+
    geom_line(stat="identity")+
    facet_grid(cols=vars(condition), rows=vars(correct), labeller = "label_both")+
    theme(legend.position = "bottom")+ ggtitle("Non-Scaled Densities")

# Use PDFtoQuantiles to get predicted RT quantiles
head(PDFtoQuantiles(preds_RT, scaled = FALSE))
```

`predictRTConfModels` *Prediction of confidence and RT distributions for several sequential sampling confidence models and parameter constellations in parallel*

Description

This function is a wrapper around the functions `predictRTConf` (see there for more information). It calls the respective function for predicting the response distribution (discrete decision and rating outcomes) and the rt density (density for decision, rating and response time) for every model and participant/subject combination in `paramDf`. Also, see `ddynaViTE`, `d2DSD`, and `dRM` for more information about the parameters.

Usage

```
predictConfModels(paramDf, maxrt = 15, subdivisions = 100L,
  simult_conf = FALSE, stop.on.error = FALSE, .progress = TRUE,
  parallel = FALSE, n.cores = NULL)

predictRTModels(paramDf, maxrt = 9, subdivisions = 100L, minrt = NULL,
  simult_conf = FALSE, scaled = FALSE, DistConf = NULL,
  .progress = TRUE, parallel = FALSE, n.cores = NULL)
```

Arguments

`paramDf`	a dataframe with one row per combination of model and participant/parameter set. Columns may include a `participant` (`sbj`, or `subject`) column, and must include a `model` column and the names of the model parameters. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `s` parameters are possible with `s1`, `s2`, `s3`... with equally many steps as for drift rates (same for `sv` parameter in dynWEV and 2DSD). Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....

`maxrt`	numeric. The maximum RT for the integration/density computation. Default: 15 (for `predictConfModels` (integration)) and 9 (for `predictRTModels`).
`subdivisions`	`integer` (default: 100). For `predictConfModels` it is used as argument for the inner integral routine. For `predictRTModels` it is the number of points for which the density is computed.
`simult_conf`	logical, only relevant for dynWEV and 2DSD. Whether in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and computations are different, when there is an observable interjudgment time (then `simult_conf` should be FALSE).
`stop.on.error`	logical. Argument directly passed on to integrate. Default is FALSE, since the densities invoked may lead to slow convergence of the integrals (which are still quite accurate) which causes R to throw an error.
`.progress`	logical. If TRUE (default) a progress bar is drawn to the console. (Works for some OS only when `parallel=FALSE`.)
`parallel`	logical. If TRUE, prediction is parallelized over participants and models (i.e. over the calls for the respective `predictRTConf` functions).
`n.cores`	integer. If `parallel` is TRUE, the number of cores used for parallelization is required. If `NULL` (default) the number of available cores -1 is used.
`minrt`	numeric or `NULL`(default). The minimum rt for the density computation. If `NULL`, the minimal possible response time possible with given parameters will be used (min(t0)).
`scaled`	logical. Whether the computed density should be scaled to integrate to one (additional column `densscaled`). Otherwise the output contains only the defective density (i.e. its integral is equal to the probability of a response and not 1). If `TRUE`, the argument `DistConf` should be given, if available. Default: `FALSE`.
`DistConf`	`NULL` or `data.frame`. A `data.frame` with participant and model columns and columns, giving the distribution of response and rating choices for different conditions and stimulus categories in the form of the output of `predictConfModels`. It is only necessary if `scaled=TRUE`, because these probabilities are used for scaling. If `scaled=TRUE` and `DistConf=NULL`, it will be computed with the function `predictConfModels`, which takes some time and the function will throw a message. Default: `NULL`

Details

These functions merely split the input data frame by model participants combinations, call the equivalent `predictRTConf` functions for the individual parameter sets and bind the outputs together. They are included for convenience and the easy parallelization, which facilitates speeding up computations considerably. For the argument `paramDf`, the output of the fitting function `fitRTConfModels` with the respective models and participants may be used.

The function `predictConf` (called by `predictConfModels`) consists merely of an integration of the reaction time density or the given model, `{d*model*}`, over the reaction time in a reasonable interval (0 to `maxrt`). The function `predictRT` (called by `predictRTModels`) wraps these density functions to a parameter set input and a data.frame output. ' Note, that the encoding for stimulus identity is different between diffusion based models (2DSD, dynWEV) and race models (IRM(t), PCRM(t)). Therefore, in the columns stimulus and response there will be a mix of encodings: -1/1 for diffusion based models and 1/2 for race models. This, usually is not important, since for further aggregation models will not be mixed.

Value

`predictConfModels` returns a `data.frame/tibble` with columns: `participant` (or `sbj`, subject depending on the input), `model`, `condition`, `stimulus`, `response`, `rating`, `correct`, `p`, `info`, `err`. `p` is the predicted probability of a response and `rating`, given the stimulus category and condition. `info` and `err` refer to the respective outputs of the integration routine used for the computation. `predictRTModels` returns a `data.frame/tibble` with columns: `participant` (or `sbj/subject` depending on the input), `model`, `condition`, `stimulus`, `response`, `rating`, `correct`, `rt` and `dens` (and `densscaled`, if `scaled=TRUE`).

Note

Different parameters for different conditions are only allowed for drift rate `v`, drift rate variability `sv` (only dynWEV and 2DSD), and process variability `s`. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

Examples

```
# First example for 2 participant and the "dynWEV" model
# (equivalent applicable for
# all other models (with different parameters!))
# 1. Define two parameter sets from different participants
paramDf <- data.frame(participant = c(1,2), model="dynWEV",
                      a=c(1.5, 2),v1=c(0.2,0.1), v2=c(1, 1.5),
                      t0=c(0.1, 0.2),z=c(0.52,0.45),
                      sz=c(0.0,0.3),sv=c(0.4,0.7), st0=c(0,0.01),
                      tau=c(2,3), w=c(0.5,0.2),
                      theta1=c(1,1.5), svis=c(0.5,0.1), sigvis=c(0.8, 1.2))
paramDf
# 2. Predict discrete Choice x Confidence distribution:
# model is not an extra argument but must be a column of paramDf
preds_Conf <- predictConfModels(paramDf, maxrt = 15, simult_conf=TRUE,
                                .progress=TRUE, parallel = FALSE)
# 3. Compute RT density
preds_RT <- predictRTModels(paramDf, maxrt=6, subdivisions=100,
                      scaled=TRUE, DistConf = preds_Conf,
                      parallel=FALSE, .progress = TRUE)
head(preds_RT)

  # produces a warning, if scaled=TRUE and DistConf missing
  preds_RT <- predictRTModels(paramDf, scaled=TRUE)

# Use PDFtoQuantiles to get predicted RT quantiles
```

```
head(PDFtoQuantiles(preds_RT, scaled = FALSE))

