Package 'bayesassurance'

June 17, 2022

Type Package

Title Bayesian Assurance Computation

Version 0.1.0

Description Computes Bayesian assurance under various settings characterized by different assumptions and objectives, including precision-based conditions, credible intervals, and goal functions. All simulation-based functions included in this package rely on a two-stage Bayesian method that assigns two distinct priors to evaluate the probability of observing a positive outcome, which addresses subtle limitations that take place when using the standard single-prior approach. For more information, please refer to Pan and Banerjee (2021) <arXiv:2112.03509>.

License GPL (>= 2)

URL https://github.com/jpan928/bayesassurance_rpackage

Depends R (>= 3.5.0),

Imports ggplot2 (>= 3.3.5), plotly (>= 4.10.0), plot3D (>= 1.4), pbapply (>= 1.5.0), dplyr (>= 1.0.8), MASS (>= 7.3.55), rlang (>= 1.0.2), stats (>= 4.0.5), mathjaxr (>= 1.5.2)

Maintainer Jane Pan <jpan1@ucla.edu>

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Jane Pan [cre, aut], Sudipto Banerjee [aut]

Repository CRAN

Date/Publication 2022-06-17 10:40:17 UTC

R topics documented:

`assurance_nd_na` *Bayesian Assurance Computation*

Description

Takes in a set of parameters and returns the exact Bayesian assurance based on a closed-formed solution.

Usage

```
assurance_nd_na(n, n_a, n_d, theta_0, theta_1, sigsq, alt, alpha = 0.05)
```

Arguments

`n`	sample size (either scalar or vector)
`n_a`	sample size at analysis stage that quantifies the amount of prior information we have for parameter θ. This should be a single scalar value.
`n_d`	sample size at design stage that quantifies the amount of prior information we have for where the data is being generated from. This should be a single scalar value.
`theta_0`	parameter value that is known a priori (typically provided by the client)
`theta_1`	alternative parameter value that will be tested in comparison to theta_0. See alt for specification options.
`sigsq`	known variance σ^2.
`alt`	specifies alternative test case, where `alt = "greater"` tests if $\theta_1 > \theta_0$, `alt = "less"` tests if $\theta_1 < \theta_0$, and `alt = "two.sided"` performs a two-sided test. `alt = "greater"` by default.
`alpha`	significance level

Value

objects corresponding to the assurance

- assurance_table: table of sample sizes and corresponding assurance values.
- assurance_plot: assurance curve that is only returned if n is a vector. This curve covers a wider range of sample sizes than the inputted values specified for n, where specific assurance values are marked in red.

Examples

```
## Assign the following fixed parameters to determine the Bayesian assurance
## for the given vector of sample sizes.
n <- seq(10, 250, 5)
n_a <- 1e-8
n_d <- 1e+8
theta_0 <- 0.15
theta_1 <- 0.25
sigsq <- 0.104
assur_vals <- assurance_nd_na(n = n, n_a = n_a, n_d = n_d,
theta_0 = theta_0, theta_1 = theta_1,
sigsq = sigsq, alt = "two.sided", alpha = 0.05)
assur_vals$assurance_plot
```

---

bayes_adcock *Bayesian Assurance Computation in the Precision-Based Setting*

Description

Returns the Bayesian assurance of observing that the absolute difference between the true underlying population parameter and the sample estimate falls within a margin of error no greater than a fixed precision level, d.

Usage

```
bayes_adcock(
  n,
  d,
  mu_beta_a,
  mu_beta_d,
  n_a,
  n_d,
  sig_sq,
  alpha,
  mc_iter = 1000
)
```

Arguments

n	sample size (either vector or scalar).
d	fixed precision level
mu_beta_a	analysis stage mean
mu_beta_d	design stage mean
n_a	sample size at analysis stage. Also quantifies the amount of prior information we have for parameter μ.
n_d	sample size at design stage. Also quantifies the amount of prior information we have for where the data is being generated from.
sig_sq	known variance σ^2.
alpha	significance level
mc_iter	number of MC samples evaluated under the analysis objective

