

# Package ‘gtDesign’

May 8, 2026

**Type** Package

**Title** Convex Optimal Designs for Group Testing Experiments

**Version** 0.1.0

**Description** Finite candidate-set approximate optimal designs for group testing and related experiments, using convex optimization and equivalence checks. Implements the information matrix and cost structure for the prevalence / sensitivity / specificity model used in Huang and colleagues (2020), as in Chi-Kuang Yeh, Weng Kee Wong, and Julie Zhou (<[doi:10.48550/arXiv.2508.08445](https://doi.org/10.48550/arXiv.2508.08445)>).

**URL** <https://github.com/chikuang/gtDesign>,  
<https://arxiv.org/abs/2508.08445>

**License** GPL-3

**BugReports** <https://github.com/chikuang/gtDesign/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** CVXR (>= 1.8.1), tibble, MASS

**Suggests** testthat (>= 3.0.0), dplyr, knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Chi-Kuang Yeh [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7057-2096>>),  
Julie Zhou [aut],  
Weng Kee Wong [ctb]

**Maintainer** Chi-Kuang Yeh <[cyehe@gsu.edu](mailto:cyehe@gsu.edu)>

**Repository** CRAN

**Date/Publication** 2026-04-09 15:40:08 UTC

## Contents

calc_Aopt . . . . .	2
calc_copt . . . . .	3
calc_directional_derivatives . . . . .	3
calc_Dopt . . . . .	4
calc_Eopt . . . . .	5
calc_eta_weights_maximin . . . . .	6
calc_multi_directional_derivative . . . . .	7
check_equivalence . . . . .	8
check_equivalence_maximin . . . . .	8
compute_design_SO . . . . .	9
compute_maximin_design . . . . .	10
exact_design_efficiency_maximin . . . . .	12
gt_huang2020_cost . . . . .	13
gt_huang2020_f . . . . .	13
gt_huang2020_lambda . . . . .	14
gt_huang2020_pi . . . . .	14
gt_huang2020_regressor . . . . .	15
maximin_design_workflow . . . . .	15
plot_equivalence . . . . .	17
plot_equivalence_maximin . . . . .	18
rounding_budget_combinations . . . . .	19
rounding_run_size_combinations . . . . .	19
round_gt_design_budget . . . . .	20
round_gt_design_budget_maximin . . . . .	21
round_gt_design_n_maximin . . . . .	22
round_gt_design_subject_budget . . . . .	23
<b>Index</b>	<b>25</b>

---

calc_Aopt	<i>A-optimal approximate design for group testing experiments</i>
-----------	---

---

### Description

A-optimal approximate design for group testing experiments

### Usage

```
calc_Aopt(u, f, solver = "CLARABEL", ..., drop_tol = 1e-08)
```

### Arguments

u	Candidate design points (e.g. pool sizes or coded pool layouts).
f	Function returning a numeric vector at one design point.
solver	Solver name passed to CVXR. Default is "CLARABEL".
...	Additional arguments passed to <code>CVXR::psolve()</code> .
drop_tol	Threshold for removing near-zero weights.

**Value**

An object of class "gt\_a\_design" and "gt\_design".

---

calc_copt	<i>c-optimal approximate design for group testing experiments</i>
-----------	---

---

**Description**

c-optimal approximate design for group testing experiments

**Usage**

```
calc_copt(u, f, cVec, solver = "CLARABEL", ..., drop_tol = 1e-10)
```

**Arguments**

u	Candidate design points (e.g. pool sizes or coded pool layouts).
f	Function returning a numeric vector at one design point.
cVec	Numeric vector specifying the linear combination $c^\top \theta$ .
solver	Solver name passed to CVXR. Default is "CLARABEL".
...	Additional arguments passed to <code>CVXR::psolve()</code> .
drop_tol	Threshold for removing near-zero weights.

**Value**

An object of class "gt\_c\_design" and "gt\_design".

---

calc_directional_derivatives	<i>Directional derivatives on a finite candidate set</i>
------------------------------	--

---

**Description**

Directional derivatives on a finite candidate set

**Usage**

```
calc_directional_derivatives(
  u,
  M,
  f,
  criteria = c("D"),
  cVec = NULL,
  use_ginv = TRUE,
  eig_tol = NULL
)
```

**Arguments**

u	Candidate design points.
M	Information matrix.
f	Regression function returning a numeric vector.
criteria	Character vector of criteria, e.g. c("D", "A", "c", "E").
cVec	Optional vector for c-optimality.
use_ginv	Logical; if TRUE, use <code>MASS::ginv()</code> when M is singular.
eig_tol	Tolerance for grouping eigenvalues with $\lambda_{\min}(\mathbf{M})$ when computing E-optimal derivatives (multiplicity $r^*$ ). Passed to the eigenvalue comparison.

**Value**

A named list of directional derivative vectors.

