Package: gst (via r-universe)

November 7, 2026

Title Global Statistical Tests for Multiple Endpoints

Version 0.4.0

Description Design and evaluate global statistical tests (GST) that combine several endpoints into one test, including O'Brien's ordinary and generalized least squares tests (O'Brien (1984) <doi:10.2307/2531158>), power and sample size calculations, and an O'Brien-based truncated closed testing procedure (Luo et al. (2025) <doi:10.1002/pst.70010>). Two 'shiny' applications support interactive power calculation and the creation of forest plots for multiple endpoints from summary statistics.

License Apache License (== 2.0)

URL https://xni7.github.io/gst/, https://github.com/xni7/gst

BugReports https://github.com/xni7/gst/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, forestploter, magrittr, MASS, multcomp, stats, parallel, readr, rhandsontable, shiny, shinyWidgets, htmltools, tidyr, plotly, grDevices, grid, pwr, DT, purrr, tibble, bslib, bsicons, datapasta, sortable, stringr, rclipboard, ggplot2, mvtnorm, reactable, rlang

Suggests knitr, rmarkdown, gt, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Lerong Li [aut, led], Anthony Pileggi [aut], Xiaolong Luo [aut], Xiao Ni [aut, cre], Sarepta Therapeutics Inc. [cph, fnd], Stephen Knapp [ctb] (ORCID: <https://orcid.org/0000-0002-5101-4555>), Preston Burns [ctb]

Maintainer Xiao Ni <mooncase@gmail.com>

Repository https://cran.r-universe.dev
Date/Publication 2026-11-07 21:00:28 UTC
RemoteUrl https://github.com/cran/gst
RemoteRef HEAD
RemoteSha d85276b3ae60c2d6961f238af7bb902a783c7153

Contents

cor_cs . . . 2
forest_data . . . 3
forest_plot . . . 3
forest_reprex . . . 6
get_n . . . 7
get_oes . . . 8
get_power . . . 8
gst_power_app . . . 9
make_forestplot_app . . . 9
obrien . . . 10
obrien_plus . . . 11
obt . . . 13
sim_power . . . 15

Index **16**

`cor_cs` *Create a compound-symmetric correlation matrix*

Description

Create a compound-symmetric correlation matrix

Usage

```
cor_cs(r, m = 2)
```

Arguments

`r`	rho, correlation coefficient (numeric/scalar)
`m`	number of rows/columns (numeric/scalar)

Value

An `m` x `m` numeric matrix with 1 on the diagonal and `r` elsewhere.

Examples

```
cor_cs(0.5, m = 3)
```

forest_data *Forestplot Example Data*

Description

Forestplot Example Data

Usage

```
forest_data()
```

Value

A data.frame that can be passed to forest_plot

Examples

```
head(forest_data())
```

forest_plot *Draw forest plot*

Description

Draw forest plot

Usage

```
forest_plot(
  dt,
  indent_subgroups = TRUE,
  indent_keyword = "Overall",
  standardize_plot = FALSE,
  flip_arrows = FALSE,
  font_size = NULL,
  footnote_size = NULL,
  plot_pch = 16,
  plot_point_size = NULL,
  plot_zoom = 80,
  plot_x_ticks = NULL,
  plot_symmetric = TRUE,
  ci_t_height = 0.2,
  ci_color = "#000000",
  ci_color_inverted = "#4226bd",
  row_height = NULL,
  color_by_endpoint = FALSE,
```

```
  significance_level = 0.05,
  display_cols = NULL,
  display_widths = NULL,
  forest_label = NULL,
  digits = 2,
  arrow_prefix = "Favors",
  arrow_label_left = NULL,
  arrow_label_right = NULL,
  ...
)
```

Arguments

dt	input data (columns: Endpoint, Group, Comparator, Control, LSMean, low, hi, p, invert)
indent_subgroups	indent subgroup labels (logical/scalar; default = TRUE)
indent_keyword	indent keyword (is this useful)
standardize_plot	normalize/standardize point estimate and CI plot display (logical/scalar)
flip_arrows	flip directional arrows below forestplot (logical/scalar)
font_size	font size (numeric/scalar)
footnote_size	footnote font size (numeric/scalar)
plot_pch	plotting character (numeric/scalar)
plot_point_size	size of plotted point estimate (numeric/scalar)
plot_zoom	zoom level for forestplot (numeric/scalar, range: 0 - 100)
plot_x_ticks	vector of x-axis tick marks for forestplot (numeric/vector)
plot_symmetric	should the forestplot have a symmetric x-axis? (logical/scalar)
ci_t_height	size of confidence interval T endpoint (numeric/scalar; range: 0 - 1)
ci_color	color used for normal forestplot confidence intervals (character/scalar; hex color code)
ci_color_inverted	color used for inverted forestplot confidence intervals (character/scalar; hex color code)
row_height	row height (numeric/scalar)
color_by_endpoint	add coloring for each endpoint (logical/scalar)
significance_level	significance level (alpha) for two-sided confidence interval
display_cols	names of columns to display, in order (character/vector; use NULL for defaults)
display_widths	column widths for the display_cols (numeric/vector; use NULL for defaults)
forest_label	label for forest plot column (character/scalar; use NULL for defaults)

