Package 'pwr4exp'

October 11, 2024

Title Power Analysis for Research Experiments

Version 0.1.0

Description Provides tools for calculating statistical power and determining sample size for a variety of experimental designs used in agricultural and biological research, including completely randomized, block, and split-plot designs. Supports customized designs and allows specification of main effects, interactions, and contrasts for accurate power analysis.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports car (>= 3.1.2), emmeans (>= 1.10.3), lme4 (>= 1.1.35.4), lmerTest (>= 3.1.3), methods

Suggests agricolae, AlgDesign, crossdes, FrF2, knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/an-ethz/pwr4exp, https://an-ethz.github.io/pwr4exp/

BugReports https://github.com/an-ethz/pwr4exp/issues

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Kai Wang [aut, cre, cph] (<https://orcid.org/0000-0002-6672-1121>), Mutian Niu [aut, cph] (<https://orcid.org/0000-0003-4484-4611>)

Maintainer Kai Wang <kai.wang@usys.ethz.ch>

Repository CRAN

Date/Publication 2024-10-11 08:20:05 UTC

Contents

`calc.theta` *Calculate variance covariance parameters*

Description

Scale variance-covariance matrices as the relative Cholesky factors of each random effect term.

Usage

```
calc.theta(VarCov, sigma)
```

Arguments

`VarCov`	variance-covariance matrices. If there are multiple random effect groups, supply the variance-covariance matrix of each group as an element in a list.
`sigma`	standard deviation of random errors.

Value

theta

See Also

"theta" in `getME`, `lmer`

customLmerMod-class *Extend lmerModLmerTest class*

Description

This class extends `lmerModLmerTest` by adding a `DenDF` slot.

Slots

`DenDF` Numeric vector of denominator degrees of freedom.

designCRD *Creation of Experimental Designs*

Description

These functions are used to create design objects for the further evaluation of statistical power.

Usage

```
designCRD(treatments, label, replicates, formula, beta, sigma2)

designRCBD(treatments, label, blocks, formula, beta, VarCov, sigma2, ...)

designLSD(
  treatments,
  label,
  squares = 1,
  reuse = c("row", "col", "both"),
  formula,
  beta,
  VarCov,
  sigma2,
  ...
)

designCOD(treatments, label, squares = 1, formula, beta, VarCov, sigma2, ...)

designSPD(
  trt.main,
  trt.sub,
  label,
  replicates,
  formula,
  beta,
```

```
  VarCov,
  sigma2,
  ...
)

designCustom(design.df, formula, beta, VarCov, sigma2, design.name, ...)
```

Arguments

treatments	An integer-valued vector specifying the treatment structure, in which the length of the vector indicates the number of treatment factors, and each value represents the number of levels for each factor. A maximum of two factors is allowed, and they are arranged in a factorial design. For instance, `treatments = n` specifies one treatment factor with n levels, and `treatments=c(2,3)` creates a "2x3" factorial design of two treatment factors with 2 and 3 levels, respectively.
label	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
replicates	The number of experimental units per treatment in a completely randomized design or the number of experimental units (main plots) per treatment of main plot factors.
formula	A model formula for testing treatment effects in post-experimental data analysis. Use the syntax of `lm` for fixed effects and lmer for random effects. The response variable is always denoted as `y`. By default, all interaction terms between treatment factors are included in the formula.
beta	A numeric vector of expected model coefficients, representing the effect sizes. The first element represents the intercept term, corresponding to the mean of the reference level for categorical variables. Subsequent elements correspond to the effect sizes of the independent variables in the order they appear in the model matrix. For categorical variables, each coefficient represents the difference between a non-reference level and the reference level (intercept), as `contr.treatment` contrast coding is used for constructing the model matrix. Ensure that `beta` aligns with the columns of the model matrix, including any dummy variables created for categorical predictors.
sigma2	error variance.
blocks	The number of blocks.
VarCov	Variance-covariance components of random effects. For multiple random effect groups, supply the variance (for a single random effect term) or variance-covariance matrix (for two or more random effect terms) of each group in a list, following the order in the model formula.
...	Additional arguments passed to the `anova` function in `lmerTest`. The type of ANOVA table (default is Type III) and the method for computing denominator degrees of freedom (default is Satterthwaite's method) can be modified. For

	balanced designs, the choice of sum of squares (SS) and degrees of freedom (df) does not affect the results.
`squares`	The number of replicated squares. By default, 1, i.e., no replicated squares.
`reuse`	A character string specifying how to replicate squares when there are multiple squares. Options are: "row" for reusing row blocks, "col" for reusing column blocks, or "both" for reusing both row and column blocks to replicate a single square.
`trt.main`	An integer-valued vector specifying the treatment structure at main plot level for a split plot design, similar to `treatments`.
`trt.sub`	An integer-valued vector specifying the treatment structure at sub plot level for a split plot design, similar to `treatments`.
`design.df`	Required input for creating a customized design. A data frame with all independent variables of the design as columns, representing the actual data structure (long format data frame) without response variables.
`design.name`	Optional input for creating a customized design. A character.