Value

approximate Bayesian assurance under precision-based conditions

Examples

```
n <- seq(20, 145, 5)
out <- bayes_adcock(n = n, d = 0.20, mu_beta_a = 0.64, mu_beta_d = 0.9,
                      n_a = 20, n_d = 10, sig_sq = 0.265,
                      alpha = 0.05, mc_iter = 1000)
head(out$assurance_table)
out$assurance_plot
```

---

bayes_goal_func *Decision Making using Rate of Correct Classification*

Description

Determines the rate of correctly classifying the linear hypothesis as true or false, where the hypothesis test is specified as

$$H0 : u'\beta = c0$$

$$vs.$$

$$H1 : u'\beta = c1$$

. See vignette for more details.

Usage

```
bayes_goal_func(n, Xn = NULL, K, pi, sigsq, u, beta_0, beta_1)
```

Arguments

n	sample size (vector or scalar).
Xn	design matrix that characterizing the data. This is specifically given by the normal linear regression model $$yn = Xn\beta + \epsilon,$$ $$\epsilon\ N(0, \sigma^2 I_n),$$ where I_n is an n by n identity matrix. When set to NULL, an appropriate Xn is automatically generated bayesassurance::gen_Xn(). Note that setting Xn = NULL also enables user to pass in a vector of sample sizes to undergo evaluation as the function will automatically adjust Xn accordingly based on the sample size.
K	The amount of utility associated with $H0$ being correctly accepted.The null hypothesis is not rejected if the posterior probability of $H0$ is at least $1/(1+K)$.
pi	constant corresponding to the prior on parameter β such that $P(u'\beta_0) = 1 - P(u'\beta_1) = \pi$.
sigsq	variance constant of the linear regression model
u	fixed scalar or vector of the same dimension as β_0 and β_1
beta_0	fixed scalar or vector that null hypothesis is set to
beta_1	fixed scalar or vector that alternative hypothesis is set to

Value

a list of objects corresponding to the rate of classifications

- rc_table: table of sample size and corresponding correct classification rates
- rc_plot: plot of correct classification rates for varying sample sizes

Examples

```
## Example
n <- seq(100, 1200, 10)
out <- bayesassurance::bayes_goal_func(n, Xn = NULL, K = 1, pi = 0.5,
sigsq = 1, u = 1, beta_0 = 0.5, beta_1 = 0.6)
out$rc_plot
```

bayes_sim *Bayesian Simulation in Conjugate Linear Model Framework*

Description

Approximates the Bayesian assurance of attaining either $u'\beta > C$, $u'\beta < C$, or $u'\beta \neq C$, for equal-sized samples through Monte Carlo sampling. The function also carries the capability to process longitudinal data. See Argument descriptions for more detail.

Usage

```
bayes_sim(
  n,
  p = NULL,
  u,
  C,
  Xn = NULL,
  Vn = NULL,
  Vbeta_d,
  Vbeta_a_inv,
  sigsq,
  mu_beta_d,
  mu_beta_a,
  alt = "two.sided",
  alpha,
  mc_iter,
  longitudinal = FALSE,
  ids = NULL,
  from = NULL,
  to = NULL,
  poly_degree = NULL
)
```

Arguments

n	sample size (either scalar or vector). When `longitudinal = TRUE`, `n` denotes the number of observations per subject.
p	column dimension of design matrix `Xn`. If `Xn = NULL`, `p` must be specified to denote the column dimension of the default design matrix generated by the function.
u	a scalar or vector included in the expression to be evaluated, e.g. $$u'\beta > C,$$ where β is an unknown parameter that is to be estimated.
C	constant to be compared

`Xn` design matrix that characterizes where the data is to be generated from. This is specifically given by the normal linear regression model

$$yn = Xn\beta + \epsilon,$$

$$\epsilon\ N(0, \sigma^2 Vn).$$

When set to `NULL`, `Xn` is generated in-function using either `bayesassurance::gen_Xn()` or `bayesassurance::gen_Xn_longitudinal()`. Note that setting `Xn = NULL` also enables user to pass in a vector of sample sizes to undergo evaluation as the function will automatically adjust `Xn` accordingly based on the sample size.