digits — number of digits to round estimate + CI column to (scalar/integer; default = 2; use NULL for no rounding)

arrow_prefix — prefix to use for favor arrows (character/scalar; default = "Favors")

arrow_label_left — label for the left-facing arrow (character/scalar; default = NULL (header for control column))

arrow_label_right — label for the left-facing arrow (character/scalar; default = NULL (header for comparator column))

... — additional arguments passed on to forest_theme

Value

A gtable object

Input Data Requirements

The format of the input data dt is rigid and must include all columns in order, even when specifying the display_cols and only using a subset. See forest_data for example input data with the required columns in the correct order.

Customizing columns

Columns can be customized using the display_cols argument, a character vector with column names in the desired display order. In additional to columns in the input data (dt), two special column names are:

- est+ci: the estimate and confidence interval column
- forest: forestplot image column

Note that when including additional columns, they should be placed after the required columns in the input data (dt).

By default, if display_cols is non-NULL then all column widths will be set to equal. Use display_widths to further customize column widths; it should be a numeric vector with length equal to display_cols.

Examples

```
# draw forestplot
forest_data() |> forest_plot()

# standardize forest CIs
forest_data() |>
  forest_plot(
    standardize_plot = TRUE
  )

# custom x-ticks
forest_data() |>
```

```
  forest_plot(
    row_height = 1.75,
    plot_x_ticks = c(-1, 0, 1),
    plot_zoom = 100
  )

# custom columns
gst::forest_data() |>
dplyr::mutate(
  newcol1 = "A",
  newcol2 = "B"
) |>
gst::forest_plot(
  display_cols = c("Group", "forest", "est+ci", "p", "newcol1", "newcol2"),
  display_widths = c(0.15, 0.2, 0.15, 0.15, 0.1, 0.1)
)
```

`forest_reprex` *Create reproducible code for current forestplot*

Description

Create reproducible code for current forestplot

Usage

```
forest_reprex(forestplot, unfold = FALSE)
```

Arguments

`forestplot`	output from forest_plot()
`unfold`	unfold the `forest_plot` function? (logical/scalar; default = FALSE)

Value

A character vector of R code lines that recreate the plot.

Note

Unfolding the code (`unfold = TRUE`) will generate a code snippet that uses the `forestploter::forest()` and related package functions directly; the resulting code does not require the `gst` package.

WARNING: This function will not create a perfect reprex if you use the `...` argument in `forest_plot`

Examples

```
fp <- forest_plot(forest_data())
cat(forest_reprex(fp), sep = "\n")
```

get_n *Calculate sample size*

Description

Calculate sample size

Usage

```
get_n(
  power,
  es0,
  es1,
  R,
  alpha = 0.05,
  k,
  method = c("ols", "gls"),
  df_method = c("vd", "vob", "vlt"),
  n0_range = c(2, 1000),
  tolerance = 1e-06
)
```

Arguments

power	target power
es0	effect size in endpoint 1
es1	effect size in endpoint 2
R	correlation matrix between endpoints
alpha	significance level
k	sample size ratio of treatment vs placebo
method	obrien "ols", "gls"
df_method	degrees-of-freedom method (one of "vd", "vob", "vlt")
n0_range	range of size in placebo to be searched
tolerance	the desired accuracy (convergence tolerance)

Value

A list with the required sample sizes `n0` (placebo) and `n1` (treatment), each rounded up to a whole number.

Examples

```
get_n(power = 0.8, es0 = 0.3, es1 = 0.4, R = cor_cs(0.5), k = 1)
```

get_oes *Calculate operational effect size*

Description

Calculate operational effect size

Usage

```
get_oes(lambda, R, method = c("ols", "gls"))
```

Arguments

lambda	vector of effect size for each endpoint
R	correlation matrix of lambda
method	obrien "ols", "gls"

Value

A numeric scalar: the operational (combined) effect size.

Examples

```
get_oes(lambda = c(0.3, 0.4), R = cor_cs(0.5))
```

get_power *Calculate Power*

Description

Calculate Power

Usage

```
get_power(oes, n0, n1 = n0, df = n0 + n1 - 2, alpha = 0.05)
```

Arguments

oes	operational effect size
n0	sample size in placebo group
n1	sample size in treatment group
df	degree of freedom
alpha	significance level

Value

A numeric scalar: the two-sided power of the t-test.

