# Package 'OptimalGoldstandardDesigns'

September 11, 2023

**Type** Package

**Title** Design Parameter Optimization for Gold-Standard Non-Inferiority Trials

**Version** 1.0.1

**Description** Methods to calculate optimal design parameters for one- and two-stage three-arm group-sequential gold-standard non-inferiority trial designs with or without binding or nonbinding futility boundaries, as described in Meis et al. (2023) <doi:10.1002/sim.9630>.

**License** GPL (>= 3)

**Encoding** UTF-8

**Depends** R (>= 3.5)

**Imports** nloptr, mvtnorm, cli, dplyr, tibble, Rdpack

**Suggests** testthat (>= 3.0.0), here, knitr, rmarkdown, covr, fpCompare, mnormt, future.apply

**Config/testthat/edition** 3

**RoxygenNote** 7.2.3

**RdMacros** Rdpack

**Collate** 'OptimalGoldstandardDesigns-package.R' 'pmv_upper_smaller_slower_fix.R' 'conditional_probability_functions.R' 'design_display_functions.R' 'global_constants.R' 'design_helper_functions.R' 'objective_functions.R' 'optimization_methods.R' 'table_format_functions.R'

**VignetteBuilder** knitr

**URL** `https://github.com/jan-imbi/OptimalGoldstandardDesigns`

**NeedsCompilation** no

**Author** Jan Meis [aut, cre] (<`https://orcid.org/0000-0001-5407-7220`>)

**Maintainer** Jan Meis `<meis@imbi.uni-heidelberg.de>`

**Repository** CRAN

**Date/Publication** 2023-09-11 10:40:05 UTC

## R topics documented:

---

`calc_ASN` *Helper function to calculate the average sample size*

---

### Description

Helper function to calculate the average sample size

### Usage

```
calc_ASN(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

---

## `calc_ASNP` *Calculate the average placebo group sample size*

---

### Description

Calculate the average placebo group sample size

### Usage

```
calc_ASNP(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

---

## `calc_c` *Helper function to calculate allocation ratios from stagewise sample sizes*

---

### Description

Helper function to calculate allocation ratios from stagewise sample sizes

### Usage

```
calc_c(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## calc_conditional_local_rejection_probs

*Calculate the (local) conditional type I errors of both hypothesis given both interim test statistics.*

### Description

Calculate the (local) conditional type I errors of both hypothesis given both interim test statistics.

### Usage

```
calc_conditional_local_rejection_probs(Z_TP1, Z_TC1, D, mu_vec = D$mu_vec$H0)
```

### Arguments

| | |
|---|---|
| Z_TP1 | First stage Z-test statistic of the superiority test of treatment over placebo. |
| Z_TC1 | First stage Z-test statistic of the non-inferiority test of treatment to the active control. |
| D | list containing all parameters of a the trial design. |
| mu_vec | vector of means of the test statistic c(Z_TP1, Z_TP2, Z_TC1, Z_TC1). |

### Value

named numeric vector with both conditional type I errors.

## calc_conditional_power

*Calculate the conditional power to reject both hypothesis given both interim test statistics.*

### Description

Calculate the conditional power to reject both hypothesis given both interim test statistics.

### Usage

```
calc_conditional_power(Z_TP1, Z_TC1, D)
```

### Arguments

| | |
|---|---|
| Z_TP1 | First stage Z-test statistic of the superiority test of treatment over placebo. |
| Z_TC1 | First stage Z-test statistic of the non-inferiority test of treatment to the active control. |
| D | list containing all parameters of a the trial design. |

### Value

numeric value of the conditional power.

---

## `calc_cumc` *Helper function to calculate "cumulative allocation ratio" from stagewise allocation ratios*

---

### Description

Helper function to calculate "cumulative allocation ratio" from stagewise allocation ratios

### Usage

```
calc_cumc(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

---

## `calc_cumn` *Helper function to calculate cumulative sample sizes from stagewise sample sizes*

---

### Description

Helper function to calculate cumulative sample sizes from stagewise sample sizes

### Usage

```
calc_cumn(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

`calc_final_state_probs` *Helper function to calculate the final state probabilities*

## Description

Helper function to calculate the final state probabilities

## Usage