# Second Example: only one parameter set but for two different models

  paramDf1 <- data.frame(model="dynWEV", a=1.5,v1=0.2, v2=1, t0=0.1,z=0.52,
                        sz=0.3,sv=0.4, st0=0,  tau=3, w=0.5,
                        theta1=1, svis=0.5, sigvis=0.8)
  paramDf2 <- data.frame(model="PCRMt", a=2,b=2, v1=0.5, v2=1, t0=0.1,st0=0,
                        wx=0.6, wint=0.2, wrt=0.2, theta1=4)
  paramDf <- dplyr::full_join(paramDf1, paramDf2)
  paramDf  # each model parameters sets hat its relevant parameters
  predictConfModels(paramDf, parallel=FALSE, .progress=TRUE)
```

`predictWEV` *Prediction of Confidence Rating and Response Time Distribution in dynaViTE, dynWEV, and 2DSD confidence models*

Description

`predictWEV_Conf` predicts the categorical response distribution of decision and confidence ratings, `predictWEV_RT` computes the predicted RT distribution (density) in the 2DSD Model (Pleskac & Busemeyer, 2010) and the dynWEV model (Hellmann et al., 2023), given specific parameter constellations. See `ddynaViTE` and `d2DSD` for more information about parameters.

Usage

```
predictWEV_Conf(paramDf, model = "dynaViTE", maxrt = 15,
  subdivisions = 100L, simult_conf = FALSE, stop.on.error = FALSE,
  precision = 3, .progress = TRUE)

predictWEV_RT(paramDf, model = NULL, maxrt = 9, subdivisions = 100L,
  minrt = NULL, simult_conf = FALSE, scaled = FALSE, DistConf = NULL,
  precision = 3, .progress = TRUE)
```

Arguments

`paramDf`	a list or dataframe with one row. Column names should match the names of dynaViTE and 2DSD model specific parameter names. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `sv` and/or `s` parameters are possible with `sv1`, `sv2`, `sv3`... (`s1`, `s2`, `s3`,... respectively) with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....
`model`	character scalar. One of "dynaViTE", "dynWEV", or "2DSD".
`maxrt`	numeric. The maximum RT for the integration/density computation. Default: 15 (for `predictWEV_Conf` (integration)), 9 (for `predictWEV_RT`).

`subdivisions`	integer (default: 100). For `predictWEV_Conf` it is used as argument for the inner integral routine. For `predictWEV_RT` it is the number of points for which the density is computed.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and computations are different, when there is an observable interjudgment time (then `simult_conf` should be FALSE).
`stop.on.error`	logical. Argument directly passed on to integrate. Default is FALSE, since the densities invoked may lead to slow convergence of the integrals (which are still quite accurate) which causes R to throw an error.
`precision`	numerical scalar value. Precision of calculation. Corresponds to the step size of integration w.r.t. `z` and `t0`. Default is 1e-5.
`.progress`	logical. if TRUE (default) a progress bar is drawn to the console.
`minrt`	numeric or NULL(default). The minimum rt for the density computation.
`scaled`	logical. For `predictWEV_RT`. Whether the computed density should be scaled to integrate to one (additional column `densscaled`). Otherwise the output contains only the defective density (i.e. its integral is equal to the probability of a response and not 1). If `TRUE`, the argument `DistConf` should be given, if available. Default: `FALSE`.
`DistConf`	`NULL` or `data.frame`. A `data.frame` or `matrix` with column names, giving the distribution of response and rating choices for different conditions and stimulus categories in the form of the output of `predictWEV_Conf`. It is only necessary, if `scaled=TRUE`, because these probabilities are used for scaling. If `scaled=TRUE` and `DistConf=NULL`, it will be computed with the function `predictWEV_Conf`, which takes some time and the function will throw a message. Default: `NULL`

Details

The function `predictWEV_Conf` consists merely of an integration of the response time density, `ddynaViTE` and `d2DSD`, over the response time in a reasonable interval (`t0` to `maxrt`). The function `predictWEV_RT` wraps these density functions to a parameter set input and a data.frame output. For the argument `paramDf`, the output of the fitting function `fitRTConf` with the respective model may be used.

Value

`predictWEV_Conf` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `p`, `info`, `err`. `p` is the predicted probability of a response and `rating`, given the stimulus category and condition. `info` and `err` refer to the respective outputs of the integration routine used for the computation. `predictWEV_RT` returns a `data.frame/tibble` with columns: `condition`, `stimulus`, `response`, `rating`, `correct`, `rt` and `dens` (and `densscaled`, if `scaled=TRUE`).

Note

Different parameters for different conditions are only allowed for drift rate `v`, drift rate variability `sv`, and process variability `s`. Otherwise, `s` is not required in `paramDf` but set to 1 by default. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Pleskac, T. J., & Busemeyer, J. R. (2010). Two-Stage Dynamic Signal Detection: A Theory of Choice, Decision Time, and Confidence, *Psychological Review*, 117(3), 864-901. doi:10.1037/a0019737

Examples

```
# Examples for "dynWEV" model (equivalent applicable for "2DSD" model (with less parameters))
# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=2.5,v1=0.5, v2=1, t0=0.1,z=0.55,
                      sz=0,sv=0.2, st0=0,  tau=3, w=0.3,
                      theta1=0.8, svis=0.5, sigvis=0.8)

# 2. Predict discrete Choice x Confidence distribution:
preds_Conf <- predictWEV_Conf(paramDf, "dynWEV", maxrt = 15)
head(preds_Conf)

  # To set simult_conf=TRUE makes a minor difference in the discrete distribution,
  # because we integrate over response times (we just adapt maxrt for comparison)
 preds_Conf2 <- predictWEV_Conf(paramDf, "dynWEV", simult_conf = TRUE, maxrt = 15+paramDf$tau)
  summary(preds_Conf$p-preds_Conf2$p) # difference in predicted probabilities


# 3. Compute RT density
preds_RT <- predictWEV_RT(paramDf, "dynWEV", maxrt=4, subdivisions=200) #(scaled=FALSE)
# same output with scaled density column:
preds_RT <- predictWEV_RT(paramDf, "dynWEV", maxrt=4, subdivisions=200,
                         scaled=TRUE, DistConf = preds_Conf)
head(preds_RT)

  # produces a warning, if scaled=TRUE and DistConf missing
  preds_RT <- predictWEV_RT(paramDf, "dynWEV", maxrt=4, subdivisions=200,
                           scaled=TRUE)



  # Example of visualization
  library(ggplot2)
  preds_Conf$rating <- factor(preds_Conf$rating, labels=c("unsure", "sure"))
  preds_RT$rating <- factor(preds_RT$rating, labels=c("unsure", "sure"))
  ggplot(preds_Conf, aes(x=interaction(rating, response), y=p))+
    geom_bar(stat="identity")+
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")
  ggplot(preds_RT, aes(x=rt, color=interaction(rating, response), y=dens))+
    geom_line(stat="identity")+
```

