Package: experDesign (via r-universe)

August 19, 2024

Title Design Experiments for Batches

Version 0.4.0.9000

Description Distributes samples in batches while making batches homogeneous according to their description. Allows for an arbitrary number of variables, both numeric and categorical. For quality control it provides functions to subset a representative sample.

License MIT + file LICENSE

URL https://experdesign.llrs.dev, https://github.com/llrs/experDesign/

BugReports https://github.com/llrs/experDesign/issues

Depends R (>= 3.5.0)

Imports methods, stats, utils

Suggests covr, knitr, MASS, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://llrs.r-universe.dev

RemoteUrl https://github.com/llrs/experdesign

RemoteRef HEAD

RemoteSha 118ad3143a0ad769a5305ed5f70d3db0f3ddc5ca

Contents

experDesign-package 2
batch_names 3
check_data 4

check_index . . . 5
compare_index . . . 6
create_subset . . . 6
design . . . 7
distribution . . . 8
entropy . . . 9
evaluate_entropy . . . 9
evaluate_independence . . . 10
evaluate_index . . . 11
evaluate_mad . . . 12
evaluate_mean . . . 12
evaluate_na . . . 13
evaluate_orig . . . 14
evaluate_sd . . . 15
extreme_cases . . . 15
follow_up . . . 16
follow_up2 . . . 17
inspect . . . 18
optimum . . . 19
position_name . . . 20
qcSubset . . . 20
replicates . . . 21
spatial . . . 22
use_index . . . 23
valid_followup . . . 23

Index **25**

`experDesign-package` *experDesign: Expert experiment design in batches*

Description

Enables easy distribution of samples per batch avoiding batch and confounding effects by randomization of the variables in each batch.

Details

The most important function is `design()`, which distributes samples in batches according to the information provided.

To help in the bench there is the `inspect()` function that appends the group to the data provided.

If you have a grid or some spatial data, you might want to look at the `spatial()` function to distribute the samples while keeping the original design.

In case an experiment was half processed and you need to extend it you can use `follow_up()` or `follow_up2()`. It helps selecting which samples already used should be used in the follow up.

Author(s)

Lluís Revilla

See Also

Useful links:

- https://experdesign.llrs.dev
- https://github.com/llrs/experDesign/
- Report bugs at https://github.com/llrs/experDesign/issues

`batch_names` *Name the batch*

Description

Given an index return the name of the batches the samples are in

Usage

```
batch_names(i)
```

Arguments

`i`	A list of numeric indices.

Value

A character vector with the names of the batch for each the index.

See Also

`create_subset()`, for the inverse look at `use_index()`.

Examples

```
index <- create_subset(100, 50, 2)
batch <- batch_names(index)
head(batch)
```

check_data *Check experiment data*

Description

In order to run a successful experiment a good design is needed even before measuring the data. This functions checks several heuristics for a good experiment and warns if they are not found.

Usage

```
check_data(pheno, omit = NULL, na.omit = FALSE)
```

Arguments

pheno	Data.frame with the variables of each sample, one row one sample.
omit	Character vector with the names of the columns to omit.
na.omit	Check the effects of missing values too.

Value

A logical value indicating if everything is alright (TRUE or not (FALSE).

See Also

valid_followup().

Examples

```
rdata <- expand.grid(sex = c("M", "F"), class = c("lower", "median", "high"))
rdata2 <- rbind(rdata, rdata)
check_data(rdata2)

#Different warnings
check_data(rdata)
check_data(rdata[-c(1, 3), ])
data(survey, package = "MASS")
check_data(survey)
```

check_index *Check index distribution on batches*

Description

Report the statistics for each subset and variable compared to the original.

Usage

```
check_index(pheno, index, omit = NULL)
```

Arguments

pheno	Data.frame with the sample information.
index	A list of indices indicating which samples go to which subset.
omit	Name of the columns of the pheno that will be omitted.

Details

The closer the values are to 0, the less difference is with the original distribution, so it is a better randomization.

Value

A matrix with the differences with the original data.

See Also

Functions that create an index `design()`, `replicates()`, `spatial()`. See also `create_subset()` for a random index.