```
calc_final_state_probs(hypothesis = "H0", D)
```

## Arguments

| | |
|---|---|
| `hypothesis` | string describing the hypothesis, either "H0" or "H1". |
| `D` | list containing all parameters of a the trial design. |

`calc_gamma` *Helper function to calculate gamma factors from group variances and cumulative allocation ratios*

## Description

Helper function to calculate gamma factors from group variances and cumulative allocation ratios

## Usage

```
calc_gamma(D)
```

## Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

`calc_local_alphas` *Helper function to calculate the local type I error rates of a Design*

## Description

Helper function to calculate the local type I error rates of a Design

## Usage

```
calc_local_alphas(D)
```

## Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `calc_local_rejection_boundaries` *Helper function to calculate the local rejection boundaries of group sequential testing procedure associated with the hypothesis belong to the groups argument*

### Description

Helper function to calculate the local rejection boundaries of group sequential testing procedure associated with the hypothesis belong to the groups argument

### Usage

```
calc_local_rejection_boundaries(groups = "TP", D)
```

### Arguments

| | |
|---|---|
| `groups` | string describing the pair of groups. Either "TP" or "TC". |
| `D` | list containing all parameters of a the trial design. |

## `calc_mu_vec` *Helper function to calculate expected value of normal test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TP2) under the null and alternative hypothesis given nT1, gamma and mu.*

### Description

Helper function to calculate expected value of normal test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TP2) under the null and alternative hypothesis given nT1, gamma and mu.

### Usage

```
calc_mu_vec(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `calc_mu_wo_nT1` *Helper function to calculate expected value of normal test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TP2) under the null and alternative hypothesis given nT1=1, gamma and mu.*

### Description

This quantity is helpful when solving for the smallest n which fulfills a certain power constraint.

### Usage

```
calc_mu_wo_nT1(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `calc_nT1_wrt_bTC2e` *Helper function to calculate the required sample size (of the stage 1 treatment group) to achieve the target power given the bTC2e*

### Description

Helper function to calculate the required sample size (of the stage 1 treatment group) to achieve the target power given the bTC2e

### Usage

```
calc_nT1_wrt_bTC2e(bTC2e, D)
```

### Arguments

| | |
|---|---|
| `bTC2e` | second stage critical value for the TC testing problem. |
| `D` | list containing all parameters of a the trial design. |

## `calc_n_from_c` *Helper function to calculate other n's given n_1,T and allocation ratios*

### Description

Helper function to calculate other n's given n_1,T and allocation ratios

### Usage

```
calc_n_from_c(nT1, D)
```

### Arguments

| | |
|---|---|
| `nT1` | first stage sample size of the treatment group. |
| `D` | list containing all parameters of a the trial design. |

## `calc_prob_reject_both` *Helper function to calculate the probability to reject both hypotheses given the mean of the normal test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TC2).*

### Description

Helper function to calculate the probability to reject both hypotheses given the mean of the normal test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TC2).

### Usage

```
calc_prob_reject_both(mu_vec, D)
```

### Arguments

| | |
|---|---|
| `mu_vec` | vector of means of the test statistic c(Z_TP1, Z_TP2, Z_TC1, Z_TC1). |
| `D` | list containing all parameters of a the trial design. |

## calc_prob_reject_both_singlestage

*Helper function to calculate the probability to reject both hypotheses given the mean of the normal test statistic vector c(Z_TP1, Z_TC1).*

### Description

Helper function to calculate the probability to reject both hypotheses given the mean of the normal test statistic vector c(Z_TP1, Z_TC1).

### Usage

```
calc_prob_reject_both_singlestage(mu_vec, D)
```

### Arguments

| | |
|---|---|
| `mu_vec` | vector of means of the test statistic c(Z_TP1, Z_TP2, Z_TC1, Z_TC1). |
| `D` | list containing all parameters of a the trial design. |

## calc_Sigma

*Helper function to calculate the covariance matrix from the group variances, cumulative allocation ratios and gamma factors*

### Description

Helper function to calculate the covariance matrix from the group variances, cumulative allocation ratios and gamma factors

### Usage

```
calc_Sigma(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `calc_worst_type_I_error` *Helper function to calculate the maximal probability of rejecting the non-inferiority hypothesis in the testing procedure featuring nonsequential futility, given a point hypothesis for the superiority hypothesis.*

### Description

This is required in designs with nonsequential futility testing, as choosing locally valid designs is insufficient to gurantee type I error control.