**E-optimality**

Implements  $d_E$  from Sec. 4 (Eq. (9)) of Yeh, Wong, and Zhou (arXiv:2508.08445): with orthonormal  $\mathbf{Q}(\mathbf{w})$  spanning the eigenspace of  $\lambda_{\min}(\mathbf{M})$  and multiplicity  $r^*$ , uses  $\mathbf{B} = (1/r^*)\mathbf{I}_{r^*}$  (trace one, positive semi-definite), so

$$d_E(u_i, \mathbf{w}) = \frac{1}{r^*} \|\mathbf{Q}^\top \mathbf{f}(u_i)\|^2 - \lambda_{\min}(\mathbf{M}).$$

When  $r^* = 1$ , this is  $(\mathbf{v}^\top \mathbf{f}(u_i))^2 - \lambda_{\min}(\mathbf{M})$  for a unit eigenvector  $\mathbf{v}$ .

---

calc\_Dopt

*D-optimal approximate design for group testing experiments*

---

**Description**

Computes a D-optimal approximate design on a finite candidate set. The regression map  $\mathbf{f}$  typically encodes contributions to the Fisher information matrix (e.g. score vectors or square-root information); see `gt_huang2020_regressor()` for the Huang et al. (2020) three-parameter model.

**Usage**

```
calc_Dopt(
  u,
  f,
  solver = "CLARABEL",
  ...,
  drop_tol = 1e-08,
  ridge = 1e-08,
  use_matrix_form = FALSE
)
```

**Arguments**

u	Candidate design points (e.g. pool sizes or coded pool layouts).
f	Function returning a numeric vector at one design point.
solver	Solver name passed to CVXR. Default is "CLARABEL".
...	Additional arguments passed to <code>CVXR::psolve()</code> .
drop_tol	Threshold for removing near-zero weights.
ridge	Small diagonal ridge for numerical stability in <code>log_det</code> .
use_matrix_form	If TRUE, build M via dense matrix multiply.

**Value**

An object of class "gt\_d\_design" and "gt\_design".

---

calc_Eopt	<i>E-optimal approximate design for group testing experiments</i>
-----------	---

---

**Description**

Minimizes  $-\lambda_{\min}(\mathbf{M})$  over  $\mathbf{w}$  on a finite candidate set via `CVXR::Minimize(-CVXR::lambda_min(...))`. The reported objective is  $\text{loss}(\mathbf{M}) = -\lambda_{\min}(\mathbf{M})$ , matching MATLAB `loss = -lambda_min(FIM)` (e.g. `calc_loss_E`).

**Usage**

```
calc_Eopt(u, f, solver = "CLARABEL", ..., drop_tol = 1e-08, ridge = 1e-08)
```

**Arguments**

u	Candidate design points (e.g. pool sizes or coded pool layouts).
f	Function returning a numeric vector at one design point.
solver	Solver name passed to CVXR. Default is "CLARABEL".
...	Additional arguments passed to <code>CVXR::psolve()</code> .
drop_tol	Threshold for removing near-zero weights.
ridge	Small diagonal ridge for numerical stability in <code>log_det</code> .

**Details**

This matches `compute_design_S0()` with `criterion = "E"` (same scalar loss).

**Value**

An object of class "gt\_e\_design" and "gt\_design". Components `value` and `optval` are the minimization objective  $-\lambda_{\min}(\mathbf{M})$ : `value` is recomputed at the cleaned support, `optval` is the solver optimum of the minimize objective (they may differ slightly after `drop_tol`).

**Examples**

```
# Huang et al. (2020) model on {1,...,61}, q = 0 (same setting as D-opt example)
theta <- c(p0 = 0.07, p1 = 0.93, p2 = 0.96)
u <- seq_len(61L)
f <- gt_huang2020_regressor(theta, q = 0)
res_E <- calc_Eopt(u, f, drop_tol = 1e-6)
res_E$design
res_E$value # = -lambda_min(M), same as -min(eigen(M)$values)
-res_E$value # lambda_min(M) at the optimum
res_E$status
```

---

calc\_eta\_weights\_maximin

*Eta weights for the maximin equivalence theorem*

---

**Description**

Solves for nonnegative weights in the maximin equivalence theorem. The returned vector `eta` satisfies the normalization condition induced by the derivative of the active right-hand sides and enforces a nonpositive weighted combination of directional derivatives across the candidate set.

**Usage**

```
calc_eta_weights_maximin(
  tstar,
  loss_ref,
  loss_model,
  directional_derivatives,
  criteria,
  q,
  tol = 1e-06,
  complementary_slack = TRUE,
  fallback_no_slack = TRUE,
  complementary_tol_mult = 100,
  solver = "CLARABEL",
  solvers_fallback = c("SCS", "ECOS"),
  ...
)
```

**Arguments**

<code>tstar</code>	Optimal maximin scalar returned by <code>compute_maximin_design()</code> .
<code>loss_ref</code>	Named list of reference losses from single-objective designs.
<code>loss_model</code>	Named list of achieved losses at the maximin design.

directional_derivatives	Named list of directional derivative vectors, such as dD, dA, dDs, dc, and dE.
criteria	Character vector of criteria, e.g. c("D", "A").
q	Parameter dimension.
tol	Numerical tolerance for the combined directional derivative and (if used) complementary slackness.
complementary_slack	Logical; if TRUE, first attempt includes complementary slackness. If that LP is infeasible and fallback_no_slack is TRUE, a second attempt omits those constraints.
fallback_no_slack	Logical; default TRUE. See complementary_slack.
complementary_tol_mult	Positive multiplier applied to tol for the complementary slackness bounds (helps feasibility).
solver	Solver passed to <code>CVXR::psolve()</code> for the first attempt.
solvers_fallback	Character vector of solvers to try if the first attempt fails (default includes SCS and ECOS, which often succeed when CLARABEL reports infeasible on small LPs).
...	Additional arguments passed to <code>CVXR::psolve()</code> .

