Package 'gpqyouden'

November 6, 2026

Title Generalized Pivotal Quantity Inference for the Youden Index

Version 0.1.0

Description Monte Carlo simulation study of generalized pivotal quantity (GPQ) based confidence intervals for the Youden index and its optimal cut-point, following Lai, Tian and Schisterman (2012) <doi:10.1016/j.csda.2010.11.023>.

License MIT + file LICENSE

URL https://github.com/jolmedilla/gpqyouden

BugReports https://github.com/jolmedilla/gpqyouden/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports parallel, kableExtra, dplyr, tidyr, stats, utils

Suggests foreach, doParallel, roxygen2, testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Juan José Olmedilla Arregui [aut, cre]

Maintainer Juan José Olmedilla Arregui <juanjo@olmedilla.com>

Repository CRAN

Date/Publication 2026-11-06 16:50:08 UTC

Contents

calculate_mu1_for_dataframe 2
default_param_grid 3
dmixnorm 3
eq.J2 4
find_mu1_from_J2 5
find_mu1_from_J_mixture 5
generate_mixture_sample 6

generate_mixture_tables 7
generate_normal_tables 8
generate_param_grid 9
gpq_c 9
gpq_from_sample 10
gpq_J 11
gpq_mu 12
gpq_sigma 12
J2.theoretical 13
J_true_mixture 14
mix_sd 15
pmixnorm 15
run_one_row 16
run_parallel_grid 17
run_simulation 18

Index **20**

`calculate_mu1_for_dataframe` *Compute μ_1 for each row of a parameter data frame*

Description

Convenience wrapper that applies `find_mu1_from_J2` row-wise to a data frame containing columns `mu2`, `var1`, `var2` and `J`.

Usage

```
calculate_mu1_for_dataframe(df)
```

Arguments

`df`	A data frame with columns `mu2`, `var1`, `var2` and `J`.

Value

A data frame with columns `mu1`, `original_J` and `theoretical_J`.

Examples

```
df <- data.frame(mu2 = 0, var1 = 1, var2 = 1, J = c(0.4, 0.6))
calculate_mu1_for_dataframe(df)
```

default_param_grid *Build the default parameter grid used in Lai et al. (2012)*

Description

Returns the full factorial combination of variances, Youden index values, and sample-size pairs used in the simulation study of Lai et al. (2012): $\sigma_1^2 \in \{0.5, 1, 3, 5\}$, $\sigma_2^2 = 1$, $\mu_2 = 0$, $J \in \{0.2, 0.4, 0.6, 0.8, 0.9\}$, and eight (n_1, n_2) pairs.

Usage

```
default_param_grid()
```

Value

A data frame with 160 rows and columns `mu2`, `sigma1`, `sigma2`, `J`, `n1`, `n2`, `var1`, `var2`.

See Also

`generate_param_grid`, `run_parallel_grid`

Examples

```
grid <- default_param_grid()
dim(grid)
head(grid)
```

dmixnorm *Density of the contaminated normal mixture*

Description

Density of the contaminated normal mixture

Usage

```
dmixnorm(x, mu, sigma, p = 0.1)
```

Arguments

`x`	Numeric vector. Points at which to evaluate the density.
`mu`	Numeric. Mean of the distribution.
`sigma`	Numeric. Base standard deviation.
`p`	Numeric. Contamination probability (default: 0.1).

Value

A numeric vector of density values.

See Also

pmixnorm, generate_mixture_sample

Examples

```
dmixnorm(x = seq(-3, 3, by = 1), mu = 0, sigma = 1, p = 0.1)
```

---

eq.J2 *Equation whose root gives μ_1*

Description

Auxiliary function used by find_mu1_from_J2: evaluates $J(\mu_1) - J_{\text{target}}$.

Usage

```
eq.J2(x, parameters, J2, model)
```

Arguments

x	Numeric. Candidate value of μ_1.
parameters	Numeric vector $(\mu_2, \sigma_2, \sigma_1)$.
J2	Numeric. Target Youden index value.
model	Character. Currently only "norm" is supported.

Value

A numeric scalar.

Examples

```
# parameters = (mu2, sigma2, sigma1)
eq.J2(x = 1.68, parameters = c(0, 1, 1), J2 = 0.6, model = "norm")
```

find_mu1_from_J2 *Find μ_1 from the Youden index under the normal model*

Description

Numerically solves for the mean of the diseased population μ_1 such that the theoretical Youden index equals `J2`, given the remaining parameters.