### Usage

```
calc_worst_type_I_error(bTC2e, D)
```

### Arguments

| | |
|---|---|
| `bTC2e` | second stage critical value for the TC testing problem. |
| `D` | list containing all parameters of a the trial design. |

## `conditional_mean` *Calculate the conditional mean of a multivariate normal distribution*

### Description

See e.g. Chapter 8.1.2 in The Matrix Cookbook.

### Usage

```
conditional_mean(x_a, mu_a, mu_b, Sigma)
```

### Arguments

| | |
|---|---|
| `x_a` | (Observed) first part of normally distributed vector. |
| `mu_a` | Mean of second part of normally distributed vector. |
| `mu_b` | Mean of first part of normally distributed vector. |
| `Sigma` | covariance matrix of test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TC2). |

### Value

numeric vector with the conditional mean

### References

Petersen, K. B., & Pedersen, M. S. (2008). The matrix cookbook. Technical University of Denmark, 7(15), 510.

## `conditional_Sigma` *Calculate the conditional mean of a multivariate normal distribution*

### Description

See e.g. Chapter 8.1.2 in The Matrix Cookbook.

### Usage

```
conditional_Sigma(x_a, mu_a, mu_b, Sigma)
```

### Arguments

| | |
|---|---|
| `x_a` | (Observed) first part of normally distributed vector. |
| `mu_a` | Mean of second part of normally distributed vector. |
| `mu_b` | Mean of first part of normally distributed vector. |
| `Sigma` | covariance matrix of test statistic vector c(Z_TP1, Z_TP2, Z_TC1, Z_TC2). |

### Value

numeric vector with the conditional covariance matrix

### References

Petersen, K. B., & Pedersen, M. S. (2008). The matrix cookbook. Technical University of Denmark, 7(15), 510.

## `objective_onestage` *Objective function for single-stage gold-standard designs*

### Description

Objective function for single-stage gold-standard designs

### Usage

```
objective_onestage(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `objective_twostage` *Objective function for two-stage gold-standard designs*

### Description

Objective function for two-stage gold-standard designs

### Usage

```
objective_twostage(D)
```

### Arguments

| | |
|---|---|
| `D` | list containing all parameters of a the trial design. |

## `OptimalGoldstandardDesigns` *OptimalGoldstandardDesigns*

### Description

Methods to calculate optimal design parameters for one- and two-stage three-arm group-sequential gold-standard non-inferiority trial designs with or without binding or nonbinding futility boundaries, as described in (Meis et al. 2023).

### Author(s)

**Maintainer**: Jan Meis `<meis@imbi.uni-heidelberg.de>` (ORCID)

### References

Meis J, Pilz M, Herrmann C, Bokelmann B, Rauch G, Kieser M (2023). "Optimization of the two-stage group sequential three-arm gold-standard design for non-inferiority trials." *Statistics in Medicine*, **42**(4), 536-558. doi:10.1002/sim.9630.

### See Also

Useful links:

- `https://github.com/jan-imbi/OptimalGoldstandardDesigns`

## optimize_design_onestage

*Calculate optimal design parameters for a single-stage gold-standard design*

### Description

Calculate optimal design parameters for a single-stage gold-standard design

### Usage

```
optimize_design_onestage(
  cP1 = NULL,
  cC1 = NULL,
  alpha = 0.025,
  beta = 0.2,
  alternative_TP = 0.4,
  alternative_TC = 0,
  Delta = 0.2,
  varT = 1,
  varP = 1,
  varC = 1,
  round_n = TRUE,
  kappa = 0,
  objective = quote(sum(unlist(n)) + kappa * n[[1]][["P"]]),
  inner_tol_objective = 1e-07,
  mvnorm_algorithm = mvtnorm::Miwa(steps = 4097, checkCorr = FALSE, maxval = 1000),
  nloptr_x0 = NULL,
  nloptr_lb = NULL,
  nloptr_ub = NULL,
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-09, xtol_abs = 1e-08,
    xtol_rel = 1e-07, maxeval = 1000, print_level = 0),
  print_progress = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| cP1 | (numeric) allocation ratio nP1 / nT1. Parameter to be optimized if left unspecified. |
| cC1 | (numeric) allocation ratio nC1 / nT1. Parameter to be optimized if left unspecified. |
| alpha | type I error rate. |
| beta | type II error rate. |
| alternative_TP | assumed difference between T and P under H1. Positive values favor T. |

