Package 'rtdists'

September 9, 2026

Type Package

Title Response Time Distributions

Depends R (>= 3.0.0)

Suggests testthat, glba, knitr, rmarkdown, dplyr, tidyr, purrr, lattice, latticeExtra, binom, RWiener, WienR

Imports evd, msm, gsl, stats, Rcpp

LinkingTo Rcpp

SystemRequirements GNU GSL (libgsl-dev/gsl)

Description Provides response time distributions (density/PDF, distribution function/CDF, quantile function, and random generation): (a) Ratcliff diffusion model (Ratcliff & McKoon, 2008, <doi:10.1162/neco.2008.12-06-420>) based on C code by Andreas and Jochen Voss, (b) linear ballistic accumulator (LBA; Brown & Heathcote, 2008, <doi:10.1016/j.cogpsych.2007.12.002>) with different distributions underlying the drift rate, and (c) racing diffusion model (RDM; Tillman, Van Zandt, & Logan, 2020, <doi:10.3758/s13423-020-01719-6>).

URL https://github.com/rtdists/rtdists/

License GPL (>= 3)

BugReports https://github.com/rtdists/rtdists/issues

VignetteBuilder knitr

LazyData true

Version 0.12-0

Config/roxygen2/version 8.1.0

Encoding UTF-8

NeedsCompilation yes

Author Henrik Singmann [aut, cre] (ORCID: <https://orcid.org/0000-0002-4842-3657>), Scott Brown [aut],

Matthew Gretton [aut],
Andrew Heathcote [aut],
Kianté Fernandez [aut] (ORCID: <https://orcid.org/0000-0002-8493-880X>),
Andreas Voss [ctb],
Jochen Voss [ctb],
Andrew Terry [ctb],
Trisha Van Zandt [ctb],
Gabriel Tillman [ctb]

Maintainer Henrik Singmann <singmann@gmail.com>

Repository CRAN

Date/Publication 2026-09-09 08:00:02 UTC

Contents

Diffusion 2
LBA 10
LBA-race 16
RDM 20
rr98 23
single-LBA 25
single-RDM 28
speed_acc 30

Index **32**

`Diffusion` *The Ratcliff Diffusion Model*

Description

Density, distribution function, quantile function, and random generation for the Ratcliff diffusion model with following parameters: `a` (threshold separation), `z` (starting point), `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). **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 and that the parameterization of `z` and `sz` have changed compared to previous versions (now absolute and not relative).**

Usage

```
ddiffusion(
  rt,
  response = "upper",
  a,
  v,
```

```
  t0,
  z = 0.5 * a,
  d = 0,
  sz = 0,
  sv = 0,
  st0 = 0,
  s = 1,
  precision = 3,
  stop_on_error = FALSE
)

pdiffusion(
  rt,
  response = "upper",
  a,
  v,
  t0,
  z = 0.5 * a,
  d = 0,
  sz = 0,
  sv = 0,
  st0 = 0,
  s = 1,
  precision = 3,
  maxt = 20,
  stop_on_error = FALSE,
  use_precise = TRUE
)

qdiffusion(
  p,
  response = "upper",
  a,
  v,
  t0,
  z = 0.5 * a,
  d = 0,
  sz = 0,
  sv = 0,
  st0 = 0,
  s = 1,
  precision = 3,
  maxt = 20,
  interval = c(0, 10),
  scale_p = FALSE,
  scale_max = Inf,
  stop_on_error = FALSE,
  max_diff = 1e-04
```

```
)

rdiffusion(
  n,
  a,
  v,
  t0,
  z = 0.5 * a,
  d = 0,
  sz = 0,
  sv = 0,
  st0 = 0,
  s = 1,
  precision = 3,
  stop_on_error = TRUE,
  maxt = 20,
  interval = c(0, 10),
  method = c("fastdm", "qdiffusion")
)
```

Arguments

`rt`	a vector of RTs. Or for convenience also a `data.frame` with columns `rt` and `response` (such as returned from `rdiffusion` or `rLBA`). See examples.
`response`	character vector. Which response boundary should be tested? Possible values are `c("upper", "lower")`, possibly abbreviated and `"upper"` being the default. Alternatively, a numeric vector with values 1=lower and 2=upper. For convenience, `response` is converted via `as.numeric` also allowing factors (see examples). Ignored if the first argument is a `data.frame`.
`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: $0.1 < t0 < 0.5$
`z`	starting point. Indicator of an a priori bias in decision making. When the relative starting point `z` deviates from `0.5*a`, the amount of information necessary for a decision differs between response alternatives. Default is `0.5*a` (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` < 0.5. Default is 0.
`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. 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`, and `sv`. 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 roughly to the number of decimals of the predicted CDFs that are calculated accurately. Default is 3.
`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`).
`maxt`	maximum `rt` allowed, used to stop integration problems. Larger values lead to considerably longer calculation times.
`use_precise`	boolean. Should `pdiffusion` use the precise version for calculating the CDF? The default is `TRUE` which is highly recommended. Using `FALSE` (i.e., the imprecise version) is hardly any faster and produces clearly wrong results for most parameter settings.
`p`	vector of probabilities. Or for convenience also a `data.frame` with columns `p` and `response`. See examples.
`interval`	a vector containing the end-points of the interval to be searched for the desired quantiles (i.e., RTs) in `qdiffusion`. Default is `c(0, 10)`.
`scale_p`	logical. Should entered probabilities automatically be scaled by maximally predicted probability? Default is `FALSE`. Convenience argument for obtaining predicted quantiles. Can be slow as the maximally predicted probability is calculated individually for each `p`.
`scale_max`	numerical scalar. Value at which maximally predicted RT should be calculated if `scale_p` is `TRUE`.
`max_diff`	numeric. Maximum acceptable difference between desired and observed probability of the quantile function (`qdiffusion`).
`n`	desired number of observations. If 'length(n) > 1', the length is taken to be the number required (as in base R RNGs).
`method`	character. Experimentally implementation of an alternative way of generating random variates via the quantile function (`qdiffusion`) and random uniform value. For simple calls, the default method `"fastdm"` is dramatically faster.

Details

The Ratcliff diffusion model (Ratcliff, 1978) 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 introductions see Ratcliff and McKoon (2008), Voss, Rothermund, and Voss (2004), Voss, Nagler, and Lerche (2013), or Wagenmakers (2009).

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 trialwise parameters for each model parameter.

For convenience, all functions (with the exception of `rdiffusion`) allow that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt`/`p` and `response`). Other columns (as well as passing `response` separately argument) will be ignored. This allows, for example, to pass the `data.frame` generated by `rdiffusion` directly to `pdiffusion`. See examples.

Quantile Function: Due to the bivariate nature of the diffusion model, the diffusion processes reaching each response boundary only return the defective CDF that does not reach 1. Only the sum of the CDF for both boundaries reaches 1. Therefore, `qdiffusion` can only return quantiles/RTs for any accumulator up to the maximal probability of that accumulator's CDF. This can be obtained by evaluating the CDF at `Inf`.

As a convenience for the user, if `scale_p = TRUE` in the call to `qdiffusion` the desired probabilities are automatically scaled by the maximal probability for the corresponding response. Note that this can be slow as the maximal probability is calculated separately for each desired probability. See examples.

Also note that quantiles (i.e., predicted RTs) are obtained by numerically minimizing the absolute difference between desired probability and the value returned from `pdiffusion` using `optimize`. If the difference between the desired probability and probability corresponding to the returned quantile is above a certain threshold (currently 0.0001) no quantile is returned but `NA`. This can be either because the desired quantile is above the maximal probability for this accumulator or because the limits for the numerical integration are too small (default is `c(0, 10)`).

Value

`ddiffusion` gives the density, `pdiffusion` gives the distribution function, `qdiffusion` gives the quantile function (i.e., predicted RTs), and `rdiffusion` generates random response times and decisions (returning a `data.frame` with columns `rt` (numeric) and `response` (factor)).

