Package ‘doofa’

May 14, 2024

Version 1.0

Date 2024-05-13

Title Designs for Order-of-Addition Experiments

Author Baidya Nath Mandal [aut, cre],
Rajender Parsad [aut],
Sukanta Dash [aut]

Maintainer Baidya Nath Mandal <mandal.stat@gmail.com>

Depends R (>= 4.4.0)

Imports lpSolve, combinat

Description A facility to generate efficient designs for order-of-additions experiments under pairwise-order model, see Dennis K. J. Lin and Jiayu Peng (2019).``Order-of-addition experiments: A review and some new thoughts". Quality Engineering, 31:1, 49-59, <doi:10.1080/08982112.2018.1548021>. It also provides a facility to generate component orthogonal arrays under component position model, see Jian-Feng Yang, Fasheng Sun & Hongquan Xu (2020): ``A Component Position Model, Analysis and Design for Order-of-Addition Experiments". Technometrics, <doi:10.1080/00401706.2020.1764394>.

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2024-05-14 08:23:19 UTC

R topics documented:

`bin` *binary representation of x (an element from 1 to m) with m components*

Description

binary representation of x (an element from 1 to m) with m components

Usage

```
bin(x, m)
```

Arguments

`x`	a positive integer less than or equal to m
`m`	number of components, a positive integer

Value

a vector with elements 1 and 0

Examples

```
bin(x = 2, m = 4);
```

`coa` *construct a component orthogonal array with m components*

Description

construct a component orthogonal array with m components such that each pair of columns contains each (i != j) combinations lambda times

Usage

```
coa(m, lambda, ntrial)
```

Arguments

`m`	a positive integer, currently supports less than 8
`lambda`	a positive integer, usually 1
`ntrial`	a positive integer, default is 10

Value

a component orthogonal array with m components

Examples

```
coa(m = 4, lambda = 1, ntrial = 10);
```

`cycle` *cycle elements of a vector by one element to right*

Description

cycle elements of a vector by one element to right

Usage

```
cycle(x)
```

Arguments

`x`	a vector

Value

cycled vector

Examples

```
cycle(c(1, 2, 3));
```

doofa.pwo *construct a design for order-of-addition experiment under pwo model with n runs and m components*

Description

construct a design for order-of-addition experiment under pwo model with n runs and m components

Usage

```
doofa.pwo(n, m)
```

Arguments

n	a positive integer, preferably less than 100
m	a positive integer, currently supports less than 8

Value

a design for order-of-addition experiment under pwo model with n runs and m components

Examples

```
doofa.pwo(5,3);
```

gen.design2 *Repeat the process of design generation using doofa.pwo several times and return the best design*

Description

Repeat the process of design generation using doofa.pwo several times and return the best design

Usage

```
gen.design2(n, m, num.repeat = 10)
```

Arguments

n	number of runs, a positive integer
m	number of components, a positive integer
num.repeat	number of repeats, a positive integer

Value

a design with D-efficiency

Examples

```
gen.design2(n = 5, m = 3, num.repeat = 10);
```

`initial.design` *create an initial design of o-of-a with n rows and m columns*

Description

create an initial design of o-of-a with n rows and m columns

Usage

```
initial.design(n, m)
```

Arguments

`n`	a positive integer
`m`	a positive integer

Value

a matrix with n rows and m columns

Examples

```
initial.design(n = 6, m = 3);
```

`one` *create a matrix of 1s with t rows*

Description

create a matrix of 1s with t rows

Usage

```
one(t)
```

Arguments

`t`	a positive integer

Value

a matrix of 1s with t rows

Examples

```
one(3);
```

`pwo` *create PWO ordering of a given run of a design*

Description

create PWO ordering of the given run

Usage

```
pwo(x)
```

Arguments

`x`	a numeric vector containing elements 1 to m in some order

Value

PWO ordering of the given run

Examples

```
row = c(3,1,2)
pwo(row);
```

`revbin` *reverse of bin function i.e., returns which elements of a binary vector is 1*

Description

reverse of bin function i.e., returns which elements of a binary vector is 1

Usage

```
revbin(x)
```

Arguments

`x`	a vector with 0 and 1s such that there is only 1

Value

a postive integer m

Examples

```
revbin(c(0,1,0,0));
```

`shuffle` *shuffle elements of a randomly chosen row of x matrix*

Description

shuffle elements of a randomly chosen row of x matrix

Usage

```
shuffle(x)
```

Arguments

`x`	a matrix

Value

a matrix with shuffled elements of a row

Examples

```
x = matrix(c(3,1,2, 1,2,3,1,3,2,2,1,3),ncol = 3, byrow = TRUE)
shuffle(x);
```

`swap` *swap elements at i and i+1 of a vector*

Description

swap elements at i and i+1 of a vector

Usage

```
swap(x, i)
```

Arguments

`x`	a vector
`i`	a positive integer, less than length of x

Value

a vector with swapped elements

Examples

```
swap(c(1,2,3),2);
```

vbin *vectorized bin function*

Description

vectorized bin function

Usage

```
vbin(x)
```

Arguments

x	a vector of length m with positive integers less than or equal to m

Value

a binary matrix

Examples

```
vbin(c(3,1,2));
```

vrevbin *vectorized revbin function*

Description

vectorized revbin function

Usage

```
vrevbin(x, m)
```

Arguments

x	a binary vector of length nm, such that each length of m has only one 1 and rest as 0
m	a positive integer

Value

a vector of n positive integers

Examples

```
vrevbin(x=c(0,0,1,0,1,0), m = 3);
```

Index