Package: millsratio (via r-universe)

June 7, 2026

Title Interactive Analysis of Mills Ratios and Tail Thickness

Version 0.2.3

Description Compute and compare Mills ratios for normal, t, and exponential distributions. Explore the t(30) paradox where heavier tails produce larger Mills ratios despite near-normal density. Includes an interactive Shiny dashboard for visualising tail behaviour and hazard functions.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr, ggplot2, plotly, scales, shiny, stats, tidyr

Suggests bslib, covr, DT, gt, htmlwidgets, knitr, purrr, quarto, rmarkdown, shinylive, tarchetypes, targets, testthat (>= 3.0.0)

VignetteBuilder quarto

Config/testthat/edition 3

Config/Needs/website pkgdown

URL https://johngavin.github.io/millsratio/, https://github.com/JohnGavin/millsratio

BugReports https://github.com/JohnGavin/millsratio/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://johngavin.r-universe.dev

Date/Publication 2026-02-23 20:05:20 UTC

RemoteUrl https://github.com/JohnGavin/millsratio

RemoteRef HEAD

RemoteSha 02649c790bb2fa3a6d0156b0896fc81188c87184

Contents

`analyze_t30_paradox` *Generate Data for t(30) Paradox Analysis*

Description

Generate Data for t(30) Paradox Analysis

Usage

```
analyze_t30_paradox(x_range = c(0, 5), n_points = 200)
```

Arguments

`x_range`	Range of x values
`n_points`	Number of points

Value

Data frame comparing t(30) and normal distributions

Examples

```
paradox_data <- analyze_t30_paradox()
```

`analyze_tail_thickness` *Analyze Tail Thickness at Specific Points*

Description

Analyze Tail Thickness at Specific Points

Usage

```
analyze_tail_thickness(
  x_points,
  distributions = c("normal", "t30", "exponential"),
  compute_ratios = FALSE
)
```

Arguments

`x_points`	Numeric vector of x values to analyze
`distributions`	Character vector of distributions
`compute_ratios`	Logical; if TRUE, computes ratios relative to normal

Value

Data frame with Mills ratios and optional comparison metrics

Examples

```
# Analyze at key points
tail_analysis <- analyze_tail_thickness(
  x_points = c(2, 3, 4, 5),
  distributions = c("normal", "t30", "exponential"),
  compute_ratios = TRUE
)
```

`compare_mills_hazard` *Compare Mills Ratio and Hazard Function*

Description

Compare Mills Ratio and Hazard Function

Usage

```
compare_mills_hazard(x, distribution = "normal", ...)
```

Arguments

`x`	Numeric vector of values
`distribution`	Character string specifying distribution
`...`	Additional distribution parameters

Value

Data frame with x, mills_ratio, and hazard columns

Examples

```
# Compare for normal distribution
compare_mills_hazard(seq(0, 5, by = 0.5), "normal")

# Tidyverse visualization
library(dplyr)
library(ggplot2)
compare_mills_hazard(seq(0.1, 5, by = 0.1), "normal") %>%
  tidyr::pivot_longer(c(mills_ratio, hazard),
                      names_to = "function_type",
                      values_to = "value") %>%
  ggplot(aes(x, value, color = function_type)) +
  geom_line() +
  scale_y_log10() +
  labs(title = "Mills Ratio vs Hazard Function",
       subtitle = "Note: h(x) = 1/m(x)")
```

compare_mills_ratios *Compare Mills Ratios Across Distributions*

Description

Compare Mills Ratios Across Distributions

Usage

```
compare_mills_ratios(x, distributions = c("normal", "t30", "exponential"))
```

Arguments

x	Numeric vector of quantiles
distributions	Character vector of distribution names ("normal", "t3", "t10", "t30", "exponential")

