Car Pricing

2022-08-18

R Markdown

```
#Removing from memory, packages
rm(list=ls())
library(Matrix) # For matrix computations
library(olsrr) # For Variable selection algorithms
```

```
## Warning: package 'olsrr' was built under R version 4.2.1
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

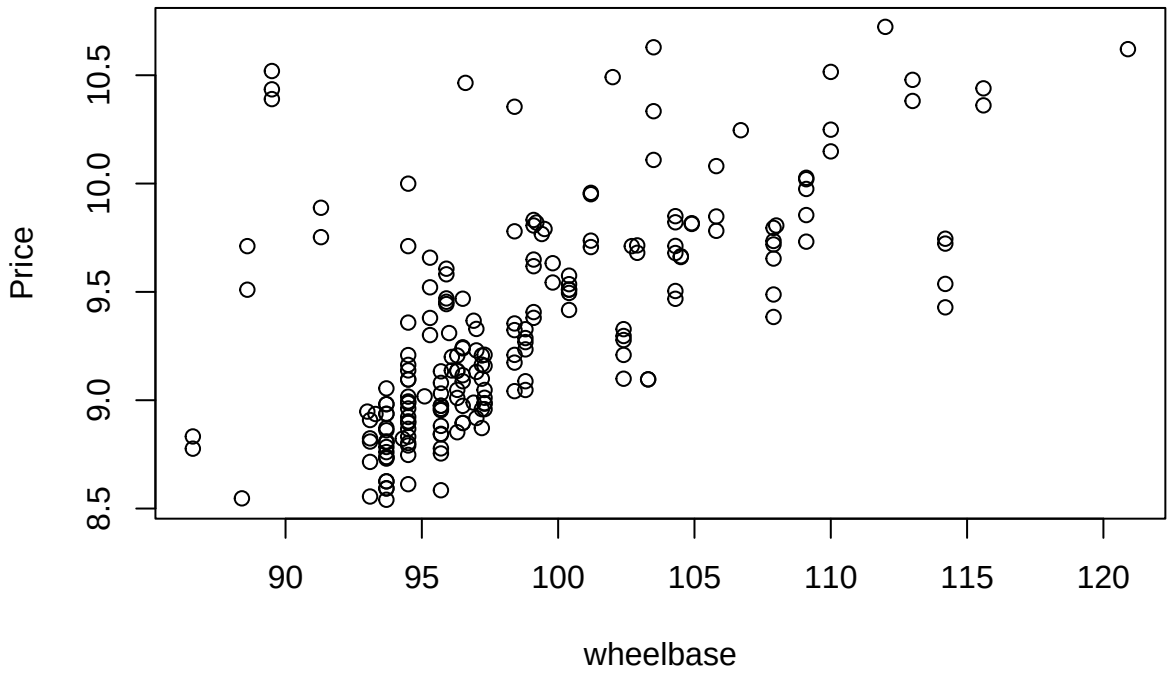
```
library(car) # For VIF
```

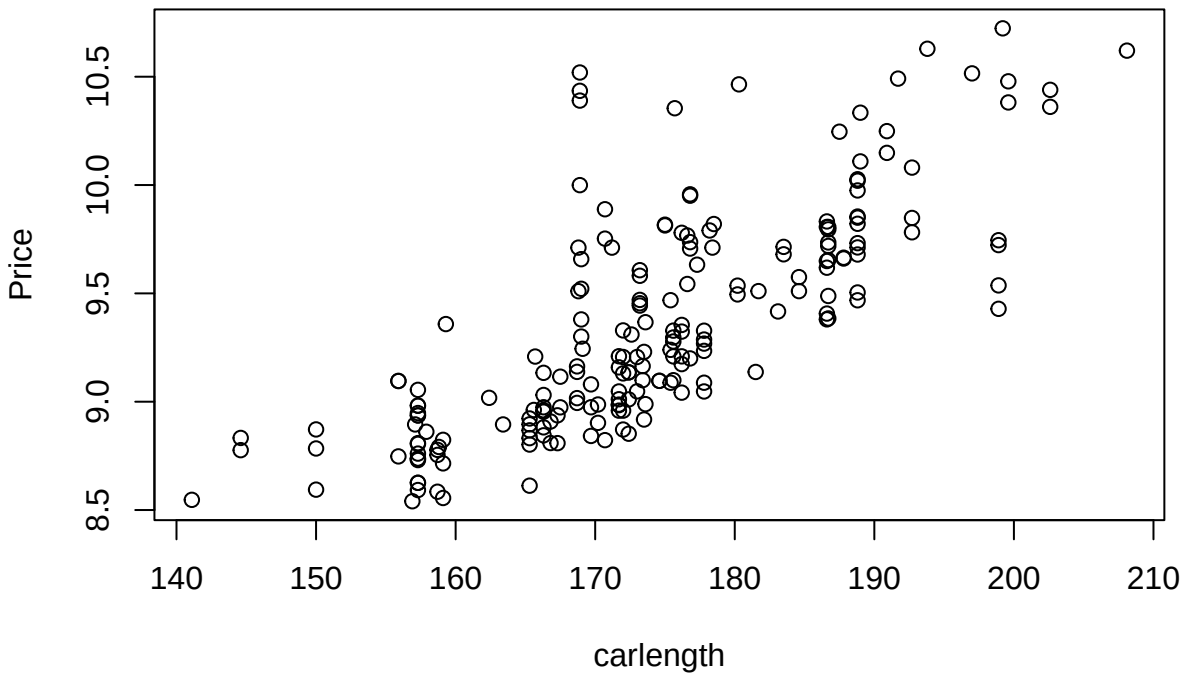
```
## Loading required package: carData
```

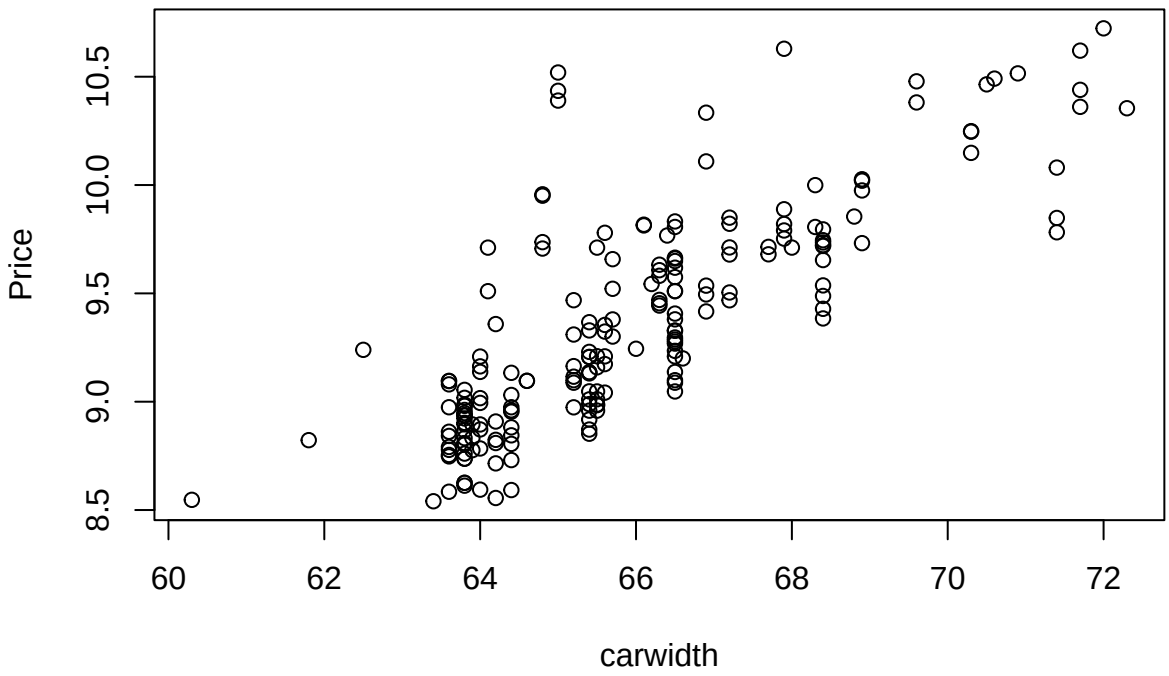
```
library(mctest) # For variance decomposition algoritms


#reading data
library(readxl)
data=read_excel("C:/Users/saubh/Downloads/CarPrice_Assignment.xlsx",col_names=TRUE)
data=data.frame(data)
attach(data)
n=length(price)