### Details

The linear program from Gao et al. also imposes approximate **complementary slackness**  $|\eta_k * \text{gap}_k| \leq \text{tol}$ . With tight tol, that set can be **empty** together with the normalization constraint (common with three or more criteria). By default this function **retries without** complementary slackness if the strict problem is infeasible, and cycles a few solvers (works well with **CVXR 1.8.x**).

### Value

A named numeric vector of eta weights.

---

calc\_multi\_directional\_derivative  
*Weighted multi-objective directional derivative*

---

### Description

Weighted multi-objective directional derivative

### Usage

calc\_multi\_directional\_derivative(dd\_list, eta)

**Arguments**

dd\_list      A named list of directional derivative vectors.  
 eta          A named numeric vector of weights.

**Value**

A numeric vector of combined directional derivative values.

---

check\_equivalence      *Check the equivalence theorem on a finite candidate set*

---

**Description**

Uses the same directional derivatives as `calc_directional_derivatives()` with a single criterion. For maximin designs, use `check_equivalence_maximin()`.

**Usage**

```
check_equivalence(design_obj, f, u = NULL, tol = 1e-06, use_ginv = TRUE)
```

**Arguments**

design\_obj      Output from `calc_Dopt()`, `calc_Aopt()`, or `calc_copt()`.  
 f              Regression function returning a numeric vector.  
 u              Candidate design points. If omitted, uses `design_obj$candidates`.  
 tol            Numerical tolerance for checking nonpositivity and equality.  
 use\_ginv      Passed to `calc_directional_derivatives()` when M is singular.

**Value**

A list with directional derivative values and theorem checks.

---

check\_equivalence\_maximin  
                                  *Check the equivalence theorem for a maximin design*

---

**Description**

Computes the weighted combined directional derivative  $\sum_j \eta_j d_j(x)$  over the candidate set and checks whether it is nonpositive, with approximate equality at the support points.

**Usage**

```

check_equivalence_maximin(
  design_obj,
  directional_derivatives,
  eta,
  tol = 1e-06
)

```

**Arguments**

`design_obj` Output from `compute_maximin_design()`.

`directional_derivatives` Named list of directional derivative vectors, such as `dD`, `dA`, `dDs`, `dc`, and `dE`.

`eta` Named numeric vector of eta weights, typically returned by `calc_eta_weights_maximin()`.

`tol` Numerical tolerance used in the checks.

**Value**

A list of class `gt_equivalence_maximin`.

---

<code>compute_design_SO</code>	<i>Single-objective optimal approximate design (group testing)</i>
--------------------------------	--

---

**Description**

Solves a convex optimization problem over a finite candidate set for one criterion. The regression function `f` encodes rank-one contributions to the (approximate) information matrix, as in nonlinear design or group testing Fisher information assembled from pool outcomes.

**Usage**

```

compute_design_SO(
  u,
  f,
  criterion = c("D", "A", "Ds", "c", "E"),
  opts = list(),
  info_weight = NULL,
  solver = "CLARABEL",
  ...,
  support_tol = 1e-04,
  drop_tol = 1e-08
)

```

**Arguments**

u	Candidate design points.
f	A function returning the regression vector at a single design point.
criterion	One of "D", "A", "Ds", "c", or "E".
opts	Named list of extra options. For "Ds" use opts\$cVec_Ds; for "c" use opts\$cVec_c.
info_weight	Optional function returning a nonnegative scalar multiplier for each rank-one information contribution.
solver	Solver passed to <code>CVXR::psolve()</code> .
...	Additional arguments passed to <code>CVXR::psolve()</code> .
support_tol	Weights smaller than this are dropped from the reported support.
drop_tol	Numerical tolerance for tiny solver noise before support cleanup.

**Value**

A list of class "gt\_so\_design" and "gt\_design".

---

compute\_maximin\_design

*Maximin multi-criterion approximate design (group testing)*

---

**Description**

Joint convex formulation for maximizing the minimum efficiency across several criteria, using reference losses from single-objective designs. Suitable when multiple experimental goals (e.g. overall prevalence versus contrasts) must be balanced in group testing allocation.

**Usage**

```
compute_maximin_design(
  u,
  f,
  loss_ref,
  criteria,
  opts = list(),
  info_weight = NULL,
  solver = "CLARABEL",
  ...,
  support_tol = 1e-04,
  drop_tol = 1e-08
)
```

**Arguments**

u	Candidate design points.
f	A function returning the regression vector at a single design point.
loss_ref	Named list of reference losses from single-objective designs, on the same scale as <code>compute_design_S0()</code> internal losses.
criteria	Character vector containing any of "D", "A", "Ds", "c", or "E".
opts	Named list of extra options. For "Ds" use <code>opts\$cVec_Ds</code> ; for "c" use <code>opts\$cVec_c</code> .
info_weight	Optional function returning a nonnegative scalar multiplier for each rank-one information contribution.
solver	Solver passed to <code>CVXR::psolve()</code> .
...	Additional arguments passed to <code>CVXR::psolve()</code> .
support_tol	Weights smaller than this are dropped from the reported support.
drop_tol	Numerical tolerance for tiny solver noise before support cleanup.

