Package 'carat'

January 15, 2020

Type Package

Title Covariate-Adaptive Randomization for Clinical Trials

Version 0.1.0

Date 2019-12-17

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. It 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 (>= 1.0.1), ggplot2 (>= 3.1.1), gridExtra (>= 2.3), stringr (>= 1.4.0)

Suggests dplyr (>= 0.8.1)

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2020-01-15 10:20:02 UTC

R topics documented:

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

Description

Provides functions and command-line user interface to generate allocation sequence by covariate-adaptive randomization for clinical trials. It currently supports seven covariate-adaptive randomization procedures, such as stratified randomization, minimization, a general family of designs proposed by Hu and Hu (2012) <doi:10.1214/12-AOS983>. 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.

Author(s)

Fuyi Tu <fuyi.tu@ruc.edu.cn>;Xiaoqing Ye <ye_xiaoq@163.com>; Wei Ma <mawei@ruc.edu.cn>; Feifang Hu <ffhu68@163.com>.

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, 2019 (in press): 1-21. <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 proposed by Baldi Antognini A, Zagoraiou M (2011) <Doi:10.1093/biomet/asr021>.

Usage

```
## S3 method for class 'carandom'
AdjBCD(data, a = 2)
```

Arguments

`data`	a dataframe. A row of the dataframe contains the covariate profile of a certain patient.
`a`	a design parameter. Default is `2`. As `a` goes to ∞, the design becomes more deterministic.

Details

Consider I covaraites and m_i levels for the ith covariate. T_j is the assignment of jth ptient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of this patient. For convenience, $(k_1, \ldots, k_I), (i; k_i)$ denote stratum and margin respectively; and $D_n(.)$ is the difference between numbers of assigned patients in treatment 1 and treatment 2 at the corresponding level after n patinets being 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 $x = 0$;

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

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

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

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 so on extract various useful features of the value returned by that function.

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

`cov_num`	number of covariates.
`n`	number of patients.
`Cov_Assign`	a `(cov_num + 1) * n` matrix containing covariates' profile for all patients and corresponding assignments. The ith column represents the ith patient. The first `cov_num` rows include patient's covariate profile respectively; and the last row contains the assignment.
`All strata`	a matrix containing all strata involved.
`Diff`	a matrix with only 1 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 through simulating; See `AdjBCD.ui` for 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))
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 for Simulated data*

Description

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

Usage

```
## S3 method for class 'carandom'
AdjBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), a = 2)
```

Arguments

n	number of patients. Default is 1000.
cov_num	number of covariates. Default if 2.
level_num	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. The default is c(2,2).
pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num. If pr = rep(0.5,4) (default), it means cov_num = 2, and level_num = c(2,2).
a	a design parameter. Default is 2. As a goes to ∞, the design becomes more deterministic.

Details

See AdjBCD.

Value

See AdjBCD.

See Also

See AdjBCD for allocating a given completely collected data; See AdjBCD.ui for command-line user interface.

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

Description

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

Usage

```
## S3 method for class 'carseq'
AdjBCD.ui(path, folder = "AdjBCD")
```

Arguments

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

Details

See AdjBCD.

Value

It returns an object of class "carseq".

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

Note

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

See Also

See AdjBCD for allocating a given completely collected data; See AdjBCD.sim for allocating patients through simulating.

---

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 = HuHuCAR, conf = 0.95, ...)
```

Arguments

data	a dataframe, consisting of patients' profiles, treatment assignments and outputs. See getData.
B	integer, the number of bootstrap samples. Default is 200.
method	the randomization to be used in allocating patients. The default randomization HuHuCAR uses Hu and Hu's general covariate-adaptive randomization; the alternatives are PocSimMIN, StrBCD, StrPBR, DoptBCD as well as AdjBCD.
conf	confidence level of the interval. Default is 0.95.
...	arguments to be passed to methods. It depends on the method used and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that omega is only needed when HuHuCAR is to be used.

weight vector of weights for maginal imbalances. It is required that at least one element is larger than 0. Notice that `weight` is only needed when `PocSimMIN` is to be used.

p probabilities of assigning one patient to treatment 1. p should be larger than `1/2` to obtain balance. Notice that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter. As `a` goes to ∞, the design becomes more deterministic.

bsize block size for stratified randomization. It is required to be multiple of 2. Notice 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, Z_i denotes the profile of the ith patient.

2) Perform covariate-adaptive procedures on patients' profile, get 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, we can get B independent boostrap samples to obtain $\hat{\theta}_b^*$, $b = 1, \ldots, B$. Then the variance of $\bar{Y}_1 - \bar{Y}_0$ can 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 so on extract various useful features of the value returned by that function.

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`	value of the t-statistic.
`pval`	p-value of the test,the null hypothesis is rejected if p-value is less than `sl`.
`conf.int`	a confidence interval under chosen significance level `conf` for the treatment effect difference 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

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)))
##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))

#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	type list, each argument consists the power generated by evalPower in this package or other sources. The length of each argument must match.
diffs	type vector, values of differences in treatment effects. Its length and the length of each argument of powers must match.
testname	type vector, its element is the name of test and randomization method used. For example, when applying rand.test under HuHuCAR and corr.test under HuHuCAR, it can be c('HH.rand','HH.corr'). Its length 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 ones can be compared in this function.

Usage

