Documentation of *mcperturb*

*mcperturb* webpage: <https://github.com/RyanZam1030/mcperturb/>

Ryan Zamora | [rz1030@yahoo.com](mailto:rz1030@yahoo.com)

Shuying Sun | [ssun5211@yahoo.com](mailto:ssun5211@yahoo.com)

Last Updated on January 6, 2020

**Table of Contents**

|  |  |
| --- | --- |
|  | **Page #** |
| **Overall/General description of the *mcperturb* package** | 3 |
| **Body dimensions dataset overview** |  |
| **Function Name** |  |
| *densPlots* | 3 |
| *implausStats* | 3 |
| *rsqdPlots* | 4 |
| *boxplotAllPerc* | 5 |
| *boxplotAllVars* | 7 |
| *overallDiagsPlots* | 8 |
| *noiseLevelDiagOutList* | 9 |
| *IsBestFit* | 11 |
| *IsRateofChange* | 12 |
| *overallDiagsRank* | 13 |

**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 R2 | *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, BoxplotsAllVars, BoxplotsAllPercent* | 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, mcperSumTables* | 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** |
| 1 | *densPlots* | Calculates 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 | *boxplotsAllVars* | 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 | *noiseLevelDiagOutList*  *rateOfChange* |
| 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 | *noiseLevelDiagOutList*  *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* |
|  | Supporting Functions | | |
| 12 | *randomNoiseMat* | Perturbs the data by add noise to regressors in a matrix | None |
| 13 | *regModelStats* | Calculates the regression model statistics | None |
| 14 | *overallDiagsDiffs* | Calculates the difference between the overall diagnostic measures as noise is added to regressors | noiseLevelDiagOutList |
| 15 | *diagout* | Organizes all the diagnostics into a list of list | noiseLevelsList |
| 16 | *noiseLevelsList* | Makes a list of the different noise levels to perturb and |  |

**Body dimensions 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 #https://github.com/RyanZam1030/mcperturb/blob/master/data\_and\_examples/body.dat.txt*  *body\_dat = read.table("C:/Users/body.dat.txt", sep = " ", strip.white = TRUE)*  *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")* |

|  |
| --- |
| ***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 = FALSE, 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*  *data(body\_dat)*  *# 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=11)*  *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) |

|  |
| --- |
| ***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\_dat)*  *# 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]*  *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 |

|  |
| --- |
| ***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**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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]*  *pdf(paste(“rsqdPlots", "pdf", sep = "."), onefile=T, width=11, height=11)*  *rsqdPlots(x,y)*  *rsqdPlots(x,y,TRUE)*  *dev.off()* |

|  |
| --- |
| > rsqdPlots(x,y,F)  > rsqdPlots(x,y,T) |

|  |
| --- |
| ***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. Then perturbs the noise regressor for n-iterations at every noise level, 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 output the boxplots to a working directory. The directory it outputs the plots to will be printed.

**Examples**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *BoxplotAllPerc(xmatrix = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)* |

|  |
| --- |
| > BoxplotAllPerc(xmatrix = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)  [1] "Your Boxplots are in C:/Users/Ryan/Desktop/Thesis/Thesis20190415/mcperturb\_4\_16\_19 directory"  # This generates multiple folders with plots. Below is one example plot |

|  |
| --- |
| ***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. It then perturbs the noise regressor for n-iterations at every noise level, 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 output the boxplots to a working directory or path. The directory it outputs the plots to will be printed.

**Examples**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *boxplotsAllVars(xmat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)* |

|  |
| --- |
| > boxplotsAllVars(xmat = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)  [1] “Your Boxplots are in C:/Users/Ryan/Desktop/Thesis/Thesis20190415/mcperturb\_4\_16\_19 directory”  # This generates multiple pdf files with plots. Below is one example plot |

|  |
| --- |
| ***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. 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 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 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**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *overallDiagsPlots(xmat = x, response = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration, choice = c(“d"))* |

|  |
| --- |
| > overallDiagsPlots(xmat = x, yvar = y, noiseLevels = noiseLevs, spec.Vars = special.Vars, iter = 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. 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.

**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\_dat)*  *# 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)*  *iteration = 50*  noiseLevelDiagOutList(xmat = x, response = y, special.Vars = special.Vars, noiseLevels = noiseLevs, iteration = iteration) |