Examples

```
get_power(oes = 0.35, n0 = 50)
```

---

gst_power_app *Launch the GST Power App*

Description

Launch the GST Power App

Usage

```
gst_power_app(runApp = TRUE)
```

Arguments

runApp	should the app be run on completion, default is TRUE

Value

With runApp = FALSE, a Shiny app object (class shiny.appobj). With runApp = TRUE (the default), the app is launched and the function returns the value of shiny::runApp() when the app stops.

Examples

```
app <- gst_power_app(runApp = FALSE)
if (interactive()) {
  gst_power_app()
}
```

---

make_forestplot_app *A Shiny app to make a forest plot*

Description

A Shiny app to make a forest plot

Usage

```
make_forestplot_app(runApp = TRUE, plot_width = 1000, plot_height = 500)
```

Arguments

runApp	should the app be run on completion, default is TRUE
plot_width	Width of the plot in pixels, default is 1200
plot_height	Height of the plot in pixels, default is 800

Value

With runApp = FALSE, a Shiny app object (class shiny.appobj). With runApp = TRUE (the default), the app is launched and the function returns the value of shiny::runApp() when the app stops.

Examples

```
app <- make_forestplot_app(runApp = FALSE)
if (interactive()) {
  make_forestplot_app()
}
```

---

obrien *O'Brien test*

Description

O'Brien test

Usage

```
obrien(
  data,
  endpoint_vars,
  treatment_vars,
  method = c("ols", "gls", "ranksum"),
  var.equal = TRUE,
  ...
)
```

Arguments

data	dataset with treatment and multiple outcomes in wide format
endpoint_vars	the character vectors of outcomes in the dataset dat
treatment_vars	treatment variable in the dataset dat
method	obrien "ols", "gls" and "ranksum" test
var.equal	default TRUE. the obrien paper assumes equal variance and using pooled variance to standardize data
...	other options passed to t.test and wilcox.test

Value

A list with t-statistic, df, pvalue, alternative, and correlation matrix R

Examples

```
# prepare example data (from multcomp::mtept)
dat <- multcomp::mtept
dat$E4 <- -dat$E4  # align signs for all endpoint

# OLS method
dat |>
   obrien(
     endpoint_vars = c("E1", "E2", "E3", "E4"),
     treatment_vars = "treatment",
     method = "ols"
     )

# GLS one-sided
dat |>
  obrien(
    endpoint_vars = c("E1", "E2", "E3"),
    treatment_vars = "treatment",
    method = "gls",
    alternative = "less"
    )

# Ranksum method
dat |>
  obrien(
    endpoint_vars = c("E1", "E2", "E3", "E4"),
    treatment_vars = "treatment",
    method = "ranksum"
    )
```

`obrien_plus` *O'Brien Plus: Analytical Power for Joint AND Any Individual Endpoint*

Description

This function analytically estimates the statistical power for the scenario where both the O'Brien joint endpoint test is significant AND at least one of the individual endpoints is significant. It also calculates the power for "Joint AND a Specific Individual" (if `ind_index` is provided). It relies on a multivariate t-distribution approximation of the test statistics.

Usage

```
obrien_plus(
  n1,
  n2 = n1,
  df = n1 + n2 - 2,
  es_ind,
  cor_mtx,
  ind_index = NULL,
  method = c("ols", "gls"),
  alpha = 0.025,
  one_sided = TRUE
)
```

Arguments

n1	Numeric, sample size for the first treatment group.
n2	Numeric, sample size for the second treatment group.
df	Numeric, degrees of freedom for the t-distribution approximation (default n1 + n2 - 2).
es_ind	Numeric vector, the standardized mean differences (Cohen's d) for each individual endpoint under the alternative hypothesis.
cor_mtx	Numeric matrix, the correlation matrix (R) among the standardized individual endpoints.
ind_index	Integer, the 1-based index of a specific individual endpoint for which "Joint AND Specific Individual" power is also calculated. Set to NULL if only "Joint AND Any Individual" power is desired.
method	Character, the O'Brien test method to use ("ols" or "gls").
alpha	Numeric, the significance level for each test (e.g., 0.05). Assumes no explicit multiplicity adjustment for the 'AND' or 'OR' conditions.
one_sided	Logical, default TRUE. If TRUE, a one-sided test is assumed. If FALSE, a two-sided test is used, with a warning about power interpretation for positive effect sizes.

