Package ‘conf.design’

October 12, 2022

Type Package
Title Construction of factorial designs
Version 2.0.0
Date 2013-02-22
Suggests stats, utils
Author Bill Venables
Maintainer Bill Venables <Bill.Venables@gmail.com>
Description This small library contains a series of simple tools for constructing and manipulating confounded and fractional factorial designs.
License GPL-2
NeedsCompilation no
Repository CRAN
Date/Publication 2013-02-23 15:18:29

R topics documented:

conf.design-package .................................................. 2
conf.design .............................................................. 4
conf.set ................................................................. 6
direct.sum ............................................................. 7
factorize ................................................................. 9
join ....................................................................... 10
primes ................................................................. 11
rjoin ................................................................. 12

Index 14
Description

Construct confounded designs with specific contrasts confounded with blocks. The package only directly handles the $p^k$ case, that is, all treatment factors having the same (prime) number of levels. Some simple facilities are provided for combining component designs into larger ones, thus providing some facilities for generating interesting designs for the more general case. Some fractional replication is also possible with the tools provided.

Details

Package: conf.design
Type: Package
Version: 2.0.0
Date: 2013-02-23
License: GPL-2

The key functions are conf.design, rjoin and direct.sum. The help information for these functions contain some fairly detailed examples.

Some ancillary functions may be of independent interest, for example primes for generating prime numbers and factorize, which has an experimental design application, but a default method can be used to factorize (not too large) integers into prime factors.

Author(s)

Bill Venables
Maintainer: Bill Venables <Bill.Venables@gmail.com>

References


See Also

The CRAN task view on Design of Experiments

Examples

### Generate a half replicate of a 2^3 x 3^2 experiment. The factors are to be A, B, C, D, E. The fractional relation is to be I = ABC and the DE effect is to be confounded with blocks.
### First construct the 2^3 design, confounded in two blocks:
```r
d1 <- conf.design(c(A = 1, B = 1, C = 1), p=2)
```

### Next the 3^2 design, with DE confounded in blocks:
```r
d2 <- conf.design(c(D = 1, E = 1), p=3)
```

### Now extract the principal block from the 2^3 design and form the direct sum with the 3^2 design
```r
dsn <- direct.sum(subset(d1, Blocks == "0"), d2)
```

```r
head(dsn)
##    Blocks A B C Blocksa D E
## 1     0 0 0     0   0 0
## 2     0 0 0     2   1
## 3     0 0 0     1  2
## 4     0 0 0  10   0
## 5     0 0 0  12 1 2
## 6     0 0 0  1 2 2
```

### Combine the two "Blocks" factors into a single block factor:
```r
dsn <- within(dsn, {
    Blocks <- join(Blocks, Blocksa)
    Blocksa <- NULL
})
```

### Now to do some checks.
```r
as.matrix(replications(~ .^2, dsn))
```

```r
## Blocks  12
##    A  18
##    B  18
##    C  18
##    D  12
##    E  12
## Blocks:A  6
## Blocks:B  6
## Blocks:C  6
## Blocks:D  4
## Blocks:E  4
##    A:B  9
##    A:C  9
##    A:D  6
##    A:E  6
##    B:C  9
##    B:D  6
##    B:E  6
##    C:D  6
##    C:E  6
##    D:E  4
```

### We can check the confounding by analysing some dummy data:
```r
dsn$y <- rnorm(nrow(dsn))
dummyAov <- aov(y ~ A*B*C*D*E + Error(Blocks), data=dsn)
summary(dummyAov)
```
### Error: Blocks
### Df Sum Sq Mean Sq
### D:E 2 8.915 4.458

### Error: Within
### Df Sum Sq Mean Sq
### A 1 2.077 2.077
### B 1 1.111 1.111
### C 1 3.311 3.311
### D 2 1.929 0.964
### E 2 0.848 0.424
### A:D 2 3.421 1.711
### B:D 2 3.231 1.615
### C:D 2 2.484 1.242
### A:E 2 0.214 0.107
### B:E 2 0.006 0.003
### C:E 2 0.349 0.174
### D:E 2 1.442 0.721
### A:D:E 4 2.560 0.640
### B:D:E 4 4.454 1.114
### C:D:E 4 7.942 1.986

### Two of the D:E degrees of freedom are confounded with Blocks, as desired.

**conf.design**

*Construct symmetric confounded factorial designs.*

**Description**

Construct designs with specified treatment contrasts confounded with blocks. All treatment factors must have the same (prime) number of levels.