#define response, predictors, design matrix, cs matrix
y=data[,14]
y=log(y) ##### Idea of Box cox method was used
data_pred = data[,-c(14)]
p=length(data_pred[1,]) #equals 13 (variable selection needed)
X = cbind(rep(1,n),as.matrix(data_pred))
Xcs = data.frame(sqrt(1/(n-1))*scale.default(data_pred, center=TRUE,scale=TRUE))
Xcs=as.matrix(Xcs)
det(t(Xcs)%*%Xcs) # equals 3.3e-07 (suspecting multicollinearity)
```

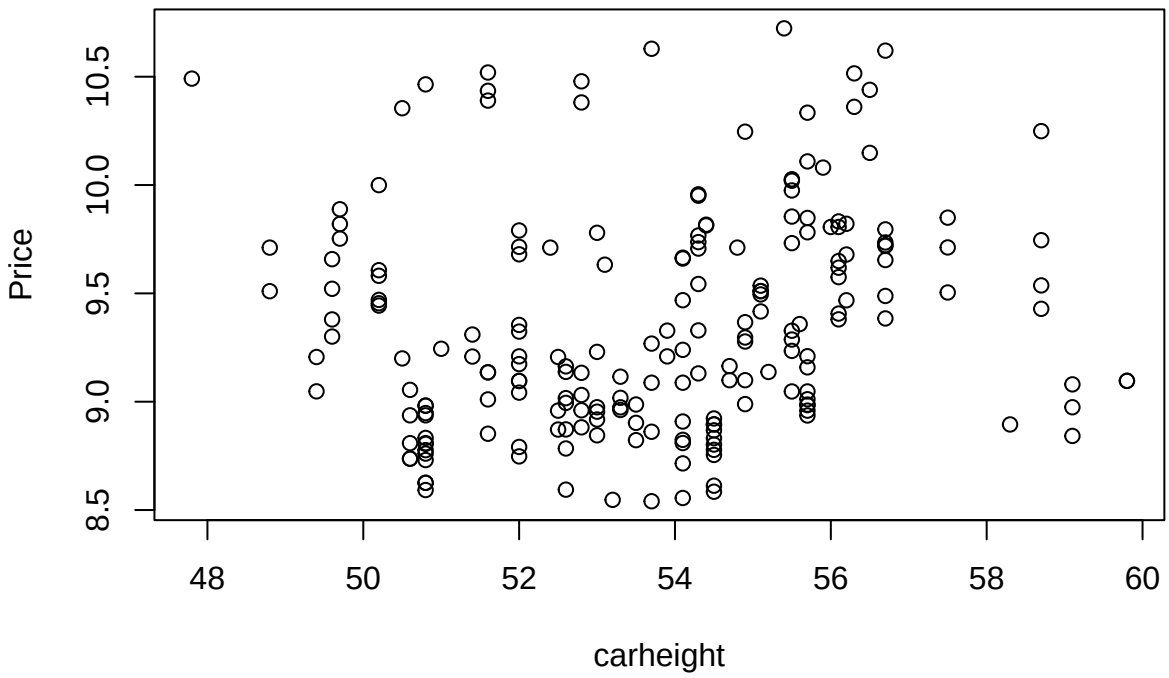
```
## [1] 3.349118e-07
```

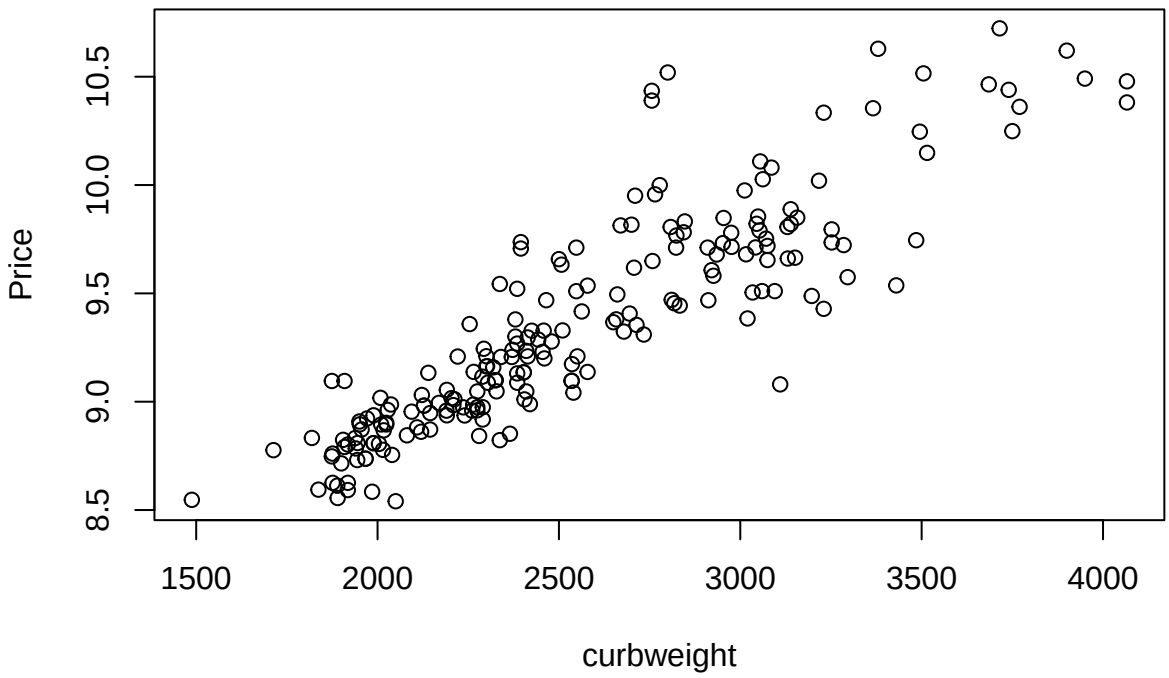
```
#pairwise scatterplot
#par(mfrow=c(3,5))
for(i in 1:12){
  plot(data[,i],y,xlab=names(data)[i],ylab="Price")
}
```

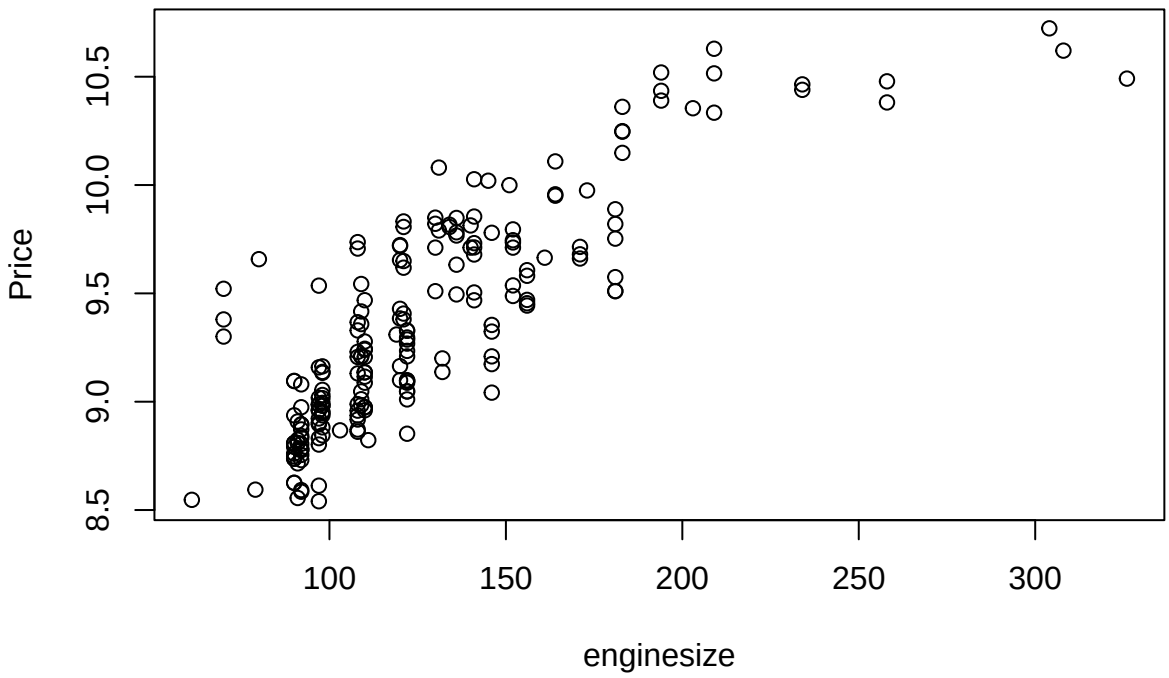


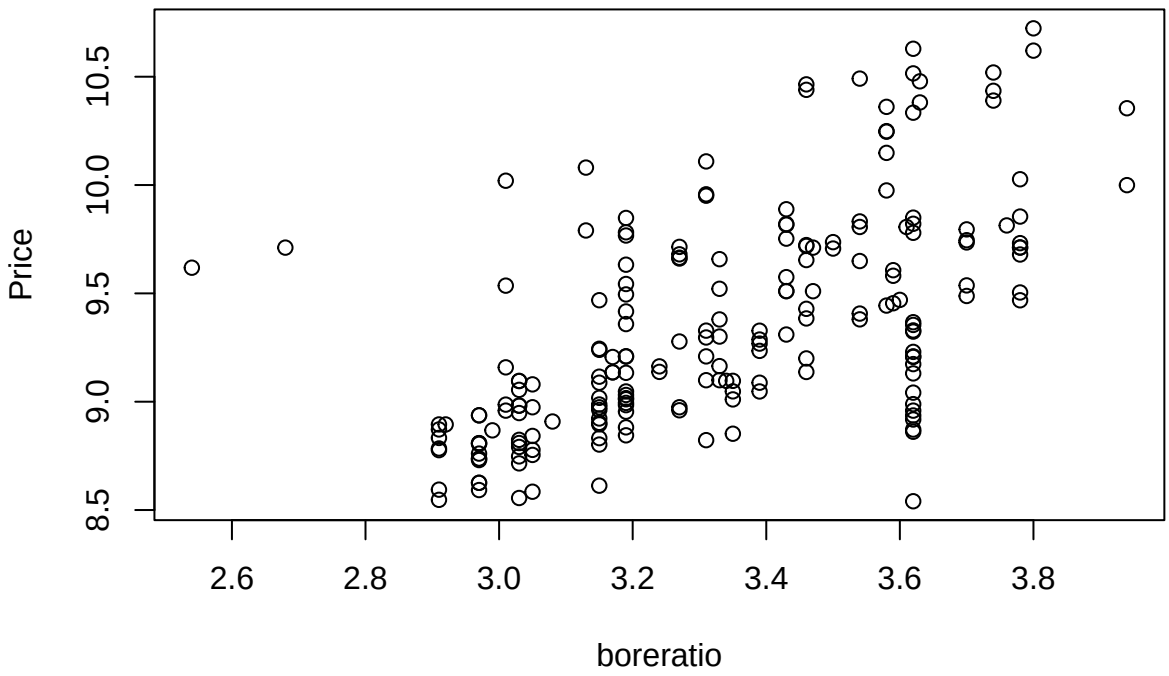

Price
8.5
9.0
9.5
10.0
10.5
140
150
160
170
180
190
