Package ‘loewesadditivity’

May 8, 2026

Title Loewe's Additivity

Version 0.1.0

Description Estimate model parameters to determine whether two compounds have synergy, antagonism, or Loewe's Additivity.

License MIT + file LICENSE

Depends R (>= 3.1.0)

Encoding UTF-8

LazyData true

Suggests testthat (>= 2.1.0), knitr, rmarkdown, kableExtra, devtools

Imports dplyr, tidyr, magrittr, rlang, ggplot2, metR, gridExtra, rootSolve, viridis

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Author Shannon Gallagher [aut, cre],
Michael Fay [aut]

Maintainer Shannon Gallagher `<skgallagher19@gmail.com>`

Repository CRAN

Date/Publication 2020-03-24 17:00:07 UTC

Contents

`base_GIA` *Estimate GIA according to the base model*

Description

Estimate GIA according to the base model

Usage

```
base_GIA(model_params, dose_A, dose_B, fn_list = NULL)
```

Arguments

`model_params`	named vector of parameters to be used in function. Specifically, the named parameters must be "beta_A", "beta_B", "gamma_A", "gamma_B", "tau_1", and "tau_2". See details for more info.
`dose_A`	numeric vector of doses (e.g. mg/mL) of dose_A
`dose_B`	numeric vector of doses (e.g. mg/mL) of dose_B
`fn_list`	NULL

Value

estimated GIA for each combination of dose A and dose B

Details

The equation is given in full as follows. The GIA (%) is given a as a function of the model parameters and the doses A_i and B_i, respectively. The doses scaled by the respective ED50s β_A and β_B are denoted by A_i^* and B_i^*, respectively. The parameters γ_A and γ_B are shape parameters. The parameters τ_1 and τ_2 are interaction parameters. Finally, λ_i is a weighted combination of dose A and dose B.

$$GIA_i = 100\%(1 - e^{-\psi_i})$$

$$\psi_i = \log(2)u_i^{v_i}$$

$$u_i = A_i^* + B_i^* + \tau_1 A_i^* B_i^*$$

$$
\begin{aligned}
v_i &= \lambda_i \gamma_A + (1-\lambda_i)\gamma_B + \tau_1\tau_2\lambda_i(1-\lambda_i)\gamma_A\gamma_B \\
\lambda_i &= \frac{A_i^*}{A_i^* + B_i^*} \\
A_i^* &= A_i/\beta_A \\
B_i^* &= B_i/\beta_B
\end{aligned}
$$

Examples

```
model_params <- c("beta_A" = 1, "beta_B" = 2, "gamma_A" = .5,
"gamma_B" = .6,  "tau_1" = 1, "tau_2" = 0)
dose_A <- c(0, 1, 0)
dose_B <- c(0, 0, 1)
base_GIA(model_params, dose_A, dose_B)
```

`boot_GIA` *Helper function for the bootstrap results*

Description

Helper function for the bootstrap results

Usage

```
boot_GIA(
  par,
  gia_df,
  gia_est,
  n_boot = 100,
  alpha = 0.05,
  GIA_fn = base_GIA,
  S_fn = calc_S_base,
  fn_list = NULL,
  verbose = FALSE
)
```

Arguments

`par`	named vector of parameters, that correspond to those used in 'GIA_fn'.
`gia_df`	data frame with the following columns • dose_Adose A mg/mL • dose_Bdose B mg/mL • GIAGIA
`gia_est`	estimated values of GIA (these will be used as the 'truth')
`n_boot`	number of boot straps to use to estimate confidence intervals of the parameters, GIA estimates, and values of S. The default is 100. If n_boot = 0, then no bootstraps will be run and only the point estimates will be returned.

`alpha`	value of alpha. Default is .05
`GIA_fn`	function to calculate the GIA from dose_A and dose_B combinations and given set of parameters. Default is base_GIA
`S_fn`	Function to calculate S. Default is calc_S_base
`fn_list`	additional arguments to pass to GIA_fn
`verbose`	logical indicating whether we should print where we are in the process. Default is FALSE.