**Usage**

```r
cnf.design(G, p, block.name = "Blocks", treatment.names = NULL)
```

**Arguments**

- **G**
  - Matrix whose rows define the contrasts to be confounded.
  - The number of columns of `G` defines the number of factors.
- **p**
  - The common number of levels for each factor. Must be a prime number.
- **block.name**
  - Name to be given to the factor defining the blocks of the design.
- **treatment.names**
  - Name to be given to the treatment factors of the design. If `NULL` and if `G` has a `dimnames` attribute, then `dimnames[[2]]` is the default, otherwise `T1, T2, ...`
Details

For example in a $3^4$ experiment with $AB^2C$ and $BCD$ confounded with blocks (together with their generalized interactions), the matrix $G$ could be given by

```r
rbind(c(A = 1, B = 2, C = 1, D = 0), c(A = 0, B = 1, C = 1, D = 1))
```

For this example, $p = 3$

Having column names for the $G$ matrix implicitly supplies the treatment factor names.

For a single replicate of treatments, blocks are calculated using the confounded contrasts in the standard textbook way. The method is related to that of Collings (1989).

Value

A design with a Blocks factor defining the blocks and treatment factors defining the way treatments are allocated to each plot. (Not in randomised order!)

Side Effects

None.

References


See Also

`conf.set`, `direct.sum`

Examples

```r
### Generate a 3^4 factorial with $A B^2 C$ and $B C D$ confounded with blocks.
###
d34 <- conf.design(rbind(c(A = 1, B = 2, C = 1, D = 0),
c(A = 0, B = 1, C = 1, D = 1)), p = 3)

head(d34)

### Blocks A B C D
### 1 0 0 0 0 0
### 2 0 0 1 2 1 0
### 3 0 0 2 1 2 0
### 4 0 0 2 2 0 1
### 5 0 0 1 1 1
### 6 0 0 1 0 2 1

as.matrix(replications(~ .^2, d34))

### [,1]
### Blocks 9
### A 27
### B 27
### C 27
### D 27
### Blocks:A 3
```
### conf.set

Find all confounded effects

#### Description

Find minimal complete sets of confounded effects from a defining set for symmetric confounded factorial designs. Useful for checking if a low order interaction will be unintentionally confounded with blocks. As in the usual convention, only effects whose leading factor has an index of one are listed.

All factors must have the same (prime) number of levels.

#### Usage

```r
conf.set(G, p)
```

#### Arguments

- **G**: Matrix whose rows define the effects to be confounded with blocks, in the same way as for `conf.design`.
- **p**: Number of levels for each factor. Must be a prime number.

#### Details

The function constructs all linear functions of the rows of G (over GF(p)), and removes those rows whose leading non-zero component is not one.

#### Value

A matrix like G with a minimal set of confounded with blocks defined in the rows.

#### Side Effects

None

#### See Also

conf.design
Examples

#### If A B^2 C and B C D are confounded with blocks, then so are A C^2 D
#### and A B D^2.

G <- rbind(c(A = 1, B = 2, C = 1, D = 0),
           c(A = 0, B = 1, C = 1, D = 1))
conf.set(G, 3)
#### A B C D
#### [1,] 1 2 1 0
#### [2,] 0 1 1 1
#### [3,] 1 0 2 1
#### [4,] 1 1 0 2

#### Only three-factor interactions are confounded, so the design is
#### presumably useful.

as.matrix(replications(~ .^2, conf.design(G, 3)))
#### [,1]
#### Blocks 9
#### A 27
#### B 27
#### C 27
#### D 27
#### Blocks:A 3
#### Blocks:B 3
#### Blocks:C 3
#### Blocks:D 3
#### A:B 9
#### A:C 9
#### A:D 9
#### B:C 9
#### B:D 9
#### C:D 9

---

direct.sum

Form the direct sum of designs.

---

direct.sum

Form the direct sum of designs.

Description

Constructs the direct sum of two or more designs. Each plot of one design is matched with every plot of the other. (This might also be called the Cartesian product of two designs).

Usage

direct.sum(D1, ..., tiebreak=letters)

Arguments

D1 First component design.
Additional component designs, if any.

tiebreak Series of characters or digits to be used for breaking ties (or repeats) in the variable names in the component designs. Augmented if necessary.

Details

Each plot of one design is matched with every plot of the next, (if any), and so on recursively.

Value

The direct sum of all component designs.

Side Effects

None.

See Also

cnf.design

Examples

```r
### Generate a half replicate of a 2^3 x 3^2 experiment. The factors are to be A, B, C, D, E. The fractional relation is to be I = ABC and the DE effect is to be confounded with blocks.
### First construct the 2^3 design, confounded in two blocks:
d1 <- conf.design(cbind(A = 1, B = 1, C = 1), p = 2)

### Next the 3^2 design, with DE partially confounded in blocks:
d2 <- conf.design(cbind(D = 1, E = 1), p = 3)

### Now extract the principal block from the 2^3 design and form the direct sum with the 3^2 design
dsn <- direct.sum(subset(d1, Blocks == "0"), d2)

### combine block factors into one
dsn <- within(dsn, {
  Blocks <- join(Blocks, Blocksa)
  Blocksa <- NULL
})
head(dsn)
```
factorize S3 generic function and methods for factorization.