`Vn` a correlation matrix for the marginal distribution of the sample data yn. Takes on an identity matrix when set to `NULL`.

`Vbeta_d` correlation matrix that helps describe the prior information on β in the design stage

`Vbeta_a_inv` inverse-correlation matrix that helps describe the prior information on β in the analysis stage

`sigsq` a known and fixed constant preceding all correlation matrices `Vn`, `Vbeta_d`, and `Vbeta_a_inv`.

`mu_beta_d` design stage mean

`mu_beta_a` analysis stage mean

`alt` specifies alternative test case, where alt = "greater" tests if $u'\beta > C$, alt = "less" tests if $u'\beta < C$, and alt = "two.sided" performs a two-sided test. By default, alt = "greater".

`alpha` significance level

`mc_iter` number of MC samples evaluated under the analysis objective

`longitudinal` when set to `TRUE`, constructs design matrix using inputs that correspond to a balanced longitudinal study design.

`ids` vector of unique subject ids, usually of length 2 for study design purposes.

`from` start time of repeated measures for each subject

`to` end time of repeated measures for each subject

`poly_degree` only needed if `longitudinal = TRUE`, specifies highest degree taken in the longitudinal model.

Value

a list of objects corresponding to the assurance approximations

- assurance_table: table of sample size and corresponding assurance values
- assur_plot: plot of assurance values
- mc_samples: number of Monte Carlo samples that were generated and evaluated

See Also

`pwr_freq` for frequentist power function, `assurance_nd_na` for a closed form assurance function, and `bayes_sim_unknownvar` for a Bayesian assurance function assumes unvariance.

Examples

```
## Example 1
## A single Bayesian assurance value obtained given a scalar sample size
## n and p=1. Note that setting p=1 implies that
## beta is a scalar parameter.

bayesassurance::bayes_sim(n=100, p = 1, u = 1, C = 0.15, Xn = NULL,
Vbeta_d = 1e-8, Vbeta_a_inv = 0, Vn = NULL, sigsq = 0.265, mu_beta_d = 0.3,
mu_beta_a = 0, alt = "two.sided", alpha = 0.05, mc_iter = 5000)


## Example 2
## Illustrates a scenario in which weak analysis priors and strong
## design priors are assigned to enable overlap between the frequentist
## power and Bayesian assurance.


library(ggplot2)
n <- seq(100, 250, 5)

 ## Frequentist Power
 power <- bayesassurance::pwr_freq(n, sigsq = 0.265, theta_0 = 0.15,
 theta_1 = 0.25, alt = "greater", alpha = 0.05)

 ## Bayesian simulation values with specified values from the n vector
 assurance <- bayesassurance::bayes_sim(n, p = 1, u = 1, C = 0.15, Xn = NULL,
 Vbeta_d = 1e-8, Vbeta_a_inv = 0, Vn = NULL, sigsq = 0.265, mu_beta_d = 0.25,
 mu_beta_a = 0, alt = "greater", alpha = 0.05, mc_iter = 1000)

## Visual representation of plots overlayed on top of one another
df1 <- as.data.frame(cbind(n, power = power$pwr_table$Power))
df2 <- as.data.frame(cbind(n, assurance =
assurance$assurance_table$Assurance))

plot_curves <- ggplot2::ggplot(df1, alpha = 0.5, ggplot2::aes(x = n, y = power,
color="Frequentist")) + ggplot2::geom_line(lwd=1.2)
plot_curves <- plot_curves + ggplot2::geom_point(data = df2, alpha = 0.5,
aes(x = n, y = assurance, color="Bayesian"),lwd=1.2) +
ggplot2::ggtitle("Bayesian Simulation vs. Frequentist Power Computation")
plot_curves


## Example 3
## Longitudinal example where n now denotes the number of repeated measures
## per subject and design matrix is determined accordingly.

## subject ids
n <- seq(10, 100, 5)
ids <- c(1,2)
sigsq <- 100
Vbeta_a_inv <- matrix(rep(0, 16), nrow = 4, ncol = 4)
```

```
Vbeta_d <- (1 / sigsq) *
matrix(c(4, 0, 3, 0, 0, 6, 0, 0, 3, 0, 4, 0, 0, 0, 0, 6),
nrow = 4, ncol = 4)

assur_out <- bayes_sim(n = n, p = NULL, u = c(1, -1, 1, -1), C = 0,
                       Xn = NULL, Vbeta_d = Vbeta_d,
                       Vbeta_a_inv = Vbeta_a_inv,
                       Vn = NULL, sigsq = 100,
                       mu_beta_d = as.matrix(c(5, 6.5, 62, 84)),
                       mu_beta_a = as.matrix(rep(0, 4)), mc_iter = 1000,
                       alt = "two.sided", alpha = 0.05,
                       longitudinal = TRUE, ids = ids,
                       from = 10, to = 120)
assur_out$assurance_plot
```