| | |
|---|---|
| `alternative_TC` | assumed difference between T and C under H1. Positive values favor T. |
| `Delta` | non-inferiority margin for the test $X_T - X_C \leq -\Delta$ vs. $X_T - X_C > -\Delta$. |
| `varT` | variance of treatment group. |
| `varP` | variance of placebo group. |
| `varC` | variance of control group. |
| `round_n` | (logical) if TRUE, a design with integer valued sample sizes is returned. |
| `kappa` | (numeric) penalty factor for placebo patients in the default objective function. |
| `objective` | (expression) objective criterion. |
| `inner_tol_objective` | (numeric) used to determine the tolerances for integrals and nuisance optimization problems inside the objective function. |
| `mvnorm_algorithm` | algorithm for multivariate integration passed to `pmvnorm`. |
| `nloptr_x0` | (numeric vector) starting point for optimization. |
| `nloptr_lb` | (numeric vector) lower bound box for box constrained optimization. |
| `nloptr_ub` | (numeric vector) upper bound box for box constrained optimization. |
| `nloptr_opts` | (list) nloptr options. See `nloptr`. |
| `print_progress` | (logical) controls whether optimization progress should be visualized during the calculation. |
| `...` | additional arguments passed along. |

## Details

This function calculates optimal design parameters for a two-stage three-arm gold-standard non-inferiority trial. Run `vignette("Introduction", package = "OptimalGoldstandardDesigns")` to see some examples related to the associated paper (Meis et al. 2023).

Parameters which can be optimized are the allocation ratios for all groups and stages and the futility and efficacy boundaries of the first stage. The allocation ratios are cT2 = nT2 / nT1, cP1 = nP1 / nT1, cP2 = nP2 / nT1, cC1 = nC1 / nT1 and cC2 = nC2 / nT1. Here, nT1 denotes the sample size of the treatment group in the first stage, nP2 the sample size of the placebo group in the second stage, etc. The first stage efficacy boundaries are bTP1e for the treatment vs placebo testing problem, and bTC1e for the treatment vs control non-inferiority testing problem. The futility boundaries are denoted by bTP1f and bTC1f.

If these parameters are left unspecified or set to NULL, they will be included into the optimization process, otherwise they will be considered boundary constraints. You may also supply quoted expressions as arguments for these parameters to solve a constrained optimization problem. For example, you can supply `cT2 = 1`, `cP2 = quote(cP1)`, `cC2 = quote(cC1)` to ensure that the first and second stage allocation ratios are equal.

The design is optimized with respect to the objective criterion given by the parameter `objective`. By default, this is the overall sample size plus an optional penalty for the placebo group sample size, controlled by the parameter `kappa`.

Designs are calculated to fulfill the following constraints: the family-wise type I error rate is controlled at `alpha` under any combination of the two null hypotheses `muT - muP = 0` and `muT - muC`

+ `Delta` = `0`. The power to reject both hypothesis given both alternative hypotheses `muT` - `muP` = `alternative_TP` and `muT` - `muC` + `Delta` = `alternative_TC` + `Delta` is at least `1` - `beta`. Variances are assumed to be given by `varT`, `varP` and `varC`.

If `binding_futility` is `TRUE`, type I error recycling is used. If `always_both_futility_tests` is `TRUE`, it is assumed that futility tests for both hypotheses are performed at interim, regardless of whether the treatment vs placebo null hypothesis was successfully rejected. If `always_both_futility_tests` is `FALSE`, the futility test for the treatment vs. control testing problem only needs to be done if the null for the treatment vs. placebo testing problem was rejected in the first stage.

## Value

Design object (a list) with optimized design parameters.

## References

Meis J, Pilz M, Herrmann C, Bokelmann B, Rauch G, Kieser M (2023). "Optimization of the two-stage group sequential three-arm gold-standard design for non-inferiority trials." *Statistics in Medicine*, **42**(4), 536-558. doi:10.1002/sim.9630.

