Package 'ggdmcPrior'

May 8, 2026

Type Package

Title Prior Probability Functions of the Standard and Truncated Distribution

Version 0.2.9.0

Date 2025-07-23

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

Description Provides tools for specifying and evaluating standard and truncated probability distributions, with support for log-space computation and joint distribution specification. It enables Bayesian computation for cognition models and includes utilities for density calculation, sampling, and visualisation, facilitating prior distribution specification and model assessment in hierarchical Bayesian frameworks.

License GPL (>= 2)

Imports Rcpp (>= 1.0.7), methods, lattice

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 2025-07-24 13:00:02 UTC

Contents

`BuildPrior` *Build a Joint Prior Distribution*

Description

`BuildPrior` sets up a joint distribution of the prior. Each model parameter is assigned one probability distribution. `p0` and `p1` refer to the first and second parameters. I use the convention of the 0-based index to work with the C++ and the Python sister package, 'pydmc' (coming soon). `p0` must comes with parameter names.

Usage

```
BuildPrior(
  p0,
  p1,
  lower = rep(NA, length(p0)),
  upper = rep(NA, length(p0)),
  dists = rep("norm", length(p0)),
  log_p = rep(TRUE, length(p0)),
  types = c("tnorm", "beta", "gamma", "lnorm", "cauchy", "unif", "norm")
)
```

Arguments

`p0`	the first parameter of a distribution (e.g., mean, shape1, etc.).
`p1`	the second parameter of a distribution (e.g., sd, shape2, etc.).
`lower`	lower support (boundary). Default is `NA`, which will be converted to a real value or -Inf based on the distribution type.
`upper`	upper support (boundary). Default is `NA`, which will be converted to a real value or Inf based on the distribution type.
`dists`	a vector of character strings specifying the distribution type for each parameter. Valid types are: `"tnorm"`, `"beta"`, `"gamma"`, `"lnorm"`, `"cauchy"`, `"unif"`, and `"norm"`. Default is `"norm"`.
`log_p`	logical; if `TRUE`, probabilities are given as `log(p)`. Default is `TRUE`.
`types`	available distribution types.

Details

Seven distributions are implemented:

1. Truncated normal distribution, where: `p0` = `mean`, `p1` = `sd`. When the lower and upper bounds are not provided, they are set to `-Inf` and `Inf`, rendering a normal distribution (see tnorm). Type name is `"tnorm"`.
2. Beta distribution, where: `p0` = `shape1` and `p1` = `shape2` (see pbeta). Note the uniform distribution is a special case of the beta with `p0` = 1 and `p1` = 1. Type name is `"beta"`.
3. Gamma distribution, where `p0` = `shape` and `p1` = `scale` (see pgamma). Note `p1` is scale, not rate. Type name is `"gamma"`.
4. Log-normal, where `p0` = `meanlog` and `p1` = `sdlog` (see plnorm). Type name is `"lnorm"`.
5. Cauchy distribution, where `p0` = `location` and `p1` = `scale` (see pcauchy). Type name is `"cauchy"`.
6. Uniform distribution, where `p0` = `lower` and `p1` = `upper` (see punif). Type name is `"unif"`.
7. Normal distribution, where `p0` = `mean` and `p1` = `sd` (see pnorm). Type name is `"norm"`.

Value

a list of lists, where each sub-list contains the parameter for its prior definition. Each sub-list includes:

- `p0`: The first parameter of the distribution.
- `p1`: The second parameter of the distribution.
- `lower`: The lower bound of the distribution.
- `upper`: The upper bound of the distribution.
- `dist`: A numeric code representing the distribution type.
- `log_p`: Logical indicating whether probabilities are logged.