Usage

```
find_mu1_from_J2(J2, mu2, sigma2, sigma1, model)
```

Arguments

`J2`	Numeric. Target Youden index value.
`mu2`	Numeric. Mean of the non-diseased population.
`sigma2`	Numeric. Standard deviation of the non-diseased population.
`sigma1`	Numeric. Standard deviation of the diseased population.
`model`	Character. Currently only `"norm"` is supported.

Value

A numeric scalar, or `NA_real_` if the root cannot be found.

See Also

`J2.theoretical`, `find_mu1_from_J_mixture`

Examples

```
find_mu1_from_J2(J2 = 0.6, mu2 = 0, sigma2 = 1, sigma1 = 1, model = "norm")
```

find_mu1_from_J_mixture *Find μ_1 from the Youden index under the mixture model*

Description

Numerically solves for μ_1 such that the true Youden index of two contaminated normal mixture distributions equals `J_target`.

Usage

```
find_mu1_from_J_mixture(
  J_target,
  mu2,
  sigma1,
  sigma2,
  p = 0.1,
  tol = 1e-06,
  max_expand = 60
)
```

Arguments

J_target	Numeric. Target Youden index value (must be in (0, 1)).
mu2	Numeric. Mean of the non-diseased population.
sigma1	Numeric. Base standard deviation of the diseased population.
sigma2	Numeric. Base standard deviation of the non-diseased population.
p	Numeric. Contamination probability (default: 0.1).
tol	Numeric. Root-finding tolerance (default: 1e-6).
max_expand	Integer. Maximum number of bracket-expansion iterations (default: 60).

Value

A numeric scalar, or NA_real_ if the root cannot be found.

See Also

find_mu1_from_J2, J_true_mixture

Examples

```
find_mu1_from_J_mixture(J_target = 0.6, mu2 = 0, sigma1 = 1, sigma2 = 1, p = 0.1)
```

---

generate_mixture_sample

Generate a sample from the contaminated normal mixture

Description

Draws n observations from the mixture distribution used in the robustness study of Lai et al. (2012):

$$Y \sim (1 - B)\, N(\mu,\, \sigma_a^2) + B\, N(\mu,\, \sigma_b^2)$$

where $B \sim \text{Bernoulli}(p)$, $\sigma_a^2 = \sigma^2/1.1$ and $\sigma_b^2 = 2\sigma^2/1.1$.

Usage

```
generate_mixture_sample(n, mu, sigma, p = 0.1)
```

Arguments

n	Integer. Number of observations to generate.
mu	Numeric. Mean of the distribution.
sigma	Numeric. Base standard deviation (before scaling).
p	Numeric. Contamination probability (default: 0.1).

Value

A numeric vector of length n.

See Also

dmixnorm, pmixnorm

Examples

```
set.seed(1)
y <- generate_mixture_sample(n = 50, mu = 0, sigma = 1, p = 0.1)
summary(y)
```

---

generate_mixture_tables

Generate LaTeX tables for the mixture model simulation

Description

Produces two wide-format tables (coverage probability only, with σ_1^2 as column headers), one for the Youden index and one for the cut-point. These correspond to Tables 3 and 4 of Lai et al. (2012) for $p = 0.1$, and to the analogous tables for other values of p.

Usage

```
generate_mixture_tables(results_df, p, out_dir)
```

Arguments

results_df	Data frame as returned by run_parallel_grid with generator = "mixture".
p	Numeric. Contamination probability used in the simulation (included in captions and file names).
out_dir	Character. Directory where the .tex files are written.

Value

Invisibly NULL. Writes table_J_mixture_p=<p>.tex and table_c_mixture_p=<p>.tex to out_dir.

Examples

```
grid <- default_param_grid()[1:2, ]
results <- run_parallel_grid(grid, generator = "mixture",
                             n_mc = 10, K = 30, p = 0.1)
generate_mixture_tables(results, p = 0.1, out_dir = tempdir())
```

---

generate_normal_tables

Generate LaTeX tables for the normal model simulation

Description

Produces two long-format tables (coverage probability and average length, one for the Youden index and one for the cut-point), with σ_1^2 as the first grouping column. These correspond to Tables 1 and 2 of Lai et al. (2012).

Usage

```
generate_normal_tables(results_df, out_dir)
```

Arguments

results_df	Data frame as returned by run_parallel_grid with generator = "normal".
out_dir	Character. Directory where the .tex files are written.