## Examples

```
# Should take about 2 second with the chosen accuracy
optimize_design_onestage(
  alpha = .025,
  beta = .2,
  alternative_TP = .4,
  alternative_TC = 0,
  Delta = .2,
  mvnorm_algorithm = mvtnorm::Miwa(steps = 512, checkCorr = FALSE, maxval = 1000),
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-03, xtol_abs = 1e-08,
                     xtol_rel = 1e-07, maxeval = 1000, print_level = 0)
)
```

---

## optimize_design_twostage

*Calculate optimal design parameters for a two-stage gold-standard design*

---

## Description

Calculate optimal design parameters for a two-stage gold-standard design

## Usage

```
optimize_design_twostage(
  cT2 = NULL,
  cP1 = NULL,
  cP2 = NULL,
  cC1 = NULL,
  cC2 = NULL,
  bTP1f = NULL,
  bTP1e = NULL,
  bTC1f = NULL,
  bTC1e = NULL,
  alpha = 0.025,
  beta = 0.2,
  alternative_TP = 0.4,
  alternative_TC = 0,
  Delta = 0.2,
  varT = 1,
  varP = 1,
  varC = 1,
  binding_futility = FALSE,
  always_both_futility_tests = TRUE,
  round_n = TRUE,
  lambda = 1,
  kappa = 0,
  eta = 0,
  objective = quote(sqrt(lambda)^2 * ASN[["H11"]] + (1 - sqrt(lambda)) * sqrt(lambda) *
    ASN[["H10"]] + (1 - sqrt(lambda)) * sqrt(lambda) * ASN[["H01"]] + (1 -
    sqrt(lambda))^2 * ASN[["H00"]] + kappa * (sqrt(lambda)^2 * ASNP[["H11"]] + (1 -
    sqrt(lambda)) * sqrt(lambda) * ASNP[["H10"]] + (1 - sqrt(lambda)) * sqrt(lambda) *
    ASNP[["H01"]] + (1 - sqrt(lambda))^2 * ASNP[["H00"]] + eta * cumn[[2]][["P"]]) + eta
    * (cumn[[2]][["T"]] + cumn[[2]][["P"]] + cumn[[2]][["C"]])),
  inner_tol_objective = .Machine$double.eps^0.25,
  mvnorm_algorithm = mvtnorm::Miwa(steps = 128, checkCorr = FALSE, maxval = 1000),
  nloptr_x0 = NULL,
  nloptr_lb = NULL,
  nloptr_ub = NULL,
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-04, xtol_abs = 0.001,
    xtol_rel = 0.01, maxeval = 1000, print_level = 0),
  print_progress = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| cT2 | (numeric) allocation ratio nT2 / nT1. Parameter to be optimized if left unspecified. |
| cP1 | (numeric) allocation ratio nP1 / nT1. Parameter to be optimized if left unspecified. |

| | |
|---|---|
| `cP2` | (numeric) allocation ratio nP2 / nT1. Parameter to be optimized if left unspecified. |
| `cC1` | (numeric) allocation ratio nC1 / nT1. Parameter to be optimized if left unspecified. |
| `cC2` | (numeric) allocation ratio nC2 / nT1. Parameter to be optimized if left unspecified. |
| `bTP1f` | (numeric) first stage futility boundary for the T vs. P testing problem. Parameter to be optimized if left unspecified. |
| `bTP1e` | (numeric) first stage critical value for the T vs. P testing problem. Parameter to be optimized if left unspecified. |
| `bTC1f` | (numeric) first stage futility boundary for the T vs. C testing problem. Parameter to be optimized if left unspecified. |
| `bTC1e` | (numeric) first stage critical value for the T vs. C testing problem. Parameter to be optimized if left unspecified. |
| `alpha` | type I error rate. |
| `beta` | type II error rate. |
| `alternative_TP` | assumed difference between T and P under H1. Positive values favor T. |
| `alternative_TC` | assumed difference between T and C under H1. Positive values favor T. |
| `Delta` | non-inferiority margin for the test $X_T - X_C \leq -\Delta$ vs. $X_T - X_C > -\Delta$. |
| `varT` | variance of treatment group. |
| `varP` | variance of placebo group. |
| `varC` | variance of control group. |
| `binding_futility` | (logical) controls if futility boundaries are binding. |
| `always_both_futility_tests` | (logical) if true, both futility tests are performed after the first stage. If false, a 'completely sequential' testing procedure is employed (see Appendix of (Meis et al. 2023)). |
| `round_n` | (logical) if TRUE, a design with integer valued sample sizes is returned. |
| `lambda` | (numeric) weight of the alternative hypothesis in the default objective function. 1-lambda is the weight of the null hypothesis. |