```
    facet_grid(cols=vars(stimulus), rows=vars(condition), labeller = "label_both")+
    theme(legend.position = "bottom")
  ggplot(aggregate(densscaled~rt+correct+rating+condition, preds_RT, mean),
         aes(x=rt, color=rating, y=densscaled))+
    geom_line(stat="identity")+
    facet_grid(cols=vars(condition), rows=vars(correct), labeller = "label_both")+
    theme(legend.position = "bottom")

# Use PDFtoQuantiles to get predicted RT quantiles
head(PDFtoQuantiles(preds_RT, scaled = FALSE))
```

`QuantModelComparison` *Quantitative Model Comparison*

Description

`subject_modelweights` computes the model weights (as probabilities) of individual subjects based on an information criterion (BIC, AIC, or AICc). `group_BMS` performs a Bayesian model comparison based on marginal likelihoods (alias model evidence), given for different models across different subject on a group level using a fixed effects model and a random effects model on the distribution of model probabilities (see Rigoux et al., 2014 and Details section). `group_BMS_fits` is a wrapper for `group_BMS` that can be used with the output of `fitRTConfModels`, i.e. a data frame with information criteria for different models and subjects, using a information criterion to approximate the model evidence.

Usage

```
subject_modelweights(fits, measure = "BIC")

group_BMS_fits(fits, measure = "BIC", opts = list(), alpha0 = NULL)

group_BMS(mlp, opts = list(), alpha0 = NULL)
```

Arguments

`fits`	a data frame as returned by `fitRTConfModels`. Should contain a column `model`indicating the model name, a column `subject` (alternatively `sbj` or `participant`) indicating the grouping structure of the data, and a column with the name given by the `measure` argument containing the values of the information criterion that should be used to approximate model evidence.
`measure`	the name of the column indicating the information criterion to approximate model evidence. For outputs of `fitRTConfModels`, the available measures are 'BIC', 'AIC', and 'AICc'. Any other approximation for the model evidence may be used, the measure is transferred to log model evidence by taking -measure/2.
`opts`	a list with options for the iteration algorithm to estimate the parameter of the Dirichlet distribution. Following values may be provided:

- `maxiter` the maximum number of iterations (Default: 200)
- `tol` the tolerance for changes in the free energy approximation to stop the algorithm, if abs(FE(i+1)-FE(i))<tol the algorithm is stopped (Default: 1e-4)
- `eps` The number to substitute values of 0 in calls to log (Default: 1e-32)

`alpha0`	a positive numeric vector representing the parameter of a Dirichlet distribution used as prior over model probabilities. The length should be equal to `nrow(mlp)` for `group_BMS`, and equal to the number of unique names in the `model` column of `fits` for `group_BMS_fits`.
`mlp`	a matrix containing the logarithm of marginal probabilities (i.e. log model evidence) with N columns representing individuals (or any other grouping structure) and K rows representing the models.

Details

This set of function can be used for model comparisons on a group level when the models were not fitted hierarchical but by fitting the models independently to different subgroups (e.g. data from different subjects).

The function `subject_modelweights` computes the model weights for each subject separately to inspect predominant models but also heterogeneity within the population. The functions `group_BMS` and `group_BMS_fits` can be used for a Bayesian model selection on the group level following the approach of Rigoux et al. (2014). The approach compares three different models for the generative structure of the data and gives estimates for model probabilities for the fixed and random effects models. The **fixed effects model** assumes that there is a single model that generated the data of all subjects. Thus, model weights may be computed directly by multiplying the model weights computed on a subject-level. This model is formulated in a Bayesian way using a Multinomial distribution over the models as prior with some prior parameter alpha0 giving the prior model weights. This is updated according to the marginal model likelihoods resulting in a single poterior vector of model probabilities, reported in the column `fx_prob` of the `model_weights` data frame. The random effects model assumes that there is a vector of model probabilities and each subject may generated by a different model, each drawn from a Multinomial distribution. The Bayesian prior on this vector of model probabilities is given by a Dirichlet distribution with some parameter alpha0. The function uses a variational technique to approximate the alpha parameter of the posterior Dirichlet distribution. Within this framework several statistics may be used for model selection. The `model_weights` data frame reports the posterior `alpha` parameter, as well as the posterior mean `r` of the corresponding dirichlet distribution. The exceedance probability `ep` represents the probability that given a random sample from the Dirichlet distribution the probability of the model is greater than all other probailities. Finally, the protected exceedance probability (`pep`) is a scaled version of the `ep` multiplying the `ep` by one minus the Bayesian omnibus risk (BOR). The Bayesian omnibus risk is the posterior probability of the **null model** against the random effects model. The **null model** assumes that all models are generating the subjects' data with equal probability and results from taking the limit of alpha0 towards infinity. The Bayesian omnibus risk is reported in the `summary_stats` together with the free energy approximation of the null, the fixed effects, and the random effects models.

Value

`subject_modelweights` returns a data frame of subject-wise model probabilities with rows for each subject and columns for the models given by name and one column for the subject ID as given in the input. `group_BMS` and `group_BMS_fits` return a list with two entries:

- `model_weights`: a matrix with rows for each model (row names indicate the model names for `group_BMS_fits` and for `group_BMS` if row names are available in `mlp`), and following columns: `alpha` (the alpha parameter of the Dirichlet posterior over model probabilities in the population), `r` (the mean probabilities of each model in the population), `ep` and `pep` (exceedance and protected exceedance probabilities for each model), and `fx_prob` (the posterior model probabilities if a fixed true model is assumed in the population).
- `summary_stats`: a vector giving statistics for the Bayesian model comparison that may be used for other analyses: Bayesian omnibus risks: `bor` (random effects model against the null model), `bor_fixed` (fixed effects model against the null model), and `bor_re_fixed` (random effects model against the fixed effects model), and estimations of the Free Energy of the Dirichlet distribution `FE` (random effects model), `FE0` (null model), and `FEfixed` (fixed effects model)

Author(s)

Sebastian Hellmann.

References

Rigoux, L., Stephan, K. E., Friston, K. J., & Daunizeau, J. (2014). Bayesian model selection for group studies - revisited. *NeuroImage*, 84, 971–985. doi: 10.1016/j.neuroimage.2013.08.065

Examples

```
# Define a data frame with information criteria from model fits
# (this is a sub-data.frame from an output of fitRTConfModels with
# 8 subjects, three models and rounded information criteria)
fits <- data.frame(
  participant = rep(1:8, each=3),
  model = rep(c("dynaViTE", "2DSD", "PCRMt"), 8),
  BIC = c(5318, 5665, 1659, 3856, 5508, 3982, 3950, 3998,
          4114, 4216, 4314, 4419, 3170, 3489, 3256, 1950,
          1934, 2051, 3194, 3317, 3359, 9656, 10161, 4024),
  AIC = c(5211, 5577, 1577, 3750, 5420, 3899, 3843, 3911,
          4031, 4109, 4226, 4337, 3063, 3401, 3173, 1844,
          1847, 1969, 3087, 3229, 3277, 9549, 10074, 3942),
  AICc = c(5212, 5578, 1577, 3751, 5421, 3900, 3844, 3911,
           4032, 4110, 4227, 4337, 3064, 3402, 3174, 1845,
           1848, 1970, 3088, 3230, 3277, 9550, 10074, 3942))
# Compute subject-wise model probabitilities based on different ICs
subject_modelweights(fits, measure = "BIC")
subject_modelweights(fits, measure = "AIC")
subject_modelweights(fits, measure = "AICc")
# Conduct group-level Bayesian model selection based on BIC
group_BMS_fits(fits, measure="BIC")


