Package 'planor'

July 10, 2014

Title Generation of regular factorial designs

Date 2014-07-11

Version 0.2-0

Author Hervé Monod, Annie Bouvier, André Kobilinsky

Description planor is dedicated to the automatic generation of regular factorial designs, including fractional designs, orthogonal block designs, row-column designs and split-plots.

Maintainer Annie Bouvier <Annie.Bouvier@jouy.inra.fr>

License GPL (>= 2)

Depends methods, conf.design

Suggests

Encoding UTF-8

URL http://www.inra.fr/miaj/public/logiciels/planor

Collate zzz.R debug.R generic.R designfactors.R keymatrix.R keyring.R listofkeyrings.R planor.R basep.R randomize.R designkey.R listofdesignkeys.R planordesign.R makedesignkey.R

LazyLoad yes

BuildVignettes no

R topics documented:

`planor-package` *Generation of regular factorial designs, big-package-free version of planor*

Description

A package dedicated to the automatic generation of regular factorial designs, including fractional designs, orthogonal block designs, row-column designs and split-plots.

Details

The user describes the factors to be controlled in the experiment and the anova model to be used when the results will be analysed. He or she also specifies the size of the design, that is, the number of available experimental units. Then **planor** looks for a design satisfying these specifications and possibly randomizes it. The core of the algorithm is the search for the key matrix, an integer matrix which determines the aliasing in the resulting factorial design.

The user may use the function `regular.design` where all these steps are integrated, and transparent by default. Alternatively, the steps can be decomposed by using successively the functions `planor.factors`, `planor.model`, `planor.designkey` and `planor.design`. For the expert user, the function `planor.designkey` can give several key matrix solutions. Alias and summary methods allow to study and compare these solutions, in order to select the most appropriate one for the final design.

Note

- Option `planor.max.print` can be set to limit the amount of the matrices that are printed, to `planor.max.print` rows and columns. Default: 20.

Author(s)

Hervé Monod, Annie Bouvier, André Kobilinsky (Applied Mathematics and Informatics Unit, INRA UR 341 - Jouy-en-Josas, France. URL: `http://www.jouy.inra.fr/mia_eng/`)

References

see `citation(planor)`

Examples

```
# DESIGN SPECIFICATIONS
# Treatments: four 3-level factors A, B, C, D
# Units: 27 in 3 blocks of size 9
# Non-negligible factorial terms:
#   block + A + B + C + D + A:B + A:C + A:D + B:C + B:D + C:D
# Factorial terms to estimate:
#   A + B + C + D
# 1. DIRECT GENERATION, USING 'regular.design'
mydesign <- regular.design(factors=c("block", LETTERS[1:4]),
  nlevels=rep(3,5), model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
  nunits=3^3, randomize=~block/UNITS)
print(mydesign)
# DUMMY ANALYSIS
# Here we omit two-factor interactions from the model, so they are
# confounded with the residuals (but not with ABCD main effects)
set.seed(123)
mydesigndata=mydesign@design
mydesigndata$Y <- runif(27)
mydesign.aov <- aov(Y ~ block + A + B + C + D, data=mydesigndata)
summary(mydesign.aov)
# 2. STEP-BY-STEP GENERATION, USING 'planor.designkey'
F0 <- planor.factors(factors=c( "block", LETTERS[1:4]), nlevels=rep(3,5),
  block=~block)
M0 <- planor.model(model=~block+(A+B+C+D)^2, estimate=~A+B+C+D)
K0 <- planor.designkey(factors=F0, model=M0, nunits=3^3, max.sol=2)
summary(K0)
mydesign.S4 <- planor.design(key=K0, select=2)
```

`alias-methods` *Methods for function* `alias` *in package* **planor***: summarize the design properties*

Description

Summarize the design properties of an object containing key matrices. Display the design keys matrix(ces) and the factorial effects confounded with the mean.

Usage

```
## S4 method for signature 'designkey'
alias(object, model, ...)

## S4 method for signature 'keymatrix'
alias(object, model, fact, block, ...)

## S4 method for signature 'listofdesignkeys'
alias(object, model, ...)

## S4 method for signature 'listofkeyrings'
alias(object, model, ...)
```

Arguments

object	an object of the class.
model	an optional model formula (by default the first model in object) or, when object is a keymatrix, a matrix representing factorial model terms
fact	a character or numeric vector of parent factor names for the columns of object
block	a logical vector to identify the columns of object associated with a block factor
...	ignored

Details

- When object is a keymatrix, "alias" displays the key matrix and the factorial effects confounded with the mean. It prints the unaliased treatment effects, then the groups of aliased treatment effects, then the treatments effects confounded with block effects and finally the unaliased block effects, when considering all the factorial terms that are represented in the model argument, which is set if missing to the identity matrix (main effects only).

Value

- When object is a keymatrix, a vector with (i) the number of unaliased treatment effecs; (ii) the number of mutually aliased treatment effects; (iii) the number of treatment effects aliased with block effects.
- When object is a designkey, an invisible NULL.
- When object is a listofkeyrings, the factors, the model and the number of solutions for each prime in a list indexed by the primes p of the object. Each element is a 3-column matrix with one row per solution for prime p. The columns give (i) the number of unaliased treatment effecs; (ii) the number of mutually aliased treatment effects; (iii) the number of treatment effects aliased with block effects.
- The method is NOT YET IMPLEMENTED on objects of class listofdesignkeys.

