R documentation

of all in 'man/'

October 28, 2016

R topics documented:

BootstrapCI_AR — *Bootstrapping AR(1) dataset*

Description

An internal function to compute the confidence interval of resampled data using bootstrap methods

Usage

```
BootstrapCI_AR(bootN = 100, data, sim = 1)
```

Arguments

`bootN`	The number of resampling to be performed
`data`	the dataset to be evaluated
`sim`	the number of simulation performed

Value

The confidence interval of resampled data

`CI.beta`	95% Confidence Interval of beta
`CI.beta.emp`	95% Empirical Confidence Interval of beta
`CI.prev`	95% Confidence Interval of prevalence
`CI.prev.emp`	95% Empirical Confidence Interval of prevalence

BootstrapCI_CS — *Bootstrapping CS dataset*

Description

An internal function to compute the confidence interval of resampled data using bootstrap methods

Usage

```
BootstrapCI_CS(bootN = 100, data, sim = 1)
```

Arguments

`bootN`	The number of resampling to be performed
`data`	the dataset to be evaluated
`sim`	the number of simulation performed

Value

The confidence interval of resampled data

`CI.beta`	95% Confidence Interval of beta
`CI.beta.emp`	95% Empirical Confidence Interval of beta
`CI.prev`	95% Confidence Interval of prevalence
`CI.prev.emp`	95% Empirical Confidence Interval of prevalence

CalculateBetaEM_AR *EM estimation*

Description

Excecute EM algorithm to estimate beta

Usage

```
CalculateBetaEM_AR(data, x.cov, sens = 0.7, spec = 0.9, MaxIt = 2000,
  crt = 10^(-8))
```

Arguments

data	the dataset to be evaluated
x.cov	matrix of correctly measured covariates
sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
spec	Specificity = Prob(Y.obs = 0 | Y = 0)
MaxIt	maximum number of iteration
crt	criteria of convergence

Value

optimal beta estimated by EM algorithm

CalculateBetaEM_CS *EM estimation*

Description

Excecute EM algorithm to estimate beta

Usage

```
CalculateBetaEM_CS(data, x.cov, sens, spec, sens.x, spec.x, MaxIt = 2000,
  crt = 10^(-8))
```

Arguments

data	the dataset to be evaluated
x.cov	matrix of correctly measured covariates
sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
spec	Specificity = Prob(Y.obs = 0 | Y = 0)
sens.x	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
spec.x	Specificity.x = Prob(X1.obs = 0 | X1 = 0)
MaxIt	maximum number of iteration
crt	criteria of convergence

Value

optimal beta estimated by EM algorithm

---

calc_group_AR *separate group calculation*

Description

an internal function to extract data from different group and estimate beta's

Usage

```
calc_group_AR(grpdata, grp, maxit = 2000, crt = 10^(-8))
```

Arguments

grpdata	data from a specific group
grp	group index
maxit	maximum number of iteration
crt	criteria of convergence

Value

optimal beta estimated by EM algorithm and GLM

---

calc_grp_CS *separate group calculation*

Description

an internal function to extract data from different group and estimate beta's

Usage

```
calc_grp_CS(grpdata, grp, maxit = 2000, crt = 10^(-8))
```

Arguments

grpdata	data from a specific group
grp	group index
maxit	maximum number of iteration
crt	criteria of convergence

Value

optimal beta estimated by EM algorithm and GLM

`ConvertCov` *convert covariates*

Description

an internal function to convert x.cov to covariate matrix used in calculation

Usage

```
ConvertCov(x.cov)
```

Arguments

`x.cov`	matrix of correctly measured covariates

Value

processed covariate matrix used in calculation

`create_dummy_mat` *create dummy matrix Create dummy matrix for categorical variables*

Description

create dummy matrix Create dummy matrix for categorical variables

Usage

```
create_dummy_mat(x.cov = x.cov, cat.index = is.cat)
```

Arguments

`x.cov`	matrix of correctly measured covariates
`cat.index`	index of categorical variable(s)

Value

A dummy matrix for categorical variables dataAR_complete *Example of complete AR(1) STI dataset*

Description

Example of complete AR(1) STI dataset

Usage

```
data(dataAR_complete)
```

Format

An object of class `data.frame` with 1205 rows and 14 columns.