The length of the result is determined by `n` for `rdiffusion`, equal to the length of `rt` for `ddiffusion` and `pdiffusion`, and equal to the length of `p` for `qdiffusion`.

The distribution parameters (as well as `response`) 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 used by, for example, Andreas Voss or Roger Ratcliff. 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 parametrization for the LBA code (where `t0` is also the lower bound).

The default diffusion constant `s` is 1 and not 0.1 as in most applications of Roger Ratcliff and others.

We have changed the parameterization of the start point z which is now the absolute start point in line with most published literature (it was the relative start point in previous versions of **rtdists**).

Author(s)

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.

References

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, 85(2), 59-108.

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

Voss, A., Rothermund, K., & Voss, J. (2004). Interpreting the parameters of the diffusion model: An empirical validation. *Memory & Cognition*. Vol 32(7), 32, 1206-1220.

Voss, A., Nagler, M., & Lerche, V. (2013). Diffusion Models in Experimental Psychology: A Practical Introduction. *Experimental Psychology*, 60(6), 385-402. doi:10.1027/1618-3169/a000218

Wagenmakers, E.-J., van der Maas, H. L. J., & Grasman, R. P. P. P. (2007). An EZ-diffusion model for response time and accuracy. *Psychonomic Bulletin & Review*, 14(1), 3-22.

Wagenmakers, E.-J. (2009). Methodological and empirical developments for the Ratcliff diffusion model of response times and accuracy. *European Journal of Cognitive Psychology*, 21(5), 641-671.

Examples

```
## identical calls (but different random values)
rt1 <- rdiffusion(500, a=1, v=2, t0=0.5)
head(rt1)
rt2 <- rdiffusion(500, a=1, v=2, t0=0.5, d=0, sz=0, sv=0, st0=0)
head(rt2)

# get density for random RTs (possible to specify arguments for pdiffusion in same way):
sum(log(ddiffusion(rt1$rt, rt1$response, a=1, v=2, t0=0.5)))  # response is factor
sum(log(ddiffusion(rt1$rt, as.numeric(rt1$response), a=1, v=2, t0=0.5))) # response is numeric
sum(log(ddiffusion(rt1$rt, as.character(rt1$response), a=1, v=2, t0=0.5))) # response is character
sum(log(ddiffusion(rt1, a=1, v=2, t0=0.5))) # response is data.frame


sum(log(ddiffusion(rt2$rt, rt2$response, a=1, v=2, t0=0.5)))

# can we recover the parameters?
ll_diffusion <- function(pars, rt, response)
{
  densities <- ddiffusion(rt, response=response,
                a=pars[1], v=pars[2], t0=pars[3],
                sz=pars[4],
                st0=pars[5], sv=pars[6])
  if (any(densities == 0)) return(1e6)
  return(-sum(log(densities)))
}
```

```
## Not run: )
start <- c(runif(2, 0.5, 3), 0.1, runif(3, 0, 0.5))
names(start) <- c("a", "v", "t0", "sz", "st0", "sv")
recov <- nlminb(start, ll_diffusion, lower = 0, rt=rt1$rt, response=rt1$response)
round(recov$par, 3)
#     a     v    t0    sz   st0    sv
# 1.019 1.879 0.496 0.000 0.000 0.389
## results of course depend on random seed for rdiffusion and runif

## End(Not run)


## Not run:
## replicate Table 1 from Wagenmakers et al. (2007) using rdiffusion:

n <- 1e5 # number of samples
# take parameter valeus from Table 2 and set s to 0.1
george <- rdiffusion(n, a = 0.12, v = 0.25, t0 = 0.3, s = 0.1)
rich   <- rdiffusion(n, a = 0.12, v = 0.25, t0 = 0.25, s = 0.1)
amy    <- rdiffusion(n, a = 0.08, v = 0.25, t0 = 0.3, s = 0.1)
mark   <- rdiffusion(n, a = 0.08, v = 0.25, t0 = 0.25, s = 0.1)

george$id <- "george"
rich$id <- "rich"
amy$id <- "amy"
mark$id <- "mark"

wag <- rbind(george, rich, amy, mark)
wag$id <- factor(wag$id, levels = c("george", "rich", "amy", "mark"))

opt <- options()
options(digits = 3)
aggregate(cbind(rt, as.numeric(response)-1) ~ id, wag, mean)
#       id    rt    V2
# 1 george 0.517 0.952
# 2   rich 0.467 0.953
# 3    amy 0.422 0.881
# 4   mark 0.372 0.882
options(digits = 1)
aggregate(rt ~ id, wag, var)
#       id    rt
# 1 george 0.024
# 2   rich 0.024
# 3    amy 0.009
# 4   mark 0.009
options(opt)

## End(Not run)


## plot density:
curve(ddiffusion(x, a=1, v=2, t0=0.5, response = "upper"),
     xlim=c(0,3), main="Density of upper responses", ylab="density", xlab="response time")
```

```
curve(ddiffusion(x, a=1, v=2, t0=0.5, st0=0.2, response = "upper"),
      add=TRUE, lty = 2)
legend("topright", legend=c("no", "yes"), title = "Starting Point Variability?", lty = 1:2)

# plot cdf:
curve(pdiffusion(x, a=1, v=2, t0=0.5, st0=0.2, response="u"),
     xlim = c(0, 3),ylim = c(0,1),
     ylab = "cumulative probability", xlab = "response time",
     main = "CDF of diffusion model with start point variability")
curve(pdiffusion(x, a=1, v=2, t0=0.5, st0=0.2, response="l"),
     add=TRUE, lty = 2)
legend("topleft", legend=c("upper", "lower"), title="response boundary", lty=1:2)

## Not run:
### qdiffusion can only return values up to maximal predicted probability:
(max_p <- pdiffusion(Inf, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, response="u"))
# [1] 0.87
# (Note that with the current integration routine for pdiffusion use Inf and not smaller values.)

qdiffusion(0.87, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, response="u")
# [1] 1.945802

qdiffusion(0.88, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, response="u")
# NA with warning.

# to get predicted quantiles, scale required quantiles by maximally predicted response rate:
qs <- c(.1, .3, .5, .7, .9)
qdiffusion(qs*max_p, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, response="u")

# or set scale_p to TRUE which scales automatically by maximum p
# (but can be slow as it calculates max_p for each probability separately)
qdiffusion(qs, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, response="u", scale_p = TRUE)


# qdiffusion also accepts a data.frame as first argument:
t3 <- data.frame(p = rep(c(0.05, 0.1, 0.87), 2), response = rep(c("upper", "lower"), each = 3))
#      p response
# 1 0.05    upper
# 2 0.10    upper
# 3 0.87    upper
# 4 0.05    lower
# 5 0.10    lower
# 6 0.87    lower
qdiffusion(t3, a=1, v=2, t0=0.5, st0=0.2, sz = 0.1, sv = 0.5, scale_p = TRUE)

## End(Not run)

## LBA and diffusion can be used interchangeably:
rt1 <- rLBA(500, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))
rt2 <- rdiffusion(500, a=1, v=2, t0=0.5)

# data can also be passed as data.frame (same is true for pLBA):
sum(log(dLBA(rt1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))
```

```
sum(log(dLBA(rt2, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))

sum(log(ddiffusion(rt1, a=1, v=2, t0=0.5)))
sum(log(ddiffusion(rt2, a=1, v=2, t0=0.5)))
```

LBA *The Linear Ballistic Accumulator (LBA)*

Description

Density, distribution function, quantile function, and random generation for the LBA model with the following parameters: `A` (upper value of starting point), `b` (response threshold), `t0` (non-decision time), and driftrate (`v`). All functions are available with different distributions underlying the drift rate: Normal (`norm`), Gamma (`gamma`), Frechet (`frechet`), and log normal (`lnorm`). The functions return their values conditional on the accumulator given in the response argument winning.