Examples

```
# Using dbeta to represent a uniform distribution of bounds(0, 1)
x <- seq(-.1, 1.1, .001)
plot(x, dbeta(x, 1, 1),
    type = "l", ylab = "Density", xlab = "x",
    lwd = 2, cex.lab = 1.5, cex.axis = 2
)

## Create an S4 prior object
p_prior <- BuildPrior(
    p0 = c(A = 0.15, B = 0.45, mean_v = 2.25, sd_v = 0.15, t0 = 0.2),
    p1 = rep(0.1, 5),
    lower = rep(NA, 5),
    upper = rep(NA, 5),
    dist = rep("tnorm", 5),
    log_p = rep(NA, 5)
)
```

```
print_prior(p_prior)

# Use the beta distribution to create uniform densities
# lower and upper set the bounds. If lower is NA, it will be set to 0.
# If upper is NA, it will be set to 1.
p_prior <- BuildPrior(
    p0 = c(A = 1, B = 1, mean_v = 1, sd_v = 1, t0 = 1),
    p1 = rep(1, 5),
    lower = rep(0, 5),
    upper = rep(5, 5),
    dist = rep("beta", 5),
    log_p = rep(FALSE, 5)
)

p0 <- plot_prior(p_prior, font_size = 3.5, cex = 3.5)
```

`dprior` *Density and Random Number Generation for a Join Prior Distribution*

Description

`dprior` computes the log-density of a joint prior distribution at a given set of parameter values. `rprior` generates random samples from the same joint prior specification.

Usage

```
dprior(p_prior_r, parameters_r)

rprior(p_prior_r, n = 1L)
```

Arguments

`p_prior_r`	A list specifying the prior distribution, typically constructed using `BuildPrior`.
`parameters_r`	For `dprior`: A numeric vector of parameter values at which the prior log-density should be evaluated.
`n`	For `rprior`: Integer specifying the number of random samples to generate (default is 1).

Details

These functions implement the core computations for evaluating and sampling from a joint prior distribution specified via `BuildPrior`:

- `dprior`: Evaluates the log-density of the joint prior at the given parameter values.
- `rprior`: Draws independent samples from the specified joint prior.

The joint prior may include truncated normal, beta, gamma, and other common distributions, possibly bounded by user-specified lower and upper limits.

Value

dprior A numeric vector of log-density values.

rprior A numeric matrix of dimension n × nparameter, containing samples from the prior distribution. Each row is one sample.

Examples

```
p0 <- c(A = 0.15, B = 0.45, mean_v = 2.25, sd_v = 0.15, t0 = 0.2)
p1 <- rep(0.1, 5)
p_prior <- BuildPrior(
    p0 = p0,
    p1 = p1,
    lower = rep(NA, 5),
    upper = rep(NA, 5),
    dist = rep("tnorm", 5),
    log_p = rep(TRUE, 5)
)

# Evaluate log-density
parameters <- seq(0.1, 0.5, by = 0.1)
res0 <- dprior(p_prior, parameters)
res1 <- dnorm(parameters, p0, 0.1, TRUE)
print(res0)
print(res1)

# Generate samples from the prior
res2 <- rprior(p_prior, 1)
res3 <- rprior(p_prior, 2)
print(res2)
print(res3)
```

---

plot_prior *Visualise Distributions*

Description

Plots density curves for specified distributions to help visualise their shape and domain.

Usage

```
plot_prior(p_prior, font_size = 5, cex = 5, return_data = FALSE)
```

Arguments

p_prior A list of distribution specifications. Each element should be a list containing:

dist A character string specifying the distribution type. Supported values include: "tnorm", "beta", "gamma", "lnorm", "cauchy", "unif", and "norm".

	p0 The first parameter of the distribution. p1 The second parameter of the distribution (if applicable). lower Optional lower bound (used for truncated distributions). upper Optional upper bound (used for truncated distributions). log_p Logical indicating whether to compute log-densities.
font_size	Numeric. Base font size for plot labels. Defaults to 5.
cex	Numeric. Scaling factor for plot elements. Defaults to 5.
return_data	Logical. If TRUE, returns the computed density data instead of plotting. Defaults to FALSE.

Details

This function:

- Automatically determines appropriate x-axis ranges based on each distribution's properties.
- Handles both truncated and unbounded distributions.
- Supports all distribution types implemented in the package.