Value

Data frame with Mills ratios for each distribution

Examples

```
x_vals <- c(1, 2, 3, 4)
compare_mills_ratios(x_vals, c("normal", "t30", "exponential"))
```

dashboard_about *Get Dashboard About Information*

Description

Get Dashboard About Information

Usage

```
dashboard_about()
```

Value

List with dashboard metadata

`find_crossover_point` *Find Crossover Points Between Distributions*

Description

Find Crossover Points Between Distributions

Usage

```
find_crossover_point(
  dist1,
  dist2,
  df1 = NULL,
  df2 = NULL,
  x_range = c(0.01, 10),
  tolerance = 1e-06
)
```

Arguments

`dist1`	First distribution
`dist2`	Second distribution
`df1`	Degrees of freedom for first distribution (if t)
`df2`	Degrees of freedom for second distribution (if t)
`x_range`	Search range for crossover
`tolerance`	Numerical tolerance for finding crossover

Value

Numeric value of x where Mills ratios are equal (NA if no crossover)

Examples

```
# Find where t(30) and normal Mills ratios cross
crossover <- find_crossover_point("t", "normal", df1 = 30)
```

hazard_from_mills *Convert Mills Ratio to Hazard Function*

Description

Convert Mills Ratio to Hazard Function

Usage

```
hazard_from_mills(mills_ratio)
```

Arguments

mills_ratio	Numeric vector of Mills ratio values

Value

Numeric vector of hazard function values

Examples

```
# Normal distribution example
x <- seq(0, 5, by = 0.5)
m <- mills_ratio_normal(x)
h <- hazard_from_mills(m)

# Tidyverse example
library(dplyr)
data.frame(x = x) %>%
  mutate(
    mills = mills_ratio_normal(x),
    hazard = hazard_from_mills(mills)
  )
```

hazard_function *Calculate Hazard Function Directly*

Description

Calculate Hazard Function Directly

Usage

```
hazard_function(x, distribution = "normal", ...)
```

Arguments

x	Numeric vector of values
distribution	Character string: "normal", "t", "exponential"
...	Additional parameters for the distribution

Value

Numeric vector of hazard function values

Examples

```
# Compare hazard functions across distributions
x <- seq(0.1, 5, by = 0.1)
h_normal <- hazard_function(x, "normal")
h_t30 <- hazard_function(x, "t", df = 30)
h_exp <- hazard_function(x, "exponential", rate = 1)

# Tidyverse comparison
library(dplyr)
library(tidyr)
data.frame(x = x) %>%
  mutate(
    Normal = hazard_function(x, "normal"),
    `t(30)` = hazard_function(x, "t", df = 30),
    Exponential = hazard_function(x, "exponential")
  ) %>%
  pivot_longer(-x, names_to = "distribution", values_to = "hazard")
```

---

hazard_functions *Hazard Function Calculations*

Description

Functions for calculating hazard functions and their relationship to Mills ratios. The hazard function h(x) = f(x)/(1-F(x)) is the reciprocal of the Mills ratio m(x).

Details

The hazard function (also called failure rate or force of mortality) represents the instantaneous rate of occurrence of an event at time x, given survival to time x.

Mathematical relationship: h(x) = 1/m(x)

hazard_properties *Hazard Function Properties*

Description

Analyze properties of the hazard function for different distributions.

Usage

```
hazard_properties(distribution = "normal", df = NULL)
```

Arguments

distribution	Character string: "normal", "t", or "exponential"
df	Degrees of freedom for t-distribution

Value

List with hazard function properties

Examples

```
# Analyze normal distribution hazard
hazard_properties("normal")

# Compare properties across distributions
library(purrr)
list(
  normal = hazard_properties("normal"),
  t30 = hazard_properties("t", df = 30),
  exponential = hazard_properties("exponential")
) %>%
  map_df(~ as.data.frame(.x), .id = "distribution")
```

launch_dashboard *Launch Mills Ratio Interactive Dashboard*

Description

Launches an interactive Shiny dashboard for exploring Mills ratios and tail thickness behavior across different distributions.