Examples

```
index <- create_subset(50, 24)
metadata <- expand.grid(height = seq(60, 80, 5), weight = seq(100, 300, 50),
                        sex = c("Male","Female"))
check_index(metadata, index)
```

compare_index *Compares two indexes*

Description

Compare the distribution of samples with two different batches.

Usage

```
compare_index(pheno, index1, index2)
```

Arguments

pheno	A data.frame of the samples with the characteristics to normalize.
index1, index2	A list with the index for each sample, the name of the column in pheno with the batch subset or the character .

Value

A matrix with the variables and the columns of of each batch. Negative values indicate index1 was better.

See Also

check_index()

Examples

```
index1 <- create_subset(50, 24)
index2 <- batch_names(create_subset(50, 24))
metadata <- expand.grid(height = seq(60, 80, 5), weight = seq(100, 300, 50),
                        sex = c("Male","Female"))
compare_index(metadata, index1, index2)
```

create_subset *Create index of subsets of a data*

Description

Index of the samples grouped by batches.

Usage

```
create_subset(size_data, size_subset = NULL, n = NULL, name = "SubSet")
```

Arguments

size_data	A numeric value of the amount of samples to distribute.
size_subset	A numeric value with the amount of samples per batch.
n	A numeric value with the number of batches.
name	A character used to name the subsets, either a single one or a vector the same size as n.

Value

A random list of indices of the samples.

See Also

batch_names(), use_index() if you already have a factor to be used as index.

Examples

```
index <- create_subset(100, 50, 2)
```

---

design *Design a batch experiment*

Description

Given some samples it distribute them in several batches, trying to have equal number of samples per batch. It can handle both numeric and categorical data.

Usage

```
design(pheno, size_subset, omit = NULL, iterations = 500, name = "SubSet")
```

Arguments

pheno	Data.frame with the sample information.
size_subset	Numeric value of the number of sample per batch.
omit	Name of the columns of the pheno that will be omitted.
iterations	Numeric value of iterations that will be performed.
name	A character used to name the subsets, either a single one or a vector the same size as n.

Value

The indices of which samples go with which batch.

See Also

The `evaluate_*` functions and `create_subset()`.

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
index
```

`distribution` *Distribution by batch*

Description

Checks if all the values are maximally distributed in the several batches. Aimed for categorical variables.

Usage

```
distribution(report, column)
```

Arguments

`report`	A data.frame which must contain a batch column. Which can be obtained with `inspect()`.
`column`	The name of the column one wants to inspect.

Value

`TRUE` if the values are maximal distributed, otherwise `FALSE`.

Examples

```
data(survey, package = "MASS")
columns <- c("Sex", "Age", "Smoke")
nas <- c(137, 70) # Omit rows with NA to avoid warnings in design
index <- design(pheno = survey[-nas, columns], size_subset = 70,
                iterations = 10)
batches <- inspect(index, survey[-nas, columns])
distribution(batches, "Sex")
distribution(batches, "Smoke")
```

entropy *Calculates the entropy*

Description

Calculates the entropy of a category. It uses the amount of categories to scale between 0 and 1.

Usage

```
entropy(x)
```

Arguments

x	A character or vector with two or more categories

Value

The numeric value of the Shannon entropy scaled between 0 and 1.

Note

It omits the NA if present.

Examples

```
entropy(c("H", "T", "H", "T"))
entropy(c("H", "T", "H", "T", "H", "H", "H"))
entropy(c("H", "T", "H", "T", "H", "H", NA))
entropy(c("H", "T", "H", "T", "H", "H"))
entropy(c("H", "H", "H", "H", "H", "H", NA))
```

evaluate_entropy *Evaluate entropy*

Description

Looks if the nominal or character columns are equally distributed according to the entropy and taking into account the independence between batches. If any column is different in each row it is assumed to be the sample names and thus omitted.

Usage

```
evaluate_entropy(i, pheno)
```

Arguments

i	list of numeric indices of the data.frame
pheno	Data.frame with information about the samples

Value

Value to minimize

See Also

Other functions to evaluate samples: evaluate_independence(), evaluate_index(), evaluate_mad(), evaluate_mean(), evaluate_na(), evaluate_orig(), evaluate_sd()

Other functions to evaluate categories: evaluate_independence(), evaluate_na()

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
# Note that numeric columns will be omitted:
evaluate_entropy(index, survey[, c("Sex", "Smoke", "Age")])
```

---

evaluate_independence *Compare independence by chisq.test*

Description

Looks the independence between the categories and the batches.