Examples

```
data(dataAR_complete)
head(dataAR_complete)
```

dataAR_user *Example of AR(1) STI dataset provided by user*

Description

Example of AR(1) STI dataset provided by user

Usage

```
data(dataAR_user)
```

Format

An object of class `data.frame` with 1145 rows and 11 columns.

Examples

```
data(dataAR_user)
head(dataAR_user)
```

dataCS_complete *Example of complete CS STI dataset*

Description

Example of complete CS STI dataset

Usage

```
data(dataCS_complete)
```

Format

An object of class `data.frame` with 1800 rows and 9 columns.

Examples

```
data(dataCS_complete)
head(dataCS_complete)
```

dataCS_user *Example of CS STI dataset provided by user*

Description

Example of CS STI dataset provided by user

Usage

```
data(dataCS_user)
```

Format

An object of class `data.frame` with 1500 rows and 7 columns.

Examples

```
data(dataCS_user)
head(dataCS_user)
```

`ExpY_AR` *probability expectation matrix*

Description

An internal function to compute the probability expectation matrix of four cases used in E step Define Four Cases: C1: y = 1; y(t-1) = 0 C2: y = 1; y(t-1) = 1 C3: y = 0; y(t-1) = 0 C4: y = 0; y(t-1) = 1

Usage

```
ExpY_AR(b, data, x.cov, sens = 0.7, spec = 0.9)
```

Arguments

`b`	the beta used to calculate negative log likelihood
`data`	the dataset to be evaluated
`x.cov`	matrix of correctly measured covariates
`sens`	Sensitivity = Prob(Y = 1 | Y.obs = 1)
`spec`	Specificity = Prob(Y.obs = 0 | Y = 0)

Value

The probability expectation matrix of given beta

`ExpY_CS` *probability expectation matrix of CS dataset*

Description

an internal function to compute the probability expectation matrix of four cases used in E step Define Four Cases: C1: y = 1; x1 = 0 C2: y = 1; x1 = 1 C3: y = 0; x1 = 0 C4: y = 0; x1 = 1

Usage

```
ExpY_CS(b, data, x.cov, sens, spec, sens.x, spec.x)
```

Arguments

`b`	the beta used to calculate negative log likelihood
`data`	the dataset to be evaluated
`x.cov`	matrix of correctly measured covariates
`sens`	Sensitivity = Prob(Y = 1 | Y.obs = 1)
`spec`	Specificity = Prob(Y.obs = 0 | Y = 0)
`sens.x`	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
`spec.x`	Specificity.x = Prob(X1.obs = 0 | X1 = 0)

Value

he expected probability of four cases for given beta

`Gradient_AR` *Gradient of AR(1) model*

Description

An internal function to omputes the gradient of objective function(nLL) respect to beta

Usage

```
Gradient_AR(b, data, x.cov, EY)
```

Arguments

`b`	the beta used to calculate negative log likelihood
`data`	the dataset to be evaluated
`x.cov`	matrix of correctly measured covariates
`EY`	the probability expectation matrix calculated in the ExpY function

Value

expected value of negative log likelihood

`Gradient_CS` *Gradient of CS model*

Description

An internal function to omputes the gradient of objective function(nLL) respect to beta

Usage

```
Gradient_CS(b, data, EY, x.cov)
```

Arguments

`b`	the beta used to calculate negative log likelihood
`data`	the dataset to be evaluated
`EY`	the probability expectation matrix calculated in the ExpY function
`x.cov`	matrix of correctly measured covariates

Value

expected value of negative log likelihood

NegativeLogLikelihood_AR

Negative Log-Likelihood of AR(1) model

Description

An internal function to compute the expected value of negative log likelihood function The E step in EM algorithm

Usage

```
NegativeLogLikelihood_AR(b, data, x.cov, EY)
```

Arguments

b	the beta used to calculate negative log likelihood
data	the dataset to be evaluated
x.cov	matrix of correctly measured covariates
EY	the probability expectation matrix calculated in the ExpY function

Value

expected value of negative log likelihood

NegativeLogLikelihood_CS

Negative Log-Likelihood

Description

An internal function to compute the expected value of negative log likelihood function The E step in EM algorithm