Usage

```
dLBA(
  rt,
  response,
  A,
  b,
  t0,
  ...,
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE
)

pLBA(
  rt,
  response,
  A,
  b,
  t0,
  ...,
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE
)

qLBA(
  p,
```

```
  response,
  A,
  b,
  t0,
  ...,
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE,
  interval = c(0, 10),
  scale_p = FALSE,
  scale_max = Inf
)

rLBA(
  n,
  A,
  b,
  t0,
  ...,
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE
)
```

Arguments

`rt`	vector of RTs. Or for convenience also a `data.frame` with columns `rt` and `response` (such as returned from `rLBA` or `rdiffusion`). See examples.
`response`	integer vector of winning accumulators/responses corresponding to the vector of RTs/p (i.e., used for specifying the response for a given RT/probability). Will be recycled if necessary. Cannot contain values larger than the number of accumulators. First response/accumulator must receive value 1, second 2, and so forth. For conmvenience, `response` is converted via `as.numeric` thereby allowing factors to be passed as well (such as returned from `rdiffusion`). Ignored if `rt` or `p` is a `data.frame`.
`A`	start point interval or evidence in accumulator before beginning of decision process. Start point varies from trial to trial in the interval [0, `A`] (uniform distribution). Average amount of evidence before evidence accumulation across trials is `A`/2.
`b`	response threshold. (`b` - `A`/2) is a measure of "response caution".
`t0`	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution).
`...`	two *named* drift rate parameters depending on `distribution` (e.g., `mean_v` and `sd_v` for `distribution=="norm"`). The parameters can either be given as a numeric vector or a list. If a numeric vector is passed each element of the vector

	corresponds to one accumulator. If a list is passed each list element corresponds to one accumulator allowing again trialwise driftrates. The shorter parameter will be recycled as necessary (and also the elements of the list to match the length of `rt`). See details.
`st0`	variability of non-decision time, such that `t0` is uniformly distributed between `t0` and `t0 + st0`. Default is 0. Can be trialwise, and will be recycled to length of `rt`.
`distribution`	character specifying the distribution of the finishing times of a single accumulator. Possible values are `c("norm", "gamma","frechet", "lnorm", "wald")`. The first four are LBA accumulators and refer to the distribution of the drift rate. `"wald"` is the accumulator of the racing diffusion model (see `RDM`). Default is `"norm"`.
`args.dist`	list of optional further arguments to the distribution functions (i.e., `posdrift` or `robust` for `distribution=="norm"`, see `single-LBA`).
`silent`	logical. Should the number of accumulators used be suppressed? Default is `FALSE` which prints the number of accumulators.
`p`	vector of probabilities. Or for convenience also a `data.frame` with columns `p` and `response`. See examples.
`interval`	a vector containing the end-points of the interval to be searched for the desired quantiles (i.e., RTs) in `qLBA`. Default is `c(0, 10)`.
`scale_p`	logical. Should entered probabilities automatically be scaled by maximally predicted probability? Default is `FALSE`. Convenience argument for obtaining predicted quantiles. Can be slow as the maximally predicted probability is calculated individually for each `p`.
`scale_max`	numerical scalar. Value at which maximally predicted RT should be calculated if `scale_p` is `TRUE`.
`n`	desired number of observations. If 'length(n) > 1', the length is taken to be the number required (as in base R RNGs).

Details

For convenience, all functions (with the exception of `rLBA`) allow that the first argument is a `data.frame` containing the information of the first and second argument in two columns (i.e., `rt`/`p` and `response`). Other columns will be ignored. This allows, for example, to pass the `data.frame` generated by `rLBA` directly to `pLBA`. See examples.

Parameters: The following arguments are allowed as `...` drift rate parameters:

- `mean_v,sd_v` mean and standard deviation of normal distribution for drift rate (`norm`). See `Normal`
- `shape_v,rate_v,scale_v` shape, rate, and scale of gamma (`gamma`) and scale and shape of Frechet (`frechet`) distributions for drift rate. See `GammaDist` or `frechet`. For Gamma, scale = 1/shape and shape = 1/scale.
- `meanlog_v,sdlog_v` mean and standard deviation of lognormal distribution on the log scale for drift rate (`lnorm`). See `Lognormal`.

As described above, the accumulator parameters can either be given as a numeric vector or a list. If a numeric vector is passed each element of the vector corresponds to one accumulator. If a list is passed each list element corresponds to one accumulator allowing trialwise driftrates. The shorter parameter will be recycled as necessary (and also the elements of the list to match the length of `rt`).

The other LBA parameters (i.e., `A`, `b`, and `t0`, with the exception of `st0`) can either be a single numeric vector (which will be recycled to reach `length(rt)` or `length(n)` for trialwise parameters) *or* a `list` of such vectors in which each list element corresponds to the parameters for this accumulator (i.e., the list needs to be of the same length as there are accumulators). Each list will also be recycled to reach `length(rt)` for trialwise parameters per accumulator.

To make the difference between both paragraphs clear: Whereas for the accumulators both a single vector or a list corresponds to different accumulators, only the latter is true for the other parameters. For those (i.e., `A`, `b`, and `t0`) a single vector always corresponds to trialwise values and a list must be used for accumulator wise values.

`st0` can only vary trialwise (via a vector). And it should be noted that `st0` not equal to zero will considerably slow done everything.

Quantile Function: Due to the bivariate nature of the LBA, single accumulators only return defective CDFs that do not reach 1. Only the sum of all accumulators reaches 1. Therefore, `qLBA` can only return quantiles/RTs for any accumulator up to the maximal probability of that accumulator's CDF. This can be obtained by evaluating the CDF at `Inf`.

As a convenience for the user, if `scale_p = TRUE` in the call to `qLBA` the desired probabilities are automatically scaled by the maximal probability for the corresponding response. Note that this can be slow as the maximal probability is calculated separately for each desired probability. See examples.

Also note that quantiles (i.e., predicted RTs) are obtained by numerically minimizing the absolute difference between desired probability and the value returned from `pLBA` using `optimize`. If the difference between the desired probability and probability corresponding to the returned quantile is above a certain threshold (currently 0.0001) no quantile is returned but `NA`. This can be either because the desired quantile is above the maximal probability for this accumulator or because the limits for the numerical integration are too small (default is `c(0, 10)`).

RNG: For random number generation at least one of the distribution parameters (i.e., `mean_v`, `sd_v`, `shape_v`, `scale_v`, `rate_v`, `meanlog_v`, and `sdlog_v`) should be of length > 1 to receive RTs from multiple responses. Shorter vectors are recycled as necessary.
Note that for random number generation from a normal distribution for the driftrate the number of returned samples may be less than the number of requested samples if `posdrifts==FALSE`.

Value

`dLBA` returns the density (PDF), `pLBA` returns the distribution function (CDF), `qLBA` returns the quantile/RT, `rLBA` return random response times and responses (in a `data.frame`).

The length of the result is determined by `n` for `rLBA`, equal to the length of `rt` for `dLBA` and `pLBA`, and equal to the length of `p` for `qLBA`.

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

Note

These are the top-level functions intended for end-users. To obtain the density and cumulative density the race functions are called for each response time with the corresponding winning accumulator as first accumulator (see LBA-race).

