Documentation of *mcperturb*

mcperturb webpage: https://github.com/ss355/mcperturb

Shuying Sun | ssun5211@yahoo.com or ssun@txstate.edu

Ryan Zamora | rz1030@yahoo.com

Last Updated on April 9, 2020

Table of Contents

Important notes

1. It is better to set a random seed using *set.seed* before running any mcperturb functions and to make sure the results are reproducible to make it easy to debug.
2. It is better use one noise setting, i.e., one set of noise start, noise end, and noise step (e.g., noise start =0.05, noise end =0.5, and noise step=10), if you want to diagnose a specific dataset. *In the following examples scripts, sometimes we use different sets of noise levels for the sake of simple/easy demonstration. For example, sometimes we use a short list of noise levels (e.g., only two noise levels) and a small number of iterations in order to show a short list of output results.*

Notes/Reminder for authors:

R scripts and example output of this R documentation and debug notes (from Mar 19 – 27, 2020) are saved in *Mar19.2020.Scripts.for.R.Documentation.txt.*

Overall/general description of the *mcperturb* package

The main goal of the *mcperturb* package is to diagnose multicollinearity. In order to achieve this goal, we execute a perturbation strategy/analysis and include an observational strategy/analysis. Performing perturbation analysis before calculating diagnostic measures can provide a dynamic element to the otherwise static information obtained by calculating diagnostic measures. That is, performing multicollinearity diagnostic measures (static) after applying small perturbations to the regressors (dynamic) may lead to a more in-depth analysis of a multicollinearity problem. In addition to the perturbation analysis, including the observational analysis is one of the new contributions of this *mcperturb* package as it is not included in the currently available software packages. The *mcperturb* package diagnoses multicollinearity by calculating both the overall and individual diagnostic measures after perturbation. This package consists of 5-steps as shown in Table 1, and Table 2 lists all R functions of the *mcperturb* package.

The *mcperturb* package is a structured approach that combines both the observational strategy/analysis and perturbation strategy/analysis. It begins with the observational strategy/analysis, and then combines a perturbation technique with the overall or individual diagnostic measures. It systematically perturbs the regressors sequentially at different noise levels before calculating the diagnostic measures. The *mcperturb* package includes functions designed specifically for each step of the 5-step multicollinearity diagnostic procedure. It generates output summary tables, graphs, or boxplots for interpretation. Performing the 5-step multicollinearity diagnostic procedure can generate a plethora of information, especially when including a large number of regressors for analysis. To make it easier for the users, we will provide examples to show how to utilize the *mcperturb* package and interpret the results.

Table 1. 5-steps of the *mcperturb* package

Steps	**Diagnostic/Purpose**	**R Functions**	**Arguments**
1. Perform observational analysis	Identify regressors with similar density functions, implausible coefficients, inflated standard errors, and little impact on the R^2	*densPlots, implausStats, rsqdPlots*	x - matrix of regressors, y - response variable
2. Perform perturbation analysis	Add small amounts of noise to each regressor at multiple levels	*noiseLevelDiagOutList*	x - matrix of regressors, y - response variable, i - # of iterations, n - # of noise levels
3. Calculate the overall/individual diagnostic measures and plot their distributions	Observe how small perturbation affects the diagnostic measures	*overallDiagsPlots, boxplotAllVars, boxplotAllPercent*	x - matrix of regressors, y - response variable, i - # of iterations, n - # of noise levels, p - path
4. Conduct summary analysis for each regressor and calculate the rate of change	Summarize the max, min, and difference values for each diagnostic and rate of change	*overallDiagOut, rateOfChange*	x - matrix of regressors, y - response variable, i - # of iterations, n - # of noise levels
5. Rank the overall diagnostics and/or identify coupling regressors	Rank the overall diagnostics by their impact on the model and identify coupling regressors.	*overallDiagsRank*, isR*ateOfChange*, *isBestFit*	x - matrix of regressors, y - response variable, i - # of iterations, n - # of noise levels

Table 2. Function documentation overview

	Index	Function Name	Short summary	Dependencies
Main Function	1	*densPlots*	Makes the density plots for regressors	None
	2	*implausStats*	Calculate the SLR, MLR Coeff and Std. err, and correlations	None
	3	*rsqdPlots*	Calculates the r-squared value as a single regressor is added into the MLR model	None
	4	*noiseLevelDiagOutList*	Returns a list of all diagnostics at different noise levels, with multiple iterations	*regModelStats, randomNoiseMat, omcdiag, imcdiag*
	5	*overallDiagsPlots*	Outputs boxplots for each overall diagnostic at different noise levels	*overallDiagsDiffs*
	6	*boxplotAllVars*	Boxplot distributions for each variable, for each diagnostic, at individual noise levels	*diagout*
	7	*boxplotAllPerc*	Boxplots distributions for individual variables and diagnostics at all noise levels	*diagout*
	8	*overallDiagOut*	Outputs a table of Min, max, and difference for each overall diagnostic	*noiseLevelDiagOutList*
	9	*rateOfChange*	Outputs the original difference, Leastsquares, and rate of change	*diagout*
	10	*overallDiagsRank*	Outputs a summary table of Ranking for each variable per diagnostic	*overallDiagsDiffs*
	11	*israteOfChange*	Outputs rate of change per diagnostic and variable *"couplingTables"*	*rateOfChange*
	12	*isBestFit*	Output least squares best fit per diagnostic and variable	*rateOfChange*
Support Function	1	*regModelStats*	Calculates the regression model statistics	None
	2	*randomNoiseMat*	Perturbs the data by adding noise to regressors in a matrix	*rnorm*
	3	*noiseLevelsList*	Makes a list of the different noise levels to perturb	*randomNoiseMat*
	4	*diagout*	Organizes all the diagnostics into a list of lists	*noiseLevelsList*
	5	*overallDiagsDiffs*	Calculates the difference between the overall diagnostic measures as noise is added to regressors	*noiseLevelDiagOutList*
	6	*varDecomp*	Calculate the variance decomposition proportions	None

The relationships of 18 R functions are shown in the following Figure 1.

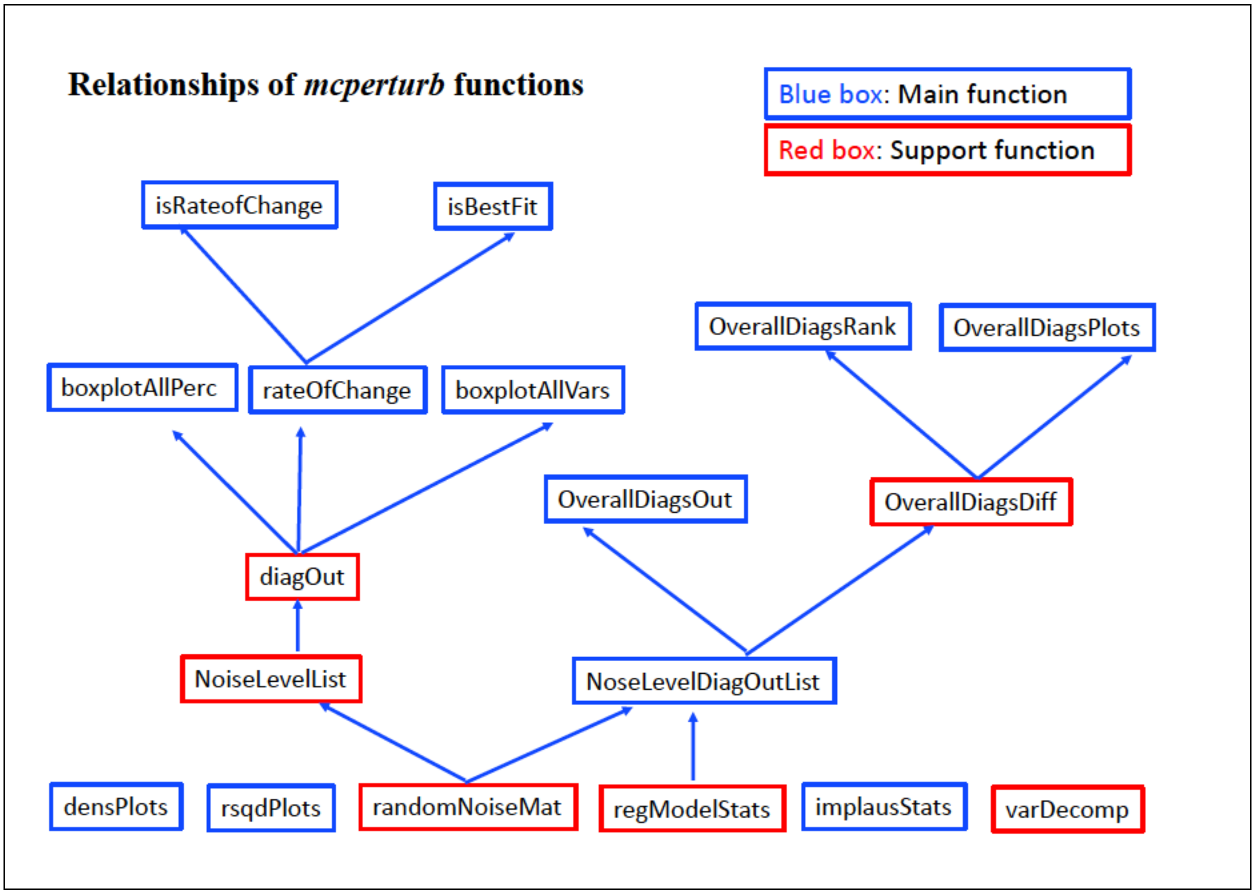


Figure 1: Relationships of mcperturb functions. The arrow from function A to B means that function B calls A or is dependent on A. Function with a blue box is a main function in the 5 -step perturbation procedure/analysis and a function with a red box mean that it is a support function.

Body dimension dataset overview

The body dimension dataset used for analysis is published as an observational study (Grete et al., 2003). The dataset is collected by the original authors, Grete Heinz and Louis J. Peterson, at San Jose State University and at the U.S. Naval Postgraduate School in Monterey California. The authors investigated relationships between individual's body frame size, frame girths, and weight of active adults in the military. Because multiple body measurements are performed on the same individual who participated in the study, the high correlation between variables is inevitable.