See Also

Classes where this method applies: designkey, keymatrix, listofkeyrings.

Examples

```
### Creation of an object of class "listofkeyrings"
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
nunits=3^3, base=~A+B+C, max.sol=2)
### alias on an object of class "keymatrix"
alias(K0[[1]][[1]])
### alias on an object of class "designkey"
alias(K0[1])
### alias on an object of class "listofkeyrings"
alias(K0)
```

`as.data.frame.planordesign` *Coerce a planordesign object to a data frame*

Description

Extracts from a `planordesign` object the slot "design", i.e the dataframe containing the final design, and stores the other slots in attributes

Usage

```
## S4 method for signature 'planordesign'
as.data.frame(x, ...)
```

Arguments

`x`	an object of class `planordesign`
`...`	Ignored

Value

A data frame with attributes "factors", "model", "designkey", "nunits", "recursive".

Examples

```
### Creation of a 'planordesign' object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
P0 <- planor.design(key=K0, select=1)
# Convert into a data frame
D0=as.data.frame(P0)
```

`bind-methods` *Methods for function* `bind` *in package* **planor***: bind two objects*

Description

Bind two objects of the same class.

Usage

```
## S4 method for signature 'designfactors,designfactors'
bind(x, y)
```

Arguments

`x`	an object of the first class in the signature.
`y`	an object of the second class in the signature.

Value

An object of the same class as `x` and `y` containing their joint content.

Note

Factors of the same name are repeated with distinct names and advertised with a warning.

See Also

Class where this method applies: `designfactors`

Examples

```
F1 <- planor.factors(factors=c("block",LETTERS[1:4]), nlevels=c(6,6,4,2,6))
F2 <- planor.factors(factors=c("block",LETTERS[11:12]), nlevels=c(6,6,4))
### Method bind on 'designfactors' objects
F3 <- bind(F1,F2)
names(F3)
```

---

designfactors-class

Class `designfactors` *and methods of the class*

Description

An S4 class to represent the design factors in the **planor** package and to store their characteristics.

Objects from the Class

Objects from this class can be created explicitly by calls to `planor.factors` or implicitly by functions such as `planor.designkey`.

Slots

`fact.info`**:** a dataframe with one row per factor and with columns progressively storing information on the factors, in particular their numbers of levels (`nlev`).

`pseudo.info`**:** a dataframe with one row per pseudofactor and with columns progressively storing information on the pseudofactors.

`levels`**:** a list of numeric or character vectors, with each vector containing the levels of one factor.

Methods

[extract a subset of factors and update all the slots.

bind bind two objects of the class and update all the slots. See `bind`.

length return the number of factors.

names return the names of the factors.

Details

Depending on the context and on the construction stage, `fact.info` may contain logical columns that identify the block factors (`block`), the ordered factors (`ordered`), the basic factors (`basic`) and so on. It may also include columns that store the information on the hierarchy relationships between factors, if any.

In **planor**, factors are systematically decomposed into pseudofactors which all have a prime number of levels and which play a key role in the design generation. The information on the pseudofactors is stored in the `pseudo.info` slot. In addition to the columns of `fact.info`, it contains a column (called `parent`) to give the factor that each pseudofactor decomposes.

Author(s)

H. Monod, and A. Bouvier

See Also

Creator function: `planor.factors`

Examples

```
F1 <- planor.factors(factors=c("block",LETTERS[1:4]), nlevels=c(6,6,4,2,6))
F2 <- planor.factors(factors=c("block",LETTERS[11:12]), nlevels=c(4,6,6))
## Method bind - see the warning because two factors in F1 and F2 have
## the same name
F3 <- bind(F1,F2)
names(F3)
length(F3)
F3@levels
F3.trt <- F3[c(2:5,7,8)]
names(F3.trt)
```

`designkey-class` *Class* `designkey` *and methods of the class*

Description

An S4 class to represent a design-key solution in package **planor**.

Objects from the Class

Objects can be created by extraction from an object of class `listofkeyrings` or class `listofdesignkeys`.

Slots

`.Data`**:** a single design-key solution, i.e a list with one `keymatrix` per prime

`factors`**:** an object of class `designfactors` which contains the factors' specifications

`model`**:** a `"list"` which contains the model and estimate specifications

`nunits`**:** the number of units of the design

`recursive`**:** a `"logical"` equal to TRUE if the design has been constructed recursively

Extends

Class "list", from data part. Class "vector", by class "list", distance 2.

Methods

alias summarize the design properties. See alias.

planor.design build the design from the design key matrix. See planor.design.

show display the object. see show

summary summarize the design properties. See summary

Author(s)

H. Monod, and A. Bouvier

Examples

```
### Creation of a 'designkey' object
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
nunits=3^3, base=~A+B+C, max.sol=2)
print(K0[1])
```

---

getDesign-methods *Methods for function* getDesign *in package* **planor***: extract a design*

Description

Extract a design data frame from an object

Usage

```
## S4 method for signature 'planordesign'
getDesign( object  )
```

Arguments

object	Object of the class

Value

A data frame which contains the design.