References

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive Psychology*, 57(3), 153-178. doi:10.1016/j.cogpsych.2007.12.002

Donkin, C., Averell, L., Brown, S., & Heathcote, A. (2009). Getting more from accuracy and response time data: Methods for fitting the linear ballistic accumulator. *Behavior Research Methods*, 41(4), 1095-1110. doi:10.3758/BRM.41.4.1095

Heathcote, A., & Love, J. (2012). Linear deterministic accumulator models of simple choice. *Frontiers in Psychology*, 3, 292. doi:10.3389/fpsyg.2012.00292

Examples

```
## generate random LBA data:
rt1 <- rLBA(500, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))
head(rt1)
prop.table(table(rt1$response))

# original parameters have 'high' log-likelihood:
sum(log(dLBA(rt1$rt, rt1$response, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))

# data can also be passed as data.frame (same is true for pLBA):
sum(log(dLBA(rt1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))

objective_fun <- function(par, rt, response, distribution = "norm") {
  # simple parameters
  spar <- par[!grepl("[12]$", names(par))]

  # distribution parameters:
  dist_par_names <- unique(sub("[12]$", "", grep("[12]$" ,names(par), value = TRUE)))
  dist_par <- vector("list", length = length(dist_par_names))
  names(dist_par) <- dist_par_names
  for (i in dist_par_names) dist_par[[i]] <- as.list(unname(par[grep(i, names(par))]))
  dist_par$sd_v <- c(1, dist_par$sd_v) # fix first sd to 1

  # get summed log-likelihood:
  d <- do.call(dLBA, args = c(rt=list(rt), response=list(response), spar, dist_par,
                              distribution=distribution, silent=TRUE))
  if (any(d < 0e-10)) return(1e6)
  else return(-sum(log(d)))
}

# gives same value as manual calculation above:
objective_fun(c(A=0.5, b=1, t0=0.5, mean_v1=2.4, mean_v2=1.6, sd_v2=1.2),
              rt=rt1$rt, response=rt1$response)

## Not run:
```

```
# can we recover the parameters?
# should be run several times with different random values of init_par
init_par <- runif(6)
init_par[2] <- sum(init_par[1:2]) # ensures b is larger than A
init_par[3] <- runif(1, 0, min(rt1$rt)) #ensures t0 is mot too large
names(init_par) <- c("A", "b", "t0", "mean_v1", "mean_v2",  "sd_v2")
nlminb(objective_fun, start = init_par, rt=rt1$rt, response=rt1$response, lower = 0)

## End(Not run)


# plot cdf (2 accumulators):
curve(pLBA(x, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2)),
     xlim = c(0, 2), ylim = c(0,1),
     ylab = "cumulative probability", xlab = "response time",
     main = "Defective CDFs of LBA")
curve(pLBA(x, response = 2, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2)),
     add=TRUE, lty = 2)
legend("topleft", legend=c("1", "2"), title="Response", lty=1:2)


# plot cdf (3 accumulators):
curve(pLBA(x, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6, 1.0), sd_v=c(1,1.2, 2.0)),
     xlim = c(0, 2), ylim = c(0,1),
     ylab = "cumulative probability", xlab = "response time",
     main = "Defective CDFs of LBA")
curve(pLBA(x, response = 2, A=0.5, b=1, t0 = 0.5,  mean_v=c(2.4, 1.6, 1.0), sd_v=c(1,1.2, 2.0)),
     add=TRUE, lty = 2)
curve(pLBA(x, response = 3, A=0.5, b=1, t0 = 0.5,  mean_v=c(2.4, 1.6, 1.0), sd_v=c(1,1.2, 2.0)),
     add=TRUE, lty = 3)
legend("topleft", legend=c("1", "2", "3"), title="Response", lty=1:2)


## qLBA can only return values up to maximal predicted probability:
(max_p <- pLBA(Inf, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2)))
# [1] 0.6604696

qLBA(0.66, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))
# 2.559532

qLBA(0.67, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))
# NA

# to get predicted quantiles, scale required quantiles by maximally predicted response rate:
qs <- c(.1, .3, .5, .7, .9)
qLBA(qs*max_p, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))

# or set scale_p to TRUE which scales automatically by maximum p
# (but can be slow as it calculates max_p for each probability separately)
qLBA(qs, response = 1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2), scale_p=TRUE)

# qLBA also accepts a data.frame as first argument. This is particularly useful
# for obtaining predicted RT quantiles for both responses at specific
```

```
# conditional quantiles. To do so, use scale_p=TRUE:
t <- data.frame(p = rep(c(.1, .5, .9), 2), response = rep(1:2, each = 3))
t$pred_rt <- qLBA(t, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2), scale_p=TRUE)
t
#     p response   pred_rt
# 1 0.1        1 0.6716606
# 2 0.5        1 0.7672016
# 3 0.9        1 0.9569697
# 4 0.1        2 0.6751672
# 5 0.5        2 0.7743184
# 6 0.9        2 0.9805128

## LBA and diffusion can be used interchangeably:
rt1 <- rLBA(500, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))
rt2 <- rdiffusion(500, a=1, v=2, t0=0.5)

# data can also be passed as data.frame (same is true for pLBA):
sum(log(dLBA(rt1, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))
sum(log(dLBA(rt2, A=0.5, b=1, t0 = 0.5, mean_v=c(2.4, 1.6), sd_v=c(1,1.2))))

sum(log(ddiffusion(rt1, a=1, v=2, t0=0.5)))
sum(log(ddiffusion(rt2, a=1, v=2, t0=0.5)))

### trial wise parameters work as expected (only since package version 0.9):
x1 <- dLBA(rt=c(1,1), response=c(1,2), A=1,b=list(c(1,3),c(2,4)),
           t0=0.1, mean_v=c(3,3), sd_v=c(1,1),distribution="norm")
x2a <- dLBA(rt=c(1), response=c(1), A=1,b=list(c(1),c(2)),
            t0=0.1,mean_v=c(3,3),sd_v=c(1,1),distribution="norm")
x2b <- dLBA(rt=c(1), response=c(2), A=1,b=list(c(3),c(4)),
            t0=0.1,mean_v=c(3,3),sd_v=c(1,1),distribution="norm")
all(x1 == c(x2a, x2b)) ## should be TRUE
```

LBA-race *LBA race functions: Likelihood for first accumulator to win.*

Description

n1PDF and n1CDF take RTs, the distribution functions of the LBA, and corresponding parameter values and put them throughout the race equations and return the likelihood for the first accumulator winning (hence n1) in a set of accumulators.

Usage

```
n1PDF(
  rt,
  A,
  b,
  t0,
  ...,
```

```
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE
)

n1CDF(
  rt,
  A,
  b,
  t0,
  ...,
  st0 = 0,
  distribution = c("norm", "gamma", "frechet", "lnorm", "wald"),
  args.dist = list(),
  silent = FALSE
)
```

Arguments

`rt`	a vector of RTs.
`A`, `b`, `t0`	LBA parameters, see `LBA`. Can either be a single numeric vector (which will be recycled to reach `length(rt)` for trialwise parameters) *or* a `list` of such vectors in which each list element corresponds to the parameters for this accumulator (i.e., the list needs to be of the same length as there are accumulators). Each list will also be recycled to reach `length(rt)` for trialwise parameters per accumulator.
`...`	two *named* drift rate parameters depending on `distribution` (e.g., `mean_v` and `sd_v` for `distribution=="norm"`). The parameters can either be given as a numeric vector or a list. If a numeric vector is passed each element of the vector corresponds to one accumulator. If a list is passed each list element corresponds to one accumulator allowing again trialwise driftrates. The shorter parameter will be recycled as necessary (and also the elements of the list to match the length of `rt`). See examples.
`st0`	parameter specifying the variability of `t0` (which varies uniformly from `t0` to `t0` + `st0`). Can be trialwise, and will be recycled to length of `rt`.
`distribution`	character specifying the distribution of the finishing times of a single accumulator. Possible values are `c("norm", "gamma","frechet", "lnorm", "wald")`. The first four are LBA accumulators and refer to the distribution of the drift rate. `"wald"` is the accumulator of the racing diffusion model (see `RDM`). Default is `"norm"`.
`args.dist`	list of optional further arguments to the distribution functions (i.e., `posdrift` or `robust` for `distribution=="norm"`).
`silent`	logical. Should the number of accumulators used be suppressed? Default is `FALSE` which prints the number of accumulators.

Details

For a set of N independent accumulators $i = 1...N$, the race likelihood for a given accumulator i is given by

$$L(\text{unit } i \text{ wins}) = f_i(t) \times \prod_{j<>i} [S_j(t)]$$

where $f(t)$ is the PDF (`dlba_...`) and $S_j(t) = 1 - F_j(t)$ is the survivor function, that is the complement of the CDF $F(t)$ (`plba_...`) at time t.