The original body dimension dataset consists of 25 variables (body measurements) and 507 observations (profiles). From the 25 body measurements, the weight measurement is selected as the response variable and the shoulder diameter, chest girth, bicep girth, forearm girth, wrist minimum girth, height, and age variables are selected as the regressors for this thesis project. Example code to read in the variables used for the function documentation examples is described below.

R-code for reading in the dataset and assigning variable names

```
# Reading in the data
# The body_dat.txt can be downloaded from http://jse.amstat.org/datasets/body.dat.txt
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
# Response variable
y = body_dat[,23]
# X-matrix
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
head(x)
```

```
> head(x)
  shoulder chest bicep forearm wrist age height
1    106.2  89.5  32.5    26.0  16.5  21  174.0
2    110.5  97.0  34.4    28.0  17.0  23  175.3
3    115.1  97.5  33.4    28.8  16.9  28  193.5
```

Preparation

```
#######################################################
# First, read all R functions of mcperturb package into R
#######################################################
code.dir =
"/Users/Shuying/Desktop/0.Current.Research/7.Ryan/Work.Record/Mar9.2020.JSS.ms/R.code/mcperturb_12_30
_19"

source(paste(code.dir, "densPlots.R", sep = "/"))
source(paste(code.dir, "diagOut.R", sep = "/"))
source(paste(code.dir, "implausStats.R", sep = "/"))
source(paste(code.dir, "rsqdPlots.R", sep = "/"))
source(paste(code.dir, "randomNoiseMat.R", sep = "/"))
source(paste(code.dir, "regModelStats.R", sep = "/"))
source(paste(code.dir, "noiseLevelDiagOutList.R", sep = "/"))
source(paste(code.dir, "noiseLevelsList.R", sep = "/"))
source(paste(code.dir, "overallDiagsDiffs.R", sep = "/"))
source(paste(code.dir, "overallDiagsout.R", sep = "/"))
source(paste(code.dir, "overallDiagsRank.R", sep = "/"))
source(paste(code.dir, "overallDiagsPlots.R", sep = "/"))
source(paste(code.dir, "boxplotsAllVars.R", sep = "/"))
source(paste(code.dir, "BoxplotAllPerc.R", sep = "/"))
source(paste(code.dir, "rateOfChange.R", sep = "/"))
source(paste(code.dir, "isBestFit.R", sep = "/"))
source(paste(code.dir, "isRateofChange.R", sep = "/"))
source(paste(code.dir, "varDecomp.R", sep = "/"))

#######################################################
# Second, install related R packages
#######################################################
```

```
install.packages("corrgram");     library(corrgram)
install.packages("perturb") ;     library(perturb)
install.packages("mctest") ;      library(mctest)
install.packages("car");          library(car)
install.packages("readr");        library(readr)
save.image()
```

Step 1: densPlots

Observational Strategies

The function *densPlots* displays the density function for the mean-centered column vectors of the **X**-matrix. (i.e., for each predictor/regressor)

Usage

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

Arguments

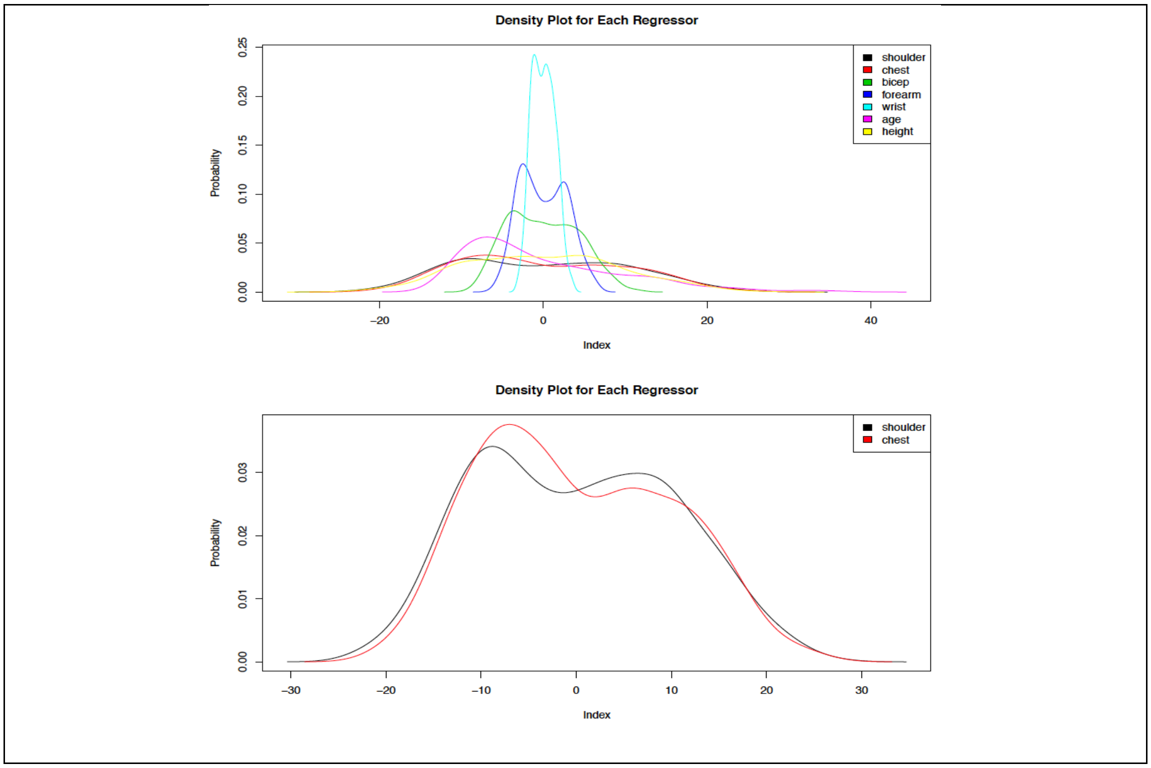
Xmat	A numeric design matrix with the equal row lengths
meanCent	Whether the columns of the matrix are mean cantered, default = FALSE
na.rm	Whether to remove missing observations

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 exists a missing value, then the whole observation must be removed.

Examples

```
# Body dimensions data
# X-matrix
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
pdf(paste("densPlots", "pdf", sep = "."), onefile=T, width=11, height=12)
par(mfrow=c(2,1))
densPlots(xmat=x, meanCent=TRUE)
densPlots(xmat=x[,1:2], meanCent=TRUE)
dev.off()

> densPlots(xmat=x, meanCent=TRUE)
> densPlots(xmat=x[,1:2], meanCent=TRUE)
```



Step 1: 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
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]  # Response Variable
implausStats(xmat = x, response = y)
```

```
> implausStats(xmat = x, response = y)
         Corr.w.Resp SLR.Coeff MLR.Coeff  SLR.Std.err MLR.Std.err
shoulder 0.8788342   1.130496  0.09085926 0.02731182  0.06488102
chest    0.8989595   1.196425  0.602339   0.02594212  0.0685496
bicep    0.8666722   2.723467  0.4674894  0.06976151  0.180125
forearm  0.8695531   4.099814  0.6479076  0.1036115   0.2997349
wrist    0.8164884   7.890809  -0.3194094 0.2482984   0.4011164
age      0.2072652   0.2878827 0.03578318 0.06046564  0.02445983
height   0.7173011   1.017617  0.3316698  0.0439868   0.03407658
```

Step 1: 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. It then performs regression models by adding one regressor at a time. Sequentially adding the highest in magnitude correlated regressor to lowest. It returns a plot of r-squared or adjusted r-squared values.

Usage

rsqdPlots(xmat, response, adjrsq = FALSE)

Arguments

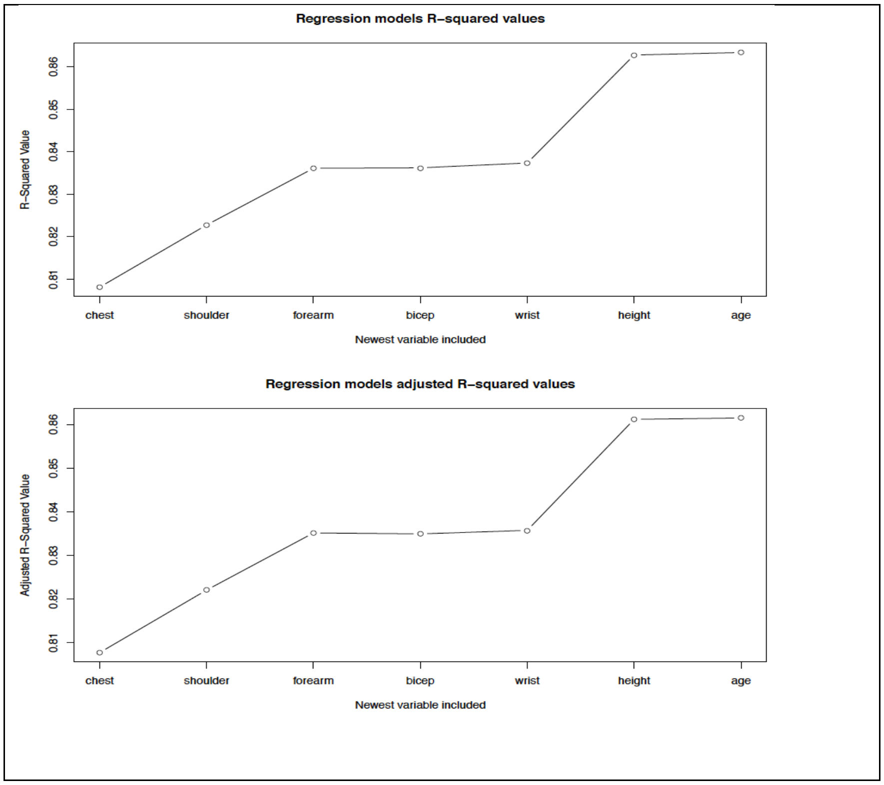
xmat	A numeric design matrix or dataframe
response	A numeric vector
adjrsq	Logical. If TRUE, the adjusted r-squared values will be calculated and plotted

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

```
> rsqdPlots(x,y,F)
> rsqdPlots(x,y,T)
```



Step 2: 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. It then 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. This R function calls the following 4 support function *regModelStats, randomNoiseMat, omcdiag,* and *imcdiag.* It involves random sampling, so we will set up a random seed first.

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

```
set.seed(6677)
# Body dimensions data
```

```
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]  # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.1
noiseEnd = 0.4
noiseSteps = 0.2
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
# > noiseLevs  # [1] 0.1 0.3
iteration =  1
NLDiagOut<-noiseLevelDiagOutList(xmat = x, y =y , special.Var= special.Var, noiseLevels =
noiseLevs, iter = iteration)
```

```
# We run a small iteration (iteration =  1 ) and 2 noise levels (0.1, and 0.3) to
# show the results and avoid generating a large output. The output is a list of lists.
# The length of the list is 3 for three 3 levels:
#  "Level 0", output based on the raw/original data, i.e., no noise added;
#  "Level 1", output based on a new data with at noise level 1= 0.1 noise added;
#  "Level 1", output based on a new data with at noise level 1= 0.3 noise added.
#  At each level, there are three different outputs, they are
#  linear regression (lm) summary, output of omcdiag (overall diagnosis),
#  and of imcdiag (individual diagnosis.
#  Below, only the level 2 output is listed to show the results and save some space.