```
## S3 method for class 'carcomp'
compRand(...)
```

Arguments

`...`	objects of class `"careval"`.

Details

We choose 5 rules to measure the absolute imbalances at all-mixed, overall, marginal and within-stratum levels, which are maximal, 95%quantile, median, mean and loss 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 \[median

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

for n is odd; median

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

for n is even. (4) Mean

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

(5) Loss

$$\frac{D_n^2(\cdot)}{n}.$$

All the 5 quntities at some aspect are calculated by taking average of all maximal, 95%quantile, median and mean of each level over N iterations.

Value

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

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

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

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

`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..

`plot` plots of the results.

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 <- 3
level_num <- c(2, 10, 10)
pr <- c(rep(0.5, times = 2), rep(0.1, times = 10), rep(0.1, times = 10))
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, 2, rep(1, times = cov_num - 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)
# compare through all absolute imbalances
RES$`All Imbalances`
# compare through absolute overall imbalances
RES$`Overall Imbalances`
# compare through absolute marginal imbalances
RES$`Marginal Imbalances`
# compare through absolute within-stratum imbalances
RES$`Marginal Imbalances`
```

```
# compare through plots
RES$plot
```

`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`	dataframe, consists of patients' profiles, treatment assignments and outputs. See `getData`.
`conf`	confidence level of the interval. Default is 0.95.

Details

When the working model is the true underlying linear model and the covariate-adaptive design used achieves that the overall imbalance as well as marginal imbalances for all covariates are bounded in probability, we can derive the asymptotic distribution under null distribution: treatment effects of groups are the same. Then we can replace the variance estimator in 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 so on extract various useful features of the value returned by that function.

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`	value of the t-statistic.
`p.value`	p-value of the test,the null hypothesis is rejected if p-value is less than `sl`.
`conf.int`	a confidence interval under chosen significance level `conf` for the treatment effect difference 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 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

```
## S3 method for class 'carandom'
DoptBCD(data)
```

Arguments

`data`	a dataframe. A row of the dataframe contains the covariate profile of a patient.

Details

In order 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 in the presence of prognostic factors of assigning the $(n+1)$th patient to treatment 1 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 $\boldsymbol{x}_i=(x_{i1},\ldots,x_{in})$ denoting the covariate-profile of ith patient; and $\boldsymbol{F}_n=[\boldsymbol{1}_n;\boldsymbol{X}]$ is the information matrix; $\boldsymbol{b}_n^T=(2\boldsymbol{T}_n-\boldsymbol{1}_n)^T\boldsymbol{F}_n$, $\boldsymbol{T}_n=(T_1,\ldots,T_n)$ which is a sequence containing the first n patients' allocations.

Detials of the the procedure can be seen in A.C.Atkinson (1982).

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 so on extract various useful features of the value returned by that function.

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

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

Description

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

Usage

```
## S3 method for class 'carandom'
DoptBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
            pr = rep(0.5, 4))
```

Arguments

n	number of patients. Default is 1000.
cov_num	number of covariates. Default is 2.
level_num	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. Default is level_num = c(2,2).
pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num. If pr = rep(0.5,4) (default), it means cov_num = 2, and level_num = c(2,2).

Details

See DoptBCD.

Value

See DoptBCD.

See Also

See DoptBCD for allocating a given completely collected data; See DoptBCD.ui for command-line user interface.