In other words, this is just the PDF/CDF for the winning accumulator at time t times the probability that no other accumulators have finished at time t.

See Also

For more user-friendly functions that return the PDF or CDF for the corresponding (and not first) accumulator winning see LBA.

Examples

```
## check random generated values against race functions:

## 1. Without st0:
r_lba <- rLBA(1e4, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1), sd_v=0.2)
x <- seq(0.5, 4, length.out = 100) # for plotting
# PDF
y <- n1PDF(x, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2) # PDF
hist(r_lba$rt[r_lba$response==1],probability = TRUE, breaks = "FD")
lines(x=x,y=y/mean(r_lba$response == 1))
# CDF
plot(ecdf(r_lba$rt[r_lba$response==1]))
y <- n1CDF(x, A=0.5, b=1, t0 = 0.5, st0 = 0, mean_v=c(1.2, 1.0), sd_v=0.2)
lines(x=x,y=y/mean(r_lba$response == 1), col = "red", lwd = 4.5, lty = 2)
# KS test
## Not run:
normalised_n1CDF = function(rt,...) n1CDF(rt,...)/n1CDF(rt=Inf,...)
ks.test(r_lba$rt[r_lba$response==1], normalised_n1CDF, A=0.5, b=1, t0 = 0.5,
        mean_v=c(1.2, 1.0), sd_v=0.2)

## End(Not run)

## Not run:
## Other examples (don't run to save time):

## 2. With st0 = 0.2:
r_lba <- rLBA(1e4, A=0.5, b=1, t0 = 0.5, st0 = 0.2, mean_v=c(1.2, 1), sd_v=0.2)
x <- seq(0.5, 4, length.out = 100) # for plotting
# PDF
y <- n1PDF(x, A=0.5, b=1, t0 = 0.5, st0 = 0.2, mean_v=c(1.2, 1.0), sd_v=0.2) # PDF
hist(r_lba$rt[r_lba$response==1],probability = TRUE, breaks = "FD")
lines(x=x,y=y/mean(r_lba$response == 1))
# CDF
```

```
plot(ecdf(r_lba$rt[r_lba$response==1]))
y <- n1CDF(x, A=0.5, b=1, t0 = 0.5, st0 = 0.2, mean_v=c(1.2, 1.0), sd_v=0.2)
lines(x=x,y=y/mean(r_lba$response == 1), col = "red", lwd = 4.5, lty = 2)
# KS test
normalised_n1CDF = function(rt,...) n1CDF(rt,...)/n1CDF(rt=Inf,...)
ks.test(r_lba$rt[r_lba$response==1], normalised_n1CDF, A=0.5, b=1, t0 = 0.5,
        st0 = 0.2, mean_v=c(1.2, 1.0), sd_v=0.2)


xx <- rLBA(10, A=0.5, b=1, t0 = 0.5, mean_v=1.2, sd_v=0.2)

# default uses normal distribution for drift rate:
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2)

# other distributions:
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, shape_v=c(1.2, 1), scale_v=c(0.2,0.3), distribution = "gamma")
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, shape_v=c(1.2, 1), scale_v=c(0.2,0.3), distribution = "frechet")
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, meanlog_v = c(0.5, 0.8), sdlog_v = 0.5, distribution = "lnorm")

# add st0:
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2, st0 = 0.4)


# use different A parameters for each RT:
n1PDF(xx$rt, A=runif(10, 0.4, 0.6),
      b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2)

# use different A parameters for each RT and each accumulator:
n1PDF(xx$rt, A=list(runif(10, 0.4, 0.6), runif(10, 0.2, 0.4)),
      b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2)


### vectorize drift rates:

# vector versus list:
v1 <- n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1.0), sd_v=0.2)
v2 <- n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=list(1.2, 1.0), sd_v=0.2)
identical(v1, v2)  # TRUE

# drift rate per trial:
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=list(rnorm(10, 1.2), rnorm(10, 1)), sd_v=0.2)

# combine list with vector:
n1PDF(xx$rt, A=0.5, b=1, t0 = 0.5, mean_v=list(rnorm(10, 1.2), rnorm(10, 1)), sd_v=c(0.2, 0.1))

# t0 per trial and accumulator:
n1PDF(xx$rt, A=0.5, b=1, t0 = c(0.5), mean_v=c(1.2, 1.0), sd_v=0.2)
n1PDF(xx$rt, A=0.5, b=1, t0 = c(0.5, 0.6), mean_v=c(1.2, 1.0), sd_v=0.2) # per trial only
n1PDF(xx$rt, A=0.5, b=1, t0 = list(0.5, 0.6), mean_v=c(1.2, 1.0), sd_v=0.2) # per drift rate only
n1PDF(xx$rt, A=0.5, b=1, t0 = list(c(0.4, 0.5), c(0.5, 0.6)), mean_v=c(1.2, 1.0), sd_v=0.2)

## End(Not run)
```

RDM *The Racing Diffusion Model (RDM)*

Description

Density, distribution, quantile, and random generation functions for the racing diffusion model (RDM) with any number of response alternatives.

Usage

```
dRDM(rt, response, A, b, t0, v, s = 1, st0 = 0, silent = FALSE)

pRDM(rt, response, A, b, t0, v, s = 1, st0 = 0, silent = FALSE)

qRDM(
  p,
  response,
  A,
  b,
  t0,
  v,
  s = 1,
  st0 = 0,
  silent = FALSE,
  interval = c(0, 10),
  scale_p = FALSE,
  scale_max = Inf
)

rRDM(n, A, b, t0, v, s = 1, st0 = 0, silent = FALSE)
```

Arguments

rt	vector of RTs. Or for convenience also a data.frame with columns rt and response (see details).
response	integer vector of winning accumulators/responses corresponding to the vector of RTs/p (i.e., used for specifying the response for a given RT/probability). Will be recycled if necessary. Cannot contain values larger than the number of accumulators. First response/accumulator must receive value 1, second 2, and so forth.
A	start point interval or evidence in accumulator before beginning of decision process. Start point varies from trial to trial in the interval [0, A] (uniform distribution). Average amount of evidence before evidence accumulation across trials is A/2.
b	response threshold. (b - A/2) is a measure of "response caution".

t0	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution).
v	drift rate. Mean rate of evidence accumulation within a trial. Needs to be positive, accumulators with a negative rate never finish.
s	within-trial standard deviation of the evidence accumulation process (i.e., the diffusion constant), scales A, b, and v. Needs to be fixed to a constant in most applications. Default is 1.
st0	variability of non-decision time, such that t0 is uniformly distributed between t0 and t0 + st0. Only available in rRDM.
silent	logical. Should the number of accumulators used be suppressed? Default is FALSE which prints the number of accumulators.
p	vector of probabilities. Or for convenience also a data.frame with columns p and response.
interval	a vector containing the end-points of the interval to be searched for the desired quantiles in qRDM. Default is c(0,10).
scale_p	logical. Should entered probabilities automatically be scaled by maximally predicted probability? Default is FALSE.
scale_max	numerical scalar. Value at which maximally predicted RT should be calculated if scale_p is TRUE.
n	desired number of observations. If 'length(n) > 1', the length is taken to be the number required (as in base R RNGs).

Details

The RDM is a race between independent diffusion processes, one per response alternative, that all start at a point drawn uniformly from [0, A] and drift towards their own threshold b at rate v with within-trial noise s. The first accumulator to reach its threshold determines both the response and the decision time, to which the non-decision time t0 is added. Unlike the LBA, the RDM has no between-trial variability in drift rate, the noise that produces both fast and slow errors is within-trial.

A, b, t0, and v can either be a single numeric vector (which will be recycled across the accumulators) or a list of numeric vectors with one element per accumulator. In the latter case, each element is recycled to the number of trials. This is the same convention as for dLBA.

The RDM is implemented as a race of Wald accumulators and shares the machinery of the LBA. dRDM, pRDM, qRDM, and rRDM are wrappers around dLBA, pLBA, qLBA, and rLBA called with distribution = "wald".