`bayes_sim_betabin` *Bayesian Assurance Computation in the Beta-Binomial Setting*

Description

Returns the Bayesian assurance corresponding to a hypothesis test for difference in two independent proportions.

Usage

```
bayes_sim_betabin(
  n1,
  n2,
  p1,
  p2,
  alpha_1,
  alpha_2,
  beta_1,
  beta_2,
  sig_level,
  alt,
  mc_iter
)
```

Arguments

n1	sample size of first group
n2	sample size of second group
p1	proportion of successes in first group. Takes on a NULL (default) assignment if unknown.
p2	proportion of successes in second group. Takes on a NULL (default) assignment if unknown.

`alpha_1, beta_1`
: shape parameters for the distribution of p1 if p1 is unknown: $p1\ Beta(\alpha_1, \beta_1)$

`alpha_2, beta_2`
: shape parameters for the distribution of p2 if p2 is unknown: $p2\ Beta(\alpha_2, \beta_2)$

`sig_level`
: significance level

`alt`
: a character string specifying the alternative hypothesis, must select one of following choices: `"two.sided"` (default), `"greater"` or `"less"`.

`mc_iter`
: number of MC samples evaluated under the analysis objective

Value

approximate Bayesian assurance of independent two-sample proportion test

Examples

```
#########################################################
# alpha1 = 0.5, beta1 = 0.5, alpha2 = 0.5, beta2 = 0.5 ##
#########################################################
n <- seq(200, 1000, 10)
assur_vals <- bayesassurance::bayes_sim_betabin(n1 = n, n2 = n,
p1 = 0.25, p2 = 0.2, alpha_1 = 0.5, beta_1 = 0.5, alpha_2 = 0.5,
beta_2 = 0.5, sig_level = 0.05, alt = "greater", mc_iter = 1000)

assur_vals$assurance_table
assur_vals$assurance_plot
```

---

bayes_sim_unbalanced *Unbalanced Bayesian Simulation in Conjugate Linear Model Framework*

Description

Approximates the Bayesian assurance of attaining $u'\beta > C$ for unbalanced study designs through Monte Carlo sampling. See Argument descriptions for more detail.

Usage

```
bayes_sim_unbalanced(
  n1,
  n2,
  repeats = 1,
  u,
  C,
  Xn = NULL,
  Vn = NULL,
  Vbeta_d,
```

```
  Vbeta_a_inv,
  sigsq,
  mu_beta_d,
  mu_beta_a,
  alt,
  alpha,
  mc_iter,
  surface_plot = TRUE
)
```