| `kappa` | (numeric) penalty factor for placebo patients in the default objective function. |
| `eta` | (numeric) penalty factor for the maximum sample size in the default objective function. |
| `objective` | (expression) objective criterion. |
| `inner_tol_objective` | (numeric) used to determine the tolerances for integrals and nuisance optimization problems inside the objective function. |
| `mvnorm_algorithm` | algorithm for multivariate integration passed to `pmvnorm`. |
| `nloptr_x0` | (numeric vector) starting point for optimization. |

| | |
|---|---|
| `nloptr_lb` | (numeric vector) lower bound box for box constrained optimization. |
| `nloptr_ub` | (numeric vector) upper bound box for box constrained optimization. |
| `nloptr_opts` | (list) nloptr options. See `nloptr`. |
| `print_progress` | (logical) controls whether optimization progress should be visualized during the calculation. |
| `...` | additional arguments passed along. |

## Details

This function calculates optimal design parameters for a two-stage three-arm gold-standard non-inferiority trial. Run `vignette("Introduction", package = "OptimalGoldstandardDesigns")` to see some examples related to the associated paper (Meis et al. 2023).

Parameters which can be optimized are the allocation ratios for all groups and stages and the futility and efficacy boundaries of the first stage. The allocation ratios are cT2 = nT2 / nT1, cP1 = nP1 / nT1, cP2 = nP2 / nT1, cC1 = nC1 / nT1 and cC2 = nC2 / nT1. Here, nT1 denotes the sample size of the treatment group in the first stage, nP2 the sample size of the placebo group in the second stage, etc. The first stage efficacy boundaries are bTP1e for the treatment vs placebo testing problem, and bTC1e for the treatment vs control non-inferiority testing problem. The futility boundaries are denoted by bTP1f and bTC1f.

If these parameters are left unspecified or set to NULL, they will be included into the optimization process, otherwise they will be considered boundary constraints. You may also supply quoted expressions as arguments for these parameters to solve a constrained optimization problem. For example, you can supply `cT2 = 1`, `cP2 = quote(cP1)`, `cC2 = quote(cC1)` to ensure that the first and second stage allocation ratios are equal.

The design is optimized with respect to the objective criterion given by the parameter `objective`. The default objective function is described in the Subsection *Optimizing group sequential gold-standard designs* in Section 2 of (Meis et al. 2023). Additionally, this objective includes a term to penalize the maximum sample size of a trial, which can be controlled by the parameter `eta` (default is `eta=0`).

Designs are calculated to fulfill the following constraints: the family-wise type I error rate is controlled at `alpha` under any combination of the two null hypotheses `muT - muP = 0` and `muT - muC + Delta = 0`. The power to reject both hypothesis given both alternative hypotheses `muT - muP = alternative_TP` and `muT - muC + Delta = alternative_TC + Delta` is at least `1 - beta`. Variances are assumed to be given by `varT`, `varP` and `varC`.

If `binding_futility` is `TRUE`, type I error recycling is used. If `always_both_futility_tests` is `TRUE`, it is assumed that futility tests for both hypotheses are performed at interim, regardless of whether the treatment vs placebo null hypothesis was successfully rejected. If `always_both_futility_tests` is `FALSE`, the futility test for the treatment vs. control testing problem only needs to be done if the null for the treatment vs. placebo testing problem was rejected in the first stage.

## Value

Design object (a list) with optimized design parameters.

## References

Meis J, Pilz M, Herrmann C, Bokelmann B, Rauch G, Kieser M (2023). "Optimization of the two-stage group sequential three-arm gold-standard design for non-inferiority trials." *Statistics in Medicine*, **42**(4), 536-558. doi:10.1002/sim.9630.

## Examples