See Also

Classes where this method applies: planordesign.

Examples

```
### Creation of a 'planordesign' object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
P0 <- planor.design(key=K0, select=1)
## Method getDesign on the 'planordesign' object
 show(getDesign(P0))
```

---

keymatrix-class *Class* keymatrix *and methods of the class*

Description

An S4 class to represent an elementary key matrix in package **planor**

Objects from the Class

Objects from this class are usually components of an object of class keyring or designkey

Slots

.Data: a matrix of integers modulo p

p: a prime number

Extends

Class matrix, from data part. Class array, by class "matrix", distance 2. Class structure, by class "matrix", distance 3. Class vector, by class "matrix", distance 4, with explicit coerce.

Methods

alias gives the aliasing relationships of the key matrix. See alias.

show display the object. See show-method

summary summarize the design properties. See summary

Author(s)

H. Monod, and A. Bouvier

See Also

keyring, designkey

Examples

```
showClass("keymatrix")
### Creation of a 'listofkeyrings' object
K0 <- planor.designkey(factors=c("block", LETTERS[1:4]), nlevels=rep(3,5),
  model=~block + (A+B+C+D)^2, estimate=~A+B+C+D,
  nunits=3^3, base=~A+B+C, max.sol=2)
# Method show on a 'keymatrix' of K0
show(K0[[1]][[1]])
```

keyring-class *Class* keyring *and methods of the class*

Description

An S4 class to represent a list of design-key matrices which are associated with the same prime and which represent alternative solutions to the same design specifications.

Objects from the Class

Each component of the structure returned by planor.designkey is a keyring when the case is not recursive.

Slots

.Data: a list of keymatrix objects.

p: a prime number.

LIB: a list containing a vector of row names and a vector of column names. The names are the same for all key matrices.

pseudo.info: a dataframe containing information on the pseudofactors associated with the key matrices. See the description of the class designfactors.

Extends

Class "list", from data part. Class "vector", by class "list", distance 2.

Methods

show display the object. See show-method.

summary summarize the design properties. See summary

Note

Each key matrix in a keyring object is a possible solution to the same factors, model and estimate specifications, with respect to the same prime number. An object of class listofkeyrings is a list of keyring objects associated with the different primes involved in a given factorial design problem.

Author(s)

H. Monod, and A. Bouvier

See Also

planor.designkey, method pick in class listofkeyrings, method summary in class keymatrix and the class keyring

Examples

```
showClass("keyring")
### Creation of a 'listofkeyrings' object
K0 <- planor.designkey(factors=c("block", LETTERS[1:4]), nlevels=rep(3,5),
  model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
  nunits=3^3, base=~A+B+C, max.sol=2)
## Method show applied on a 'keyring' component of K0
show(K0[[1]])
```

---

listofdesignkeys-class

Class `listofdesignkeys` *and methods of the class*

Description

An S4 class to represent a list of design key solutions

Objects from the Class

Objects are created by `planor.designkey`, when the search is recursive.

Slots

`.Data`**:** a list of objects of class `designkey`.

`factors`**:** an object of class `designfactors` which contains the factors' specifications.

`model`**:** a `"list"` which contains the model and estimate specifications.

`nunits`**:** the number of units in the design.

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Methods

alias FUNCTION NOT YET IMPLEMENTED

[extract one design key in the list.

pick extract one design key in the list. See `pick`.

planor.design build a design from one design key in the list. See `planor.design`.

show display the object. See `show`.

summary summarize the design properties. See `summary`

Author(s)

H. Monod, and A. Bouvier

See Also

Creator function: `planor.designkey`

Examples

```
showClass("listofdesignkeys")
### Creation of a "listofdesignkeys" object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
# Show the object
show(K0)
## Method length
length(K0)
## Extraction: the following two commands are equivalent
K <- K0[2]
K <- pick(K0,2)
```

---

listofkeyrings-class

Class `listofkeyrings` *and methods of the class*

Description

An S4 class to store design key solutions when there is only one prime involved or when the solutions are independent between primes.

Objects from the Class

Objects are created by `planor.designkey`, when the case is not recursive.

Slots

`.Data`: a list of objects of class `keyring` associated with different primes.

`factors`: an object of class `designfactors` which contains the factors' specifications.

`model`: a `"list"` which contains the model and estimate specifications.

`nunits`: the number of units of the design.

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Methods

alias give the aliasing for each key-matrix. See `alias`.

[extract one design key by taking one key matrix per prime.

pick extract one design key by taking one key matrix per prime. See `pick`.

planor.design build a design using one key matrix per prime. See `planor.design`.

show display the object. See show.

summary summarize the design properties from `object`. See `summary`.