Arguments

n1	first sample size (vector or scalar).
n2	second sample size (vector or scalar).
repeats	an positive integer specifying number of times to repeat c(n1, n2). Applicable for studies that consider multiple measures within each group. Default setting is repeats = 1 if not applicable.
u	a scalar or vector to evaluate $$u'\beta > C,$$ where β is an unknown parameter that is to be estimated. Default setting is u = 1.
C	constant value to be compared to when evaluating $u'\beta > C$
Xn	design matrix that characterizes where the data is to be generated from. This is specifically designed under the normal linear regression model $$yn = Xn\beta + \epsilon,$$ $$\epsilon\ N(0, \sigma^2 Vn).$$ When set to NULL, Xn is generated in-function using bayesassurance::gen_Xn(). Note that setting Xn = NULL also enables user to pass in a vector of sample sizes to undergo evaluation.
Vn	a correlation matrix for the marginal distribution of the sample data yn. Takes on an identity matrix when set to NULL.
Vbeta_d	correlation matrix that helps describe the prior information on β in the design stage
Vbeta_a_inv	inverse-correlation matrix that helps describe the prior information on β in the analysis stage
sigsq	a known and fixed constant preceding all correlation matrices Vn, Vbeta_d and Vbeta_a_inv.
mu_beta_d	design stage mean
mu_beta_a	analysis stage mean
alt	specifies alternative test case, where alt = "greater" tests if $u'\beta > C$, alt = "less" tests if $u'\beta < C$, and alt = "two.sided" performs a two-sided test. By default, alt = "greater".

alpha	significance level
mc_iter	number of MC samples evaluated under the analysis objective
surface_plot	when set to TRUE and $n1$ and $n2$ are vectors, a contour map showcasing various assurance values corresponding to different combinations of $n1$ and $n2$ is produced.

Value

a list of objects corresponding to the assurance approximations

- assurance_table: table of sample size and corresponding assurance values
- contourplot: contour map of assurance values
- mc_samples: number of Monte Carlo samples that were generated and evaluated

Examples

```
## Example 1
## Sample size vectors are passed in for n1 and n2 to evaluate
## assurance.
n1 <- seq(20, 75, 5)
n2 <- seq(50, 160, 10)

assur_out <- bayes_sim_unbalanced(n1 = n1, n2 = n2, repeats = 1, u = c(1, -1),
C = 0, Xn = NULL, Vbeta_d = matrix(c(50, 0, 0, 10),nrow = 2, ncol = 2),
Vbeta_a_inv = matrix(rep(0, 4), nrow = 2, ncol = 2),
Vn = NULL, sigsq = 100,  mu_beta_d = c(1.17, 1.25),
mu_beta_a = c(0, 0), alt = "two.sided", alpha = 0.05, mc_iter = 1000,
surface_plot = FALSE)

assur_out$assurance_table


## Example 2
## We can produce a contour plot that evaluates unique combinations of n1
## and n2 simply by setting `surfaceplot = TRUE`.

n1 <- seq(20, 75, 5)
n2 <- seq(50, 160, 10)
assur_out <- bayes_sim_unbalanced(n1 = n1, n2 = n2, repeats = 1,
u = c(1, -1), C = 0, Xn = NULL, Vbeta_d = matrix(c(50, 0, 0, 10),
nrow = 2, ncol = 2), Vbeta_a_inv = matrix(rep(0, 4), nrow = 2, ncol = 2),
Vn = NULL, sigsq = 100,  mu_beta_d = c(1.17, 1.25),
mu_beta_a = c(0, 0), alt = "two.sided", alpha = 0.05, mc_iter = 1000,
surface_plot = TRUE)

assur_out$assurance_table
assur_out$contourplot
```

bayes_sim_unknownvar *Bayesian Simulation with Composite Sampling*

Description

Approximates the Bayesian assurance of a one-sided hypothesis test through Monte Carlo sampling with the added assumption that the variance is unknown.

Usage

```
bayes_sim_unknownvar(
  n,
  p = NULL,
  u,
  C,
  R,
  Xn = NULL,
  Vn,
  Vbeta_d,
  Vbeta_a_inv,
  mu_beta_d,
  mu_beta_a,
  a_sig_a,
  b_sig_a,
  a_sig_d,
  b_sig_d,
  alt = "greater",
  alpha,
  mc_iter
)
```

Arguments

n	sample size (either vector or scalar)
p	column dimension of design matrix Xn. If Xn = NULL, p must be specified to denote the column dimension of the default design matrix generated by the function.
u	a scalar or vector to evaluate $$u'\beta > C,$$ where β is an unknown parameter that is to be estimated.
C	constant value to be compared to when evaluating $u'\beta > C$
R	number of iterations we want to pass through to check for satisfaction of the analysis stage objective. The proportion of those iterations meeting the analysis objective corresponds to the approximated Bayesian assurance.