Usage

```
launch_dashboard(port = NULL, launch_browser = TRUE, host = "127.0.0.1")
```

Arguments

port	Port number for the Shiny app (default = NULL for auto-selection)
launch_browser	Logical; if TRUE, opens dashboard in browser
host	Host address (default = "127.0.0.1" for local access)

Value

Runs the Shiny application

Examples

```
## Not run:
# Launch the dashboard
launch_dashboard()

# Launch on specific port
launch_dashboard(port = 8080)

## End(Not run)
```

---

launch_dashboard_no_browser

Launch Dashboard Without Browser

Description

Convenience function to launch dashboard without attempting to open browser. Useful when browser settings are not configured.

Usage

```
launch_dashboard_no_browser(port = NULL, host = "127.0.0.1")
```

Arguments

port	Port number for the Shiny app (default = NULL for auto-selection)
host	Host address (default = "127.0.0.1" for local access)

Value

Runs the Shiny application

Examples

```
## Not run:
# Launch without browser
launch_dashboard_no_browser()
# Then manually open the displayed URL in your browser

## End(Not run)
```

`launch_dashboard_simple` *Launch Simplified Dashboard*

Description

Launch the simplified, content-focused version of the Mills ratio dashboard. This version emphasizes clarity and substance over visual effects.

Usage

```
launch_dashboard_simple(
  port = 4628,
  launch_browser = FALSE,
  host = "127.0.0.1"
)
```

Arguments

`port`	Port number for the Shiny app (default = 4628)
`launch_browser`	Launch browser? (default = FALSE)
`host`	Host address (default = "127.0.0.1" for local access)

Value

Runs the simplified Shiny application

Examples

```
## Not run:
# Launch simplified dashboard
launch_dashboard_simple()

## End(Not run)
```

mills_asymptotic *Calculate Asymptotic Approximation of Mills Ratio*

Description

Calculate Asymptotic Approximation of Mills Ratio

Usage

```
mills_asymptotic(x, distribution, df = NULL)
```

Arguments

x	Numeric vector of quantiles
distribution	Character string: "normal", "t", or "exponential"
df	Degrees of freedom for t-distribution

Value

Numeric vector of asymptotic approximations

Examples

```
# Normal asymptotic approximation (1/x for large x)
mills_asymptotic(10, "normal")

# Student's t asymptotic approximation (x/df for large x)
mills_asymptotic(10, "t", df = 30)
```

mills_ratio_comparison *Compute Ratio of Mills Ratios*

Description

Compute Ratio of Mills Ratios

Usage

```
mills_ratio_comparison(x, dist1, dist2, df1 = NULL, df2 = NULL)
```

Arguments

x	Numeric vector of quantiles
dist1	First distribution ("normal", "t", "exponential")
dist2	Second distribution ("normal", "t", "exponential")
df1	Degrees of freedom for first distribution (if t)
df2	Degrees of freedom for second distribution (if t)

Value

Numeric vector of ratios m1(x)/m2(x)

Examples

```
# Ratio of t(30) to normal Mills ratios
x_vals <- seq(1, 5, by = 0.5)
ratio <- mills_ratio_comparison(x_vals, "t", "normal", df1 = 30)
```

---

mills_ratio_exp *Compute Mills Ratio for Exponential Distribution*

Description

Compute Mills Ratio for Exponential Distribution

Usage

```
mills_ratio_exp(x, rate = 1, log = FALSE)
```

Arguments

x	Numeric vector of quantiles
rate	Rate parameter (> 0)
log	Logical; if TRUE, returns log(Mills ratio)

Value

Numeric vector of Mills ratios

Examples

```
# Exponential with rate = 1 at x = 2
mills_ratio_exp(2, rate = 1)

# Mills ratio is constant for exponential!
mills_ratio_exp(c(1, 2, 3, 4), rate = 1)
```

mills_ratio_generic *Generic Mills Ratio Function*

Description

Generic Mills Ratio Function

Usage

```
mills_ratio_generic(x, cdf_fun, pdf_fun, log = FALSE, ...)
```

Arguments

x	Numeric vector of quantiles
cdf_fun	CDF function (must accept lower.tail and log.p arguments)
pdf_fun	PDF function (must accept log argument)
log	Logical; if TRUE, returns log(Mills ratio)
...	Additional arguments passed to cdf_fun and pdf_fun