200
210
carlength

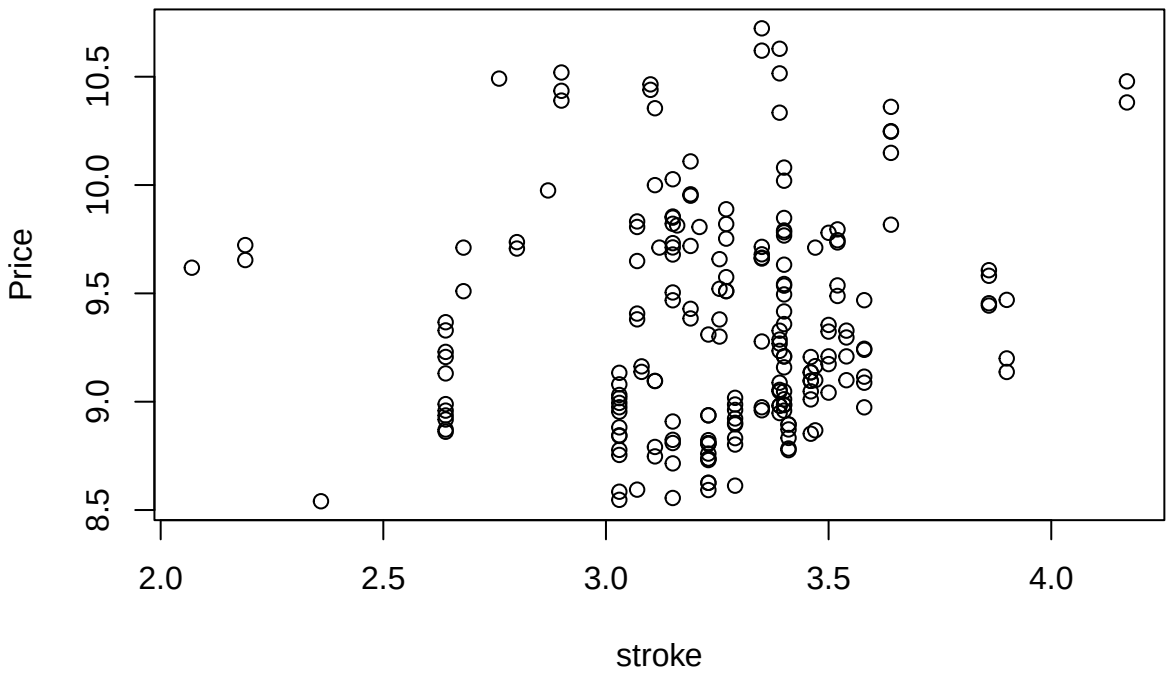

Price
8.5
9.0
9.5
10.0
10.5
60
62
64
66
68
70
72
carwidth

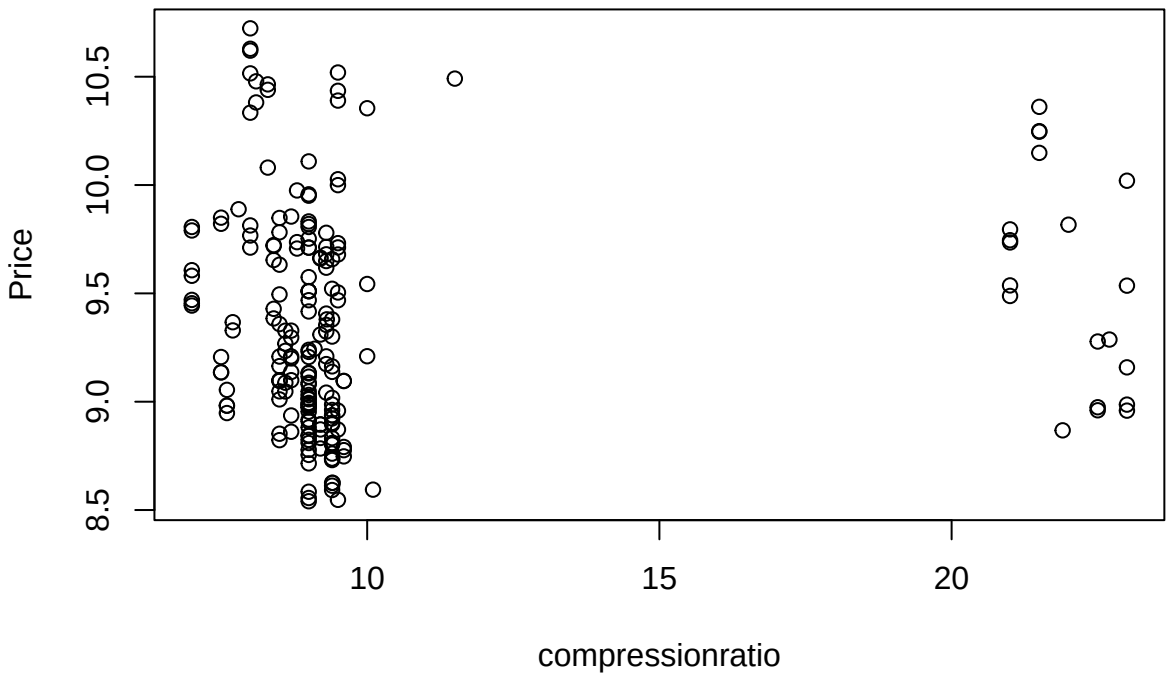

Price
8.5
9.0
9.5
10.0
10.5
48
50
52
54
56
58
60
carheight

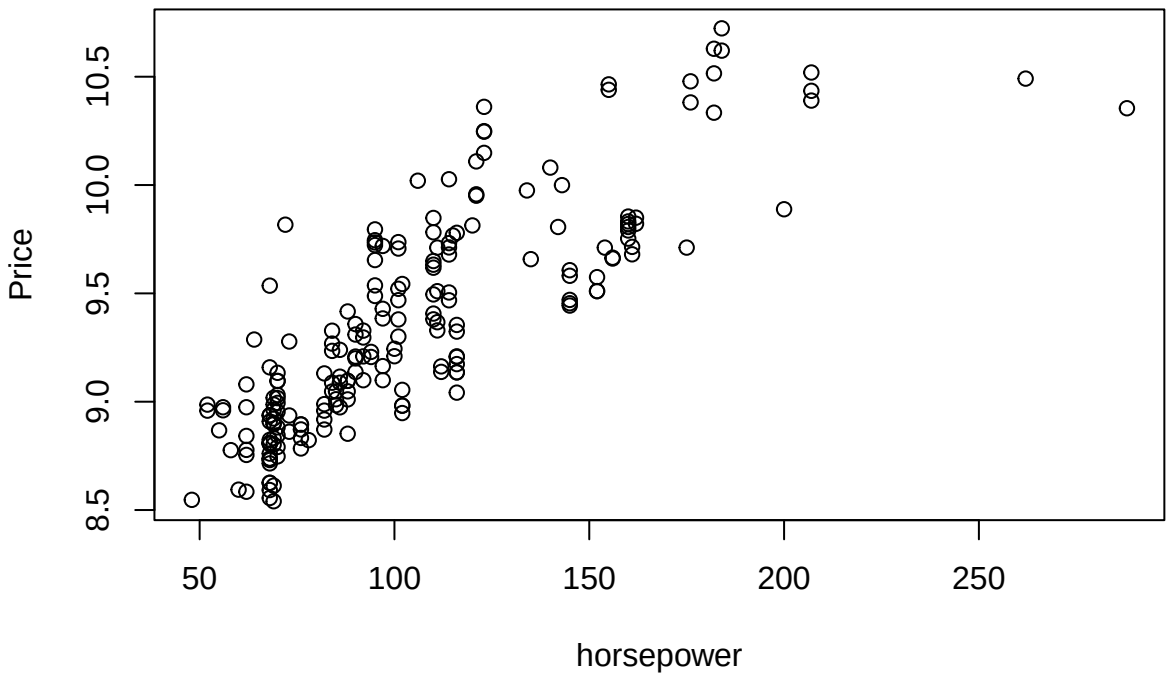

Price
8.5
9.0
9.5
10.0
10.5
1500
2000
2500
3000
3500
4000
curbweight

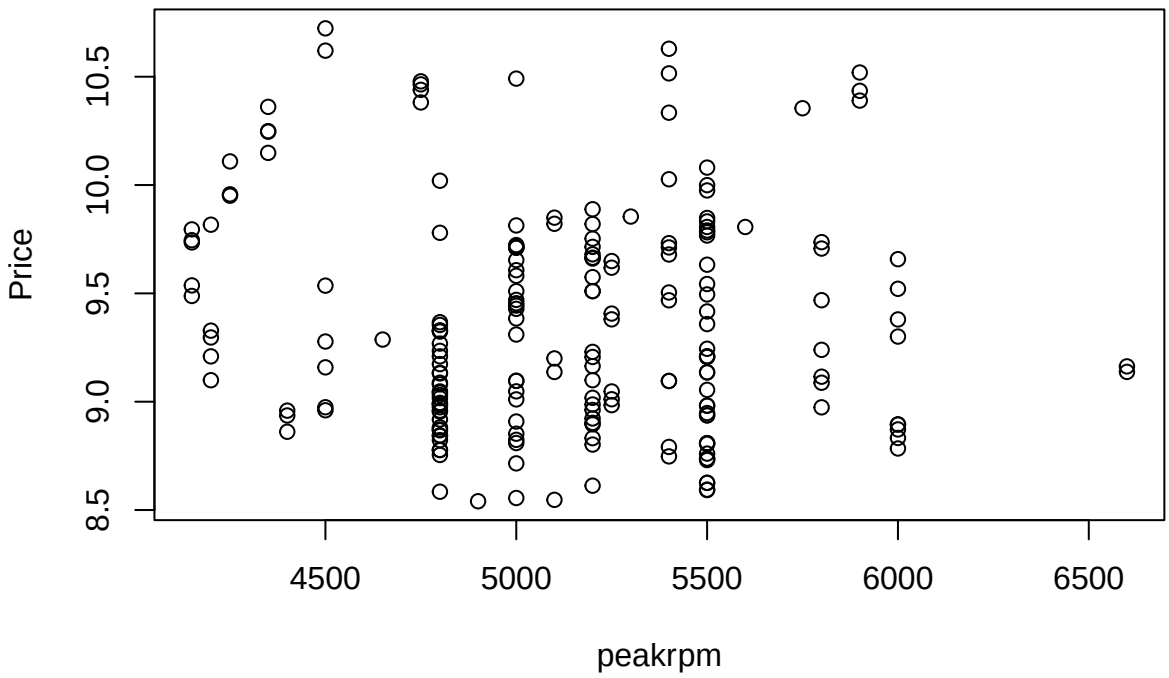

Price
8.5
9.0
9.5
10.0
10.5
100
150
200
250
300
enginesize

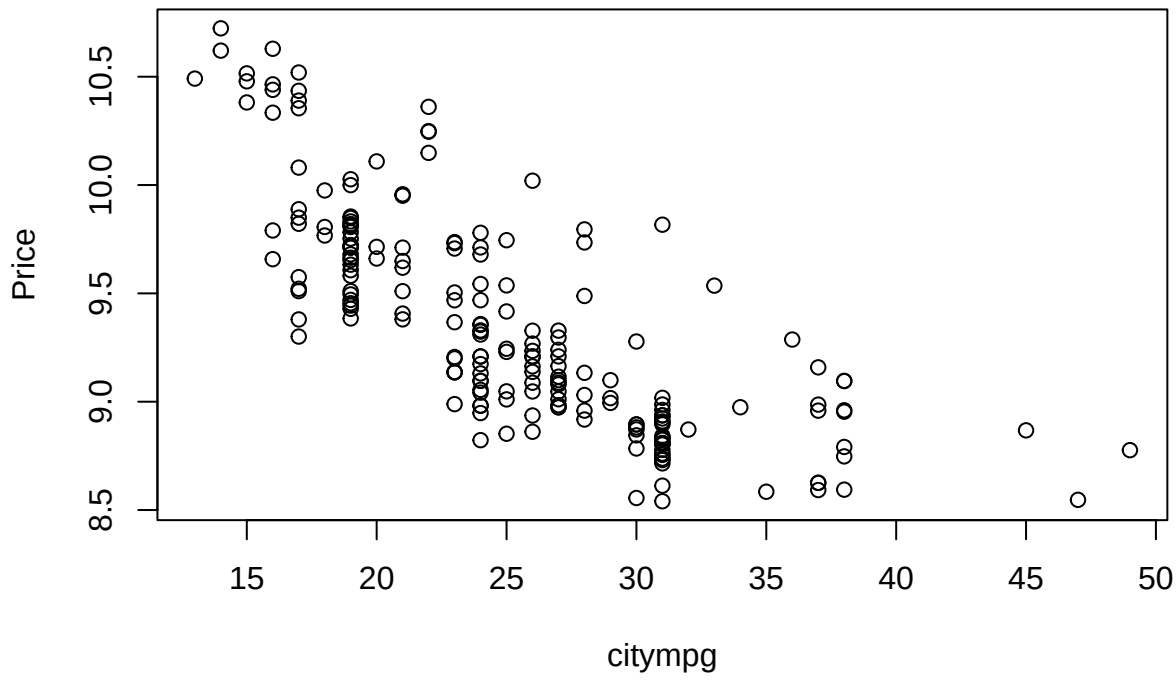

Price
8.5
9.0
9.5
10.0
10.5
2.6
2.8
3.0
3.2
3.4
3.6
3.8
boreratio


Price
8.5
9.0
9.5
10.0
10.5
2.0
2.5
3.0
3.5
4.0
stroke


Price
8.5
9.0
9.5
10.0
10.5
10
15
20
compressionratio


Price
8.5
9.0
9.5
10.0
10.5
50
100
150
200
250
horsepower


Price
8.5
9.0
9.5
10.0
10.5
4500
5000
5500
6000
6500
peakrpm




