Package: covadap (via r-universe)

June 9, 2026

Type Package

Date 2023-12-06

Title Implement Covariate-Adaptive Randomization

Version 1.0.1

Author Rosamarie Frieri [aut, cre], Marco Novelli [aut]

Maintainer Rosamarie Frieri <rosamarie.frieri2@unibo.it>

Imports stats

Description Implementing seven Covariate-Adaptive Randomization to assign patients to two treatments. Three of these procedures can also accommodate quantitative and mixed covariates. Given a set of covariates, the user can generate a single sequence of allocations or replicate the design multiple times by simulating the patients' covariate profiles. At the end, an extensive assessment of the performance of the randomization procedures is provided, calculating several imbalance measures. See Baldi Antognini A, Frieri R, Zagoraiou M and Novelli M (2022) <doi:10.1007/s00362-022-01381-1> for details.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation no

Repository https://rfrieri91-sys.r-universe.dev

Date/Publication 2023-12-07 02:39:56 UTC

RemoteUrl https://github.com/cran/covadap

RemoteRef HEAD

RemoteSha d2fc52d63fe8f1fb3a33333ce9f4a039fc834d88

Contents

---

covadap-package *covadap: Implements Covariate-Adaptive Randomization procedures*

Description

Implementing seven Covariate-Adaptive Randomization to assign patients to two treatments. Three of these procedures can also accommodate quantitative and mixed covariates. Given a set of covariates, the user can generate a single sequence of allocations or replicate the design multiple times by simulating the patients' covariate profiles. At the end, an extensive assessment of the performance of the randomization procedures is provided, calculating several imbalance measures.

Acknowledgement

This work was supported by the EU funding within the NextGenerationEU PRIN2022 *Optimal and adaptive designs for modern medical experimentation* (2022TRB44L).

Author(s)

R. Frieri <rosamarie.frieri2@unibo.it>, M. Novelli <m.novelli@unibo.it>

References

Atkinson A C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

Baldi Antognini A, Frieri R, Zagoraiou M, Novelli M. *The Efficient Covariate-Adaptive Design for high-order balancing of quantitative and qualitative covariates*. Statistical Papers, 2022.

Baldi Antognini A and Zagoraiou M. *The covariate-adaptive biased coin design for balancing clinical trials in the presence of prognostic factors*. Biometrika, 2011, 98(3): 519-535.

Efron B, *Forcing a sequential experiment to be balanced*. Biometrika, 1971, 58(3): 403-418.

Hu Y, Hu F. *Asymptotic properties of covariate-adaptive randomization*. The Annals of Statistics, 2012, 40(3): 1974-1815.

Ma Z and Hu F. *Balancing continuous covariates based on Kernel densities*. Contemporary Clinical Trials, 2013, 34(2): 262-269.

Pocock S J, Simon R. *Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial*. Biometrics, 1975, 31(1): 103-115.

Soares F, Wu CFJ. *Some restricted randomization rules in sequential designs*. Communications in Statistics Theory and Methods 1983, 12: 2017-2034.

See Also

CABCD, HuHu, PocSim, BSD, DABCD, ECADE, KER.

BSD *Big Stick Design*

Description

Implements the Big Stick Design by Soares and Wu (1963) for assigning patients to two treatments A and B. The procedure works with qualitative covariates only.

Usage

```
BSD(data, bound = 3, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. Each row of data corresponds to the covariate profile of a patient.
bound	integer parameter representing the maximum tolerated imbalance. The default value is set to 3.
print.results	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B with the Big Stick Design as described in Soares and Wu (1983).

The argument bound is the maximum tolerated imbalance that the experiment can accept: complete randomization is used as long as the imbalance of the treatment allocation does not exceed bound. When the imbalance reaches the value set in bound, a deterministic assignment is made to lower the imbalance.

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B. The strata imbalances measures report, for each stratum, the total number of patients assigned (N.strata), the number of patients assigned to A (A.strata) and the within-stratum imbalance (D.strata), calculated as 2*A.strata-N.strata. The within-covariate imbalances report, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B. See also Value.

Value

It returns an object of class "covadap", which is a list containing the following elements:

summary.info — Design name of the design,
Sample_size number of patients,
n_cov number of covariates,
n_levels number of levels of each covariate,
var_names name of covariates and levels,
parameter_a design parameter (see above).

Assignments — a vector with the treatment assignments.

Imbalances.summary — summary of overall imbalance measures at the end of the study (Loss loss, Mahal Mahalanobis distance, overal.imb difference in the total number of patients assigned to A and B).

Strata.measures — a data frame containing for each possiblue stratum the corresponding imbalances: N.strata is the total number of patients assigned to the stratum; A.strata is the total number of patients assigned to A within the stratum; D.strata is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum.

Imbalances — a list containing all the imbalance measures:
Imb.measures (Loss loss, Mahal Mahalanobis distance),
Overall.imb difference in the total number of patients assigned to A and B,
Within.strata within-stratum imbalance for all strata,
Within.cov within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate.

data — the data provided in input.

observed.strata — a data frame with all the observed strata.

References

Soares F, Wu CFJ. *Some restricted randomization rules in sequential designs*. Communications in Statistics Theory and Methods 1963, 12: 2017-2034.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

See Also as BSD.sim for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)

# Create a sample dataset
```

```
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
BSD(data = df1, bound = 3, print.results = TRUE)
# To view a summary and create a list containing all the metrics of the design
res <- BSD(data = df1, bound = 3, print.results = TRUE)
res
```

BSD.sim *Simulations of the Big Stick Design*

Description

Implements the Big Stick Design by Soares and Wu (1963) for assigning patients to two treatments A and B by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative covariates only.

Usage

```
#With existing dataframe
BSD.sim(data, covar = NULL, n = NULL, bound = 3, nrep = 1000,
        print.results = TRUE)
#With covariates
BSD.sim(data = NULL, covar, n, bound = 3, nrep = 1000,
        print.results = TRUE)
```

Arguments

data	a data frame or a matrix. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.
covar	either a vector or a list to be specified only if data = NULL. It could be a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise it is a list containing the covariates with their levels (e.g. one covariate with two levels and one with three covar = list(cov1 = c("lev1", "lev2"), cov2 = c("lev1", "lev2", "lev3")).
n	number of patients (to be specified only if data = NULL).
bound	integer parameter representing the maximum tolerated imbalance. The default value is set to 3.
nrep	number of trial replications.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the Big Stick Design by Soares and Wu (1983) (see BSD).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covar requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications. See also BSD.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design, Sample_size number of patients, n_cov number of covariates, n_levels number of levels of each qualitative covariate, var_names name of the covariates, n.rep number of replications, Maximum_tolerated_imbalance for the BSD procedure.
Imbalances	a list with the imbalance measures at the end of each simulated trial: Imb.measures summary of overall imbalances (Loss loss, Mahal Mahalanobis distance, overall.imb difference in the total number of patients assigned to A and B), within.imb within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate, strata.imb the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum), strata.A total number of patients assigned to A within the stratum, strata.N total number of patients assigned to each stratum, obs.strata matrix of the possible strata.
out	For each replication returns a list of the data provided in input (data) and the resulting assignments (Assignment)

References

Soares F, Wu CFJ. *Some restricted randomization rules in sequential designs*. Communications in Statistics Theory and Methods 1963, 12: 2017-2034.

