Package ‘ggdmcLikelihood’

July 30, 2025

Type Package

Title Likelihood Computation for 'ggdmc' Package

Version 0.2.9.0

Date 2025-07-28

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

Description Efficient computation of likelihoods in design-based choice response time models, including the Decision Diffusion Model, is supported. The package enables rapid evaluation of likelihood functions for both single- and multi-subject models across trial-level data. It also offers fast initialisation of starting parameters for genetic sampling with many Markov chains, facilitating estimation in complex models typically found in experimental psychology and behavioural science. These optimisations help reduce computational overhead in large-scale model fitting tasks.

License GPL (>= 2)

URL https://github.com/yxlin/ggdmcLikelihood

Imports Rcpp (>= 1.0.7)

Depends R (>= 3.5.0)

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

Suggests testthat, ggdmcModel

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Yi-Shin Lin [aut, cre]

Repository CRAN

Date/Publication 2025-07-30 08:00:12 UTC

Contents

compute_likelihood *Compute Likelihood for Behavioural Models*

Description

Computes the likelihood for multiple subjects by aggregating results from individual subject computations.

Usage

```
compute_likelihood(dmis, parameter_r, debug = FALSE)

compute_subject_likelihood(dmi, parameter_r, debug = FALSE)
```

Arguments

`dmis`	A list of S4 data model instances (one per subject)
`parameter_r`	A list (one per subject) of or one numeric vector containing model parameters
`debug`	Logical flag for debug mode (default = FALSE)
`dmi`	One S4 data model instance (for one subject)

Details

These functions compute likelihoods for behavioural models, with `compute_subject_likelihood` handling a single subject and `compute_likelihood` handling multiple subjects.

These functions provide access to the internal mechanism of the design-based likelihood computation. They primarily intended to initialise new 'samples' or to verify that the likelihood evaluations, when associated with a particular design, are computed accurately.

Value

with:

- `compute_likelihood` returns a list. Each element is the likelihood for a subject. The element in the inner list is the likelihood for a condition.
- `compute_subject_likelihood` returns also a list. Each element is the likelihood for a condition.

Examples

```
# Example dataset
hdat <- data.frame(
  RT = round(runif(15, min = 0.4, max = 1.2), 7),
  R  = sample(c("r1", "r2", "r3"), size = 15, replace = TRUE),
  s  = rep(1:3, each = 5),
  S  = rep(c("s1", "s2", "s3"), each = 5),
  stringsAsFactors = FALSE
)
dat <- hdat[hdat$s==1, ]

p_vector <- c(A = .75, B = 1.25, mean_v.false = 1.5, mean_v.true = 2.5, t0 = .15)
nsubject <- length(unique(hdat$s))

if(requireNamespace("ggdmcModel", quietly = TRUE)) {
    BuildModel <- getFromNamespace("BuildModel", "ggdmcModel")
    BuildDMI   <- getFromNamespace("BuildDMI", "ggdmcModel")

    model <- BuildModel(
        p_map = list(A = "1", B = "1", t0 = "1", mean_v = "M", sd_v = "1", st0 = "1"),
        match_map = list(M = list(s1 = "r1", s2 = "r2")),
        factors = list(S = c("s1", "s2")),
        constants = c(st0 = 0, sd_v = 1),
        accumulators = c("r1", "r2"),
        type = "lba")
    pop_dmis <- BuildDMI(hdat, model)
    sub_dmis <- BuildDMI(dat, model)

    parameters <- list()
    for (i in seq_len(nsubject)) {
        new_p_vector <- p_vector[model@pnames]
        parameters[[i]] <- new_p_vector
    }

    result1 <- compute_subject_likelihood(sub_dmis[[1]], parameters[[1]], FALSE)
    result2 <- compute_likelihood(pop_dmis, parameters, FALSE)
}

print(result1)
print(result2)
```

Index