Usage

```
evaluate_independence(i, pheno)
```

Arguments

i	Index of subsets.
pheno	A data.frame with the information about the samples.

Value

Returns a vector with the p-values of the chisq.test between the category and the subset.

See Also

Other functions to evaluate samples: evaluate_entropy(), evaluate_index(), evaluate_mad(), evaluate_mean(), evaluate_na(), evaluate_orig(), evaluate_sd()

Other functions to evaluate categories: evaluate_entropy(), evaluate_na()

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
# Note that numeric columns will be omitted:
evaluate_independence(index, survey[, c("Sex", "Smoke", "Age")])
```

`evaluate_index` *Evaluates a data.frame*

Description

Measures several indicators per group

Usage

```
evaluate_index(i, pheno)
```

Arguments

i	Index
pheno	Data.frame with information about the samples

Value

An array of three dimensions with the mean, standard deviation (`sd()`), and median absolute deviation (`mad()`) of the numeric variables, the entropy of the categorical and the number of `NA` by each subgroup.

See Also

If you have already an index you can use `use_index()`.

Other functions to evaluate samples: `evaluate_entropy()`, `evaluate_independence()`, `evaluate_mad()`, `evaluate_mean()`, `evaluate_na()`, `evaluate_orig()`, `evaluate_sd()`

Examples

```
data(survey, package = "MASS")
index <- create_subset(nrow(survey), 50, 5)
ev_index <- evaluate_index(index, survey[, c("Sex", "Smoke")])
ev_index["entropy", , ]
```

evaluate_mad *Evaluate median absolute deviation*

Description

Looks for the median absolute deviation values in each subgroup.

Usage

```
evaluate_mad(i, pheno)
```

Arguments

i	List of indices
pheno	Data.frame with information about the samples

Value

A vector with the mean difference between the median absolute deviation of each group and the original mad.

See Also

Other functions to evaluate samples: evaluate_entropy(), evaluate_independence(), evaluate_index(), evaluate_mean(), evaluate_na(), evaluate_orig(), evaluate_sd()

Other functions to evaluate numbers: evaluate_mean(), evaluate_na(), evaluate_sd()

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
# Note that categorical columns will be omitted:
evaluate_mad(index, survey[, c("Sex", "Smoke", "Age")])
```

evaluate_mean *Evaluates the mean of the numeric values*

Description

Looks for the mean of the numeric values

Usage

```
evaluate_mean(i, pheno)
```

Arguments

i	List of indices
pheno	Data.frame with information about the samples

Value

A matrix with the mean value for each column for each subset

See Also

Other functions to evaluate samples: evaluate_entropy(), evaluate_independence(), evaluate_index(), evaluate_mad(), evaluate_na(), evaluate_orig(), evaluate_sd()

Other functions to evaluate numbers: evaluate_mad(), evaluate_na(), evaluate_sd()

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
# Note that categorical columns will be omitted:
evaluate_mean(index, survey[, c("Sex", "Smoke", "Age")])
```

---

evaluate_na *Evaluate the dispersion of NAs*

Description

Looks how are NA distributed in each subset

Usage

```
evaluate_na(i, pheno)
```

Arguments

i	list of numeric indices of the data.frame
pheno	Data.frame

Value

The optimum value to reduce

See Also

Other functions to evaluate samples: evaluate_entropy(), evaluate_independence(), evaluate_index(), evaluate_mad(), evaluate_mean(), evaluate_orig(), evaluate_sd()

Other functions to evaluate categories: evaluate_entropy(), evaluate_independence()

Other functions to evaluate numbers: evaluate_mad(), evaluate_mean(), evaluate_sd()

Examples

```
samples <- 10
m <- matrix(rnorm(samples), nrow = samples)
m[sample(seq_len(samples), size = 5), ] <- NA # Some NA
i <- create_subset(samples, 3, 4) # random subsets
evaluate_na(i, m)
```

`evaluate_orig` *Evaluate each variable provided*

Description

Measure some summary statistics of the whole cohort of samples

Usage