Value

Invisibly NULL. Writes table_J_normal.tex and table_c_normal.tex to out_dir.

Examples

```
grid <- default_param_grid()[1:2, ]
results <- run_parallel_grid(grid, n_mc = 10, K = 30)
generate_normal_tables(results, out_dir = tempdir())
```

generate_param_grid *Build a custom parameter grid*

Description

Constructs a data frame of all combinations of the supplied parameter vectors and sample-size pairs, adding derived standard deviation columns.

Usage

```
generate_param_grid(var1, var2, mu2, J, n_pairs)
```

Arguments

var1	Numeric vector. Variances for the diseased population.
var2	Numeric vector. Variances for the non-diseased population.
mu2	Numeric vector. Means for the non-diseased population.
J	Numeric vector. Target Youden index values.
n_pairs	Integer matrix with two columns (n1, n2).

Value

A data frame with columns mu2, sigma1, sigma2, J, n1, n2, var1, var2.

See Also

default_param_grid

Examples

```
n_pairs <- matrix(c(10, 10, 20, 20), ncol = 2, byrow = TRUE)
generate_param_grid(var1 = c(1, 2), var2 = 1, mu2 = 0,
                    J = c(0.5, 0.7), n_pairs = n_pairs)
```

gpq_c *GPQ for the optimal cut-point*

Description

Computes the generalized pivotal quantity for the optimal cut-point (Equations 10-11 in Lai et al., 2012).

Usage

```
gpq_c(R_mu1, R_mu2, R_sigma1, R_sigma2, tol = 0.01)
```

Arguments

R_mu1	Numeric. GPQ realization for the mean of the diseased group.
R_mu2	Numeric. GPQ realization for the mean of the non-diseased group.
R_sigma1	Numeric. GPQ realization for the SD of the diseased group.
R_sigma2	Numeric. GPQ realization for the SD of the non-diseased group.
tol	Numeric. Tolerance for the equal-variance case (default: 1e-2).

Value

A numeric scalar.

Examples

```
gpq_c(R_mu1 = 2, R_mu2 = 0, R_sigma1 = 1, R_sigma2 = 1.2)
```

---

gpq_from_sample *Generate generalized pivotal quantities for one simulated sample*

Description

Implements the GPQ algorithm of Lai et al. (2012) for a single dataset drawn from two normal populations, producing Monte Carlo realizations of the GPQs for the Youden index and its optimal cut-point.

Usage

```
gpq_from_sample(y1, y2, K = 250, tol = 0.01)
```

Arguments

y1	Numeric vector. Observed sample from the diseased population.
y2	Numeric vector. Observed sample from the non-diseased population.
K	Integer. Number of GPQ iterations (default: 250).
tol	Numeric. Tolerance for the equal-variance case in gpq_c (default: 1e-2).

Value

A list with two numeric vectors of length K:

RJ GPQ realizations for the Youden index.

Rc GPQ realizations for the optimal cut-point.

References

Lai, C.-Y., Tian, L., & Schisterman, E. F. (2012). Exact confidence interval estimation for the Youden index and its corresponding optimal cut-point. *Computational Statistics & Data Analysis*, 56(5), 1103-1114.

Examples

```
set.seed(1)
y1 <- rnorm(30, mean = 2, sd = 1)
y2 <- rnorm(30, mean = 0, sd = 1)
gpq <- gpq_from_sample(y1, y2, K = 100)
quantile(gpq$RJ, probs = c(0.025, 0.975))
```

---

gpq_J *GPQ for the Youden index*

Description

Computes the generalized pivotal quantity for the Youden index (Equation 12 in Lai et al., 2012).

Usage

```
gpq_J(R_mu1, R_mu2, R_sigma1, R_sigma2, Rc)
```

Arguments

R_mu1	Numeric. GPQ realization for the mean of the diseased group.
R_mu2	Numeric. GPQ realization for the mean of the non-diseased group.
R_sigma1	Numeric. GPQ realization for the SD of the diseased group.
R_sigma2	Numeric. GPQ realization for the SD of the non-diseased group.
Rc	Numeric. GPQ realization for the optimal cut-point.

Value

A numeric scalar in $(-1, 1)$.

Examples

```
gpq_J(R_mu1 = 2, R_mu2 = 0, R_sigma1 = 1, R_sigma2 = 1.2, Rc = 1)
```

gpq_mu *GPQ for the mean*

Description

Computes the generalized pivotal quantity for the mean of one population (Equation 7 in Lai et al., 2012).