```
#fitting
fit=lm(y~.,data=data_pred)
summary(fit) # r2=0.8792
```

```
##
## Call:
## lm(formula = y ~ ., data = data_pred)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58368 -0.10611 -0.02615  0.10144  0.44373
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.024e+00  8.663e-01   6.953 5.51e-11 ***
## wheelbase         3.222e-03  5.700e-03   0.565 0.572591
## carlength         6.875e-04  3.152e-03   0.218 0.827575
## carwidth          2.036e-02  1.396e-02   1.458 0.146384
## carheight         4.066e-03  7.700e-03   0.528 0.598093
## curbweight        2.708e-04  9.857e-05   2.747 0.006585 **
## enginesize        2.679e-03  7.851e-04   3.413 0.000785 ***
## boreratio        -1.734e-02  6.785e-02  -0.256 0.798530
## stroke           -1.155e-01  4.418e-02  -2.615 0.009648 **
## compressionratio  2.262e-02  4.704e-03   4.808 3.08e-06 ***
## horsepower        3.050e-03  9.201e-04   3.315 0.001096 **
## peakrpm           9.387e-05  3.806e-05   2.466 0.014541 *
```

```
## citympg          -3.170e-02  1.009e-02  -3.143 0.001938 **
## highwaympg        1.645e-02  9.064e-03   1.815 0.071063 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1809 on 191 degrees of freedom
## Multiple R-squared:  0.8792, Adjusted R-squared:  0.871
## F-statistic:   107 on 13 and 191 DF,  p-value: < 2.2e-16
```

```
#########################################################


# Evaluating all possible subset linear regression models is not advisable since total no. would be 2^1

# Forward selection method
f = ols_step_forward_p(fit, details = FALSE)
summary(f$model) # r2=0.8789
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58880 -0.10819 -0.02394  0.10387  0.45408
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.162e+00  7.455e-01   8.266 2.15e-14 ***
## curbweight        2.837e-04  9.364e-05   3.029 0.002784 **
## horsepower        2.936e-03  8.883e-04   3.305 0.001130 **
## compressionratio  2.266e-02  4.667e-03   4.855 2.47e-06 ***
## citympg          -3.223e-02  9.483e-03  -3.399 0.000820 ***
## carwidth          1.940e-02  1.339e-02   1.448 0.149195
## enginesize        2.621e-03  7.709e-04   3.400 0.000817 ***
## stroke           -1.182e-01  4.242e-02  -2.787 0.005849 **
## peakrpm           9.564e-05  3.633e-05   2.633 0.009154 **
## highwaympg        1.705e-02  8.800e-03   1.937 0.054191 .
## wheelbase         5.110e-03  4.516e-03   1.131 0.259292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1798 on 194 degrees of freedom
## Multiple R-squared:  0.8789, Adjusted R-squared:  0.8727
## F-statistic: 140.8 on 10 and 194 DF,  p-value: < 2.2e-16
```

```
# backward elimination method
b = ols_step_backward_p(fit, details=FALSE)
summary(b$model) # r2=0.8789
```