Author(s)

H. Monod, and A. Bouvier

See Also

Creator function: planor.designkey

Examples

```
showClass("listofkeyrings")
### Creation of a 'listofkeyrings' objct
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
   model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
   nunits=3^3, base=~A+B+C, max.sol=2)
show(K0)
```

---

makedesignkey *A function to turn integer matrices into an object of class designkey*

Description

Creates an object of class designkey directly from a list of integer matrices

Usage

```
makedesignkey(keys, primes)
```

Arguments

keys	a list of n integer matrices with column names
primes	a vector of n prime numbers

Details

the names of the factors are extracted from the matrix column names

Value

an object of class designkey

Author(s)

H. Monod

See Also

Class designkey

Examples

```
mat1 <- cbind(diag(3),1)
colnames(mat1) <- c("A","B","C","D")
mat2 <- cbind(diag(2),c(1,2))
colnames(mat2) <- c("E","F","G")
mat.dk <- makedesignkey(list(mat1,mat2), primes=c(2,3))
print(mat.dk)
summary(mat.dk)
alias(mat.dk)
mat.plan <- planor.design(mat.dk)
```

---

pick-methods *Methods for function* `pick` *(or* `[]` *) in package* **planor***: extract a single result from an object of class list*

Description

Extract a single `designkey` object (with one key matrix per prime) from a complex object

Usage

```
## S4 method for signature 'listofdesignkeys'
pick(keys, selection)
## S4 method for signature 'listofkeyrings'
pick(keys, selection)
```

Arguments

`keys`	an object of the class
`selection`	when `keys` is a `listofdesignkeys` object, an integer scalar equal to the position of the required solution. when `keys` is a `listofkeyrings` object, the index vector to select the key matrix for each prime.

Value

An object of class `designkey`, which contains the selected design

Note

`K <- pick(K0,1)` can be simply written `K <- K0[1]`

See Also

Classes where this method applies: `listofdesignkeys`, `listofkeyrings`.

Examples

```
# Creation of an object of class "listofdesignkeys"
K2 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
nlevels=c(3,2,2,3,2,2),  model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2 , nunits=12,
base=~R+C+U, max.sol=2)
# Method 'pick' applied on the "listofdesignkeys" object
K2.1 <- pick(K2,1)
K2.1 <- K2[1] ## Another way of extracting ([ is synonym of pick)

# Creation of an object of class "listofkeyrings"
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"),
nlevels=rep(3,5), model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
nunits=3^3, base=~A+B+C, max.sol=2)
# Method 'pick' applied on the "listofkeyrings" object
K0.1 <- pick(K0,1)
K0.1 <- K0[1] ## the same
```

---

planor.design-methods

Methods for function `planor.design` *in package* **planor***: build a design from a design-key solution*

Description

Construction of a factorial design from an object containing key matrices

Usage

```
## S4 method for signature 'designkey'
planor.design(key, randomize=NULL, ...)

## S4 method for signature 'listofdesignkeys'
planor.design(key, randomize=NULL, selection=1, ...)

## S4 method for signature 'listofkeyrings'
planor.design(key, randomize=NULL, selection,...)

## S4 method for signature 'numeric'
planor.design(key, start=1)
```

Arguments

`key`	an object of the first class in the signature, or a vector of integers.
`randomize`	an optional formula to specify the block structure for design randomization.
`selection`	when `key` is a `listofdesignkeys` object, an integer scalar. when `key` is a `listofkeyrings` object, should be an index vector to select the key matrix for each prime.
`...`	additional arguments, in particular those related to randomization (see `planor.randomize`).
`start`	an integer from where to start the series of symbols.

Details

- When `key` is numeric, it should be a vector of integers of length `s`. Then, the function generates a full factorial $n_1 x n_2 x ... x n_s$ design with columns considered as factors. It returns an integer matrix with prod(n) rows and s columns giving all combinations along the rows, in lexicographic order.
- When `key` is a `listofdesignkeys` object, build one design from a selected solution.

Value

An object of class `planordesign`, which contains the design built from the input. This function is restricted to giving a single design. When `key` is numeric, see Details

See Also

Classes where this method applies: `designkey`, `listofdesignkeys`, `listofkeyrings`.

Examples

```
### Creation of a 'listofdesignkeys' object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
## Method planor.design applied on the 'listofdesignkeys' object
P0 <- planor.design(key=K0, select=1)
## Method planor.design applied on a designkey' object
P0 <- planor.design(K0[1])


### Creation of a 'listofkeyrings' object
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
nunits=3^3, base=~A+B+C, max.sol=2, verbose=TRUE)
## Method planor.design applied on a designkey' object
P0 <- planor.design(K0[1])
P0.R <- planor.design(K0[1], randomize=~A+B+C+D) ## randomize the final design
```

`planor.designkey` *Search for a design key or a collection of design keys*

Description

Search for a design key or a collection of design keys that satisfy the design properties specified by the arguments. This function calls the core algorithms implemented in **planor**.

Usage

```
planor.designkey(factors, nlevels, block, ordered, hierarchy, model,
    estimate, listofmodels, resolution, nunits, base, max.sol=1,
    randomsearch=FALSE, verbose=TRUE)
```

Arguments

factors	an object of class designfactors, typically an output from planor.factors. Alternatively, the arguments factors, nlevels, ordered, hierarchy may follow the syntax of planor.factors.
nlevels	See planor.factors. Ignored if factors is of class designfactors.
block	See planor.factors. Ignored if factors is of class designfactors.
ordered	See planor.factors. Ignored if factors is of class designfactors.
hierarchy	See planor.factors. Ignored if factors is of class designfactors.
model	a list of model-estimate pairs of formulae, typically an output from planor.model. Alternatively, the arguments model, estimate, listofmodels and resolution may follow the syntax of planor.model.
estimate	See planor.model. Ignored if model is a list.
listofmodels	See planor.model. Ignored if model is a list.
resolution	See planor.model. Ignored if model is a list.
nunits	a scalar giving the total number of units in the design
base	an optional additive formula to specify the basic factors. See Note.
max.sol	maximum number of solutions before exit.
randomsearch	a logical; if TRUE, the searches for a key matrix are performed in a random order.
verbose	a logical to set to TRUE for verbose display.