## General group-level Bayesian model selection based on any marginal log-probabilities
```

```
# Compute marginal log-likelihood based on BIC from fits
mlp <- matrix(NA, ncol=8, nrow=3)
for (i in 1:8) mlp[,i] <- fits[(i-1)*3 + 1:3, "BIC"]
mlp <- - mlp/(2)
rownames(mlp) <- c("dynaViTE", "2DSD", "PCRMt")
# conduct group BMS:
group_BMS(mlp)
```

`RaceModels` *Independent and partially anti-correlated Race Model for Decision Confidence*

Description

Probability densities and random number generators for response times, decisions and confidence judgments in the independent Race Model (`dIRM`/`rIRM`) or partially (anti-)correlated Race Model (`dPCRM`/`rPCRM`), i.e. the probability of a given response (response: winning accumulator (1 or 2)) at a given time (rt) and the confidence measure in the interval between `th1` and `th2` (Hellmann et al., 2023). The definition of the confidence measure depends on the argument `time_scaled` (see Details). The computations are based on Moreno-Bote (2010). The parameters for the models are `mu1` and `mu2` for the drift rates, `a`, `b` for the upper thresholds of the two accumulators and `s` for the incremental standard deviation of the processes and `t0` and `st0` for the minimum and range of uniformly distributed non-decision times (including encoding and motor time). For the computation of confidence judgments, the parameters `th1` and `th2` for the lower and upper bound of the interval for confidence measure and if `time_scaled` is `TRUE` the weight parameters `wx`, `wrt`, `wint` for the computation of the confidence measure are required (see Details).

Usage

```
dIRM(rt, response = 1, mu1, mu2, a, b, th1, th2, wx = 1, wrt = 0,
  wint = 0, t0 = 0, st0 = 0, s1 = 1, s2 = 1, s = NULL,
  time_scaled = TRUE, precision = 6, step_width = NULL)

dIRM2(rt, response = 1, mu1, mu2, a, b, th1, th2, wx = 1, wrt = 0,
  wint = 0, t0 = 0, st0 = 0, s1 = 1, s2 = 1, smu1 = 0, smu2 = 0,
  sza = 0, szb = 0, s = NULL, time_scaled = TRUE, precision = 6,
  step_width = NULL)

dIRM3(rt, response = 1, mu1, mu2, a, b, th1, th2, wx = 1, wrt = 0,
  wint = 0, t0 = 0, st0 = 0, s1 = 1, s2 = 1, smu1 = 0, smu2 = 0,
  s = NULL, time_scaled = TRUE, precision = 6, step_width = NULL)

dPCRM(rt, response = 1, mu1, mu2, a, b, th1, th2, wx = 1, wrt = 0,
  wint = 0, t0 = 0, st0 = 0, s1 = 1, s2 = 1, s = NULL,
  time_scaled = TRUE, precision = 6, step_width = NULL)

rIRM(n, mu1, mu2, a, b, wx = 1, wrt = 0, wint = 0, t0 = 0, st0 = 0,
```

```
  s1 = 1, s2 = 1, s = NULL, smu1 = 0, smu2 = 0, sza = 0, szb = 0,
  time_scaled = TRUE, delta = 0.01, maxrt = 15)

rPCRM(n, mu1, mu2, a, b, wx = 1, wrt = 0, wint = 0, t0 = 0, st0 = 0,
  s1 = 1, s2 = 1, s = NULL, smu1 = 0, smu2 = 0, sza = 0, szb = 0,
  time_scaled = TRUE, delta = 0.01, maxrt = 15)
```

Arguments

`rt`	a numeric vector of RTs. For convenience also a `data.frame` with columns `rt` and `response` is possible.
`response`	numeric vector with values in `c(1, 2)`, giving the accumulator that hit its boundary first.
`mu1`	numeric. Drift rate for the first accumulator
`mu2`	numeric. Drift rate for the second accumulator
`a`	positive numeric. Distance from starting point to boundary of the first accumulator.
`b`	positive numeric. Distance from starting point to boundary of the second accumulator.
`th1`	numeric. Lower bound of interval range for the confidence measure.
`th2`	numeric. Upper bound of interval range for the confidence measure.
`wx`	numeric. Weight on losing accumulator for the computation of the confidence measure. (Used only if `time_scale=TRUE`, 1)
`wrt`	numeric. Weight on reaction time for the computation of the confidence measure. (Used only if `time_scale=TRUE`, Default 0)
`wint`	numeric. Weight on the interaction of losing accumulator and reaction time for the computation of the confidence measure. (Used only if `time_scale=TRUE`, Default 0)
`t0`	numeric. Lower bound of non-decision time component in observable response times. Range: `t0>=0`. Default: 0.
`st0`	numeric. Range of a uniform distribution for non-decision time. Range: `st0>=0`. Default: 0.
`s1`	numeric. Diffusion constant of the first accumulator. Usually fixed to 1 for most purposes as it scales other parameters (see Details). Range: `s1>0`, Default: 1.
`s2`	numeric. Diffusion constant of the second accumulator. Usually fixed to 1 for most purposes as it scales other parameters (see Details). Range: `s2>0`, Default: 1.
`s`	numeric. Alternative way to specify diffusion constants, if both are assumed to be equal. If both (`s1`, `s2` and `s`) are given, only `s1` and `s2` will be used.
`time_scaled`	logical. Whether the confidence measure should be time-dependent. See Details.
`precision`	numerical scalar value. Precision of calculation. Determines the step size of integration w.r.t. `t0`. Represents the number of decimals precisely computed on average. Default is 6.

`step_width`	numeric. Alternative way to define the precision of integration w.r.t. `t0` by directly providing the step size for the integration.
`smu1`	numeric. Between-trial variability in the drift rate of the first accumulator.
`smu2`	numeric. Between-trial variability in the drift rate of the second accumulator.
`sza`	numeric. Between-trial variability in starting point of the first accumulator.
`szb`	numeric. Between-trial variability in starting point of the second accumulator.
`n`	integer. The number of samples generated.
`delta`	numeric. Discretization step size for simulations in the stochastic process
`maxrt`	numeric. Maximum decision time returned. If the simulation of the stochastic process exceeds a decision time of `maxrt`, the `response` will be set to 0 and the `maxrt` will be returned as `rt`.

Details

The parameters are formulated, s.t. both accumulators start at 0 and trigger a decision at their positive boundary `a` and `b` respectively. That means, both parameters have to be positive. Internally the computations adapt the parametrization of Moreno-Bote (2010).

`time_scaled` determines whether the confidence measure is computed in accordance to the Balance of Evidence hypothesis (if `time_scaled=FALSE`), i.e. if `response` is 1 at time T and X_2 is the second accumulator, then

$$conf = b - X_2(T)$$

. Otherwise, if `time_scaled=TRUE` (default), confidence is computed as linear combination of Balance of Evidence, decision time, and an interaction term, i.e.

$$conf = wx(b - X_2(T)) + wrt\frac{1}{\sqrt{T}} + wint\frac{b - X_2(T)}{\sqrt{T}}.$$

Usually the weights (`wx`, `wrt`, `wint`) should sum to 1, as the confidence thresholds (`th1` and `th2`) may be scaled according to their sum. If this is not the case, they will be scaled accordingly internally! Usually, this formula results in lower confidence when the reaction time is longer but the state of the second accumulator is held constant. It is based on the optimal decision confidence in Moreno-Bote (2010).

For convenience, the likelihood function allows that the first argument is a `data.frame` containing the information of the first and second argument in the columns (i.e., `rt` and `response`). Other columns (as well as passing `response` separately as argument) will be ignored.

The simulations are done by simulating normal variables in discretized steps until one process reaches the boundary. If no boundary is met within the maximum time, response is set to 0.

Value

`dIRM` and `dPCRM` return the numerical value of the probability density in a numerical vector of the same length as `rt`.

`rIRM` and `dPCRM` return a `data.frame` with four columns and `n` rows. Column names are `rt` (response time), `response` (1 or 2, indicating which accumulator hit its boundary first), `xl` (the final state of the loosing accumulator), and `conf` (the value of the confidence measure; not discretized!).

The race parameters (as well as `response`, `th1`, and `th2`) are recycled to the length of the result (either `rt` or `n`). In other words, the functions are completely vectorized for all parameters and even the response.

Note

Similarly to the drift diffusion models (like `ddiffusion` and `ddynaViTE`), `s1` and `s2` are scaling factors (`s1` scales: `mu1` and `a`, `s2` scales: `mu2` and `b`, and depending on response: if `response=2`, `s1` scales `th1`,`th2`,and `wrt`), otherwise `s2` is the scaling factor. It is sometimes assumed (Moreno-Bote, 2010), that both noise terms are equal, then they should definitely be fixed for fitting.

Author(s)

Sebastian Hellmann

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Moreno-Bote, R. (2010). Decision confidence and uncertainty in diffusion models with partially correlated neuronal integrators. Neural Computation, 22(7), 1786–1811. doi:10.1162/neco.2010.12-08-930

Examples

