Package 'carat'

September 23, 2021

Type Package

Title Covariate-Adaptive Randomization for Clinical Trials

Version 2.0

Date 2021-09-22

Author Fuyi Tu [aut],
Xiaoqing Ye [aut, cre],
Wei Ma [aut, ths],
Feifang Hu [aut, ths]

Maintainer Xiaoqing Ye <ye_xiaoq@163.com>

Description
Provides functions and command-line user interface to generate allocation sequence by covariate-adaptive randomization for clinical trials. The package currently supports six covariate-adaptive randomization procedures. Three hypothesis testing methods that are valid and robust under covariate-adaptive randomization are also available in the package to facilitate the inference for treatment effect under the included randomization procedures. Additionally, the package provides comprehensive and efficient tools to allow one to evaluate and compare the performance of randomization procedures and tests based on various criteria.

License GPL (>= 2)

Imports Rcpp (>= 1.0.4.6), ggplot2 (>= 3.3.0), gridExtra (>= 2.3), stringr (>= 1.4.0), methods

Suggests dplyr (>= 0.8.5)

Depends R (>= 3.6.0)

NeedsCompilation yes

LinkingTo Rcpp, RcppArmadillo

Repository CRAN

Date/Publication 2021-09-23 12:30:02 UTC

R topics documented:

`carat-package` *carat-package: Covariate-Adaptive Randomization for Clinical Trials*

Description

Provides functions and a command-line user interface to generate allocation sequences for clinical trials with covariate-adaptive randomization methods. It currently supports six different covariate-adaptive randomization procedures, including stratified randomization, minimization, and a general family of designs proposed by Hu and Hu (2012) <doi:10.1214/12-AOS983>. Three hypothesis testing methods, all valid and robust under covariate-adaptive randomization are also included in the package to facilitate the inference for treatment effects under the included randomization procedures. Additionally, the package provides comprehensive and efficient tools for the performance evaluation and comparison of randomization procedures and tests based on various criteria.

Acknowledgement

This work was supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China [grant number 20XNA023].

Author(s)

Fuyi Tu <fuyi.tu@ruc.edu.cn>;Xiaoqing Ye <ye_xiaoq@163.com>; Wei Ma <mawei@ruc.edu.cn>; Feifang Hu <feifang@gwu.edu>.

References

Atkinson A C. *Optimum biased coin designs for sequential clinical trials with prognostic factors*[J]. Biometrika, 1982, 69(1): 61-67. <doi:10.2307/2335853>

Baldi Antognini A, Zagoraiou M. *The covariate-adaptive biased coin design for balancing clinical trials in the presence of prognostic factors*[J]. Biometrika, 2011, 98(3): 519-535. <doi:10.1093/biomet/asr021>

Hu Y, Hu F. *Asymptotic properties of covariate-adaptive randomization*[J]. The Annals of Statistics, 2012, 40(3): 1794-1815. <doi:10.1214/12-AOS983>

Ma W, Hu F, Zhang L. *Testing hypotheses of covariate-adaptive randomized clinical trials*[J]. Journal of the American Statistical Association, 2015, 110(510): 669-680. <doi:10.1080/01621459.2014.922469>

Ma W, Qin Y, Li Y, et al. *Statistical Inference for Covariate-Adaptive Randomization Procedures*[J]. Journal of the American Statistical Association, 2020, 115(531): 1488-1597. <doi:10.1080/01621459.2019.1635483>

Pocock S J, Simon R. *Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial*[J]. Biometrics, 1975: 103-115. <doi:10.2307/2529712>

Rosenberger W F, Lachin J M. *Randomization in clinical trials: theory and practice*[M]. John Wiley & Sons, 2015. <doi:10.1002/9781118742112>

Shao J., Yu, X. *Validity of tests under covariate-adaptive biased coin randomization and generalized linear models*[J]. Biometrics, 2013, 69(4), 960-969. <doi:10.1111/biom.12062>

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360. <doi:10.1093/biomet/asq014>

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375. <doi:10.1016/0021-9681(74)90015-0>

`AdjBCD` *Covariate-adjusted Biased Coin Design*

Description

Allocates patients to one of two treatments based on covariate-adjusted biased coin design as proposed by Baldi Antognini A, Zagoraiou M (2011) <doi:10.1093/biomet/asr021>.

Usage

```
AdjBCD(data, a = 3)
```

Arguments

`data`	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
`a`	a design parameter governing the degree of randomness. The default is `3`.

Details

Consider I covaraites and m_i levels for the ith covariate. T_j is the assignment of the jth patient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of the jth patient. For convenience, $(k_1, \ldots, k_I)$ and $(i; k_i)$ denote stratum and margin respectively. $D_n(.)$ is the difference between numbers of assigned patients in treatment 1 and treatment 2 at the corresponding level after n patients have been assigned.

Let F^a be a decreasing and symmetric function of $D_n(.)$, which depends on a design parameter $a \geq 0$. Then the probability of allocating the $(n+1)$th patient to treatment 1 is $F^a(D_n(.))$, where

$$F^a(x) = \frac{|x|^a}{|a|^a + 1},$$

for $x \leq -1$,

$$F^a(x) = 1/2,$$

for $x = 0$, and

$$F^a(x) = \frac{1}{|x|^a + 1},$$

for $x \geq 1$. As a goes to ∞, the design becomes more deteministic.

Details of the procedure can be found in Baldi Antognini and M. Zagoraiou (2011).

Value

It returns an object of `class` `"carandom"`.

The function `print` is used to obtain results. The generic accessor functions `Cov_Assig`, `Diff`, `data`, `All strata` and others extract various useful features of the value returned by `AdjBCD`.

An object of class `"carandom"` is a list containing at least the following components:

`cov_num`	the number of covariates.
`n`	the number of patients.
`Cov_Assign`	a `(cov_num + 1) * n` matrix containing covariate profiles for all patients and the corresponding assignments. The ith column represents the ith patient. The first `cov_num` rows include patients' covariate profiles, and the last row contains the assignment.
`All strata`	a matrix containing all strata involved.
`Diff`	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
`Data Type`	data type. `Real` or `Simulated`.

References

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

See Also

See `AdjBCD.sim` for allocating patients with covariate data generating mechanism; See `AdjBCD.ui` for the command-line user interface.

Examples

```
# a simple use
## Real Data
## create a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 1000, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 1000, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 1000, TRUE),
                 stringsAsFactors = TRUE)
Res <- AdjBCD(df, a = 2)
## view the output
Res

  ## view all patients' profile and assignments
  Res$Cov_Assig


## Simulated Data
n <- 1000
cov_num <- 3
level_num <- c(2, 3, 5)
# Set pr to follow two tips:
#(1) length of pr should be sum(level_num);
#(2) sum of probabilities for each margin should be 1.
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, rep(0.2, times = 5))
# set the design parameter
a <- 1.8
# obtain result
Res.sim <- AdjBCD.sim(n, cov_num, level_num, pr, a)

  # view the assignments of patients
  Res.sim$Cov_Assig[cov_num + 1, ]
  # view the differences between treatment 1 and treatment 2 at all levels
  Res.sim$Diff
```

`AdjBCD.sim` *Covariate-adjusted Biased Coin Design with Covariate Data Generating Mechanism*

Description

Allocates patients to one of two treatments based on the covariate-adjusted biased coin design as proposed by Baldi Antognini A, Zagoraiou M (2011) <doi:10.1093/biomet/asr021>, by simulating the covariates-profile under the assumption of independence between covariates and levels within each covariate.

Usage

```
AdjBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), a = 3)
```

Arguments

n	the number of patients. The default is 1000.
cov_num	the number of covariates. The default is 2.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates. The default is c(2,2).
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1. The default is rep(0.5,4), which corresponds to cov_num = 2, and level_num = c(2,2).
a	a design parameter governing the degree of randomness. The default is 3.

Details

See AdjBCD.

Value

See AdjBCD.

References

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

See Also

See AdjBCD for allocating patients with complete covariate data; See AdjBCD.ui for the command-line user interface.

AdjBCD.ui *Command-line User Interface Using Covariate-adjusted Biased Coin Design*

Description

A call to the user-interface function for allocation of patients to one of two treatments, using covariate-adjusted biased coin design, as proposed by Baldi Antognini A, Zagoraiou M (2011) <doi:10.1093/biomet/asr021>.

Usage

```
AdjBCD.ui(path, folder = "AdjBCD")
```

Arguments

path	the path in which a folder used to store variables will be created.
folder	name of the folder. If it is the default, a folder named "AdjBCD" will be created.

Details

See AdjBCD.

Value

It returns an object of class "carseq".

The function print is used to obtain results. The generic accessor functions assignment, covariate, cov_num, cov_profile and others extract various useful features of the value returned by AdjBCD.ui.

Note

This function provides a command-line user interface, and users should follow the prompts to enter data including covariates as well as levels for each covariate, design parameter a and the covariate profile of the new patient.

References

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

See Also

See AdjBCD for allocating patients with complete covariate data; See AdjBCD.sim for allocating patients with covariate data generating mechanism.

---

boot.test *Bootstrap t-test*

Description

Performs bootstrap t-test on treatment effects. This test is proposed by Shao et al. (2010) <doi:10.1093/biomet/asq014>.

Usage

```
boot.test(data, B = 200, method = c("HuHuCAR", "PocSimMIN", "StrBCD",
                                    "StrPBR", "DoptBCD", "AdjBCD"),
          conf = 0.95, ...)
```

Arguments

`data`	a data frame. It consists of patients' profiles, treatment assignments and outputs. See `getData`.
`B`	an integer. It is the number of bootstrap samples. The default is `200`.
`method`	the randomization procedure to be used for testing. This package provides tests for `"HuHuCAR"`, `"PocSimMIN"`, `"StrBCD"`, `"StrPBR"`, `"AdjBCD"`, and `"DoptBCD"`.
`conf`	confidence level of the interval. The default is `0.95`.
`...`	arguments to be passed to `method`. These arguments depend on the randomization method used and the following arguments are accepted:

omega a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that `omega` is only needed when `HuHuCAR` is to be used.

weight a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that `weight` is only needed when `PocSimMIN` is to be used.

p the biased coin probability. `p` should be larger than `1/2` and less than `1`. Note that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter governing the degree of randomness. Note that `a` is only needed when `"AdjBCD"` is to be used.

bsize the block size for stratified randomization. It is required to be a multiple of 2. Note that `bsize` is only needed when `"StrPBR"` is to be used.