See Also

See Also BSD.

Examples

```
require(covadap)
# Here we set nrep = 100 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.


### With existing dataframe
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# Simulate the design
res1 <- BSD.sim(data = df1, covar = NULL, n = NULL, bound = 3, nrep = 100,
                print.results = TRUE)
### With covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- BSD.sim(data = NULL, covar = c(2,2,3), n = 100, bound = 3,  nrep = 100,
                print.results = TRUE)
```

CABCD *Covariate-Adjusted Biased Coin Design*

Description

Implements the Covariate-adjusted Biased Coin Design by Baldi Antognini and Zagoraiou (2011), a stratified randomization procedure for two treatments A and B. The procedure works with qualitative covariates only.

Usage

```
CABCD(data, a = 3, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. Each row of data corresponds to the covariate profile of a patient.
a	(non-negative) design parameter determining the degree of randomness: $a = 0$ gives the completely randomized design; $a \to \infty$ gives a deterministic design. The default value is set to 3.
print.results	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B as described in Baldi Antognini and Zagoraiou (2011).

The parameter a determines the degree of randomness of the procedure.

At the end of the study, the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B. The strata imbalances measures report, for each stratum, the total number of patients assigned (`N.strata`), the number of patients assigned to A (`A.strata`) and the within-stratum imbalance (`D.strata`), calculated as `2*A.strata-N.strata`. The within-covariate imbalances report, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B. See also Value.

Value

It returns an object of `class` `"covadap"`, which is a list containing the following elements:

`summary.info` `Design` name of the design,
`Sample_size` number of patients,
`n_cov` number of covariates,
`n_levels` number of levels of each covariate,
`var_names` name of covariates and levels,
`parameter_a` design parameter (see above).

`Assignments` a vector with the treatment assignments.

`Imbalances.summary` summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Strata.measures` a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum.

`Imbalances` a list containing all the imbalance measures:
`Imb.measures` (`Loss` loss, `Mahal` Mahalanobis distance),
`Overall.imb` difference in the total number of patients assigned to A and B,
`Within.strata` within-stratum imbalance for all strata,
`Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate.

`data` the data provided in input.

`observed.strata` a data frame with all the observed strata.

References

Baldi Antognini A and Zagoraiou M. *The covariate-adaptive biased coin design for balancing clinical trials in the presence of prognostic factors*. Biometrika, 2011, 98(3): 519-535.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

CABCD.sim for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)

# Create a sample dataset
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
CABCD(data = df1, a = 3)
# To view a summary
# and create a list containing all the metrics of the design
res <- CABCD(data = df1, a = 3)
res
```

CABCD.sim *Simulations of the Covariate-Adjusted Biased Coin Design*

Description

Implements the Covariate-adjusted Biased Coin Design by Baldi Antognini and Zagoraiou (2011) by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative covariates only.

Usage

```
#With existing dataframe
CABCD.sim(data, covar = NULL, n = NULL, a = 3, nrep = 1000,
          print.results = TRUE)
#With covariates
CABCD.sim(data = NULL, covar, n, a = 3, nrep = 1000,
          print.results = TRUE)
```

Arguments

data	a data frame or a matrix. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.
covar	either a vector or a list to be specified only if data = NULL. It could be a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise it is a list containing the covariates with their levels (e.g. one covariate with two levels and one with three covar = list(cov1 = c("lev1", "lev2"), cov2 = c("lev1", "lev2", "lev3")).

n	number of patients (to be specified only if data = NULL).
a	(non-negative) design parameter determining the degree of randomness: $a = 0$ gives the completely randomized design; $a \to \infty$ gives a deterministic design. The default value is set to 3.
nrep	number of trial replications.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the Covariate-Adjusted Biased Coin Design (see CABCD).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covat requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications. See also CABCD.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design, Sample_size number of patients, n_cov number of covariates, n_levels number of levels of each qualitative covariate, var_names name of the covariates, n.rep number of replications, parameter_a design parameter (see above).
Imbalances	a list with the imbalance measures at the end of each simulated trial: Imb.measures summary of overall imbalances (Loss loss, Mahal Mahalanobis distance, overall.imb difference in the total number of patients assigned to A and B), within.imb within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate, strata.imb the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum), strata.A total number of patients assigned to A within the stratum, strata.N total number of patients assigned to each stratum, obs.strata matrix of the possible strata.
out	For each replication returns a list of the data provided in input (data) and the resulting assignments (Assignment).

References

Baldi Antognini A and Zagoraiou M. *The covariate-adaptive biased coin design for balancing clinical trials in the presence of prognostic factors*. Biometrika, 2011, 98(3): 519-535.

See Also

See Also CABCD.

Examples

```
require(covadap)
# Here we set nrep = 100 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.


### With existing dataframe
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                  stringsAsFactors = TRUE)
# Simulate the design
res1 <- CABCD.sim(data = df1, n = NULL, a = 3, nrep = 100)
### With covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- CABCD.sim(covar = c(2,2,3), n = 100, a = 3,  nrep = 100)
```

DABCD *D_A-optimum biased coin design*

Description

Implements the D_A-optimum BCD by A. Atkinson (1982) for assigning patients to two treatments A and B in order to minimize the variance of the estimated treatment difference sequentially. The procedure works with qualitative and quantitative covariates.