|  |
| --- |
| # The tail end of the output is listed below. (Interation 50)  $`Level 10`[[50]]  $`Level 10`[[50]][[1]]  Estimate Std. Error t value  shoulder 0.05684829 0.05835723 0.9741430  chest 0.62353200 0.06597551 9.4509609  bicep 0.47359812 0.18066906 2.6213571  forearm 0.67027513 0.29922969 2.2400021  wrist -0.31011757 0.40184651 -0.7717314  age 0.03538116 0.02453214 1.4422373  height 0.33595792 0.03385610 9.9231141  $`Level 10`[[50]][[2]]  results detection  Determinant 1.911740e-04 1  Farrar Chi-Square 4.305423e+03 1  Red Indicator 6.922432e-01 1  sum of Lambda Invers 5.295173e+01 1  Theil Indicator -9.136677e-02 0  Condition Number 1.204873e+02 1  $`Level 10`[[50]][[3]]  VIF TOL Wi Fi Leamer CVIF Klein IND1 IND2  shoulder 7.690015 0.13003876 557.50124 670.33949 0.3606089 -0.31811674 1 0.0015604651 1.1978856  chest 8.942223 0.11182902 661.85191 795.81073 0.3344085 -0.36991746 1 0.0013419482 1.2229594  bicep 11.947492 0.08369958 912.29100 1096.93869 0.2893088 -0.49423795 1 0.0010043949 1.2616920  forearm 14.789556 0.06761528 1149.12971 1381.71356 0.2600294 -0.61180707 1 0.0008113834 1.2838391  wrist 6.287482 0.15904617 440.62352 529.80573 0.3988059 -0.26009746 0 0.0019085541 1.1579441  age 1.128214 0.88635704 10.68446 12.84699 0.9414654 -0.04667138 0 0.0106362845 0.1564797  height 2.093439 0.47768290 91.11991 109.56258 0.6911461 -0.08660035 0 0.0057321948 0.7192000 |

|  |
| --- |
| ***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. 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.

**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 regressor as it is being perturbed. Coupling regressors should be identified after analysis.

**Examples**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *bestfitvalues = isBestFit(xmat = x, response = y, noiseLevs = noiseLevs,iteration = iteration)*  *vifBestFit = bestfitvalues[[1]]* |

|  |
| --- |
| > bestfitvalues = isBestFit(x = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)  > vifBestFit = bestfitvalues[[1]]  > vifBestFit  shoulder noise chest noise bicep noise forearm noise wrist noise age noise height noise  shoulder -1.222 -0.463 -0.048 -0.080 -0.053 -0.003 -0.029  chest -0.460 -1.351 -0.389 0.002 0.006 -0.025 0.000  bicep -0.025 -0.191 -4.266 -2.715 -0.026 0.001 -0.029  forearm -0.029 0.001 -2.127 -8.405 -3.641 -0.011 -0.007  wrist -0.006 0.001 -0.005 -0.964 -5.108 -0.008 -0.038  age -0.001 -0.010 0.000 -0.014 -0.026 -0.006 -0.001  height -0.012 0.000 -0.026 -0.009 -0.151 0.000 -0.085 |

|  |
| --- |
| ***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. 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.

**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 regressor as it is being perturbed. Coupling regressors should be identified after the analysis is performed.

**Examples**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *rateofchangevalues = isRateofChange(xmat = x, response = y, noiseLevs = noiseLevs, iteration = iteration)*  *vifRateofChange = rateofchangevalues[[1]]* |

|  |
| --- |
| > rateofchangevalues = isRateofChange(x = x, y = y, noiseLevs = noiseLevs, special.Vars = special.Vars, iteration = iteration)  > vifRateofChange = rateofchangevalues[[1]]  > vifRateofChange  shoulder noise chest noise bicep noise forearm noise wrist noise age noise height noise  shoulder -1.116 -0.409 -0.048 -0.067 -0.054 -0.003 -0.031  chest -0.413 -1.223 -0.363 -0.004 -0.013 -0.027 -0.001  bicep -0.021 -0.174 -3.872 -2.428 -0.029 -0.001 -0.029  forearm -0.029 -0.001 -1.906 -7.606 -3.221 -0.013 -0.008  wrist -0.005 -0.001 -0.005 -0.883 -4.556 -0.009 -0.039  age -0.001 -0.009 0.000 -0.013 -0.020 -0.006 -0.001  height -0.012 0.000 -0.024 -0.010 -0.142 -0.001 -0.082 |

|  |
| --- |
| ***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. 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.

**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 regressor as it is being perturbed.

**Examples**

|  |
| --- |
| *# Body dimensions data*  *data(body\_dat)*  *# 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)*  *iteration = 50*  *ov = overallDiagsOut(x = x, y = y, noiseLevs = noiseLevs, iteration = iteration)*  *colnames(ov[[1]]) = c("Noise.Var", "Min.Mean", "Max.Mean", "Difference")* |

|  |
| --- |
| > ov = overallDiagsOut(x = x, y = y, noiseLevs = noiseLevs, iteration = iteration)  > colnames(ov[[1]]) = c("Noise.Var", "Min.Mean", "Max.Mean", "Difference")  > ov[[1]]  Noise.Var Min.Mean Max.Mean Difference  [1,] "shoulder" "0.000158236233098198" "0.000275895693832784" "0.000117659460734585"  [2,] "chest" "0.000158236233098198" "0.000278138711858045" "0.000119902478759847"  [3,] "bicep" "0.000158236233098198" "0.000310614008856761" "0.000152377775758563"  [4,] "forearm" "0.000158236233098198" "0.000349947963314291" "0.000191711730216093"  [5,] "wrist" "0.000158236233098198" "0.00023168191239056" "7.34456792923611e-05"  [6,] "age" "0.000158236233098198" "0.000160303808727468" "2.06757562926986e-06"  [7,] "height" "0.000158236233098198" "0.000174051865060504" "1.58156319623055e-05" |