```
evaluate_orig(pheno)
```

Arguments

pheno	Data.frame with information about the samples

Value

A matrix with the mean, standard deviation, MAD values of the numeric variables, the entropy of the categorical, and the amount of NA per variable.

See Also

Other functions to evaluate samples: `evaluate_entropy()`, `evaluate_independence()`, `evaluate_index()`, `evaluate_mad()`, `evaluate_mean()`, `evaluate_na()`, `evaluate_sd()`

Examples

```
data(survey, package = "MASS")
evaluate_orig(survey[, c("Sex", "Age", "Smoke")])
```

evaluate_sd *Evaluates the mean of the numeric values*

Description

Looks for the standard deviation of the numeric values

Usage

```
evaluate_sd(i, pheno)
```

Arguments

i	List of indices
pheno	Data.frame with the samples

Value

A matrix with the standard deviation value for each column for each subset

See Also

Other functions to evaluate samples: evaluate_entropy(), evaluate_independence(), evaluate_index(), evaluate_mad(), evaluate_mean(), evaluate_na(), evaluate_orig()

Other functions to evaluate numbers: evaluate_mad(), evaluate_mean(), evaluate_na()

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
# Note that categorical columns will be omitted:
evaluate_sd(index, survey[, c("Sex", "Smoke", "Age")])
```

extreme_cases *Select the subset of extreme cases to evaluation*

Description

Subset some samples that are mostly different.

Usage

```
extreme_cases(pheno, size, omit = NULL, iterations = 500)
```

Arguments

pheno	Data.frame with the sample information.
size	The number of samples to subset.
omit	Name of the columns of the pheno that will be omitted.
iterations	Numeric value of iterations that will be performed.

Value

A vector with the number of the rows that are selected.

See Also

optimum()

Examples

```
metadata <- expand.grid(height = seq(60, 80, 5), weight = seq(100, 300, 50),
 sex = c("Male","Female"))
sel <- extreme_cases(metadata, 10)
# We can see that it selected both Female and Males and wide range of height
# and weight:
metadata[sel, ]
```

---

follow_up *Follow up experiments*

Description

If an experiment was carried out with some samples and you want to continue with some other samples later on.

Usage

```
follow_up(
  original,
  follow_up,
  size_subset,
  omit = NULL,
  old_new = "batch",
  iterations = 500
)
```

Arguments

original	A data.frame with the information of the samples used originally.
follow_up	A data.frame with the information of the new samples.
size_subset	Numeric value of the number of sample per batch.
omit	Name of the columns of the pheno that will be omitted.
old_new	Name of the column where the batch status will be stored. If it matches the name of a column in original it will be used to find previous batches.
iterations	Numeric value of iterations that will be performed.

Value

A data.frame with the common columns of data, a new column old_new, and a batch column filled with the new batches needed.

See Also

follow_up2()

Examples

```
data(survey, package = "MASS")
survey1 <- survey[1:118, ]
survey2 <- survey[119:nrow(survey), ]
folu <- follow_up(survey1, survey2, size_subset = 50, iterations = 10)
```

---

follow_up2 *Follow up experiments in batches*

Description

Design experiment with all the data new and old together.

Usage

```
follow_up2(all_data, batch_column = "batch", ...)
```

Arguments

all_data	A data.frame with all the data about the samples. Each row is a sample.
batch_column	The name of the column of all_data with the batches used. If NA it is interpreted as a new data, if not empty it is considered a batch.
...	Arguments passed on to design size_subset Numeric value of the number of sample per batch. omit Name of the columns of the pheno that will be omitted. iterations Numeric value of iterations that will be performed. name A character used to name the subsets, either a single one or a vector the same size as n.

Details

If the `batch_column` is empty the samples are considered new. If the `size_subset` is missing, it will be estimated from the previous batch Similarly, iterations and name will be guessed or inferred from the samples.

Value

A `data.frame` with the `batch_column` filled with the new batches needed.