Usage

```
DABCD(data, all.cat, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. It can be a matrix only when all.cat = TRUE. Each row of data corresponds to the covariate profile of a patient.
all.cat	logical. If all the covariates in data are qualitative must be set equal to TRUE, otherwise must be set equal to FALSE.
print.results	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B with the D_A-optimum BCD as described in Atkinson (1982).

This randomization procedure can be used when `data` contains only qualitative covariate, in this case set `all.cat = TRUE`, when `data` contains only quantitative covariates or when covariates of mixed nature are present, in these two latter cases set `all.cat = FALSE`. The function's output is slighly different according to these three scenarios as described in Value.

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B.

Only when `all.cat = TRUE`, the function returns the strata imbalances measures, that report, for each stratum, the total number of patients assigned (`N.strata`), the number of patients assigned to A (`A.strata`) and the within-stratum imbalance (`D.strata`), calculated as `2*A.strata-N.strata`.

If at least one qualitative covariate is present, the function returns the within-covariate imbalances reporting, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B.

If at least one quantitative covariate is present, the function returns the difference in means. For each quantitative covariate, is reported the difference in the mean in group A and B.

See Value for more details.

Value

It returns an object of `class "covadap"`, which is a list containing the following elements:

`summary.info` `Design` name of the design.

`Sample_size` number of patients.

`n_cov` number of covariates.

`n_categorical_variables` number of levels of each covariate. Is `NULL` if `all.cat = TRUE` or only quantitative covariates are present).

`n_levels` number of levels of each qualitative covariate. Is `NULL` if only quantitative covariates are present.

`var_names` name of the covariates.

`cov_levels_names` levels of each qualitative covariate. Is `NULL` if only quantitative covariates are present.

`n_quantitative_variables` number of quantitative covariates. Is `NULL` if `all.cat = TRUE`.

`Assignments` a vector with the treatment assignments.

`Imbalances.summary` summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Strata.measures` (only if `all.cat = TRUE`) a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the

stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum).

`Imbalances` a list containing all the imbalance measures.

`Imb.measures` summary of overall imbalances (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Within.strata` (only if `all.cat = TRUE`) within-stratum imbalance for all strata.

`Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is `NULL` if only quantitative covariates are present).

`data` the data provided in input.

`diff_mean` (only if `all.cat = FALSE`) the difference in mean of the quantitative covariates in group A and B.

`observed.strata`
(only if `all.cat = TRUE`) a data frame with all the observed strata.

References

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

See Also as `DABCD.sim` to for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)

### Implement with qualitative covariates (set all.cat = TRUE)
# Create a sample dataset with qualitative covariates
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
DABCD(data = df1,  all.cat = TRUE, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res1 <- DABCD(data = df1, all.cat = TRUE, print.results = TRUE)
res1


### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                 "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                  TRUE),
```

```
                  "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                  "cholesterol" = round(rnorm(100, 200, 8),1),
                  "height" = rpois(100,160),
                   stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
DABCD(data = ff1[,5:6],  all.cat = FALSE, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res2 <- DABCD(data = ff1[,5:6], all.cat = FALSE, print.results = TRUE)
res2

### With mixed covariates (set all.cat = FALSE)
# To just view a summary of the metrics of the design
DABCD(data = ff1,  all.cat = FALSE, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res3 <- DABCD(data = ff1, all.cat = FALSE, print.results = TRUE)
res3
```

DABCD.sim *Simulations of the D_A-optimum biased coin design*

Description

Implements the D_A-optimum biased coin design BCD by A. Atkinson (1982) for assigning patients to two treatments A and B in order to minimize the variance of the estimated treatment difference sequentially by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative and quantitative covariates.

Usage

```
#With existing dataframe provided in data
DABCD.sim(data, covar = NULL,  n = NULL, all.cat, nrep = 1000,
          print.results = TRUE)
#With covariates
DABCD.sim(data = NULL, covar, n, all.cat, nrep = 1000,
          print.results = TRUE)
```

Arguments

data	a data frame or a matrix. It can be a matrix only when all.cat = TRUE. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.

covar	either a vector or a list to be specified only if data = NULL. If all.cat = TRUE can be a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise is a list containing cat, the list of the qualitative covariates with their level and quant, the list the quantitative covariates that are simulated from the normal distribution with the given mean and standard deviation (see Examples).
n	number of patients (to be specified only if data = NULL).
all.cat	logical. If all the covariates in data are qualitative must be set equal to TRUE, otherwise must be set equal to FALSE.
nrep	number of trial replication.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the D_A-optimum BCD by Atkinson (see DABCD).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covar requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications according to the nature of the covariates. See also DABCD.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design. Sample_size number of patients. n_cov number of covariates. var_names name of the covariates. n_quantitative_variables number of quantitative covariates. Is NULL if all.cat = TRUE. n_categorical_variables number of levels of each covariate. Is NULL if all.cat = TRUE or only quantitative covariates are present. n_levels number of levels of each qualitative covariate. Is NULL if only quantitative covariates are present. n.rep number of replications.
Imbalances	a list with the imbalance measures at the end of each simulated trial Imb.measures summary of overall imbalances (Loss loss, Mahal Mahalanobis distance, overall.imb difference in the total number of patients assigned to A and B).

`within.imb` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is `NULL` if only quantitative covariates are present).

`strata.imb` (only if `all.cat = TRUE`) the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum).

`strata.A` (only if `all.cat = TRUE`) is the total number of patients assigned to A within the stratum.

`strata.N` (only if `all.cat = TRUE`) is the total number of patients assigned to each stratum.

`diff_mean` the difference in mean in group A and B for each quantitative covariate. Is `NULL` if `all.cat = TRUE`.

`obs.strata` (only if `all.cat = TRUE`) matrix of the possible strata.

`out` For each replication returns a list of the data provided in input (`data`) and the resulting assignments (`Assignment`).

References

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

See Also as DABCD.

Examples

```
require(covadap)

# Here we set nrep = 50 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.

### Implement with qualitative covariates (set all.cat = TRUE)
#### With an existing dataset
# Create a sample dataset with qualitative covariates
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
DABCD.sim(data = df1,  covar = NULL, n = NULL, all.cat = TRUE, nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res1 <- DABCD.sim(data = df1, covar = NULL, n = NULL, all.cat = TRUE,
                  nrep = 50)
#### By specifying the covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- DABCD.sim(data = NULL, covar = c(2,3,3), n = 100,
                  all.cat = TRUE, nrep = 50)
```

```
### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                   TRUE),
                  "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                  "cholesterol" = round(rnorm(100, 200, 8),1),
                  "height" = rpois(100,160),
                   stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
#### With an existing dataset
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
DABCD.sim(data = ff1[,5:6], covar = NULL, n = NULL,  all.cat = FALSE,
          nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res3 <- DABCD.sim(data = ff1[,5:6],  covar = NULL, n = NULL,
                  all.cat = FALSE, nrep = 50)
#### By specifying the covariates
# e.g. 2 quantitative covariates:
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
covar = list(quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
# To just view a summary of the metrics of the design
DABCD.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
          nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res4 <- DABCD.sim(data = NULL, covar = covar, n = 100,
                  all.cat = FALSE,  nrep = 50)


### With mixed covariates (set all.cat = FALSE)
#### With an existing dataset
# To just view a summary of the metrics of the design
DABCD.sim(data = ff1, covar = NULL, n = NULL,  all.cat = FALSE,
          nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res5 <- DABCD.sim(data = ff1, covar = NULL, n = NULL,
                  all.cat = FALSE, nrep = 50)
#### By specifying the covariates
# e.g. one qualitative covariate and 2 quantitative covariates:
# gender with levels M and F
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
covar = list(cat = list(gender = c("M", "F")),
             quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
#To just view a summary of the metrics of the design
DABCD.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
```

```
          nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res6 <- DABCD.sim(data = NULL, covar = covar, n = 100,
                  all.cat = FALSE,  nrep = 50)
```

ECADE *Efficient Covariate-Adaptive Design*

Description

Implements the Efficient Covariate-Adaptive DEsign by Baldi Antognini et al. (2022) for assigning patients to two treatments A and B. The procedure works with qualitative and quantitative covariates.