Value

dRDM returns the density (PDF), pRDM returns the distribution function (CDF), qRDM returns the quantile function, and rRDM returns random response times and responses (in a data.frame).

The density, distribution, and quantile functions are defective, that is, they are scaled by the probability of the corresponding response.

Note

These are the top-level functions intended for end-users. To obtain the density and distribution of a single accumulator, see single-RDM.

References

Tillman, G., Van Zandt, T., & Logan, G. D. (2020). Sequential sampling models without random between-trial variability: The racing diffusion model of speeded decision making. *Psychonomic Bulletin & Review*, 27(5), 911-936. doi:10.3758/s13423-020-01719-6

See Also

LBA for the linear ballistic accumulator and Diffusion for the Ratcliff diffusion model.

Examples

```
## generate random RDM data:
rt1 <- rRDM(500, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6))
head(rt1)
prop.table(table(rt1$response))

# original parameters have 'high' log-likelihood:
sum(log(dRDM(rt1$rt, rt1$response, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6))))

# data can also be passed as data.frame (same is true for pRDM):
sum(log(dRDM(rt1, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6))))

objective_fun <- function(par, rt, response) {
  d <- dRDM(rt, response, A=par["A"], b=par["b"], t0=par["t0"],
            v=c(par["v1"], par["v2"]), silent=TRUE)
  if (any(d < 0e-10)) return(1e6)
  else return(-sum(log(d)))
}

# gives same value as manual calculation above:
objective_fun(c(A=0.5, b=1, t0=0.5, v1=2.4, v2=1.6),
              rt=rt1$rt, response=rt1$response)

## Not run:
# can we recover the parameters?
# should be run several times with different random values of init_par
init_par <- runif(5)
init_par[2] <- sum(init_par[1:2]) # ensures b is larger than A
init_par[3] <- runif(1, 0, min(rt1$rt)) # ensures t0 is not too large
names(init_par) <- c("A", "b", "t0", "v1", "v2")
nlminb(objective_fun, start = init_par, rt=rt1$rt, response=rt1$response, lower = 0)

## End(Not run)

# plot defective cdfs (2 accumulators):
curve(pRDM(x, response = 1, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6), silent=TRUE),
      xlim = c(0, 2), ylim = c(0, 1),
      ylab = "cumulative probability", xlab = "response time",
      main = "Defective CDFs of the RDM")
curve(pRDM(x, response = 2, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6), silent=TRUE),
      add=TRUE, lty = 2)
legend("topleft", legend=c("1", "2"), title="Response", lty=1:2)
```

```
# quantiles of the first response:
qRDM(c(0.1, 0.5, 0.9), response = 1, A=0.5, b=1, t0 = 0.5, v=c(2.4, 1.6),
     scale_p = TRUE)
```

`rr98` *Ratcliff and Rouder (1998, Exp. 1) Luminance Discrimination Data*

Description

Responses and response times from an experiment in which three participants were asked to decide whether the overall brightness of pixel arrays displayed on a computer monitor was "high" or "low". In addition, instruction manipulated speed and accuracy between blocks.

Usage

```
rr98
```

Format

A `data.frame` with 24,358 obs. and 12 variables:

id participant id, factor with three levels

session session number, integer

block block number, integer

trial trial number within a block, integer

instruction factor with two levels: `"accuracy"` for blocks with accuracy instructions; `"speed"` for blocks with speed instruction

source factor with two levels: distribution strength was drawn from, `"dark"` and `"light"`

strength proportion of white to black pixels were varied by 33 equally spaced proportions from zero (all 1,024 pixels were black) to 1 (all 1,024 pixels were white). with 0 darkest and 32 lightest. Integer.

response factor with two levels: `"dark"` and `"light"`

response_num numeric response variable such that `1="dark"` and `2="light"`

correct boolean indicating whether or not `source==response`. (Does not seem to be used in the original analysis.)

rt response time in seconds

outlier boolean indicating whether or not the response was considered an outlier by Ratcliff and Rouder (1998), i.e., RTs outside of (200ms, 2500ms)

Details

The Experiment is described in the following by Ratcliff and Rouder (1998, pp. 349):

In Experiment 1, subjects were asked to decide whether the overall brightness of pixel arrays displayed on a computer monitor was "high" or "low". The brightness of a display was controlled by the proportion of the pixels that were white. For each trial, the proportion of white pixels was chosen from one of two distributions, a high distribution [i.e., light] or a low [i.e., dark] distribution, each with fixed mean and standard deviation. Feedback was given after each trial to tell the subject whether his or her decision had correctly indicated the distribution from which the stimulus had been chosen. Other than this feedback, a subject had no information about the distributions. Because the distributions overlapped substantially, a subject could not be highly accurate. A display with 50 from the high distribution on one trial and the low distribution on another.

Stimuli:

The stimulus display for Experiment 1 was a square that was 64 pixels on each side and subtended 3.8 degree of visual angle on a PC-VGA monitor. [...] In each square, 3,072 randomly chosen pixels were neutral gray, like the background, and the remaining 1,024 pixels were either black or white; the proportion of white to black pixels provided the brightness manipulation. There were 33 equally spaced proportions from zero (all 1,024 pixels were black) to 1 (all 1,024 pixels were white). The two distributions from which the bright and dark stimuli were chosen were centered at .375 (low brightness) and .625 (high brightness), and they each had a standard deviation of .1875.

Procedure: A subject's task was to decide, on each trial, from which distribution, high or low brightness in Experiment 1, the observed stimulus (stimuli) had been sampled. Subjects made their decision by pressing one of two response keys. On each trial, a 500-ms foreperiod, during which the display consisted solely of neutral gray, was followed by presentation of the stimulus; presentation was terminated by the subject's response. In Experiment 1, speed-versus-accuracy instructions were manipulated. For some blocks of trials, subjects were instructed to respond as quickly as possible, and a "too slow" message followed every response longer than 550 ms. For other blocks of trials, subjects were instructed to be as accurate as possible, and a "bad error" message followed incorrect responses to stimuli from the extreme ends of the distributions. Experiment 1 had ten 35-min sessions, and Experiments 2 and 3 had four sessions. In Experiment 1, subjects switched from emphasis on speed to emphasis on accuracy every 204 trials. Each session consisted of eight blocks of 102 trials per block, for a total of 8,160 trials per subject. Each session consisted of eight blocks of 102 trials, for a total of 3,264 trials per subject in each experiment. For all trials in each experiment, subjects were instructed to maintain a high level of accuracy while responding quickly, and an "error" message indicated incorrect responses. Responses were followed by a 300-ms blank interval, and the error message was displayed for 300 ms after the blank interval.

Note

The data is already prepared following Ratcliff and Rouder (1998) by removing the following trials:

- the first session for each participant
- the first 20 trials of each session
- the first trial of each block (each change in speed accuracy starts a new block)

To fully replicate the data used by Ratcliff and Rouder (1998) one only needs to remove the trials that are TRUE in column outlier (i.e., RTs outside of (200ms, 2500ms)). The full raw data is also available as part of this package, see:
system.file("extdata", "rr98-data", package = "rtdists") and system.file("extdata", "rr98-data.codes", package = "rtdists")

Source

Ratcliff, R., & Rouder, J. N. (1998). Modeling Response Times for Two-Choice Decisions. *Psychological Science*, 9(5), 347-356. http://doi.org/10.1111/1467-9280.00067

Examples

```
data(rr98)
rr98 <- rr98[!rr98$outlier,]  #remove outliers
head(rr98)
#  id session block trial instruction source strength response response_num correct   rt outlier
# 1 jf      2    1    21    accuracy   dark       8    dark           1    TRUE 0.801   FALSE
# 2 jf      2    1    22    accuracy   dark       7    dark           1    TRUE 0.680   FALSE
# 3 jf      2    1    23    accuracy  light      19   light           2    TRUE 0.694   FALSE
# 4 jf      2    1    24    accuracy   dark      21   light           2   FALSE 0.582   FALSE
# 5 jf      2    1    25    accuracy  light      19    dark           1   FALSE 0.925   FALSE
# 6 jf      2    1    26    accuracy   dark      10    dark           1    TRUE 0.605   FALSE

## See vignette for more examples.
```

---

single-LBA *Single accumulator of linear ballistic accumulator (LBA)*

Description

Density, distribution function, and random generation for a single accumulator of the LBA model with the following parameters: A (upper value of starting point), b (response threshold), t0 (non-decision time), and driftrate (v). All functions are available with different distributions underlying the drift rate: Normal (norm), Gamma (gamma), Frechet (frechet), and log normal (lnorm).