See Also

`follow_up()`

Examples

```
data(survey, package = "MASS")
# Create the first batch
first_batch_n <- 118
variables <- c("Sex", "Smoke", "Age")
survey1 <- survey[seq_len(first_batch_n), variables]
index1 <- design(survey1, size_subset = 50, iterations = 10)
r_survey <- inspect(index1, survey1)
# Create the second batch with "new" students
survey2 <- survey[seq(from = first_batch_n +1, to = nrow(survey)), variables]
survey2$batch <- NA
# Prepare the follow up
all_classroom <- rbind(r_survey, survey2)
follow_up2(all_classroom, size_subset = 50, iterations = 10)
```

`inspect` *Inspect the index*

Description

Given the index and the data of the samples append the batch assignment

Usage

```
inspect(i, pheno, omit = NULL, index_name = "batch")
```

Arguments

`i`	List of indices of samples per batch
`pheno`	Data.frame with the sample information.
`omit`	Name of the columns of the pheno that will be omitted.
`index_name`	Column name of the index of the resulting data.frame.

Value

The data.frame with a new column batch with the name of the batch the sample goes to.

Examples

```
data(survey, package = "MASS")
columns <- c("Sex", "Age", "Smoke")
index <- design(pheno = survey[, columns], size_subset = 70,
                iterations = 10)
batches <- inspect(index, survey[, columns])
head(batches)
```

`optimum` *Optimum values for batches*

Description

Calculates the optimum values for number of batches or size of the batches. If you need to do several batches it can be better to distribute it evenly and add replicates.

Usage

```
optimum_batches(size_data, size_subset)

optimum_subset(size_data, batches)

sizes_batches(size_data, size_subset, batches)
```

Arguments

`size_data`	A numeric value of the number of samples to use.
`size_subset`	Numeric value of the number of sample per batch.
`batches`	A numeric value of the number of batches.

Value

`optimum_batches` A numeric value with the number of batches to use.

`optimum_subset` A numeric value with the maximum number of samples per batch of the data.

`sizes_batches` A numeric vector with the number of samples in each batch.

Examples

```
size_data <- 50
size_batch <- 24
(batches <- optimum_batches(size_data, size_batch))
# So now the best number of samples for each batch is less than the available
(size <- optimum_subset(size_data, batches))
# The distribution of samples per batch
sizes_batches(size_data, size, batches)
```

position_name *Create position names for a grid.*

Description

Create position names for a grid.

Usage

```
position_name(rows, columns)
```

Arguments

rows	Names of the rows.
columns	Names of the columns.

Value

A data.frame with the rows and columns and the resulting name row+column. The name column is a factor for easier sorting in row, column order.

Examples

```
position_name(c("A", "B"), 1:2)
```

qcSubset *Random subset*

Description

Select randomly some samples from an index

Usage

```
qcSubset(index, size, each = FALSE)
```

Arguments

index	A list of indices indicating which samples go to which subset.
size	The number of samples that should be taken.
each	A logical value if the subset should be taken from all the samples or for each batch.

Examples

```
set.seed(50)
index <- create_subset(100, 50, 2)
QC_samples <- qcSubset(index, 10)
QC_samplesBatch <- qcSubset(index, 10, TRUE)
```

`replicates` *Design a batch experiment with experimental controls*

Description

To ensure that the batches are comparable some samples are processed in each batch. This function allows to take into account that effect. It uses the most different samples as controls as defined with `extreme_cases()`.

Usage

```
replicates(pheno, size_subset, controls, omit = NULL, iterations = 500)
```

Arguments

`pheno`	Data.frame with the sample information.
`size_subset`	Numeric value of the number of sample per batch.
`controls`	The numeric value of the amount of technical controls per batch.
`omit`	Name of the columns of the pheno that will be omitted.
`iterations`	Numeric value of iterations that will be performed.

Details

To control for variance replicates are important, see for example https://www.nature.com/articles/nmeth.3091.

Value

A index with some samples duplicated in the batches.

See Also

`design()`, `extreme_cases()`.

Examples

```
samples <- data.frame(L = letters[1:25], Age = rnorm(25),
                      type = sample(LETTERS[1:5], 25, TRUE))
index <- replicates(samples, 5, controls = 2, omit = "L", iterations = 10)
head(index)
```

spatial *Distribute the sample on the plate*

Description

This function assumes that to process the batch the samples are distributed in a plate with a grid scheme.