Details

The bootstrap t-test is described as follows:

1) Generate bootstrap data $(Y_1^*, Z_1^*), \ldots, (Y_n^*, Z_n^*)$ as a simple random sample with replacement from the original data $(Y_1, Z_1), \ldots, (Y_n, Z_n)$, where Y_i denotes the outcome and Z_i denotes the profile of the ith patient.

2) Perform covariate-adaptive procedures on the patients' profiles to obtain new treatment assignments $T_1^*, \ldots, T_n^*$, and define

$$\hat{\theta}^* = -\frac{1}{n_1^*}\sum_{i=1}^{n}(T_i^* - 2) \times Y_i^* - \frac{1}{n_0^*}\sum_{i=1}^{n}(T_i^* - 1) \times Y_i$$

where n_1^* is the number of patients assigned to treatment 1 and n_0^* is the number of patients assigned to treatment 2.

3) Repeat step 2 B times to generate B independent boostrap samples to obtain $\hat{\theta}_b^*$, $b = 1, \ldots, B$. The variance of $\bar{Y}_1 - \bar{Y}_0$ can then be approximated by the sample variance of $\hat{\theta}_b^*$.

Value

It returns an object of class `"htest"`.

The function `print` is used to obtain results. The generic accessor functions `statistic`, `p.value`, `conf.int` and others extract various useful features of the value returned by `boot.test`.

An object of class `"htest"` is a list containing at least the following components:

data.name	a character string giving the name(s) of the data.
statistic	the value of the t-statistic.
pval	the p-value of the test,the null hypothesis is rejected if p-value is less than the pre-determined significance level.
conf.int	a confidence interval under the chosen level conf for the difference in treatment effect between treatment 1 and treatment 2.
estimate	the estimated treatment effect difference between treatment 1 and treatment 2.
method	a character string indicating what type of test was performed.

References

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

Examples

```
#Suppose the data used is patients' profile from real world,
#  while it is generated here. Data needs to be preprocessed
#  and then get assignments following certain randomization.
set.seed(100)
df<- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
               "jobs" = sample(c("stu.", "teac.", "other"), 100, TRUE, c(0.4, 0.2, 0.4)),
                stringsAsFactors = TRUE)
##data preprocessing
data.pd <- StrPBR(data = df, bsize = 4)$Cov_Assig

#Then we need to combine patients' profiles and outcomes after randomization and treatments.
outcome = runif(100)
data.combined = data.frame(rbind(data.pd,outcome), stringsAsFactors = TRUE)

#run the bootstrap t-test
B = 200
Strbt = boot.test(data.combined, B, "StrPBR", bsize = 4)
Strbt
```

---

compPower *Comparison of Powers for Different Tests under Different Randomization methods*

Description

Compares the power of tests under different randomization methods and treatment effects through matrices and plots.

Usage

```
compPower(powers, diffs, testname)
```

Arguments

powers	a list. Each argument consists the power generated by `evalPower()` in this package or by other sources. The length of each argument must match.
diffs	a vector. It contains values of group treatment effect differences. The length of this argument and the length of each argument of `powers` must match.
testname	a vector. Each element is the name of test and the randomization method used. For example, when applying `rand.test` and `corr.test` under `HuHuCAR`, it can be `c("HH.rand","HH.corr")`. The length of this argument must match the length of `diffs`.

Value

This function returns a list. The first element is a matrix consisting of powers of chosen tests under different values of treatment effects. The second element of the list is a plot of powers. `diffs` forms the vertical axis of the plot.

Examples

```
##settings
set.seed(100)
n = 1000
cov_num = 5
level_num = c(2,2,2,2,2)
pr = rep(0.5,10)
beta = c(1,4,3,2,5)
di = seq(0,0.5,0.1)
sigma = 1
type = "linear"
p=0.85
Iternum = 10 #<<for demonstration,it is suggested to be around 1000
sl = 0.05
weight = rep(0.1,5)

#comparison of corrected t-test under StrBCD and PocSim
##data generation
library("ggplot2")
Strctp=evalPower(n,cov_num,level_num,pr,type,beta,di,
                 sigma,Iternum,sl,"StrBCD","corr.test",FALSE,p)
PSctp=evalPower(n,cov_num,level_num,pr,type,beta,di,sigma,
                 Iternum,sl,"PocSimMIN","corr.test",FALSE,weight,p)
powers = list(Strctp,PSctp)
testname = c("StrBCD.corr","PocSimMIN.corr")

#get plot and matrix for comparison
cp = compPower(powers,di,testname)
cp
```

compRand *Compare Different Randomization Procedures via Tables and Plots*

Description

Compares randomization procedures based on several different quantities of imbalances. Among all included randomization procedures of class "careval", two or more procedures can be compared in this function.

Usage

```
compRand(...)
```

Arguments

...	objects of class "careval".

Details

The primary goal of using covariate-adaptive randomization in practice is to achieve balance with respect to the key covariates and to the overall treatment assignments. We choose four rules to measure the absolute imbalances at overall, marginal and withinn-stratum levels, which are maximal, 95%quantile, median and mean of the absolute imbalances at different aspects.

(1) Maximal

$$\max_{i=1,\ldots,n} |D_n(\cdot)|.$$

(2) 95% quantile

$$|D_{\lceil 0.95n \rceil}(\cdot)|.$$

(3) Median

$$(|D_n(\cdot)|) = |D_{(n+1)/2}(\cdot)|$$

for n is odd;

$$(|D_n(\cdot)|) = \frac{1}{2}(|D_{(n/2)}(\cdot)| + |D_{(n/2+1)}(\cdot)|)$$

for n is even.

(4) Mean

$$\frac{1}{n}\sum_{j=1}^{n} |D_j(\cdot)|.$$

The Monte Carlo method is used to calculate the four types of imbalances.

Value

It returns an object of class "carcomp".

The function print is used to obtain results. The generic accessor functions Assig, Diff, data, All strata and others extract various useful features of the value returned by compRand.

An object of class "carcomp" is a list containing at least the following components:

Overall Imbalances
: a matrix containing maximum, 95%-quantile, median, mean, and loss of absolute overall imbalances for all the input methods.

Marginal Imbalances
: a matrix containing maximum, 95%-quantile, median, mean, and loss of absolute marginal imbalances for all the input methods.

Within-stratum Imbalances
: a matrix containing maximum, 95%-quantile, median, mean, loss of absolute imbalances, and also containing mean absolute imbalances of the strata with i patients falling in, where $i = 1, \ldots, bsize$ for all the input methods.

References

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

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

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

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

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375.

See Also

See evalRand or evalRand.sim to evaluate a specific randomization procedure.

Examples

```
## Compare stratified permuted block randomization and Hu and Hu's general CAR
cov_num <- 2
level_num <- c(2, 2)
pr <- rep(0.5, 4)
n <- 500
N <- 20 # <<adjust according to CPU
bsize <- 4
# set weight for Hu and Hu's method, it satisfies
# (1)Length should equal to cov_num
```

```
omega <- c(1, 2, 1, 1)
# Assess Hu and Hu's general CAR
Obj1 <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                     level_num = level_num, pr = pr, method = "HuHuCAR",
                     omega, p = 0.85)
# Assess stratified permuted block randomization
Obj2 <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                     level_num = level_num, pr = pr, method = "StrPBR",
                     bsize)

RES <- compRand(Obj1, Obj2)
```

`corr.test` *Corrected t-test*

Description

Performs corrected t-test on treatment effects. This test follows the idea of Ma et al. (2015) <doi:10.1080/01621459.2014.922469>.

Usage

```
corr.test(data, conf = 0.95)
```

Arguments

`data`	a data frame. It consists of patients' profiles, treatment assignments and outputs. See `getData`.
`conf`	confidence level of the interval. The default is `0.95`.

Details

When the working model is the true underlying linear model, and the chosen covariate-adaptive design achieves that the overall imbalance and marginal imbalances for all covariates are bounded in probability, we can derive the asymptotic distribution under the null distribution, where the treatment effect of each group is the same. Subsequently, we can replace the variance estimator in a simple two sample t-test with an adjusted variance estimator. Details can be found in Ma et al.(2015).

Value

It returns an object of class `"htest"`.

The function `print` is used to obtain results. The generic accessor functions `statistic`, `p.value`, `conf.int` and others extract various useful features of the value returned by `corr.test`.

An object of class `"htest"` is a list containing at least the following components:

`data.name`	a character string giving the name(s) of the data.

statistic	the value of the t-statistic.
p.value	the p-value of the test,the null hypothesis is rejected if p-value is less than sl.
conf.int	a confidence interval under chosen level conf for the difference in treatment effect between treatment 1 and treatment 2.
estimate	estimated treatment effect difference between treatment 1 and treatment 2.
method	a character string indicating what type of test was performed.

References

Ma W, Hu F, Zhang L. *Testing hypotheses of covariate-adaptive randomized clinical trials*[J]. Journal of the American Statistical Association, 2015, 110(510): 669-680.

Examples

```
##generate data
set.seed(100)
n = 1000
cov_num = 5
level_num = c(2,2,2,2,2)
pr = rep(0.5,10)
beta = c(0.1,0.4,0.3,0.2,0.5)
omega = c(0.1, 0.1, rep(0.8 / 5, times = 5))
mu1 = 0
mu2 = 0.7
sigma = 1
type = "linear"
p = 0.85

dataH = getData(n,cov_num,level_num,pr,type,beta,
                mu1,mu2,sigma,"HuHuCAR",omega,p)

#run the corrected t-test
HHct=corr.test(dataH)
HHct
```

DoptBCD *Atkinson's D_A-optimal Biased Coin Design*

Description

Allocates patients to one of two treatments based on the D_A-optimal biased coin design with in the presence of the prognostic factors proposed by Atkinson A C (1982) <doi:10.2307/2335853>.

Usage

```
DoptBCD(data)
```

Arguments

data	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.