**Details**

**Link to Yeh, Wong, and Zhou (arXiv:2508.08445, Sec. 4.1).** Let  $\mathbf{M}(\mathbf{w}) = \mathbf{I}(\mathbf{w}, \theta^*)$  be the information matrix on the candidate set, and let  $\phi_D, \phi_A, \phi_c, \phi_E$  be the single-objective functionals in their eq. (4), with efficiencies  $\text{Eff}_j(\mathbf{w})$  in eq. (5).

- **Eq. (6)** — Maximin objective:

$$\max_{\mathbf{w} \in \Delta_M} \min\{\text{Eff}_1(\mathbf{w}), \dots, \text{Eff}_K(\mathbf{w})\}.$$

At the optimum,  $\min_j \text{Eff}_j = 1/t^*$  for the scalar  $t^*$  below (paper's  $t$ ).

- **Eq. (7)** — Equivalent convex program (Gao et al., 2025):

$$\min_{\mathbf{w}, t \geq 0} t \quad \text{s.t.} \quad \Phi_j(\mathbf{w}) \leq h_j(t), \quad j = 1, \dots, K,$$

where  $\Phi_j$  is  $\log \phi_D$  for **D**,  $\phi_A$  for **A**,  $\phi_c$  for **c** / **Ds**, and  $\phi_E$  for **E**, as in the paper. This function minimizes  $t^*$  ( $t$  in the paper).

- **Eq. (8)** — Bounds  $h_j(t)$  on the right-hand side (with  $s = \dim(\theta)$ ):

$$h_D(t) = \log \phi_D(\mathbf{w}_D^*) + s \log t, \quad h_A(t) = t \phi_A(\mathbf{w}_A^*), \quad h_c(t) = t \phi_c(\mathbf{w}_c^*), \quad h_E(t) = \phi_E(\mathbf{w}_E^*)/t.$$

Implemented constraints (same structure; build `loss_ref` from `calc_Dopt()`, `calc_Aopt()`, `calc_copt()`, `calc_Eopt()`):

- **D**:  $-\log \det(\mathbf{M}) \leq \text{loss}_D^{\text{ref}} + s \log(t^*)$  with  $\text{loss}_D^{\text{ref}} = -\log \det(\mathbf{M}_D^*)$ .
- **A**:  $\text{tr}(\mathbf{M}^{-1}) \leq t^* \text{loss}_A^{\text{ref}}$ .
- **c / Ds**:  $\mathbf{c}^\top \mathbf{M}^{-1} \mathbf{c} \leq t^* \text{loss}_c^{\text{ref}}$ .
- **E**:  $-\lambda_{\min}(\mathbf{M}) \leq \text{loss}_E^{\text{ref}}/t^*$  with  $\text{loss}_E^{\text{ref}} = -\lambda_{\min}(\mathbf{M}_E^*)$  as from `calc_Eopt()`.

Reference losses must match the paper's  $\phi$ -scale and the same conventions as `compute_design_S0()` (internal scalar loss).

**Value**

A list of class "gt\_maximin\_design" and "gt\_design".

**References**

Yeh, C.-K., Wong, W. K., Zhou, J. (2025). Single and multi-objective optimal designs for group testing experiments. *arXiv* 2508.08445. Sec. 4.1, eq. (6)–(8).

---

exact\_design\_efficiency\_maximin

*Efficiencies of an exact design relative to single-objective optima (maximin)*

---

**Description**

Computes criterion-wise efficiencies as in eq. (5) of Yeh, Wong, and Zhou (arXiv:2508.08445), and their minimum (**MinEff** in Table 4). Use after `round_gt_design_budget()` with a maximin approximate design: pass `M_exact = out$M_exact`, the same `loss_ref` as in `compute_maximin_design()`, and the same criteria vector.

**Usage**

```
exact_design_efficiency_maximin(
  M_exact,
  loss_ref,
  criteria,
  opts = list(),
  p = nrow(M_exact)
)
```

**Arguments**

<code>M_exact</code>	Information matrix of the exact (rounded) design (e.g. <code>out\$M_exact</code> from <code>round_gt_design_budget()</code> ).
<code>loss_ref</code>	Named list of reference losses from single-objective designs, on the same scale as <code>compute_maximin_design()</code> .
<code>criteria</code>	Character vector of criteria (e.g. <code>c("D", "A")</code> ).
<code>opts</code>	Optional list for contrasts ( <code>cVec_Ds</code> , <code>cVec_c</code> ) for $D_s$ - and $c$ -type criteria.
<code>p</code>	Parameter dimension; defaults to <code>nrow(M_exact)</code> .

**Value**

A list with efficiencies (named numeric vector) and `min_efficiency` ( $\min_j \text{Eff}_j$ ).

---

gt\_huang2020\_cost      *Standardized cost for pool size (Huang et al. 2020)*

---

**Description**

$$c(x) = 1 - q + qx$$

with  $q = q_1/(q_0 + q_1)$  for assay cost  $q_0$  and enrollment cost  $q_1$ . Value  $q = 0$  gives  $c(x) \equiv 1$ .