Usage

```
gpq_mu(ybar, s, n, t_value)
```

Arguments

ybar	Numeric. Observed sample mean.
s	Numeric. Observed sample standard deviation.
n	Integer. Sample size.
t_value	Numeric. Realization of a Student's t with n - 1 degrees of freedom.

Value

A numeric scalar.

Examples

```
gpq_mu(ybar = 5, s = 1.2, n = 30, t_value = 0.5)
```

gpq_sigma *GPQ for the standard deviation*

Description

Computes the generalized pivotal quantity for the standard deviation of one population (Equation 8 in Lai et al., 2012).

Usage

```
gpq_sigma(s2, v, n)
```

Arguments

s2	Numeric. Observed sample variance.
v	Numeric. Realization of a chi-squared with n - 1 degrees of freedom.
n	Integer. Sample size.

Value

A numeric scalar (standard deviation scale).

Examples

```
gpq_sigma(s2 = 1.5, v = 28.3, n = 30)
```

J2.theoretical *Theoretical Youden index under the normal model*

Description

Computes the true Youden index for two normal populations given their means and standard deviations.

Usage

```
J2.theoretical(parameters, model)
```

Arguments

`parameters`	Numeric vector or matrix with four columns in the order $(\mu_2, \mu_1, \sigma_2, \sigma_1)$. If a vector is supplied it is coerced to a one-row matrix.
`model`	Character. Currently only `"norm"` is supported.

Value

A numeric vector of Youden index values, one per row of `parameters`.

See Also

`find_mu1_from_J2`

Examples

```
# (mu2, mu1, sigma2, sigma1)
J2.theoretical(c(0, 1.68, 1, 1), model = "norm")
```

`J_true_mixture` *True Youden index and optimal cut-point under the mixture model*

Description

Numerically maximises the Youden index over all possible cut-points for two contaminated normal mixture distributions.

Usage

```
J_true_mixture(mu1, mu2, sigma1, sigma2, p = 0.1)
```

Arguments

`mu1`	Numeric. Mean of the diseased population.
`mu2`	Numeric. Mean of the non-diseased population.
`sigma1`	Numeric. Base standard deviation of the diseased population.
`sigma2`	Numeric. Base standard deviation of the non-diseased population.
`p`	Numeric. Contamination probability (default: 0.1).

Value

A list with two elements:

J The maximum Youden index.

c The optimal cut-point that achieves it.

See Also

`find_mu1_from_J_mixture`

Examples

```
J_true_mixture(mu1 = 1.68, mu2 = 0, sigma1 = 1, sigma2 = 1, p = 0.1)
```

mix_sd *Standard deviations of the two mixture components*

Description

Returns the standard deviations of the non-contaminated and contaminated components of the mixture as defined in Lai et al. (2012).

Usage

```
mix_sd(sigma)
```

Arguments

sigma	Numeric. Base standard deviation.

Value

A named numeric vector with elements sd_a ($\sigma/\sqrt{1.1}$) and sd_b ($\sigma\sqrt{2/1.1}$).

Examples

```
mix_sd(sigma = 1)
```

pmixnorm *CDF of the contaminated normal mixture*

Description

CDF of the contaminated normal mixture

Usage

```
pmixnorm(x, mu, sigma, p = 0.1)
```

Arguments

x	Numeric vector. Points at which to evaluate the CDF.
mu	Numeric. Mean of the distribution.
sigma	Numeric. Base standard deviation.
p	Numeric. Contamination probability (default: 0.1).

Value

A numeric vector of cumulative probabilities.

See Also

dmixnorm, generate_mixture_sample

Examples

```
pmixnorm(x = 0, mu = 0, sigma = 1, p = 0.1)
```

---

run_one_row *Run the simulation for a single row of the parameter grid*

Description

Convenience wrapper around run_simulation that accepts a single-row data frame (as returned by subsetting the output of default_param_grid) and returns a one-row summary data frame.

Usage

```
run_one_row(
  row,
  generator = c("normal", "mixture"),
  n_mc = 200,
  K = 250,
  p = 0.1
)
```

Arguments

row	A one-row data frame with columns mu2, sigma1, sigma2, J, n1, n2, var1, var2.
generator	Character. "normal" or "mixture".
n_mc	Integer. Number of Monte Carlo replications (default: 200).
K	Integer. Number of GPQ iterations per replication (default: 250).
p	Numeric. Contamination probability. Ignored when generator = "normal" (default: 0.1).

