Package: GRCFE (via r-universe)

October 30, 2026

Type Package

Title Generate Optimal Row-Column Factorial Experiments

Version 0.1.0

Maintainer Sukanta Dash <sukanta.iasri@gmail.com>

Description Provides tools for constructing row-column factorial experiment layouts for the estimation of main effects and two-factor interactions in factorial and fractional factorial experiments. The package implements generator-matrix based design construction methods motivated by 2fi-optimal row-column designs, where all main effects are estimable and as many two-factor interactions as possible are unconfounded; see Zhang, Pan and Shi (2025) <doi:10.1016/j.jspi.2024.106192>. It also includes theorem-based constructions, heuristic D-optimal search routines for unsupported or composite-level cases, utilities for building generator matrices, and diagnostic functions for evaluating aliasing and estimability properties of the generated designs.

License GPL (>= 3)

Depends R (>= 4.1.0)

Imports stats

Suggests roxygen2, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.3

Author Sukanta Dash [aut, cre], Amrit Kumar Paul [aut], Med Ram Verma [aut], Anurag Rawat [aut]

Repository https://sukantaiasri.r-universe.dev

Date/Publication 2026-04-28 20:44:33 UTC

RemoteUrl https://github.com/cran/GRCFE

RemoteRef HEAD

RemoteSha 1f269671fa5967c1890a10f435cc90c95561cc2b

Contents

build_G 2
eval_2fi_from_design 3
generate_2fi_optimal_rowcol 3
generate_2fi_rowcol 5
generate_row_column_design 6
search_2fi_optimal_G_k23 7

Index **9**

`build_G` *Build a generator matrix for theorem-based row-column designs*

Description

Dispatches to the implemented theorem-based generator-matrix construction for full designs and one-replicate fractions.

Usage

```
build_G(s, p, q, n)
```

Arguments

`s`	Integer. Number of levels. Prime values are supported by the theorem-based constructions.
`p`	Integer. Number of row blocking factors.
`q`	Integer. Number of column blocking factors.
`n`	Integer. Number of treatment factors. Must be either `p + q` or `p + q + 1`.

Value

Integer generator matrix with `p + q` rows and `n` columns.

See Also

`generate_row_column_design`

Examples

```
G <- build_G(s = 3, p = 1, q = 1, n = 2)
G
stopifnot(nrow(G) == 2, ncol(G) == 2)
```

eval_2fi_from_design *Evaluate unconfounded main effects and two-factor interactions*

Description

Checks whether treatment main effects are unconfounded with the remaining model terms and counts the number of unconfounded two-factor interactions in a row-column factorial design.

Usage

```
eval_2fi_from_design(design_df, s, tol = 1e-08)
```

Arguments

design_df	Data frame with columns Row, Col, and treatment factor columns.
s	Integer. Number of levels per treatment factor.
tol	Numeric tolerance for declaring cross-products to be zero.

Value

A list with components main_ok, a logical flag indicating whether all main effects are unconfounded, and unconf_2fi_num, the number of unconfounded two-factor interactions.

Examples

```
res <- generate_2fi_rowcol(s = 3, p = 1, q = 1, n = 2)
ev <- eval_2fi_from_design(res$design, s = 3)
ev
stopifnot(is.logical(ev$main_ok), length(ev$main_ok) == 1)
```

generate_2fi_optimal_rowcol *Generate a 2FI-optimal row-column factorial design*

Description

Constructs a row-column design for estimating main effects and two-factor interactions (2FIs). For prime s, theorem-based generator matrices are used when available. For composite s, and for the full two-level case, the function falls back to a D-optimality heuristic.

Usage

```
generate_2fi_optimal_rowcol(
  s,
  p,
  q,
  n = p + q,
  max_iter_nonprime = 5000,
  n_starts_nonprime = 5,
  verbose_nonprime = FALSE
)
```

Arguments

s	Integer. Number of levels for each treatment factor. Must be at least 2.
p	Integer. Number of row blocking factors; rows are indexed by s^p combinations.
q	Integer. Number of column blocking factors; columns are indexed by s^q combinations.
n	Integer. Number of treatment factors. Defaults to p + q.
max_iter_nonprime	Integer. Maximum coordinate-exchange iterations per heuristic start for non-prime or unsupported cases.
n_starts_nonprime	Integer. Number of random heuristic starts.
verbose_nonprime	Logical. If TRUE, reports heuristic progress using message(). The default is FALSE.

Details

The function supports p + q <= n. The heuristic path currently supports k = n - (p + q) in 0:3. The theorem-based path supports prime s when the corresponding generator-matrix construction is implemented.

Value

A list with components s, p, q, n, G, design, and construction metadata. For theorem-based designs, G is the generator matrix. For heuristic designs, G = NULL and D_opt_objective is included.

References

Zhang, Y., Pan, J. and Shi, L. (2025). Construction of 2fi-optimal row-column designs. *Journal of Statistical Planning and Inference*, 234, 106192. ISSN 0378-3758. doi:10.1016/j.jspi.2024.106192

See Also

generate_2fi_rowcol, build_G

Examples

```
res <- generate_2fi_optimal_rowcol(s = 3, p = 1, q = 1, n = 2)
head(res$design)
stopifnot(nrow(res$design) == 3^(1 + 1))
```

`generate_2fi_rowcol` *Generate a row-column design for main effects and two-factor interactions*

Description

Main user-facing wrapper for generating row-column factorial layouts. The function automatically chooses between theorem-based constructions, heuristic D-optimal search, and random generator-matrix search depending on `s`, `p`, `q`, `n`, and `mode`.