$`Level 2`
$`Level 2`[[1]]
$`Level 2`[[1]][[1]]
            Estimate Std. Error    t value
shoulder  0.03428237 0.04783340  0.7167035
chest     0.63574288 0.06361726  9.9932445
bicep     0.48574489 0.17983104  2.7011182
forearm   0.67731383 0.29932884  2.2627750
wrist    -0.29011302 0.40097810 -0.7235134
age       0.03385730 0.02444551  1.3850105
height    0.33761750 0.03381569  9.9840479

$`Level 2`[[1]][[2]]
                          results detection
Determinant           2.657659e-04         1
Farrar Chi-Square     4.139774e+03         1
Red Indicator         6.849702e-01         1
sum of Lambda Invers  4.993366e+01         1
Theil Indicator      -1.598519e-01         0
Condition Number      1.203939e+02         1

$`Level 2`[[1]][[3]]
               VIF        TOL         Wi         Fi    Leamer         CVIF Klein
shoulder  5.533139 0.18072925  377.76155  454.22049 0.4251226 -0.23359085     0
chest     8.322549 0.12015550  610.21241  733.71940 0.3466345 -0.35135054     1
bicep    11.928677 0.08383159  910.72312 1095.05347 0.2895369 -0.50358938     1
forearm  14.681091 0.06811483 1140.09094 1370.84534 0.2609882 -0.61978721     1
wrist     6.270406 0.15947931  439.20050  528.09468 0.3993486 -0.26471584     0
age       1.128282 0.88630336   10.69015   12.85384 0.9414369 -0.04763234     0
height    2.069514 0.48320523   89.12617  107.16531 0.6951296 -0.08736805     0
```

```
                IND1      IND2
shoulder 0.0021687510 1.1428235
chest    0.0014418660 1.2273195
bicep    0.0010059791 1.2779888
forearm  0.0008173779 1.2999125
wrist    0.0019137517 1.1724657
age      0.0106356403 0.1585986
height   0.0057984627 0.7208914
```

Step3: overallDiagsPlots

Perturbation Analysis

The function *overallDiagPlots* takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and a number of iterations. It then 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 to show the distribution of a specific diagnostic measure.

Usage

overallDiagsPlots(xmat = x, yvar = y, noiseLevels = NL.2, spec.Vars = special.Vars, iter = iteration, choice = c(), mainLev=length(NL.2))

Arguments

xmat	A numeric design matrix or dataframe
response	A numeric vector
special.Vars	A regressor 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. The following is the explanation of 6 different "choice" d is for the determinant of |X'X| ch is for the Farr's Chi Sqaured statistics r is for the Red's indicator s is for the inverse sum of eigenvalues t is for the Theil's indicator co is for the condition number
mainLev	A noise level at which the overall diagnostic measure will be compared with the original one when no noise is added. *Note, changing it will NOT affect plots drawn by this function "overallDiagsPlots", but will affect the results of "overallDigasRank".*

Note

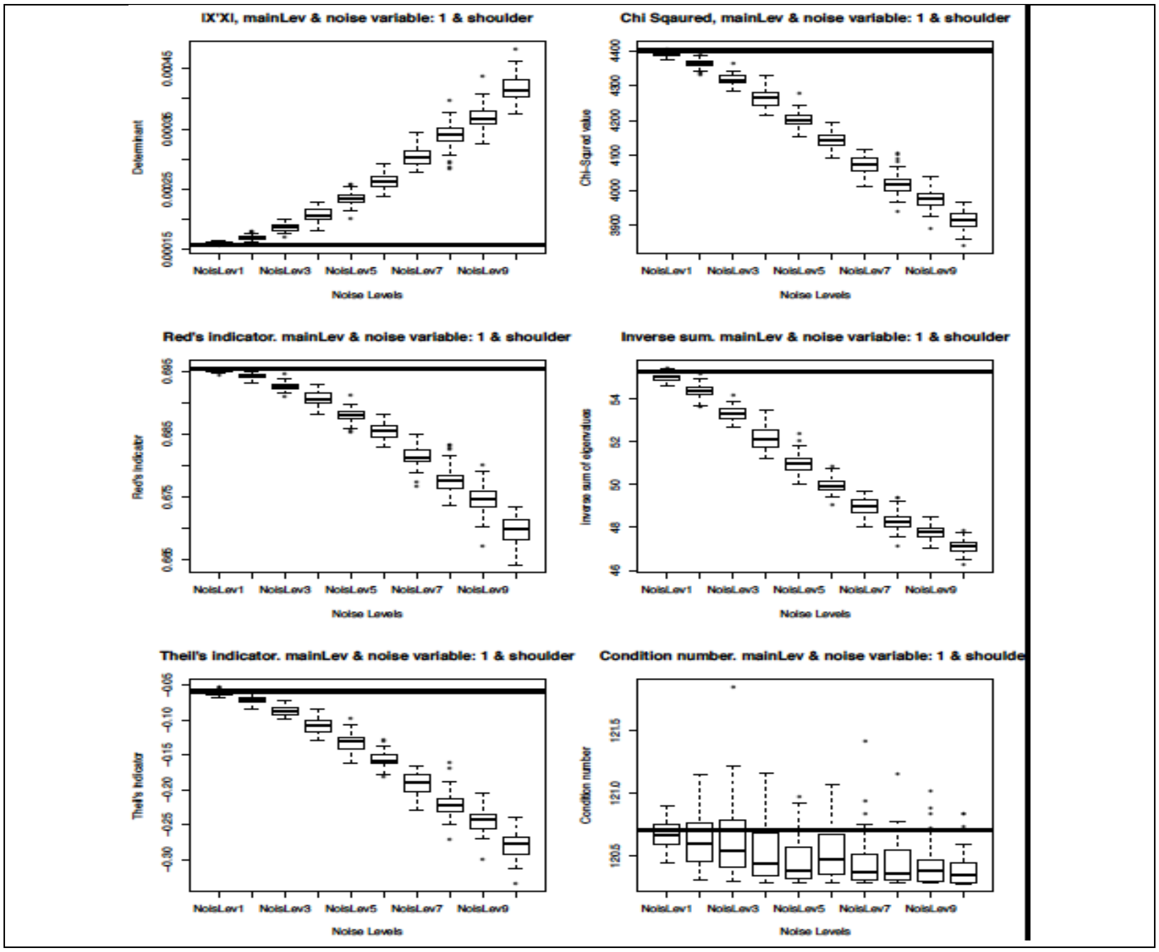
This function is made to display the distributions of an overall diagnostic as the noise level and special variable changes. The comparison to the original measurement should be observed. This function is dependent on the R "mctest" package. This function calls on the *overallDiagsDiffs* function to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

Examples

```
set.seed(6677)
```

```
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]   # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]     # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs.2 = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs.2   # [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  50
```

```
pdf("Mar26.2020.OverallDiag.6plots.pdf", width=8, height=11)
par(mfrow=c(3, 2))

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "d", mainLev=1)

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "ch", mainLev=1)

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "r", mainLev=1)
set.seed(6677)

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "s", mainLev=1)

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "t", mainLev=1)

set.seed(6677)
overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs.2, spec.Vars = special.Var, iter =
iteration, choice = "co", mainLev=1)