```
# Should take about 15 seconds.

optimize_design_twostage(
  cT2 = 1,
  cP2 = quote(cP1),
  cC2 = quote(cC1),
  bTP1f = -Inf,
  bTC1f = -Inf,
  beta = 0.2,
  alternative_TP = 0.4,
  alternative_TC = 0,
  Delta = 0.2,
  binding_futility = TRUE,
  lambda = .9,
  kappa = 1,
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-01)
)
```

---

`padd_whitespace` *Add whitespace padding to string*

---

## Description

Add whitespace padding to string

## Usage

```
padd_whitespace(str)
```

## Arguments

| | |
|---|---|
| `str` | a character string |

## Value

string with whitespace padding until the full console width

| | |
|---|---|
| `pmvnorm_` | *mvtnorm::pmvnorm, but returns 0 if any lower boundary is larger than any upper boundary* |

## Description

mvtnorm::pmvnorm, but returns 0 if any lower boundary is larger than any upper boundary

## Usage

```
pmvnorm_(upper, lower, ...)
```

## Arguments

| | |
|---|---|
| `upper` | the vector of upper limits of length n. |
| `lower` | the vector of lower limits of length n. |
| `...` | additional parameters passed to `pmvnorm`. |

## Value

The evaluated distribution function is returned, if `keepAttr` is true, with attributes

| | |
|---|---|
| `error` | estimated absolute error |
| `msg` | status message(s). |
| `algorithm` | a `character` string with `class(algorithm)`. |

## See Also

`pmvnorm`

| | |
|---|---|
| `pmvt_` | *mvtnorm::pmvt, but returns 0 if any lower boundary is larger than any upper boundary* |

## Description

mvtnorm::pmvt, but returns 0 if any lower boundary is larger than any upper boundary

## Usage

```
pmvt_(upper, lower, ...)
```

### Arguments

| | |
|---|---|
| `upper` | the vector of upper limits of length n. |
| `lower` | the vector of lower limits of length n. |
| `...` | additional parameters passed to `pmvt`. |

### Value

The evaluated distribution function is returned, if `keepAttr` is true, with attributes

| | |
|---|---|
| `error` | estimated absolute error |
| `msg` | status message(s). |
| `algorithm` | a `character` string with `class(algorithm)`. |

### See Also

`pmvt`

---

## `print.OneStageGoldStandardDesign`

*Printing method for optimal single-stage goldstandard designs*

---

### Description

Printing method for optimal single-stage goldstandard designs

### Usage

```
## S3 method for class 'OneStageGoldStandardDesign'
print(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An object of class OneStageGoldStandardDesign |
| `...` | additional parameters |

### Value

returns the input x invisibly. This functions is used for its side effects, i.e. printing design characteristics to the screen.

## Examples

```
# Should take about 2 second with the chosen accuracy
optimize_design_onestage(
  alpha = .025,
  beta = .2,
  alternative_TP = .4,
  alternative_TC = 0,
  Delta = .2,
  mvnorm_algorithm = mvtnorm::Miwa(steps = 512, checkCorr = FALSE, maxval = 1000),
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-03, xtol_abs = 1e-08,
                     xtol_rel = 1e-07, maxeval = 1000, print_level = 0)
)
```

---

`print.TwoStageGoldStandardDesign`
*Printing method for optimal two-stage goldstandard designs*

---

## Description

Printing method for optimal two-stage goldstandard designs

## Usage

```
## S3 method for class 'TwoStageGoldStandardDesign'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class TwoStageGoldStandardDesign |
| ... | additional parameters |

## Value

returns the input x invisibly. This functions is used for its side effects, i.e. printing design characteristics to the screen.

## Examples

```
# Should take about 15 seconds.

optimize_design_twostage(
  cT2 = 1,
  cP2 = quote(cP1),
  cC2 = quote(cC1),
  bTP1f = -Inf,
  bTC1f = -Inf,
  beta = 0.2,
  alternative_TP = 0.4,
  alternative_TC = 0,
  Delta = 0.2,
  binding_futility = TRUE,
  lambda = .9,
  kappa = 1,
  nloptr_opts = list(algorithm = "NLOPT_LN_SBPLX", ftol_rel = 1e-01)
)
```

# Index