Usage

```
ECADE(data, all.cat, rho = 0.85, alloc.function = "Efron",
      print.results = TRUE)
```

Arguments

`data`	a data frame or a matrix. It can be a matrix only when `all.cat = TRUE`. Each row of `data` corresponds to the covariate profile of a patient.
`all.cat`	logical. If all the covariates in `data` are qualitative must be set equal to `TRUE`, otherwise must be set equal to `FALSE`.
`rho`	biasing probability, to be used only with the Efron allocation function ($1/2 \leq \rho \leq 1$). The default value is 0.85.
`alloc.function`	a character specifying the allocation function used in the randomization procedure: `="Efron"` for Efron's, if the assignment probability to A is $e+(1-2e)[1-\Phi(x)]$ with $e = 0.1$ `alloc.function="norm1"` while `alloc.function="norm2"` for $e = 0.2$.
`print.results`	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B with the Efficient Covariate-Adaptive Design as described in Baldi Antognini et al. (2022).

This randomization procedure can be used when `data` contains only qualitative covariate, in this case set `all.cat = TRUE`, when `data` contains only quantitative covariates or when covariates of mixed nature are present, in these two latter cases set `all.cat = FALSE`. The function's output is slighly different according to these three scenarios as described in Value.

The assignment probability to A of each patient is based on the Efron's allocation function (Efron, 1971) with biasing probability equal to `rho` if `alloc.function = "Efron"`. Otherwise the allocation ptobability to A is based on the cumulative distribution function of the standard normal distribution Φ (see Arguments).

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B.

Only when `all.cat = TRUE`, the function returns the strata imbalances measures, that report, for each stratum, the total number of patients assigned (`N.strata`), the number of patients assigned to A (`A.strata`) and the within-stratum imbalance (`D.strata`), calculated as `2*A.strata-N.strata`.

If at least one qualitative covariate is present, the function returns the within-covariate imbalances reporting, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B.

If at least one quantitative covariate is present, the function returns the difference in means. For each quantitative covariate, is reported the difference in the mean in group A and B.

See Value for more details.

Value

It returns an object of `class "covadap"`, which is a list containing the following elements:

`summary.info` `Design` name of the design.

`Sample_size` number of patients.

`n_cov` number of covariates.

`n_categorical_variables` number of levels of each covariate. Is `NULL` if `all.cat = TRUE` or only quantitative covariates are present).

`n_levels` number of levels of each qualitative covariate. Is `NULL` if only quantitative covariates are present.

`var_names` name of the covariates.

`cov_levels_names` levels of each qualitative covariate. Is `NULL` if only quantitative covariates are present.

`n_quantitative_variables` number of quantitative covariates. Is `NULL` if `all.cat = TRUE`.

`Assignments` a vector with the treatment assignments.

`Imbalances.summary` summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Strata.measures` (only if `all.cat = TRUE`) a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum).

`Imbalances` a list containing all the imbalance measures.

`Imb.measures` summary of overall imbalances (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Within.strata` (only if `all.cat = TRUE`) within-stratum imbalance for all strata.

`Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is `NULL` if only quantitative covariates are present).

`data` the data provided in input.

`diff_mean` (only if `all.cat = FALSE`) the difference in mean of the quantitative covariates in group A and B.

`observed.strata` (only if `all.cat = TRUE`) a data frame with all the observed strata.

References

Baldi Antognini A, Frieri R, Zagoraiou M, Novelli M. *The Efficient Covariate-Adaptive Design for high-order balancing of quantitative and qualitative covariates*. Statistical Papers, 2022.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

Efron B, *Forcing a sequential experiment to be balanced*. Biometrika, 1971, 58(3): 403-418.

See Also

See Also as `ECADE.sim` for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)
# Assume we choose Efron's allocation function with rho  = 0.85

### Implement with qualitative covariates (set all.cat = TRUE)
# Create a sample dataset with qualitative covariates
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
ECADE(data = df1,  all.cat = TRUE, alloc.function = "Efron",
      rho = 0.85, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res1 <- ECADE(data = df1,  all.cat = TRUE, alloc.function = "Efron",
              rho = 0.85, print.results = TRUE)
res1


### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                   TRUE),
                  "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                  "cholesterol" = round(rnorm(100, 200, 8),1),
```

```
                "height" = rpois(100,160),
                 stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
ECADE(data = ff1[,5:6],  all.cat = FALSE, alloc.function = "Efron",
      rho = 0.85, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res2 <- ECADE(data = ff1[,5:6],  all.cat = FALSE, alloc.function = "Efron",
              rho = 0.85, print.results = TRUE)
res2

### With mixed covariates (set all.cat = FALSE)
# To just view a summary of the metrics of the design
ECADE(data = ff1, all.cat = FALSE, alloc.function = "Efron",
      rho = 0.85, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res3 <- ECADE(data = ff1, all.cat = FALSE, alloc.function = "Efron",
              rho = 0.85, print.results = TRUE)
res3
```

ECADE.sim *Simulations of the Efficient Covariate-Adaptive Design*

Description

Implements the Efficient Covariate-Adaptive DEsign by Baldi Antognini et al. (2022) for assigning patients to two treatments A and B by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative and quantitative covariates.

Usage

```
#With existing dataframe provided in data
ECADE.sim(data, covar = NULL, n = NULL, all.cat, nrep = 1000,
          rho = 0.85, alloc.function = "Efron", print.results = TRUE)
#With covariates
ECADE.sim(data = NULL, covar, n, all.cat, nrep = 1000, rho = 0.85,
          alloc.function = "Efron", print.results = TRUE)
```

Arguments

data	a data frame or a matrix. It can be a matrix only when all.cat = TRUE. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.

covar	either a vector or a list to be specified only if data = NULL. If all.cat = TRUE is a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise is a list containing cat, the list of the qualitative covariates with their level and quant, the list the quantitative covariates that are simulated from the normal distribution with the given mean and standard deviation.
n	number of patients (to be specified only if data = NULL)
all.cat	logical. If all the covariates in data are qualitative must be set equal to TRUE, otherwise must be set equal to FALSE.
nrep	number of trial replications.
rho	biasing probability, to be used only with the Efron allocation function ($1/2 \leq \rho \leq 1$). The default value is 0.85. Is NULL if other allocation functions are used.
alloc.function	a character specifying the allocation function used in the randomization procedure: ="Efron" for Efron's, if the assignment probability to A is $e+(1-2e)[1-\Phi(x)]$ with $e = 0.1$ alloc.function="norm1" while alloc.function="norm2" for $e = 0.2$.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the Efficient Covariate-Adaptive Design as described in Baldi Antognini et al. (see ECADE).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covar requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications according to the nature of the covariates. See also ECADE.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design. Sample_size number of patients. n_cov number of covariates. var_names name of the covariates. n_quantitative_variables number of quantitative covariates. Is NULL if all.cat = TRUE. n_categorical_variables number of levels of each covariate. Is NULL if all.cat = TRUE or only quantitative covariates are present.

`n_levels` number of levels of each qualitative covariate. Is `NULL` if only quantitative covariates are present.

`n.rep` number of replications.

`Imbalances` a list with the imbalance measures at the end of each simulated trial

`Imb.measures` summary of overall imbalances (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`within.imb` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is `NULL` if only quantitative covariates are present).

`strata.imb` (only if `all.cat` = `TRUE`) the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum).

`strata.A` (only if `all.cat` = `TRUE`) is the total number of patients assigned to A within the stratum.

`strata.N` (only if `all.cat` = `TRUE`) is the total number of patients assigned to each stratum.

`diff_mean` (only if `all.cat` = `FALSE`) the difference in mean in group A and B for each quantitative covariate.

`obs.strata` (only if `all.cat` = `TRUE`) matrix of the possible strata.

`out` For each replication returns a list of the data provided in input (`data`) and the resulting assignments (`Assignment`).

References

Baldi Antognini A, Frieri R, Zagoraiou M, Novelli M. *The Efficient Covariate-Adaptive Design for high-order balancing of quantitative and qualitative covariates*. Statistical Papers, 2022.

See Also

See Also `ECADE`.

Examples

