Package 'ddModel'

May 8, 2026

Type Package

Title The Decision Diffusion Model

Version 0.2.9.0

Date 2025-08-07

Maintainer Yi-Shin Lin <yishinlin001@gmail.com>

Description

Provides functions for computing the density, distribution, and random generation of the Decision Diffusion model (DDM), a widely used cognitive model for analysing choice and response time data. The package allows model specification, including the ability to fix, constrain, or vary parameters across experimental conditions. While it does not include a built-in optimiser, it supports likelihood evaluation and can be integrated with external tools for parameter estimation. Functions for simulating synthetic datasets are also provided. This package is intended for researchers modelling speeded decision-making in behavioural and cognitive experiments. For more information, see Voss, Rothermund, and Voss (2004) <doi:10.3758/BF03196893>, Voss and Voss (2007) <doi:10.3758/BF03192967>, and Ratcliff and McKoon (2008) <doi:10.1162/neco.2008.12-06-420>.

License GPL (>= 2)

Imports ggdmcPrior, ggdmcModel, Rcpp (>= 1.0.7), methods

Depends R (>= 3.3.0)

LinkingTo Rcpp (>= 1.0.7), RcppArmadillo (>= 0.10.7.5.0), ggdmcHeaders

Suggests testthat

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Yi-Shin Lin [aut, cre]

Repository CRAN

Date/Publication 2025-09-02 20:50:24 UTC

Contents

ddm-class 2
dfastdm 3
simulate,ddm-method 5
validate_ddm_parameters 7

Index **10**

`ddm-class` *An S4 Class Representing a Decision Diffusion Model*

Description

The `ddm` class represents a complete Decision Diffusion Model (DDM) specification. It encapsulates a `model` objects from the **ggdmcModel** package, and optionally a population distribution created via the `BuildPrior` function from the **ggdmcPrior** package.

Usage

```
setDDM(model, population_distribution = NULL)
```

Arguments

`model`	A required `model` object defining the model specification.
`population_distribution`	Optional A population-level prior distribution, typically constructed with `BuildPrior`. This argument enables simulations of subject-level parameters. Default is `NULL`.

Details

The constructor function `setDDM` creates a `ddm` object by wrapping the model and optionally a population distribution into a single structure. This class is designed for compatibility with sampling and simulation functions within the **ggdmc** ecosystem.

Value

An object of S4 class `ddm`, which includes:

- `model`: the model specification
- `population_distribution`: the prior distribution (if provided)

Slots

`model` A `model` object from **ggdmcModel** package that defines the DDM structure, including parameter names, mapping, and data attributes.

`population_distribution` An optional prior distribution, typically created via `BuildPrior` from the **ggdmcPrior** package. This slot is primarily used for parameters or hierarchical modelling and can be `NULL`.

See Also

model-class, BuildPrior

---

dfastdm *Density, Distribution and Random Generation of the Decision Diffusion Model*

Description

A set of functions implementing the Decision Diffusion Model (DDM), providing: probability density (dfastdm), cumulative distribution (pfastdm), and random choices and response times generation (rfastdm).

Usage

```
dfastdm(rt_r, parameters_r, is_lower = TRUE)

pfastdm(rt_r, parameters_r, is_lower = TRUE)

rfastdm(
  n = 1L,
  parameters_r = as.numeric(c()),
  time_parameters_r = as.numeric(c()),
  debug = FALSE
)
```

Arguments

rt_r	A numeric vector of response times (in seconds)
parameters_r	A numeric vector of model parameters: **a** Boundary separation. **z** Starting point (bias), must lie between 0 anda. **v** Drift rate. **t0** Non-decision time (must be ≥ 0). **d** Difference in motor execution time between boundaries (e.g., left vs. right response). When d > 0, responses at the upper boundary are delayed by d seconds. Symmetric execution is assumed when d = 0 (Voss and Voss, 2007.) **sz** Inter-trial variability in starting point (optional) **sv** Inter-trial variability in drift rate (optional) **st0** Inter-trial variability in non-decision time (optional)
is_lower	Logical; if TRUE, compute probability for the lower boundary. Defaults to TRUE. Note: under standard DDM assumptions, a response at the upper boundary indicates a 'correct' choice.

n	Integer; number of random samples to generate (used in rfastdm).
time_parameters_r	Optional numeric vector specifying time resolution for simulation (used in rfastdm, which employs inverse transform sampling).
debug	Logical; if TRUE, return debugging information.