For truncated distributions, the density is plotted only within the specified bounds. A heuristic is used to generate axis limits and labels using the internal generate_x_value function.

Value

If return_data = FALSE (default), a lattice plot object is returned displaying the density curves for each prior. If return_data = TRUE, a data frame is returned with the following columns:

- x: Numeric vector of x-values generated for each prior using a heuristic.
- y: Corresponding density (or log-density) values.
- gp: Group label or parameter name for each distribution.

Examples

```
# Define a joint distribution
p_prior <- BuildPrior(
    p0 = c(A = 0.15, B = 0.45, mean_v = 2.25, sd_v = 0.15, t0 = 0.2),
    p1 = rep(0.1, 5),
    lower = rep(NA, 5),
    upper = rep(NA, 5),
    dist = rep("tnorm", 5),
    log_p = rep(FALSE, 5)
)

plot_prior(p_prior)
```

print_prior *Print a Joint Prior Distribution*

Description

Displays the structure of a prior distribution specification passed to C++ functions. This function is primarily intended for debugging and inspection of the internal representation.

Usage

```
print_prior(p_prior_r)
```

Arguments

p_prior_r	A list specifying the prior distributions, typically generated by BuildPrior.

Details

This function is mainly used for debugging purposes. It provides a readable summary of the prior distribution list as received by C++ code. The input p_prior_r should be the output of BuildPrior.

Value

A character vector summarising the prior specification.

Examples

```
p0 <- c(A = 0.15, B = 0.45, mean_v = 2.25, sd_v = 0.15, t0 = 0.2)
p1 <- rep(0.1, 5)
p_prior <- BuildPrior(
    p0 = p0,
    p1 = p1,
    lower = rep(NA, 5),
    upper = rep(NA, 5),
    dist = rep("tnorm", 5),
    log_p = rep(TRUE, 5)
)
print_prior(p_prior)
```

`prior-class` *An S4 Class to Represent a Joint Prior Distribution*

Description

This class encapsulates the structure of prior distributions used in hierarchical Bayesian modelling. It stores both subject-level and population-level (hyperparameter) priors for a model's parameters, and is used in Bayesian inference workflows, particularly with models from the **lbaModel** or **ddModel** packages.

Value

An S4 object of class `"prior"`, used in computing prior densities and visualising prior distributions.

Slots

`nparameter` Integer. Number of free parameters in the model.

`pnames` Character vector. Names of the free parameters.

`p_prior` List. Represents the joint prior distribution at the subject level, usually constructed from standard or truncated distributions.

`h_prior` List. Representing the joint prior at the population level, typically containing location and scale parameters for hierarchical models. The 'h' prefix refers to hyperparameters.

Structure

An object of class `"prior"` contains the following components:

`nparameter` Number of free parameters.

`pnames` Names of the model's free parameters.

`p_prior` Subject-level prior specification. Conceptually analogous to the model likelihood in a hierarchical Bayesian model.

`h_prior` Hyperparameter-level (group-level) prior specification.

Usage

Used to define priors for hierarchical Bayesian cognitive models. This class allows structured specification of priors at both individual and group levels. Prior objects are commonly constructed using `set_priors`, which integrates multiple `BuildPrior` outputs into a single prior structure.

See Also

`BuildPrior`

`set_priors` *Set up Prior Distributions*

Description

Configures a set of joint prior distributions for:

- Subject-level parameters (`p_prior`), which also serve as the likelihood function in population-level models.
- Population-level location parameters (`l_prior`).
- Population-level scale parameters (`s_prior`).

Usage

```
set_priors(p_prior, l_prior = NULL, s_prior = NULL)
```

Arguments

`p_prior`	A list specifying prior distributions for subject-level parameters (or the likelihood function for the population-level model). Each element in the list should contain: • `p0`: The first parameter of the distribution. • `p1`: The second parameter of the distribution. • `lower`: The lower bound of the distribution. • `upper`: The upper bound of the distribution. • `dist`: A numeric code representing the distribution type. • `log_p`: Logical indicating whether to compute in log space.
`l_prior`	Optional list specifying prior distributions for population-level location parameters. Should have the same structure as `p_prior`. Defaults to `NULL`.
`s_prior`	Optional list specifying prior distributions for population-level scale parameters. Should have the same structure as `p_prior`. Defaults to `NULL`.