Value

a list with the following elements

- params_esta data frame of dimension # of params x 4 where each row in the data frame is a parameter and where the columns are the mean, lower, alpha/2 quantile,and upper,100 - alpha/2 quantile
- S_est a data frame of one row x 4 where we provide the mean, lower, and upper estimates
- GIA_estthe original data with additional columns of the mean, lower, and upper estimates for each dose combination

---

`calc_S` *Calculate S generally*

Description

Calculate S generally

Usage

```
calc_S(best_pars, S_fn = calc_S_base, fn_list = NULL)
```

Arguments

`best_pars`	named vector of parameters. "tau_1" must be a name. As must "tau_2" and "gamma_A" and "gamma_B"
`S_fn`	function to calculate
`fn_list`	NULL

Value

Hewlett's S for the given model

Examples

```
best_pars <- c("tau_1" = 0,
               "tau_2" = 1,
               "gamma_A" = 1,
                "gamma_B" = 1)
calc_S_base(best_pars) # should be 1
```

`calc_S_base` *Calculate S from given tau_1 for base model*

Description

Calculate S from given tau_1 for base model

Usage

```
calc_S_base(best_pars, fn_list = NULL)
```

Arguments

`best_pars`	named vector of parameters. "tau_1" must be a name. As must "tau_2" and "gamma_A" and "gamma_B"
`fn_list`	NULL

Value

Hewlett's S for the base model.

Examples

```
best_pars <- c("tau_1" = 0,
              "tau_2" = 1,
              "gamma_A" = 1,
               "gamma_B" = 1)
calc_S_base(best_pars) # should be 1
```

`cyrpa_ripr` *CyRPA and RIPR*

Description

The data is the raw data for a combination dose of CyRPA and RIPR.

well one of iRBC (the max), uRBC (the min), RPMI (??), or comb (which is short for combination)

RIPR dose of RIPR in mg/mL

CyRPA dose of CyRPA in mg/mL

expxyrepz the results from experiment x, sub item y, repetition z

Usage

```
cyrpa_ripr
```

Format

An object of class `data.frame` with 38 rows and 15 columns.

Examples

```
data("cyrpa_ripr")
head(cyrpa_ripr)
```

`design_experiment` *Helper function to generate code to run an experiment*

Description

Helper function to generate code to run an experiment

Usage

```
design_experiment(
  levels_A = c(0, 1 * 2^(-4:2)),
  levels_B = c(0, 2 * 2^(-4:2)),
 par = c(beta_A = 1, beta_B = 2, gamma_A = 0.5, gamma_B = 0.5, tau_1 = 3, tau_2 = 0.05),
  n_rep = 1,
  n_sims = 100,
  noise_par = c(a0 = 3, a1 = 0.01)
)
```

Arguments

`levels_A`	levels of A used in the combination
`levels_B`	levels of B used in the combination
`par`	named vector of model parameters
`n_rep`	number of total repetitions of experiment
`n_sims`	number of simulations to run
`noise_par`	named vector with ’a0’ and ’a1’ which are used to generate noise for the GIA.

Details

prints out code to copy and paste into `R` to simulate the expected coverage of your experiment under your designed hypothesis

---

`design_grid` *Function to design an experimental grid of combinations*

Description

Function to design an experimental grid of combinations

Usage

```
design_grid(
  levels_A = c(0, 1 * 2^(-4:2)),
  levels_B = c(0, 2 * 2^(-4:2)),
  n_rep = 1
)
```

Arguments

`levels_A`	levels of A used in the combination
`levels_B`	levels of B used in the combination
`n_rep`	number of total repetitions of experiment

Value

data frame with columns dose_A, dose_B, and GIA for all possible combinations

---

`estimate_GIA` *Take in dose A and dose B combinations and estimate GIA*

Description

Take in dose A and dose B combinations and estimate GIA

Usage

```
estimate_GIA(model_params, dose_A, dose_B, fn = base_GIA, fn_list = NULL)
```

Arguments

`model_params`	named vector of parameters to be used in function
`dose_A`	numeric vector of doses (e.g. mg/mL) of dose_A
`dose_B`	numeric vector of doses (e.g. mg/mL) of dose_B
`fn`	the function used to calculate GIA. The default is base_GIA. See ?base_GIA for more details.
`fn_list`	additional parameters to pass to the function to estimate GIA

Value

vector of the same size of dose_A and dose_B where each entry is the estimated GIA for the combination of dose A and dose B

Examples

```
model_params <- c("beta_A" = 1, "beta_B" = 2, "gamma_A" = .5,
"gamma_B" = .6,  "tau_1" = 1, "tau_2" = 0)
dose_A <- c(0, 1, 0)
dose_B <- c(0, 0, 1)
estimate_GIA(model_params, dose_A, dose_B)
```

`estimate_params` *Estimate the parameters for a given data set and model*

Description

Estimate the parameters for a given data set and model

Usage

```
estimate_params(
  data,
 init_params = c(beta_A = 0.25, beta_B = 0.25, gamma_A = 0.5, gamma_B = 0.5, tau_1 = 0,
    tau_2 = 0),
  n_boot = 100,
  GIA_fn = base_GIA,
  S_fn = calc_S_base,
  fn_list = NULL,
  alpha = 0.05,
  verbose = FALSE
)
```

