Package ‘pg’

July 23, 2025

Title Polya Gamma Distribution Sampler

Version 0.2.4

Description Provides access to a high performant random distribution sampler for the Polya Gamma Distribution using either 'C++' headers for 'Rcpp' or 'RcppArmadillo' and 'R'.

URL https://tmsalab.github.io/pg/, https://github.com/tmsalab/pg

BugReports https://github.com/tmsalab/pg/issues

License GPL (>= 3)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 2.1.0)

NeedsCompilation yes

Author James Balamuta [aut, cre, cph]

Maintainer James Balamuta <balamut2@illinois.edu>

Repository CRAN

Date/Publication 2023-07-22 10:30:06 UTC

Contents

pg_mean *Theoretical Polya Gamma Distribution's Mean and Variance*

Description

Compute the theoretical mean and variance for a Polya Gamma variable.

Usage

```
pg_mean(h, z)

pg_var(h, z)
```

Arguments

h	A single `integer` value corresponding to the "shape" parameter.
z	A single `numeric` value corresponding to the "scale" parameter.

Value

Either the theoretical mean or theoretical variance for a Polya Gamma distribution.

Examples

```
# Fixed parameter distribution simulation ----

## Parameters  ----
h = 1; z = .5
## Attempt distribution recovery  ----
vector_of_pg_samples = rpg_vector(1e6, h, z)

head(vector_of_pg_samples)
length(vector_of_pg_samples)

## Obtain the empirical results   ----
empirical_mean = mean(vector_of_pg_samples)
empirical_var = var(vector_of_pg_samples)

## Take the theoretical values ----
theoretical_mean = pg_mean(h, z)
theoretical_var = pg_var(h, z)

## Form a comparison table ----

# empirically sampled vs. theoretical values
rbind(c(empirical_mean, theoretical_mean),
      c(empirical_var, theoretical_var))
```

`rpg_scalar` *Sample from the Polya Gamma distribution PG(h, z)*

Description

Chooses the most efficient implemented method to sample from a Polya Gamma distribution. Details on algorithm selection presented below.

Usage

```
rpg_scalar(h, z)

rpg_vector(n, h, z)

rpg_hybrid(h, z)

rpg_gamma(h, z, trunc = 1000L)

rpg_devroye(h, z)

rpg_sp(h, z)

rpg_normal(h, z)
```

Arguments

`h`	`integer` values corresponding to the "shape" parameter.
`z`	`numeric` values corresponding to the "scale" parameter.
`n`	The number of samples to taken from a PG(h, z). Used only by the vector sampler.
`trunc`	Truncation cut-off. Only used by the gamma sampler.

Details

The following sampling cases are enabled:

- `h > 170`: Normal approximation method
- `h > 13`: Saddlepoint approximation method
- `h = 1` or `h = 2`: Devroye method
- `h > 0`: Sum of Gammas method.
- `h < 0`: Result is automatically set to zero.

Value

A single `numeric` value.

Examples

```
# Fixed parameter distribution simulation ----

## Parameters  ----
h = 1; z = .5

## Sample only one value  ----
single_value = rpg_scalar(h, z)
single_value

## Attempt distribution recovery  ----
vector_of_pg_samples = rpg_vector(1e6, h, z)

head(vector_of_pg_samples)
length(vector_of_pg_samples)

## Obtain the empirical results   ----
empirical_mean = mean(vector_of_pg_samples)
empirical_var = var(vector_of_pg_samples)

## Take the theoretical values ----
theoretical_mean = pg_mean(h, z)
theoretical_var = pg_var(h, z)

## Form a comparison table ----

# empirically sampled vs. theoretical values
rbind(c(empirical_mean, theoretical_mean),
      c(empirical_var, theoretical_var))

# Varying distribution parameters ----

## Generate varying parameters ----
u_h = 20:100
u_z = 0.5*u_h

## Sample from varying parameters ----
x = rpg_hybrid(u_h, u_z)
```

Index