Value

A one-row data frame with columns mu2, var1, var2, J, n1, n2, coverage_J, length_J, coverage_c, length_c, generator, p.

See Also

run_simulation, run_parallel_grid

Examples

```
row <- default_param_grid()[1, ]
set.seed(1)
run_one_row(row, n_mc = 10, K = 30)
```

`run_parallel_grid` *Run the simulation in parallel, detecting the OS automatically*

Description

Uses `mclapply` on Unix/macOS and `doParallel+foreach` on Windows.

Usage

```
run_parallel_grid(
  df_params,
  generator = c("normal", "mixture"),
  n_mc = 200,
  K = 250,
  p = 0.1,
  n_cores = getOption("mc.cores", 2L)
)
```

Arguments

`df_params`	Data frame of parameter combinations (from `default_param_grid` or `generate_param_grid`).
`generator`	Character. `"normal"` or `"mixture"`.
`n_mc`	Integer. Number of Monte Carlo replications.
`K`	Integer. Number of GPQ iterations per replication.
`p`	Numeric. Mixture proportion. Ignored when `generator = "normal"`; passed through to `run_one_row` but has no effect on the normal simulation.
`n_cores`	Integer. Number of parallel workers to use. Defaults to `getOption("mc.cores", 2L)`, capped at the number of cores actually detected; pass a higher value explicitly to use more (CRAN policy caps the *default* number of cores a package may use at 2, but the caller may always opt into more).

Value

A data frame with one row per parameter combination.

Examples

```
grid <- default_param_grid()[1:2, ]
run_parallel_grid(grid, n_mc = 10, K = 30)
```

`run_simulation` *Run the Monte Carlo simulation for a single parameter scenario*

Description

Repeatedly draws paired samples under the normal or contaminated-mixture model, computes GPQ-based confidence intervals for the Youden index and its optimal cut-point via `gpq_from_sample`, and summarises empirical coverage and mean interval length over `n_mc` replications.

Usage

```
run_simulation(
  mu2,
  sigma1,
  sigma2,
  J,
  n1,
  n2,
  n_mc = 200,
  K = 250,
  alpha = 0.05,
  generator = c("normal", "mixture"),
  p = 0.1
)
```

Arguments

`mu2`	Numeric. Mean of the non-diseased population.
`sigma1`	Numeric. Standard deviation of the diseased population.
`sigma2`	Numeric. Standard deviation of the non-diseased population.
`J`	Numeric. Target (nominal) Youden index.
`n1`	Integer. Sample size for the diseased population.
`n2`	Integer. Sample size for the non-diseased population.
`n_mc`	Integer. Number of Monte Carlo replications (default: 200).
`K`	Integer. Number of GPQ iterations per replication, passed to `gpq_from_sample` (default: 250).
`alpha`	Numeric. Significance level for the confidence intervals (default: 0.05, giving 95% intervals).
`generator`	Character. `"normal"` or `"mixture"`.
`p`	Numeric. Contamination probability, used only when `generator = "mixture"` (default: 0.1).

Value

A list with elements `params` (the scenario's parameters, including the derived `mu1`), `J` and `c`, each a list with `coverage` and `mean_length`.

See Also

`run_one_row`, `run_parallel_grid`

Examples

```
set.seed(1)
run_simulation(mu2 = 0, sigma1 = 1, sigma2 = 1, J = 0.6,
                n1 = 20, n2 = 20, n_mc = 20, K = 50)
```

Index

calculate_mu1_for_dataframe, 2

default_param_grid, 3, *9*, *16*
dmixnorm, 3, *7*, *16*

eq.J2, 4

find_mu1_from_J2, *2*, *4*, 5, *6*, *13*
find_mu1_from_J_mixture, *5*, 5, *14*

generate_mixture_sample, *4*, 6, *16*
generate_mixture_tables, 7
generate_normal_tables, 8
generate_param_grid, *3*, 9
gpq_c, 9, *10*
gpq_from_sample, 10, *18*
gpq_J, 11
gpq_mu, 12
gpq_sigma, 12

J2.theoretical, *5*, 13
J_true_mixture, *6*, 14

mix_sd, 15

pmixnorm, *4*, *7*, 15

run_one_row, 16, *19*
run_parallel_grid, *3*, *16*, 17, *19*
run_simulation, *16*, 18