Arguments

`data`	data frame with the following columns • dose_Adose A mg/mL • dose_Bdose B mg/mL • GIAGIA
`init_params`	named vector of parameters, that correspond to those used in 'GIA_fn'. These will be used as the initial guesses. A default is provided.
`n_boot`	number of boot straps to use to estimate confidence intervals of the parameters, GIA estimates, and values of S. The default is 100. If n_boot = 0, then no bootstraps will be run and only the point estimates will be returned.
`GIA_fn`	function to calculate the GIA from dose_A and dose_B combinations and given set of parameters. Default is base_GIA

`S_fn`	Function to calculate S. Default is calc_S_base
`fn_list`	additional arguments to pass to GIA_fn
`alpha`	alpha level used to produce CIs. The bootstrap will use a two-tailed method. The default is .05 to produce a 95% CI
`verbose`	logical indicating whether we should print where we are in the process. Default is FALSE.

Value

a list with the following elements

- params_esta data frame of dimension # of params x 4 where each row in the data frame is a parameter and where the columns are the mean, lower, alpha/2 quantile, and upper,100 - alpha/2 quantile
- S_est a data frame of one row x 4 where we provide the mean, lower, and upper estimates
- GIA_estthe original data with additional columns of the mean, lower, and upper estimates for each dose combination
- SSESum of Square Error for the model under the best (mean) parameters

Examples

```
df <- loewesadditivity::cyrpa_ripr
df$dose_A <- df$CyRPA
df$dose_B <- df$RIPR
data <- fortify_gia_data(df)
model_params <- c("beta_A" = .5, "beta_B" = .5,
                 "gamma_A" = .5, "gamma_B" = .5,
                 "tau_1" = 0, "tau_2" = 0)
n_boot <- 10
GIA_fn <- base_GIA
S_fn <- calc_S_base
fn_list <- NULL
alpha <- .05
verbose <- FALSE
out <- estimate_params(data = data,
init_params = model_params,
n_boot = n_boot,
GIA_fn = GIA_fn,
S_fn = S_fn,
fn_list = fn_list,
alpha = alpha,
verbose = verbose)
names(out)
```

`fortify_gia_data` *Put GIA measurements into a dplyr format*

Description

Put GIA measurements into a dplyr format

Usage

```
fortify_gia_data(data)
```

Arguments

`data`	data frame of GIA measurements **well** one of "IRBC", "uRBC", "RPMI", or "comb" **dose_A** dose of A in mg/mL **dose_B** dose of B in mg/mL **exp(X)(Y)rep(Z)** where X = 1 or 2, Y = a or b, and Z = 1, 2, or 3

Value

long data frame with columns well, dose_A, dose_B, plate, exp_num (experiment number), plate (a or b), rep_num (repetition number), gia_mean, and average iRBC and uRBC

Examples

```
df <- loewesadditivity::rh5_ama1ron2
df$dose_A <- df$RH5
df$dose_B <- df$AMA1RON2
fortified_df <- fortify_gia_data(df)
head(fortified_df)
```

`get_ed_line` *Helper function to get the ED50 line*

Description

Helper function to get the ED50 line

Usage

```
get_ed_line(
  grid_width = 50,
  par,
  GIA_fn = base_GIA,
  fn_list = NULL,
  ed_val = 50
)
```

Arguments

`grid_width`	number of levels to find points at
`par`	named vector of parameters
`GIA_fn`	function to calculate GIA
`fn_list`	additional parameters to pass to GIA_fn
`ed_val`	Which line to compute. Default is 50

Value

data frame with the following columns

dose_A dose of A (unscaled)

dose_B dose of B (unscaled)

GIA value of GIA %

`make_grid` *Make a grid of points*

Description

Make a grid of points

Usage

```
make_grid(n = 40, par, Amax = 2, Bmax = 2, n_reps = 1)
```

Arguments

`n`	number of levels on each side (Total grid is n^2). Default is 40
`par`	named vector of model parameters
`Amax`	max amount of number of ED50s. Default is 2
`Bmax`	max amount of number of ED50s. Default is 2.
`n_reps`	number of replicates to repeat entire grid/experiment. Default is 1.