```
require(covadap)
# Here we set nrep = 50 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.


### Implement with qualitative covariates (set all.cat = TRUE)
#### With an existing dataset
# Create a sample dataset with qualitative covariates
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
ECADE.sim(data = df1,  covar = NULL, n = NULL, all.cat = TRUE,
          alloc.function = "Efron", rho = 0.85, nrep = 50)
# To view a summary
```

```
# and create a list containing all the metrics of the design
res1 <- ECADE.sim(data = df1, covar = NULL, n = NULL, all.cat = TRUE,
                  alloc.function = "Efron", rho = 0.85, nrep = 50)

#### By specifying the covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- ECADE.sim(data = NULL, covar = c(2,3,3), n = 100, all.cat = TRUE,
                  alloc.function = "Efron",  rho = 0.85, nrep = 50)



### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                 "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                  TRUE),
                 "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                 "cholesterol" = round(rnorm(100, 200, 8),1),
                 "height" = rpois(100,160),
                  stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
#### With an existing dataset
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
ECADE.sim(data = ff1[,5:6], covar = NULL, n = NULL,  all.cat = FALSE,
          alloc.function = "Efron", rho = 0.85, nrep = 50)
# To view a summary and create a list containing all the metrics of the design
res3 <- ECADE.sim(data = ff1[,5:6],  covar = NULL, n = NULL, all.cat = FALSE,
                  alloc.function = "Efron", rho = 0.85, nrep = 50)
#### By specifying the covariates
# e.g. 2 quantitative covariates:
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
covar = list(quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
# To just view a summary of the metrics of the design
ECADE.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
          alloc.function = "Efron", rho = 0.85, nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res4 <- ECADE.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
                  alloc.function = "Efron", rho = 0.85,  nrep = 50)



### With mixed covariates (set all.cat = FALSE)
#### With an existing dataset
# To just view a summary of the metrics of the design
ECADE.sim(data = ff1, covar = NULL, n = NULL,  all.cat = FALSE,
          alloc.function = "Efron",  rho = 0.85, nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
```

```
res5 <- ECADE.sim(data = ff1, covar = NULL, n = NULL,  all.cat = FALSE,
                  alloc.function = "Efron", rho = 0.85, nrep = 50)
#### By specifying the covariates
# e.g. one qualitative covariate and 2 quantitative covariates:
# gender with levels M and F
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
covar = list(cat = list(gender = c("M", "F")),
             quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
# To just view a summary of the metrics of the design
ECADE.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
          alloc.function = "Efron", rho = 0.85, nrep = 50)
# To view a summary and
# create a list containing all the metrics of the design
res6 <- ECADE.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
                  alloc.function = "Efron", rho = 0.85, nrep = 50)
```

HuHu *Covariate-Adaptive randomization by Hu and Hu*

Description

Implements the Covariate-Adaptive randomization by Hu and Hu (2012) for assigning patients to two treatments A and B. The procedure works with qualitative covariates only.

Usage

```
HuHu(data, p = 0.85, omega = NULL, print.results = TRUE)
```

Arguments

`data`	a data frame or a matrix. Each row of `data` corresponds to the covariate profile of a patient.
`p`	biased coin probability for the Efron's allocation function ($1/2 \leq p \leq 1$). The default value is 0.85.
`omega`	vector of weights for the overall, within-stratum, and within-covariate-margin levels. If `NULL` (default) `omega = rep(1/(n_cov + 2), n_cov + 2)`.
`print.results`	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B as described in Hu and Hu (2012).

The assignment probability to A of each patient is based on the Efron's allocation function (Efron, 1971) with biasing probability equal to `p`.

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the

difference in the total number of patients assigned to treatment A and B. The strata imbalances measures report, for each stratum, the total number of patients assigned (`N.strata`), the number of patients assigned to A (`A.strata`) and the within stratum imbalance (`D.strata`), calculated as `2*A.strata-N.strata`. The within covariate imbalances report, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B. See also Value.

Value

It returns an object of `class` `"covadap"`, which is a list containing the following elements:

`summary.info` `Design` name of the design,
`Sample_size` number of patients,
`n_cov` number of covariates,
`n_levels` number of levels of each covariate,
`var_names` name of covariates and levels,
`parameter_a` design parameter (see above).

`Assignments` a vector with the treatment assignments.

`Imbalances.summary` summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overal.imb` difference in the total number of patients assigned to A and B).

`Strata.measures` a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum.

`Imbalances` a list containing all the imbalance measures:
`Imb.measures` (`Loss` loss, `Mahal` Mahalanobis distance),
`Overall.imb` difference in the total number of patients assigned to A and B,
`Within.strata` within-stratum imbalance for all strata,
`Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate.

`data` the data provided in input.

`observed.strata` a data frame with all the observed strata.

References

Hu Y, Hu F. *Asymptotic properties of covariate-adaptive randomization*. The Annals of Statistics, 2012, 40(3): 1794-1815.

Efron B, *Forcing a sequential experiment to be balanced*. Biometrika, 1971, 58(3): 403-418.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

HuHu.sim for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)


# Create a sample dataset
df1 <- data.frame("gender" = sample(c("female", "male"), 200, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 200, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 200, TRUE),
                   stringsAsFactors = TRUE)
# To view a summary of the metrics of the design
HuHu(df1, p = 0.85, omega = NULL, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res <- HuHu(df1, p = 0.85, omega = NULL, print.results = TRUE)
res
```

HuHu.sim *Simulations of the Covariate-Adaptive randomization by Hu and Hu*

Description

Implements the Covariate-Adaptive randomization by Hu and Hu (2012) for assigning patients to two treatments A and B by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative covariates only.