Details

To minimize the loss associated with an experiment involving n patients, Atkinson's optimal applied D_A-optimality to the method, in which the probability of assigning the $(n+1)$th patient to treatment 1 in the presence of prognostic factors is

$$\frac{[1-(1;\boldsymbol{x}_{n+1}^t)(\boldsymbol{F}_n^t\boldsymbol{F}_n)^{-1}\boldsymbol{b}_n]^2}{[1-(1;\boldsymbol{x}_{n+1}^t)(\boldsymbol{F}_n^t\boldsymbol{F}_n)^{-1}\boldsymbol{b}_n]^2+[1+(1;\boldsymbol{x}_{n+1}^t)(\boldsymbol{F}_n^t\boldsymbol{F}_n)^{-1}\boldsymbol{b}_n]^2},$$

where $\boldsymbol{X}=(\boldsymbol{x_i}, i=1,\ldots,n)$ and $\boldsymbol{x}_i=(x_{i1},\ldots,x_{in})$ denote the covariate profile of the ith patient; and $\boldsymbol{F}_n=[\boldsymbol{1}_n;\boldsymbol{X}]$ is the information matrix; and $\boldsymbol{b}_n^T=(2\boldsymbol{T}_n-\boldsymbol{1}_n)^T\boldsymbol{F}_n$, $\boldsymbol{T}_n=(T_1,\ldots,T_n)$ is a sequence containing the first n patients' allocations.

Details of the procedure can be found in A.C.Atkinson (1982).

Value

It returns an object of class "carandom".

The function print is used to obtain results. The generic accessor functions Cov_Assig, Diff, data, All strata and others extract various useful features of the value returned by DoptBCD.

An object of class "carandom" is a list containing at least the following components:

cov_num	the number of covariates.
n	the number of patients.
Cov_Assign	a (cov_num + 1) * n matrix containing covariate profiles for all patients and the corresponding assignments. The ith column represents the ith patient. The first cov_num rows include patients' covariate profiles and the last row contains the assignment.
All strata	a matrix containing all strata involved.
Diff	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
Data Type	the data type. Real or Simulated.

References

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

See Also

See DoptBCD.sim for allocating patients with covariate data generating mechanism. See DoptBCD.ui for the command-line user interface.

Examples

```
# a simple use
## Real Data
df <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 100, TRUE),
                 stringsAsFactors = TRUE)
Res <- DoptBCD(df)
## view the output
Res

## view all patients' profile and assignments
## Res$Cov_Assig

## Simulated Data
n <- 1000
cov_num <- 2

level_num <- c(2, 5)
# Set pr to follow two tips:
#(1) length of pr should be sum(level_num);
#(2)sum of probabilities for each margin should be 1.
pr <- c(0.4, 0.6, rep(0.2, times = 5))
Res.sim <- DoptBCD.sim(n, cov_num, level_num, pr)
## view the output
Res.sim

## view the difference between treatment 1 and treatment 2
##             at overall, within-strt. and overall levels
Res.sim$Diff


N <- 5
n <- 100
cov_num <- 2
level_num <- c(3, 5) # << adjust to your CPU and the length should correspond to cov_num
## Set pr to follow two tips:
## (1) length of pr should be sum(level_num);
## (2)sum of probabilities for each margin should be 1
pr <- c(0.3, 0.4, 0.3, rep(0.2, times = 5))
omega <- c(0.2, 0.2, rep(0.6 / cov_num, times = cov_num))

## generate a container to contain Diff
DH <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
DA <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
for(i in 1 : N){
  result <- HuHuCAR.sim(n, cov_num, level_num, pr, omega)
  resultA <- StrBCD.sim(n, cov_num, level_num, pr)
  DH[ , i] <- result$Diff; DA[ , i] <- resultA$Diff
}
## do some analysis
require(dplyr)
```

```
## analyze the overall imbalance
Ana_O <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_O) <- c("HuHuCAR", "DoptBCD")
colnames(Ana_O) <- c("mean", "median", "95%quantile")
temp <- DH[1, ] %>% abs
tempA <- DA[1, ] %>% abs
Ana_O[1, ] <- c((temp %>% mean), (temp %>% median),
                (temp %>% quantile(0.95)))
Ana_O[2, ] <- c((tempA %>% mean), (tempA %>% median),
                (tempA %>% quantile(0.95)))

## analyze the within-stratum imbalances
tempW <- DH[2 : (1 + prod(level_num)), ] %>% abs
tempWA <- DA[2 : 1 + prod(level_num), ] %>% abs
Ana_W <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_W) <- c("HuHuCAR", "DoptBCD")
colnames(Ana_W) <- c("mean", "median", "95%quantile")
Ana_W[1, ] = c((tempW %>% apply(1, mean) %>% mean),
               (tempW %>% apply(1, median) %>% mean),
               (tempW %>% apply(1, mean) %>% quantile(0.95)))
Ana_W[2, ] = c((tempWA %>% apply(1, mean) %>% mean),
               (tempWA %>% apply(1, median) %>% mean),
               (tempWA %>% apply(1, mean) %>% quantile(0.95)))

## analyze the marginal imbalance
tempM <- DH[(1 + prod(level_num) + 1) :
             (1 + prod(level_num) + sum(level_num)), ] %>% abs
tempMA <- DA[(1 + prod(level_num) + 1) :
               (1 + prod(level_num) + sum(level_num)), ] %>% abs
Ana_M <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_M) <- c("HuHuCAR", "DoptBCD")
colnames(Ana_M) <- c("mean", "median", "95%quantile")
Ana_M[1, ] = c((tempM %>% apply(1, mean) %>% mean),
               (tempM %>% apply(1, median) %>% mean),
               (tempM %>% apply(1, mean) %>% quantile(0.95)))
Ana_M[2, ] = c((tempMA %>% apply(1, mean) %>% mean),
               (tempMA %>% apply(1, median) %>% mean),
               (tempMA %>% apply(1, mean) %>% quantile(0.95)))

AnaHP <- list(Ana_O, Ana_M, Ana_W)
names(AnaHP) <- c("Overall", "Marginal", "Within-stratum")

AnaHP
```

DoptBCD.sim *Atkinson's D_A-optimal Biased Coin Design with Covariate Data Generating Mechanism*

Description

Allocates patients generated by simulating covariates-profile under the assumption of independence between covariates and levels within each covariate, to one of two treatments based on the D_A-optimal biased coin design in the presence of prognostic factors, as proposed by Atkinson A C (1982) <doi:10.2307/2335853>.

Usage

```
DoptBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
            pr = rep(0.5, 4))
```

Arguments

`n`	the number of patients. The default is `1000`.
`cov_num`	the number of covariates. The default is `2`.
`level_num`	a vector of level numbers for each covariate. Hence the length of `level_num` should be equal to the number of covariates. The default is `c(2,2)`.
`pr`	a vector of probabilities. Under the assumption of independence between covariates, `pr` is a vector containing probabilities for each level of each covariate. The length of `pr` should correspond to the number of all levels, and the sum of the probabilities for each margin should be `1`. The default is `rep(0.5,4)`, which corresponds to `cov_num = 2`, and `level_num = c(2,2)`.

Details

See `DoptBCD`.

Value

See `DoptBCD`.

References

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

See Also

See `DoptBCD` for allocating patients with complete covariate data; See `DoptBCD.ui` for the command-line user interface.

DoptBCD.ui *Command-line User Interface Using Atkinson's D_A-optimal Biased Coin Design*

Description

A call to the user-interface function used to allocate patients to one of two treatments using Atkinson's D_A-optimal biased coin design proposed by Atkinson A C (1982) <doi:10.2307/2335853>.

Usage

```
DoptBCD.ui(path, folder = "DoptBCD")
```

Arguments

path	the path in which a folder used to store variables will be created.
folder	name of the folder. If default, a folder named "DoptBCD" will be created.

Details

See DoptBCD.

Value

It returns an object of class "carseq".

The function print is used to obtain results. The generic accessor functions assignment, covariate, cov_num, cov_profile and others extract various useful features of the value returned by that function.

Note

This function provides a command-line user interface and users should follow the prompts to enter data including covariates, as well as levels for each covariate and the covariate profile of the new patient.

References

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

See Also

See DoptBCD for allocating patients with complete covariate data; See DoptBCD.sim for allocating patients with covariate data generating mechanism.

`evalPower` *Evaluation of Tests and Randomization Procedures through Power*

Description

Returns powers and a plot of the chosen test and method under different treatment effects.

Usage

```
evalPower(n, cov_num, level_num, pr, type, beta, di = seq(0,0.5,0.1), sigma = 1,
      Iternum, sl = 0.05, method = c("HuHuCAR", "PocSimMIN", "StrBCD", "StrPBR",
                                     "DoptBCD","AdjBCD"),
         test = c("rand.test", "boot.test", "corr.test"), plot = TRUE, ...)
```

Arguments

`n`	the number of patients.
`cov_num`	the number of covariates.
`level_num`	a vector of level numbers for each covariate. Hence the length of `level_num` should be equal to the number of covariates.
`pr`	a vector of probabilities. Under the assumption of independence between covariates, `pr` is a vector containing probabilities for each level of each covariate. The length of `pr` should correspond to the number of all levels, and the sum of the probabilities for each margin should be `1`.
`type`	a data-generating method. Optional input: `"linear"` or `"logit"`.
`beta`	a vector of coefficients of covariates. The length of `beta` must correspond to `cov_num`.
`di`	a value or a vector of values of difference in treatment effects. The default value is a sequence from `0` to `0.5` with increments of `0.1`. The value(s) forms the horizontal axis of the plot.
`sigma`	the error variance for the linear model. The default is 1. This should be a positive value and is only used when `type = linear`.
`Iternum`	an integer. It is the number of iterations required for power calculation.
`sl`	the significance level. If the p value returned by the test is less than `sl`, the null hypothesis will be rejected. The default value is `0.05`.
`method`	the randomization procedure to be used for power calculation. This package provides power calculation for `"HuHuCAR"`, `"PocSimMIN"`, `"StrBCD"`, `"StrPBR"`, `"AdjBCD"`, and `"DoptBCD"`.
`test`	a character string specifying the alternative tests used to verify hypothesis, must be one of `"boot.test"`, `"corr.test"` or `"rand.test"`, which are the randomization test, the bootstrap t test, and the corrected t test, respectively. The arguments associated with the testing function can be specified; otherwise, the default value will be used.

`plot` a bool. It indicates whether to plot or not. Optional input: `TRUE` or `FALSE`.

`...` arguments to be passed to `method`. These arguments depend on the randomization method used and the following arguments are accepted:

omega a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that `omega` is only needed when `HuHuCAR` is to be used.

weight a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that `weight` is only needed when `PocSimMIN` is to be used.

p the biased coin probability. `p` should be larger than `1/2` and less than `1`. Note that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter governing the degree of randomness. Note that `a` is only needed when `"AdjBCD"` is to be used.

bsize the block size for the stratified randomization. It is required to be a multiple of 2. Note that `bsize` is only needed when `"StrPBR"` is to be used.