```
# Plot rt distribution ignoring confidence
curve(dPCRM(x, 1, mu1=0.5, mu2=-0.5, a=1, b=1, th1=-Inf, th2=Inf, t0=0.1), xlim=c(0,2.5))
curve(dPCRM(x, 2, mu1=0.5, mu2=-0.5, a=1, b=1, th1=-Inf, th2=Inf, t0=0.1), col="red", add=TRUE)
curve(dIRM(x, 1, mu1=0.5, mu2=-0.5, a=1, b=1, th1=-Inf, th2=Inf, t0=0.1), lty=2,add=TRUE)
curve(dIRM(x, 2, mu1=0.5, mu2=-0.5, a=1, b=1, th1=-Inf, th2=Inf, t0=0.1),
      col="red", lty=2, add=TRUE)
# t0 indicates minimal response time possible
abline(v=0.1)
## Following example may be equivalently used for the IRM model functions.
# Generate a random sample
df1 <- rPCRM(5000,  mu1=0.2, mu2=-0.2, a=1, b=1, t0=0.1,
            wx = 1) # Balance of Evidence
# Same RT and response distribution but different confidence distribution
df2 <- rPCRM(5000,  mu1=0.2, mu2=-0.2, a=1, b=1, t0=0.1,
             wint = 0.2, wrt=0.8)
head(df1)

# Compute density with rt and response as separate arguments
dPCRM(seq(0, 2, by =0.4), response= 2, mu1=0.2, mu2=-0.2, a=1, b=1, th1=0.5,
         th2=2, wx = 0.3, wint=0.4, wrt=0.1, t0=0.1)
# Compute density with rt and response in data.frame argument
df1 <- subset(df1, response !=0) # drop trials where no accumulation hit its boundary
dPCRM(df1[1:5,], mu1=0.2, mu2=-0.2, a=1, b=1, th1=0, th2=Inf, t0=0.1)
# s1 and s2 scale other decision relevant parameters
 s <- 2  # common (equal) standard deviation
dPCRM(df1[1:5,], mu1=0.2*s, mu2=-0.2*s, a=1*s, b=1*s, th1=0, th2=Inf, t0=0.1, s1=s, s2=s)
s1 <- 2  # different standard deviations
s2 <- 1.5
dPCRM(df1[1:5,], mu1=0.2*s1, mu2=-0.2*s2, a=1*s1, b=1*s2, th1=0, th2=Inf, t0=0.1, s1=s1, s2=s2)
```

```
# s1 and s2 scale also confidence parameters
df1[1:5,]$response <- 2   # set response to 2
# for confidence it is important to scale confidence parameters with
# the right variation parameter (the one of the loosing accumulator)
dPCRM(df1[1:5,], mu1=0.2, mu2=-0.2, a=1, b=1,
     th1=0.5, th2=2, wx = 0.3, wint=0.4, wrt=0.1, t0=0.1)
dPCRM(df1[1:5,], mu1=0.2*s1, mu2=-0.2*s2, a=1*s1, b=1*s2,
      th1=0.5, th2=2, wx = 0.3/s1, wint = 0.4/s1, wrt = 0.1, t0=0.1, s1=s1, s2=s2)
dPCRM(df1[1:5,], mu1=0.2*s1, mu2=-0.2*s2, a=1*s1, b=1*s2,
      th1=0.5*s1, th2=2*s1, wx = 0.3, wint = 0.4, wrt = 0.1*s1, t0=0.1, s1=s1, s2=s2)

two_samples <- rbind(cbind(df1, ws="BoE"),
                   cbind(df2, ws="RT"))
# drop not finished decision processes
two_samples <- two_samples[two_samples$response!=0,]
# no difference in RT distributions
boxplot(rt~ws+response, data=two_samples)
# but different confidence distributions
boxplot(conf~ws+response, data=two_samples)
if (requireNamespace("ggplot2", quietly = TRUE)) {
require(ggplot2)
ggplot(two_samples, aes(x=rt, y=conf))+
  stat_density_2d(aes(fill = after_stat(density)),
                  geom = "raster", contour = FALSE, h=c(0.3, 0.7)) +
  xlim(c(0.2, 1.3))+ ylim(c(0, 2.5))+
  facet_grid(cols=vars(ws), rows=vars(response), labeller = "label_both")
}
# Restricting to specific confidence region
df1 <- df1[df1$conf >0 & df1$conf <1,]
dPCRM(df1[1:5,], th1=0, th2=1,mu1=0.2, mu2=-0.2, a=1, b=1, t0=0.1,wx = 1 )
```

---

rLCA *Simulation of confidence ratings and RTs in leaky competing accumulator model*

Description

Simulates the decision responses, reaction times and state of the loosing accumulator together with a confidence measure in the leaky competing accumulator model. Optionally, there is a post-decisional accumulation period, where the processes continues.

Usage