**Usage**

gt\_huang2020\_cost(x, q)

**Arguments**

x                      Pool size (positive integer).  
q                        Cost ratio in  $[\theta, 1]$ .

**Value**

Scalar cost.

---

gt\_huang2020\_f      *Gradient vector  $\mathbf{f}(x)$  for the Huang et al. (2020) model*

---

**Description**

Partial derivatives of  $\pi(x | \theta)$  with respect to  $(p_0, p_1, p_2)$  as in arXiv:2508.08445 (Eq. below their Eq. (1)).

**Usage**

gt\_huang2020\_f(x, theta)

**Arguments**

x                        Pool size.  
theta                     $(p_0, p_1, p_2)$ .

**Value**

Length-3 numeric vector.

---

gt\_huang2020\_lambda     *Weight  $\lambda(x)$  in the information matrix (Huang et al. 2020)*

---

### Description

$$\lambda(x) = [c(x) \pi(x) \{1 - \pi(x)\}]^{-1}$$

### Usage

gt\_huang2020\_lambda(x, theta, q)

### Arguments

x                     Pool size.  
 theta                Same as for [gt\\_huang2020\\_pi](#).  
 q                     Same as for [gt\\_huang2020\\_cost](#).

### Value

Scalar (positive when  $\pi$  is interior).

---

gt\_huang2020\_pi     *Positive test probability (Huang et al. 2020 / arXiv:2508.08445 Sec. 2)*

---

### Description

$$\pi(x | \theta) = p_1 - (p_1 + p_2 - 1)(1 - p_0)^x$$

### Usage

gt\_huang2020\_pi(x, theta)

### Arguments

x                     Pool size (positive integer).  
 theta                 $(p_0, p_1, p_2)$  prevalence, sensitivity, specificity.

### Value

Scalar probability in  $(0, 1)$  when well-defined.

---

 gt\_huang2020\_regressor

*Effective regressor  $\sqrt{\lambda(x)} \mathbf{f}(x)$  for convex design code*


---

### Description

The information matrix in Huang et al. (2020) is  $\sum_i w_i \lambda(x_i) \mathbf{f}(x_i) \mathbf{f}(x_i)^\top$ . The same matrix equals  $\sum_i w_i \mathbf{h}(x_i) \mathbf{h}(x_i)^\top$  with  $\mathbf{h}(x) = \sqrt{\lambda(x)} \mathbf{f}(x)$ , which matches the regression form used by [calc\\_Dopt\(\)](#) and [check\\_equivalence\(\)](#) without an `info_weight` argument.

### Usage

```
gt_huang2020_regressor(theta, q = 0)
```

### Arguments

theta	Nominal $(p_0, p_1, p_2)$ for local optimality.
q	Cost ratio in $[\theta, 1]$ ; use $\theta$ for constant cost per test.

### Value

A function `function(x)` returning a length-3 vector.

### References

Yeh, C.-K., Wong, W. K., and Zhou, J. (2025). Single and multi-objective optimal designs for group testing experiments. arXiv:2508.08445.

Huang, S.-Y., Chen, Y.-H., and Wang, W. (2020). Optimal group testing designs for estimating prevalence with imperfect tests. Journal of the Royal Statistical Society Series C.

---

 maximin\_design\_workflow

*Maximin design with reference losses, equivalence, and eta (workflow)*


---

### Description

Runs the usual multi-step maximin pipeline: single-objective designs to build `loss_ref`, [compute\\_maximin\\_design\(\)](#), [calc\\_directional\\_derivatives\(\)](#), [calc\\_eta\\_weights\\_maximin\(\)](#), and optionally [check\\_equivalence\\_maximin\(\)](#).

**Usage**

```

maximin_design_workflow(
  u,
  f,
  criteria,
  opts = list(),
  drop_tol = 1e-06,
  q = NULL,
  tol_eta = 0.001,
  tol_equiv = 0.002,
  check_equiv = TRUE,
  keep_reference_designs = FALSE,
  info_weight = NULL,
  solver = "CLARABEL",
  eta_args = list(),
  deriv_args = list(),
  make_figure = FALSE,
  plot_args = list(),
  ...
)

```

**Arguments**

<code>u</code>	Candidate design points.
<code>f</code>	Regression function (same as in single-objective and maximin calls).
<code>criteria</code>	Character vector of criteria, subset of <code>c("D", "A", "c")</code> .
<code>opts</code>	Named list passed to contrast-based criteria; use <code>cVec_c</code> for <code>c</code> .
<code>drop_tol</code>	Passed to <code>calc_Dopt()</code> , <code>calc_Aopt()</code> , <code>calc_copt()</code> .
<code>q</code>	Parameter dimension for <code>calc_eta_weights_maximin()</code> ; default is the length of <code>f(u[1])</code> .
<code>tol_eta</code>	Tolerance passed to <code>calc_eta_weights_maximin()</code> (unless overridden in <code>eta_args</code> ).
<code>tol_equiv</code>	Tolerance for <code>check_equivalence_maximin()</code> when <code>check_equiv</code> is TRUE.
<code>check_equiv</code>	If TRUE, run <code>check_equivalence_maximin()</code> .
<code>keep_reference_designs</code>	If TRUE, include full outputs of each single-objective solve in <code>reference_designs</code> .
<code>info_weight</code>	Passed to <code>compute_maximin_design()</code> .
<code>solver</code>	Passed to all convex solves in this pipeline.
<code>eta_args</code>	Named list of extra arguments for <code>calc_eta_weights_maximin()</code> (e.g. <code>list(solver = "SCS", complementary_slack = FALSE)</code> ).
<code>deriv_args</code>	Named list of extra arguments for <code>calc_directional_derivatives()</code> .
<code>make_figure</code>	If TRUE, draw the maximin equivalence figure via <code>plot_equivalence_maximin()</code> (opens a graphics device).
<code>plot_args</code>	Named list of extra arguments for <code>plot_equivalence_maximin()</code> (merged over defaults that pass <code>tol = tol_equiv</code> ).
<code>...</code>	Additional arguments passed to <code>compute_maximin_design()</code> (e.g. <code>support_tol</code> ).