Details

These functions implement the DDM, which describes a one-dimensional evidence accumulation process between two absorbing boundaries, representing competing response alternatives. The implementation includes an extended version following the fast-dm framework, allowing for inter-trial variability in key parameters (e.g., drift rate, starting point, non-decision time).

The density and distribution functions are based on the g_minus and g_plus formulations (in fastdm source code), which were described in Equation A1–A4 in Voss, Rothermund, and Voss (2004). These equations described analytic solutions to Ratcliff's (1978) diffusion model. The original C implementation was part of the fast-dm 30.2. This version was rewritten in modern C++ and adapted for use in Differential Evolution Markov Chain Monte Carlo (DE-MCMC) estimation.

Value

dfastdm Numeric vector of probability densities

pfastdm Numeric vector of cumulative probabilities

rfastdm DataFrame with columns: rt (response time), response (0 = lower/incorrect, 1 = upper/correct).

References

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

Voss, A., & Voss, J. (2007). Fast-dm: A free program for efficient diffusion model analysis. *Behavior Research Methods*, **39**, 767–775. doi:10.3758/BF03192967

Examples

```
# Density function
RT <- 0.550
unsorted_p_vector <- c(
    a = 1, v = 1.5, z = 0.5 * 1, d = 0, sz = 0, sv = 0, t0 = 0.15, st0 = 0,
    s = 1, precision = 3
)
p_vector <- unsorted_p_vector[sort(names(unsorted_p_vector))]

result0 <- dfastdm(RT, p_vector)


# Cumulative distribution function
RT <- seq(0.1, 1.2, 0.01)
unsorted_p_vector <- c(
    a = 1, v = 1.5, zr = 0.5 * 1, d = 0, sz = 0.0,
```

```
    sv = 0, t0 = 0.15, st0 = 0, s = 1, precision = 3
)
p_vector <- unsorted_p_vector[sort(names(unsorted_p_vector))]

result1 <- pfastdm(RT, p_vector)

sz <- 0.05
sv <- 0.01
st0 <- 0.001

unsorted_p_vector <- c(
    a = 1, v = 1.5, zr = 0.5 * 1, d = 0, sz = sz, sv = sv, t0 = 0.15,
    st0 = st0,
    s = 1, precision = 3
)
p_vector <- unsorted_p_vector[sort(names(unsorted_p_vector))]
result2 <- pfastdm(RT, p_vector, is_lower = TRUE)

# Random generation
unsorted_p_vector <- c(
    a = 1, v = 1.5, zr = 0.5 * 1, d = 0, szr = 0, sv = 0.0, t0 = 0.15,
    st0 = 0, s = 1, precision = 3
)
p_vector <- unsorted_p_vector[sort(names(unsorted_p_vector))]
set.seed(123)
time_parameters <- c(-1, 1, 0.01)
result3 <- rfastdm(
    n = 1, parameters_r = p_vector,
    time_parameters_r = time_parameters, debug = TRUE
)

# Debugging information
# st0 = 0 < 1.50017e-08. sv = 0 < 1e-05. sz = 0 < 1e-05. Selecting f_plain.
#
# Initial search range: t_min = -1, t_max = 1
# Point 0: t = -1, F = 0.00139754
# Point 1: t = -0.99, F = 0.00148506
# Point 2: t = -0.98, F = 0.0015778
# Point 3: t = -0.97, F = 0.00167698
# Point 4: t = -0.96, F = 0.00178253
# Point 196: t = 0.96, F = 0.99201
# Point 197: t = 0.97, F = 0.992483
# Point 198: t = 0.98, F = 0.992927
# Point 199: t = 0.99, F = 0.993343
# Point 200: t = 1, F = 0.993735
```