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

Description

A call to 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

```
## S3 method for class 'carseq'
DoptBCD.ui(path, folder = "DoptBCD")
```

Arguments

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

Details

See DoptBCD.

Value

It returns an object of class "carseq".

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

Note

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

See Also

See DoptBCD for allocating a given completely collected data; See DoptBCD.sim for allocating patients through simulating.

---

evalPower *Evaluation of Tests and Randomization Procedures through Power*

Description

Returns powers and a plot of 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 = HuHuCAR, test, plot = "TRUE", ...)
```

Arguments

n	number of patients.
cov_num	number of covariates.
level_num	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates.
pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num.

`type` way of generating data. Optional input: `linear` or `logit`.

`beta` vector of coefficients of covariates. The length of `beta` must correspond to `cov_num`.

`di` vector of values of difference in treatment effects. Default value is a sequence from `0` to `0.5` and the unit is `0.1`.

`sigma` error variance for linear model. Default value is 1. It is only used when `type` is `linear`.

`Iternum` integer, the number of iterations to calculate the average power.

`sl` significance level, if the p-value returned by the test is less than `sl`, we will reject the null hypothesis. Default value is `0.05`.

`method` the randomization to be used in allocating patients. The default randomization `HuHuCAR` uses Hu and Hu's general covariate-adaptive randomization; the alternatives are `PocSimMIN`, `StrBCD`, `StrPBR`, `DoptBCD` as well as `AdjBCD`.

`test` test used to verify hypothesis. Optional input: `rand.test`, `boot.test` or `corr.test`. They are randomization test, bootstrap t-test and corrected t-test respectively.

`plot` string, whether to plot or not. Optional input: `TRUE` or `FALSE`.

`...` arguments to be passed to methods. It depends on the `method` and `test` used and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that `omega` is only needed when `HuHuCAR` is to be used.

weight vector of weights for maginal imbalances. It is required that at least one element is larger than 0. Notice that `weight` is only needed when `PocSimMIN` is to be used.

p probabilities of assigning one patient to treatment 1. p should be larger than `1/2` to obtain balance. Notice that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter. As `a` goes to ∞, the design becomes more deterministic.

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

B integer,the number of bootstrap samplings. It is needed only when `test` is `boot.test`.

Pernum integer,the number of randomized replications. It is needed only when `test` is `rand.test`.

ncores number of cores used in parallel computation. It is needed only when `test` is `rand.test` or `boot.test`. It is suggested not to be greater than half the cores of servers or laptops.

Value

This function returns a list. The first element is a dataframe representing the powers of chosen test under different values of treatment effects. The second element of the list 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
Pernum = 10#<<for demonstration,it is suggested to be 200
ncores = 1

#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,Pernum,ncores)
Strtp
```

`evalRand` *Evaluation of Randomization Procedures*

Description

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

Usage

```
## S3 method for class 'careval'
evalRand(data, method = "HuHuCAR", N = 500, ...)
```

Arguments

`data`	a dataframe. A row of the dataframe contains the covariate profile of a patient.
`N`	iteration number.
`method`	the randomization to be used in allocating patients. The default randomization "`HuHuCAR`" uses Hu and Hu's general covariate-adaptive randomization; the alternatives are "`PocSimMIN`", "`StrBCD`", "`StrPBR`", "`DoptBCD`" as well as "`AdjBCD`".
`...`	arguments to be passed to methods. It depends on the `method` and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that `omega` is only needed when `HuHuCAR` is to be assessed.

weight vector of weights for all involved margins. It is required that at least one element is NOT 0 and `length(weight) = cov_num`. Notice that `weight` is only needed when `PocSimMIN` is to be assessed.

p probabilities of assigning one patinet to treatment 1. `p` should be larger than `1/2` to obtain balance. Notice that `p` is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be assessed.

a a design parameter. As `a` goes to ∞, the design becomes more deteministic. Notice that `a` is only needed when `"AdjBCD"` is to be assessed.

bsize block size for stratified permuted block randomization. It should be multiple of `2`. Notice that `bsize` is only needed when `"StrPBR"` is to be assessed.

Details

`data` is designed for `N` times using `method`.