dev.off()
```



Step 3: boxplotAllVars

Perturbation Analysis

The function *boxplotAllVars* takes in a matrix of numeric regressors and a numeric response variable, a list of noise levels, a noise variable, and number of iterations. It then perturbs the noise regressor for n-iterations at every noise level, calculates 3 linear regression (lm) statistics and 7 multicollinearity diagnostic measures using *diagout* function that calls the *imcdiag* function of *mctest*. The 3 lm statistics are coefficients, standard errors, and t-statistics for the estimate of β, i.e., β_hat). The 7 multicollinearity diagnostic measures (based on the the *imcdiag* function of *mctest*) are VIF, TOL, Wi, Fi, Leamer statistic, Corrected VIF, and Klein. It plots the distributions of each these 10 diagnostic measures for each variable with respect to <u>*all regressors*</u>.

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

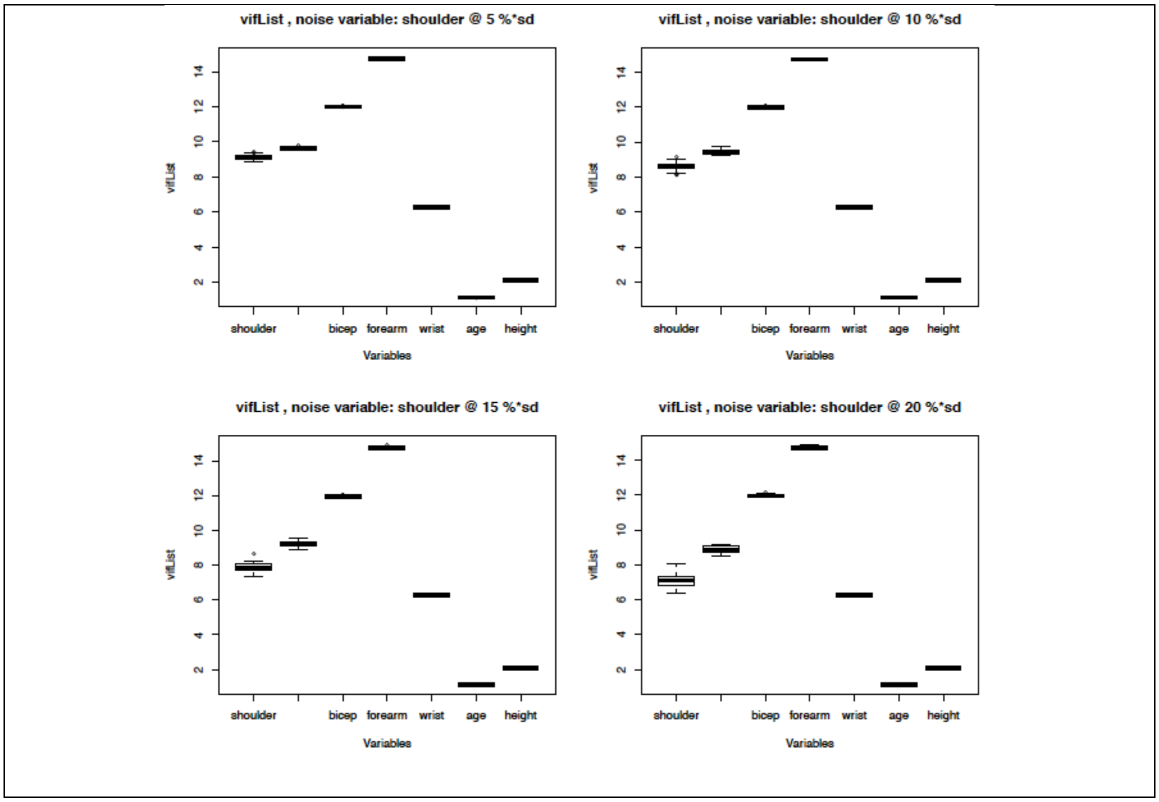
Note

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

Examples

```
# Body dimensions data
set.seed(6677)
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
# X-matrix
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
# Response Variable
y = body_dat[,23]
# Noise Variable
special.Var = "shoulder"
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs
# [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  50
boxplotAllVars(xmat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration = 
iteration)
```

```
> boxplotAllVars(xmat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration
= iteration)
[1] "Figures in the boxplotAllVars folder under
/Users/ssun/Desktop/Work/Mar24.2020.mcperturb/R.documentation/boxplotAllVars directory"
# This generates multiple pdf files with plots. Below is one example plot
```



Step 3: boxplotAllPerc

Perturbation Analysis

The function *boxplotAllPerc* takes in a matrix of numeric regressors and a numeric response variable, a list of noise levels, a noise variable, and number of iterations. It then perturbs the noise regressor for n-iterations at every noise level, calculates 3 linear regression (lm) statistics and 7 multicollinearity diagnostic measures using *diagout* function that calls the *imcdiag* function of *mctest*. The 3 lm statistics are coefficients, standard errors, and t-statistics for the estimate of β, i.e., β_hat). The 7 multicollinearity diagnostic measures (based on the the *imcdiag* function of *mctest*) are VIF, TOL, Wi, Fi, Leamer statistic, Corrected VIF, and Klein. It plots the distributions of each these 10 diagnostic measures for each variable with respect to <u>*all noise levels*</u>.

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

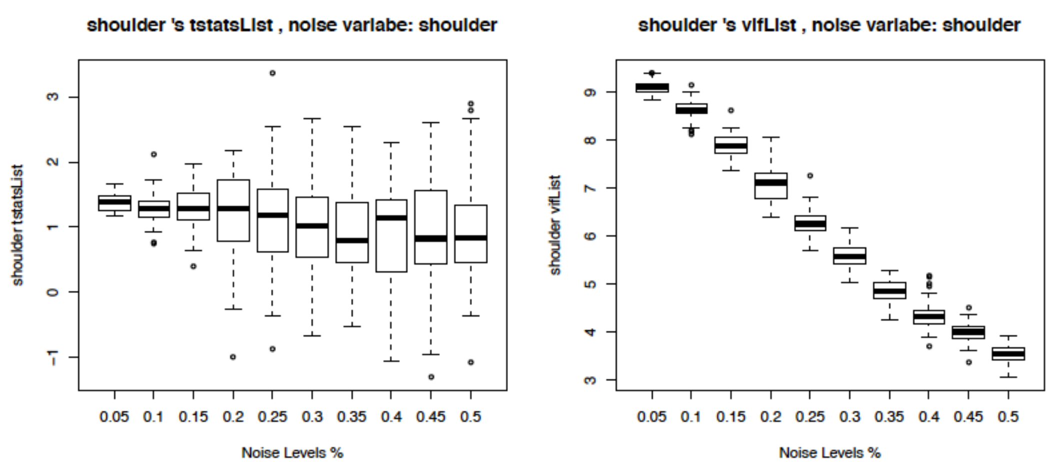
Note

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

Examples

```
# Next draw a plot of different levels
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
# X-matrix
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
# Response Variable
y = body_dat[,23]
# Noise Variable
special.Var = "shoulder"
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.25
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
# > noiseLevs  # [1] 0.05 0.10 0.15 0.20 0.25

iteration =  50
boxplotAllPerc(xmatrix = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration =
iteration)
```

```
> boxplotAllPerc(xmatrix = x, y = y, noiseLevs = noiseLevs,
special.Vars = special.Var, iteration = iteration)
[1] "Figures in the boxplotAllPerc folder under
/Users/ssun/Desktop/Work/Mar24.2020.mcperturb/R.documentation
directory  # This generates multiple folders with plots. Below are two example plots
```



Step 4: overallDiagsOut

Perturbation Analysis

The function *overallDiagsOut* takes an X-matrix of numeric regressors and a numeric response variable, performs perturbation analysis multiple (iter) times for ALL special variables or reressors (all columns in the X-matrix) one by one by calling the function "*noiseLevelDigaOutList*" to add noise to each variable/regressor. Then within the *noiseLevelDigaOutList*, the *omcdiag* function is called to calculate the 6 overall multicollinearity diagnostic measures, which are "Determinant of the correlation matrix |X'X| (Cooley and Lohnes, 1971), "Farrar test of chi-square for presence of multicollinearity (Farrar and Glauber, 1967)", "Red Indicator (Kovacs et al., 2015) ", "Sum of lambda inverse values (Chatterjee and Price (1977)", "Theil's indicator (Theil, 1971)", and "condition number (Belsley et al., 1980) ". This function returns an object with a list of the min.mean, max.mean, and difference (max.mean – min.mean) for each overall diagnostic of each noise variable.

Usage

overallDiagsOut(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 regressor as it is being perturbed.

Examples

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
# X-matrix
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
# Response Variable
y = body_dat[,23
special.Var = "shoulder"  ]# Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs  # [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  5
ODiag.Out<-overallDiagsOut(xmat = x, y = y, noiseLevs = noiseLevs, iteration = iteration)
```

```
> class(ODiag.Out)   # [1] "list"
> length(ODiag.Out)  # [1] 6
> ODiag.Out
```

```
$det
     Noise.Variable min.mean              max.mean              max-min
[1,] "shoulder" "0.000158236233098198" "0.000440067032119923" "0.000281830799021725"
[2,] "chest"    "0.000158236233098198" "0.000416376547928338" "0.000258140314830139"
[3,] "bicep"    "0.000158236233098198" "0.000499878199379834" "0.000341641966281635"
[4,] "forearm"  "0.000158236233098198" "0.000582542138662768" "0.000424305905564569"
[5,] "wrist"    "0.000158236233098198" "0.00032835795200353"  "0.000170121718905331"
[6,] "age"      "0.000158236233098198" "0.000163325568428765" "5.08933533056687e-06"
[7,] "height"   "0.000158236233098198" "0.000191848002835002" "3.3611769736804e-05"

$chisquare
     Noise.Variable min.mean          max.mean          max-min
[1,] "shoulder" "3886.98411493826" "4400.50644191905" "513.522326980791"
[2,] "chest"    "3914.63312503305" "4400.50644191905" "485.873316886007"
[3,] "bicep"    "3822.36424700147" "4400.50644191905" "578.142194917583"
[4,] "forearm"  "3745.57749751224" "4400.50644191905" "654.928944406817"
[5,] "wrist"    "4033.53388521165" "4400.50644191905" "366.972556707403"
[6,] "age"      "4384.59836023736" "4400.50644191905" "15.9080816816922"
[7,] "height"   "4303.89500434806" "4400.50644191905" "96.6114375709976"

$red
     Noise.Variable min.mean            max.mean            max-min
[1,] "shoulder" "0.667719616169839" "0.695359747777371" "0.0276401316075324"
[2,] "chest"    "0.671114529440227" "0.695359747777371" "0.0242452183371441"
[3,] "bicep"    "0.670651668988814" "0.695359747777371" "0.0247080787885569"
[4,] "forearm"  "0.66938999137101"  "0.695359747777371" "0.025969756406361"
[5,] "wrist"    "0.670652365454885" "0.695359747777371" "0.0247073823224863"
[6,] "age"      "0.693858514293327" "0.69546771197699"  "0.00160919768366252"
[7,] "height"   "0.680544857214764" "0.695359747777371" "0.0148148905626072"

$sumOfLambd
     Noise.Variable min.mean          max.mean          max-min
[1,] "shoulder" "46.8562671997692" "55.2837268489057" "8.42745964913645"
[2,] "chest"    "46.4782123144326" "55.2837268489057" "8.80551453447309"
[3,] "bicep"    "42.0726112183072" "55.2837268489057" "13.2111156305985"
[4,] "forearm"  "39.70788422925 7" "55.2837268489057" "15.5758426196486"
[5,] "wrist"    "49.5892357467693" "55.2837268489057" "5.6944911021364"
[6,] "age"      "54.9579918572007" "55.2852294904724" "0.3272376332717"
[7,] "height"   "54.5304096352675" "55.2837268489057" "0.753317213638141"

$theil
     Noise.Variable min.mean             max.mean              max-min
[1,] "shoulder" "-0.297368223695961"  "-0.0583289307508905" "0.239039292945071"
[2,] "chest"    "-0.211162626715811"  "-0.0583289307508905" "0.15283369596492"
[3,] "bicep"    "-0.275794676962166"  "-0.0583289307508905" "0.217465746211276"
[4,] "forearm"  "-0.326240431531088"  "-0.0583289307508905" "0.267911500780197"
[5,] "wrist"    "-0.27641731215505"   "-0.0583289307508905" "0.21808838140416"
[6,] "age"      "-0.0883785496508823" "-0.0583289307508905" "0.0300496188999918"
[7,] "height"   "-0.107545693939428"  "-0.0583289307508905" "0.0492167631885376"

$condi
     Noise.Variable min.mean          max.mean          max-min
[1,] "shoulder" "120.314634246313" "120.704525994086" "0.389891747772154"
[2,] "chest"    "120.105810058646" "120.704484331351" "0.598674272705182"
[3,] "bicep"    "108.298634016033" "120.704484331351" "12.4058503153184"
[4,] "forearm"  "108.57709794833"  "120.704484331351" "12.1273863830215"
[5,] "wrist"    "112.582862051874" "120.704484331351" "8.12162227947705"
[6,] "age"      "120.356670812942" "120.707232985201" "0.350562172258833"
[7,] "height"   "120.700910696312" "120.790007471808" "0.0890967754953493"
```