```
##
## Call:
```

```
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.58880 -0.10819 -0.02394  0.10387  0.45408
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.162e+00  7.455e-01   8.266 2.15e-14 ***
## wheelbase         5.110e-03  4.516e-03   1.131 0.259292
## carwidth          1.940e-02  1.339e-02   1.448 0.149195
## curbweight        2.837e-04  9.364e-05   3.029 0.002784 **
## enginesize        2.621e-03  7.709e-04   3.400 0.000817 ***
## stroke           -1.182e-01  4.242e-02  -2.787 0.005849 **
## compressionratio  2.266e-02  4.667e-03   4.855 2.47e-06 ***
## horsepower        2.936e-03  8.883e-04   3.305 0.001130 **
## peakrpm           9.564e-05  3.633e-05   2.633 0.009154 **
## citympg          -3.223e-02  9.483e-03  -3.399 0.000820 ***
## highwaympg        1.705e-02  8.800e-03   1.937 0.054191 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1798 on 194 degrees of freedom
## Multiple R-squared:  0.8789, Adjusted R-squared:  0.8727
## F-statistic: 140.8 on 10 and 194 DF,  p-value: < 2.2e-16
```

```
# Step-wise selection method
s = ols_step_both_p(fit, details=FALSE)
summary(s$model) # r2=0.8781
```

```
##
## Call:
## lm(formula = paste(response, "~", paste(preds, collapse = " + ")),
##     data = l)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60078 -0.11198 -0.02174  0.09495  0.45516
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.138e+00  7.457e-01   8.232 2.61e-14 ***
## curbweight        3.308e-04  8.395e-05   3.940 0.000113 ***
## horsepower        2.538e-03  8.160e-04   3.110 0.002153 **
## compressionratio  2.195e-02  4.628e-03   4.743 4.06e-06 ***
## citympg          -3.091e-02  9.416e-03  -3.282 0.001221 **
## carwidth          2.637e-02  1.190e-02   2.216 0.027862 *
## enginesize        2.598e-03  7.711e-04   3.368 0.000911 ***
## stroke           -1.179e-01  4.245e-02  -2.777 0.006014 **
## peakrpm           9.546e-05  3.635e-05   2.626 0.009324 **
## highwaympg        1.595e-02  8.753e-03   1.822 0.069985 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.1799 on 195 degrees of freedom
## Multiple R-squared:  0.8781, Adjusted R-squared:  0.8725
## F-statistic: 156.1 on 9 and 195 DF,  p-value: < 2.2e-16
```

```
# g = r2(M)/r2(whole)
# So for forward/backward selection, g is close to 1


#####################################################


# we drop the boreratio,carlength,carheight, updating X,Xcs
data_pred2=subset(data_pred,select=-c(carlength,boreratio,carheight))
X = cbind(rep(1,n),as.matrix(data_pred2))
Xcs = data.frame(sqrt(1/(n-1))*scale.default(data_pred2, center=TRUE,scale=TRUE))
Xcs=as.matrix(Xcs)
ycs=(y-mean(y))/((n-1)*(sd(y)))


# fitting MLRM with CS data
scaled_fit = lm(ycs~.-1,data=data.frame(Xcs))
summary(scaled_fit)
```

```
##
## Call:
## lm(formula = ycs ~ . - 1, data = data.frame(Xcs))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.005729 -0.001053 -0.000233  0.001011  0.004418
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## wheelbase         0.004276   0.003770   1.134 0.258065
## carwidth          0.005783   0.003983   1.452 0.148148
## curbweight        0.020528   0.006759   3.037 0.002714 **
## enginesize        0.015168   0.004450   3.409 0.000792 ***
## stroke           -0.005152   0.001844  -2.794 0.005723 **
## compressionratio  0.012507   0.002569   4.867 2.33e-06 ***
## horsepower        0.016135   0.004869   3.314 0.001097 **
## peakrpm           0.006340   0.002402   2.639 0.008976 **
## citympg          -0.029306   0.008599  -3.408 0.000795 ***
## highwaympg        0.016314   0.008400   1.942 0.053573 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001745 on 195 degrees of freedom
## Multiple R-squared:  0.8789, Adjusted R-squared:  0.8727
## F-statistic: 141.6 on 10 and 195 DF,  p-value: < 2.2e-16
```

```
# correlation matrix
corr_mat=t(Xcs)%*%(Xcs)
corr_mat # high pairwise correlation between predictors is present (suspecting multicollinearity)
```

```
##                    wheelbase   carwidth curbweight  enginesize      stroke
## wheelbase          1.0000000  0.7951436  0.7763863  0.56932868  0.16095905
## carwidth           0.7951436  1.0000000  0.8670325  0.73543340  0.18294169
## curbweight         0.7763863  0.8670325  1.0000000  0.85059407  0.16879004
## enginesize         0.5693287  0.7354334  0.8505941  1.00000000  0.20312859
## stroke             0.1609590  0.1829417  0.1687900  0.20312859  1.00000000
## compressionratio   0.2497858  0.1811286  0.1513617  0.02897136  0.18611011
## horsepower         0.3532945  0.6407321  0.7507393  0.80976865  0.08093954
## peakrpm           -0.3604687 -0.2200123 -0.2662432 -0.24465983 -0.06796375
## citympg           -0.4704136 -0.6427043 -0.7574138 -0.65365792 -0.04214475
## highwaympg        -0.5440819 -0.6772179 -0.7974648 -0.67746991 -0.04393093
##                   compressionratio  horsepower     peakrpm     citympg
## wheelbase               0.24978585  0.35329448 -0.36046875 -0.47041361
## carwidth                0.18112863  0.64073208 -0.22001230 -0.64270434
## curbweight              0.15136174  0.75073925 -0.26624318 -0.75741378
## enginesize              0.02897136  0.80976865 -0.24465983 -0.65365792
## stroke                  0.18611011  0.08093954 -0.06796375 -0.04214475
## compressionratio        1.00000000 -0.20432623 -0.43574051  0.32470142
## horsepower             -0.20432623  1.00000000  0.13107251 -0.80145618
## peakrpm                -0.43574051  0.13107251  1.00000000 -0.11354438
## citympg                 0.32470142 -0.80145618 -0.11354438  1.00000000
## highwaympg              0.26520139 -0.77054389 -0.05427481  0.97133704
##                    highwaympg
## wheelbase         -0.54408192
## carwidth          -0.67721792
## curbweight        -0.79746479
## enginesize        -0.67746991
## stroke            -0.04393093
## compressionratio   0.26520139
## horsepower        -0.77054389
## peakrpm           -0.05427481
## citympg            0.97133704
## highwaympg         1.00000000
```

```
# Determinant of X'X
det(t(Xcs)%*%Xcs) # equals 1.35e-05 (suspecting multicollinearity)
```

```
## [1] 1.356289e-05
```

```
# Computing VIF
round(vif(scaled_fit),2)
```

```
## Warning in vif.default(scaled_fit): No intercept: vifs may not be sensible.
```

```
##        wheelbase         carwidth       curbweight       enginesize
##             4.67             5.21            15.01             6.51
##           stroke compressionratio       horsepower          peakrpm
##             1.12             2.17             7.79             1.90
##          citympg       highwaympg
##            24.30            23.19
```

```r
# We use  cut-off 10.
# Multi-collinearity is suspected and the regression coefficients corresponding to curbweight, highwaym


# Condition indices
e = eigen(corr_mat)$values  # eigen(A) computes both eigenvalues and eigenvectors
Condition_indices = array(0)
for(i in 1:length(e))
{
  Condition_indices[i] = max(e)/e[i]
}
round(Condition_indices,2)
```

```
##  [1]   1.00   2.67   5.56   9.37  10.11  17.49  35.95  62.96 101.24 243.46
```

```r
# We use cut-off 25.
# Last four condition indices are larger than 25.
# Last four principal components of the predictor variables are suspected to be responsible for multi-c


# Calculating measures based on variance decomposition
scaled_fit_intercept = lm(y ~ ., data = data.frame(Xcs))
VP = eigprop(scaled_fit_intercept, Inter=FALSE)
VP$pi[c(7,9,8,3,10),]
```

```
##      wheelbase     carwidth   curbweight   enginesize      stroke
## 7  0.258547720 0.7955971264 0.0126775826 0.1237058744 0.003380628
## 9  0.285083125 0.0045815702 0.9743783365 0.1363310742 0.001059359
## 8  0.215837989 0.1044133174 0.0019231834 0.5510365296 0.024772665
## 3  0.003845473 0.0001323175 0.0001674864 0.0003940061 0.813182732
## 10 0.009830957 0.0072100410 0.0071348413 0.0498118113 0.011116077
##    compressionratio   horsepower    peakrpm      citympg   highwaympg
## 7      0.0215145347 0.0065624467 0.12127661 8.145205e-04 0.0017293219
## 9      0.2463499096 0.0733056838 0.03420179 2.559639e-02 0.0140254593
## 8      0.0051883280 0.7369467746 0.16912461 2.855819e-04 0.0230609106
## 3      0.0001899242 0.0009064886 0.05235804 1.716118e-05 0.0001192479
## 10     0.0087898641 0.1089218345 0.01131816 9.387845e-01 0.9241836241
```

```r
# Regression coefficient correspond to :
# carwidth is suffering from the 7th pc. The measure is 0.79
# curbweight is suffering from the 9th pc. The measure is 0.97
# stroke is suffering from the 3rd pc. The measure is 0.81
# horsepower is suffering from the 8th pc. The measure is 0.73
# citympg is suffering from the 10th pc. The measure is 0.93
# highwaympg is suffering from the 10th pc. The measure is 0.92