**Details**

Only criteria **D**, **A**, and **c** are supported end-to-end (directional derivatives for **Ds** / **E** are not wired here). For **c**, supply `opts = list(cVec_c = ...)` as in `compute_maximin_design()`.

**Value**

A list with components: `loss_ref`, `maximin` (output of `compute_maximin_design()`), `directional_derivatives`, `eta`, `equivalence` (or `NULL` if `check_equiv` is `FALSE`), `q`, and optionally `reference_designs`. Plotting is a side effect when `make_figure` is `TRUE`.

---

plot_equivalence	<i>Plot equivalence theorem directional derivative</i>
------------------	--

---

**Description**

Plot equivalence theorem directional derivative

**Usage**

```
plot_equivalence(
  eq_obj,
  main = NULL,
  xlab = "Design point",
  ylab = "Directional derivative",
  pch_support = 19,
  ...
)
```

**Arguments**

<code>eq_obj</code>	Output from <code>check_equivalence()</code> .
<code>main</code>	Plot title.
<code>xlab</code>	X-axis label.
<code>ylab</code>	Y-axis label.
<code>pch_support</code>	Plotting symbol for support points.
<code>...</code>	Passed to <code>graphics::plot()</code> .

**Value**

Invisibly returns `eq_obj`.

---

`plot_equivalence_maximin`*Plot equivalence diagnostics for a maximin design*

---

**Description**

Plot equivalence diagnostics for a maximin design

**Usage**

```
plot_equivalence_maximin(  
  design_obj,  
  directional_derivatives,  
  eta,  
  criteria = NULL,  
  main_prefix = NULL,  
  line_width = 2,  
  show_support = TRUE,  
  tol = 1e-06,  
  cex_lab = 1.2,  
  cex_axis = 1,  
  cex_main = 1.1,  
  mar = c(5, 6.5, 3, 1)  
)
```

**Arguments**

<code>design_obj</code>	Output from <code>compute_maximin_design()</code> .
<code>directional_derivatives</code>	Named list of directional derivative vectors.
<code>eta</code>	Named numeric vector of eta weights.
<code>criteria</code>	Character vector of criteria to display.
<code>main_prefix</code>	Optional prefix for panel titles.
<code>line_width</code>	Line width for curves.
<code>show_support</code>	Logical; if TRUE, show vertical lines at support points.
<code>tol</code>	Numerical tolerance used for the horizontal reference line.
<code>cex_lab, cex_axis, cex_main, mar</code>	Graphics parameters.

**Value**

Invisibly returns the object produced by `check_equivalence_maximin()`.

---

rounding\_budget\_combinations

*Enumerate budget-feasible integer allocations (cost-aware rounding)*


---

### Description

All nonnegative integer vectors  $\Delta$  with  $\Delta^\top c \leq C_r$ , then restricted to **tight** rows where remaining budget is insufficient for any additional unit at minimum cost (Step II in **Rounding Algorithm II**, arXiv:2508.08445, Sec. 5.1).

### Usage

```
rounding_budget_combinations(cxx, Cr)
```

### Arguments

cxx	Per-unit costs $c(x_j)$ for each extended support location (same length as the number of columns in the search grid).
Cr	Remaining budget after floor allocation.

### Value

Matrix whose first  $\text{length}(\text{cxx})$  columns are counts, then used cost, then remaining budget. Sorted like the MATLAB reference implementation.

---

rounding\_run\_size\_combinations

*Enumerate nonnegative integer allocations (run-size rounding)*


---

### Description

All vectors  $(z_1, \dots, z_m)$  with  $z_j \geq 0$ ,  $\sum_j z_j \leq \text{max\_run\_size}$ , matching the supplementary **Rounding Algorithm I** search (arXiv:2508.08445, Sec. 5.1).

### Usage

```
rounding_run_size_combinations(n_items, max_run_size)
```

### Arguments

n_items	Number of locations (columns).
max_run_size	Upper bound on $\sum_j z_j$ .

**Value**

Numeric matrix with columns  $z_1, \dots, z_m$ , then total count, then remaining slots ( $\text{max\_run\_size} - \text{total}$ ), sorted lexicographically with the last coordinate changing fastest (MATLAB sortrows order).

---

round\_gt\_design\_budget

*Exact group-testing design under a fixed budget (Rounding Algorithm II)*

---

**Description**

Implements the floor allocation, zero-fix, extended-support search, and best-merge steps in Sec. 5.1 of Yeh, Wong, and Zhou (arXiv:2508.08445), as in the reference MATLAB scripts `GT_Single_budget.m` / `calc_budget_round_combinations.m`.

