

Package ‘conf.design’

June 17, 2009

Type Package

Title Construction of factorial designs

Version 1.0

Date 2009-06-13

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 file LICENSE

LazyLoad yes

Repository CRAN

Date/Publication 2009-06-17 13:04:42

R topics documented:

conf.design	2
conf.set	3
direct.sum	4
factorize	5
join	6
primes	7
rjoin	8
Index	9

 conf.design

Construct symmetric confounded factorial designs.

Description

Construct designs with specified treatment contrasts confounded with blocks.

Usage

```
conf.design(G, p, block.name="Blocks", treatment.names=)
```

Arguments

<code>G</code>	Matrix whose rows define the contrasts to be confounded. For example in a 3^4 experiment with <code>A B^2 C</code> and <code>B C D</code> confounded with blocks (together with their generalized interactions), the matrix <code>G</code> would be <code>rbind(c(1,2,1,0), c(0,1,1,1))</code> . The number of columns of <code>G</code> is the number of factors.
<code>p</code>	The common number of levels for each factor. Must be a prime number.
<code>block.name</code>	Name to be given to the factor defining the blocks of the design.
<code>treatment.names</code>	Name to be given to the treatment factors of the design. If <code>G</code> has a <code>dimnames</code> attribute, then <code>dimnames[[2]]</code> is the default, otherwise <code>T1, T2,</code>

Details

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 random order.

Side Effects

None.

References

Collings, B. J. (1989) Quick confounding. *Technometrics*, v31, pp107-110.

See Also

`conf.set`, `direct.sum`, `fac.design`, `fractionate`

Examples

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

`conf.set`*Find 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.

Usage

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

Arguments

G	Matrix whose rows define the effects to be confounded with blocks, in the same way as for <code>conf.design()</code> .
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

```
G <- rbind(c(1,2,1,0), c(0,1,1,1))
dimnames(G) <- list(NULL, LETTERS[1:4])
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
# 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. Only three-factor interactions are confounded, so the
# design is presumably useful.
```

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, D2, ..., tiebreak=letters)
```

Arguments

D1	First component design.
D2	Second 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.

Details

Each plot of one design is matched with every plot of the other, and so on recursively.

Value

The direct sum of all component designs.

Side Effects

None.

See Also

conf.design, fac.design, fractionate

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:
d1 <- conf.design(c(1,1,1), p=2, treatment.names=LETTERS[1:3])

# Next the 3^2 design, with DE confounded in blocks:
d2 <- conf.design(c(1,1), p=3, treatment.names=LETTERS[4:5])

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

factorize

Generic function.

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 design with factors *fa*, *fb*, ... each with a prime number of levels such that *f* is model equivalent to *join(fa, fb, ...)*.

Usage

```
factorize(x, ...)
```

Arguments

<i>x</i>	Principal argument. At this stage, it may be a numeric vector to elicit the default method, or a factor to elicit the factor method.
<i>...</i>	Additional arguments, if any.

Details

Factorizes by a clumsy though effective enough way for small integers. May become very slow if some prime factors are large. For the factor method it generates pseudo factors in the usual way.

Value

A vector of (numeric) factors for numeric arguments, or a design with (S-PLUS) factors with prime numbers of levels for factor arguments.

Side Effects

None.

See Also

conf.design, join

Examples

```
factorize(18)
# [1] 2 3 3
f <- factor(rep(0,5), rep(6,5))
fd <- factorize(f)
```

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 objects of mode list 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

```
d1 <- conf.design(c(1,1,1), 2, treatment.names=LETTERS[1:3])
d2 <- conf.design(c(0,1,1), 2, treatment.names=LETTERS[1:3])
fd1d2 <- join(d1,d2)
```

`primes`*Prime numbers*

Description

Generate a table of prime numbers.

Usage

```
primes(n)
```

Arguments

`n` Positive integer value.

Details

Uses an elementary sieve method, not suitable for very large `n`. This function is no longer used as part of the design library and is included for curiosity value only.

Value

A vector of all prime numbers less than the argument. 1 is not a prime.

Side Effects

None

See Also

`factorize`

Examples

```
primes(50)
# [1]  2  3  5  7 11 13 17 19 23 29 31 37 41 43 47
```

rjoin

Concatenate designs by rows.

Description

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

Usage

```
rjoin(..., part.name="Part")
```

Arguments

`...` 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 `rbind()`, but an additional factor in the result separates the original components.

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

`conf.design`, `join`, `direct.sum`

Examples

```
# A two replicate partially confounded factorial design.  
d1 <- conf.design(c(1,1,1), 2, treatment.names=LETTERS[1:3])  
d2 <- conf.design(c(0,1,1), 2, treatment.names=LETTERS[1:3])  
dsn <- rjoin(d1, d2)
```

Index

*Topic **design**

- conf.design, 1
- conf.set, 3
- direct.sum, 4
- factorize, 5
- join, 6
- primes, 7
- rjoin, 8

- conf.design, 1
- conf.set, 3

- direct.sum, 4

- factorize, 5

- join, 6

- primes, 7

- rjoin, 8