Value

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

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

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

`N`	number of patients.
`Assig`	a `n*N` matrix containing assignments for each patient for N iterationsd.
`Imb`	matrix containing maximum, 95%-quantile, median, loss and mean of absolute imbalances at overall, each within-stratum and each marginal levels.
`Within-strt. by num of pats`	a vector containing mean absolute imbalances of the strata with i patients falling in, where $i = 1, \ldots, bsize$.
`Data Type`	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 by generating simulated data.

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))
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_All = C_M = C_O = matrix(NA, nrow = 4, ncol = 5)
colnames(C_All) = colnames(C_M) = colnames(C_O) =
  c("max", "95%quan", "med", "mean", "loss")
C_WS = matrix(NA, nrow = 4, ncol = bsize + 5)
colnames(C_WS) = c("max", "95%quan", "med", "mean", "loss",
                   "num = 1", "num = 2", "num = 3", "num = 4")
rownames(C_All) = rownames(C_M) = rownames(C_O) = rownames(C_WS) =
  c("HH", "PocSim", "Shao", "StraRand")

# access the imbalaces at overall, within-stratum, and marginal levels
C_All[1, ] = RH$`All Imbalances`
C_All[2, ] = RPS$`All Imbalances`
C_All[3, ] = RS$`All Imbalances`
C_All[4, ] = RSR$`All Imbalances`
# view the result
C_All

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

# assess the marginal imbalances
C_M[1, ] = RH$`Marginal Imbalances`
C_M[2, ] = RPS$`Marginal Imbalances`
C_M[3, ] = RS$`Marginal Imbalances`
C_M[4, ] = RSR$`Marginal Imbalances`
# view the result
C_M

# assess the within-stratum imbalances
C_WS[1, ] = RH$`Within-strt. Imbalances`
C_WS[2, ] = RPS$`Within-strt. Imbalances`
C_WS[3, ] = RS$`Within-strt. Imbalances`
C_WS[4, ] = RSR$`Within-strt. Imbalances`
# view the result
C_WS
```

```
# Compare the four procedures through plots
meth = rep(c("Hu", "PS", "Shao", "STR"), times = 4)
shape <- rep(1 : 4, times = 4)
crt <- rep(1 : 4, each = 4)
crt_c <- rep(c("All", "O", "M", "WS"), each = 4)
mean <- c(C_All[, 4], C_O[, 4], C_M[, 4], C_WS[, 4])
loss <- c(C_All[, 5], C_O[, 5], C_M[, 5], C_WS[, 5])
df_1 <- data.frame(meth, shape, crt, crt_c, mean, loss)

require(ggplot2)
require(gridExtra)
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))

# analyze within-stratum imbalances especially
df_2 <- as.data.frame(c(C_WS[, 6], C_WS[, 7], C_WS[, 8], C_WS[, 9]))
colnames(df_2) = "val"
df_2$num <- rep(c("num = 1", "num = 2", "num = 3", "num = 4"), each = 4)
df_2$meth <- meth

p2 <- ggplot(df_2, aes(x = num, y = val, color = meth, group = meth,
                       linetype = meth, shape = meth)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  xlab("numbers of patients for each strata") +
  ylab("absolute within-stratum mean") +
  theme(plot.title = element_text(hjust = 0.5))

grid.arrange(p1, p2, ncol = 1)
```

`evalRand.sim` *Evaluation Randomization Procedures Using Simulated Data*

Description

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

Usage

```
## S3 method for class 'careval'
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	iteration number.
n	number of patients. Default is 1000.
Replace	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	number of covariates. Default is 2.
level_num	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. Default is level_num = c(2,2).
pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num. If pr = (0.5,0.5,0.5,0.5) (default), it means cov_num = 2, and level_num = c(2,2).
method	the randomization to be used in allocating patients. The default randomization "HuHuCAR" uses Hu and Hu's general covariate-adaptive randomization; the alternatives are "PocSimMIN", "StrBCD", "StrPBR", "DoptBCD" as well as "AdjBCD".
...	arguments to be passed to methods. It depends on the method and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that omega is only needed when HuHuCAR are to be assessed.

weight vector of weights for all involved margins. It is required that at least one element is NOT 0 and length(weight) = cov_num. Notice that weight is only needed when PocSimMIN is to be assessed.

p probabilities of assigning one patinet to treatment 1. p should be larger than 1/2 to obtain balance. Notice that p is only needed when "HuHuCAR", "PocSimMIN" and "StrBCD" is to be assessed.