```
rLCA(n, mu1, mu2, th1, th2, k = 0, beta = 0, SPV = 0, tau = 0,
  wx = 1, wrt = 0, wint = 0, t0 = 0, st0 = 0, pi = 0, sig = 1,
  time_scaled = TRUE, simult_conf = FALSE, delta = 0.01, maxrt = 15)
```

Arguments

`n`	integer. number of samples.
`mu1`	mean momentary evidence for alternative 1
`mu2`	mean momentary evidence for alternative 2
`th1`	decision threshold for alternative 1
`th2`	decision threshold for alternative 2
`k`	leakage (default: 0)
`beta`	inhibition (default: 0)
`SPV`	variation in starting points (default: 0)
`tau`	fixed post decisional accumulation period (default: 0)
`wx`	weight on balance of evidence in confidence measure (default: 1)
`wrt`	weight on RT in confidence measure (default: 0)
`wint`	weight on interaction of evidence and RT in confidence measure (default: 0)
`t0`	minimal non-decision time (default: 0)
`st0`	range of uniform distribution of non-decision time (default: 0)
`pi`	factor for input dependent noise of infinitesimal variance of processes (default: 0)
`sig`	input independent component of infinitesimal variance of processes (default: 1)
`time_scaled`	logical. Whether a time_scaled transformation for the confidence measure should be used.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision. If that is the case decision and confidence judgment are assumed to have happened subsequent before the response. Therefore `tau` is included in the response time. If the decision was reported before the confidence report, `simul_conf` should be `FALSE`.
`delta`	numerical. Size of steps for the discretized simulation (see details).
`maxrt`	numerical. Maximum reaction time to be simulated (see details). Default: 15.

Details

The simulation is done by simulating discretized steps until one process reaches the boundary with an update rule:

$$\delta X_i(t) = \max(0, X_i(t) + \delta_t((k-1)X_i(t) - \beta X_{j=i}(t) + \mu_i + \varepsilon_i(t)),$$

with $\varepsilon_i(t) \sim N(0, (\pi\mu_i)^2 + \sigma^2)$. If no boundary is met within the maximum time, response is set to 0. After the decision, the accumulation continues for a time period (tau), until the final state is used for the computation of confidence.

Value

Returns a `data.frame` with three columns and `n` rows. Column names are `rt` (response time), `response` (1 or 2, indicating which accumulator hit its boundary first), and `conf` (the value of the confidence measure; not discretized!).

Author(s)

Sebastian Hellmann.

Examples

```
# minimal arguments
simus<- rLCA(n=20, mu1=1, mu2=-0.5, th1=1, th2=0.8)
head(simus)

# specifying all relevant parameters
simus <- rLCA(n=1000, mu1 = 2.5, mu2=1, th1=1.5, th2=1.6,
               k=0.1, beta=0.1, SPV=0.2, tau=0.1,
               wx=0.8, wrt=0.2, wint=0, t0=0.2, st0=0.1,
               pi=0.2, sig=1)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  require(ggplot2)
  ggplot(simus, aes(x=rt, y=conf))+
    stat_density_2d(aes(fill = after_stat(density)), geom = "raster", contour = FALSE) +
    facet_wrap(~response)
}
boxplot(conf~response, data=simus)
```

`simulateRM` *Simulation of confidence ratings and RTs in race confidence models*

Description

Simulates the decision responses, reaction times and state of the loosing accumulator together with a discrete confidence judgment in the independent and partially anti-correlated race model (IRM and PCRM) (Hellmann et al., 2023), given specific parameter constellations. See RaceModels for more information about parameters. Also computes the Gamma rank correlation between the confidence ratings and condition (task difficulty), reaction times and accuracy in the simulated output. Basically, this function is a wrapper for `rIRM` and `rPCRM` for application in confidence experiments with manipulation of specific parameters. `rRM_Kiani` simulates a different version of race models, presented in Kiani et al. (2014), but without a confidence measure.

Usage

```
simulateRM(paramDf, n = 10000, model = "IRM", time_scaled = FALSE,
  gamma = FALSE, agg_simus = FALSE, stimulus = c(1, 2), delta = 0.01,
  maxrt = 15, seed = NULL)

rRM_Kiani(paramDf, n = 10000, time_scaled = FALSE, gamma = FALSE,
  agg_simus = FALSE, stimulus = c(1, 2), delta = 0.01, maxrt = 15,
  seed = NULL)
```

Arguments

`paramDf`	a list or data frame with one row. Column names should match the names of RaceModels parameter names (only `mu1` and `mu2` are not used in this context but replaced by the parameter `v`). For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `s` parameters are possible with `s1`, `s2`, `s3`,... with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....
`n`	integer. The number of samples (per condition and stimulus direction) generated. Total number of samples is `n*nConditions*length(stimulus)`.
`model`	character scalar. One of "IRM" or "PCRM". ("IRMt" and "PCRMt" will also be accepted. In that case, time_scaled is set to TRUE.)
`time_scaled`	logical. Whether a time_scaled transformation for the confidence measure should be used.
`gamma`	logical. If TRUE, the gamma correlation between confidence ratings, rt and accuracy is computed.
`agg_simus`	logical. Simulation is done on a trial basis with RTs outcome. If TRUE, the simulations will be aggregated over RTs to return only the distribution of response and confidence ratings. Default: FALSE.
`stimulus`	numeric vector. Either 1, 2 or c(1, 2) (default). Together with condition represents the experimental situation. In a binary decision task the presented stimulus belongs to one of two categories. In the default setting trials with both categories presented are simulated but one can choose to simulate only trials with the stimulus coming from one category (each associated with positive drift in one of two accumulators).
`delta`	numerical. Size of steps for the discretized simulation (see details).
`maxrt`	numerical. Maximum reaction time to be simulated (see details). Default: 15.
`seed`	numerical. Seeding for non-random data generation. (Also possible outside of the function.)

Details

The simulation is done by simulating normal variables in discretized steps until one process reaches the boundary. If no boundary is met within the maximum time, response is set to 0. The output of the fitting function `fitRTConf` with the respective model fits the argument `paramDf` for simulation. The Gamma coefficients are computed separately for correct/incorrect responses for the correlation of confidence ratings with condition and rt and separately for conditions for the correlation of accuracy and confidence. The resulting data frames in the output thus have two columns. One for the grouping variable and one for the Gamma coefficient.

Value

Depending on `gamma` and `agg_simus`.

If `gamma` is `FALSE`, returns a `data.frame` with columns: `condition`, `stimulus`, `response`, `correct`, `rt`, `conf` (the continuous confidence measure) and `rating` (the discrete confidence rating) or (if `agg_simus=TRUE`): `condition`, `stimulus`,`response`, `correct`, `rating` and `p` (for the probability of a response and rating, given the condition and stimulus).

If `gamma` is `TRUE`, returns a `list` with elements: `simus` (the simulated data frame) and `gamma`, which is again a `list` with elements `condition`, `rt` and `correct`, each a `tibble` with two columns (see details for more information).

Note

Different parameters for different conditions are only allowed for drift rate, `v`, and process variability, `s`. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Kiani, R., Corthell, L., & Shadlen, M.N. (2014) Choice certainty is informed by both evidence and decision time. Neuron, 84(6), 1329-1342. doi:10.1016/j.neuron.2014.12.015

Examples