Value

Numeric vector of Mills ratios

Examples

```
# Using generic function for normal distribution
mills_ratio_generic(2, cdf_fun = pnorm, pdf_fun = dnorm)

# Using for custom distribution
# mills_ratio_generic(x, cdf_fun = my_cdf, pdf_fun = my_pdf, param1 = value1)
```

mills_ratio_normal *Mills Ratio Functions for Various Distributions*

Description

Functions to compute Mills ratios for different probability distributions. The Mills ratio m(x) is defined as the ratio of the complementary cumulative distribution function (CCDF) to the probability density function (PDF): m(x) = \[1 - F(x)\] / f(x)

Usage

```
mills_ratio_normal(x, mean = 0, sd = 1, log = FALSE)
```

Arguments

x	Numeric vector of quantiles
mean	Mean of the distribution (default = 0)
sd	Standard deviation (default = 1)
log	Logical; if TRUE, returns log(Mills ratio) for numerical stability

Value

Numeric vector of Mills ratios

Author(s)

John Gavin john.b.gavin@gmail.com Compute Mills Ratio for Normal Distribution

Examples

```
# Standard normal Mills ratio at x = 2
mills_ratio_normal(2)

# Multiple values
mills_ratio_normal(c(1, 2, 3, 4))

# Log Mills ratio for extreme values
mills_ratio_normal(10, log = TRUE)
```

---

mills_ratio_t *Compute Mills Ratio for Student's t Distribution*

Description

Compute Mills Ratio for Student's t Distribution

Usage

```
mills_ratio_t(x, df, log = FALSE)
```

Arguments

x	Numeric vector of quantiles
df	Degrees of freedom (> 0)
log	Logical; if TRUE, returns log(Mills ratio)

Value

Numeric vector of Mills ratios

Examples

```
# t-distribution with df = 30 at x = 2
mills_ratio_t(2, df = 30)

# Compare different degrees of freedom
x_vals <- seq(0, 5, by = 0.5)
m_t3 <- mills_ratio_t(x_vals, df = 3)
m_t30 <- mills_ratio_t(x_vals, df = 30)
```

---

monte_carlo_mills *Monte Carlo Simulation for Mills Ratio Estimation*

Description

Monte Carlo Simulation for Mills Ratio Estimation

Usage

```
monte_carlo_mills(
  n_sim = 10000,
  x_val = 2,
  distribution = "normal",
  df = NULL,
  seed = NULL
)
```

Arguments

n_sim	Number of simulations
x_val	Value at which to estimate Mills ratio
distribution	Distribution to sample from
df	Degrees of freedom (for t-distribution)
seed	Random seed for reproducibility

Value

List with empirical Mills ratio estimate and confidence interval

Examples

```
# Empirically verify Mills ratio for normal at x=2
mc_result <- monte_carlo_mills(n_sim = 10000, x_val = 2, distribution = "normal")
```

plot_mills_comparison *Plot Mills Ratio Comparison*

Description

Plot Mills Ratio Comparison

Usage

```
plot_mills_comparison(
  x_range = c(0, 5),
  dist1,
  dist2,
  df1 = NULL,
  df2 = NULL,
  show_ratio = TRUE
)
```

Arguments

x_range	Range of x values
dist1	First distribution
dist2	Second distribution
df1	Degrees of freedom for first distribution
df2	Degrees of freedom for second distribution
show_ratio	Logical; if TRUE, shows ratio plot

Value

ggplot2 object

Examples

```
plot_mills_comparison(c(0, 5), "t", "normal", df1 = 30)
```

plot_mills_curves *Visualization Functions for Mills Ratio Analysis*

Description

Functions to create static and interactive visualizations of Mills ratios, supporting both ggplot2 and plotly outputs.

