mcperturb Function Documentation

mcperturb webpage:
https://github.com/RyanZam1030/mcperturb/

Ryan Zamora | rz1030@yahoo.com

Shuying Sun | ssun5211@yahoo.com

Last Updated on October 19, 2019

Table of Contents

densPlots

Observational Strategies

The function *densPlots* displays the density function for the mean-centered column vectors of the **X**-matrix.

Usage

densPlots(xmat, meanCent = FALSE, na.rm = TRUE)

Arguments

Xmat	A nummeric design matrix with the equal row lengths
meanCent	Wether the columns of the matrix are mean cantered, default = FALSE
na.rm	Whether to remove missing observations, if missing values will be filled

Note

This function is made to plot the variables on one plot. In order for this function to work, the length of each column vector must be the same. Therefore, if there exist a missing value, then the whole observation must be removed.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

densPlots(xmat=x, meanCent=TRUE)
```

implausStats

Observational Strategies

The function *implausStats* takes in a matrix of variables, correlates each variable with the response, performs a simple linear regression (SLR) model with each variable, performs a multiple linear regression model (MLR) using all of the variables, outputs a summary table of correlations, SLR statistics, and MLR statistics.

Usage

```
implausStats(xmat, response)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector

Note

This function is made to calculate correlations and SLR and MLR model statistics. Any inconsistencies between the statistics should be investigated.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

implausStats(xmat = x, response = y)
```

---

rsqdPlots

Observational Strategies

The function rsqdPlots takes in a matrix of numeric regressors and a numeric response variable, ranks the regressors by their Pearson correlations with the response. Then performs a regression models by adding one regressor at a time. Sequentially adding the highest in magnitude correlated regressor to lowest. Returns a plot of r-squared or adjusted r-squared values .

Usage

```
rsqdPlots(xmat, response, adjrsq = FALSE)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
adjrsq	Logical. If TRUE, the adjusted r-squared values will be calculated and plotted

Note

This function is made to sequentially perform Regression models by including each variable one at a time. The order that the variables are included into the model is based on their correlation with response.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

rsqdPlots(x,y)

rsqdPlots(x,y,TRUE)
```

boxplotAllPerc

Perturbation Analysis

The function boxplotAllPerc takes in a matrix of numeric regressors and a numeric response variable, a list of noise levels, noise variables, and amount of iterations. Perturbs the noise regressor for n-iterations at every noise level and calculates the multicollinearity diagnostic measures and Linear regression statistics and plots the distribution with respect to all noise levels.

Usage

```
boxplotAllPerc(xmat = x, response = y, noiseLevs = noiseLevs, special.Vars = special.Vars,
iteration = iteration)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
special.Vars	A list of noise variables to perturb
iteration	An integer

Note

This function is made to out put the boxplots to a working directory. The directory it outputs the plots to will be printed.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

# Noise Variable

special.Var = “shoulder”

# Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

BoxplotAllPerc(xmatrix = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration =
iteration)
```

boxplotAllVars

Perturbation Analysis

The function boxplotAllVars takes in a matrix of numeric regressors and a numeric response variable, a list of noise levels, noise variables, and amount of iterations. Perturbs the noise regressor for n-iterations at every noise level and calculates the multicollinearity diagnostic measures and Linear regression statistics and plots the distribution with respect to all regressors.

Usage

boxplotAllVars(xmat = x, response = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration, path = NULL)

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
special.Vars	A list of noise variables to perturb
iteration	An integer
path	A character vector or string of the output path

Note

This function is made to out put the boxplots to a working directory or path. The directory it outputs the plots to will be printed.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

# Noise Variable
```

```
special.Var = "shoulder"

# Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

boxplotsAllVars(xmat = x, response = y, noiseLevs = noiseLevs, special.Vars = special.Vars,
iteration = iteration)
```

---

overallDiagsPlots

Perturbation Analysis

The function overallDiagPlots takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and amount of iterations. Perturbs the selected noise regressor for n-iterations at every noise level and calculates the overall multicollinearity diagnostic measures. The function returns a boxplot of the distributions.