Usage

```
dlba_norm(rt, A, b, t0, mean_v, sd_v, posdrift = TRUE, robust = FALSE)

plba_norm(rt, A, b, t0, mean_v, sd_v, posdrift = TRUE, robust = FALSE)

rlba_norm(n, A, b, t0, mean_v, sd_v, st0 = 0, posdrift = TRUE)

dlba_gamma(rt, A, b, t0, shape_v, rate_v, scale_v)

plba_gamma(rt, A, b, t0, shape_v, rate_v, scale_v)
```

```
rlba_gamma(n, A, b, t0, shape_v, rate_v, scale_v, st0 = 0)

dlba_frechet(rt, A, b, t0, shape_v, scale_v)

plba_frechet(rt, A, b, t0, shape_v, scale_v)

rlba_frechet(n, A, b, t0, shape_v, scale_v, st0 = 0)

dlba_lnorm(rt, A, b, t0, meanlog_v, sdlog_v, robust = FALSE)

plba_lnorm(rt, A, b, t0, meanlog_v, sdlog_v, robust = FALSE)

rlba_lnorm(n, A, b, t0, meanlog_v, sdlog_v, st0 = 0)
```

Arguments

rt	a vector of RTs.
A	start point interval or evidence in accumulator before beginning of decision process. Start point varies from trial to trial in the interval [0, A] (uniform distribution). Average amount of evidence before evidence accumulation across trials is A/2.
b	response threshold. (b - A/2) is a measure of "response caution".
t0	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution).
mean_v, sd_v	mean and standard deviation of normal distribution for drift rate (norm). See Normal
posdrift	logical. Should driftrates be forced to be positive? Default is TRUE. (Uses truncated normal for random generation).
robust	logical. Should robust normal distributions be used for norm and lnorm? Can be helpful in rare cases but is approximately three times slower than the non-robust versions. Default is FALSE.
n	desired number of observations (scalar integer).
st0	variability of non-decision time, such that t0 is uniformly distributed between t0 and t0 + st0. Only available in random number generation functions rlba_.
shape_v, rate_v, scale_v	shape, rate, and scale of gamma (gamma) and scale and shape of Frechet (frechet) distributions for drift rate. See GammaDist or frechet. For Gamma, scale = 1/shape and shape = 1/scale.
meanlog_v, sdlog_v	mean and standard deviation of lognormal distribution on the log scale for drift rate (lnorm). See Lognormal.

Details

These functions are mainly for internal purposes. We do not recommend to use them. Use the high-level functions described in /link{LBA} instead.

Value

All functions starting with a `d` return the density (PDF), all functions starting with `p` return the distribution function (CDF), and all functions starting with `r` return random response times and responses (in a `matrix`).

Note

Density (i.e., `dlba_`), distribution (i.e., `plba_`), and random derivative (i.e., `rlba_`) functions are vectorized for all parameters (i.e., in case parameters are not of the same length as `rt`, parameters are recycled). Furthermore, the random derivative functions also accept a matrix of length `n` in which each column corresponds to a accumulator specific value (see `rLBA` for a more user-friendly way).

References

Brown, S. D., & Heathcote, A. (2008). The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive Psychology*, 57(3), 153-178. doi:10.1016/j.cogpsych.2007.12.002

Donkin, C., Averell, L., Brown, S., & Heathcote, A. (2009). Getting more from accuracy and response time data: Methods for fitting the linear ballistic accumulator. *Behavior Research Methods*, 41(4), 1095-1110. doi:10.3758/BRM.41.4.1095

Heathcote, A., & Love, J. (2012). Linear deterministic accumulator models of simple choice. *Frontiers in Psychology*, 3, 292. doi:10.3389/fpsyg.2012.00292

Examples

```
## random number generation using different distributions for v:
rlba_norm(10, A=0.5, b=1, t0 = 0.5, mean_v=c(1.2, 1), sd_v=c(0.2,0.3))
rlba_gamma(10, A=0.5, b=1, t0 = 0.5, shape_v=c(1.2, 1), scale_v=c(0.2,0.3))
rlba_frechet(10, A=0.5, b=1, t0 = 0.5, shape_v=c(1.2, 1), scale_v=c(0.2,0.3))
rlba_lnorm(10, A=0.5, b=1, t0 = 0.5, meanlog_v=c(1.2, 1), sdlog_v=c(0.2, 0.3))

# use somewhat plausible values for plotting:
A <- 0.2
b <- 0.5
t0 <- 0.3

# plot density:
curve(dlba_norm(x, A=A, b=b, t0=t0, mean_v = 1.0, sd_v = 0.5), ylim = c(0, 4),
    xlim=c(0,3), main="Density/PDF of LBA versions", ylab="density", xlab="response time")
curve(dlba_gamma(x, A=A, b=b, t0=t0, shape_v=1, scale_v=1), add=TRUE, lty = 2)
curve(dlba_frechet(x, A=A, b=b, t0=t0, shape_v=1,scale_v=1.0), add=TRUE, lty = 3)
curve(dlba_lnorm(x, A=A, b=b, t0=t0, meanlog_v = 0.5, sdlog_v = 0.5), add=TRUE, lty = 4)
legend("topright", legend=c("Normal", "Gamma", "Frechet", "Log-Normal"),
      title = expression("Distribution of"~~italic(v)), lty = 1:4)


# plot cdf:
curve(plba_norm(x, A=A, b=b, t0=t0, mean_v=1.0, sd_v=1.0),
      xlim = c(0, 3),ylim = c(0,1),
      ylab = "cumulative probability", xlab = "response time",
```

```
      main = "Distribution/CDF of LBA versions")
curve(plba_gamma(x, A=A, b=b, t0=t0, shape_v=1,scale_v=1), add=TRUE, lty = 2)
curve(plba_frechet(x, A=A, b=b, t0=t0, shape=1, scale=1), add=TRUE, lty = 3)
curve(plba_lnorm(x, A=A, b=b, t0=t0, meanlog_v=0.5, sdlog_v = 0.5), add=TRUE, lty = 4)
legend("bottomright", legend=c("Normal", "Gamma", "Frechet", "Log-Normal"),
       title = expression("Distribution of"~~italic(v)), lty = 1:4)
```

`single-RDM` *Single accumulator of the racing diffusion model (RDM)*

Description

Density, distribution, and random generation functions for a single Wald accumulator, that is, a diffusion process with within-trial variability in drift rate racing towards a single threshold with a uniformly distributed start point.

Usage

```
dwald(rt, A, b, t0, v, s = 1)

pwald(rt, A, b, t0, v, s = 1)

rwald(n, A, b, t0, v, s = 1, st0 = 0)
```

Arguments

`rt`	a vector of RTs.
`A`	start point interval or evidence in accumulator before beginning of decision process. Start point varies from trial to trial in the interval [0, `A`] (uniform distribution). Average amount of evidence before evidence accumulation across trials is `A`/2.
`b`	response threshold. (`b` - `A`/2) is a measure of "response caution".
`t0`	non-decision time or response time constant (in seconds). Lower bound for the duration of all non-decisional processes (encoding and response execution).
`v`	drift rate. Mean rate of evidence accumulation within a trial. Needs to be positive, accumulators with a negative rate never finish.
`s`	within-trial standard deviation of the evidence accumulation process (i.e., the diffusion constant), scales `A`, `b`, and `v`. Needs to be fixed to a constant in most applications. Default is 1.
`n`	desired number of observations (scalar integer).
`st0`	variability of non-decision time, such that `t0` is uniformly distributed between `t0` and `t0` + `st0`. Only available in random number generation function `rwald`.