**Usage**

```
round_gt_design_budget(
    approx_design,
    u,
    theta,
    C,
    q_cost,
    criterion = c("D", "A", "c", "Ds", "E"),
    opts = list(),
    n_index = 2L,
    fix_zero_floor = TRUE,
    repair_floor_budget = TRUE,
    ...
)
```

**Arguments**

approx_design	Output of <code>calc_Dopt()</code> , <code>calc_Aopt()</code> , <code>calc_copt()</code> , <code>calc_Eopt()</code> , or <code>compute_design_SO()</code> on the same candidate set $u$ .
u	Integer candidate pool sizes (same grid as used for <code>approx_design</code> ).
theta	Nominal $(p_0, p_1, p_2)$ .
C	Total cost budget (sum of run costs $n_i c(x_i)$ ).
q_cost	Cost ratio $q$ in $c(x) = 1 - q + qx$ .
criterion	One of "D", "A", "c", "Ds", "E".
opts	Contrast options: <code>cVec_c</code> and/or <code>cVec_Ds</code> when needed (not used for "E").
n_index	Half-width of the extended support window $(x_i \pm n\_index)$ .

`fix_zero_floor` If TRUE (default), redistribute runs so no support pool has zero runs after floor, as in `GT_MO_rounding_budget.m`. If FALSE, skip this step (as in `GT_Single_budget.m`, which only floors).

`repair_floor_budget` If TRUE (default), after zero-fix, remove runs from the most expensive active pools until  $\sum_i n_i c(x_i) \leq C$ , so `design_round1` never exceeds the budget. Zero-fix can increase total cost because  $c(x)$  varies by pool size; without this step the floor design can strictly overspend. Set to FALSE only to match reference MATLAB MO scripts that skip this repair (then overspend is possible).

... Reserved.

**Value**

A list with `design_round1` (2-row matrix: pool sizes, floor counts; columns with zero runs are removed; no column names), `design_exact` (same format after merge), `delta` (2-row matrix: pool sizes  $x$  and  $\Delta_n = n_{\text{exact}} - n_{\text{round1}}$ ; only pool sizes with nonzero change are columns), `M_approx`, `M_exact`, `efficiency`, `loss_approx`, `loss_exact`, `C_remaining` after the floor step (with default `repair_floor_budget`, total cost is at most  $C$ ):  $\max\{0, \text{round}(C - \sum_i n_i c(x_i), 4)\}$  (never negative), `extension_table` (`tibble::tibble()`) of tight extensions; first columns named by extended-support pool sizes, then `used_cost`, `remaining`, `loss` when present), and `criterion`.

---

round\_gt\_design\_budget\_maximin

*Exact maximin design under fixed budget (Algorithm II + search)*

---

**Description**

Applies cost-aware floor rounding, then modified Step II search over nearby support points ( $x_i \pm n_{\text{index}}$ ) and selects the extension with the largest minimum efficiency (MinEff) across criteria.

**Usage**

```
round_gt_design_budget_maximin(
  approx_design,
  u,
  theta,
  C,
  q_cost,
  loss_ref,
  criteria,
  opts = list(),
  n_index = 2L,
  fix_zero_floor = TRUE,
  repair_floor_budget = TRUE,
  ...
)
```

**Arguments**

approx_design	A maximin/approximate design object with component design containing columns point and weight (e.g. output of <code>compute_maximin_design()</code> ).
u	Integer candidate pool sizes (same grid as used for approx_design).
theta	Nominal $(p_0, p_1, p_2)$ .
C	Total cost budget (sum of run costs $n_i c(x_i)$ ).
q_cost	Cost ratio $q$ in $c(x) = 1 - q + qx$ .
loss_ref	Named list of reference losses from single-objective designs (same scale as <code>compute_maximin_design()</code> ).
criteria	Character vector of criteria to evaluate MinEff, e.g. <code>c("D", "A")</code> or <code>c("D", "A", "Ds")</code> .
opts	Contrast options: <code>cVec_c</code> and/or <code>cVec_Ds</code> when needed.
n_index	Half-width of extension window around each support point.
fix_zero_floor	If TRUE (default), run MATLAB-style zero-fix after floor allocation.
repair_floor_budget	If TRUE (default), cap floor design to satisfy budget by removing runs from the most expensive active pools.
...	Reserved.

**Value**

A list with `design_round1`, `design_exact`, `delta`, `M_round1`, `M_exact`, `efficiencies`, `min_efficiency`, `C_remaining` after floor, and `extension_table` (tibble sorted by decreasing `min_efficiency`).

---

round\_gt\_design\_n\_maximin

*Exact maximin design under fixed run size (Algorithm I + search)*

---

**Description**

Rounds a maximin approximate design to an exact design with total run size  $n$ , then applies the modified Step II search over nearby support points  $(x_i \pm n\_index)$  and selects the extension with the largest minimum efficiency (MinEff) across criteria.