Value

data frame with the following columns

dose_A unscaled dose of A

dose_B unscaled dose o B

rep replicate number

Examples

```
n <- 40
 par <- c("beta_A" = 1, "beta_B" = 2)
out <- make_grid(n = 2, par = par)
exp_out <- data.frame(dose_A = c(0, 2, 0, 2),
                      dose_B = c(0, 0, 4, 4),
                      rep = 1)
```

`plot_curves` *Plot the surface and observations*

Description

Plot the surface and observations

Usage

```
plot_curves(
  est_list,
  dose_A = "Dose A",
  dose_B = "Dose B",
  title = "Curves of Dose Combos",
  subtitle = "",
  base_size = 14
)
```

Arguments

`est_list`	output from `estimate_params`
`dose_A`	to pass to ggplot
`dose_B`	to pass to ggplot
`title`	to pass to ggplot
`subtitle`	to pass to ggplot
`base_size`	to pass to ggplot

Value

ggplot object

Examples

```
df <- loewesadditivity::cyrpa_ripr
df$dose_A <- df$CyRPA
df$dose_B <- df$RIPR
data <- fortify_gia_data(df)
model_params <- c("beta_A" = .5, "beta_B" = .5,
```

```
                 "gamma_A" = .5, "gamma_B" = .5,
                 "tau_1" = 0, "tau_2" = 0)
n_boot <- 10
GIA_fn <- base_GIA
S_fn <- calc_S_base
fn_list <- NULL
alpha <- .05
verbose <- FALSE
out <- estimate_params(data = data,
init_params = model_params,
n_boot = n_boot,
GIA_fn = GIA_fn,
S_fn = S_fn,
fn_list = fn_list,
alpha = alpha,
verbose = verbose)
plots <- plot_curves(out, dose_A = "CyRPA",
dose_B = "RIPR")
```

`plot_isobologram` *Plot the estimated isobologram*

Description

Plot the estimated isobologram

Usage

```
plot_isobologram(
  est_list,
  dose_A = "Dose A",
  dose_B = "Dose B",
  GIA_fn = base_GIA,
  fn_list = NULL,
  title = "Isobologram Dose Combos",
  subtitle = "",
  base_size = 14
)
```

Arguments

`est_list`	output from `estimate_params`
`dose_A`	to pass to ggplot
`dose_B`	to pass to ggplot
`GIA_fn`	function to calculate GIA
`fn_list`	additional arguments to pass to GIA fn

`title`	to pass to ggplot
`subtitle`	to pass to ggplot
`base_size`	to pass to ggplot

Value

ggplot object

Examples

```
df <- loewesadditivity::cyrpa_ripr
df$dose_A <- df$CyRPA
df$dose_B <- df$RIPR
data <- fortify_gia_data(df)
model_params <- c("beta_A" = .5, "beta_B" = .5,
                 "gamma_A" = .5, "gamma_B" = .5,
                 "tau_1" = 0, "tau_2" = 0)
n_boot <- 10
GIA_fn <- base_GIA
S_fn <- calc_S_base
fn_list <- NULL
alpha <- .05
verbose <- FALSE
out <- estimate_params(data = data,
init_params = model_params,
n_boot = n_boot,
GIA_fn = GIA_fn,
S_fn = S_fn,
fn_list = fn_list,
alpha = alpha,
verbose = verbose)
plot_curves(out, dose_A = "CyRPA",
dose_B = "RIPR")
```

`plot_surface` *Plot the surface and observations*

Description

Plot the surface and observations

Usage

```
plot_surface(
  est_list,
  GIA_fn = base_GIA,
  fn_list = NULL,
  xlab = "Dose A",
```

```
  ylab = "Dose B",
  title = "Surface Plot of Doses",
  subtitle = "",
  base_size = 14
)
```

Arguments

`est_list`	output from `estimate_params`
`GIA_fn`	function to calculate GIA
`fn_list`	additional arguments to pass to GIA fn
`xlab`	to pass to ggplot
`ylab`	to pass to ggplot
`title`	to pass to ggplot
`subtitle`	to pass to ggplot
`base_size`	to pass to ggplot

Value

ggplot object

Examples

```
df <- loewesadditivity::cyrpa_ripr
df$dose_A <- df$CyRPA
df$dose_B <- df$RIPR
data <- fortify_gia_data(df)
model_params <- c("beta_A" = .5, "beta_B" = .5,
                  "gamma_A" = .5, "gamma_B" = .5,
                  "tau_1" = 0, "tau_2" = 0)
n_boot <- 10
GIA_fn <- base_GIA
S_fn <- calc_S_base
fn_list <- NULL
alpha <- .05
verbose <- FALSE
out <- estimate_params(data = data,
init_params = model_params,
n_boot = n_boot,
GIA_fn = GIA_fn,
S_fn = S_fn,
fn_list = fn_list,
alpha = alpha,
verbose = verbose)
plot_surface(out)
```

`rh5_ama1ron2` *RH5 and AMA1RON2*

Description

The data is the raw data for a combination dose of RH5 and AMA1RON2. The data was collected by PEOPLE and on DATE on this GRANT.