Details

This function performs the following:

- Validates the structure of all prior specifications.
- Ensures required distribution parameters are present and bounds are valid.
- Merges `l_prior` and `s_prior` into a single `h_prior` using `.merge_prior`.
- Returns a structured `prior` object for use in model fitting and simulation.

The argument `log_p` should be set to `TRUE` for density evaluation and `FALSE` when generating samples (e.g., for initial parameter values).

Value

An S4 object of class "prior" with the following slots:

- nparameter: Integer; number of parameters in the joint prior.
- pnames: Character vector of parameter names.
- p_prior: List containing prior specifications for subject-level parameters.
- h_prior: List containing merged prior specifications for l_prior and s_prior.

Examples

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

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

  ####################################
  # priors for subject-level modelling
  ####################################
  p0 <- rep(0, model@npar)
  names(p0) <- model@pnames
  p_prior <- BuildPrior(
    p0 = p0,
    p1 = rep(10, model@npar),
    lower = rep(0, model@npar),
    upper = rep(NA, model@npar),
    dist = rep("unif", model@npar),
    log_p = rep(TRUE, model@npar)
  )
  sub_priors <- set_priors(p_prior = p_prior)

  ####################################
  # priors for hierarchical modelling
  ####################################
  p0 <- runif(model@npar)
  names(p0) <- model@pnames
  model_likelihood <- BuildPrior(
    p0 = p0,
    p1 = rep(10, model@npar),
    lower = rep(0, model@npar),
    upper = rep(NA, model@npar),
    dist = rep("tnorm", model@npar),
```

```
    log_p = rep(TRUE, model@npar)
  )

  p0 <- rep(0, model@npar)
  names(p0) <- model@pnames
  l_prior <- BuildPrior(
    p0 = p0,
    p1 = rep(10, model@npar),
    lower = rep(0, model@npar),
    upper = rep(NA, model@npar),
    dist = rep("unif", model@npar),
    log_p = rep(TRUE, model@npar)
  )
  s_prior <- BuildPrior(
    p0 = p0,
    p1 = rep(10, model@npar),
    lower = rep(NA, model@npar),
    upper = rep(NA, model@npar),
    dist = rep("unif", model@npar),
    log_p = rep(TRUE, model@npar)
  )

  pop_priors <- set_priors(
    p_prior = model_likelihood,
    l_prior = l_prior, s_prior = s_prior
  )
}
```

`sumlogprior` *Sum of Log Prior Densities for a Joint Prior Distribution*

Description

Computes the sum of log-densities for a vector of parameters, based on their respective prior distribution specifications. This function is used in Bayesian computations where the joint prior is the product of independent priors—thus, the log of the joint prior is the sum of log-densities.

Usage

```
sumlogprior(p_prior_r, parameters_r)
```

Arguments

`p_prior_r` A list of prior specifications. Each element is itself a list specifying the prior for one parameter, typically created by `BuildPrior`. Each sublist should contain:

- `p0`: First parameter of the distribution (e.g., mean).
- `p1`: Second parameter (e.g., standard deviation or shape).
- `lower`: Lower bound for the parameter (if applicable).
- `upper`: Upper bound (if applicable).

- `dist`: Character string specifying the distribution type.
- `log_p`: Logical; should be `TRUE` for log-density computation.

`parameters_r` A numeric vector of parameter values at which to evaluate the prior. Must be the same length as `p_prior_r`.

Details

This function performs the following steps:

- Iterates over each parameter and its associated prior specification
- Evaluates the log-density for each parameter
- Sums all log-densities to compute the joint log prior

This is useful, for example, in computing the log-posterior of hierarchical Bayesian models where priors are assumed to be independent.

Value

A single numeric value: the sum of log-densities evaluated at the given parameter vector.

Examples