B an integer. It is the number of bootstrap samples. It is needed only when `test` is `boot.test`.

Reps an integer. It is the number of randomized replications used in the randomization test. It is needed only when `test` is `rand.test`.

nthreads the number of threads to be used in parallel computation. This is needed only under `rand.test` and `boot.test`. The default is `1`.

Value

This function returns a list. The first element is a dataframe representing the powers of the chosen test under different values of treatment effects. The second element is the execution time. An optional element is the plot of power in which `di` forms the vertical axis.

Examples

```
##settings
set.seed(2019)
n = 100#<<for demonstration,it is suggested to be larger than 1000
cov_num = 5
level_num = c(2,2,2,2,2)
pr = rep(0.5,10)
beta = c(0.1,0.4,0.3,0.2,0.5)
omega = c(0.1, 0.1, rep(0.8 / 5, times = 5))
di = seq(0,0.5,0.1)
sigma = 1
type = "linear"
p = 0.85
Iternum = 10#<<for demonstration,it is suggested to be around 1000
sl = 0.05
Reps = 10#<<for demonstration,it is suggested to be 200

#Evaluation of Power
library("ggplot2")
```

```
Strtp=evalPower(n,cov_num,level_num,pr,type,beta,di,sigma,
                Iternum,sl,"HuHuCAR","rand.test",TRUE,omega,p,Reps, nthreads = 1)
Strtp
```

---

evalRand *Evaluation of Randomization Procedures*

Description

Evaluates a specific randomization procedure based on several different quantities of imbalances.

Usage

```
evalRand(data, method = "HuHuCAR", N = 500, ...)
```

Arguments

data	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
N	the iteration number. The default is 500.
method	the randomization procedure to be evaluated. This package provides assessment for "HuHuCAR", "PocSimMIN", "StrBCD", "StrPBR", "AdjBCD", and "DoptBCD".
...	arguments to be passed to method. These arguments depend on the randomization method assessed and the following arguments are accepted:

omega a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that omega is only needed when HuHuCAR is to be assessed.

weight a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that weight is only needed when PocSimMIN is to be assessed.

p the biased coin probability. p should be larger than 1/2 and less than 1. Note that p is only needed when "HuHuCAR", "PocSimMIN" and "StrBCD" are to be assessed.

a a design parameter governing the degree of randomness. Note that a is only needed when "AdjBCD" is to be assessed.

bsize the block size for stratified permuted block randomization. It is required to be a multiple of 2. Note that bsize is only needed when "StrPBR" is to be assessed.

Details

The data is designed for N times using method.

Value

It returns an object of class "careval".

The function print is used to obtain results. The generic accessor functions Assig, Diff, data, All strata and others extract various useful features of the value returned by evalRand.

An object of class "careval" is a list containing at least the following components:

N	the number of patients.
Assig	a n*N matrix containing assignments for each patient for N iterations.
Imb	a matrix containing maximum, 95%-quantile, median, and mean of absolute imbalances at overall, each within-stratum and each marginal levels. Note that, we refer users to the ith column of `All strata` for details of level $i, i = 1, \ldots,$str_num.
SNUM	a matrix with N colunms containing number of patients in each stratum for each iteration.
Data Type	the data type. Real or Simulated.

References

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

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

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

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

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375.

See Also

See evalRand.sim to evaluate a randomization procedure with covariate data generating mechanism.

Examples

```
# a simple use
## Access by real data
## create a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 1000, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 1000, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 1000, TRUE),
                 stringsAsFactors = TRUE)
Res <- evalRand(data = df, method = "HuHuCAR", N = 500,
```

```
                omega = c(1, 2, rep(1, ncol(df))), p = 0.85)
## view the output
Res

  ## view all patients' assignments
  Res$Assig

## Assess by simulated data
cov_num <- 3
level_num <- c(2, 3, 5)
pr <- c(0.35, 0.65, 0.25, 0.35, 0.4, 0.25, 0.15, 0.2, 0.15, 0.25)
n <- 1000
N <- 50
omega = c(1, 2, 1, 1, 2)
# assess Hu and Hu's procedure with the same group of patients
Res.sim <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                        level_num = level_num, pr = pr, method = "HuHuCAR",
                        omega, p = 0.85)

  ## Compare four procedures
  cov_num <- 3
  level_num <- c(2, 10, 2)
  pr <- c(rep(0.5, times = 2), rep(0.1, times = 10), rep(0.5, times = 2))
  n <- 100
  N <- 200 # <<adjust according to CPU
  bsize <- 4
  ## set weights for HuHuCAR
  omega <- c(1, 2, rep(1, cov_num));
  ## set weights for PocSimMIN
  weight = rep(1, cov_num);
  ## set biased probability
  p = 0.80
  # assess Hu and Hu's procedure
  RH <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                     level_num = level_num, pr = pr, method = "HuHuCAR",
                     omega = omega, p = p)
  # assess Pocock and Simon's method
  RPS <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                      level_num = level_num, pr = pr, method = "PocSimMIN",
                      weight, p = p)
  # assess Shao's procedure
  RS <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                     level_num = level_num, pr = pr, method = "StrBCD",
                     p = p)
  # assess stratified randomization
  RSR <- evalRand.sim(n = n, N = N, Replace = FALSE, cov_num = cov_num,
                      level_num = level_num, pr = pr, method = "StrPBR",
                      bsize)

  # create containers
  C_M = C_O = C_WS = matrix(NA, nrow = 4, ncol = 4)
  colnames(C_M) = colnames(C_O) = colnames(C_WS) =
    c("max", "95%quan", "med", "mean")
```

```
rownames(C_M) = rownames(C_O) = rownames(C_WS) =
  c("HH", "PocSim", "Shao", "StraRand")

# assess the overall imbalance
C_O[1, ] = RH$Imb[1, ]
C_O[2, ] = RPS$Imb[1, ]
C_O[3, ] = RS$Imb[1, ]
C_O[4, ] = RSR$Imb[1, ]
# view the result
C_O

# assess the marginal imbalances
C_M[1, ] = apply(RH$Imb[(1 + RH$strt_num) : (1 + RH$strt_num + sum(level_num)), ], 2, mean)
C_M[2, ] = apply(RPS$Imb[(1 + RPS$strt_num) : (1 + RPS$strt_num + sum(level_num)), ], 2, mean)
C_M[3, ] = apply(RS$Imb[(1 + RS$strt_num) : (1 + RS$strt_num + sum(level_num)), ], 2, mean)
C_M[4, ] = apply(RSR$Imb[(1 + RSR$strt_num) : (1 + RSR$strt_num + sum(level_num)), ], 2, mean)
# view the result
C_M

# assess the within-stratum imbalances
C_WS[1, ] = apply(RH$Imb[2 : (1 + RH$strt_num), ], 2, mean)
C_WS[2, ] = apply(RPS$Imb[2 : (1 + RPS$strt_num), ], 2, mean)
C_WS[3, ] = apply(RS$Imb[2 : (1 + RS$strt_num), ], 2, mean)
C_WS[4, ] = apply(RSR$Imb[2 : (1 + RSR$strt_num), ], 2, mean)
# view the result
C_WS

# Compare the four procedures through plots
meth = rep(c("Hu", "PS", "Shao", "STR"), times = 3)
shape <- rep(1 : 4, times = 3)
crt <- rep(1 : 3, each = 4)
crt_c <- rep(c("O", "M", "WS"), each = 4)
mean <- c(C_O[, 4], C_M[, 4], C_WS[, 4])
df_1 <- data.frame(meth, shape, crt, crt_c, mean,
                   stringsAsFactors = TRUE)

require(ggplot2)
p1 <- ggplot(df_1, aes(x = meth, y = mean, color = crt_c, group = crt,
                       linetype = crt_c, shape = crt_c)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  xlab("method") +
  ylab("absolute mean") +
  theme(plot.title = element_text(hjust = 0.5))
p1
```

`evalRand.sim` *Evaluation Randomization Procedures with Covariate Data Generating Mechanism*

Description

Evaluates randomization procedure based on several different quantities of imbalances by simulating patients' covariate profiles under the assumption of independence between covariates and levels within each covariate.

Usage

```
evalRand.sim(n = 1000, N = 500, Replace = FALSE, cov_num = 2,
             level_num = c(2, 2), pr = rep(0.5, 4), method = "HuHuCAR", ...)
```

Arguments

N	the iteration number. The default is 500.
n	the number of patients. The default is 1000.
Replace	a bool. If Replace = FALSE, the function does clinical trial design for N iterations for one group of patients. If Replace = TRUE, the function dose clinical trial design for N iterations for N different groups of patients.
cov_num	the number of covariates. The default is 2.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates. The default is c(2,2).
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1. The default is rep(0.5,4), which corresponds to cov_num = 2, and level_num = c(2,2).
method	the randomization procedure to be evaluated. This package provides assessment for "HuHuCAR", "PocSimMIN", "StrBCD", "StrPBR", "AdjBCD", and "DoptBCD".
...	arguments to be passed to method. These arguments depend on the randomization method assessed and the following arguments are accepted: **omega** a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that omega is only needed when HuHuCAR are to be assessed. **weight** a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that weight is only needed when PocSimMIN is to be assessed. **p** the biased coin probability. p should be larger than 1/2 and less than 1. Note that p is only needed when "HuHuCAR", "PocSimMIN" and "StrBCD" is to be assessed. **a** a design parameter governing the degree of randomness. Note that a is only needed when "AdjBCD" is to be assessed. **bsize** the block size for stratified permuted block randomization. It is required to be a multiple of 2. Note that bsize is only needed when "StrPBR" is to be assessed.

Details

See evalRand.

Value

See evalRand.

See Also

See evalRand to evaluate a randomization procedure with complete covariate data.

---

getData *Data Generation*

Description

Generates continuous or binary outcomes given patients' covariates, the underlying model and the randomization procedure.