Usage

```
overallDiagPlots (xmat, response, special.Vars, noiseLevs, iteration, choice = c())
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
special.Vars	A list of regressors to perturb
noiseLevs	A list or a sequence of numeric noise levels
iteration	An integer
choice	A character vector with the first letter of the diagnostic.

Note

This function is made to display the distributions of an overall diagnostic as the noise level and special variable changes. The comparison to the original measurement should be observed. This function is dependent on the cran "mctest" package. This function calls on the overallDiagsDiffs function to calculate the difference statistics for plotting, the mctest::omcdiag function to calculate the overall multicollinearity diagnostic measures.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

special.Vars = c("shoulder")

#Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

overallDiagsPlots(xmat = x, response = y, noiseLevs = noiseLevs, special.Vars = special.Vars,
iteration = iteration, choice = c("d"))
```

---

noiseLevelDiagOutList

Perturbation Analysis

The function noiseLevelDiagOutList takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and amount of iterations. Perturbs the regressor sequentially for n-iterations at every noise level and calculates the multicollinearity diagnostic measures. The function returns an object with a list of the different diagnostics at every noise step and a name for every noise level.

Usage

```
noiseLevelDiagOutList(xmat, response, special.Vars, noiseLevs, iteration)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
iteration	An integer

Note

This function is made to output a list of summary tables with all of the model statistics arranged at different noise levels. This function acts as a supporting function.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

special.Vars = c("shoulder")

#Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

noiseLevelDiagOutList(xmat = x, response = y, special.Vars = special.Vars, noiseLevels =
noiseLevs, iteration = iteration)
```

IsBestFit

Perturbation Analysis

The function isBestFit takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and amount of iterations. Perturbs the regressor sequentially for n-iterations at every noise level and calculates the multicollinearity diagnostic measures then calculates the least squares best fit line using the noise levels as the factors and the calculated diagnostic as the response variable.

Usage

```
isBestFit(xmat = x, response = y, noiseLevs = noiseLevs, iteration = iteration)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
iteration	An integer

Note

This function is made to output a list of summary tables with the best fit line for each regressors as it is being perturb. Coupling regressors should be identified.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

# Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05
```

```
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

bestfitvalues = isBestFit(xmat = x, response = y, noiseLevs = noiseLevs,iteration = iteration)

vifBestFit = bestfitvalues[[1]]
```

IsRateofChange

Perturbation Analysis

The function isRateofChange takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and amount of iterations. Perturbs the regressor sequentially for n-iterations at every noise level and calculates the multicollinearity diagnostic measures then calculates the average rate of change per diagnostic with respect to the difference between the maximum and minimum noise level.

Usage

```
isRateofChange(xmat = x, response = y, noiseLevs = noiseLevs, iteration = iteration)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
iteration	An integer

Note

This function is made to output a list of summary tables with the average rate of change values for each regressors as it is being perturb. Coupling regressors should be identified.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
```

```
# Response Variable

y = body_Dim[,23]

# Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

rateofchangevalues = isRateofChange(xmat = x, response = y, noiseLevs = noiseLevs, iteration =
iteration)

vifRateofChange = rateofchangevalues[[1]]
```

---

overallDiagsRank

Perturbation Analysis

The function isRateofChange takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and amount of iterations. Perturbs the regressor sequentially for n-iterations at every noise level and calculates the multicollinearity diagnostic measures then calculates the average rate of change per diagnostic with respect to the difference between the maximum and minimum noise level.

Usage

```
isRateofChange(xmat = x, response = y, noiseLevs = noiseLevs, iteration = iteration)
```

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
noiseLevs	A list or a sequence of numeric noise levels
iteration	An integer

Note

This function is made to output a list of summary tables with the average rate of change values for each regressors as it is being perturb. Coupling regressors should be identified.

Examples

```
# Body dimensions data

data(body_Dim)

# X-matrix

x = body_Dim[,c(10, 11, 16, 17, 21, 22, 24)]

colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")

# Response Variable

y = body_Dim[,23]

# Making the noiselevels

noiseStart = 0.05

noiseEnd = 0.25

noiseSteps = 0.05

noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)

iteration =  50

rateofchangevalues = isRateofChange(xmat = x, response = y, noiseLevs = noiseLevs, iteration =
iteration)

vifRateofChange = rateofchangevalues[[1]]
```