Details

Each function creates a specific design as described below:

`designCRD` Completely Randomized Design. By default, the model formula is `y ~ trt` for one factor and `y ~ facA*facB` for two factors, unless explicitly specified. If the `label` argument is provided, the formula is automatically updated with the specified treatment factor names.

`designRCBD` Randomized Complete Block Design. The default model formula is `y ~ trt + (1|block)` for one factor and `y ~ facA*facB + (1|block)` for two factors. If `label` is provided, the fixed effect parts of the formula are automatically updated with the specified names. The label of block factor ("block") in the formula is not changeable.

`designLSD` Latin Square Design. The default formula is `y ~ trt + (1|row) + (1|col)` for one factor and `y ~ facA*facB + (1|row) + (1|col)` for two factors. If `label` is provided, the fixed effect parts of the formula are automatically updated with the specified names. The labels of row ("row") and column ("col") block factors are not changeable.

`designCOD` Crossover Design, which is a special case of LSD with time periods and individuals as blocks. Period blocks are reused when replicating squares. The default formula is `y ~ trt + (1|subject) + (1|period)` for one factor and `y ~ facA*facB + (1|subject) + (1|period)` for two factors. If `label` is provided, the fixed effect parts of the formula are automatically updated with the specified names. Note that "subject" and "period" are the labels for the two blocking factors and cannot be changed.

`designSPD` Split Plot Design. The default formula includes the main effects of all treatment factors at both the main and sub-plot levels, their interactions, and the random effects of main plots: `y ~ . + (1|mainplot)`. If `label` is provided, the fixed effect parts of the formula are automatically updated with the specified names. The experimental unit at the main plot level (i.e., the block factor at the subplot level) is always denoted as "mainplot".

`designCustom` Customized Design.

Value

a list with the design name, data structure (data frame), model formula, and a pseudo model object with the expected fixed and random effects.

See Also

pwr.anova(), pwr.contrast()

Examples

```
# Example 1: Evaluate the power of a CRD with one treatment factor

## Create a design object

crd <- designCRD(
  treatments = 4, # 4 levels of one treatment factor
  replicates = 12, # 12 units per level, 48 units totally
  # mean of level1, and the means of other levels minus level1, respectively
  beta = c(30, -2, 3, 5),
  sigma2 = 10 # error variance
)

## power of omnibus test
pwr.anova(crd)

## power of contrast
pwr.contrast(crd, specs = "trt", method = "pairwise") # pairwise comparisons
pwr.contrast(crd, specs = "trt", method = "poly") # polynomial contrasts

# Example 2: Evaluate the power of an RCBD with 2 x 2 factorial treatments

# Treatment factors are A (A1 vs. A2) and B (B1 vs. B2).
# To illustrate how to provide `beta`, treatment means are presented:
#     B1  B2
# A1  20  24
# A2  17  22
#
# From these means, we calculate:
# 1. the mean of reference level (A1B1): 20
# 2. the effect of A2 alone: Effect_A2 = A2B1 - A1B1 = 17 - 20 = -3
# 3. the effect of B2 alone: Effect_A2 = A1B2 - A1B1 = 24 - 20 = 4
# 4. the interaction effect of A2 and B2:
#    Interaction_A2B2 = A2B2 - A2B1 - A1B2 + A1B1 = 22 - 17 - 24 + 20 = 1, representing
#    the additional effect of combining A2B2 compared to what would be expected
#    from the sum of individual effects of A2 and B2.

# The `beta` vector is constructed as:
# beta = c(mean_A1B1, Effect_A2, Effect_B2, Interaction_A2B2)
# beta = c(20, -3, 4, 1)

## Create a design object

rcbd <- designRCBD(
  # 2x2 factorial design
  treatments = c(2, 2),
  # Specify treatment names
  label = list(A = c("A1", "A2"), B = c("B1", "B2")),
```

```
  # 12 blocks, totaling 48 experimental units
  blocks = 12,
  # Mean of the reference level and effect sizes as calculated above
  beta = c(20, -3, 4, 1),
  # Variance of block effects (between-block variance)
  VarCov = 30,
  # Error variance (within-block variance)
  sigma2 = 20
)

## power of omnibus test

pwr.anova(rcbd)

## power of B2 vs. B1 at each level of A
pwr.contrast(rcbd, specs = ~B|A, method = "pairwise")

# More examples are available in the package vignette("pwr4exp")
# and on the package website: https://an-ethz.github.io/pwr4exp/
```

`df.cod` *Create a data frame for Crossover design*

Description

Create a data frame for Crossover design

Usage

```
df.cod(treatments, label, squares)
```

Arguments

`treatments`	An integer-valued vector specifying the treatment structure, in which the length of the vector indicates the number of treatment factors, and each value represents the number of levels for each factor. A maximum of two factors is allowed, and they are arranged in a factorial design. For instance, `treatments = n` specifies one treatment factor with n levels, and `treatments=c(2,3)` creates a "2x3" factorial design of two treatment factors with 2 and 3 levels, respectively.
`label`	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
`squares`	The number of replicated squares. By default, 1, i.e., no replicated squares.