a a design parameter. As a goes to ∞, the design becomes more deteministic. Notice that a is only needed when "AdjBCD" is to be assessed.

bsize block size for stratified permuted block randomization. It should be multiple of 2. Notice 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 a given completely collected data.

getData *Data Generation*

Description

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

Usage

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

Arguments

n	number of patients.
cov_num	number of covariates.
level_num	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates.
pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num.
type	way of generating data. Optional input: linear or logit.
beta	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	error variance for linear model. Default is 1. It is only used when type is linear.
method	the randomization to be used in allocating patients. The default randomization HuHuCAR uses Hu and Hu's general covariate-adaptive randomization; the alternatives are PocSimMIN, StrBCD, StrPBR, DoptBCD as well as AdjBCD.
...	arguments to be passed to methods. It depends on the method used and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that omega is only needed when HuHuCAR is to be used.

weight vector of weights for maginal imbalances. It is required that at least one element is larger than 0. Notice that weight is only needed when PocSimMIN is to be used.

p probabilities of assigning one patient to treatment 1. p should be larger than 1/2 to obtain balance. Notice that p is only needed when "HuHuCAR","PocSimMIN" and "StrBCD" are to be used.

a a design parameter. As a goes to ∞, the design becomes more deterministic.

bsize block size for stratified randomization. It is required to be multiple of 2. Notice 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 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"
method = HuHuCAR
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

```
## S3 method for class 'carandom'
HuHuCAR(data, omega = NULL, p = 0.85)
```

Arguments

data	a dataframe or matrix. A row of the dataframe contains the covariate profile of some patient.
omega	vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. If omega = NULL (default), it weights the overall, within-stratum as well as marginal levels with porpotion 1/cov_num.
p	probabilities of assigning one patient to treatment 1. p should be larger than 1/2 to obtain balance. The default is 0.85.

Details

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

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

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

(3)The nth patient were assigned to treatment 1, then the potential overall, marginal as well as within-stratum differences in 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, potential differences if nth patinent were 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 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 $Imb_n^{(1)} < Imb_n^{(2)}$;

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

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 so on extract various useful features of the value returned by that function.

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

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

Description

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

Usage

```
## S3 method for class 'carandom'
HuHuCAR.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
            pr = rep(0.5, 4), omega = NULL, p = 0.85)
```

Arguments

`n`	number of patients. Default is `1000`.
`cov_num`	number of covariates. Default is `2`.
`level_num`	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. Default is `level_num = c(2,2)`.

`pr`	vector of probabilities. Under assumption of independence between covariates, `pr` is a vector containing probabilities for each levels of each covariates. The length of `pr` should correspond to number of all levels; and sum of `pr` should equal `cov_num`. If `pr = rep(0.5,4)` (default), it means `cov_num = 2`, and `level_num = c(2,2)`.
`omega`	vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. If `omega = NULL` (default), it weights the overall, within-stratum as well as marginal levels with porpotion `1/cov_num`.
`p`	probabilities of assigning one patient to treatment 1. p should be larger than `1/2` to obtain balance. The default is `0.85`.

Details

See `HuHuCAR`.

Value

See `HuHuCAR`.

See Also

See `HuHuCAR` for allocating a given completely collected data; See `HuHuCAR.ui` for command-line user interface.

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

Description

A call to user-iterface function used to allocate 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

```
## S3 method for class 'carseq'
HuHuCAR.ui(path, folder = "HuHuCAR")
```

Arguments

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

Details

See `HuHuCAR`

Value

It returns an object of class "carseq".

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

Note

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

See Also

See HuHuCAR for allocating a given completely collected data; See HuHuCAR.sim for allocating patients through simulating.

PocSimMIN *Pocock and Simon's Method in 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

```
## S3 method for class 'carandom'
PocSimMIN(data, weight = NULL, p = 0.85)
```

Arguments

data	a dataframe or matrix. A row of the dataframe contains the covariate profile of some patient.
weight	vector of weights for maginal imbalances. It is required that at least one element is larger than 0. If weight = NULL (default), it weights marginal imbalances with equal proportion 1/cov_num for each margin.
p	probabilities of assigning one patient to treatment 1. p should be larger than 1 / 2 to obtain balance. The default is 0.85.