Details

The methods implemented in **planor** rely on a decomposition of the design search according to prime numbers. The prime numbers involved are those that decompose the numbers of levels of the factors. For example, if all factors have 2, 4, or 8 levels, then the number of units must be a power of 2 and the only prime number involved is 2. This is called the *symmetric* case. But if at least one factor has 6 levels, or if factor A has 2 levels and factor B has 3 levels, then the number of units must be the product of a power of 2 by a power of 3. In this case the search is automatically decomposed into one for prime 2 and one for prime 3. This is called the *asymmetric* case.

In the symmetric case with prime p, a regular factorial design requires a single key matrix of integers modulo p. In the asymmetric case, it requires one key matrix per prime. In **planor**, key matrices are stored in objects of class keymatrix. The lists made of one key matrix per prime are called design keys. They are stored in objects of class designkey.

The function planor.designkey essentially searches for design keys that satisfy the user specifications. For technical reasons, however, its output can take two different forms: either an object of class listofkeyrings or an object of class listofdesignkeys. The function planor.designkey detects automatically which case applies. In the first case (*independent case*), the key matrix solutions can be searched independently between primes and they are stored in objects of class listofkeyrings. The second case (*recursive case*) occurs exceptionnally. In that case the search cannot be independent between primes and so the different solutions are directly stored in a list of class listofdesignkeys.

Value

an object of class listofkeyrings in most cases. Otherwise, i.e in recursive cases, an object of class listofdesignkeys.

Note

The `base` formula must be an additive formula involving a subset of factors, called the basic factors. Using the `base` argument ensures that the design solutions will include the full factorial design for the basic factors. This option can speed up the search because it restricts the domain to be explored by the search algorithm.

Author(s)

H. Monod, and A. Bouvier

See Also

`planor.factors`, `planor.model`, and the classes `designfactors`, `listofkeyrings`, `listofdesignkeys`

Examples

```
K0 <- planor.designkey(factors=c("block", LETTERS[1:4]),
  nlevels=rep(3,5), model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
  nunits=3^3, base=~A+B+C, max.sol=2)
## With automatic model generation
Km <- planor.designkey(factors=c("block", LETTERS[1:4]),
  nlevels=rep(2,5), resolution=3, nunits=2^4)
```

`planor.factors` *Create an object of class 'designfactors'*

Description

A function to create an object of class `designfactors`, either by giving the factor names and their numbers of levels, or by giving a named list of factor levels. Both ways can be used in the same call. Additional information can be provided that will be used during the design search or in the summary functions.

Usage

```
planor.factors(factors = NULL, nlevels = NULL,
  block = NULL, ordered = NULL, hierarchy = NULL,
  dummy = FALSE)
```

Arguments

`factors`	a character vector of factor names, or possibly a scalar, a dataframe or a list (see DETAILS)
`nlevels`	a vector of level numbers for each factor name (see DETAILS)
`block`	an additive model formula to indicate the block factors
`ordered`	an additive model formula to indicate the quantitative factors (not used at all in the present version)
`hierarchy`	a formula or a list of formulae to indicate hierarchy relationships between factors (see the **planor** vignette for details)
`dummy`	a logical to identify dummy factors created and deleted by PLANOR functions for technical reasons

Value

An object of class designfactors

Note

The basic usage is to specify the names of the factors by a character vector of length n in argument factors and their numbers of levels by a numeric vector of length n in argument nlevels. Alternatively, the factors argument can be an integer n, in which case the first n capital letters of the alphabet are used as factor names. If nlevels is a scalar s, it is considered that all factors have s levels. There are two more possibilities which allow for alphanumeric factor levels. If factors is a dataframe, the factors in this dataframe are extracted together with their levels. Finally factors can be a named list of n vectors, with each vector containing the levels of one factor. Note that nlevels is ignored in these latter two cases. See the examples. The argument block allows to specify the block or nuisance factors. This information is used by the alias and summary functions but it has no effect on the design generation and randomization which depend on other arguments.