`Xn` design matrix that characterizes where the data is to be generated from. This is specifically given by the normal linear regression model

$$yn = Xn\beta + \epsilon,$$

$$\epsilon\ N(0, \sigma^2 Vn),$$

where σ^2 is unknown in this setting. Note that `Xn` must have column dimension `p`.

`Vn` an `n` by `n` correlation matrix for the marginal distribution of the sample data `yn`. Takes on an identity matrix when set to `NULL`.

`Vbeta_d` correlation matrix that helps describe the prior information on β in the design stage

`Vbeta_a_inv` inverse-correlation matrix that helps describe the prior information on β in the analysis stage

`mu_beta_d` design stage mean

`mu_beta_a` analysis stage mean

`a_sig_a, b_sig_a` shape and scale parameters of the inverse gamma distribution where variance σ^2 is sampled from in the analysis stage

`a_sig_d, b_sig_d` shape and scale parameters of the inverse gamma distribution where variance σ^2 is sampled from in the design stage

`alt` specifies alternative test case, where `alt = "greater"` tests if $u'\beta > C$, `alt = "less"` tests if $u'\beta < C$, and `alt = "two.sided"` performs a two-sided test. By default, `alt = "greater"`.

`alpha` significance level

`mc_iter` number of MC samples evaluated under the analysis objective

Value

a list of objects corresponding to the assurance approximations

- assurance_table: table of sample size and corresponding assurance values
- assur_plot: plot of assurance values
- mc_samples: number of Monte Carlo samples that were generated and evaluated

See Also

`bayes_sim` for the Bayesian assurance function for known variance.

Examples

```
## O'Hagan and Stevens (2001) include a series of examples with
## pre-specified parameters that we will be using to replicate their
## results through our Bayesian assurance simulation.
```

```
## The inputs are as follows:

n <- 285
p <- 4 ## includes two parameter measures (cost and efficacy) for each of
       ## the two treatments, for a total of p = 4 parameters.
K <- 20000
C <- 0
u <- as.matrix(c(-K, 1, K, -1))

## Set up correlation matrices
Vbeta_a_inv <- matrix(rep(0, p^2), nrow = p, ncol = p)

sigsq <- 4.04^2
tau1 <- tau2 <- 8700
sig <- sqrt(sigsq)
Vn <- matrix(0, nrow = n*p, ncol = n*p)
Vn[1:n, 1:n] <- diag(n)
Vn[(2*n - (n-1)):(2*n), (2*n - (n-1)):(2*n)] <- (tau1 / sig)^2 * diag(n)
Vn[(3*n - (n-1)):(3*n), (3*n - (n-1)):(3*n)] <- diag(n)
Vn[(4*n - (n-1)):(4*n), (4*n - (n-1)):(4*n)] <- (tau2 / sig)^2 * diag(n)

Vbeta_d <- (1 / sigsq) *
matrix(c(4, 0, 3, 0, 0, 10^7, 0, 0, 3, 0, 4, 0, 0, 0, 0, 10^7),
nrow = 4, ncol = 4)
mu_beta_d <- as.matrix(c(5, 6000, 6.5, 7200))
mu_beta_a <- as.matrix(rep(0, p))
alpha <- 0.05
epsilon <- 10e-7
a_sig_d <- (sigsq / epsilon) + 2
b_sig_d <- sigsq * (a_sig_d - 1)
a_sig_a <- -p / 2
b_sig_a <- 0

bayesassurance::bayes_sim_unknownvar(n = n, p = 4,
u = as.matrix(c(-K, 1, K, -1)), C = 0, R = 40,
Xn = NULL, Vn = Vn, Vbeta_d = Vbeta_d,
Vbeta_a_inv = Vbeta_a_inv, mu_beta_d = mu_beta_d,
mu_beta_a = mu_beta_a, a_sig_a = a_sig_a, b_sig_a = b_sig_a,
a_sig_d = a_sig_d, b_sig_d = b_sig_d, alt = "two.sided", alpha = 0.05,
mc_iter = 500)
```

---

gen_Xn *Design Matrix Generator*

Description

Constructs design matrix using given sample size(s). Used for assurance analysis in the Bayesian setting.