Usage

```
NegativeLogLikelihood_CS(b, data, EY, x.cov)
```

Arguments

b	the beta used to calculate negative log likelihood
data	the dataset to be evaluated
EY	the probability expectation matrix calculated in the ExpY function
x.cov	matrix of correctly measured covariates

Value

expected value of negative log likelihood

`parse_cov` *parse covariates*

Description

an internal function to parses the order of covariates

Usage

```
parse_cov(x.cov, vv.index)
```

Arguments

`x.cov`	original covariates matrix
`vv.index`	index of time-dependent variables

Value

A matrix of covariates in the order of [subID, time-dependent variable, fixed varlable, t]

`PerformanceEvaluation_AR` *Learning Performance Evaluation of AR(1) model*

Description

An internal function to Evaluate the performance of statistical methods for betas estimated by EM algorithm and GLM function. Performance measures including: Bias Mean Standard Deviation Percentage Bias Standard Bias Mean Square Error

Usage

```
PerformanceEvaluation_AR(beta.est, sim = simN, cvg = FALSE)
```

Arguments

`beta.est`	A matrix of betas estimated from different datasets
`sim`	the number of simulation performed
`cvg`	the option to find 95% CI coverage

Value

A dataframe of the performance of statistical methods for given set of betas

PerformanceEvaluation_CS *Learning Performance Evaluation of CS model*

Description

An internal function to Evaluate the performance of statistical methods for betas estimated by EM algorithm and GLM function. Performance measures including: Bias Mean Standard Deviation Percentage Bias Standard Bias Mean Square Error

Usage

```
PerformanceEvaluation_CS(beta.est, sim = simN, cvg = FALSE)
```

Arguments

beta.est	A matrix of betas estimated from different datasets
sim	the number of simulation performed
cvg	the option to find 95% CI coverage

Value

A dataframe of the performance of statistical methods for given set of betas

predict_AR *Predict STI probability*

Description

Predict STI infection probability based on provided dataset and patient info

Usage

```
predict_AR(data, patInfo, n.cov, n.grp, n.v, sens, spec, getBeta = TRUE,
  CI = FALSE, bootN = 10)
```

Arguments

data	dataset for beta estimation
patInfo	patient's diagonistic information
n.cov	number of covariates
n.grp	n.grp: number of testing technology
n.v	Number of time varying variable
sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
spec	Specificity = Prob(Y.obs = 0 | Y = 0)
getBeta	the option to save beta estimated from given dataset
CI	the option to get 95% confidence interval
bootN	the number of BootStrapping used to construct CI

Value

specific patient's infection probability

beta.est estimated beta from provided daaset

pat.CI.prev patient's 95% CI of prevalence

pat.CI.prev.emp patient's 95% empirical CI of prevalence

Examples

```
data <- simExample_AR()
patInfo <- data[5, ]
predict_AR(data,patInfo,4,3,1,c(0.8,0.85,0.9),c(0.9,0.85,0.8))
beta.est
```

---

predict_CS *Predict STI probability*

Description

Predict STI infection probability based on provided dataset and patient info

Usage

```
predict_CS(data, patInfo, n.cov, n.grp, sensitivity, specificity, sensitivity.x,
  specificity.x, getBeta = TRUE, CI = FALSE, bootN = 10)
```

Arguments

data	dataset for beta estimation
patInfo	patient's diagonistic information
n.cov	number of covariates
n.grp	n.grp: number of testing technology
sensitivity	Sensitivity = Prob(Y = 1 | Y.obs = 1)
specificity	Specificity = Prob(Y.obs = 0 | Y = 0)
sensitivity.x	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
specificity.x	Specificity.x = Prob(X1.obs = 0 | X1 = 0)
getBeta	the option to save beta estimated from given dataset
CI	the option to get 95% confidence interval
bootN	the number of BootStrapping used to construct CI

Value

pecific patient's infection probability

beta.est estimated beta from provided daaset

pat.CI.prev patient's 95% CI of prevalence

pat.CI.prev.emp patient's 95% empirical CI of prevalence

Examples

```
data <- simExample_CS(sampleN=500)
patInfo <- data[5, ]
predict_CS(data,patInfo,3,3,c(0.8,0.85,0.9),c(0.9,0.85,0.8),c(0.8,0.85,0.9),c(0.9,0.85,0.8))
beta.est
```