```
# Examples for "PCRM" model (equivalent applicable for "IRM" model)
# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=2,b=2, v1=0.5, v2=1, t0=0.1,st0=0,
                      wx=0.6, wint=0.2, wrt=0.2,
                      theta1=4)

# 2. Simulate trials for both stimulus categories and all conditions (2)
simus <- simulateRM(paramDf, n=30,model="PCRM", time_scaled=TRUE)
head(simus)
# equivalent:
simus <- simulateRM(paramDf, model="PCRMt")

  library(ggplot2)
  simus <- simus[simus$response!=0,]
  simus$rating <- factor(simus$rating, labels=c("unsure", "sure"))
  ggplot(simus, aes(x=rt, group=interaction(correct, rating),
                    color=as.factor(correct), linetype=rating))+
    geom_density(linewidth=1.2)+
    facet_grid(rows=vars(condition), labeller = "label_both")


# automatically aggregate simulation distribution
# to get only accuracy x confidence rating distribution for
# all conditions
agg_simus <- simulateRM(paramDf, n = 20, model="PCRMt", agg_simus = TRUE)
head(agg_simus)
```

```
agg_simus$rating <- factor(agg_simus$rating, labels=c("unsure", "sure"))
library(ggplot2)
ggplot(agg_simus, aes(x=rating, group=correct, fill=as.factor(correct), y=p))+
  geom_bar(stat="identity", position="dodge")+
  facet_grid(cols=vars(condition), labeller = "label_both")



# Compute Gamma correlation coefficients between
# confidence and other behavioral measures
# output will be a list
simu_list <- simulateRM(paramDf, model="IRMt", gamma=TRUE, n=200)
simu_list
```

`simulateRTConf` *Simulation of confidence ratings and RTs in sequential sampling confidence models*

Description

Simulates the decision responses, reaction times and confidence measure together with a discrete confidence judgment for the sequential sampling confidence model specified by the argument `model`, given specific parameter constellations. This function is a wrapper that calls the respective functions for diffusion based models (dynaViTE and 2DSD: `simulateWEV`) and race models (IRM, PCRM, IRMt, and PCRMt: `simulateRM`. It also computes the Gamma rank correlation between the confidence ratings and condition (task difficulty), reaction times and accuracy in the simulated output.

Usage

```
simulateRTConf(paramDf, n = 10000, model = NULL, gamma = FALSE,
  agg_simus = FALSE, simult_conf = FALSE, stimulus = c(1, 2),
  delta = 0.01, maxrt = 15, seed = NULL)
```

Arguments

`paramDf`	a list or dataframe with one row with the required parameters.
`n`	integer. The number of samples (per condition and stimulus direction) generated. Total number of samples is `n*nConditions*length(stimulus)`.
`model`	character scalar. One of "dynaViTE", "dynWEV", "2DSD", "2DSDT", "IRM", "PCRM", "IRMt", or "PCRMt". Could also be passed as a column in the paramDf argument.
`gamma`	logical. If TRUE, the gamma correlation between confidence ratings, rt and accuracy is computed.

`agg_simus`	logical. Simulation is done on a trial basis with RTs outcome. If TRUE, the simulations will be aggregated over RTs to return only the distribution of response and confidence ratings. Default: FALSE.
`simult_conf`	logical. Whether in the experiment confidence was reported simultaneously with the decision. If that is the case decision and confidence judgment are assumed to have happened subsequent before the response. Therefore `tau` is included in the response time. If the decision was reported before the confidence report, `simul_conf` should be `FALSE`.
`stimulus`	numeric vector. Either 1, 2 or c(1, 2) (default). Together with condition represents the experimental situation. In a 2AFC task the presented stimulus belongs to one of two categories. In the default setting trials with both categories presented are simulated but one can choose to simulate only trials with the stimulus coming from one category.
`delta`	numerical. Size of steps for the discretized simulation.
`maxrt`	numerical. Maximum reaction time to be simulated. Default: 15.
`seed`	numerical. Seeding for non-random data generation. (Also possible outside of the function.)

Details

The output of the fitting function `fitRTConf` with the respective model fits the argument `paramDf` for simulation. The function calls the respective simulation function for diffusion based models, i.e. dynaViTE and 2DSD (`simulateWEV`) or race models, i.e. IRM(t) and PCRM(t), (`simulateRM`). See there for more information.

Simulation Method: The simulation is done by simulating normal variables in discretized steps until the processes reach the boundary. If no boundary is met within the maximum time, response is set to 0.

Gamma correlations: The Gamma coefficients are computed separately for correct/incorrect responses for the correlation of confidence ratings with condition and rt and separately for conditions for the correlation of accuracy and confidence. The resulting data frames in the output thus have two columns. One for the grouping variable and one for the Gamma coefficient.

Value

Depending on `gamma` and `agg_simus`.

If `gamma` is `FALSE`, returns a `data.frame` with columns: `condition`, `stimulus`, `response`, `correct`, `rt`, `conf` (the continuous confidence measure) and `rating` (the discrete confidence rating) or (if `agg_simus=TRUE`): `condition`, `stimulus`,`response`, `correct`, `rating` and `p` (for the probability of a response and rating, given the condition and stimulus).

If `gamma` is `TRUE`, returns a `list` with elements: `simus` (the simulated data frame) and `gamma`, which is again a `list` with elements `condition`, `rt` and `correct`, each a `tibble` with two columns (see details for more information).

Author(s)

Sebastian Hellmann.

Examples

```
# The function is particularly useful, when having a collection
# of parameter sets for different models (e.g. output by fitRTConfModels for
# more than one model).
library(dplyr)
# 1. Generate only one parameter set but for two different models
paramDf1 <- data.frame(model="dynWEV", a=1.5,v1=0.2, v2=1, t0=0.1,z=0.52,
                       sz=0.3,sv=0.4, st0=0,  tau=3, w=0.5,
                       theta1=1, svis=0.5, sigvis=0.8)
paramDf2 <- data.frame(model="PCRMt", a=2,b=2, v1=0.5, v2=1, t0=0.1,st0=0,
                       wx=0.6, wint=0.2, wrt=0.2, theta1=4)
paramDf <- full_join(paramDf1, paramDf2)
paramDf  # each model parameters sets hat its relevant parameters
# Split paramDf by model (maybe also other columns) and simulate data
simus <- paramDf %>% group_by(model) %>%
 reframe(simulateRTConf(cbind(cur_group(), pick(everything())), n=200, simult_conf = TRUE))
head(simus)
```

`simulateWEV` *Simulation of confidence ratings and RTs in dynWEV and 2DSD confidence models*

Description

Simulates the decision responses and reaction times together with a discrete confidence judgment in the dynaViTE model, the 2DSD model (Pleskac & Busemeyer, 2010) and the dynWEV model (Hellmann et al., 2023), given specific parameter constellations. See `ddynaViTE` and `d2DSD` for more information about parameters. Also computes the Gamma rank correlation between the confidence ratings and condition (task difficulty), reaction times and accuracy in the simulated output. Basically, this function is a wrapper for `rdynaViTE` and `r2DSD` for application in confidence experiments with manipulation of specific parameters.

Usage