Usage

```
getData(n, cov_num, level_num, pr, type, beta,
        mu1, mu2, sigma = 1, method = "HuHuCAR", ...)
```

Arguments

n	the number of patients.
cov_num	the number of covariates.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates.
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1.
type	a data-generating method. Optional input: "linear" or "logit".
beta	a vector of coefficients of covariates. The length of beta must correspond to cov_num.
mu1,mu2	main effects of treatment 1 and treatment 2.
sigma	the error variance for the linear model. The default is 1. This should be a positive value and is only used when type = linear.
method	the randomization procedure to be used for generating randomization sequences. This package provides data-generating function for "HuHuCAR", "PocSimMIN", "StrBCD", "StrPBR", "AdjBCD", and "DoptBCD".
...	arguments to be passed to method. These arguments depend on the randomization method used and the following arguments are accepted: **omega** a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that omega is only needed when HuHuCAR is to be used.

weight a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that `weight` is only needed when `PocSimMIN` is to be used.

p the biased coin probability. `p` should be larger than `1/2` and less than `1`. Note that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter governing the degree of randomness. Note that `a` is only needed when `"AdjBCD"` is to be used.

bsize the block size for stratified randomization. It is required to be a multiple of 2. Note that `bsize` is only needed when `"StrPBR"` is to be used.

Details

To generate continuous outcomes, we use the linear model:

$$y_i = \mu_j + x_i^T \beta + \epsilon_i,$$

to generate binary outcomes, we use the logit link function:

$$P(y_i = 1) = \frac{exp\{\mu_j + x_i^T \beta\}}{1 + exp\{\mu_j + x_i^T \beta}$$

,

where j indicates patient i belongs to treatment j.

Value

`getData` returns a size $cov_num + 2 \times n$ dataframe. The first `cov_num` rows represent patients' profile. The next row consists of patients' assignments and the final row consists of generated outcomes.

Examples

```
#Parameters' Setting
set.seed(100)
n = 1000
cov_num = 5
level_num = c(2,2,2,2,2)
beta = c(1,4,3,2,5)
mu1 = 0
mu2 = 0
sigma = 1
type = "linear"
p = 0.85
omega = c(0.1, 0.1, rep(0.8 / 5, times = 5))
pr = rep(0.5,10)

#Data Generation
dataH = getData(n, cov_num,level_num, pr, type, beta,
                mu1, mu2, sigma, "HuHuCAR", omega, p)
dataH[1:(cov_num+2),1:5]
```

HuHuCAR *Hu and Hu's General Covariate-Adaptive Randomization*

Description

Allocates patients to one of two treatments using Hu and Hu's general covariate-adaptive randomization proposed by Hu Y, Hu F (2012) <doi:10.1214/12-AOS983>.

Usage

```
HuHuCAR(data, omega = NULL, p = 0.85)
```

Arguments

`data`	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
`omega`	a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. If `omega = NULL` (default), the overall, within-stratum, and within-covariate-margin imbalances are weighted with porportions `0.2`, `0.3`, and `0.5/cov_num` for each covariate-margin, respectively, where `cov_num` is the number of covariates of interest.
`p`	the biased coin probability. `p` should be larger than `1/2` and less than `1`. The default is `0.85`.

Details

Consider I covariates and m_i levels for the ith covariate. T_j is the assignment of the jth ptient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of this patient. For convenience, $(k_1, \ldots, k_I)$ and $(i; k_i)$ denote the stratum and margin respectively. $D_n(.)$ is the difference between the numbers of assigned patients in treatment 1 and treatment 2 at the corresponding level after n patinets have been assigned.The general CAR procedure is as follows:

(1) The first patient is assigned to treatment 1 with probability $1/2$;

(2) Suppose that $n-1$ patients have been assigned to a treatment ($n > 1$), and the nth patient falls within $(k_1^*, \ldots, k_I^*)$;

(3)If the nth patient was assigned to treatment 1, then the potential overall, marginal, and within-stratum differences in the two groups are

$$D_n^{(1)} = D_{n-1} + 1$$

$$D_n^{(1)}(i; k_i^*) = D_{n-1}(i, k_i^*) + 1$$

$$D_n^{(1)}(k_1^*, \ldots, k_I^*) = D_n(k_1^*, \ldots, k_I^*) + 1.$$

Similarly, the potential differences if the nth patinent was assigned to treatment 1 would be obtained in the same way.

(4) An imbalance measure is defined by

$$Imb_n^{(l)} = \omega_0[D_n^{(1)}]^2 + \sum_{i=1}^{I} \omega_{m,i}[D_n^{(1)}(i; k_i^*)]^2 + \omega_s[D_n^{(1)}(k_1^*, \ldots, k_I^*)]^2, l = 1, 2;$$

(5)Conditional on the assignments of the first $(n-1)$ patients as well as the covariate profiles of the first n patients, assign the nth patient to treatment 1 with probability

$$P(T_n = 1|Z_n, T_1, \ldots, T_{n-1}) = q$$

for $Imb_n^{(1)} > Imb_n^{(2)}$,

$$P(T_n = 1|Z_n, T_1, \ldots, T_{n-1}) = p$$

for $Imb_n^{(1)} < Imb_n^{(2)}$, and

$$P(T_n = 1|Z_n, T_1, \ldots, T_{n-1}) = 0.5,$$

for $Imb_n^{(1)} = Imb_n^{(2)}$.

Value

It returns an object of class "carandom".

The function print is used to obtain results. The generic accessor functions Cov_Assig, Diff, data, All strata and others extract various useful features of the value returned by HuHuCAR.

An object of class "carandom" is a list containing at least the following components:

n	the number of patients.
cov_num	the number of covariates.
Cov_Assign	a (cov_num + 1) * n matrix containing covariate profiles for all patients and corresponding assignments. The ith column represents the ith patient. The first cov_num rows include a patient's covariate profile and the last row contains the assignment.
All strata	a matrix containing all strata involved.
Diff	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
Data Type	the data type. Real or Simulated.

References

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

See Also

See HuHuCAR.sim for allocating patients with covariate data generating mechanism. See HuHuCAR.ui for the command-line user interface.

Examples

```
# a simple use
## Real Data
## create a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 1000, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 1000, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 1000, TRUE),
                 stringsAsFactors = TRUE)
omega <- c(1, 2, rep(1, 3))
Res <- HuHuCAR(data = df, omega)
## view the output
Res

## view all patients' profile and assignments
Res$Cov_Assig

## Simulated data
cov_num <- 3
level_num <- c(2, 3, 3)
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, 0.4, 0.3, 0.3)
omega <- rep(0.2, times = 5)
Res.sim <- HuHuCAR.sim(n = 100, cov_num, level_num, pr, omega)
## view the output
Res.sim

## view the detials of difference
Res.sim$Diff


N <- 100 # << adjust according to your CPU
n <- 1000
cov_num <- 3
level_num <- c(2, 3, 5) # << adjust to your CPU and the length should correspond to cov_num
# Set pr to follow two tips:
#(1) length of pr should be sum(level_num);
#(2)sum of probabilities for each margin should be 1.
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, rep(0.2, times = 5))
omega <- c(0.2, 0.2, rep(0.6 / cov_num, times = cov_num))
# Set omega0 = omegaS = 0
omegaP <- c(0, 0, rep(1 / cov_num, times = cov_num))

## generate a container to contain Diff
DH <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
DP <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
for(i in 1 : N){
  result <- HuHuCAR.sim(n, cov_num, level_num, pr, omega)
  resultP <- HuHuCAR.sim(n, cov_num, level_num, pr, omegaP)
  DH[ , i] <- result$Diff; DP[ , i] <- resultP$Diff
}

## do some analysis
require(dplyr)
```

```
## analyze the overall imbalance
Ana_O <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_O) <- c("NEW", "PS")
colnames(Ana_O) <- c("mean", "median", "95%quantile")
temp <- DH[1, ] %>% abs
tempP <- DP[1, ] %>% abs
Ana_O[1, ] <- c((temp %>% mean), (temp %>% median),
                (temp %>% quantile(0.95)))
Ana_O[2, ] <- c((tempP %>% mean), (tempP %>% median),
                (tempP %>% quantile(0.95)))
## analyze the within-stratum imbalances
tempW <- DH[2 : (1 + prod(level_num)), ] %>% abs
tempWP <- DP[2 : 1 + prod(level_num), ] %>% abs
Ana_W <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_W) <- c("NEW", "PS")
colnames(Ana_W) <- c("mean", "median", "95%quantile")
Ana_W[1, ] = c((tempW %>% apply(1, mean) %>% mean),
               (tempW %>% apply(1, median) %>% mean),
               (tempW %>% apply(1, mean) %>% quantile(0.95)))
Ana_W[2, ] = c((tempWP %>% apply(1, mean) %>% mean),
               (tempWP %>% apply(1, median) %>% mean),
               (tempWP %>% apply(1, mean) %>% quantile(0.95)))

## analyze the marginal imbalance
tempM <- DH[(1 + prod(level_num) + 1) : (1 + prod(level_num) + sum(level_num)), ] %>% abs
tempMP <- DP[(1 + prod(level_num) + 1) : (1 + prod(level_num) + sum(level_num)), ] %>% abs
Ana_M <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_M) <- c("NEW", "PS"); colnames(Ana_M) <- c("mean", "median", "95%quantile")
Ana_M[1, ] = c((tempM %>% apply(1, mean) %>% mean),
               (tempM %>% apply(1, median) %>% mean),
               (tempM %>% apply(1, mean) %>% quantile(0.95)))
Ana_M[2, ] = c((tempMP %>% apply(1, mean) %>% mean),
               (tempMP %>% apply(1, median) %>% mean),
               (tempMP %>% apply(1, mean) %>% quantile(0.95)))

AnaHP <- list(Ana_O, Ana_M, Ana_W)
names(AnaHP) <- c("Overall", "Marginal", "Within-stratum")

AnaHP
```

`HuHuCAR.sim` *Hu and Hu's General Covariate-Adaptive Randomization with Covariate Data Generating Mechanism*

Description

Allocates patients to one of two treatments using general covariate-adaptive randomization proposed by Hu Y, Hu F (2012) <doi:10.1214/12-AOS983>, by simulating covariate profiles based on the assumption of independence between covariates and levels within each covariate.

Usage

```
HuHuCAR.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
            pr = rep(0.5, 4), omega = NULL, p = 0.85)
```

Arguments

n	the number of patients. The default is 1000.
cov_num	the number of covariates. The default is 2.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates. The default is c(2,2).
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1. The default is rep(0.5,4), which corresponds to cov_num = 2, and level_num = c(2,2).
omega	a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. If omega = NULL (default), the overall, within-stratum, and within-covariate-margin imbalances are weighted with porportions 0.2, 0.3, and 0.5/cov_num for each covariate-margin, respectively, where cov_num is the number of covariates of interest.
p	the biased coin probability. p should be larger than 1/2 and less than 1. The default is 0.85.