Details

These functions are mainly for internal purposes. We do not recommend to use them. Use the high-level functions described in RDM instead.

Value

All functions starting with a `d` return the density (PDF), all functions starting with `p` return the distribution function (CDF), and all functions starting with `r` return random response times and responses (in a `matrix`).

Note

Density (i.e., `dwald`) and distribution (i.e., `pwald`) functions are vectorized for all parameters (i.e., in case parameters are not of the same length as `rt`, parameters are recycled). Furthermore, the random derivative function also accepts a matrix of length `n` in which each column corresponds to a accumulator specific value (see `rRDM` for a more user-friendly way).

References

Tillman, G., Van Zandt, T., & Logan, G. D. (2020). Sequential sampling models without random between-trial variability: The racing diffusion model of speeded decision making. *Psychonomic Bulletin & Review*, 27(5), 911-936. doi:10.3758/s13423-020-01719-6

Logan, G. D., Van Zandt, T., Verbruggen, F., & Wagenmakers, E.-J. (2014). On the ability to inhibit thought and action: General and special theories of an act of control. *Psychological Review*, 121(1), 66-95. doi:10.1037/a0035230

Examples

```
## random number generation for a single Wald accumulator:
rwald(10, A=0.5, b=1, t0 = 0.5, v=c(1.2, 1))

# use somewhat plausible values for plotting:
A <- 0.2
b <- 0.5
t0 <- 0.3

# plot density:
curve(dwald(x, A=A, b=b, t0=t0, v=2), ylim = c(0, 4), xlim = c(0, 3),
      main="Density/PDF of the Wald accumulator", ylab="density",
      xlab="response time")
curve(dwald(x, A=0, b=b, t0=t0, v=2), add=TRUE, lty = 2)
legend("topright", legend=c("0.2", "0"),
       title = expression("Start point variability"~~italic(A)), lty = 1:2)

# plot cdf:
curve(pwald(x, A=A, b=b, t0=t0, v=2), xlim = c(0, 3), ylim = c(0, 1),
      ylab = "cumulative probability", xlab = "response time",
      main = "Distribution/CDF of the Wald accumulator")
curve(pwald(x, A=0, b=b, t0=t0, v=2), add=TRUE, lty = 2)
legend("bottomright", legend=c("0.2", "0"),
       title = expression("Start point variability"~~italic(A)), lty = 1:2)
```

speed_acc *Speed-Accuracy Data from Wagenmakers, Ratcliff, Gomez, & McKoon (2008, Experiment 1)*

Description

Responses and response times from an experiment in which instruction manipulated speed and accuracy between blocks. This data was also analyzed by Heathcote and Love (2012) who were the first to use the 17 participants also included here.

Usage

```
speed_acc
```

Format

A `data.frame` with 31,522 obs. and 9 variables:

id participant id

block block number

condition `accuracy` for blocks with accuracy instructions; `speed` for blocks with speed instruction

stim unique identifier of stimulus, stimuli are nested in frequency conditions

stim_cat category of stimulus, either word or non-word

frequency "high frequency word", "low frequency word", "very low frequency word", or non-words derived from the first three categories

response `word`, `nonword`, or not interpretable response (`error`, i.e., pushed a button, but not the right one and also not the one next to the right button)

rt response time in seconds

censor boolean indicating whether or not a response should be eliminated prior to analysis; uninterpretable response, too fast response (<180 ms), too slow response (>3 sec)

Details

The data excludes the practice blocks but includes all trials. Variable `censor` can be used for excluding all trials also excluded from the papers using it namely uninterpretable response, too fast response (<180 ms), too slow response (>3 sec). Heathcote and Love (2012, p. 7) describe the data as follows:

We fit the LBA and LNR models to data from Wagenmaker et al.'s (2008) experiment one, where participants made decisions about whether a string of letters constituted a word. These lexical decisions were made about four types of stimuli, non-words (nw) and high-frequency (hf), low-frequency (lf), and very low-frequency (vlf) words. Participants made decisions either under speed or accuracy emphasis instructions in different experimental blocks. Accuracy blocks were preceded by the message "Try to respond accurately" and "ERROR" was displayed after each wrong response. Speed blocks were preceded by the message "Try to respond accurately" and "TOO SLOW" was displayed after each response slower than 0.75 s.We report analyses of data from 17 participants (31,412 data points) in their Experiment 1, including the 15 participants analyzed in Wagenmakers et al. (2008) and two extras (we thank Eric-Jan Wagenmakers for supplying this data).

Source

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & McKoon, G. (2008). A diffusion model account of criterion shifts in the lexical decision task. *Journal of Memory and Language*, 58(1), 140-159.

References

Heathcote, A., & Love, J. (2012). Linear deterministic accumulator models of simple choice. *Frontiers in Psychology: Cognitive Science*, 3, 292. doi:10.3389/fpsyg.2012.00292

Examples

```
data(speed_acc)
str(speed_acc)

# remove excluded trials:
speed_acc <- droplevels(speed_acc[!speed_acc$censor,])

# new factors for obtaining values as in Table 1, Wagenmakers et al. (2008, p. 152)
speed_acc$freq <- with(speed_acc,
                       factor(ifelse(stim_cat == "nonword", "nonword",
                                     as.character(frequency)),
                              levels = c("high", "low", "very_low", "nonword")))
# corr = correct (0 = correct, 1 = error)
speed_acc$corr <- with(speed_acc, 1-as.numeric(stim_cat == response))

str(speed_acc)

## aggregated RTs:
aggregate(rt ~ condition + freq + corr, speed_acc, mean)
## Error Rate:
aggregate(corr ~ condition + freq + corr, speed_acc, mean)
```

Index

∗ **dataset**
    rr98, 23
    speed_acc, 30

ddiffusion *(*Diffusion*)*, 2
Diffusion, 2, *22*
diffusion *(*Diffusion*)*, 2
dLBA, *21*
dLBA *(*LBA*)*, 10
dlba_frechet *(*single-LBA*)*, 25
dlba_gamma *(*single-LBA*)*, 25
dlba_lnorm *(*single-LBA*)*, 25
dlba_norm *(*single-LBA*)*, 25
dRDM *(*RDM*)*, 20
dwald *(*single-RDM*)*, 28

frechet, *12*, *26*

GammaDist, *12*, *26*

LBA, *6*, 10, *16–18*, *22*
LBA-race, 16
Lognormal, *12*, *26*

n1CDF *(*LBA-race*)*, 16
n1PDF *(*LBA-race*)*, 16
Normal, *12*, *26*

optimize, *6*, *13*

pdiffusion *(*Diffusion*)*, 2
pLBA, *21*
pLBA *(*LBA*)*, 10
plba_frechet *(*single-LBA*)*, 25
plba_gamma *(*single-LBA*)*, 25
plba_lnorm *(*single-LBA*)*, 25
plba_norm *(*single-LBA*)*, 25
pRDM *(*RDM*)*, 20
pwald *(*single-RDM*)*, 28

qdiffusion *(*Diffusion*)*, 2
qLBA, *21*
qLBA *(*LBA*)*, 10
qRDM *(*RDM*)*, 20

rdiffusion, *11*
rdiffusion *(*Diffusion*)*, 2
RDM, *12*, *17*, 20, *28*
rLBA, *4*, *21*, *27*
rLBA *(*LBA*)*, 10
rlba_frechet *(*single-LBA*)*, 25
rlba_gamma *(*single-LBA*)*, 25
rlba_lnorm *(*single-LBA*)*, 25
rlba_norm *(*single-LBA*)*, 25
rr98, 23
rRDM, *29*
rRDM *(*RDM*)*, 20
rwald *(*single-RDM*)*, 28

single-LBA, 25
single-RDM, 28
speed_acc, 30