```
simulateWEV(paramDf, n = 10000, model = "dynWEV", simult_conf = FALSE,
  gamma = FALSE, agg_simus = FALSE, stimulus = c(-1, 1), delta = 0.01,
  maxrt = 15, seed = NULL, process_results = FALSE)
```

Arguments

`paramDf`	a list or dataframe with one row. Column names should match the names of dynaViTE and 2DSD model specific parameter names. For different stimulus quality/mean drift rates, names should be `v1`, `v2`, `v3`,.... Different `sv` and/or `s` parameters are possible with `sv1`, `sv2`, `sv3`... (`s1`, `s2`, `s3`,... respectively) with equally many steps as for drift rates. Additionally, the confidence thresholds should be given by names with `thetaUpper1`, `thetaUpper2`,..., `thetaLower1`,... or, for symmetric thresholds only by `theta1`, `theta2`,....

`n`	integer. The number of samples (per condition and stimulus direction) generated. Total number of samples is `n*nConditions*length(stimulus)`.
`model`	character scalar. One of "dynaViTE", "dynWEV", or "2DSD".
`simult_conf`	logical. `TRUE`, if in the experiment confidence was reported simultaneously with the decision, as then decision and confidence judgment are assumed to have happened subsequent before response and tau is added to the simulated decision time. If `FALSE` returned response time will only be decision time plus non-judgment time component.
`gamma`	logical. If TRUE, the gamma correlation between confidence ratings, rt and accuracy is computed.
`agg_simus`	logical. Simulation is done on a trial basis with RTs outcome. If TRUE, the simulations will be aggregated over RTs to return only the distribution of response and confidence ratings. Default: FALSE.
`stimulus`	numeric vector. Either 1, -1 or c(-1, 1) (default). Together with condition represents the experimental situation. In a binary decision task the presented stimulus belongs to one of two categories. In the default setting trials with both categories presented are simulated but one can choose to simulate only trials with the stimulus coming from one category (1 for the category that is associated with positive drift in the decision process where "upper"/1 responses are considered correct and -1 correspondingly for negative drifts and "lower"/-1 correct decisions).
`delta`	numeric. Discretization steps for simulations with the stochastic process.
`maxrt`	numeric. Maximum reaction time returned. If the simulation of the stochastic process exceeds a rt of `maxrt`, the response will be set to 0 and `maxrt` will be returned as rt.
`seed`	numerical. Seeding for non-random data generation.
`process_results`	logical. Whether the output simulations should contain the final state of the decision (and visibility) process as additional column. Default is FALSE, meaning that no additional columns for the final process states are returned.

Details

Simulation of response and decision times is done by simulating normal variables in discretized steps until the lower or upper boundary is met (or the maximal rt is reached). Afterwards, a confidence measure is simulated according to the respective model.

The confidence outputs are then binned according to the given thresholds. The output of the fitting function `fitRTConf` with the respective model fits the argument `paramDf` for simulation. The Gamma coefficients are computed separately for correct/incorrect responses for the correlation of confidence ratings with condition and rt and separately for conditions for the correlation of accuracy and confidence. The resulting data frames in the output thus have two columns. One for the grouping variable and one for the Gamma coefficient.

Value

Depending on `gamma` and `agg_simus`.

If `gamma` is `FALSE`, returns a `data.frame` with columns: `condition`, `stimulus`, `response`, `correct`, `rt`, `conf` (the continuous confidence measure) and `rating` (the discrete confidence rating), and `dec` and `vis` (only if `process_results=TRUE`) for the final states of accumulators in the simulation or (if `agg_simus=TRUE`): `condition`, `stimulus,response`, `correct`, `rating` and `p` (for the probability of a response and rating, given the condition and stimulus).

If `gamma` is `TRUE`, returns a `list` with elements: `simus` (the simulated data frame) and `gamma`, which is again a `list` with elements `condition`, `rt` and `correct`, each a `tibble` with two columns (see details for more information).

Note

Different parameters for different conditions are only allowed for drift rate, `v`, drift rate variability, `sv` and diffusion constant `s`. All other parameters are used for all conditions.

Author(s)

Sebastian Hellmann.

References

Hellmann, S., Zehetleitner, M., & Rausch, M. (2023). Simultaneous modeling of choice, confidence and response time in visual perception. *Psychological Review* 2023 Mar 13. doi: 10.1037/rev0000411. Epub ahead of print. PMID: 36913292.

Examples

```
# Examples for "dynWEV" model (equivalent applicable
# for "2DSD" model (with less parameters))
# 1. Define some parameter set in a data.frame
paramDf <- data.frame(a=2.5,v1=0.1, v2=1, t0=0.1,z=0.55,
                      sz=0.3,sv=0.8, st0=0,  tau=3, w=0.1,
                      theta1=0.8, svis=0.5, sigvis=0.8)

# 2. Simulate trials for both stimulus categories and all conditions (2)
simus <- simulateWEV(paramDf, model="dynWEV")
head(simus)

  library(ggplot2)
  simus <- simus[simus$response!=0,]
  simus$rating <- factor(simus$rating, labels=c("unsure", "sure"))
  ggplot(simus, aes(x=rt, group=interaction(correct, rating),
                    color=as.factor(correct), linetype=rating))+
    geom_density(linewidth=1.2)+xlim(c(0,5))+
    facet_grid(rows=vars(condition), labeller = "label_both")


# automatically aggregate simulation distribution
# to get only accuracy x confidence rating distribution for
# all conditions
agg_simus <- simulateWEV(paramDf, model="dynWEV", agg_simus = TRUE)
head(agg_simus)
```

```
agg_simus$rating <- factor(agg_simus$rating, labels=c("unsure", "sure"))
library(ggplot2)
ggplot(agg_simus, aes(x=rating, group=correct, fill=as.factor(correct), y=p))+
  geom_bar(stat="identity", position="dodge")+
  facet_grid(cols=vars(condition), labeller = "label_both")


# Compute Gamma correlation coefficients between
# confidence and other behavioral measures
# output will be a list
simu_list <- simulateWEV(paramDf,n = 400, model="dynWEV", gamma=TRUE)
simu_list
```

Index