Value

a named list containing computation settings and results, including:

- power_joint_any - power for joint endpoints and any additional endpoint
- power_joint_one - power for joint endpoints and a specific endpoint (specified by ind_index)
- power_joint - power for the joint endpoint test
- power_ind - power for the specific endpoint (specified by ind_index)
- n1 - sample size for the first treatment group
- n2 - sample size for the second treatment group
- alpha - significance level
- method - the O'Brien test method used
- one_sided - logical indicating if the test is one-sided

Examples

```
# Generate GST power for 3 endpoints (joint + any)
obrien_plus(
  n1 = 25,
  n2 = 30,
  es_ind = c(1, 0.5, 0.5),
  cor_mtx = cor_cs(0.25, m = 3),
  ind_index = NULL,
  method = "ols",
  alpha = 0.05,
  one_sided = TRUE
  )

# Generate GST power for 4 endpoints (joint + 3rd endpoint)
obrien_plus(
  n1 = 28,
  n2 = 40,
  es_ind = c(1, 0.25, 0.75, 0.5),
  cor_mtx = cor_cs(0.35, m = 4),
  ind_index = 3,
  method = "ols",
  alpha = 0.05,
  one_sided = TRUE
  )
```

---

obt *Perform Obrien truncated closed testing*

Description

Perform Obrien truncated closed testing

Usage

```
obt(
  data,
  endpoint_vars,
  jendpoint = NULL,
  treatment_vars,
  alpha = 0.05,
  nu = 0.8,
  method = c("ols", "gls", "ranksum"),
  var.equal = TRUE,
  ...
)
```

Arguments

data	input data; columns should contain endpoint data and must be named E1, E2
endpoint_vars	vector of endpoint column names (character/vector)
jendpoint	joint endpoint, a vector of individual endpoints (character/vector)
treatment_vars	treatment column name (character/scalar)
alpha	significance level (numeric/scalar)
nu	truncation parameter, between 0 and 1 (numeric/scalar; default = 0.8)
method	estimation method (character/scalar; default = 'OLS')
var.equal	assume equal variances? (logical/scalar; default = TRUE)
...	additional arguments passed to obrien

Value

a list containing the following elements:

- F: target family of hypotheses (character/vector)
- df: data.frame with relevant hypotheses and parameters
- df_gst: data.frame with results from the different procedures
- alpha_A: unused alpha available for propagation to the next family

Input Data Requirements

The endpoint variables in data must be named E1, E2, E3, etc. for the function to work correctly. If you endpoint columns do not have this naming convention, you must rename them.

Note

This is the truncated closed test as proposed by: Luo, X., Li, L., Savenkov, O., Liu, W., Ni, X., Tang, W. and Guo, W., 2025. Multiple Comparisons Procedures for Analyses of Joint Primary Endpoints and Secondary Endpoints. Pharmaceutical Statistics, 24(3), p.e70010. doi: https://doi.org/10.1002/pst.70010

Examples

```
dat <- multcomp::mtept
dat$E4 <- -dat$E4  # align signs for all endpoints
obt(
    data = dat,
    endpoint_vars = c("E1","E2","E3","E4"),
    jendpoint = c("E3", "E4"),
    alpha = 0.05,
    nu = 0.6,
    treatment_vars = "treatment",
    method = "ols"
)
```

`sim_power` *Calculate Power by simulation*

Description

Calculate Power by simulation

Usage

```
sim_power(
  n0,
  n1 = n0,
  lambda,
  R,
  alpha = 0.05,
  N = 1000,
  ncore = 1L,
  method = c("ols", "gls"),
  ...,
  verbose = FALSE
)
```

Arguments

`n0`	sample size in placebo group
`n1`	sample size in treatment group
`lambda`	vector of effect size for each endpoint
`R`	correlation matrix of lambda
`alpha`	significance level
`N`	number of replication
`ncore`	number of cores for `parallel::mclapply()` (default 1; values above 1 are not supported on Windows)
`method`	obrien "ols", "gls"
`...`	additional arguments passed to `obrien`
`verbose`	print progress every 100 replications? (logical/scalar; default = FALSE)

Value

A numeric scalar: the proportion of simulated trials with an O'Brien test p-value at or below `alpha`.

Examples

```
sim_power(n0 = 30, lambda = c(0.3, 0.4), R = cor_cs(0.5), N = 20, ncore = 1)
```

Index

cor_cs, 2

forest_data, 3, *5*
forest_plot, 3, *3*, *6*
forest_reprex, 6
forest_theme, *5*
forestploter::forest(), *6*

get_n, 7
get_oes, 8
get_power, 8
gst_power_app, 9
gtable, *5*

make_forestplot_app, 9

obrien, 10, *14, 15*
obrien_plus, 11
obt, 13

parallel::mclapply(), *15*

shiny::runApp(), *9, 10*
sim_power, 15

t.test, *10*

wilcox.test, *10*