Step 4: rateOfChange

Perturbation Analysis

This function takes an X-matrix of numeric regressors and a numeric response variable, performs perturbation analysis multiple (iter) times for each noiselevel. Calculates the rate of change and least square best fit line for all the individual/overall multicollinearity diagnostic measures. This function returns an object with a list of the minimum mean, maximum mean, rate of change and least squares betst fit sllope value for each diagnostic with respect to the noise added to each variable. The output of *RateofChange* is a list of lists with the length = 10 for 10 diagnostic measures: coefficient, standard error, t-statistics, Vif, Tol, Wi, Fi, Leamer, cVif, and Klein. This function is a support function of *isBestFit* and *isRateOfChange*. It calls or depends on the function *diagout*.

Usage

rateOfChange(x_mat = x, y = 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 regressor to perturb
iteration	An integer

Note

This function provides an opportunity to look at the rate each multicollineatity diagnostic measure changes with respect to the noise added to a regressor. User can use this function to observe the rate at which each regressor's individual diagnostic measures change as more noise is added to a noise variable. The variable whose rate of change values is large is the variable that have a coupling relationship with the noise regressor. Note, rateOfChange is a support function for *isRateofChange* and *IsBestFit* in step 5.

Example

```
# Body dimensions data
set.seed(6677)
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # Response Variable
special.Var = "shoulder" # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs  # [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  50
RofChange.out<-rateOfChange(xmat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration =
iteration)
```

class(RofChange.out) # [1] "list"
length(RofChange.out) # [1] 10

```
> RofChange.out
$Coeff.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           0.091       0.061            -0.013         -0.012
chest              0.602       0.037             0.008         -0.007
bicep              0.467       0.020             0.004         -0.004
forearm            0.648       0.036             0.008         -0.007
wrist             -0.319       0.033             0.007         -0.006
age                0.036       0.002             0.000          0.000
height             0.332       0.007             0.001         -0.001

$StdError.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           0.065       0.029            -0.006         -0.006
chest              0.069       0.008            -0.002         -0.002
bicep              0.180       0.001             0.000          0.000
forearm            0.300       0.001             0.000          0.000
wrist              0.401       0.000             0.000          0.000
age                0.024       0.000             0.000          0.000
height             0.034       0.000             0.000          0.000

$`t-stats.Table`
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           1.400       0.606            -0.117         -0.117
chest              8.787       1.714             0.363         -0.330
bicep              2.595       0.121             0.026         -0.023
forearm            2.162       0.128             0.027         -0.025
wrist             -0.796       0.083             0.018         -0.016
age                1.463       0.087            -0.019         -0.017
height             9.733       0.332             0.068         -0.064

$VIF.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           9.293       5.746            -1.221         -1.108
chest              9.691       2.075            -0.450         -0.400
bicep             12.002       0.105            -0.022         -0.020
forearm           14.764       0.136            -0.029         -0.026
wrist              6.293       0.030            -0.006         -0.006
age                1.133       0.006            -0.001         -0.001
height             2.108       0.061            -0.012         -0.012

$TOL.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           0.108       0.174             0.035         -0.034
chest              0.103       0.028             0.006         -0.005
bicep              0.083       0.001             0.000          0.000
forearm            0.068       0.001             0.000          0.000
wrist              0.159       0.001             0.000          0.000
age                0.883       0.005             0.001         -0.001
height             0.474       0.014             0.003         -0.003

$Wi.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder         691.099     478.873          -101.724        -92.314
```

```
chest             724.258     172.928            -37.465         -33.336
bicep             916.865       8.767             -1.847          -1.690
forearm          1146.968      11.345             -2.410          -2.187
wrist             441.076       2.483             -0.463          -0.479
age                11.073       0.484             -0.105          -0.093
height             92.305       5.120             -1.038          -0.987

$Fi.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder          830.978     575.796           -122.313        -110.999
chest             870.848     207.928            -45.048         -40.083
bicep            1102.439      10.542             -2.220          -2.032
forearm          1379.114      13.642             -2.898          -2.630
wrist             530.350       2.985             -0.557          -0.575
age                13.314       0.581             -0.126          -0.112
height            110.987       6.157             -1.249          -1.187

$Leamer.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder            0.328       0.203              0.041          -0.039
chest               0.321       0.041              0.009          -0.008
bicep               0.289       0.001              0.000           0.000
forearm             0.260       0.001              0.000           0.000
wrist               0.399       0.001              0.000           0.000
age                 0.940       0.002              0.001           0.000
height              0.689       0.010              0.002          -0.002

$CVIF.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder           -0.383       0.230              0.049          -0.044
chest              -0.400       0.072              0.016          -0.014
bicep              -0.495       0.020             -0.004          -0.004
forearm            -0.609       0.025             -0.005          -0.005
wrist              -0.260       0.012             -0.002          -0.002
age                -0.047       0.002              0.000           0.000
height             -0.087       0.002              0.000           0.000

$Klein.Table
         Original-Values Diff-Median Least-squares-fit Rate-of-Change
shoulder                1           1             -0.245          -0.193
chest                   1           0              0.000           0.000
bicep                   1           0              0.000           0.000
forearm                 1           0              0.000           0.000
wrist                   0           0              0.000           0.000
age                     0           0              0.000           0.000
height                  0           0              0.000           0.000
```

Step 5: overallDiagsRank

Perturbation Analysis

The function *overallDiagRank* Takes an X-matrix of numeric regressors and a numeric response variable, performs perturbation analysis multiple (iter) times for each noiselevel. Calculates the overall multicollinearity diagnostic measures. Calculates the difference between the original overall diagnostic and the mean change in diagnostic. Then ranks the influence that each variable has on the overall diagnostic. It sums the overall ranks then ranks them. Rank 1 means the most influential one.

Usage

overallDiagsRank(xmat = x, resp = y, noiseL = NL.1, itera = iteration, mainLev=length(NL.1))

Arguments

xmat	A numeric design matrix or dataframe
resp	A numeric vector
noiseL	A list or a sequence of numeric noise levels
itera	An integer for the number of interations
mainLev	A noise level at which the overall diagnostic measure will be compared with the original one when no noise is added. *Note, changing it will NOT affect plots drawn by this function "overallDiagsPlots", but will affect the results of "overallDigasRank".*

Note

The *overallDiagsRank* function provides an opportunity to look at how the each variable affects the overall diagnostic individually and ranks the magnitude of the difference. It then sums up the ranking for a final rank of overall influence to a multicollinearity problem. It allows the user to observe how each variable influences the overall diagnostics as noise is added to each variable. Ranking the difference between the original and the relative values can help rank the influence each regressor has with respect to multicollinearity.

Examples

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs.2 = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs.2
# [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  5

set.seed(6677)
date();    ODiagRank.mainLev1.Out<-overallDiagsRank(xmat = x, resp = y,  noiseL = noiseLevs.2, itera
=iteration, mainLev= 1 ); date()

set.seed(6677)
date();  ODiagRank.mainLev10.Out<-overallDiagsRank(xmat = x, resp = y, noiseL = noiseLevs.2, itera
=iteration, mainLev= length(noiseLevs.2) );  date()
```

```
> class(ODiagRank.mainLev1.Out) # [1] "matrix"
> dim( ODiagRank.mainLev1.Out) # [1] 7 9
>
> options(width=150)
> ODiagRank.mainLev1.Out
     NoiseVariable detDiff chiSqrDiff redIndDiff sumOfLamDiff theilIndDiff conditionDiff Rank.Sum Overall.Rank
[1,] "shoulder"    "3"     "3"        "2"        "3"          "2"          "7"           "20"     "3"
[2,] "chest"       "2"     "2"        "1"        "2"          "4"          "4"           "15"     "2"
[3,] "bicep"       "4"     "4"        "7"        "4"          "6"          "2"           "27"     "4.5"
[4,] "forearm"     "1"     "1"        "5"        "1"          "1"          "1"           "10"     "1"
[5,] "wrist"       "5"     "5"        "4"        "5"          "5"          "3"           "27"     "4.5"
[6,] "age"         "7"     "7"        "6"        "7"          "7"          "6"           "40"     "7"
[7,] "height"      "6"     "6"        "3"        "6"          "3"          "5"           "29"     "6"
>
> ODiagRank.mainLev10.Out
     NoiseVariable detDiff chiSqrDiff redIndDiff sumOfLamDiff theilIndDiff conditionDiff Rank.Sum Overall.Rank
[1,] "shoulder"    "3"     "3"        "1"        "4"          "2"          "6"           "19"     "3"
[2,] "chest"       "4"     "4"        "5"        "3"          "5"          "4"           "25"     "4.5"
[3,] "bicep"       "2"     "2"        "3"        "2"          "4"          "1"           "14"     "2"
[4,] "forearm"     "1"     "1"        "2"        "1"          "1"          "2"           "8"      "1"
[5,] "wrist"       "5"     "5"        "4"        "5"          "3"          "3"           "25"     "4.5"
[6,] "age"         "7"     "7"        "7"        "7"          "7"          "5"           "40"     "7"
[7,] "height"      "6"     "6"        "6"        "6"          "6"          "7"           "37"     "6"
```

Step 5: isBestFit

Perturbation Analysis

The function (*isBestFit)* takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and number of iterations. It then 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. The output of *isBestFit* is a list of lists with the length = 6 for 6 diagnostic measures: vif, tol, wi, fi, leamer, and cVif.

Usage

isBestFit(x.mat = x, y = 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 regressor to perturb
iteration	An integer

Note

This function is made to output a list of summary tables with the best fit line for each regressor as it is being perturbed. Coupling regressors can be identified after analysis.

Examples