---

simExample_AR *Simulate an example of AR(1) dataset*

Description

Simulation an example for the users to prepare their data structure

Usage

```
simExample_AR(sampleN = 500, max.obs = 5, n.binom = 1, n.cat = 1,
  n.cont = 1, n.v = 1, n.grp = 3, sens = c(0.7, 0.8, 0.9),
  spec = c(0.9, 0.8, 0.7), seed = 616)
```

Arguments

sampleN	number of tuples in each group
max.obs	maximum number of obervation for each subject
n.binom	number of binomial covariates
n.cat	number of categorial covariates
n.cont	number of continuous covariates
n.v	number of varying variable
n.grp	number of testing technology
sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
spec	Specificity = Prob(Y.obs = 0 | Y = 0)
seed	the root seed of simulation

Value

an example of dataset

Examples

```
data <- simExample_AR()
head(data)
```

simExample_CS *Simulate an example of CS dataset*

Description

Simulation an example for the users to prepare their data structure

Usage

```
simExample_CS(sampleN = 100, sensitivity = c(0.8, 0.85, 0.9),
  specificity = c(0.9, 0.85, 0.8), sensitivity.x = c(0.8, 0.85, 0.9),
  specificity.x = c(0.9, 0.85, 0.8), n.grp = 3, n.binom = 1, n.cat = 1,
  n.cont = 1, seed = 616)
```

Arguments

sampleN	number of tuples in each group
sensitivity	Sensitivity = Prob(Y = 1 | Y.obs = 1)
specificity	Specificity = Prob(Y.obs = 0 | Y = 0)
sensitivity.x	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
specificity.x	Specificity.x = Prob(X1.obs = 0 | X1 = 0)
n.grp	number of testing technology
n.binom	number of binomial covariates
n.cat	number of categorial covariates
n.cont	number of continuous covariates
seed	the root seed of simulation

Value

an example of dataset

Examples

```
data <- simExample_CS()
head(data)
```

SimImpl_AR *Simulation Implementation*

Description

Implement simulation of AR(1) model

Usage

```
SimImpl_AR(simN = 3, sampleN = 500, max.obs = 5, n.binom = 0,
  n.cat = 0, n.cont = 0, n.v = 0, n.lev = 3, n.grp = 3,
  saveEst = TRUE, sens = c(0.7, 0.8, 0.9), spec = c(0.9, 0.8, 0.7),
  bootN = 10, seed = 616)
```

Arguments

simN	Number of dataset simulation
sampleN	Number of subjects in sample
max.obs	maximum number of obervation for each subject
n.binom	number of binomial covariates
n.cat	number of categorial covariates
n.cont	number of continuous covariates
n.v	number of varying (time dependent) variable
n.lev	number of levels for categorical covariate
n.grp	number of testing technology
saveEst	the option to save intermediate estimation
sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
spec	Specificity = Prob(Y.obs = 0 | Y = 0)
bootN	The number of resampling to be performed
seed	the root seed of simulation

Value

save simulation results to environment

b_AR	evaluation of GLM estimates for correctly measured Y
b.obs_AR	evaluation of GLM estimates for misclassified Y
b.EM_AR	evaluation of EM estimates for misclassified Y

Examples

```
SimImpl_AR()
```

SimImpl_CS *Simulation Implementation of CS model*

Description

Implement simulation of Cross-sectional model

Usage

```
SimImpl_CS(simN = 5, sampleN = 500, n.binom = 1, n.cat = 1,
  n.cont = 1, n.lev = 3, n.grp = 3, saveEst = TRUE,
  sensitivity = c(0.8, 0.85, 0.9), specificity = c(0.9, 0.85, 0.8),
  sensitivity.x = c(0.8, 0.85, 0.9), specificity.x = c(0.9, 0.85, 0.8),
  seed = 616, bootN = 2)
```

Arguments

`simN`	Number of dataset simulation
`sampleN`	Number of subjects in sample
`n.binom`	number of binomial covariates
`n.cat`	number of categorial covariates
`n.cont`	number of continuous covariates
`n.lev`	number of levels for categorical covariate
`n.grp`	number of testing technology
`saveEst`	the option to save intermediate estimation
`sensitivity`	Sensitivity = Prob(Y = 1 | Y.obs = 1)
`specificity`	Specificity = Prob(Y.obs = 0 | Y = 0)
`sensitivity.x`	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
`specificity.x`	Specificity.x = Prob(X1.obs = 0 | X1 = 0)
`seed`	the root seed of simulation
`bootN`	The number of resampling to be performed