Details

Consider I covaraites and m_i levels for the ith covariate. T_j is the assignment of jth ptient and $Z_j = (k_1, \ldots, k_I)$ indicates the covariate profile of this patient. For convenience, $(k_1, \ldots, k_I), (i; k_i)$ denote stratum and margin respectively; and $D_n(.)$ is the difference between 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 two-arms case is as follows:

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

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

(3)The nth patient were assigned to treatment 1, then the potential marginal differences in two groups are

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

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

(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 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 $Imb_n^{(1)} < Imb_n^{(2)}$;

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

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 so on extract various useful features of the value returned by that function.

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

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

`All strata`	a matrix containing all strata involved.
`Diff`	a matrix with only 1 column. There are final differences at the overall, within-stratum, and marginal levels.
`Data Type`	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 through simulating. See `PocSimMIN.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))
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 Two-arms Case for Simulated Data*

Description

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

Usage

```
## S3 method for class 'carandom'
PocSimMIN.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
              pr = rep(0.5, 4), weight = NULL, p = 0.85)
```

Arguments

`n`	number of patients. Default is `1000`.
`cov_num`	number of covariates. Default is `2`.
`level_num`	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. Default is `level_num = c(2,2)`.
`pr`	vector of probabilities. Under assumption of independence between covariates, `pr` is a vector containing probabilities for each levels of each covariates. The length of `pr` should correspond to number of all levels; and sum of `pr` should equal `cov_num`. If `pr = rep(0.5,4)` (default), it means `cov_num = 2`, and `level_num = c(2,2)`.
`weight`	vector of weights for maginal imbalances. It is required that at least one element is larger than 0. If `weight = NULL` (default), it weights marginal imbalances with equal proportion `1/cov_num` for each margin.
`p`	probabilities of assigning one patient to treatment 1. p should be larger than 1/2 to obtain balance. The default is `0.85`.

Details

See `PocSimMIN`.

Value

See `PocSimMIN`.

See Also

See PocSimMIN for allocating a given completely collected data; See PocSimMIN.ui for command-line user interface.

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

Description

A call to 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

```
## S3 method for class 'carseq'
PocSimMIN.ui(path, folder = "PocSimMIN")
```

Arguments

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

Details

See PocSimMIN.

Value

It returns an object of class "carseq".

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

Note

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

See Also

See PocSimMIN for allocating a given completely collected data; See PocSimMIN.sim for allocating patients through simulating.

print.carandom *Print Methods for Sequences, Evaluation and Compariason for Covariate-adaptive Randomization Procedures*

Description

Printing objects of class "carandom", "careval", "carcomp" or carseq respectively for different functions, by simple print method.

Usage

```
## S3 method for class 'carandom'
print(x, digits = getOption("digits"), prefix = "\t", ...)

## S3 method for class 'careval'
print(x, digits = getOption("digits"), prefix = "\t", ...)

## S3 method for class 'carcomp'
print(x, digits = getOption("digits"), prefix = "\t", ...)

## S3 method for class 'carseq'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

x	objects of class "carandom", "careval", "carcomp", or carseq.
digits	number of significant digits to be used.
prefix	string, passed to strwrap for displaying the method component of the carandom object, careval object, carcomp object and carseq object.
...	further arguments to be passed to or from methods.

Value

the argument x, invisibly, as for all print methods.

See Also

HuHuCAR, evalRand, compRand, HuHuCAR.ui

`rand.test` *Randomization Test*

Description

Performs randomization test on treatment effects.

Usage