Details

See HuHuCAR.

Value

See HuHuCAR.

References

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

See Also

See HuHuCAR for allocating patients with complete covariate data; See HuHuCAR.ui for the command-line user interface.

HuHuCAR.ui *Command-line User Interface Using Hu and Hu's General Covariate-adaptive Randomization*

Description

A call to the user-iterface function used to allocate patients to one of two treatments using Hu and Hu's general covariate-adaptive randomization method as proposed by Hu Y, Hu F (2012) <doi:10.1214/12-AOS983>.

Usage

```
HuHuCAR.ui(path, folder = "HuHuCAR")
```

Arguments

path	the path in which a folder used to store variables will be created.
folder	name of the folder. If default, a folder named "HuHuCAR" will be created.

Details

See HuHuCAR

Value

It returns an object of class "carseq".

The function print is used to obtain results. The generic accessor functions assignment, covariate, cov_num, cov_profile and others extract various useful features of the value returned by HuHuCAR.ui.

Note

This function provides a command-line interface so that users should follow the prompts to enter data, including covariates as well as levels for each covariate, weights omega, biased probability p and the covariate profile of the new patient.

References

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

See Also

See HuHuCAR for allocating patients with complete covariate data; See HuHuCAR.sim for allocating patients with covariate data generating mechanism.

pats *Data of Covariate Profile of Patients*

Description

gives the simulated covariate profile of patients for clincal trials.

Usage

```
data(pats)
```

Arguments

pats	a data frame. Each row contains an individual's covariate profile and each column corresponds to a covariate. It contains the following columns **gender** Options are male and female. **employment status** Options are "unemployment" (unemp), "part time" (part.), "full time" (full.). **income** Options are >= 1w, <= 0.5w, 0.5~1w. **marriage status** Options are unmarried, married, divorced

PocSimMIN *Pocock and Simon's Method in the Two-Arms Case*

Description

Allocates patients to one of two treatments using Pocock and Simon's method proposed by Pocock S J, Simon R (1975) <doi:10.2307/2529712>.

Usage

```
PocSimMIN(data, weight = NULL, p = 0.85)
```

Arguments

data	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
weight	a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. If weight = NULL (default), the within-covariate-margin imbalances are weighted with an equal proportion, 1/cov_num, for each covariate-margin.
p	the biased coin probability. p should be larger than 1/2 and less than 1. The default is 0.85.

Details

Consider I covariates and m_i levels for the ith covariate. T_j is the assignment of the jth patient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of this patient. For convenience, $(k_1, \ldots, k_I)$ and $(i; k_i)$ denote the stratum and margin respectively. $D_n(.)$ is the difference between the numbers of assigned patients in treatment 1 and treatment 2 at the corresponding level after n patinets being assigned.The Pocock and Simon's procedure in the two-arms case is then as follows:

(1) The first patient is assigned to treatment 1 with probability $1/2$;

(2) Suppose that $n-1$ patients have been assigned to a treatment ($n > 1$) and the nth patient falls within $(k_1^*, \ldots, k_I^*)$;

(3)If the nth patient was assigned to treatment 1, then the potential marginal differences between the two groups are

$$D_n^{(1)}(i; k_i^*) = D_{n-1}(i, k_i^*) + 1.$$

Similarly, the potential differences would be obtained in the same way if the nth patinent was assigned to treatment 2.

(4) An imbalance measure is defined by

$$Imb_n^{(l)} = \sum_{i=1}^{I} \omega_{m,i} [D_n^{(1)}(i; k_i^*)]^2, l = 1, 2;$$

(5)Conditional on the assignments of the first $(n-1)$ patients as well as the covariate profiles of the first n patients, assign the nth patient to treatment 1 with the probability

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = q,$$

for $Imb_n^{(1)} > Imb_n^{(2)}$,

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = p,$$

for $Imb_n^{(1)} < Imb_n^{(2)}$, and

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = 0.5,$$

for $Imb_n^{(1)} = Imb_n^{(2)}$.

Value

It returns an object of `class` `"carandom"`.

The functions `print` is used to obtain results. The generic accessor functions `Cov_Assig`, `Diff`, `data`, `All strata` and othes extract various useful features of the value returned by `PocSimMIN`.

An object of class `"carandom"` is a list containing at least the following components:

`cov_num`	the number of covariates.
`n`	the number of patients.
`Cov_Assign`	a `(cov_num + 1) * n` matrix containing covariate profiles for all patients and the corresponding assignments. The ith column represents the ith patient. The first `cov_num` rows include patients' covariate profiles, and the last row contains the assignments.

`All strata`	a matrix containing all strata involved.
`Diff`	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
`Data Type`	the data type. `Real` or `Simulated`.

References

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

See Also

See `PocSimMIN.sim` for allocating patients with covariate data generating mechanism. See `PocSimMIN.ui` for the command-line user interface.

Examples

```
# a simple use
## Real Data
## creat a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 1000, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 1000, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 1000, TRUE),
                 stringsAsFactors = TRUE)
weight <- c(1, 2, 1)
Res <- PocSimMIN(data = df, weight)
## view the output
Res

## view all patients' profile and assignments
Res$Cov_Assig

## Simulated Data
cov_num = 3
level_num = c(2, 3, 3)
pr = c(0.4, 0.6, 0.3, 0.3, 0.4, 0.4, 0.3, 0.3)
Res.sim <- PocSimMIN.sim(n = 1000, cov_num, level_num, pr)
## view the output
Res.sim

## view the detials of difference
Res.sim$Diff


N <- 5
n <- 1000
cov_num <- 3
level_num <- c(2, 3, 5)
# Set pr to follow two tips:
# (1) length of pr should be sum(level_num);
# (2)sum of probabilities for each margin should be 1.
```

```
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, rep(0.2, times = 5))
omega <- c(0.2, 0.2, rep(0.6 / cov_num, times = cov_num))
weight <- c(2, rep(1, times = cov_num - 1))

## generate a container to contain Diff
DH <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
DP <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
for(i in 1 : N){
  result <- HuHuCAR.sim(n, cov_num, level_num, pr, omega)
  resultP <- PocSimMIN.sim(n, cov_num, level_num, pr, weight)
  DH[ , i] <- result$Diff; DP[ , i] <- resultP$Diff
}

## do some analysis
require(dplyr)

## analyze the overall imbalance
Ana_O <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_O) <- c("NEW", "PS")
colnames(Ana_O) <- c("mean", "median", "95%quantile")
temp <- DH[1, ] %>% abs
tempP <- DP[1, ] %>% abs
Ana_O[1, ] <- c((temp %>% mean), (temp %>% median),
                (temp %>% quantile(0.95)))
Ana_O[2, ] <- c((tempP %>% mean), (tempP %>% median),
                (tempP %>% quantile(0.95)))

## analyze the within-stratum imbalances
tempW <- DH[2 : (1 + prod(level_num)), ] %>% abs
tempWP <- DP[2 : 1 + prod(level_num), ] %>% abs
Ana_W <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_W) <- c("NEW", "PS")
colnames(Ana_W) <- c("mean", "median", "95%quantile")
Ana_W[1, ] = c((tempW %>% apply(1, mean) %>% mean),
               (tempW %>% apply(1, median) %>% mean),
               (tempW %>% apply(1, mean) %>% quantile(0.95)))
Ana_W[2, ] = c((tempWP %>% apply(1, mean) %>% mean),
               (tempWP %>% apply(1, median) %>% mean),
               (tempWP %>% apply(1, mean) %>% quantile(0.95)))

## analyze the marginal imbalance
tempM <- DH[(1 + prod(level_num) + 1) :
             (1 + prod(level_num) + sum(level_num)), ] %>% abs
tempMP <- DP[(1 + prod(level_num) + 1) :
               (1 + prod(level_num) + sum(level_num)), ] %>% abs
Ana_M <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_M) <- c("NEW", "PS")
colnames(Ana_M) <- c("mean", "median", "95%quantile")
Ana_M[1, ] = c((tempM %>% apply(1, mean) %>% mean),
               (tempM %>% apply(1, median) %>% mean),
               (tempM %>% apply(1, mean) %>% quantile(0.95)))
Ana_M[2, ] = c((tempMP %>% apply(1, mean) %>% mean),
               (tempMP %>% apply(1, median) %>% mean),
```

```
                (tempMP %>% apply(1, mean) %>% quantile(0.95)))

AnaHP <- list(Ana_O, Ana_M, Ana_W)
names(AnaHP) <- c("Overall", "Marginal", "Within-stratum")

AnaHP
```

PocSimMIN.sim *Pocock and Simon's Method in the Two-Arms Case with Covariate Data Generating Mechanism*

Description

Allocates patients to one of two treatments using Pocock and Simon's method proposed by Pocock S J, Simon R (1975) <doi:10.2307/2529712>, by simulating covariate profiles under the assumption of independence between covariates and levels within each covariate.

Usage

```
PocSimMIN.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
              pr = rep(0.5, 4), weight = NULL, p = 0.85)
```

Arguments

n	the number of patients. The default is 1000.
cov_num	the number of covariates. The default is 2.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates. The default is c(2,2).
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1. The default is rep(0.5,4), which corresponds to cov_num = 2, and level_num = c(2,2).
weight	a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. If weight = NULL (default), the within-covariate-margin imbalances are weighted with an equal proportion, 1/cov_num, for each covariate-margin.
p	the biased coin probability. p should be larger than 1/2 and less than 1. The default is 0.85.

Details

See PocSimMIN.

Value

See PocSimMIN.

References

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

See Also

See PocSimMIN for allocating patients with complete covariate data; See PocSimMIN.ui for the command-line user interface.

`PocSimMIN.ui` *Command-line User Interface Using Pocock and Simon's Procedure with Two-Arms Case*

Description

A call to the user-iterface function used to allocate patients to one of two treatments using Pocock and Simon's method proposed by Pocock S J, Simon R (1975) <doi:10.2307/2529712>.

Usage

```
PocSimMIN.ui(path, folder = "PocSimMIN")
```

Arguments

`path`	the path in which a folder used to storage variables will be created.
`folder`	name of the folder. If default, a folder named `"PocSimMIN"` will be created.

Details

See PocSimMIN.

Value

It returns an object of `class` `"carseq"`.

The function `print` is used to obtain results. The generic accessor functions `assignment`, `covariate`, `cov_num`, `cov_profile` and others extract various useful features of the value returned by `PocSimMIN.ui`.