**Usage**

```
round_gt_design_n_maximin(
  approx_design,
  u,
  theta,
  n,
  q_cost,
  loss_ref,
  criteria,
```

```

  opts = list(),
  n_index = 2L,
  ...
)

```

### Arguments

approx_design	A maximin/approximate design object with component design containing columns point and weight (e.g. output of <code>compute_maximin_design()</code> ).
u	Integer candidate pool sizes (same grid as used for approx_design).
theta	Nominal $(p_0, p_1, p_2)$ .
n	Total run size (fixed sample size).
q_cost	Cost ratio $q$ in $c(x) = 1 - q + qx$ .
loss_ref	Named list of reference losses from single-objective designs (same scale as <code>compute_maximin_design()</code> ).
criteria	Character vector of criteria to evaluate MinEff, e.g. <code>c("D", "A")</code> or <code>c("D", "A", "Ds")</code> .
opts	Contrast options: <code>cVec_c</code> and/or <code>cVec_Ds</code> when needed.
n_index	Half-width of extension window around each support point.
...	Reserved.

### Value

A list with `design_round1`, `design_exact`, `delta`, `M_round1`, `M_exact`, `efficiencies` (named vector), `min_efficiency`, and `extension_table` (tibble sorted by decreasing `min_efficiency`).

---

round\_gt\_design\_subject\_budget

*Exact design under subject-count constraint via budget rounding*

---

### Description

Converts a subject cap (`n_subject_allow`) into a test budget  $C_{\text{tests}} = \text{floor}(n_{\text{subject\_allow}} / \sum(w_i x_i))$  using the approximate design, runs `round_gt_design_budget()`, then selects the best rounded extension under the subject-count constraint.

### Usage

```

round_gt_design_subject_budget(
  approx_design,
  u,
  theta,
  n_subject_allow,
  q_cost,
  criterion = c("D", "A", "c", "Ds", "E"),
  opts = list(),

```

```

    n_index = 2L,
    fix_zero_floor = TRUE,
    repair_floor_budget = TRUE,
    ...
)

```

### Arguments

approx_design	Output of <code>calc_Dopt()</code> , <code>calc_Aopt()</code> , <code>calc_copt()</code> , or <code>compute_design_S0()</code> on candidate set <code>u</code> .
u	Integer candidate pool sizes.
theta	Nominal $(p_0, p_1, p_2)$ .
n_subject_allow	Subject-count cap.
q_cost	Cost ratio $q$ in $c(x) = 1 - q + qx$ .
criterion	One of "D", "A", "c", "Ds", "E".
opts	Contrast options: <code>cVec_c</code> and/or <code>cVec_Ds</code> when needed.
n_index	Half-width of extension window.
fix_zero_floor	Passed to <code>round_gt_design_budget()</code> .
repair_floor_budget	Passed to <code>round_gt_design_budget()</code> .
...	Reserved.

### Details

Selection rule follows the application script logic: among feasible candidates (subjects  $\leq$  n\_subject\_allow), prefer using more subjects (closest to the cap), then smaller loss. If none are feasible, pick the minimum-excess candidate.

### Value

A list with `C_tests`, `n_subject_allow`, `n_subjects_used`, `design_round1`, `design_exact`, `delta`, `M_approx`, `M_exact`, `loss_approx`, `loss_exact`, `efficiency`, `criterion`, and `extension_table`.

# Index

`calc_Aopt`, 2  
`calc_Aopt()`, 8, 11, 16, 20, 24  
`calc_copt`, 3  
`calc_copt()`, 8, 11, 16, 20, 24  
`calc_directional_derivatives`, 3  
`calc_directional_derivatives()`, 8, 15, 16  
`calc_Dopt`, 4  
`calc_Dopt()`, 8, 11, 15, 16, 20, 24  
`calc_Eopt`, 5  
`calc_Eopt()`, 11, 20  
`calc_eta_weights_maximin`, 6  
`calc_eta_weights_maximin()`, 9, 15, 16  
`calc_multi_directional_derivative`, 7  
`check_equivalence`, 8  
`check_equivalence()`, 15, 17  
`check_equivalence_maximin`, 8  
`check_equivalence_maximin()`, 8, 15, 16, 18  
`compute_design_S0`, 9  
`compute_design_S0()`, 5, 11, 20, 24  
`compute_maximin_design`, 10  
`compute_maximin_design()`, 6, 9, 12, 15–18, 22, 23  
`CVXR::psolve()`, 2, 3, 5, 7, 10, 11  
  
`exact_design_efficiency_maximin`, 12  
  
`graphics::plot()`, 17  
`gt_huang2020_cost`, 13, 14  
`gt_huang2020_f`, 13  
`gt_huang2020_lambda`, 14  
`gt_huang2020_pi`, 14, 14  
`gt_huang2020_regressor`, 15  
`gt_huang2020_regressor()`, 4  
  
`MASS::ginv()`, 4  
`maximin_design_workflow`, 15  
  
`plot_equivalence`, 17  
  
`plot_equivalence_maximin`, 18  
`plot_equivalence_maximin()`, 16  
  
`round_gt_design_budget`, 20  
`round_gt_design_budget()`, 12, 23, 24  
`round_gt_design_budget_maximin`, 21  
`round_gt_design_n_maximin`, 22  
`round_gt_design_subject_budget`, 23  
`rounding_budget_combinations`, 19  
`rounding_run_size_combinations`, 19  
  
`tibble::tibble()`, 21