Usage

```
#With existing dataframe
HuHu.sim(data, covar = NULL, n = NULL, p = 0.85,
         omega = NULL, nrep = 1000, print.results = TRUE)
#With covariates
HuHu.sim(data = NULL, covar, n, p = 0.85, omega = NULL,
         nrep = 1000, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.
covar	either a vector or a list to be specified only if data = NULL. It could be a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise it is a list containing the covariates with their levels (e.g. one covariate with two levels and one with three covar = list(cov1 = c("lev1", "lev2"), cov2 = c("lev1", "lev2", "lev3")).

n	number of patients (to be specified only if data = NULL).
p	biased coin probability for the Efron's allocation function ($1/2 \leq p \leq 1$). The default is 0.85.
omega	vector of weights for the overall, within-stratum, and within-covariate-margin levels. If NULL (default) omega = rep(1/(n_cov + 2), n_cov + 2).
nrep	number of trial replications.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the Covariate-Adaptive randomization procedure proposed by Hu and Hu (see HuHu).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covar requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications. See also HuHu.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design, Sample_size number of patients, n_cov number of covariates, n_levels number of levels of each qualitative covariate, var_names name of the covariates, n.rep number of replications, Maximum_tolerated_imbalance for the BSD procedure.
Imbalances	a list with the imbalance measures at the end of each simulated trial: Imb.measures summary of overall imbalances (Loss loss, Mahal Mahalanobis distance, overall.imb difference in the total number of patients assigned to A and B), within.imb within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate, strata.imb the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum), strata.A total number of patients assigned to A within the stratum, strata.N total number of patients assigned to each stratum, obs.strata matrix of the possible strata.
out	For each replication returns a list of the data provided in input (data) and the resulting assignments (Assignment).

References

Hu Y, Hu F. *Asymptotic properties of covariate-adaptive randomization*. The Annals of Statistics, 2012, 40(3): 1794-1815.

See Also

See Also HuHu.

Examples

```
require(covadap)
# Here we set nrep = 100 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.


### With existing dataframe
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# Simulate the design
res2 <- HuHu.sim(data = df1, covar = NULL, n = NULL, p = 0.85,  omega = NULL,
                 nrep = 100)
### With covariate
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- HuHu.sim(data = NULL, covar = c(2,2,3), n = 100, p = 0.85,  omega = NULL,
                 nrep = 100)
```

KER *Covariate-Adaptive randomization by Ma and Hu*

Description

Implements the Covariate-Adaptive randomization by Ma and Hu (2013) for assigning patients to two treatments A and B in order to minimize the distance between the covariate distribution in the two treatment groups. The procedure works with qualitative and quantitative covariates.

Usage

```
KER(data, all.cat, p = 0.8, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. It can be a matrix only when all.cat = TRUE. Each row of data corresponds to the covariate profile of a patient.
all.cat	logical. If all the covariates in data are qualitative must be set equal to TRUE, otherwise must be set equal to FALSE.

p	biasing probability for the Efron's allocation function ($1/2 \leq p \leq 1$). The default value is 0.8.
print.results	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B with the Covariate-Adaptive randomization based on kernel density estimation as described in Ma and Hu (2013).

This randomization procedure can be used when data contains only qualitative covariate, in this case set all.cat = TRUE, when data contains only quantitative covariates or when covariates of mixed nature are present, in these two latter cases set all.cat = FALSE. The function's output is slighly different according to these three scenarios as described in Value.

The assignment probability to A of each patient is based on the Efron's allocation function (Efron, 1971) with biasing probability equal to p.

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B.

Only when all.cat = TRUE, the function returns the strata imbalances measures, that report, for each stratum, the total number of patients assigned (N.strata), the number of patients assigned to A (A.strata) and the within-stratum imbalance (D.strata), calculated as 2*A.strata-N.strata.

If at least one qualitative covariate is present, the function returns the within-covariate imbalances reporting, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B.

If at least one quantitative covariate is present, the function returns the difference in means. For each quantitative covariate, is reported the difference in the mean in group A and B.

See Value for more details.

Value

It returns an object of class "covadap", which is a list containing the following elements:

summary.info	Design name of the design. Sample_size number of patients. n_cov number of covariates. n_categorical_variables number of levels of each covariate. Is NULL if all.cat = TRUE or only quantitative covariates are present). n_levels number of levels of each qualitative covariate. Is NULL if only quantitative covariates are present. var_names name of the covariates. cov_levels_names levels of each qualitative covariate. Is NULL if only quantitative covariates are present. n_quantitative_variables number of quantitative covariates. Is NULL if all.cat = TRUE.
Assignments	a vector with the treatment assignments.

`Imbalances.summary`
: summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

`Strata.measures`
: (only if `all.cat = TRUE`) a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum).

`Imbalances`
: a list containing all the imbalance measures.

  `Imb.measures` summary of overall imbalances (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B).

  `Within.strata` (only if `all.cat = TRUE`) within-stratum imbalance for all strata.

  `Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is `NULL` if only quantitative covariates are present).

`data`
: the data provided in input.

`diff_mean`
: (only if `all.cat = FALSE`) the difference in mean of the quantitative covariates in group A and B.

`observed.strata`
: (only if `all.cat = TRUE`) a data frame with all the observed strata.

References

Ma Z and Hu F. *Balancing continuous covariates based on Kernel densities*. Contemporary Clinical Trials, 2013, 34(2): 262-269.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

Efron B, *Forcing a sequential experiment to be balanced*. Biometrika, 1971, 58(3): 403-418.

See Also

See Also as `KER.sim` for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)

### Implement with qualitative covariates (set all.cat = TRUE)
# Create a sample dataset with qualitative covariates
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
KER(data = df1,  all.cat = TRUE, p = 0.8, print.results = TRUE)
```

```
# To view a summary
# and create a list containing all the metrics of the design
res1 <- KER(data = df1,  all.cat = TRUE, p = 0.8, print.results = TRUE)
res1


### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                   TRUE),
                  "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                  "cholesterol" = round(rnorm(100, 200, 8),1),
                  "height" = rpois(100,160),
                   stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
KER(data = ff1[,5:6],  all.cat = FALSE, p = 0.8, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res2 <- KER(data = ff1[,5:6], p = 0.8, all.cat = FALSE, print.results = TRUE)
res2

### With mixed covariates
# In this case the user must set all.cat = FALSE
# To just view a summary of the metrics of the design
KER(data = ff1, all.cat = FALSE, p = 0.8, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res3 <- KER(data = ff1, all.cat = FALSE, p = 0.8, print.results = TRUE)
res3
```

KER.sim *Simulations of the Covariate-Adaptive randomization by Ma and Hu*

Description

Implements the Covariate-Adaptive randomization by Ma and Hu (2013) for assigning patients to two treatments A and B in order to minimize the distance between the covariate distribution in the two treatment groups by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative and quantitative covariates.

Usage