`simulate,ddm-method` *Simulate Data from a Decision Diffusion Model*

Description

Simulate response times and choices from a Decision Diffusion model (DDM) associated with a specific experimental design. The specification is typically specified, using `ggdmcModel::BuildModel`).

Usage

```
## S4 method for signature 'ddm'
simulate(
  object,
  nsim = 4L,
  seed = NULL,
  n_subject = 3L,
  parameter_vector = NULL,
  time_parameters = c(t_min = -1, t_max = 1, dt = 0.001),
  debug = FALSE
)
```

Arguments

`object`	An object of class `"ddm"` that defines the model structure and parameters.
`nsim`	Integer. Number of trials to simulate per subject. Defaults to `4`. The function adjusts `nsim` if not divisible by number of conditions.
`seed`	Optional integer. Sets the random seed for reproducibility. Defaults to `NULL`.
`n_subject`	Integer. Number of subjects to simulate. Defaults to `3`.
`parameter_vector`	Optional. A named vector or list of parameters (e.g., `a`, `z`, `v`, `t0`) used as the true value to simulate. The function by default generates one or many true parameter vector based on the population distribution stored in the `ddm` object.
`time_parameters`	Numeric vector of for setting the time range and time step for simulation (using inverse transform method).
`debug`	Logical. If `TRUE`, print debugging output during simulation. Defaults to `FALSE`.

Details

This method simulates data from a design-based Decision Diffusion model (DDM). You can simulate multiple subjects, override default parameters, and use conduct inverse sampling with a fine time step. You may turn on debugging output when something goes wrong.

Value

A data frame with simulated trial data, including columns like `subject`, `trial`, `choice`, and `rt`.

Examples

```
if (requireNamespace("ggdmcModel", quietly = TRUE)) {
  BuildModel <- getFromNamespace("BuildModel", "ggdmcModel")
  model <- ggdmcModel::BuildModel(
```

```
    p_map = list(
      a = "1", v = "1", z = "1", d = "1", sz = "1", sv = "1",
      t0 = "1", st0 = "1", s = "1", precision = "1"
    ),
    match_map = list(M = list(s1 = "r1", s2 = "r2")),
    factors = list(S = c("s1", "s2")),
    constants = c(d = 0, s = 1, st0 = 0, sv = 0, precision = 3),
    accumulators = c("r1", "r2"),
    type = "fastdm"
  )

# Simulate one subject ------------------
sub_model <- setDDM(model)
p_vector <- c(a = 1, sz = 0.25, t0 = 0.15, v = 2.5, z = .38)
dat <- simulate(sub_model,
  nsim = 256, parameter_vector = p_vector,
  n_subject = 1
)
# Simulate multiple subjects ------------------
# 1. First build a population distribution to sample many p_vector's

pop_mean <- c(a = 1, sz = 0.25, t0 = 0.15, v = 2.5, z = 0.38)
pop_scale <- c(a = 0.05, sz = 0.01, t0 = 0.02, v = .5, z = 0.01)
pop_dist <- ggdmcPrior::BuildPrior(
  p0    = pop_mean,
  p1    = pop_scale,
  lower = c(0, 0, 0, -10, 0),
  upper = rep(NA, model@npar),
  dists = rep("tnorm", model@npar),
  log_p = rep(FALSE, model@npar)
)

# 2. Enter the population distribution to a DDM model
pop_model <- setDDM(model, population_distribution = pop_dist)

# 3. simulate method will use the distribution to sample new p_vector's
# Note: You must enter sensible pop_mean, pop_scale and distribution.
hdat <- simulate(pop_model, nsim = 128, n_subject = 32)
}
```

`validate_ddm_parameters` *Simulate Design-based Decision Diffusion Model (DDM) Trials*

Description

Generates synthetic response time (RT) and choice data using a diffusion decision model (DDM) with specified parameters. The simulation allows for flexible parameter settings and time-domain controls.