Value

writes simulation results into csv file

`b_AR`	evaluation of GLM estimates for correctly measured X_1
`b.obs_AR`	evaluation of GLM estimates for misclassified X_1
`b.EM_AR`	evaluation of EM estimates for misclassified X_1

Examples

```
SimImpl_CS()
```

`simulate_AR` *simulate AR dataset*

Description

an internal function to simulates binary status and observations with misclassification

Usage

```
simulate_AR(n = sampleN, sens = 0.7, spec = 0.9, max.obs = 5,
  x.cov = x.cov, n.vv, subID = subID, k = 2836)
```

Arguments

`n`	Number of subjects in sample
`sens`	Sensitivity = Prob(Y = 1 | Y.obs = 1)
`spec`	Specificity = Prob(Y.obs = 0 | Y = 0)
`max.obs`	Maximum number of obervation for each subject
`x.cov`	matrix of correctly measured covariates
`n.vv`	number of varying (time dependent) variable
`subID`	ID for each subject
`k`	Seed for simulation

Value

A dataframe including subject ID, binary status and observations with misclassification

---

simulate_CS *Simulate CS dataset*

Description

an internal function to simulate binary status and observations with misclassification

Usage

```
simulate_CS(n = sampleN, p = 0.5, x.cov = x.cov, sub.sens, sub.spec,
  sub.sens.x, sub.spec.x, k = 2836)
```

Arguments

n	Number of subjects in sample
p	probability of X_1 to be 1
x.cov	matrix of correctly measured covariates
sub.sens	Sensitivity = Prob(Y = 1 | Y.obs = 1)
sub.spec	Specificity = Prob(Y.obs = 0 | Y = 0)
sub.sens.x	Sensitivity.x = Prob(X1 = 1 | X1.obs = 1)
sub.spec.x	Specificity.x = Prob(X1.obs = 0 | X1 = 0)
k	Seed for simulation

Value

A dataframe including subject ID, binary status and observations with misclassification

---

sim_covariates_AR *simulate covariates for AR(1) model*

Description

Simulates correctly measured covariates

Usage

```
sim_covariates_AR(n = n, n.binom = 1, n.cat = 1, n.cont = 1, n.vv = 1,
  n.lev = 3, max.obs = 5, grpIdx, subID, p = 0.6, p.cat = c(0.1, 0.2,
  0.7), k = 2812)
```

Arguments

n	Number of subjects in sample
n.binom	number of correctly measured binary covariates
n.cat	number of correctly measured categorical covariates
n.cont	number of correctly measured continuous covariates
n.vv	number of varying (time dependent) variable
n.lev	number of levels for categorical covariate
max.obs	Maximum number of obervation for each subject
grpIdx	group index for each subject
subID	ID for each subject
p	probability of binary covariate to be 1
p.cat	probability of different levels of categorical covariate
k	Seed for simulation

Value

A dataframe including correctly measured covariates

subID	ID for each subject
grpIdx	group index for each subject
X_n	correctly measured covariates
t	time

---

sim_covariates_CS   *simulate covariates for CS model*

Description

an internal function to simulate correctly measured covariates

Usage

```
sim_covariates_CS(n = sampleN, n.binom = 1, n.cat = 1, n.cont = 1,
  n.lev = 3, grpIdx, subID, p = 0.6, p.cat = c(0.1, 0.2, 0.7), k = 2812)
```

Arguments

n	Number of subjects in sample
n.binom	number of correctly measured binary covariates
n.cat	number of correctly measured categorical covariates
n.cont	number of correctly measured continuous covariates
n.lev	number of levels for categorical covariate
grpIdx	group index for each subject
subID	ID for each subject
p	probability of binary covariate to be 1
p.cat	probability of different levels of categorical covariate
k	Seed for simulation

Value

A dataframe including correctly measured covariates

subID	ID for each subject
grpIdx	group index for each subject
X_n	correctly measured covariates

Index