Note

This function provides a command-line interface and users should follow the prompts to enter data including covariates as well as levels for each covariate, `weight`, biased probability `p` and the covariate profile of the new patient.

References

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

See Also

See PocSimMIN for allocating a given completely collected data; See PocSimMIN.sim for allocating patients with covariate data generating mechanism.

`rand.test` *Randomization Test*

Description

Performs randomization test on treatment effects.

Usage

```
rand.test(data, Reps = 200, method = c("HuHuCAR", "PocSimMIN", "StrBCD",
                                       "StrPBR", "DoptBCD", "AdjBCD"),
          conf = 0.95, binwidth = 30, ...)
```

Arguments

`data`	a data frame. It consists of patients' profiles, treatment assignments and outputs. See `getData`.
`Reps`	an integer. It is the number of randomized replications used in the randomization test. The default is `200`.
`method`	the randomization procedure to be used for testing. This package provides tests for `"HuHuCAR"`, `"PocSimMIN"`, `"StrBCD"`, `"StrPBR"`, `"AdjBCD"`, and `"DoptBCD"`.
`conf`	confidence level of the interval. The default is `0.95`.
`binwidth`	the number of bins for each bar in histogram. The default is 30.
`...`	arguments to be passed to `method`. These arguments depend on the randomization method used and the following arguments are accepted: **omega** a vector of weights at the overall, within-stratum, and within-covariate-margin levels. It is required that at least one element is larger than 0. Note that `omega` is only needed when `HuHuCAR` is to be used. **weight** a vector of weights for within-covariate-margin imbalances. It is required that at least one element is larger than 0. Note that `weight` is only needed when `PocSimMIN` is to be used. **p** the biased coin probability. `p` should be larger than `1/2` and less than `1`. Note that `p` is only needed when `"HuHuCAR"`,`"PocSimMIN"` and `"StrBCD"` are to be used. **a** a design parameter governing the degree of randomness. Note that `a` is only needed when `"AdjBCD"` is to be used. **bsize** the block size for stratified randomization. It is required to be a multiple of 2. Note that `bsize` is only needed when `"StrPBR"` is to be used.

Details

The randomization test is described as follows: 1) For the observed responses $Y_1, \ldots, Y_n$ and the treatment assignments $T_1, T_2, \ldots, T_n$, compute the observed test statistic

$$S_{obs} = \frac{-\sum_{i=1}^{n} Y_i * (T_i - 2)}{n_1} - \frac{\sum_{i=1}^{n} Y_i * (T_i - 1)}{n_0}$$

where n_1 is the number of patients assigned to treatment 1 and n_0 is the number of patients assigned to treatment 2;

2) Perform the covariate-adaptive randomization procedure to obtain the new treatment assignments and calculate the corresponding test statistic S_i. And repeat this process L times;

3) Calculate the two-sided Monte Carlo p-value estimator

$$p = \frac{\sum_{l=1}^{L} I(|S_l| \geq |S_{obs}|)}{L}$$

Value

It returns an object of class `"htest"`.

The function `print` is used to obtain results.The generic accessor functions `statistic`, `p.value` and others extract various useful features of the value returned by `rand.test`.

An object of class `"htest"` is a list containing at least the following components:

`data.name`	a character string giving the name(s) of the data.
`statistic`	the value of the t-statistic. As the randomization test is a nonparametric method, we cannot calculate the t-statistic, so it is hidden in this result.
`p.value`	p-value of the test, the null hypothesis is rejected if the p-value is less than `sl`.
`conf.int`	a confidence interval under the chosen level `conf` for the difference in treatment effect between treatment `1` and treatment `2`. As the randomization test is a non-parametric method, we cannot calculate the confidence interval, so it is hidden in this result.
`estimate`	the estimated difference in treatment effects between treatment `1` and treatment `2`.
`method`	a character string indicating what type of test was performed.

References

Rosenberger W F, Lachin J M. Randomization in clinical trials: *theory and practice*[M]. John Wiley & Sons, 2015.

Examples

```
##generate data
set.seed(100)
n = 1000
cov_num = 5
level_num = c(2,2,2,2,2)
```

```
pr = rep(0.5,10)
beta = c(0.1,0.4,0.3,0.2,0.5)
mu1 = 0
mu2 = 0.01
sigma = 1
type = "linear"
p = 0.85

dataS = getData(n, cov_num, level_num, pr, type,
               beta, mu1, mu2, sigma, "StrBCD", p)

#run the randomization test
library("ggplot2")
Strt = rand.test(data = dataS, Reps = 200,method = "StrBCD",
                conf = 0.95, binwidth = 30,
                p = 0.85)
Strt
```

StrBCD *Shao's Method in the Two-Arms Case*

Description

Allocates patients to one of the two treatments using Shao's method proposed by Shao J, Yu X, Zhong B (2010) <doi:10.1093/biomet/asq014>.

Usage

```
StrBCD(data, p = 0.85)
```

Arguments

data	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
p	the biased coin probability. p should be larger than 1/2 and less than 1. The default is 0.85.

Details

Consider I covaraites and m_i levels for the ith covariate. T_j is the assignment of the jth patient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of this patient. For convenience, $(k_1, \ldots, k_I)$ and $(i; k_i)$ denote the stratum and margin respectively. $D_n(.)$ is the difference between the numbers of assigned patients in treatment 1 and treatment 2 at the corresponding level after n patinets have been assigned.Then Shao's procedure is as follows:

(1) The first patient is assigned to treatment 1 with probability $1/2$;

(2) Suppose $n-1$ patients have each been assigned to a treatment ($n > 1$) and the nth patinent falls within $(k_1^*, \ldots, k_I^*)$;

(3)If the nth patient was assigned to treatment 1, then the potential within-stratum difference between the two groups is

$$D_n^{(1)}(k_1^*, \ldots, k_I^*) = D_n(k_1^*, \ldots, k_I^*) + 1.$$

Similarly, the potential differences would be obtained in the same way if the nth patinent was assigned to treatment 2.

(4) An imbalance measure is defined by

$$Imb_n^{(l)} = [D_n^{(1)}(k_1^*, \ldots, k_I^*)]^2, l = 1, 2;$$

(5)Conditional on the assignments of the first $(n-1)$ patients as well as the covariates'profiles of the first n patients, assign the nth patient to treatment 1 with probability

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = q,$$

for $Imb_n^{(1)} > Imb_n^{(2)}$,

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = p,$$

for $Imb_n^{(1)} < Imb_n^{(2)}$, and

$$P(T_n = 1 | Z_n, T_1, \ldots, T_{n-1}) = 0.5,$$

for $Imb_n^{(1)} = Imb_n^{(2)}$.

Value

It returns an object of class `"carandom"`.

The function `print` is used to obtain results. The generic accessor functions `Cov_Assig`, `Diff`, `data`, `All strata` and others extract various useful features of the value returned by `StrBCD`.

An object of class `"carandom"` is a list containing at least the following components:

`cov_num`	the number of covariates.
`n`	the number of patients.
`Cov_Assign`	a `(cov_num + 1) * n` matrix containing covariate profiles for all patients and corresponding assignments. The ith column represents the ith patient. The first `cov_num` rows include patients' covariate profiles, and the last row contains the assignment.
`All strata`	a matrix containing all the strata involved.
`Diff`	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
`Data Type`	the data type. `Real` or `Simulated`.

References

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

See Also

See StrBCD.sim for allocating patients with covariate data generating mechanism. See StrBCD.ui for command-line user interface.

Examples

```
# a simple use
## Real Data
## creat a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 1000, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 1000, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 1000, TRUE),
                 stringsAsFactors = TRUE)
Res <- StrBCD(data = df)
## view the output
Res

## view all patients' profile and assignments
Res$Cov_Assig

## Simulated Data
cov_num = 3
level_num = c(2, 3, 3)
pr = c(0.4, 0.6, 0.3, 0.4, 0.3, 0.4, 0.3, 0.3)
Res.sim <- StrBCD.sim(n = 1000, cov_num, level_num, pr)
## view the output
Res.sim

## view the detials of difference
Res.sim$Diff


N <- 5
n <- 1000
cov_num <- 3
level_num <- c(2, 3, 5)
# Set pr to follow two tips:
# (1) length of pr should be sum(level_num);
# (2)sum of probabilities for each margin should be 1
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, rep(0.2, times = 5))
omega <- c(0.2, 0.2, rep(0.6 / cov_num, times = cov_num))

## generate a container to contain Diff
DH <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
DS <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
for(i in 1 : N){
  result <- HuHuCAR.sim(n, cov_num, level_num, pr, omega)
  resultS <- StrBCD.sim(n, cov_num, level_num, pr)
  DH[ , i] <- result$Diff; DS[ , i] <- resultS$Diff
}

## do some analysis
```

```
require(dplyr)

## analyze the overall imbalance
Ana_O <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_O) <- c("NEW", "Shao")
colnames(Ana_O) <- c("mean", "median", "95%quantile")
temp <- DH[1, ] %>% abs
tempS <- DS[1, ] %>% abs
Ana_O[1, ] <- c((temp %>% mean), (temp %>% median),
                (temp %>% quantile(0.95)))
Ana_O[2, ] <- c((tempS %>% mean), (tempS %>% median),
                (tempS %>% quantile(0.95)))

## analyze the within-stratum imbalances
tempW <- DH[2 : (1 + prod(level_num)), ] %>% abs
tempWS <- DS[2 : 1 + prod(level_num), ] %>% abs
Ana_W <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_W) <- c("NEW", "Shao")
colnames(Ana_W) <- c("mean", "median", "95%quantile")
Ana_W[1, ] = c((tempW %>% apply(1, mean) %>% mean),
               (tempW %>% apply(1, median) %>% mean),
               (tempW %>% apply(1, mean) %>% quantile(0.95)))
Ana_W[2, ] = c((tempWS %>% apply(1, mean) %>% mean),
               (tempWS %>% apply(1, median) %>% mean),
               (tempWS %>% apply(1, mean) %>% quantile(0.95)))

## analyze the marginal imbalance
tempM <- DH[(1 + prod(level_num) + 1) :
              (1 + prod(level_num) + sum(level_num)), ] %>% abs
tempMS <- DS[(1 + prod(level_num) + 1) :
               (1 + prod(level_num) + sum(level_num)), ] %>% abs
Ana_M <- matrix(NA, nrow = 2, ncol = 3)
rownames(Ana_M) <- c("NEW", "Shao")
colnames(Ana_M) <- c("mean", "median", "95%quantile")
Ana_M[1, ] = c((tempM %>% apply(1, mean) %>% mean),
               (tempM %>% apply(1, median) %>% mean),
               (tempM %>% apply(1, mean) %>% quantile(0.95)))
Ana_M[2, ] = c((tempMS %>% apply(1, mean) %>% mean),
               (tempMS %>% apply(1, median) %>% mean),
               (tempMS %>% apply(1, mean) %>% quantile(0.95)))

AnaHP <- list(Ana_O, Ana_M, Ana_W)
names(AnaHP) <- c("Overall", "Marginal", "Within-stratum")

AnaHP
```

`StrBCD.sim` *Shao's Method in the Two-Arms Case with Covariate Data Generating Mechanism*

Description

Allocates patients to one of two treatments using Shao's method proposed by Shao J, Yu X, Zhong B (2010) <doi:10.1093/biomet/asq014>, by simulating covariate profiles under the assumption of independence between covariates and levels within each covariate.