Description

The default method factorizes positive numeric integer arguments, returning a vector of prime factors. The factor method can be used to generate pseudo-factors. It accepts a factor, f, as principal argument and returns a data frame with factors fa, fb, ... each with a prime number of levels such that f is model equivalent to join(fa, fb, ...).

Usage

```r
## Default S3 method:
factorize(x, divisors = primes(max(x)), ...)  
## S3 method for class 'factor'
factorize(x, name = deparse(substitute(x)), extension = letters, ...)
```

Arguments

- **x**: Principal argument.  
The default method factorizes (smallish) positive integers;  
The factor method generates prime pseudo-factors from a factor with a composite number of levels (as required for partial confounding).

- **divisors**: Candidate prime divisors for all numbers to be factorized.

- **name**: Stem of the name to be given to component pseudo-factors.

- **extension**: Distinguishing strings to be added to the stem to nominate the pseudo-factors.

- **...**: Additional arguments, if any. (Presently ignored.)

Details

Primarily intended to split a factor with a non-prime number of levels into a series of pseudo-factors, each with a prime number of levels and which jointly define the same classes as the factor itself.

The main reason to do this would be to confound one or more of the pseudo-factors, or their interactions, with blocks using constructions that only apply for prime numbers of levels. In this way the experiment can be made smaller, at the cost of some treatment contrasts being confounded with blocks.

The default method factorizes integers by a clumsy, though effective enough way for small integers. The function is vectorized in the sense that if a vector of integers to factorize is specified, the object returned is a list of numeric vectors, giving the prime divisors (including repeats) of the given integers respectively.

As with any method of factorizing integers, it may become very slow if the prime factors are large.
join

Amalgamate two or more factors.

Description

Joins two or more factors together into a single composite factor defining the subclasses. In a model formula join(f1, f2, f3) is equivalent to f1:f2:f3.
Usage

join(...)

Arguments

... Two or more factors or numeric vectors, or lists containing these kinds of component.

Details

Similar in effect to paste, which it uses.

Value

A single composite factor with levels made up of the distinct combinations of levels or values of the arguments which occur.

Side Effects

None.

See Also

:.paste, rjoin, direct.sum

Examples

within(data.frame(f = gl(2, 3)), {
  g <- gl(3,2,length(f))
  fg <- join(f, g)
})
### f fg g
### 1 1 1:1 1
### 2 1 1:1 1
### 3 1 1:2 2
### 4 2 2:2 2
### 5 2 2:3 3
### 6 2 2:3 3

primes

Prime numbers

Description

Generate a table of prime numbers.

Usage

primes(n)
Arguments
n
A positive integer value, or vector or such values.

Details
Uses an elementary sieve method.

Value
A vector of all prime numbers less than the \text{max}(n).
NB: 1 is not a prime number!

Side Effects
None

See Also
factorize

Examples
\texttt{primes(1:50)}

\begin{verbatim}
[1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47
\end{verbatim}

\begin{verbatim}
\texttt{rjoin}  \texttt{Concatenate designs by rows.}
\end{verbatim}

Description
Combine two or more designs with the same names into a single design by row concatenation.

Usage
\texttt{rjoin(..., part.name="Part")}

Arguments
\ldots
Two or more designs with identical component names.

part.name
Name for an additional factor to identify the original components in the result.

Details
Almost the same as \texttt{rbind}, but an additional factor in the result separates the original components.
rjoin

Value

A single design with the arguments stacked above each other (in a similar manner to `rbind`), together with an additional factor whose levels identify the original component designs, or parts.

Side Effects

None.

See Also

`rbind`

Examples

```r
### A two replicate partially confounded factorial design.
d1 <- conf.design(c(A = 1, B = 1, C = 1), 2)
d2 <- conf.design(c(A = 0, B = 1, C = 1), 2)
dsn <- within(rjoin(d1, d2), {
  Blocks <- join(Part, Blocks)
  Part <- NULL
})
as.matrix(replications(~ .^2, dsn))
```

```r
### [,1]
### Blocks 4
### A 8
### B 8
### C 8
### Blocks:A 2
### Blocks:B 2
### Blocks:C 2
### A:B 4
### A:C 4
### B:C 4
```
Index

* design
  conf.design, 4
  conf.design-package, 2
  conf.set, 6
  direct.sum, 7
  factorize, 9
  join, 10
  primes, 11
  rjoin, 12
  ::, 11

conf.design, 4, 6, 8
conf.design-package, 2
conf.set, 5, 6

direct.sum, 5, 7, 11

factorize, 9, 12

join, 10

paste, 11
primes, 11

rbind, 13
rjoin, 11, 12