Usage

```
spatial(
  index,
  pheno,
  omit = NULL,
  remove_positions = NULL,
  rows = LETTERS[1:5],
  columns = 1:10,
  iterations = 500
)
```

Arguments

index	A list with the samples on each subgroup, as provided from design() or replicates().
pheno	Data.frame with the sample information.
omit	Name of the columns of the pheno that will be omitted.
remove_positions	Character, name of positions to be avoided in the grid.
rows	Character, name of the rows to be used.
columns	Character, name of the rows to be used.
iterations	Numeric value of iterations that will be performed.

Value

The indices of which samples go with which batch.

Examples

```
data(survey, package = "MASS")
index <- design(survey[, c("Sex", "Smoke", "Age")], size_subset = 50,
                iterations = 10)
index2 <- spatial(index, survey[, c("Sex", "Smoke", "Age")], iterations = 10)
head(index2)
```

use_index *Convert a factor to index*

Description

Convert a given factor to an accepted index

Usage

```
use_index(x)
```

Arguments

x	A character or a factor to be used as index

See Also

You can use `evaluate_index()` to evaluate how good an index is. For the inverse look at `batch_names()`.

Examples

```
plates <- c("P1", "P2", "P1", "P2", "P2", "P3", "P1", "P3", "P1", "P1")
use_index(plates)
```

valid_followup *Check the data for a follow up experiment.*

Description

Sometimes some samples are collected and analyzed, later another batch of samples is analyzed. This function tries to detect if there are problems with the data or when the data is combined in a single analysis. To know specific problems with the data you need to use check_data()

Usage

```
valid_followup(
  old_data = NULL,
  new_data = NULL,
  all_data = NULL,
  omit = NULL,
  column = "batch"
)
```

Arguments

old_data, new_data	A data.frame with the old and new data respectively.
all_data	A data.frame with all the data about the samples. Each row is a sample.
omit	Name of the columns of the pheno that will be omitted.
column	The name of the column where the old data has the batch information, or whether the data is new or not (NA) in the case of all_data.

Value

Called by its side effects of warnings, but returns a logical value if there are some issues (FALSE) or not (TRUE)

See Also

check_data()

Examples

```
data(survey, package = "MASS")
survey1 <- survey[1:118, ]
survey2 <- survey[119:nrow(survey), ]
valid_followup(survey1, survey2)
survey$batch <- NA
survey$batch[1:118]  <- "old"
valid_followup(all_data = survey)
```

Index

∗ **functions to evaluate categories**
evaluate_entropy, 9
evaluate_independence, 10
evaluate_na, 13
∗ **functions to evaluate numbers**
evaluate_mad, 12
evaluate_mean, 12
evaluate_na, 13
evaluate_sd, 15
∗ **functions to evaluate samples**
evaluate_entropy, 9
evaluate_independence, 10
evaluate_index, 11
evaluate_mad, 12
evaluate_mean, 12
evaluate_na, 13
evaluate_orig, 14
evaluate_sd, 15

batch_names, 3
batch_names(), *7*, *23*

check_data, 4
check_data(), *24*
check_index, 5
check_index(), *6*
compare_index, 6
create_subset, 6
create_subset(), *3*, *5*, *8*

design, 7, *17*
design(), *2*, *5*, *21, 22*
distribution, 8

entropy, 9
evaluate_entropy, 9, *10–15*
evaluate_independence, *10*, 10, *11–15*
evaluate_index, *10*, 11, *12–15*
evaluate_index(), *23*
evaluate_mad, *10, 11*, 12, *13–15*
evaluate_mean, *10–12*, 12, *13–15*
evaluate_na, *10–13*, 13, *14, 15*
evaluate_orig, *10–13*, 14, *15*
evaluate_sd, *10–14*, 15
experDesign *(*experDesign-package*)*, 2
experDesign-package, 2
extreme_cases, 15
extreme_cases(), *21*

follow_up, 16
follow_up(), *2*, *18*
follow_up2, 17
follow_up2(), *2*, *17*

inspect, 18
inspect(), *2*, *8*

mad(), *11*

optimum, 19
optimum(), *16*
optimum_batches *(*optimum*)*, 19
optimum_subset *(*optimum*)*, 19

position_name, 20

qcSubset, 20

replicates, 21
replicates(), *5*, *22*

sd(), *11*
sizes_batches *(*optimum*)*, 19
spatial, 22
spatial(), *2*, *5*

use_index, 23
use_index(), *3*, *7*, *11*

valid_followup, 23
valid_followup(), *4*