############ PC REGRESSION #############
pca = prcomp(Xcs)
pca
```

```
## Standard deviations (1, .., p=10):
##  [1] 0.16168414 0.09898412 0.06858343 0.05280622 0.05084676 0.03865789
##  [7] 0.02696705 0.02037598 0.01606878 0.01036220
##
## Rotation (n x k) = (10 x 10):
##                           PC1         PC2         PC3          PC4         PC5
## wheelbase          0.321675796 -0.29350643  0.13125872 -0.266763140 -0.53481463
## carwidth           0.382912594 -0.16192752  0.02572578 -0.262279485 -0.09225934
## curbweight         0.415504492 -0.11917170  0.04911328 -0.049982240  0.05467128
## enginesize         0.381042448 -0.05618778 -0.04959359  0.282978977  0.40397773
## stroke             0.077279467 -0.21532032 -0.93369364  0.179148820 -0.19020244
## compressionratio -0.007058283 -0.59477638 -0.01988290 -0.483030223  0.43602487
## horsepower         0.363671300  0.23106403 -0.08231035  0.058430375  0.49907480
## peakrpm           -0.068098222  0.53576773 -0.30859438 -0.712656903  0.08151663
## citympg           -0.374996408 -0.28060028 -0.02000226 -0.027193513  0.14177905
## highwaympg        -0.385391845 -0.22578428 -0.05150709 -0.005061678  0.19276413
##                           PC6         PC7         PC8          PC9         PC10
## wheelbase         -0.31684921  0.42319237 -0.29215713 -0.264790636 -0.03170910
## carwidth          -0.29288094 -0.78436962  0.21470254 -0.035467512  0.02869205
## curbweight         0.04373586  0.16801249 -0.04944459  0.877680677 -0.04843218
## enginesize        -0.35929495  0.34552816  0.55101518 -0.216139982 -0.08425060
## stroke             0.06611404 -0.02367136 -0.04841679  0.007895789  0.01649373
## compressionratio   0.43118054  0.08320891  0.03087469 -0.167775849 -0.02043683
## horsepower        -0.11262205 -0.08708039 -0.69725336 -0.173422599  0.13632123
## peakrpm           -0.18281219  0.18467115  0.16477792  0.058436518 -0.02167789
## citympg           -0.47275313  0.05418397 -0.02424210  0.180991663  0.70684031
## highwaympg        -0.47215927 -0.07712475 -0.21280370  0.130877143 -0.68509935
```

```
# prcomp(A) performs principal component analysis on the data matrix A.

pc_variances = summary(pca) # summary(prcomp()) prints sample variance of the proncipal components.
pc_variances
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.1617 0.09898 0.06858 0.05281 0.05085 0.03866 0.02697
## Proportion of Variance 0.5333 0.19988 0.09596 0.05689 0.05274 0.03049 0.01484
## Cumulative Proportion  0.5333 0.73317 0.82912 0.88601 0.93875 0.96924 0.98407
##                            PC8     PC9    PC10
## Standard deviation     0.02038 0.01607 0.01036
## Proportion of Variance 0.00847 0.00527 0.00219
## Cumulative Proportion  0.99254 0.99781 1.00000
```

```
pc = pca$x # The i-th column of prcomp(A)$x provides the i-th principal component of the data matrix A.
pc_data = data.frame(pc)
summary(lm(ycs~.-1,pc_data)) # Fit MLRM for the standarized response on the principal components obtain
```

```
##
## Call:
## lm(formula = ycs ~ . - 1, data = pc_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.005729 -0.001053 -0.000233  0.001011  0.004418
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## PC1   0.0275510  0.0007554  36.470  < 2e-16 ***
## PC2  -0.0001546  0.0012340  -0.125  0.90043
## PC3   0.0019895  0.0017809   1.117  0.26531
## PC4  -0.0092159  0.0023130  -3.984 9.55e-05 ***
## PC5   0.0184219  0.0024022   7.669 8.04e-13 ***
## PC6   0.0006269  0.0031596   0.198  0.84292
## PC7   0.0040460  0.0045293   0.893  0.37280
## PC8  -0.0049957  0.0059944  -0.833  0.40565
## PC9   0.0056652  0.0076012   0.745  0.45699
## PC10 -0.0324110  0.0117873  -2.750  0.00653 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001745 on 195 degrees of freedom
## Multiple R-squared:  0.8789, Adjusted R-squared:  0.8727
## F-statistic: 141.6 on 10 and 195 DF,  p-value: < 2.2e-16
```

```
fit_good=lm(ycs~.-1,pc_data[,c(1,4,5,10)])
summary(fit_good) # Fit final multiple linear regression model with significant principal components.
```

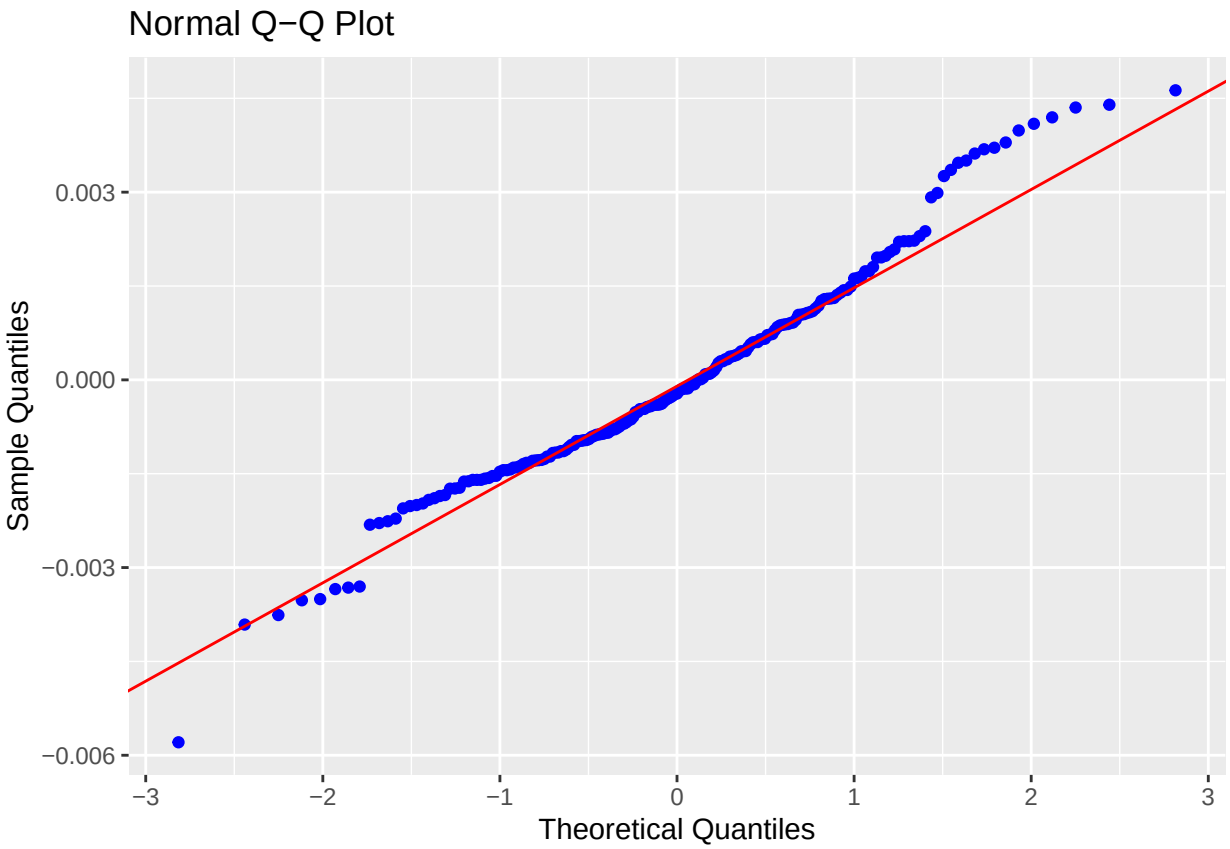
```
##
## Call:
## lm(formula = ycs ~ . - 1, data = pc_data[, c(1, 4, 5, 10)])
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0057926 -0.0011610 -0.0002189  0.0009591  0.0046280
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## PC1   0.0275510  0.0007504  36.713  < 2e-16 ***
## PC4  -0.0092159  0.0022977  -4.011 8.53e-05 ***
## PC5   0.0184219  0.0023863   7.720 5.36e-13 ***
## PC10 -0.0324110  0.0117094  -2.768  0.00617 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.001733 on 201 degrees of freedom
## Multiple R-squared:  0.8769, Adjusted R-squared:  0.8744
## F-statistic: 357.8 on 4 and 201 DF,  p-value: < 2.2e-16
```

```
round(pca$rotation[,c(1,4,5,10)],2) # This provides the linear combination for the significant principa
```

```
##                    PC1   PC4   PC5  PC10
## wheelbase         0.32 -0.27 -0.53 -0.03
## carwidth          0.38 -0.26 -0.09  0.03
## curbweight        0.42 -0.05  0.05 -0.05
## enginesize        0.38  0.28  0.40 -0.08
## stroke            0.08  0.18 -0.19  0.02
```