```
set.seed(6677)
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
```

```
y = body_dat[,23] # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs
# [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  50
isBestFit.out<-isBestFit(x.mat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration = iteration)
```

```
> isBestFit.out<-isBestFit(x.mat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var, iteration = iteration)
> length(isBestFit.out) # [1] 6
> isBestFit.out # List the first two diagnostic measure
$Vif
         shoulder.noise chest.noise bicep.noise forearm.noise wrist.noise age.noise height.noise
shoulder         -1.221      -0.460      -0.050        -0.074      -0.050    -0.004       -0.030
chest            -0.450      -1.338      -0.383         0.000       0.007    -0.026        0.000
bicep            -0.022      -0.197      -4.241        -2.706      -0.022     0.000       -0.029
forearm          -0.029       0.001      -2.120        -8.372      -3.521    -0.015       -0.008
wrist            -0.006       0.001      -0.005        -0.957      -5.019    -0.008       -0.037
age              -0.001      -0.009       0.000        -0.013      -0.024    -0.006       -0.001
height           -0.012       0.000      -0.029        -0.010      -0.152    -0.001       -0.085

$Tol
         shoulder.noise chest.noise bicep.noise forearm.noise wrist.noise age.noise height.noise
shoulder          0.035       0.007       0.001         0.001       0.001     0.000        0.000
chest             0.006       0.036       0.004         0.000       0.000     0.000        0.000
bicep             0.000       0.001       0.087         0.027       0.000     0.000        0.000
forearm           0.000       0.000       0.013         0.133       0.019     0.000        0.000
wrist             0.000       0.000       0.000         0.030       0.243     0.000        0.001
age               0.001       0.008       0.000         0.010       0.019     0.005        0.001
height            0.003       0.000       0.007         0.002       0.036     0.000        0.023
```

Step 5: isRateofChange

Perturbation Analysis

The function *(isRateofChange)* takes in a matrix of numeric regressors and a numeric response variable, a list of noise level, and number of iterations. It then 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. The output of *isRateofChange* is a list of lists with the length = 6 for 6 diagnostic measures: vif, tol, wi, fi, leamer, and cVif.

Usage

isRateofChange(x_mat = x, y = 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 regressor to perturb
iteration	An integer

Note

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

Examples

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]  # Response Variable
special.Var = "shoulder" # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs # [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  50
isRofChange.out<-isRateofChange(x_mat = x, y = y, noiseLevs = noiseLevs, special.Vars =
special.Var, iteration = iteration)
```

```
> isRofChange.out<-isRateofChange(x_mat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Var,
iteration = iteration)
> class(isRofChange.out) # [1] "list"
> length(isRofChange.out) # [1] 6
> isRofChange.out
$Vif
         shoulder.noise chest.noise bicep.noise forearm.noise wrist.noise age.noise height.noise
shoulder         -1.108      -0.411      -0.049        -0.065      -0.052    -0.006       -0.028
chest            -0.400      -1.219      -0.336        -0.004      -0.010    -0.025       -0.001
bicep            -0.020      -0.185      -3.873        -2.480      -0.028     0.000       -0.031
forearm          -0.026      -0.002      -1.954        -7.598      -3.251    -0.015       -0.009
wrist            -0.006      -0.001      -0.005        -0.875      -4.637    -0.008       -0.038
age              -0.001      -0.009       0.000        -0.013      -0.022    -0.007       -0.001
height           -0.012       0.000      -0.027        -0.010      -0.140    -0.001       -0.081
$Tol
         shoulder.noise chest.noise bicep.noise forearm.noise wrist.noise age.noise height.noise
shoulder         -0.034      -0.006       0.000        -0.001       0.000     0.000        0.000
chest            -0.005      -0.035      -0.004         0.000       0.000     0.000        0.000
bicep             0.000      -0.001      -0.085        -0.024       0.000     0.000        0.000
forearm           0.000       0.000      -0.012        -0.129      -0.017     0.000        0.000
wrist             0.000       0.000       0.000        -0.028      -0.239     0.000       -0.001
age              -0.001      -0.007       0.000        -0.010      -0.017    -0.005        0.000
height           -0.003       0.000      -0.006        -0.002      -0.033     0.000       -0.022
```

Support Functions

For the 6 support functions, that are called by the above main functions, I list example scripts and output files below.

Support Function 1: regModelStats

This function takes an X-matrix of numeric regressors and a numeric response variable, Clculates the # regression model and returns a Matix full of summary statistics.

```
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # Response Variable
regModelStats(xmat = x, response = y)
```

```
> regModelStats(xmat = x, response = y)
            Estimate Std. Error    t value
shoulder  0.09085926 0.06488102  1.4003982
chest     0.60233904 0.06854960  8.7869086
bicep     0.46748943 0.18012496  2.5953618
forearm   0.64790765 0.29973494  2.1616020
wrist    -0.31940943 0.40111638 -0.7963011
age       0.03578318 0.02445983  1.4629364
height    0.33166978 0.03407658  9.7330724
```

Support Function 2: randomNoiseMat

Additional notes: "noiseLevelDiagOutList" calls the support function "randomNoiseMat", we also briefly show an example of this support function and its output.

*"randomNoiseMat" t*akes an X-matrix of numeric regressors and adds random noise to a variable selected as a special variable. Adds random noise to the selected variable and returns a new x-matrix with noise added to this regressor. The following examples show that random noise is added to variables "shoulder" and "wrist".

```
set.seed(6677)
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]  # Response Variable
special.Var = "shoulder"  # Noise Variable
x.w.noise.to.shoulder<-randomNoiseMat(x_mat=x, special_Var = special.Var , noiseLevel=0.1)

special.Var = "wrist"
```

```
x.w.noise.to.wrist<-randomNoiseMat(x_mat=x, special_Var = special.Var , noiseLevel=0.1)

head(x)
head(x.w.noise.to.shoulder)
head(x.w.noise.to.wrist)

> head(x)
  shoulder chest bicep forearm wrist age height
1    106.2  89.5  32.5    26.0  16.5  21  174.0
2    110.5  97.0  34.4    28.0  17.0  23  175.3
3    115.1  97.5  33.4    28.8  16.9  28  193.5
> head(x.w.noise.to.shoulder)
  shoulder chest bicep forearm wrist age height
1 107.1536  89.5  32.5    26.0  16.5  21  174.0
2 111.8044  97.0  34.4    28.0  17.0  23  175.3
3 114.8669  97.5  33.4    28.8  16.9  28  193.5
> head(x.w.noise.to.wrist)
  shoulder chest bicep forearm    wrist age height
1    106.2  89.5  32.5    26.0 16.48948  21  174.0
2    110.5  97.0  34.4    28.0 16.96326  23  175.3
3    115.1  97.5  33.4    28.8 16.97485  28  193.5
```

Support Function 3: noiseLevelsList

This function adds random noise any variable in the xmatrix at multiple noise steps. It puts these different noise levels x matricies in a list, then repeats multiple iterations. It finally returns a list noise steps with a list of noisexmatrices

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # # Response Variable
special.Var = "shoulder" # # Noise Variable
# Making the noiselevels
noiseStart = 0.1
noiseEnd = 0.4
noiseSteps = 0.2
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps) # > noiseLevs  # [1] 0.1 0.3

NLL.output.1iter<-noiseLevelsList(x.mat = x, noiseLevels = noiseLevs , special.Vars = special.Var, itera = 1)
# For one iteration, two noise level (0.1, 0.3), it generate a list of length 2.