```
p0 <- c(A = 0.15, B = 0.45, mean_v = 2.25, sd_v = 0.15, t0 = 0.2)
tnorm_prior <- BuildPrior(
    p0 = p0,
    p1 = rep(1, 5),
    lower = rep(0, 5),
    upper = rep(NA, 5),
    dist = rep("tnorm", 5),
    log_p = rep(TRUE, 5)
)

npar <- length(tnorm_prior)
theta <- runif(npar, 0, 10)
result <- sumlogprior(p_prior_r = tnorm_prior, parameters_r = theta)
print(result)
```

---

tnorm *Truncated Normal Distribution Functions*

Description

Density, distribution function, and random number generation for the truncated normal distribution with mean `p0`, standard deviation `p1`, and truncation bounds `[lower, upper]`.

Usage

```
rtnorm(n, p0, p1, lower, upper)

ptnorm(x, p0, p1, lower, upper, lower_tail, log_p = FALSE)

dtnorm(x, p0, p1, lower, upper, log_p = FALSE)
```

Arguments

n	Integer. Number of random variates to generate (for rtnorm).
p0	Mean of the underlying (untruncated) normal distribution.
p1	Standard deviation of the underlying normal distribution. Must be positive.
lower	Lower bound of truncation (can be -Inf).
upper	Upper bound of truncation (can be Inf).
x	Numeric vector of quantiles (for dtnorm and ptnorm).
lower_tail	Logical; if TRUE (default), probabilities are computed as $P[X \leq x]$, otherwise $P[X > x]$.
log_p	Logical; if TRUE, probabilities or densities are returned on the log scale.

Details

These functions implement the truncated normal distribution:

- rtnorm: Generates random samples using inverse transform sampling.
- ptnorm: Computes the cumulative distribution function (CDF).
- dtnorm: Computes the probability density function (PDF).

The truncated normal distribution is defined as:

$$X \sim \mathcal{N}(\mu, \sigma^2), \quad \text{truncated to } [a, b]$$

where $\mu =$ p0, $\sigma =$ p1, $a =$ lower, and $b =$ upper.

Truncation can be one-sided (e.g., lower = 0, upper = Inf) or two-sided.

Value

rtnorm A numeric vector of length n, containing random variates from the truncated normal distribution.

ptnorm A numeric vector of cumulative probabilities evaluated at x.

dtnorm A numeric vector of (log) density values evaluated at x.

References

- Olmsted, J. (2020). *RcppTN: Rcpp-Based Truncated Normal Distribution*. https://github.com/olmjo/RcppTN
- Jackson, C. (2011). **msm**: Multi-state Modelling. https://cran.r-project.org/package=msm
- Robert, C. P. (1995). "Simulation of truncated normal variables." *Statistics and Computing*, 5(2), 121–125. doi:10.1007/BF00143942

Examples

```
# Generate random samples from truncated normal
n <- 1e5
p0 <- 0
p1 <- 1
lower <- 0
upper <- Inf
rtnorm_dat <- rtnorm(n, p0, p1, lower, upper)

# Density at quantiles
x <- seq(-5, 5, length.out = 1e3)
dtnorm_dat <- dtnorm(x, p0, p1, lower = -2, upper = 2, log_p = FALSE)

# Cumulative probabilities
q <- seq(-5, 5, length.out = 1e3)
ptnorm_dat <- ptnorm(q, p0, p1, lower = -2, upper = 3,
                     lower_tail = TRUE, log_p = FALSE)

# Plotting
cex_lab <- 1
cex_axis <- 0.5
line_width <- 1.5

hist(rtnorm_dat, breaks = "fd", freq = FALSE, xlab = "",
     cex.lab = cex_lab, cex.axis = cex_axis, main = "")
plot(x, dtnorm_dat, type = "l", lwd = line_width, xlab = "",
     ylab = "Density", cex.lab = cex_lab, cex.axis = cex_axis, main = "")
```

Index