```
## compressionratio -0.01 -0.48  0.44 -0.02
## horsepower        0.36  0.06  0.50  0.14
## peakrpm          -0.07 -0.71  0.08 -0.02
## citympg          -0.37 -0.03  0.14  0.71
## highwaympg       -0.39 -0.01  0.19 -0.69
```

```
############## Assumption Verification ####################
# Normality assumption
ols_plot_resid_qq(fit_good) # Outliers present and hence we can note departure from normality
```

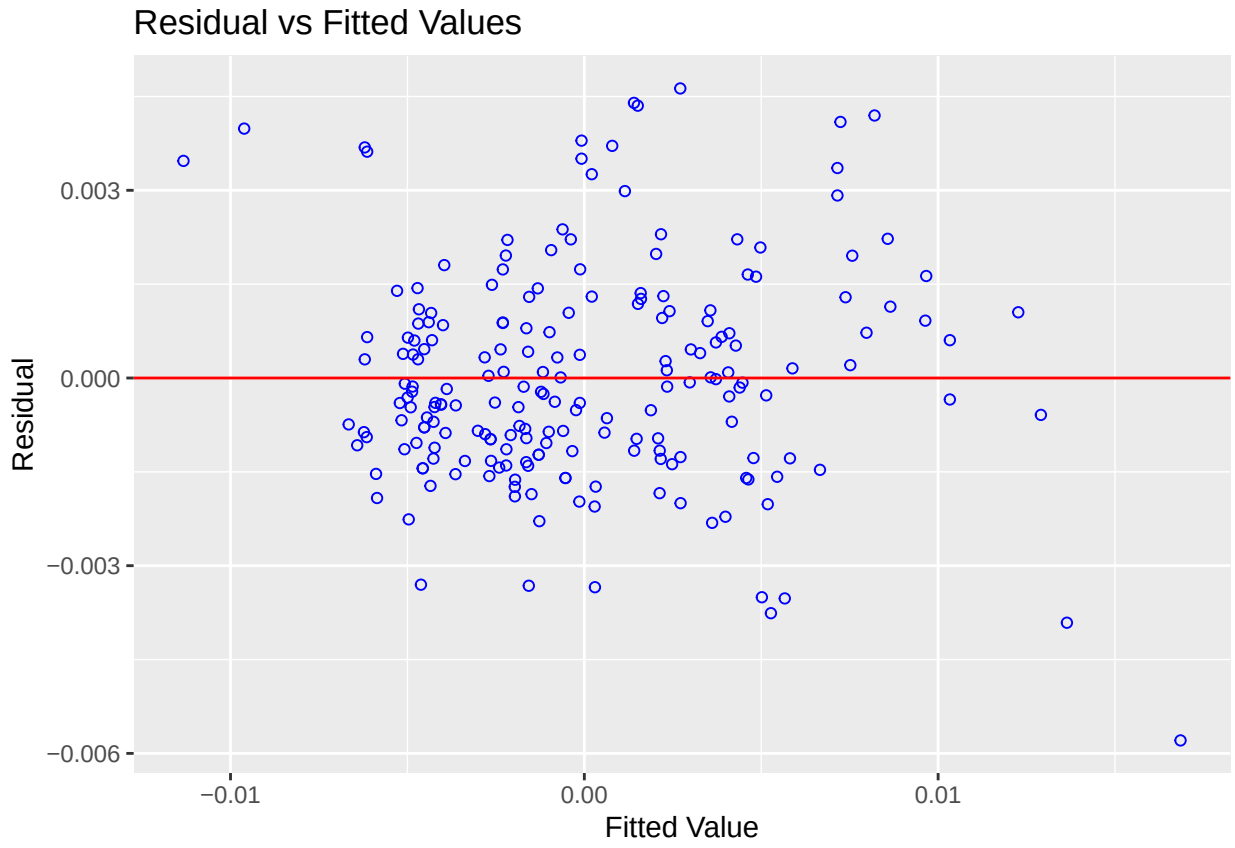



```
ols_test_normality(fit_good) # All tests result in rejection of null hypothesis; therefore normality as
```

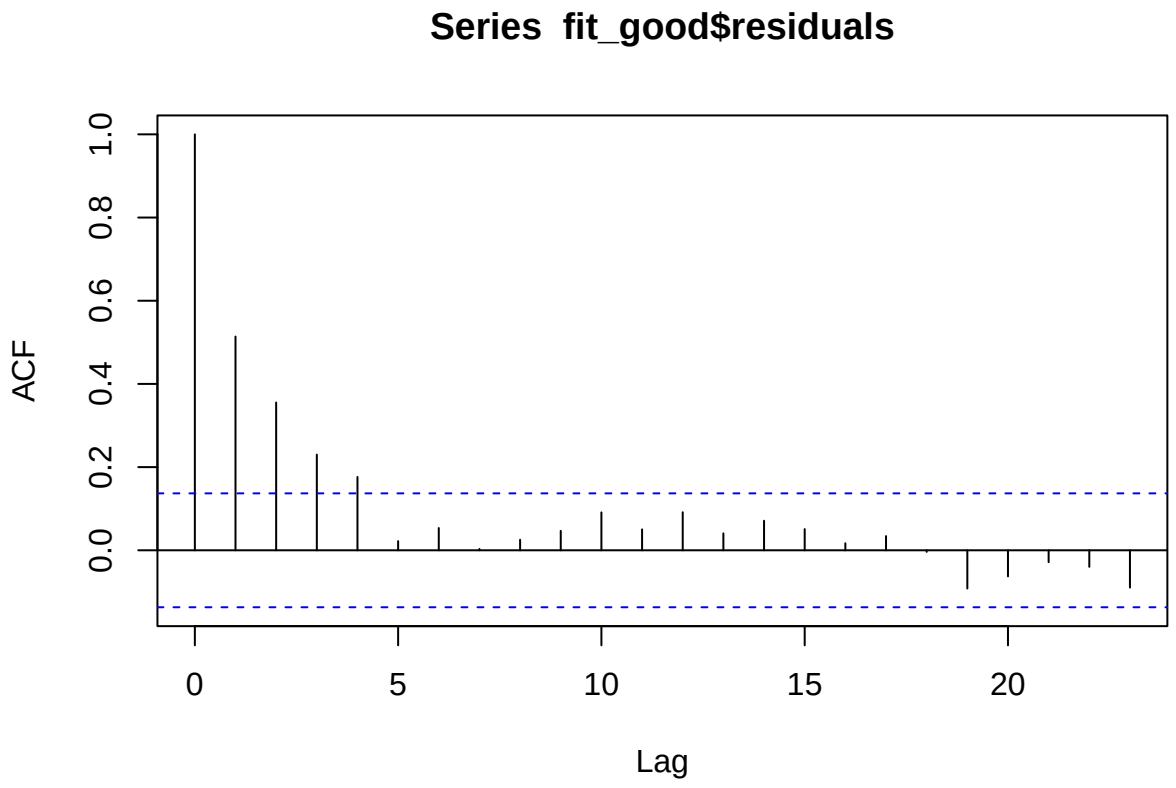
```
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9746          9e-04
## Kolmogorov-Smirnov        0.0601         0.4487
## Cramer-von Mises         68.0947         0.0000
## Anderson-Darling          1.6648          3e-04
## -----------------------------------------------
```

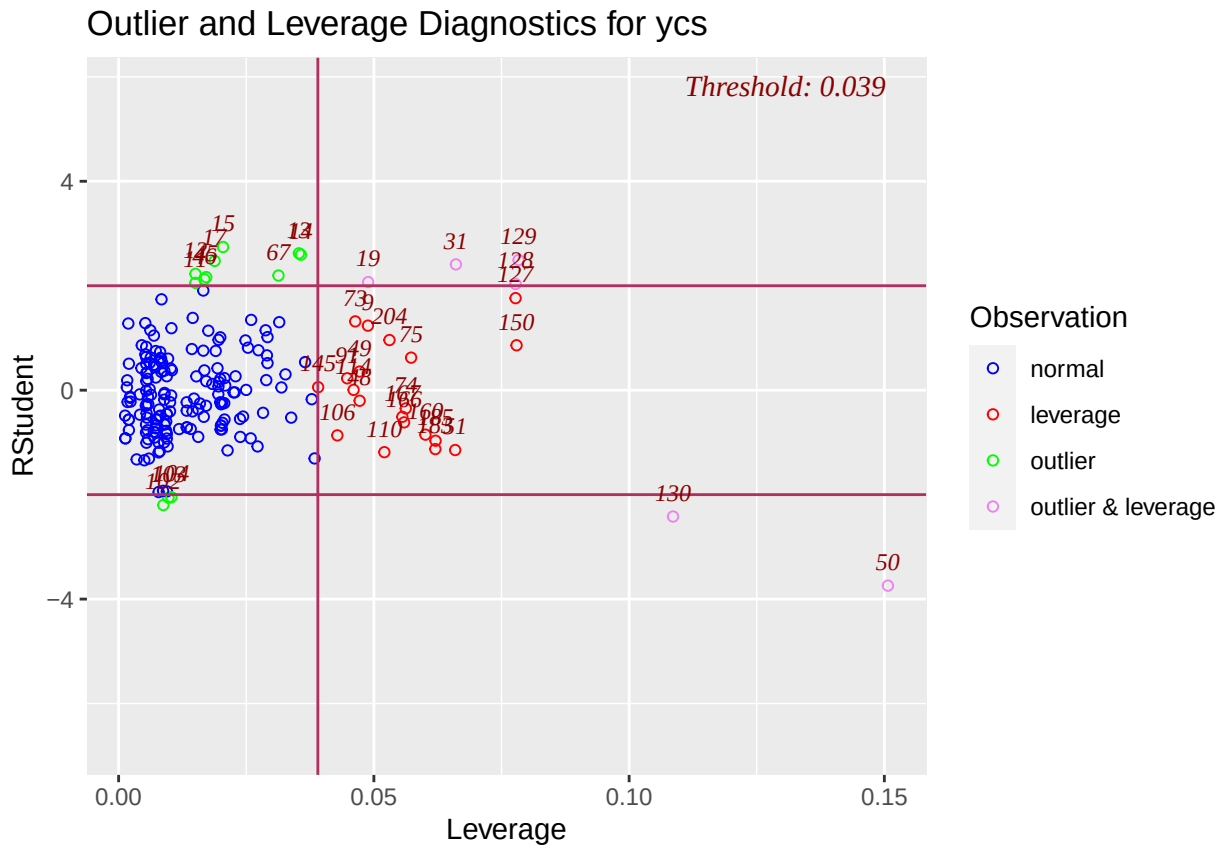
```
# Homoscedasticity assumption
ols_plot_resid_fit(fit_good) # Clustering is observed  and hence homoscedasticity assumption does not h
```




