Package: pRepDOE (via r-universe)

November 14, 2026

Type Package

Title Construction of p-Rep Designs from Graph and Finite-Field Structures

Version 0.1.0

Maintainer Vinaykumar L.N. <vinaymandya123@gmail.com>

Description Constructs partially replicated (p-rep) designs in which every treatment occurs in every location, once or twice in Methods I and III and q or q + 1 times in Method II, with constant overall replication. Method I decomposes the complete graph into Hamiltonian paths, Method II uses cyclic blocks derived from a Hamiltonian cycle, and Method III uses mutually orthogonal Latin squares over a Galois field. Three treatment-replacement extensions (block enlargement, block splitting, and replacement followed by column-wise splitting) can be applied to Methods I and II. Each design is checked for connectedness and is returned with its parameters, pairwise concurrence distribution, canonical efficiency factor (CEF), average variance factor (AVF) and a randomized layout. Background references: Vinaykumar et al. (2026) <doi:10.1007/s12355-025-01684-1>; Vinaykumar et al. (2024) <doi:10.1080/03610926.2023.2185753>; Williams et al. (2011) <doi:10.1002/bimj.201000102>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Vinaykumar L.N. [aut, cre] (ORCID: <https://orcid.org/0009-0008-3411-3392>), Cini Varghese [aut] (ORCID: <https://orcid.org/0000-0001-9518-7178>), Mohd Harun

[aut] (ORCID: <https://orcid.org/0000-0002-7980-5711>),
Sayantani Karmakar [aut] (ORCID:
<https://orcid.org/0009-0002-2479-7803>), Vinayaka [aut]
(ORCID: <https://orcid.org/0000-0001-5004-0084>)

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

Date/Publication 2026-11-13 23:10:08 UTC

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

RemoteRef HEAD

RemoteSha 255132a5153c05255188673e526e504e38683dca

Contents

pRepDOE-package 2
design_summary 3
method1_prep 4
method2_prep 5
method3_prep 7

Index **9**

`pRepDOE-package` *Construction of p-rep Designs*

Description

Functions to construct p-rep designs by three methods (Hamiltonian path decomposition, cyclic blocks from a Hamiltonian cycle, and mutually orthogonal Latin squares) and to extend them by treatment replacement.

Details

- `method1_prep`: Method I, Hamiltonian path decomposition of the complete graph (even number of treatments).
- `method2_prep`: Method II, cyclic blocks taken from a Hamiltonian cycle.
- `method3_prep`: Method III, mutually orthogonal Latin squares over a Galois field (`v = s^2`).
- `design_summary`: one row per design, for building catalogues.

Methods I and II accept a treatment-replacement `case`: `1` (block enlargement), `2` (block splitting) and `3` (replacement followed by column-wise splitting). In Cases 2 and 3 the split blocks use a linked (twisted) matching, which keeps every extended design connected. Every design is verified to be connected before it is returned.

Author(s)

Maintainer: Vinaykumar L.N. <vinaymandya123@gmail.com> (ORCID)

Authors:

- Vinaykumar L.N. <vinaymandya123@gmail.com> (ORCID)
- Cini Varghese <cini2204@gmail.com> (ORCID)
- Mohd Harun <harun.agribhu@gmail.com> (ORCID)
- Sayantani Karmakar <sayantanikarmakar20@gmail.com> (ORCID)
- Vinayaka <vinayaka.b3vs@gmail.com> (ORCID)

`design_summary` *One-Row-per-Design Summary*

Description

Condenses the output of `method1_prep`, `method2_prep` or `method3_prep` into a data frame with one row per design, convenient for building catalogues.

Usage

```
design_summary(x)
```

Arguments

`x`	A list returned by one of the construction functions.

Value

A data frame with columns `design`, `method`, `case`, `f`, `g`, `v`, `b`, `r`, `k`, `L`, `CEF`, `AVF`, `connected`, `n_nonestimable_pairs`, `concurrence_levels` and `concurrence_counts`. The last two are comma-separated strings giving the distinct pairwise concurrence values and how many treatment pairs have each value.

Examples

```
design_summary(method1_prep(6))

## Several designs of one method (one row per feasible L)
design_summary(method2_prep(6, 5))

## Catalogue rows from different methods and cases
rows <- list(
  design_summary(method1_prep(6)),
  design_summary(method1_prep(6, case = 2, f = 2)),
  design_summary(method2_prep(10, 7)),
  design_summary(method3_prep(5))
)
catalogue <- do.call(rbind, rows)
catalogue[, c("method", "case", "v", "b", "r", "k", "L", "CEF", "AVF", "connected")]
```

`method1_prep` *Method I: Hamiltonian Path Decomposition*

Description

Constructs a p-rep design from a decomposition of the complete graph on `v` vertices into `v/2` edge-disjoint Hamiltonian paths, with optional treatment-replacement extensions (Cases 1 to 3).