Author(s)

H. Monod, and A. Bouvier

See Also

Class designfactors

Examples

```
planor.factors(c("A","B","C","P"),c(2,3,6,3))
planor.factors(LETTERS[1:12],2)
planor.factors(12,2)
planor.factors( c("A","B","Block"), 3, block=~Block )
zz <- planor.factors( c("A","B","Block"), c(2,3,5))
zz@levels$A <- c("plus","moins")
planor.factors(factors=list(A=c("plus","moins"), B=1:3, Block=1:5))
AB <- data.frame( A=c(rep(c("a","b"),3)), B=rep(c("z","zz","zzz"),rep(2,3)), C=1:6  )
planor.factors(factors=AB)
```

---

planor.harmonize *Harmonize the factors*

Description

Harmonize the factors originating from a list of factors, a list of models, and a list of basic factors (this function is essentially for internal use)

Usage

```
planor.harmonize(factors, nlevels, ordered, hierarchy, model, estimate,
    listofmodels, base)
```

Arguments

factors	an object of class designfactors, typically an output from planor.factors). Otherwise the arguments factors, nlevels, ordered, hierarchy follow the syntax of planor.factors.
nlevels	See planor.factors. Ignored if factors is of class designfactors.
ordered	See planor.factors. Ignored if factors is of class designfactors.
hierarchy	See planor.factors. Ignored if factors is an object of class designfactors.
model	a list of model-estimate pairs of formulae, typically an output from planor.model. Otherwise the arguments model, estimate and listofmodels follow the syntax of planor.model.
estimate	See planor.model. Ignored if model is a list.
listofmodels	See planor.model. Ignored if model is a list.
base	an optional formula to specify the basic factors. These factors must belong to the factors argument

Value

An object of class designfactors very similar to factors, but with two additional logical columns in slots fact.info and pseudo.info:

- model (TRUE for the factors present in at least one model formula),

- basic (TRUE for the basic factors).

Note

This function is called at the start of the design search. It is essentially a check that the factors in all three arguments are coherent, even though it performs some additional tasks. The function stops if it detects a model or basic factor that is absent from factors. This is because the number of levels of such a factor is unknown and so the design search cannot proceed. Besides, the function eliminates the factors that do appear neither in model nor in base and it reorders the factors by putting first the basic ones.

Author(s)

H. Monod, and A. Bouvier

Examples

```
F2 <- planor.factors(factors=c("block",LETTERS[1:4]), nlevels=c(6,6,6,4,2))
M2 <- planor.model( model=~block+(A+B+C)^2, estimate=~A+B+C )
F2.h <- planor.harmonize(factors=F2, model=M2, base=~A+B)
names(F2)
names(F2.h)
```

`planor.model` *Model and estimate specifications for a design search*

Description

A function to declare the factorial terms that must be considered as non-negligible and the factorial terms that must be estimable when the experiment will be analysed.

Usage

```
planor.model(model, estimate, listofmodels, resolution, factors)
```

Arguments

`model`	main model formula. It contains all the non-negligible factorial terms.
`estimate`	optional formula specifying the factorial terms to estimate. If missing, it is considered that all factorial terms in `model` have to be estimated.
`listofmodels`	list of `c(model, estimate)` pairs, where `model` and `estimate` are formulae; using several pairs allows more flexibility in the design constraints (see Kobilinsky, 2005, or the split-plot example in the vignette); `estimate` is optional.
`resolution`	an integer larger than or equal to 3, to specify the design resolution. When set, the `model` and `estimate` arguments are ignored. See Note.
`factors`	a `designfactors` object, typically an output from `planor.factors`. It must be set only when the `resolution` argument is used.

Value

A list of `c(model, estimate)` pairs, where `model` and `estimate` are formulae

Note

The user can specify:
1/ either, `model` or `listofmodels` or both,
2/ or, `resolution` and `factors`, and possibly `listofmodels`.

When `model` and `resolution` are both set, `model` is ignored.

The second case, — when `resolution` and `factors` are set —, causes the automatic generation of the main `c(model, estimate)` pair. Assuming `S` denotes the additive formula including all factors,
- if `resolution` is odd, the model formula is `~(S)^(resolution-1)/2`,
- if `resolution` is even, the model formula is `~(S)^(resolution/2)` and the estimate formula is `~(S)^(resolution/2)-1`.

Author(s)

H. Monod, and A. Bouvier

Examples

```
# Basic example
planor.model(model=~block + (A+B+C)^2, estimate=~(A+B+C)^2)
# Resolution: both calls to 'planor.model' below are equivalent
planor.model(model=~(A+B+C+D)^2, estimate=~A+B+C+D)
myfactors <- planor.factors(factors=c(LETTERS[1:4]), nlevels=rep(2,4))
planor.model(resolution=4, factors=myfactors)
# Complicated examples
planor.model(~A+B+C+D+A:B, ~A+B+C+D, listofmodels=list(c(~E+F,~E)))
planor.model(~A+B+C+D+A:B,~A+B+C+D, listofmodels=
                              list(c(~E+F,~E), ~G, ~H, c(~M+N,~N)))
```

---

planor.randomize — *A function to randomize a factorial design according to an orthogonal block structure*

Description

A function to randomize a factorial design according to a specified block structure formula

Usage

```
planor.randomize(blockformula, data, out.order, keep.initial=FALSE)
```

Arguments

`blockformula`	the block structure formula
`data`	a data frame.
`out.order`	a list of `data` factors that will be used to order the rows of the randomized design; if missing, the factors of the block formula are used.
`keep.initial`	if TRUE, the initial row order of the design is stored in column `InitialUNITS` of the returned dataframe.

Value

the input data frame after randomization.