Usage

```
validate_ddm_parameters(rt_model_r, parameters_r, debug = FALSE)

simulate_ddm_trials(
  rt_model_r,
  parameters_r,
  time_parameters_r = as.numeric(c()),
  n_trial = 1L,
  debug = FALSE
)
```

Arguments

`rt_model_r`	An S4 object of class `ddm` specifying the DDM configuration. Must contain slots for boundary separation, drift rate, and other DDM-relevant parameters.
`parameters_r`	A numeric vector of DDM parameters. Expected elements: **a** Boundary separation (threshold) **v** Drift rate (evidence accumulation speed) **t0** Non-decision time (encoding + motor response baseline) **z** Starting point (bias; must satisfy `0 < z < a`) **d** (Optional) Difference in motor execution time between boundaries **sv** (Optional) Inter-trial variability in drift rate **st0** (Optional) Inter-trial variability in non-decision time **sz** (Optional) Inter-trial variability in starting point
`debug`	Logical. If `TRUE`, prints internal simulation states and timing diagnostics.
`time_parameters_r`	A numeric vector controlling simulation's temporal dynamics: **[1]** Minimum time (default: `-0.5`) **[2]** Maximum time (default: `0.5`) **[3]** Time step for numerical integration (default: `0.01`)
`n_trial`	Integer. Number of trials to simulate (default: `1`).

Details

The core simulation is implemented in C++ via `simulate_ddm_trials` for high performance in large-scale experiments. A complementary R wrapper (`.simulate_ddm_trials`) provides additional functionality, such as random seed control and input validation.

The function `validate_ddm_parameters` checks whether the input parameter vector satisfies the constraints of the diffusion model.

Value

A `data.frame` with the following columns:

rt Simulated response times in seconds

response Binary decision outcome (0 = lower boundary, 1 = upper boundary)

References

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

See Also

ddm-class, setDDM, validate_ddm_parameters

Examples

```
# Basic simulation with default parameters
if (requireNamespace("ggdmcModel", quietly = TRUE)) {
  BuildModel <- getFromNamespace("BuildModel", "ggdmcModel")
  model <- ggdmcModel::BuildModel(
    p_map = list(
      a = "1", v = "1", z = "1", d = "1", sz = "1", sv = "1",
      t0 = "1", st0 = "1", s = "1", precision = "1"
    ),
    match_map = list(M = list(s1 = "r1", s2 = "r2")),
    factors = list(S = c("s1", "s2")),
    constants = c(d = 0, s = 1, st0 = 0, sv = 0, precision = 3),
    accumulators = c("r1", "r2"),
    type = "fastdm"
  )
}
sub_model <- setDDM(model)
p_vector <- c(a = 1, sz = 0.25, t0 = 0.15, v = 2.5, z = .38)
time_parameters <- c(t_min = -0.5, tmax = 0.5, dt = 0.01)

# Simulation with custom time parameters
sim_data <- simulate_ddm_trials(
  rt_model_r = sub_model,
  parameters_r = p_vector,
  time_parameters_r = time_parameters,
  n_trial = 32
)
```

Index

BuildPrior, *3*

ddm-class, 2
dfastdm, 3

fastdm *(*dfastdm*)*, 3

pfastdm *(*dfastdm*)*, 3

rfastdm *(*dfastdm*)*, 3

setDDM, *9*
setDDM *(*ddm-class*)*, 2
simulate,ddm-method, 5
simulate_ddm_trials
        *(*validate_ddm_parameters*)*, 7

validate_ddm_parameters, 7, *9*