```
#With existing dataframe
```

```
KER.sim(data, covar = NULL, n = NULL, all.cat, nrep = 1000,
        p = 0.8, print.results = TRUE)
#With covariates
KER.sim(data = NULL, covar, n, all.cat, nrep = 1000,
        p = 0.8, print.results = TRUE)
```

Arguments

data	a data frame or a matrix. It can be a matrix only when all.cat = TRUE. Each row of data corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in data.
covar	either a vector or a list to be specified only if data = NULL. If all.cat = TRUE is a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise is a list containing cat, the list of the qualitative covariates with their level and quant, the list the quantitative covariates that are simulated from the normal distribution with the given mean and standard deviation.
n	number of patients.
all.cat	logical. If all the covariates in data are qualitative must be set equal to TRUE, otherwise must be set equal to FALSE.
nrep	number of trial replications.
p	biasing probability for the Efron allocation function ($1/2 \leq p \leq 1$). The default value is 0.8.
print.results	logical. If TRUE a summary of the results is printed.

Details

This function simulates nrep times a clinical study assigning patients to treatments A and B with the Efficient Covariate-Adaptive Design as described in Ma and Hu (see KER).

When covar is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of covar requires the specification of the number of patients n.

When data is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of data.

The summary printed when print.results = TRUE reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications according to the nature of the covariates. See also KER.

Value

It returns an object of class "covadapsim", which is a list containing the following elements:

summary.info	Design name of the design. Sample_size number of patients.

n_cov number of covariates.

var_names name of the covariates.

n_quantitative_variables number of quantitative covariates. Is NULL if all.cat = TRUE.

n_categorical_variables number of levels of each covariate. Is NULL if all.cat = TRUE or only quantitative covariates are present.

n_levels number of levels of each qualitative covariate. Is NULL if only quantitative covariates are present.

n.rep number of replications.

Imbalances a list with the imbalance measures at the end of each simulated trial

Imb.measures summary of overall imbalances (Loss loss, Mahal Mahalanobis distance, overall.imb difference in the total number of patients assigned to A and B).

within.imb within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate (is NULL if only quantitative covariates are present).

strata.imb (only if all.cat = TRUE) the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum).

strata.A (only if all.cat = TRUE) is the total number of patients assigned to A within the stratum.

strata.N (only if all.cat = TRUE) is the total number of patients assigned to each stratum.

diff_mean (only if all.cat = FALSE) the difference in mean in group A and B for each quantitative covariate.

obs.strata (only if all.cat = TRUE) matrix of the possible strata.

out For each replication returns a list of the data provided in input (data) and the resulting assignments (Assignment).

References

Ma Z and Hu F. *Balancing continuous covariates based on Kernel densities*. Contemporary Clinical Trials, 2013, 34(2): 262-269.

See Also

See Also as KER.

Examples

```
require(covadap)
# Here we set nrep = 50 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.


### Implement with qualitative covariates (set all.cat = TRUE)
#### With an existing dataset
# Create a sample dataset with qualitative covariates
```

```
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
KER.sim(data = df1,  covar = NULL, n = NULL, all.cat = TRUE,
        p = 0.8, nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res1 <- KER.sim(data = df1, covar = NULL, n = NULL, all.cat = TRUE,
                p = 0.8, nrep = 50)
#### By specifying the covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- KER.sim(data = NULL, covar = c(2,3,3), n = 100, all.cat = TRUE,
                p = 0.8, nrep = 50)



### Implement with quantitative or mixed covariates
# Create a sample dataset with covariates of mixed nature
ff1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hypertension"), 10,
                   TRUE),
                  "smoke" = sample(c("yes", "no"), 100, TRUE, c(2 / 3, 1 / 3)),
                  "cholesterol" = round(rnorm(100, 200, 8),1),
                  "height" = rpois(100,160),
                   stringsAsFactors = TRUE)

### With quantitative covariates only (set all.cat = FALSE)
#### With an existing dataset
# select only column 5 and 6 of the sample dataset
# To just view a summary of the metrics of the design
KER.sim(data = ff1[,5:6], covar = NULL, n = NULL,  all.cat = FALSE, p = 0.8,
        nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res3 <- KER.sim(data = ff1[,5:6],  covar = NULL, n = NULL, all.cat = FALSE,
                p = 0.8, nrep = 50)

#### By specifying the covariates
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
covar = list(quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
# To just view a summary of the metrics of the design
KER.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
        p = 0.8, nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res4 <- KER.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
                  p = 0.8,  nrep = 50)

### With mixed covariates (set all.cat = FALSE)
```

```
#### With an existing dataset
# To just view a summary of the metrics of the design
KER.sim(data = ff1, covar = NULL, n = NULL, all.cat = FALSE, p = 0.8,
        nrep = 50)
# To view a summary
# and create a list containing all the metrics of the design
res5 <- KER.sim(data = ff1, covar = NULL, n = NULL, all.cat = FALSE,
                 p = 0.8, nrep = 50)
#### By specifying the covariates
# e.g. one qualitative covariate and 2 quantitative covariates:
# BMI normally distributed with mean 26 and standard deviation 5
# cholesterol normally distributed with mean 200 and standard deviation 34
# gender with levels M and F
covar = list(cat = list(gender = c("M", "F")),
             quant = list(BMI = c(26, 5), cholesterol = c(200, 34)))
# To just view a summary of the metrics of the design
KER.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
        p = 0.8, nrep = 50)
# To view a summary and create a list containing all the metrics of the design
res6 <- KER.sim(data = NULL, covar = covar, n = 100, all.cat = FALSE,
                 p = 0.8, nrep = 50)
```

`Pocock and Simon design`

Pocock and Simon's minimization method

Description

Implements the Pocock and Simon's minimization method by Pocock and Simon (1975) for assigning patients to two treatments A and B. The procedure works with qualitative covariates only.

Usage