Usage

```
generate_2fi_rowcol(
  s,
  p,
  q,
  n,
  mode = c("auto", "theorem", "search"),
  max_iter_nonprime = 5000,
  n_starts_nonprime = 5,
  verbose_nonprime = FALSE,
  max_tries_search = 5000,
  N_max_runs_search = 10000,
  verbose_search = FALSE
)
```

Arguments

`s`	Integer. Number of levels for each treatment factor. Must be at least 2.
`p`	Integer. Number of row blocking factors.
`q`	Integer. Number of column blocking factors.
`n`	Integer. Number of treatment factors. Must satisfy `p + q <= n`.
`mode`	Character. One of `"auto"`, `"theorem"`, or `"search"`. `"auto"` uses theorem/heuristic construction for `k = 0, 1` and search for prime-level `k = 2, 3` cases.
`max_iter_nonprime`	Integer. Maximum heuristic iterations per start for composite-level or unsupported theorem cases.
`n_starts_nonprime`	Integer. Number of random heuristic starts.

`verbose_nonprime`
: Logical. If `TRUE`, reports heuristic progress using `message()`. The default is `FALSE`.

`max_tries_search`
: Integer. Maximum random generator matrices to try in search mode.

`N_max_runs_search`
: Integer. Maximum allowed number of runs for search.

`verbose_search`
: Logical. If `TRUE`, reports improvements found by the search routine using `message()`. The default is `FALSE` so the function is quiet in normal package use and during examples/tests.

Details

Let `k = n - (p + q)`. For non-prime `s`, this wrapper currently supports `k` in `0:3` through the heuristic path. For prime `s`, theorem mode is available for `k = 0` or `k = 1`; search mode is intended for `k = 2` or `k = 3`.

Value

A list containing the generated design and associated construction information. The `design` component is a data frame with columns `Row`, `Col`, and treatment factors `F1`, `F2`, ..., `Fn`.

See Also

`generate_2fi_optimal_rowcol`, `search_2fi_optimal_G_k23`, `eval_2fi_from_design`

Examples

```
# A small theorem-based row-column layout using the auto wrapper
res <- generate_2fi_rowcol(s = 3, p = 1, q = 1, n = 2, mode = "auto")
dim(res$design)
head(res$design)
stopifnot(is.data.frame(res$design))
```

---

generate_row_column_design

Generate the row-column treatment layout from a generator matrix

---

Description

Converts a generator matrix `G` into a row-column design by combining the row-generator and column-generator parts over `GF(s)`.

Usage

```
generate_row_column_design(G, s, p, q)
```

Arguments

G	Integer generator matrix with p + q rows.
s	Integer. Number of levels.
p	Integer. Number of row blocking factors.
q	Integer. Number of column blocking factors.

Value

A list containing s, p, q, n, G, design, Gc, and Gr. The design component is a data frame with columns Row, Col, and F1, ..., Fn.

Examples

```
G <- build_G(s = 3, p = 1, q = 1, n = 2)
res <- generate_row_column_design(G, s = 3, p = 1, q = 1)
head(res$design)
stopifnot(all(c("Row", "Col", "F1", "F2") %in% names(res$design)))
```

---

search_2fi_optimal_G_k23

Search for prime-level generator matrices for k = 2 or k = 3

Description

Performs a random search over generator matrices for prime-level designs when k = n − (p + q) is 2 or 3, retaining the design with the largest number of unconfounded two-factor interactions among candidates with unconfounded main effects.

Usage

```
search_2fi_optimal_G_k23(
  s,
  p,
  q,
  n,
  max_tries = 5000,
  N_max_runs = 10000,
  verbose = FALSE
)
```

Arguments

s	Prime integer number of levels. The search is tuned for s <= 7.
p	Integer. Number of row blocking factors.
q	Integer. Number of column blocking factors.
n	Integer. Number of treatment factors.
max_tries	Integer. Maximum number of random generator matrices to try.
N_max_runs	Integer. Maximum allowed number of runs s^(p + q).
verbose	Logical. If TRUE, reports improvements during search using message(). The default is FALSE.

Value

A list containing the best generator matrix, design, number of unconfounded two-factor interactions, and search metadata.

Examples

```
# The search is random, so a deliberately small number of tries may or may
# not find a feasible design on every platform. The example is therefore
# written to be executable and checkable without requiring stochastic success.
set.seed(1)
res <- try(
  search_2fi_optimal_G_k23(
    s = 5, p = 1, q = 1, n = 4,
    max_tries = 25, N_max_runs = 25, verbose = FALSE
  ),
  silent = TRUE
)
stopifnot(is.list(res) || inherits(res, "try-error"))
```

Index

build_G, 2, *4*

eval_2fi_from_design, 3, *6*

generate_2fi_optimal_rowcol, 3, *6*
generate_2fi_rowcol, *4*, 5
generate_row_column_design, *2*, 6

search_2fi_optimal_G_k23, *6*, 7