Usage

```
gen_Xn(n)
```

Arguments

n	vector of sample sizes. Length of n corresponds to the number of groups being assessed in the study design as well as the column dimension of the design matrix.

Value

Xn: a design matrix that can be used to assess the Bayesian assurance through Monte Carlo sampling using functions presented in this package.

See Also

gen_Xn_longitudinal

Examples

```
## In the following example, notice that passing in a vector
## of length 4 returns a design matrix of column dimension 4, where
## each column is comprised of ones vectors with lengths that correspond
## to the inputted sample sizes.

n <- c(1,3,5,8)
gen_Xn(n = n)
```

---

gen_Xn_longitudinal *Design Matrix Generator in Longitudinal Setting*

Description

Constructs design matrix using inputs that correspond to a balanced longitudinal study design. Used for power and sample size analysis in the Bayesian setting.

Usage

```
gen_Xn_longitudinal(ids, from, to, num_repeated_measures, poly_degree = 1)
```

Arguments

ids	vector of unique subject ids, usually of length 2 for study design purposes.
from	start time of repeated measures for each subject
to	end time of repeated measures for each subject
num_repeated_measures	desired length of the repeated measures sequence. Should be a non-negative number, will be rounded up if fractional.
poly_degree	degree of polynomial in longitudinal model, set to 1 by default.

Value

Xn: a design matrix that can be used to assess the Bayesian assurance through Monte Carlo sampling using functions presented in this package.

See Also

gen_Xn

Examples

```
## Example 1
## We pass in a vector of subject IDs and specify the start and end
## timepoints along with the desired length of the sequence.
## The resulting design matrix contains vectors of
## ones with lengths that correspond to the number of repeated
## measures for each unique subject.

ids <- c(1,2,3,4)
gen_Xn_longitudinal(ids, from = 1, to = 10, num_repeated_measures = 4)

## Example 2
## If we wish to fit a longitudinal model of a higher degree (e.g.
## parabolic, cubic), we need to adjust the `poly_degree` variable

# parabolic
ids <- c(1,2,3,4)
gen_Xn_longitudinal(ids, from = 1, to = 10, num_repeated_measures = 4,
poly_degree = 2)

# cubic
ids <- c(1,2,3,4)
gen_Xn_longitudinal(ids, from = 1, to = 10, num_repeated_measures = 4,
poly_degree = 3)
```

---

pwr_curve *Plotting Power and Assurance Curves Simultaneously*

Description

Constructs a single plot based on the combined set of inputs used to compute the frequentist power and Bayesian assurance. The plot includes three curves that include the power curve, the assurance curve obtained under a closed-form solution, and optionally, the assurance curve obtained under a simulation setting.

Usage

```
pwr_curve(
  n,
  n_a,
```

```
  n_d,
  theta_0,
  theta_1,
  sigsq,
  alt = "greater",
  alpha,
  bayes_sim = FALSE,
  mc_iter = NULL
)
```

Arguments

n	sample size (either a scalar or vector)
n_a	sample size at analysis stage (setting n_a close to 0 corresponds to a weak analysis prior)
n_d	sample size at design stage
theta_0	parameter value that is known a priori (typically provided by the client)
theta_1	alternative parameter value that will be tested in comparison to theta_0. See alt for specification options.
sigsq	known variance σ^2
alt	specifies alternative test case, where alt = "greater" tests if theta_1 > theta_0, "less" tests if theta_1 < theta_0, and "two.sided" performs a two-sided test. alt = "greater" by default.
alpha	significance level
bayes_sim	when set to "TRUE", this indicates that the user wants to include simulated assurance results in the outputted plot. Default setting is "FALSE".
mc_iter	number of MC samples provided that bayes_sim = TRUE