Note

Each name in `blockformula` must correspond to a factor of the dataframe `data`. The only exception is `UNITS`. If `UNITS` is used in `blockformula` but absent from `data`, a factor is added to `data`, with one level per row. See the examples below for the usage of `UNITS` in `blockformula`.

Author(s)

H. Monod, and A. Bouvier

References

Bailey, R.A., 1983. Generalized wreath products of permutation groups. *Proc. London Math. Soc.*, 47, 69-82.

Kobilinsky A., 1989. Randomization of a cartesian block structure. Technical Report. Laboratoire de Biométrie de l'INRA Versailles.

Examples

```
## Block design
Design <- data.frame(block=rep(1:4,rep(2,4)),
treatment=c("A1","B1","A2","B2","A3","B3","A4","B4"))
planor.randomize(~block, data=Design)       ##  no within-block randomization
planor.randomize(~block/UNITS, data=Design) ##  blocks and units within blocks randomizat
## Row-Column design
RowColDes <- data.frame(row=rep(1:3,rep(3,3)),col=rep(1:3,3),
treatment=LETTERS[c(1:3,2,3,1,3,1,2)],
oldRow=rep(1:3,rep(3,3)),oldCol=rep(1:3,3))
planor.randomize(~row*col, data=RowColDes)
```

---

planordesign-class *Class* planordesign *and methods of the class*

Description

An S4 class to represent a final design

Objects from the Class

Objects can be created by calls to method planor.design applied on an object of class designkey or on an object of class listofkeyrings, and by calls to regular.design when argument output is equal to 'planordesign'

Slots

design: a dataframe containing the final design

factors: an object of class designfactors which contains the factors' specifications

model: a list containing the model and estimate specifications

designkey: a list which contains the designkey matrices used to create the object

nunits: the number of units of the design

recursive: a "logical" equal to TRUE if the design has been constructed recursively

Methods

getDesign extract a design data frame. See getDesign

as.data.frame turn into a data frame. See as.data.frame.planordesign

Author(s)

H. Monod, and A. Bouvier

See Also

Creators: method planor.design applied on an object of class designkey or class listofkeyrings or class listofdesignkeys
See also class designfactors.

Examples

```
showClass("planordesign")
### Creation of a 'listofdesignkeys' object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
## Creation of a 'planordesign' object from K0
P0 <- planor.design(key=K0, select=1)
show(P0)
```

`regular.design` *Construct and randomize a regular factorial design*

Description

Construct and randomize a regular factorial design

Usage

```
regular.design(factors = NULL, nlevels = NULL, block = NULL,
ordered = NULL, hierarchy = NULL, model = NULL, estimate = NULL,
listofmodels = NULL, resolution = NULL, nunits = NULL,
base = NULL, randomize = NULL, randomsearch = FALSE,
output = "planordesign", verbose = FALSE, ...)
```

Arguments

`factors`	an object of class `designfactors`, typically an output from `planor.factors`). Otherwise the arguments `factors`, `nlevels`, `ordered`, `hierarchy` follow the syntax of `planor.factors`.
`nlevels`	See `planor.factors`. Ignored if `factors` is of class `designfactors`.
`block`	See `planor.factors`. Ignored if `factors` is of class `designfactors`.
`ordered`	See `planor.factors`. Ignored if `factors` is of class `designfactors`.
`hierarchy`	See `planor.factors`. Ignored if `factors` is of class `designfactors`.
`model`	a list of model-estimate pairs of formulae, typically an output from `planor.model`. Otherwise the arguments `model`, `estimate`, `listofmodels` and `resolution` follow the syntax of `planor.model`.
`estimate`	See `planor.model`. Ignored if `model` is a list.
`listofmodels`	See `planor.model`. Ignored if `model` is a list.
`resolution`	See `planor.model`. Ignored if `model` is a list.
`nunits`	See `planor.designkey`.
`base`	See `planor.designkey`.
`randomize`	an optional formula to randomize the design.
`randomsearch`	See `planor.designkey`.
`output`	a string to specify the class of the output value: either a `data.frame` or a `planordesign` object
`verbose`	a `logical` to set to `TRUE` for verbose display
`...`	additional arguments, in particular those related to randomization

Value

An object of class `data.frame` or `planordesign`, depending on the `output` argument

Author(s)

H. Monod, and A. Bouvier

See Also

`planor.factors`, `planor.model`, and the classes `designfactors`, `listofkeyrings`, `listofdesignkeys`

Examples

```
mydesign <- regular.design(factors=c("block", LETTERS[1:4]),
  nlevels=rep(3,5), model=~block + (A+B+C+D)^2, estimate=~A+B+C+D,
  nunits=3^3, randomize=~block/UNITS)
print(mydesign)
```

`show-methods` *Methods for function* `show` *in package* **planor**

Description

print the design key matrices.

Usage

```
## S4 method for signature 'designkey'
show(object)

## S4 method for signature 'keymatrix'
show(object)

## S4 method for signature 'keyring'
show(object)

## S4 method for signature 'listofdesignkeys'
show(object)

## S4 method for signature 'listofkeyrings'
show(object)
```

Arguments

`object`	object of the class

Details

The slot `pseudo.info` of the objects of class `keymatrix` is invisible.