Usage

```
rh5_ama1ron2
```

Format

a 38 x 15 data set where the columns are of the following format

well one of iRBC (the max), uRBC (the min), RPMI (??), or comb (which is short for combination)

AMA1RON2 dose of AMA1RON2 in mg/mL

RH5 dose of RH5 in mg/mL

expxyrepz the results from experiment x, sub item y, repetition z

Examples

```
data("rh5_ama1ron2")
head(rh5_ama1ron2)
```

`rh5_rh4` *RH5 and RH4*

Description

The data is the raw data for a combination dose of RH5 and RH4. The data was originally published in Williams et al. (2018).

Usage

```
rh5_rh4
```

Format

a 48 x 3 data set where the columns are of the following format

RH4 dose of RH4 in mg/mL

RH5 dose of RH5 in mg/mL

GIA Percent Growth inhibition assay averaged over two experiments

Examples

```
data("rh5_rh4")
head(rh5_rh4)
```

`simulate_coverage` *Simulate a GIA model with an assumed error structure*

Description

Simulate a GIA model with an assumed error structure

Usage

```
simulate_coverage(
  n_sims = 10,
  n_boot = 100,
  verbose = TRUE,
  experimental_grid,
  model_par,
  alpha = 0.05,
  noise_par = c(a0 = 2, a1 = 0.01),
  GIA_fn = base_GIA,
  S_fn = calc_S_base,
  fn_list = NULL
)
```

Arguments

`n_sims`	number of coverage simulations
`n_boot`	number of bootstraps to use in each simulation
`verbose`	logical indicating whether we should use print statements. Default is TRUE
`experimental_grid`	data frame with columns 'dose_A' and 'dose_B'
`model_par`	named vector of parameters corresponding to those used in GIA_fn()
`alpha`	alpha level used to produce confidence intervals for each bootstrap
`noise_par`	named vector for the noise parameter. Must have names "a0" and "a1". See `?base_gia` for more details.
`GIA_fn`	function used to calculate GIA. Default is base_GIA().
`S_fn`	function to calculate S
`fn_list`	additional parameters to pass to GIA_fn

Value

list with the following entries

interaction_cov This is the percent of times 0 was in the (1-alpha)% confidence interval for the interaction term "tau_1" from the simulated results

params_cov This is the percent of times the true model parameter (those from model_par) lies in the (marginal) 95% confidence interval for that model parameter.

tau_pos This is the percent of times the (1-alpha)% CI of "tau_1" was completely above 0.

tau_neg This is the percent of times (1-alpha)% CI of "tau_1" is completely below zero

Examples

```
df <- loewesadditivity::cyrpa_ripr
df$dose_A <- df$CyRPA
df$dose_B <- df$RIPR
data <- fortify_gia_data(df)
model_params <- c("beta_A" = .247, "beta_B" = .224,
                  "gamma_A" = .734, "gamma_B" = .806,
                  "tau_1" = .28, "tau_2" = -.28)
experimental_grid <- make_grid(par = model_params,
                               n = 5)
n_boot <- 100
n_sims <- 10
GIA_fn <- base_GIA
S_fn <- calc_S_base
fn_list <- NULL
alpha <- .05
verbose <- TRUE
## NOT RUN
##out <- simulate_coverage(n_sims = n_sims,
  ##                      n_boot = n_boot,
   ##                     verbose = TRUE,
    ##                    experimental_grid = experimental_grid,
     ##                   model_par = model_params,
     ##                   alpha = .05,
   ##                     noise_par = c("a0" = 3, "a1" = .01),
   ##                     GIA_fn = base_GIA,
   ##                     fn_list = NULL)
##out
```

`SSE_GIA` *Calculate the Sum of Squared Error*

Description

Calculate the Sum of Squared Error

Usage

```
SSE_GIA(par, data, GIA_fn = base_GIA, fn_list = NULL)
```

Arguments

`par`	named vector of parameters
`data`	• dose_Adose A mg/mL • dose_Bdose B mg/mL • GIAGIA
`GIA_fn`	function to calculate GIA
`fn_list`	additional arguments to pass GIA_fn

Value

sum of square error between observed and estimated

Index