Value

plot of overlayed power and assurance curves produced using ggplot2

a list of objects corresponding to the power/assurance curves

- power: table of sample sizes and corresponding power values obtained from bayesassurance::pwr_freq().
- assurance_table: table of sample sizes and corresponding assurance values obtained from bayesassurance::assurance_nd_na().
- bayes_sim_table: table of sample sizes and corresponding assurance values obtained from MC sampling using bayesassurance::bayes_sim(). Returned only if bayes_sim = TRUE.
- mc_samples: number of Monte Carlo samples that were generated and evaluated if bayes_sim = TRUE.
- plot: plot of overlayed power/assurance curves.

See Also

ggplot2, pwr_freq for frequentist power function and bayes_sim for the Bayesian assurance function

Examples

```
## Case 1: Weak Analysis Prior (n_a set to be small) + Strong Design Prior
## (n_d set to be large) that results in the Bayesian assurance and
## frequentist curve perfectly overlapping one another.

n <- seq(10, 200, 10)
n_a <- 1e-8
n_d <- 1e+8
theta_0 <- 0.15
theta_1 <- 0.25
sigsq <- 0.104
alpha <- 0.05

## outputs all three plots
out <- bayesassurance::pwr_curve(n = n, n_a = n_a, n_d = n_d,
theta_0 = theta_0, theta_1 = theta_1, sigsq = sigsq, alt = "greater",
alpha = alpha, bayes_sim = TRUE, mc_iter = 5000)

## only outputs the closed-form solution power and assurance curves
pwr_curve(n = n, n_a = n_a, n_d = n_d, theta_0 = theta_0, theta_1 = theta_1,
sigsq = sigsq, alt = "greater", alpha = alpha, bayes_sim = FALSE)



## Case 2: Weak Analysis Prior (n_a set to be small) + Weak Design Prior
## (n_d set to be small) that results in a assurance curve,
## which illustrates the noninformative prior setting.
n <- seq(10, 200, 10)
n_a <- 1e-8
n_d <- 1e-8
theta_0 <- 0.15
theta_1 <- 0.25
sigsq <- 0.104
alpha <- 0.05

bayesassurance::pwr_curve(n = n, n_a = n_a, n_d = n_d, theta_0 = theta_0,
theta_1 = theta_1, sigsq = sigsq, alt = "greater", alpha = alpha,
bayes_sim = TRUE, mc_iter = 1000)
```

---

pwr_freq *Frequentist Power Computation*

Description

Constructs a simple hypothesis testing framework based on the parameters specified and returns the corresponding frequentist power.

Usage

```
pwr_freq(n, theta_0, theta_1, sigsq, alt, alpha)
```

Arguments

n	sample size (either scalar or vector)
theta_0	initial value the parameter is set equal to in the null hypothesis, where $H0 : \theta = \theta 0$.
theta_1	alternative value to be compared to theta_0. See alt for specification options.
sigsq	known variance σ^2
alt	specifies comparison between θ_1 and θ_0, where `alt = "greater"` tests if $\theta_1 > \theta_0$, `alt = "less"` tests if $\theta_1 < \theta_0$, and `alt = "two.sided"` performs a two-sided test. `alt = "greater"` by default.
alpha	significance level

Value

objects corresponding to the power

- pwr_table: table of sample sizes and corresponding power values.
- pwr_plot: power curve that is only returned if n is a vector. This power curve covers a wider range of sample sizes than the inputted values specified for n, where specific power values are marked in red.
- power_val: single power value that is returned if n is a scalar.

Examples

```
n <- seq(10, 140, 5)
theta_0 <- 0.15
theta_1 <- 0.35
sigsq <- 0.3

pwr_vals <- pwr_freq(n = n, theta_0 = theta_0, theta_1 = theta_1,
sigsq = sigsq, alt = "greater", alpha = 0.05)
pwr_vals$pwr_plot
```

Index