```
rand.test(data, Pernum = 200, method = HuHuCAR, conf = 0.95, binwidth = 30, ...)
```

Arguments

`data`	a dataframe, consists of patients' profiles, treatment assignments and outputs. See `getData`.
`Pernum`	integer, the number of randomized replications. It is suggested to be 200.
`method`	the randomization to be used in allocating patients. The default randomization `HuHuCAR` uses Hu and Hu's general covariate-adaptive randomization; the alternatives are `PocSimMIN`, `StrBCD`, `StrPBR`, `DoptBCD` as well as `AdjBCD`.
`conf`	confidence level of the interval. Default is 0.95.
`binwidth`	number of bins for each bar in histogram. Default is 30.
`...`	arguments to be passed to methods. It depends on the `method` used and the following arguments are accepted:

omega vector of weights at the overall, within-stratum as well as maginal levels. It is required that at least one element is larger than 0. Notice that `omega` is only needed when `HuHuCAR` is to be used.

weight vector of weights for maginal imbalances. It is required that at least one element is larger than 0. Notice that `weight` is only needed when `PocSimMIN` is to be used.

p probabilities of assigning one patient to treatment 1. p should be larger than `1/2` to obtain balance. Notice that p is only needed when `"HuHuCAR"`, `"PocSimMIN"` and `"StrBCD"` are to be used.

a a design parameter. As a goes to ∞, the design becomes more deterministic.

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

Details

The randomization test is decribed 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, get new treatment assignments and calculate the corresponding test statistic S_i, repeat 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 so on extract various useful features of the value returned by that function.

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	value of the t-statistic. Since the randomization test is a nonparametric method, we can't calculate the t-statistic, so it is hidden in this result.
p.value	p-value of the test,the null hypothesis is rejected if p-value is less than sl.
conf.int	a confidence interval under chosen significance level conf for the treatment effect difference between treatment 1 and treatment 2. Since the randomization test is a nonparametric method, we can't calculate the confidence interval, so it is hidden in this result.
estimate	estimated treatment effect difference 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")
```

```
Pernum = 200
Strt=rand.test(dataS,Pernum,StrBCD,p=p)
Strt
```

StrBCD *Shao's Method in Two-arms Case*

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>.

Usage

```
## S3 method for class 'carandom'
StrBCD(data, p = 0.85)
```

Arguments

data	a dataframe. A row of the dataframe contains the covariate profile of some patient.
p	probabilities of assigning one patient to treatment 1. p should be larger than 1/2 to obtain balance. The default is 0.85.

Details

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

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

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

(3)The nth patient were assigned to treatment 1, then the potential within-stratum difference in two groups are

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

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

(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 $Imb_n^{(1)} < Imb_n^{(2)}$;

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

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 so on extract various useful features of the value returned by that function.

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

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

Description

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

Usage

```
## S3 method for class 'carandom'
StrBCD.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), p = 0.85)
```

Arguments

`n`	number of patients. Default is `1000`.
`cov_num`	number of covariates. Default is `2`.
`level_num`	vector of level numbers for each covariates. Hence the length of level_num should equal to number of covariates. Default is `level_num = c(2,2)`.

pr	vector of probabilities. Under assumption of independence between covariates, pr is a vector containing probabilities for each levels of each covariates. The length of pr should correspond to number of all levels; and sum of pr should equal cov_num. If pr = rep(0.5,4) (default), it means cov_num = 2, and level_num = c(2,2).
p	probabilities of assigning one patient to treatment 1. p should be larger than 1 / 2 to obtain balance. The default is 0.85.

Details

See StrBCD.

Value

See StrBCD.

See Also

See StrBCD for allocating a given completely collected data; See StrBCD.ui for command-line user interface.

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

Description

A call to user-iterface 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

```
## S3 method for class 'carseq'
StrBCD.ui(path, folder = "StrBCD")
```

Arguments

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

Details

See StrBCD.

Value

It returns an object of class "carseq".

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

Note

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

See Also

See StrBCD for allocating a given completely collected data; See StrBCD.sim for allocating patients through simulating.

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

```
## S3 method for class 'carandom'
StrPBR(data, bsize = 4)
```

Arguments

data	a dataframe. A row of the dataframe contains the covariate profile of a patient.
bsize	block size for stratified randomization. It is required to be multiple of 2. Default is 4.

Details

Different covariate-profiles are defined to be strata, and then permuted block randomization is applied to each strata. It works efficiently when the number of strata is small while with the number of strata increasing, 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. 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 so on extract various useful features of the value returned by that function.

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

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

Description

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

Usage

```
## S3 method for class 'carandom'
StrPBR.sim(n = 1000, cov_num = 2, level_num = c(2, 2),
           pr = rep(0.5, 4), bsize = 4)
```

Arguments

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

Details

See `StrPBR`.

Value

See `StrPBR`.

See Also

See `StrPBR` for allocating a given completely collected data; See `StrPBR.ui` for command-line user interface.

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

Description

A call to 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

```
## S3 method for class 'carseq'
StrPBR.ui(path, folder = "StrPBR")
```

Arguments

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

Details

See StrPBR.

Value

It returns an object of class "carseq".

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

Note

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

See Also

See StrPBR for allocating a given completely collected data; See StrPBR.sim for allocating patients through simulating.

Index