> class(NLL.output.1iter) # [1] "list"
> length(NLL.output.1iter) # [1] 2
> NLL.output.1iter
```

```
$`10 %*sd`
$`10 %*sd`[[1]]

# Next I make the 1 matrix has a smaller number of observations (e.g., just 3 rows)
and run 5 iterations to see what we get:

set.seed(6677)
NLL.output.smallX.5iter<-noiseLevelsList(x.mat = x[1:3,], noiseLevels = noiseLevs ,
special.Vars = special.Var, itera = 5)

class(NLL.output.smallX.5iter)
length(NLL.output.smallX.5iter)

> NLL.output.smallX.5iter
$`10 %*sd`
$`10 %*sd`[[1]]
  shoulder chest bicep forearm wrist age height
1 106.6091  89.5  32.5    26.0  16.5  21  174.0
2 111.0596  97.0  34.4    28.0  17.0  23  175.3
3 115.0000  97.5  33.4    28.8  16.9  28  193.5

$`10 %*sd`[[2]]
  shoulder chest bicep forearm wrist age height
1 106.7904  89.5  32.5    26.0  16.5  21  174.0
2 110.6307  97.0  34.4    28.0  17.0  23  175.3
3 114.5010  97.5  33.4    28.8  16.9  28  193.5

$`10 %*sd`[[3]]
  shoulder chest bicep forearm wrist age height
1 106.1844  89.5  32.5    26.0  16.5  21  174.0
2 110.5832  97.0  34.4    28.0  17.0  23  175.3
3 115.1485  97.5  33.4    28.8  16.9  28  193.5

$`10 %*sd`[[4]]
  shoulder chest bicep forearm wrist age height
1 106.3591  89.5  32.5    26.0  16.5  21  174.0
2 110.1576  97.0  34.4    28.0  17.0  23  175.3
3 114.8659  97.5  33.4    28.8  16.9  28  193.5

$`10 %*sd`[[5]]
  shoulder chest bicep forearm wrist age height
1 106.3180  89.5  32.5    26.0  16.5  21  174.0
2 109.5355  97.0  34.4    28.0  17.0  23  175.3
3 115.5222  97.5  33.4    28.8  16.9  28  193.5

$`30 %*sd`
$`30 %*sd`[[1]]
  shoulder chest bicep forearm wrist age height
1 106.7179  89.5  32.5    26.0  16.5  21  174.0
2 109.6624  97.0  34.4    28.0  17.0  23  175.3
3 116.1772  97.5  33.4    28.8  16.9  28  193.5

$`30 %*sd`[[2]]
```

```
  shoulder chest bicep forearm wrist age height
1 105.1338  89.5  32.5    26.0  16.5  21  174.0
2 109.4705  97.0  34.4    28.0  17.0  23  175.3
3 114.7973  97.5  33.4    28.8  16.9  28  193.5

$`30 %*sd`[[3]]
  shoulder chest bicep forearm wrist age height
1 108.5420  89.5  32.5    26.0  16.5  21  174.0
2 111.2659  97.0  34.4    28.0  17.0  23  175.3
3 112.7919  97.5  33.4    28.8  16.9  28  193.5

$`30 %*sd`[[4]]
  shoulder chest bicep forearm wrist age height
1 106.2673  89.5  32.5    26.0  16.5  21  174.0
2 110.3795  97.0  34.4    28.0  17.0  23  175.3
3 117.4688  97.5  33.4    28.8  16.9  28  193.5

$`30 %*sd`[[5]]
  shoulder chest bicep forearm wrist age height
1 104.8503  89.5  32.5    26.0  16.5  21  174.0
2 109.8555  97.0  34.4    28.0  17.0  23  175.3
3 114.4687  97.5  33.4    28.8  16.9  28  193.5
```

Support Function 4: diagout

---

1. This function organizes all the diagnostics into a list of list
2. It calls the "noiseLevelsList" (which calls "randomNoiseMat") and "imcdiag" (of the mctest package to calculate indiviudal measures)
3. It loops through each noise level "j in 1:length(noiseLevels)", and for each noise level, it loops through each iteration "i in 1:iter".
4. The output is a list of lists. This list with a lengh of 10 for 10 different diagnostic measures with the following names: "coeffList", "stderrorList", "tstatsList", "vifList", "tolList", "wiList", "fiList", "leamerList", "cvifList", "kleinList". The first 3 are basic lm output (cofficient, standard_error, t_statis of the β_hat).The next 7 are the 7 multicolinearity diagnoistic measures: vif, tol, wi, fi, leamer, cv, klein.

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.1
noiseEnd = 0.4
noiseSteps = 0.2
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
# > noiseLevs  # [1] 0.1 0.3
```

iteration = 1

diagout.output<-diagout(xmat = x, response = y, noiseLevels = noiseLevs, special.Var= special.Var, iter = 1)
# When there is only one 1 iteration, it output a list with length 10.
# For output of more iterations, see below.

```
> class(diagout.output)  # [1] "list"
> length(diagout.output) # [1] 10
> diagout.output
$coeffList
$coeffList[[1]]
          V1       V2        V3        V4        V5         V6       V7
1 0.09366936 0.600273 0.4678251 0.6390162 -0.306476 0.03590239 0.331409
$coeffList[[2]]
          V1        V2        V3        V4        V5        V6        V7
1 0.03428237 0.6357429 0.4857449 0.6773138 -0.290113 0.0338573 0.3376175

$stderrorList
$stderrorList[[1]]
          V1         V2        V3        V4        V5         V6         V7
1 0.06249853 0.06789511 0.1798944 0.3000946 0.4003224 0.02444809 0.03401256
$stderrorList[[2]]
         V1         V2       V3        V4        V5         V6         V7
1 0.0478334 0.06361726 0.179831 0.2993288 0.4009781 0.02444551 0.03381569

$tstatsList
$tstatsList[[1]]
        V1       V2       V3       V4        V5       V6       V7
1 1.498745 8.841181 2.600555 2.129382 -0.765573 1.468515 9.743722
$tstatsList[[2]]
         V1       V2       V3       V4         V5       V6       V7
1 0.7167035 9.993244 2.701118 2.262775 -0.7235134 1.385011 9.984048

$vifList
$vifList[[1]]
        V1       V2       V3       V4       V5       V6       V7
1 8.715567 9.512335 11.97848 14.80749 6.271593 1.132434 2.100943
$vifList[[2]]
        V1       V2       V3       V4       V5       V6       V7
1 5.533139 8.322549 11.92868 14.68109 6.270406 1.128282 2.069514

$tolList
$tolList[[1]]
         V1        V2         V3        V4        V5        V6        V7
1 0.1147372 0.1051267 0.08348301 0.0675334 0.1594491 0.8830535 0.4759767
$tolList[[2]]
         V1        V2         V3         V4        V5        V6        V7
1 0.1807293 0.1201555 0.08383159 0.06811483 0.1594793 0.8863034 0.4832052

$wiList
$wiList[[1]]
       V1       V2       V3       V4       V5       V6       V7
```

```
1 642.964 709.3613 914.8737 1150.624 439.2994 11.03619 91.74526
$wiList[[2]]
        V1       V2       V3       V4       V5       V6       V7
1 377.7615 610.2124 910.7231 1140.091 439.2005 10.69015 89.12617

$fiList
$fiList[[1]]
        V1      V2       V3      V4       V5       V6       V7
1 773.0999 852.936 1100.044 1383.51 528.2136 13.26991 110.3145
$fiList[[2]]
        V1       V2       V3       V4       V5       V6       V7
1 454.2205 733.7194 1095.053 1370.845 528.0947 12.85384 107.1653

$leamerList
$leamerList[[1]]
         V1        V2        V3        V4        V5        V6        V7
1 0.3387288 0.3242324 0.2889343 0.2598719 0.3993108 0.9397092 0.6899107
$leamerList[[2]]
         V1        V2        V3        V4        V5        V6        V7
1 0.4251226 0.3466345 0.2895369 0.2609882 0.3993486 0.9414369 0.6951296

$cvifList
$cvifList[[1]]
          V1         V2         V3         V4         V5         V6          V7
1 -0.3597972 -0.3926895 -0.4944974 -0.6112847 -0.2589047 -0.0467493 -0.08673141
$cvifList[[2]]
          V1         V2         V3         V4         V5          V6
1 -0.2335908 -0.3513505 -0.5035894 -0.6197872 -0.2647158 -0.04763234
           V7
1 -0.08736805

$kleinList
$kleinList[[1]]
  V1 V2 V3 V4 V5 V6 V7
1  1  1  1  1  0  0  0
$kleinList[[2]]
  V1 V2 V3 V4 V5 V6 V7
1  0  1  1  1  0  0  0
```

##

With the same random seed, 2 noiseLevels, we run 3 iterations to see what we got (other variables are not changed)

```
set.seed(6677)
# Body dimensions data
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)]  ## X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23]  ## Response Variable
special.Var = "shoulder" ## Noise Variable
# Making the noiselevels
noiseStart = 0.1
noiseEnd = 0.4
noiseSteps = 0.2
```

```
noiseLevs = seq(noiseStart, noiseEnd, by = noiseSteps)
# > noiseLevs  # [1] 0.1 0.3
diagout.output.3iter<-diagout(xmat = x, response = y,  noiseLevels = noiseLevs, special.Var= special.Var, iter = 3)
diagout.output.3iter

# Mar 24, 2020 Summary notes of "dignout" output:   A list of list 10, for 10 different diagnostics
# > diagout.output.3iter[[4]][[2]] # means I want to get the 4th diagnoistic measure (VIF) and the 2nd noise level (0.3)
        V1       V2       V3       V4       V5       V6       V7
1 5.408821 7.894485 11.94366 14.78930 6.260021 1.127068 2.096182
2 5.262340 8.326743 11.89284 14.62762 6.277886 1.125659 2.109309
3 5.576560 8.390851 12.00558 14.59277 6.302922 1.126524 2.050332

# "diagout.output.3iter[[4]][[2]][3]" means I want to get the 4th diagnoistic measure (VIF) at the 2nd noise level
#  (0.3) and the 3rd variable/regressor. So the "diagout" output summaryResults[[j]][[i]][k] means
#  the "j_th" diagnoistic measure, the i_th noise level, and the k_th variable/regressor.

> diagout.output.3iter[[4]][[2]][3]
        V3
1 11.94366
2 11.89284
3 12.00558

> set.seed(6677)
> diagout.output.3iter<-diagout(xmat = x, response = y,  noiseLevels = noiseLevs, special.Var= special.Var, iter = 3)
> diagout.output.3iter
$coeffList
$coeffList[[1]]
          V1       V2        V3        V4         V5         V6        V7
1 0.09366936 0.600273 0.4678251 0.6390162 -0.3064760 0.03590239 0.3314090
2 0.07352055 0.612487 0.4730550 0.6567054 -0.3094755 0.03514970 0.3334754
3 0.09597529 0.598935 0.4648141 0.6500660 -0.3256696 0.03611602 0.3312532

$coeffList[[2]]
          V1        V2        V3        V4         V5         V6        V7
1 0.02269351 0.6445660 0.4889416 0.6786466 -0.2811974 0.03339881 0.3382568
2 0.09004415 0.6007502 0.4735739 0.6597428 -0.3223751 0.03428728 0.3278697
3 0.08324318 0.6063812 0.4606918 0.6758822 -0.3355406 0.03456809 0.3345468
 … …
```