Usage

```
StrBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), p = 0.85)
```

Arguments

`n`	the number of patients. The default is `1000`.
`cov_num`	the number of covariates. The default is `2`.
`level_num`	a vector of level numbers for each covariate. Hence the length of `level_num` should be equal to the number of covariates. The default is `c(2,2)`.
`pr`	a vector of probabilities. Under the assumption of independence between covariates, `pr` is a vector containing probabilities for each level of each covariate. The length of `pr` should correspond to the number of all levels, and the sum of the probabilities for each margin should be `1`. The default is `rep(0.5,4)`, which corresponds to `cov_num` = `2`, and `level_num` = `c(2,2)`.
`p`	the biased coin probability. `p` should be larger than `1/2` and less than `1`. The default is `0.85`.

Details

See `StrBCD`.

Value

See `StrBCD`.

References

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

See Also

See `StrBCD` for allocating patients with complete covariate data; See `StrBCD.ui` for the command-line user interface.

StrBCD.ui *Command-line User Interface Using Shao's Method*

Description

A call to the user-interface function used to allocate patients to one of two treatments using Shao's method proposed by Shao J, Yu X, Zhong B (2010) <doi:10.1093/biomet/asq014>.

Usage

```
StrBCD.ui(path, folder = "StrBCD")
```

Arguments

path	the path in which a folder used to storage variables will be created.
folder	name of the folder. If default, a folder named "StrBCD" will be created.

Details

See StrBCD.

Value

It returns an object of class "carseq".

The function print is used to obtain results. The generic accessor functions assignment, covariate, cov_num, cov_profile and others extract various useful features of the value returned by StrBCD.ui.

Note

This function provides a command-line interface and users should follow the prompts to enter data including covariates as well as levels for each covariate, biased probability p and the covariate profile of the new patient.

References

Shao J, Yu X, Zhong B. *A theory for testing hypotheses under covariate-adaptive randomization*[J]. Biometrika, 2010, 97(2): 347-360.

See Also

See StrBCD for allocating patients with complete covariate data; See StrBCD.sim for allocating patients with covariate data generating mechanism.

`StrPBR` *Stratified Permuted Block Randomization*

Description

Allocates patients to one of two treatments using stratified permuted block randomization proposed by Zelen M (1974) <doi:10.1016/0021-9681(74)90015-0>.

Usage

```
StrPBR(data, bsize = 4)
```

Arguments

`data`	a data frame. A row of the dataframe corresponds to the covariate profile of a patient.
`bsize`	the block size for stratified randomization. It is required to be a multiple of 2. The default is `4`.

Details

Different covariate profiles are defined to be strata, and then permuted block randomization is applied to each stratum. It works efficiently when the number of strata is small, but when the number of strata increases, the stratified permuted block randomization fails to obtain balance between two treatments.

`Permuted-block randomization`, or blocking, is used to balance treatment arms within a block so that there are the same number of subjects in each treatment arm. A block contains the same number of each treatment and blocks of different sizes are combined to make up the randomization list.

Value

It returns an object of `class` `"carandom"`.

The functions `print` is used to obtain results. The generic accessor functions `Cov_Assig`, `Diff`, `data`, `All strata` and others extract various useful features of the value returned by `StrPBR`.

An object of class `"carandom"` is a list containing at least the following components:

`cov_num`	the number of covariates.
`n`	the number of patients.
`Cov_Assign`	a `(cov_num + 1) * n` matrix containing covariate profiles for all patients and corresponding assignments. The ith column represents the ith patient. The first `cov_num` rows include patients' covariate profiles, and the last row contains the assignments.
`All strata`	a matrix containing all strata involved.
`Diff`	a matrix with only one column. There are final differences at the overall, within-stratum, and marginal levels.
`Data Type`	the data type. `Real` or `Simulated`.

References

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375.

See Also

See StrPBR.sim for allocating patients with covariate data generating mechanism. See StrPBR.ui for the command-line user interface.

Examples

```
# a simple use
## Real Data
## creat a dataframe
df <- data.frame("gender" = sample(c("female", "male"), 100, TRUE, c(1 / 3, 2 / 3)),
                 "age" = sample(c("0-30", "30-50", ">50"), 100, TRUE),
                 "jobs" = sample(c("stu.", "teac.", "others"), 100, TRUE),
                 stringsAsFactors = TRUE)
Res <- StrPBR(data = df, bsize = 4)
## view the output
Res

## view all patients' profile and assignments
Res$Cov_Assig

## Simulated data
cov_num <- 3
level_num <- c(2, 3, 3)
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, 0.4, 0.3, 0.3)
Res.sim <- StrPBR.sim(n = 100, cov_num, level_num, pr)
## view the output
Res.sim

## view the detials of difference
Res.sim$Diff


N <- 5
n <- 1000
cov_num <- 3
level_num <- c(2, 3, 5)
# Set pr to follow two tips:
#(1) length of pr should be sum(level_num);
#(2)sum of probabilities for each margin should be 1.
pr <- c(0.4, 0.6, 0.3, 0.4, 0.3, rep(0.2, times = 5))
omega <- c(0.2, 0.2, rep(0.6 / cov_num, times = cov_num))
# Set block size for stratified randomization
bsize <- 4

## generate a container to contain Diff
DS <- matrix(NA, ncol = N, nrow = 1 + prod(level_num) + sum(level_num))
for(i in 1 : N){
```

```
  rtS <- StrPBR.sim(n, cov_num, level_num, pr, bsize)
  DS[ , i] <- rtS$Diff
}

## do some analysis
require(dplyr)

## analyze the overall imbalance
Ana_O <- matrix(NA, nrow = 1, ncol = 3)
rownames(Ana_O) <- c("Str.R")
colnames(Ana_O) <- c("mean", "median", "95%quantile")
tempS <- DS[1, ] %>% abs
Ana_O[1, ] <- c((tempS %>% mean), (tempS %>% median),
                (tempS %>% quantile(0.95)))
## analyze the within-stratum imbalances
tempWS <- DS[2 : 1 + prod(level_num), ] %>% abs
Ana_W <- matrix(NA, nrow = 1, ncol = 3)
rownames(Ana_W) <- c("Str.R")
colnames(Ana_W) <- c("mean", "median", "95%quantile")
Ana_W[1, ] = c((tempWS %>% apply(1, mean) %>% mean),
               (tempWS %>% apply(1, median) %>% mean),
               (tempWS %>% apply(1, mean) %>% quantile(0.95)))

## analyze the marginal imbalance
tempMS <- DS[(1 + prod(level_num) + 1) : (1 + prod(level_num) + sum(level_num)), ] %>% abs
Ana_M <- matrix(NA, nrow = 1, ncol = 3)
rownames(Ana_M) <- c("Str.R");
colnames(Ana_M) <- c("mean", "median", "95%quantile")
Ana_M[1, ] = c((tempMS %>% apply(1, mean) %>% mean),
               (tempMS %>% apply(1, median) %>% mean),
               (tempMS %>% apply(1, mean) %>% quantile(0.95)))

AnaHP <- list(Ana_O, Ana_M, Ana_W)
names(AnaHP) <- c("Overall", "Marginal", "Within-stratum")

AnaHP
```

`StrPBR.sim` *Stratified Permuted Block Randomization with Covariate Data Generating Mechanism*

Description

Allocates patients to one of two treatments using stratified randomization proposed by Zelen M (1974) <doi:10.1016/0021-9681(74)90015-0>, by simulating covariates-profile on assumption of independence between covariates and levels within each covariate.

Usage

```
StrPBR.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), bsize = 4)
```

Arguments

n	the number of patients. The default is 1000.
cov_num	the number of covariates. The default is 2.
level_num	a vector of level numbers for each covariate. Hence the length of level_num should be equal to the number of covariates. The default is c(2,2).
pr	a vector of probabilities. Under the assumption of independence between covariates, pr is a vector containing probabilities for each level of each covariate. The length of pr should correspond to the number of all levels, and the sum of the probabilities for each margin should be 1. The default is rep(0.5,4), which corresponds to cov_num = 2, and level_num = c(2,2).
bsize	the block size for the stratified randomization. It is required to be a multiple of 2. The default is 4.

Details

See StrPBR.

Value

See StrPBR.

References

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375.

See Also

See StrPBR for allocating patients with complete covariate data; See StrPBR.ui for the command-line user interface.

StrPBR.ui *Command-line User Interface Using Stratified Permuted Block Randomization with Two-Arms Case*

Description

A call to the user-iterface function used to allocate patients to one of two treatments using stratified permuted block randomization proposed by Zelen M (1974) <doi: 10.1016/0021-9681(74)90015-0>.

Usage

```
StrPBR.ui(path, folder = "StrPBR")
```

Arguments

path	the path in which a folder used to storage variables will be created.
folder	name of the folder. If default, a folder named "StrPBR" will be created.

Details

See StrPBR.

Value

It returns an object of class "carseq".

The function print is used to obtain results. The generic accessor functions assignment, covariate, cov_num, cov_profile and others extract various useful features of the value returned by StrPBR.ui.

Note

This function provides a command-line interface and users should follow the prompts to enter data including covariates as well as levels for each covariate, block size bsize and the covariate profile of the new patient.

References

Zelen M. *The randomization and stratification of patients to clinical trials*[J]. Journal of chronic diseases, 1974, 27(7): 365-375.

See Also

See StrPBR for allocating patients with complete covariate data; See StrPBR.sim for allocating patients with covariate data generating mechanism.

Index