Usage

```
plot_mills_curves(
  data,
  log_y = FALSE,
  log_x = FALSE,
  interactive = FALSE,
  title = "Mills Ratio Comparison",
  show_asymptotic = TRUE
)
```

Arguments

data	Data frame from simulate_mills_curves()
log_y	Logical; if TRUE, uses log scale for y-axis
log_x	Logical; if TRUE, uses log scale for x-axis
interactive	Logical; if TRUE, returns plotly object
title	Plot title
show_asymptotic	Logical; if TRUE, shows asymptotic approximations

Value

ggplot2 or plotly object

Author(s)

John Gavin john.b.gavin@gmail.com Plot Mills Ratio Curves

Examples

```
# Generate and plot Mills ratio curves
curves <- simulate_mills_curves(distributions = c("normal", "t30", "exponential"))
plot_mills_curves(curves, log_y = TRUE)
```

---

plot_mills_vs_hazard *Plot Mills Ratio vs Hazard Function Side-by-Side*

Description

Plot Mills Ratio vs Hazard Function Side-by-Side

Usage

```
plot_mills_vs_hazard(
  x_range = c(0.1, 5),
  distributions = c("normal", "t30", "exponential"),
  n_points = 200,
  interactive = FALSE,
  log_y = TRUE
)
```

Arguments

x_range	Numeric vector of length 2 giving the range of x values
distributions	Character vector of distribution names (e.g., c("normal", "t30", "exponential"))
n_points	Number of points to compute (default 200)
interactive	Logical; if TRUE, returns plotly object
log_y	Logical; if TRUE, uses log scale for y-axis

Value

ggplot2 or plotly object with faceted m(x) and h(x) panels

Examples

```
plot_mills_vs_hazard()
plot_mills_vs_hazard(x_range = c(0.5, 8), distributions = c("normal", "t30"))
```

---

plot_t30_paradox *Plot t(30) Paradox Visualization*

Description

Plot t(30) Paradox Visualization

Usage

```
plot_t30_paradox(data, focus = "mills", interactive = FALSE)
```

Arguments

data	Data frame from analyze_t30_paradox()
focus	Character: "mills", "pdf", "cdf", or "all"
interactive	Logical; if TRUE, returns plotly object

Value

ggplot2 or plotly object

Examples

```
paradox_data <- analyze_t30_paradox()
plot_t30_paradox(paradox_data, focus = "mills")
```

`plot_tail_thickness_heatmap` *Plot Tail Thickness Heatmap*

Description

Plot Tail Thickness Heatmap

Usage

```
plot_tail_thickness_heatmap(x_points, distributions, interactive = FALSE)
```

Arguments

`x_points`	Vector of x values
`distributions`	Vector of distribution names
`interactive`	Logical; if TRUE, returns plotly object

Value

ggplot2 or plotly heatmap

Examples

```
plot_tail_thickness_heatmap(
  x_points = seq(1, 5, by = 0.5),
  distributions = c("normal", "t3", "t10", "t30", "exponential")
)
```

`simulate_mills_curves` *Simulation Framework for Mills Ratio Analysis*

Description

Functions to generate and compare Mills ratio curves across distributions, supporting the interactive dashboard and analysis vignettes.

Usage

```
simulate_mills_curves(
  x_range = c(0.5, 5),
  n_points = 100,
  distributions = c("normal", "t30", "exponential"),
  log_scale = FALSE,
  include_asymptotic = FALSE
)
```

Arguments

x_range	Numeric vector of two elements: c(min, max)
n_points	Number of points to evaluate (default = 100)
distributions	Character vector of distributions to include
log_scale	Logical; if TRUE, uses log-spaced x values
include_asymptotic	Logical; if TRUE, includes asymptotic approximations

Value

Data frame in long format suitable for plotting

Author(s)

John Gavin john.b.gavin@gmail.com Generate Mills Ratio Curves

Examples

```
# Generate curves for comparison
curves <- simulate_mills_curves(
  x_range = c(0.5, 5),
  distributions = c("normal", "t3", "t30", "exponential")
)
```

Index