Support Function 5: overallDiagsDiffs

This function takes an X-matrix of numeric regressors and a numeric response variable, performs pertubation analysis multiple (iter) times for each noiselevel. Calculates the overall multicollinearity diagnostic measures. Calculates the difference between the original overall diagnostic and the mean change in diagnostic.

```
set.seed(6677)
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)  # Body dimensions data
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # # Response Variable
special.Var = "shoulder"  # Noise Variable
# Making the noiselevels
noiseStart = 0.05
noiseEnd = 0.5
noiseSteps = 0.05
noiseLevs.1 = seq(noiseStart, noiseEnd, by = noiseSteps)
noiseLevs.1 # [1] 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50
iteration =  5
date();     ODiagDiff.Out<-overallDiagsDiffs(xmat = x, y = y, special.Vars= special.Var, noiseLevs =
noiseLevs.1, iteration = iteration, mainLev=length(noiseLevs.1))  ;  date()
```

```
[1] "Thu Mar 26 16:13:33 2020"
> ODiagDiff.Out<-overallDiagsDiffs(xmat = x, y = y, special.Vars= special.Var, noiseLevs = noiseLevs.1, iteration = iteration,
mainLev=length(noiseLevs.1))
-------------------------------------------------------------------
In overallDiagsDiffs, the main noise level mainLel that is used to
compare with the original data (with no noise added) is 10
-------------------------------------------------------------------
> date()
[1] "Thu Mar 26 16:13:36 2020"
> length(ODiagDiff.Out) # [1] 7

> ODiagDiff.Out
$difference
       det.diff chis2.diff    red.diff sumLam.diff  thei.diff cond.diff
[1,] 0.0002818308  -513.5223 -0.02764013    -8.42746 -0.2390393 -0.330889
```

```
$det
           NoisLev1     NoisLev2     NoisLev3     NoisLev4     NoisLev5     NoisLev6     NoisLev7     NoisLev8
iteration1 0.0001604316 0.0001702599 0.0001926501 0.0002207549 0.0002342143 0.0002665794 0.0003000236 0.0003373025
iteration2 0.0001628088 0.0001707581 0.0001872432 0.0001898446 0.0002286977 0.0002672586 0.0003004891 0.0003684817
iteration3 0.0001613447 0.0001678857 0.0001870119 0.0001936547 0.0002094295 0.0002274001 0.0002781329 0.0003543114
iteration4 0.0001620856 0.0001673719 0.0001818624 0.0002157498 0.0002362886 0.0002739606 0.0002911337 0.0003393598
iteration5 0.0001645796 0.0001804843 0.0001896504 0.0002070613 0.0002374891 0.0002697182 0.0002987763 0.0003285691
           NoisLev9     NoisLev10
iteration1 0.0003827971 0.0004554758
iteration2 0.0003709600 0.0004235937
iteration3 0.0003531603 0.0004822867
iteration4 0.0003634708 0.0004194508
iteration5 0.0004064478 0.0004195281

$chisquare
```

```
           NoisLev1 NoisLev2 NoisLev3 NoisLev4 NoisLev5 NoisLev6 NoisLev7 NoisLev8 NoisLev9 NoisLev10
iteration1 4393.578 4363.680 4301.556 4233.081 4203.322 4138.237 4078.808 4019.916 3956.295  3868.884
iteration2 4386.182 4362.211 4315.870 4308.932 4215.307 4136.958 4078.028 3975.460 3972.090  3905.374
iteration3 4390.724 4370.742 4316.491 4298.940 4259.563 4218.168 4116.903 3995.179 3996.815  3840.124
iteration4 4388.420 4372.283 4330.531 4244.613 4198.888 4124.504 4093.932 4016.859 3982.345  3910.316
iteration5 4380.743 4334.356 4309.447 4265.281 4196.340 4132.351 4080.903 4033.107 3926.150  3910.223

$red
            NoisLev1  NoisLev2  NoisLev3  NoisLev4  NoisLev5  NoisLev6  NoisLev7  NoisLev8  NoisLev9 NoisLev10
iteration1 0.6950509 0.6941192 0.6920457 0.6885629 0.6876437 0.6848276 0.6816527 0.6777275 0.6735259 0.6671219
iteration2 0.6949090 0.6939478 0.6921928 0.6924470 0.6884897 0.6843026 0.6830333 0.6744605 0.6754121 0.6698724
iteration3 0.6950218 0.6946059 0.6925140 0.6917533 0.6897442 0.6881560 0.6844659 0.6753766 0.6774929 0.6616666
iteration4 0.6951686 0.6942589 0.6928927 0.6901573 0.6876806 0.6847211 0.6814491 0.6781834 0.6753858 0.6699308
iteration5 0.6948020 0.6931468 0.6921938 0.6902749 0.6875119 0.6845882 0.6821494 0.6793044 0.6712347 0.6700064

$sumOfLambd
           NoisLev1 NoisLev2 NoisLev3 NoisLev4 NoisLev5 NoisLev6 NoisLev7 NoisLev8 NoisLev9 NoisLev10
iteration1 55.13197 54.36718 52.84894 51.62506 51.08519 49.85872 49.13048 48.29575 47.54884  46.55209
iteration2 54.91707 54.42411 53.36867 52.98767 51.25396 49.99709 48.73145 47.84362 47.66028  46.90101
iteration3 55.04389 54.38506 53.27776 52.91363 52.24789 51.25330 49.40314 48.00240 47.83858  46.76433
iteration4 54.89032 54.67638 53.64548 51.71269 50.92936 49.50054 49.47415 48.21315 47.97267  47.07306
iteration5 54.79773 53.65847 53.12920 52.33466 50.93384 49.73807 48.91133 48.39288 47.16995  46.99085

$theil
              NoisLev1    NoisLev2    NoisLev3    NoisLev4   NoisLev5   NoisLev6   NoisLev7   NoisLev8   NoisLev9
iteration1 -0.06109542 -0.07568180 -0.08975173 -0.11971487 -0.1407146 -0.1605263 -0.1871956 -0.2221323 -0.2552014
iteration2 -0.06309433 -0.06747341 -0.08685978 -0.07817530 -0.1173751 -0.1638061 -0.1924457 -0.2430981 -0.2579822
iteration3 -0.06171446 -0.06740706 -0.08743128 -0.08842263 -0.1149200 -0.1063832 -0.1717632 -0.2173310 -0.2273780
iteration4 -0.06196378 -0.06422922 -0.07766651 -0.11971843 -0.1402756 -0.1722017 -0.1787188 -0.2259139 -0.2412366
iteration5 -0.06428866 -0.08474795 -0.08927748 -0.09697780 -0.1427392 -0.1684380 -0.2028632 -0.2109321 -0.2715569
            NoisLev10
iteration1 -0.3133551
iteration2 -0.2891283
iteration3 -0.3135730
iteration4 -0.2868272
iteration5 -0.2839575

$condi
           NoisLev1 NoisLev2 NoisLev3 NoisLev4 NoisLev5 NoisLev6 NoisLev7 NoisLev8 NoisLev9 NoisLev10
iteration1 120.8960 120.3602 120.5219 120.9840 120.4168 120.3346 120.3662 120.2895 120.3893  120.2999
iteration2 120.6666 120.6841 120.7366 120.4072 120.3474 120.2899 120.2881 120.3213 120.6545  120.2798
iteration3 120.5818 120.4006 120.5117 120.3909 120.5123 120.2928 120.4885 120.3096 120.3659  120.3281
iteration4 120.7849 120.4580 120.3266 120.2930 120.6873 120.3392 120.5841 120.4789 120.4075  120.4659
iteration5 120.5933 120.4524 120.4107 120.9695 120.3964 120.3166 120.3148 120.5777 120.3245  120.4942
```

Support Function 6: varDecomp

This function will give the user the chance to calculate variance decomposition proportions based on the functions provided by the mctest and/or perturb package

```
set.seed(6677)
# Body dimensions data
body_dat = read.table("http://jse.amstat.org/datasets/body.dat.txt", header=F)
dim( body_dat) # 507 * 25
x = body_dat[,c(10, 11, 16, 17, 21, 22, 24)] # # X-matrix
colnames(x) = c("shoulder", "chest", "bicep", "forearm", "wrist", "age", "height")
y = body_dat[,23] # # Response Variable
varDecomp(x)

> varDecomp(x)
$mctest_eigprop

Call:
eigprop(x = df, na.rm = na.rm, Inter = intercept)

  Eigenvalues       CI shoulder  chest  bicep forearm  wrist    age height
1      6.9114   1.0000   0.0000 0.0000 0.0000  0.0000 0.0000 0.0016 0.0001
2      0.0776   9.4390   0.0003 0.0002 0.0006  0.0003 0.0003 0.9124 0.0007
3      0.0070  31.4596   0.0000 0.0049 0.0771  0.0037 0.0075 0.0024 0.2143
4      0.0018  61.4061   0.1127 0.2315 0.0295  0.0684 0.1518 0.0151 0.0011
5      0.0010  82.0938   0.0433 0.0510 0.4233  0.0027 0.3985 0.0156 0.7480
6      0.0007 100.9978   0.8100 0.6633 0.0021  0.0244 0.0506 0.0413 0.0328
7      0.0005 112.7137   0.0338 0.0490 0.4674  0.9005 0.3913 0.0117 0.0030
===============================
Row 6==> shoulder, proportion 0.809960 >= 0.50
Row 6==> chest, proportion 0.663315 >= 0.50
Row 7==> forearm, proportion 0.900491 >= 0.50
Row 2==> age, proportion 0.912432 >= 0.50
Row 5==> height, proportion 0.748016 >= 0.50

$perturb_colldiag
Condition
Index   Variance Decomposition Proportions
          shoulder chest bicep forearm wrist age   height
1    1.000 0.004    0.004 0.003 0.002   0.005 0.002 0.010
2    2.286 0.000    0.000 0.000 0.000   0.000 0.864 0.016
3    3.190 0.003    0.008 0.015 0.004   0.001 0.045 0.756
4    5.057 0.092    0.106 0.000 0.025   0.418 0.002 0.055
5    6.866 0.197    0.042 0.281 0.080   0.336 0.025 0.144
6    8.707 0.691    0.773 0.000 0.036   0.052 0.056 0.008
7   10.826 0.013    0.067 0.700 0.852   0.188 0.006 0.011
```