Usage

```
method1_prep(v, case = 0, f = 1, g = 1, seed = NULL)
```

Arguments

`v`	Even integer ≥ 4: the number of treatments of the base design.
`case`	`0`: base design (`b = v(v-1)/2`, `k = 2`, `L = v/2`, `r = v-1`); `1`: replacement with block enlargement (needs `f >= 2`); `2`: replacement with block splitting (needs `f >= 2`); `3`: replacement with factor `f` followed by column-wise splitting with factor `g`.
`f`	Replacement factor (Cases 1 to 3).
`g`	Splitting factor (Case 3 only).
`seed`	Optional integer seed used only for the randomized layout. The global random-number stream of the caller is left unchanged.

Details

Each Hamiltonian path is assigned to one location, so every treatment occurs once or twice in every location. In Cases 2 and 3 the split blocks are formed with a linked (twisted) matching: the smallest-label treatment of each block takes the next member of its group, the others take member `m`. This links the copies of the design and makes it connected. The function stops with an error if a constructed design is not connected, or if replication is not as expected.

Value

A list with one element, `Design1`, which is a list with components

`design` data frame with columns `Location`, `Block`, `BaseBlock` and `Treatment1`, `Treatment2`, ...

`Design_parameters` list with `method`, `case`, `f`, `g`, `v`, `b`, `r`, `k`, `L`, `lambda_is_constant`, `pairwise_concurrence_values` and `concurrence_frequency`.

`randomized_layout` data frame with columns `Location`, `Block`, `Plot`, `Treatment`.

`location_replication` matrix (locations by treatments) of within-location replication.

`Efficiency_Factors` list with `CEF` (canonical efficiency factor) and `AVF` (average variance factor).

`Estimability` list with `rank`, `n_components`, `connected`, `n_pairs`, `n_estimable_pairs` and `n_nonestimable_pairs`.

See Also

method2_prep, method3_prep, design_summary

Examples

```
## Base design (Illustration 1): v = 6, b = 15, r = 5, k = 2, L = 3
x <- method1_prep(6)
d <- x$Design1
d$design                        # blocks, grouped by location
d$Design_parameters             # parameters and concurrence distribution
d$location_replication          # each treatment occurs once or twice per location
d$Efficiency_Factors            # CEF = 0.6, AVF = 0.6667
head(d$randomized_layout, 10)   # randomized field layout

## Reproducible randomization (the caller's random numbers are untouched)
method1_prep(6, seed = 123)$Design1$randomized_layout[1:6, ]

## Case 1: replacement with block enlargement (v = 4, f = 2 gives v' = 8)
c1 <- method1_prep(4, case = 1, f = 2)$Design1
c1$Design_parameters[c("v", "b", "r", "k", "L")]
c1$Efficiency_Factors           # CEF = 0.8235, AVF = 0.8095

## Case 2: replacement with block splitting (linked matching, connected)
c2 <- method1_prep(4, case = 2, f = 2)$Design1
c2$design[c2$design$BaseBlock == 1, ]   # base block (1, 2) -> (1, 4) and (2, 3)
c2$Estimability$connected               # TRUE
c2$Efficiency_Factors                   # CEF = 0.4828, AVF = 1.381

## Case 3: replacement (f) followed by column-wise splitting (g)
c3 <- method1_prep(4, case = 3, f = 2, g = 2, seed = 123)$Design1
c3$Design_parameters$pairwise_concurrence_values
c3$Efficiency_Factors                   # CEF = 0.6634, AVF = 1.0048

## One-row summary of several designs
design_summary(method1_prep(6))
```

---

method2_prep *Method II: Cyclic Blocks from a Hamiltonian Cycle*

Description

Constructs p-rep designs from the v cyclic blocks $B_i = \{i, i+1, \ldots, i+k-1\}$ (labels modulo v), one design for each feasible number of locations, with optional treatment-replacement extensions (Cases 1 to 3).

Usage

```
method2_prep(v, k, case = 0, f = 1, g = 1, L = NULL, seed = NULL)
```

Arguments

v	Integer ≥ 4: number of treatments of the base design.
k	Block size, 3 <= k < v.
case, f, g	As in method1_prep.
L	Optional number of locations. If NULL, one design is returned for every feasible L. The feasible values are the divisors L of v with 2 <= L <= k that do not divide k.
seed	Optional integer seed for the randomized layout (the caller's random-number stream is left unchanged).

Details

Block B_i is placed in location ((i - 1) mod L) + 1. Within a location each treatment occurs q or q + 1 times, where q = floor(k / L); overall replication is r = k. Pairwise concurrence is not necessarily constant. Cases 2 and 3 use the linked matching described in method1_prep. The function stops with an error if a design is not connected.