```
PocSim(data, p = 0.85, print.results = TRUE)
```

Arguments

`data`	a data frame or a matrix. Each row of `data` corresponds to the covariate profile of a patient.
`p`	biased coin probability for the Efron's allocation function ($1/2 \leq p \leq 1$). The default value is 0.85.
`print.results`	logical. If TRUE a summary of the results is printed.

Details

The function assigns patients to treatments A or B as described in Pocock and Simon (1975).

The assignment probability to A of each patient is based on the Efron's allocation function (Efron, 1971) with biasing probability equal to `p`.

At the end of the study the imbalance measures reported are the loss of estimation precision as described in Atkinson (1982), the Mahalanobis distance and the overall imbalance, defined as the difference in the total number of patients assigned to treatment A and B. The strata imbalances measures report, for each stratum, the total number of patients assigned (`N.strata`), the number of patients assigned to A (`A.strata`) and the within-stratum imbalance (`D.strata`), calculated as `2*A.strata-N.strata`. The within-covariate imbalances report, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B. See also Value.

Value

It returns an object of `class` `"covadap"`, which is a list containing the following elements:

`summary.info` `Design` name of the design,
`Sample_size` number of patients,
`n_cov` number of covariates,
`n_levels` number of levels of each covariate,
`var_names` name of covariates and levels,
`parameter_a` design parameter (see above).

`Assignments` a vector with the treatment assignments.

`Imbalances.summary` summary of overall imbalance measures at the end of the study (`Loss` loss, `Mahal` Mahalanobis distance, `overal.imb` difference in the total number of patients assigned to A and B).

`Strata.measures` a data frame containing for each possiblue stratum the corresponding imbalances: `N.strata` is the total number of patients assigned to the stratum; `A.strata` is the total number of patients assigned to A within the stratum; `D.strata` is the within-stratum imbalance, i.e. difference in the total number of patients assigned to A and B within the stratum.

`Imbalances` a list containing all the imbalance measures:
`Imb.measures` (`Loss` loss, `Mahal` Mahalanobis distance),
`Overall.imb` difference in the total number of patients assigned to A and B,
`Within.strata` within-stratum imbalance for all strata,
`Within.cov` within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate.

`data` the data provided in input.

`observed.strata` a data frame with all the observed strata.

References

Pocock S J, Simon R. *Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial*. Biometrics, 1975, 31(1): 103-115.

Efron B, *Forcing a sequential experiment to be balanced*. Biometrika, 1971, 58(3): 403-418.

Atkinson A. C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*. Biometrika, 1982, 69(1): 61-67.

See Also

See Also as `PocSim.sim` for allocating patients by simulating their covariate profiles.

Examples

```
require(covadap)

# Create a sample dataset
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# To just view a summary of the metrics of the design
PocSim(df1, p = 0.85, print.results = TRUE)
# To view a summary
# and create a list containing all the metrics of the design
res <- PocSim(df1, p = 0.85, print.results = TRUE)
res
```

Pocock and Simon design simulations

Simulations of the Pocock and Simon's minimization method

Description

Implements the Pocock and Simon's minimization method by Pocock and Simon (1975) for assigning patients to two treatments A and B by simulating the covariate profile of each patient using an existing dataset or specifying number and levels of the covariates. The procedure works with qualitative covariates only.

Usage

```
#With existing dataframe
PocSim.sim(data, covar = NULL, n = NULL, p = 0.85, nrep = 1000,
           print.results = TRUE)
#With covariates
PocSim.sim(data = NULL, covar, n, p = 0.85, nrep = 1000,
           print.results = TRUE)
```

Arguments

`data`	a data frame or a matrix. Each row of `data` corresponds to the covariate profile of a patient. To be specified if covariate profiles should be sampled from an existing dataset provided in `data`.
`covar`	either a vector or a list to be specified only if `data = NULL`. It could be a vector with length equal to the number of covariates and elements equal to the number of levels for each covariate. Otherwise it is a list containing the covariates with their levels (e.g. one covariate with two levels and one with three `covar = list(cov1 = c("lev1", "lev2"), cov2 = c("lev1", "lev2", "lev3"))`
`n`	number of patients (to be specified only if `data = NULL`).
`p`	biased coin probability for the Efron's allocation function ($1/2 \leq p \leq 1$). The default value is 0.85.
`nrep`	number of trial replications.
`print.results`	logical. If TRUE a summary of the results is printed.

Details

This function simulates `nrep` times a clinical study assigning patients to treatments A and B with the minimization method by Pocock and Simon (see `PocSim`).

When `covar` is provided, the function finds all the possible combination of the levels of the covariates, i.e., the strata and, at each trial replication, the patients' covariate profiles are uniformly sampled within those strata. The specification of `covar` requires the specification of the number of patients `n`.

When `data` is provided, at each trial replication, the patients' covariate profiles are sampled from the observed strata with uniform distribution. In this case the number of patients equals the number of rows of `data`.

The summary printed when `print.results = TRUE` reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the `nrep` trial replications. See also `PocSim`.

Value

It returns an object of `class` `"covadapsim"`, which is a list containing the following elements:

`summary.info`	`Design` name of the design, `Sample_size` number of patients, `n_cov` number of covariates, `n_levels` number of levels of each qualitative covariate, `var_names` name of the covariates, `n.rep` number of trial replications, `Maximum_tolerated_imbalance` for the BSD procedure.
`Imbalances`	a list with the imbalance measures at the end of each simulated trial: `Imb.measures` summary of overall imbalances (`Loss` loss, `Mahal` Mahalanobis distance, `overall.imb` difference in the total number of patients assigned to A and B),

within.imb within-covariate imbalance: difference in the number of patients assigned to A and B for each level of each qualitative covariate,

strata.imb the within-stratum imbalance (i.e. difference in the total number of patients assigned to A and B within the stratum),

strata.A total number of patients assigned to A within the stratum,

strata.N total number of patients assigned to each stratum,

obs.strata matrix of the possible strata.

out For each replication returns a list of the data provided in input (data) and the resulting assignments (Assignment).

References

Pocock S J, Simon R. *Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial*. Biometrics, 1975, 31(1): 103-115.

See Also

See Also as PocSim.

Examples

```
require(covadap)
# Here we set nrep = 100 for illustrative purposes,
# Set it equal to at least 5000 for more reliable Monte Carlo estimates.

### With existing dataframe
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                   stringsAsFactors = TRUE)
# Simulate the design
res1 <- PocSim.sim(data = df1, covar = NULL, n = NULL, p = 0.85, nrep = 100)
### With covariates
# e.g. two binary covariates and one with three levels and 100 patients
res2 <- PocSim.sim(data = NULL, covar = c(2,2,3), n = 100,
                   p = 0.85, nrep = 100)
```

---

summary_covadap *Summary of Covariate-Adaptive Designs*

Description

This function automatically recognizes the design implemented and provides a summary of the results.

Usage

```
summary_covadap(res)
```

Arguments

res	An object of class "covadap" or "covadapsim" resulting from the application of a covariate-adaptive design.

Details

When applied to an object of class "covadap": if at least one qualitative covariate is present, the function returns the within-covariate imbalances reporting, for each level of each qualitative covariate, the difference in the number of patients assigned to A and B. If instead at least one quantitative covariate is present, the function returns the difference in means. For each quantitative covariate, the difference in the mean in group A and B is reported.

When applied to an object of class "covadapsim", it reports the averages, in absolute value, of the imbalance measures, strata imbalances and within-covariate imbalances of the nrep trial replications according to the nature of the covariates.

Value

The form of the value returned by summary_covadap depends on the class of the argument provided (see Details).

Examples

```
#Create a sample dataset
df1 <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                  "age" = sample(c("18-35", "36-50", ">50"), 100, TRUE),
                  "bloodpressure" = sample(c("normal", "high", "hyper"), 100, TRUE),
                  stringsAsFactors = TRUE)

res1 <- ECADE(data = df1, all.cat = TRUE,
                   alloc.function = "Efron", rho = 0.85)
summary_covadap(res1)


res2 <- ECADE.sim(data = df1, cov = NULL, n = NULL, all.cat = TRUE,
                   alloc.function = "Efron", rho = 0.85, nrep = 100)
summary_covadap(res2)
```

Index