Value

a data.frame with columns for treatment factors, individuals (row block factor), period (column block factor), and squares

---

`df.crd` *Create a data frame of completely randomized design*

Description

Create a data frame of completely randomized design

Usage

```
df.crd(treatments, label, replicates)
```

Arguments

`treatments`	An integer-valued vector specifying the treatment structure, in which the length of the vector indicates the number of treatment factors, and each value represents the number of levels for each factor. A maximum of two factors is allowed, and they are arranged in a factorial design. For instance, `treatments = n` specifies one treatment factor with n levels, and `treatments=c(2,3)` creates a "2x3" factorial design of two treatment factors with 2 and 3 levels, respectively.
`label`	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
`replicates`	The number of experimental units per treatment.

Value

a data.frame with columns for treatment factors and replicates

df.lsd *Create a data frame for Latin square design*

Description

Create a data frame for Latin square design

Usage

```
df.lsd(treatments, label, squares = 1, reuse = c("row", "col", "both"))
```

Arguments

treatments	An integer-valued vector specifying the treatment structure, in which the length of the vector indicates the number of treatment factors, and each value represents the number of levels for each factor. A maximum of two factors is allowed, and they are arranged in a factorial design. For instance, `treatments = n` specifies one treatment factor with n levels, and `treatments=c(2,3)` creates a "2x3" factorial design of two treatment factors with 2 and 3 levels, respectively.
label	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
squares	the number of replicated squares
reuse	A character string specifying how to replicate squares when there are multiple squares. Options are: "row" for reusing row blocks, "col" for reusing column blocks, or "both" for reusing both row and column blocks to replicate a single square.

Value

a data.frame with columns for treatment factors, row and column block factors, and squares

df.rcbd *Create a data frame of randomized complete block design*

Description

Create a data frame of randomized complete block design

Usage

```
df.rcbd(treatments, label, blocks)
```

Arguments

`treatments`	An integer-valued vector specifying the treatment structure, in which the length of the vector indicates the number of treatment factors, and each value represents the number of levels for each factor. A maximum of two factors is allowed, and they are arranged in a factorial design. For instance, `treatments = n` specifies one treatment factor with n levels, and `treatments=c(2,3)` creates a "2x3" factorial design of two treatment factors with 2 and 3 levels, respectively.
`label`	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
`blocks`	the number of blocks

Value

a data.frame with columns for blocks and treatment factors

`df.spd` *Create data frame for split-plot design*

Description

Create data frame for split-plot design

Usage

```
df.spd(trt.main, trt.sub, label, replicates)
```

Arguments

`trt.main`	an integer-valued vector specifying the treatment structure at main plot level, similar to `df.crd`.
`trt.sub`	an integer-valued vector specifying the treatment structure at sub plot level, similar to `trt.main`.
`label`	Optional. A list of character vectors specifying the names of treatment factors and factor levels. Each vector in the list represents a treatment factor, where the name of the vector specifies the name of the factor, and the values in the vector are the labels for that factor's levels. If not provided, factors and levels for one and two treatment factors are labeled as `list(trt = c("1", "2", ...))` and `list(facA = c("1", "2", ...), facB = c("1", "2", ...))`, respectively.
`replicates`	the number of experimental units (main plots) per treatment of main plot factors.

Value

a data.frame with columns for main plots, main treatments, and sub-treatments

`find_sample_size` *Determine the sample size required to achieve the target power*

Description

This function finds the minimum sample size needed to achieve the target power for a given design. It uses an iterative approach to determine the minimum number of replications by traversing through a series of integers.

Usage

```
find_sample_size(
  design.quote,
  alpha = 0.05,
  target.power = 0.8,
  n_init = 2,
  n_max = 99,
  ...
)
```

Arguments

`design.quote`	a quoted design object with unknown and unevaluated replications to be evaluated with varying values
`alpha`	type I error rate, default is 0.05
`target.power`	the target power can be a single value for all factors or a vector of containing individual values for different factors, default is 0.8
`n_init`	the initial replications for the iterative process, default is 2
`n_max`	the maximum number of replications for the iterative process, default is 99
`...`	additional arguments passed to `pwr.anova`

Value

A data frame with type I error rate (alpha), realized power (power), and minimum sample size (best_n).