Value

A list Design1, Design2, ..., one element per feasible (or requested) L. Each element is a list with components

design data frame with columns Location, Block, BaseBlock and Treatment1, Treatment2, ...

Design_parameters list with method, case, f, g, v, b, r, k, L, lambda_is_constant, pairwise_concurrence_values and concurrence_frequency.

randomized_layout data frame with columns Location, Block, Plot, Treatment.

location_replication matrix (locations by treatments) of within-location replication.

Efficiency_Factors list with CEF (canonical efficiency factor) and AVF (average variance factor).

Estimability list with rank, n_components, connected, n_pairs, n_estimable_pairs and n_nonestimable_pairs.

See Also

method1_prep, method3_prep, design_summary

Examples

```
## Base designs (Illustration 2): v = 6, k = 5 has two feasible L
y <- method2_prep(6, 5)
names(y)                                  # Design1 (L = 2), Design2 (L = 3)
y$Design1$design                          # blocks grouped by location
y$Design1$location_replication            # each treatment 2 or 3 times per location
y$Design2$Design_parameters               # every pair occurs together in 4 blocks
y$Design1$Efficiency_Factors              # CEF = 0.96, AVF = 0.4167

## Request one number of locations only
method2_prep(6, 5, L = 3)
```

```
## Case 1: replacement with block enlargement
c1 <- method2_prep(10, 7, case = 1, f = 2)$Design1
c1$Efficiency_Factors                    # CEF = 0.9753, AVF = 0.2929

## Case 2: replacement with block splitting (connected)
c2 <- method2_prep(6, 5, case = 2, f = 2)$Design1
c2$Estimability$connected                # TRUE
c2$Efficiency_Factors                    # CEF = 0.5198, AVF = 0.7695

## Case 3: replacement followed by column-wise splitting
c3 <- method2_prep(10, 7, case = 3, f = 2, g = 2, seed = 1)$Design1
c3$Efficiency_Factors                    # CEF = 0.7323, AVF = 0.3902

## One row per design (here L = 2 and L = 4)
design_summary(method2_prep(8, 5))
## Invalid input: no feasible number of locations
try(method2_prep(8, 4))
```

`method3_prep` *Method III: Mutually Orthogonal Latin Squares*

Description

Constructs a p-rep design on `v = s^2` treatments from the `s + 1` parallel classes given by the rows, the columns and the `s - 1` mutually orthogonal Latin squares $L_m(i, j) = i + mj$ over the Galois field $GF(s)$.

Usage

```
method3_prep(s, augmented_class = NULL, seed = NULL)
```

Arguments

`s`	A prime or prime power, $s \geq 3$.
`augmented_class`	Integer from 1 to `s + 1`: the parallel class whose `s` blocks are added, one per location. The other `s` classes are assigned one per location. Default `s + 1`.
`seed`	Optional integer seed for the randomized layout (the caller's random-number stream is left unchanged).

Details

Every treatment occurs once in every location and once more in the block of the augmented class, so within-location replication is one or two and overall replication is `r = s + 1`. The base design has `b = s(s+1)`, `k = L = s` and `lambda = 1`; the value of lambda is reported from the computed concurrence distribution rather than assumed. For non-prime `s` finite-field arithmetic is used.

Value

A list with one element, `Design1`, structured as in `method1_prep`; its `Design_parameters` also contain `s` and `blocks_per_location`.

See Also

`method1_prep`, `method2_prep`, `design_summary`

Examples

```
## Illustration 3: s = 3 gives v = 9, b = 12, r = 4, k = 3, L = 3
z <- method3_prep(3)
z$Design1$design                          # four blocks per location
z$Design1$Design_parameters               # lambda = 1
z$Design1$location_replication            # each treatment once or twice per location
z$Design1$Efficiency_Factors              # CEF = 0.75, AVF = 0.6667

## Use a different parallel class for the extra block in each location
method3_prep(3, augmented_class = 2)$Design1$design

## Prime power s: GF(4) arithmetic
g4 <- method3_prep(4)$Design1
g4$Design_parameters$pairwise_concurrence_values   # 1
g4$Efficiency_Factors                              # CEF = 4/5, AVF = 0.5

## GF(8), with a reproducible randomized layout
method3_prep(8, seed = 123)$Design1$Efficiency_Factors   # CEF = 8/9

## Invalid input: 6 is not a prime power
try(method3_prep(6))
```

Index

∗ **design**
pRepDOE-package, 2

design_summary, *2*, 3, *5*, *6*, *8*

method1_prep, *2*, *3*, 4, *6*, *8*
method2_prep, *2*, *3*, 5, *5*, *8*
method3_prep, *2*, *3*, *5*, *6*, 7

pRepDOE *(*pRepDOE-package*)*, 2
pRepDOE-package, 2