Value

NULL

Note

- The number of rows and columns of the matrices that are printed are limited by the option `planor.max.print`
- Objects of the class are automatically displayed by invocation of 'show' (see examples).

See Also

Classes where this method applies: `designkey`, `keymatrix`, `keyring`, `listofdesignkeys`, `listofkeyrings`

Examples

```
# Creation of a "listofdesignkeys" object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
 nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
 nunits=12, base=~R+C+U, max.sol=2)
## Method show applied on a "keymatrix" object
show(K0[[1]][[1]])
## Method show applied on a "designkey"  object
show(K0[1])
## Method show applied on the "listofdesignkeys" object
show(K0)
K0 # the same

### Creation of a "listofkeyrings" object
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
nunits=3^3, base=~A+B+C, max.sol=2)
## Method show applied on a "keyring" object
show(K0[[1]])
print(K0[[1]]) # the same
K0[[1]] # the same
## Method show applied on the "listofkeyrings" object
show(K0)
```

---

summary-methods *Methods for function* `summary` *in package* **planor**

Description

Summarize the design properties of an object, by printing the summary of each key matrix

Usage

```
## S4 method for signature 'designkey'
summary(object, show="dtbw", save="k", ...)

## S4 method for signature 'keymatrix'
summary(object, fact, block,  show="dtbw", save="k", ...)
```

```
## S4 method for signature 'keyring'
summary(object, show="tbw",save ="kw", ...)

## S4 method for signature 'listofdesignkeys'
summary(object, show="tbw", save="kw", ...)

## S4 method for signature 'listofkeyrings'
summary(object,  show="tbw",  save="kw", ...)
```

Arguments

object	an object of the class
fact	a character or numeric vector of parent factor names for the columns of the object
block	a logical vector to identify the columns of the object associated with a block factor
show	an optional string to identify the type of information to display. The recognized letters are: 'd' for the design keys matrices, 't' for the treatment effects confounded with the mean, 'b' for the block-and-treatment effects confounded with the mean, 'w' for the weight profiles
save	an optional string to identify the type of information to return. The recognized letters are: 'k' for the kernel matrices, 'w' for the weight profiles of the treatment effects confounded with the mean.
...	ignored

Details

The amount of display depends on the value of the argument show, and the type of returned information depends on the value of the argument save.

- When object is of class keymatrix, "summary" prints the key matrix, the factorial effects confounded with the mean, and the weight profiles of the effects confounded with the mean, according to the value of the argument show.
  The keymatrix argument being denoted by key,
  - The rows of key are associated with units factors (or pseudofactors) while its columns are associated with treatment or block factors (or pseudofactors).
  - The vectors in the arguments fact and block give information on the treatment and block factors, so that their length is expected to be equal to the number of columns of key.
  - If missing, fact attributes a distinct parent factor to each column of key and block is set to TRUE for all columns.

  "summary" returns a list with the components required by the argument save.
- When object is of class designkey, "summary" prints the summary of each of the key matrices. It returns a list with as many components as key matrices, each one with the components required by the argument save.
- When object is of class listofdesignkeys, "summary" prints the summary of each key matrix in each design key. It returns a list with as many components as design keys, each one is a list of the key matrices summaries.
- When object is of class listofkeyrings, "summary" prints the summary of each key matrix in each keyring. It returns a list with as many components as keyrings, each one is a list of the key matrices summaries.

- When `object` is of class `keyring`, "summary" prints the summary of each of its key matrices. It returns a list with as many components as key matrices.

Value

A list. See Details
Information returned for each key matrix depends on the argument `save`.

- When `save` includes the character 'k', the returned list has a component named 'k'.It is a matrix, the columns of which are kernel generators of the key matrices.
- When `save` includes the character 'w', the returned list has a component named 'w', which contains the weight profiles of the effects confounded with the mean.

Note

The number of rows and columns of the matrices that are printed are limited by the option `planor.max.print`.

See Also

Classes where this method applies: `designkey`, `keymatrix`, `keyring`, `listofdesignkeys`, `listofkeyrings`

Examples

```
### Creation of a "listofdesignkeys" object
K0 <- planor.designkey(factors=c("R","C","U","A","B1","B2"),
  nlevels=c(3,2,2,3,2,2), model=~R*C + (A+B1+B2)^2, estimate=~A:B1+A:B2,
  nunits=12, base=~R+C+U, max.sol=2)
## Method summary applied on a "keymatrix" object
r <- summary(K0[[1]][[1]])
## Method summary applied on a "designkey"  object
summary(K0[1], save=NULL)
# Method summary applied on the "listofdesignkeys" object
r <-summary(K0, show="dt")

### Creation of a "listofkeyrings" object
K0 <- planor.designkey(factors=c(LETTERS[1:4], "block"), nlevels=rep(3,5),
   model=~block+(A+B+C+D)^2, estimate=~A+B+C+D,
   nunits=3^3, base=~A+B+C, max.sol=2)
# Method summary applied on the "keymatrix" object
r <-summary(K0[[1]][[1]])
# Method summary applied on the "keyring" object
r <-summary(K0[[1]])
# Method summary applied on the "listofkeyrings" object
r <- summary(K0, show="dtb", save ="k")
print(r)
```

Index