Examples

```
# create a LSD object with unknown replications (\code{squares = n})
# simply \code{\link{quote}} the design generating function with
lsd_quote <- quote(
  designLSD(
    treatments = 4,
    squares = n,
    reuse = "row",
    beta = c(10, 2, 3, 4),
```

```
    VarCov = list(5, 2),
    sigma2 = 10
  )
)

# find the minimum number of squares required to achieve the target power of 0.8
find_sample_size(lsd_quote)
```

`fit.pseu.model` *Create an artificial model object*

Description

Create a pseudo-model object with the response variable being simulated according to the fixed and random effects. Model coefficients are replaced by the expectations specified in the argument `beta`. Variance-covariance components of random effects are replaced by the values specified in argument `VarCov`. The standard deviation of random error is replaced by the argument `sigma`. Creating such a pseudo-model facilitates power calculations by leveraging the `anova` function in `lmerTest` and the `Anova` function in `car`.

Usage

```
fit.pseu.model(formula, data, beta, VarCov, sigma, ...)
```

Arguments

`formula`	an object of class `formula`
`data`	a data frame with the independent variables of the design as columns, e.g., treatment factors and block factors.
`beta`	a vector of the expectations of model coefficients.
`VarCov`	variance-covariance matrices. If there are multiple random effect groups, supply the variance-covariance matrix of each group as an element in a list.
`sigma`	standard deviation of error
`...`	other arguments passed to the `anova` function in `lmerTest`. The type of sum of squares, with Type III as the default, and the method for computing the denominator degrees of freedom, with Satterthwaite's method as the default, can be changed. For more details, see anova.lmerModLmerTest.

Value

a pseudo model object.

milk *An exemplary dataset of a 4x4 crossover design with 2 squares*

Description

Milk yield records from 8 cows over 4 different periods in a 4x4 crossover design. The design includes 2 Latin squares, each consisting of 4 cows and 4 periods.

Usage

```
milk
```

Format

A data frame with 32 rows and 4 variables:

Cow Factor: Cow index (8 levels)

Period Factor: Period index (4 levels)

Treatment Factor: Treatment index (4 levels)

MilkYield Numeric: milk yield recordings (in kg)

Source

Simulated data for package demonstration purposes.

pwr.anova *Power of omnibus test*

Description

Calculate power for testing overall effects of treatment factors and their interactions, i.e., statistical power of ANOVA.

Usage

```
pwr.anova(design, alpha = 0.05, ...)
```

Arguments

design	a design object created using design generating functions.
alpha	significance level (type I error rate), default 0.05
...	Additional arguments passed to anova.lmerModLmerTest for linear mixed models and to Anova for linear models. The type of sum of squares (SS, default is Type III) and the method for computing denominator degrees of freedom (DDF, default is Satterthwaite's method) can be modified. For balanced designs, types of SS and DDF do not affect results. Note that these additional arguments should be consistent in the design-generating function and pwr.anova for linear mixed models.

Value

a data frame with numerator degrees of freedom (NumDF), denominator degrees of freedom (DenDF), non-centrality parameter, type I error rate (alpha), and power.

See Also

designCRD(), designRCBD(), designLSD(), designCOD(), designSPD(), designCustom(), and pwr.contrast()

Examples

```
# generate an RCBD
rcbd = designRCBD(treatments = c(2, 2), blocks = 10, beta = c(10, 9, 8, 7), VarCov = 10, sigma2 = 9)
# power of omnibus test
pwr.anova(rcbd, alpha  = 0.05)
```

---

pwr.contrast *Power of contrasts*

Description

Calculate power for testing various contrasts. The same syntax of emmeans package is employed to specify contrast types.

Usage

```
pwr.contrast(design, specs, method, alpha = 0.05, ...)
```

Arguments

design	a design object created using design generating functions.
specs	an argument inherited from emmeans specifying the names of the factors over which the contrasts are performed.
method	an argument inherited from contrast specifying the method of contrasts, e.g., pairwise, linear, and polynomials.
alpha	significance level (type I error rate), default 0.05
...	other arguments passed to contrast.

Value

a data frame showing the power of the specific contrast

Examples

```
rcbd = designRCBD(treatments = c(2, 2), blocks = 10, beta = c(10, 9, 8, 7), VarCov = 10, sigma2 = 9)
pwr.contrast(rcbd, specs = ~ facA|facB, method = "pairwise")
```

`theta.names` *Naming theta Naming the vector in the order of model specification and in the actual order used in the model*

Description

Naming theta Naming the vector in the order of model specification and in the actual order used in the model

Usage

```
theta.names(data, formula)
```

Arguments

`data`	data frame
`formula`	model formula

Index