```
# Assumption of random errors being uncorrelated
acf(fit_good$residuals,plot=TRUE) # Autocorrelation is present
```




```
############ Outlier detection ####################

lev1=ols_plot_resid_lev(fit_good) # Computing leverage measure for all observations
```




```
# leverage points
which(lev1$data$color=="leverage")
```

```
##  [1]   9  48  49  51  73  74  75  91 106 110 114 127 145 150 160 166 167 183 185
## [20] 204
```

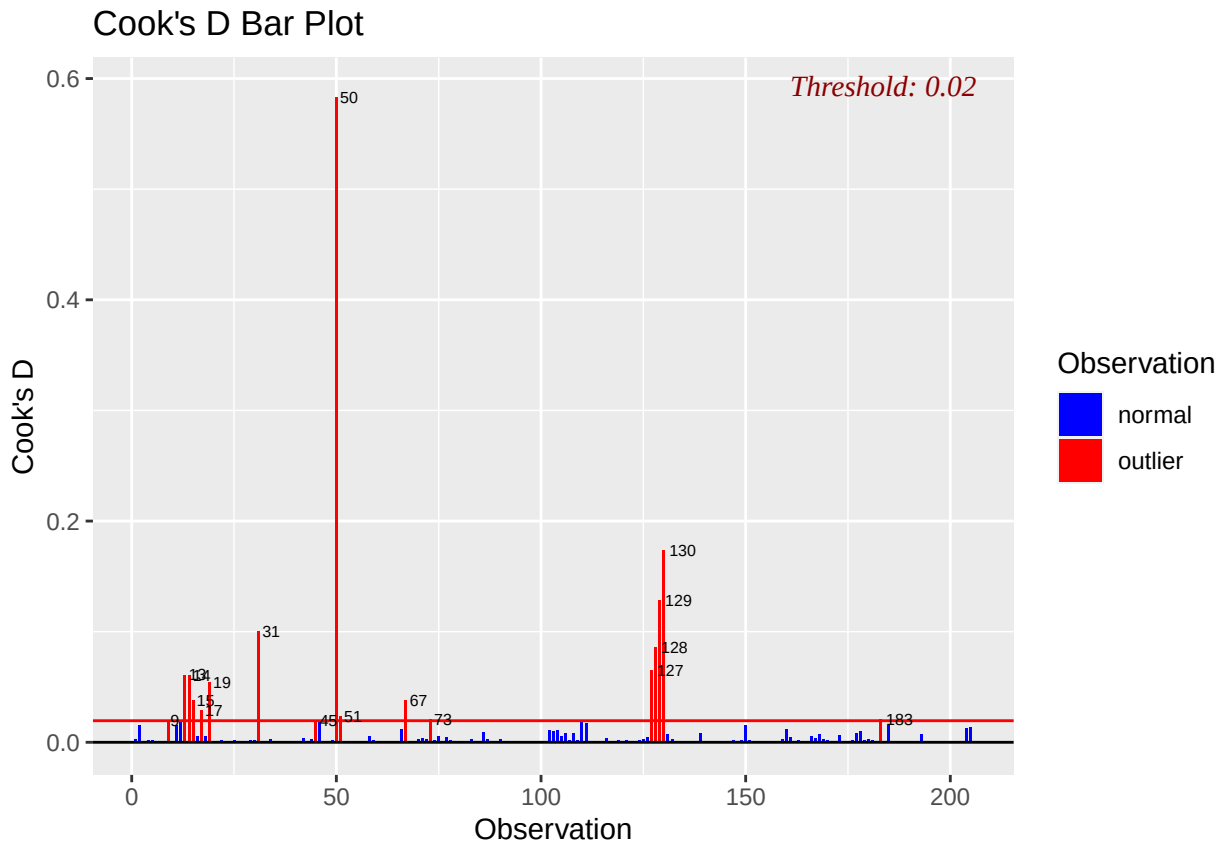
```
# outliers
which(lev1$data$color=="outlier")
```

```
##  [1]  11  12  13  14  15  17  45  46  67 102 103 104
```

```
# outlier and leverage
which(lev1$data$color=="outlier & leverage")
```

```
## [1]  19  31  50 128 129 130
```

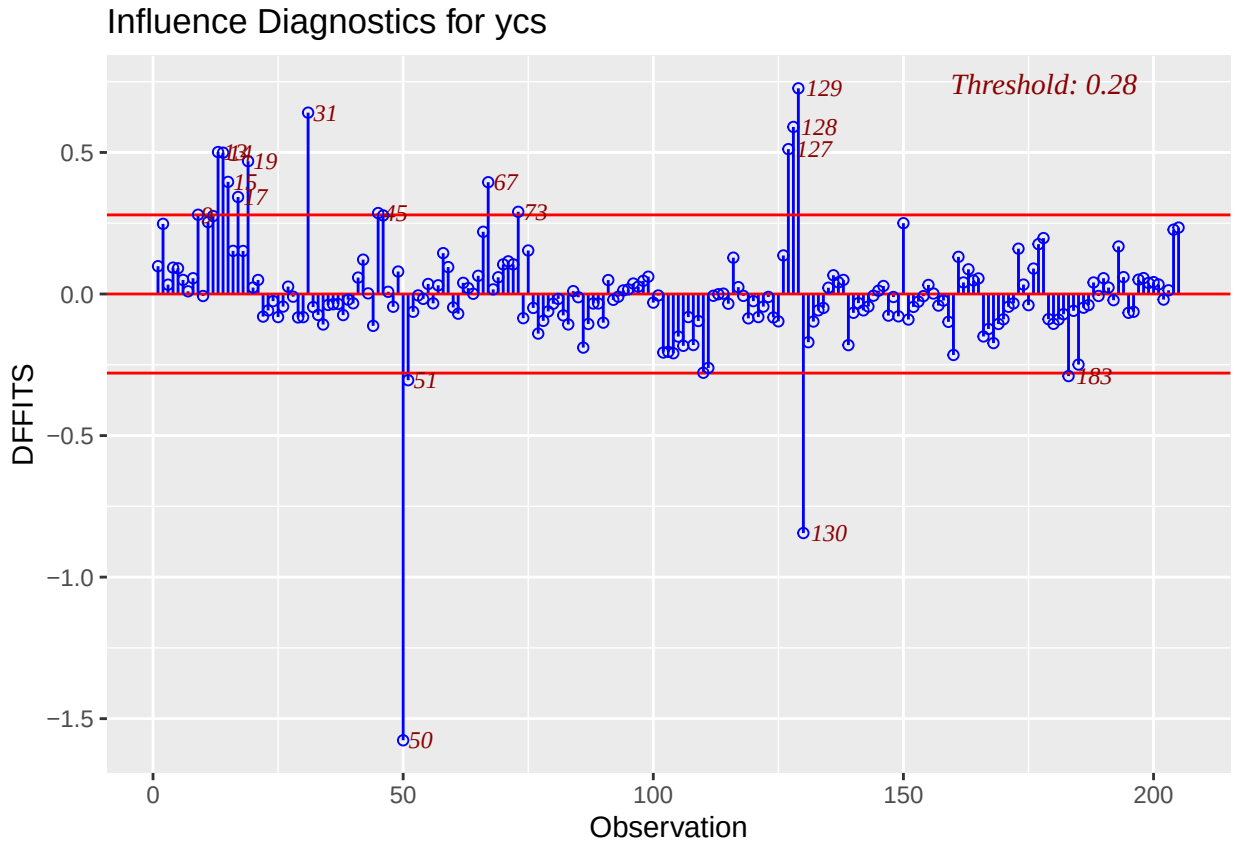
```
lev2=ols_plot_cooksd_bar(fit_good) # Computing Cook's distance statistics
```




```
which(lev2$data$color=="outlier")
```

```
##  [1]   9  13  14  15  17  19  31  45  50  51  67  73 127 128 129 130 183
```

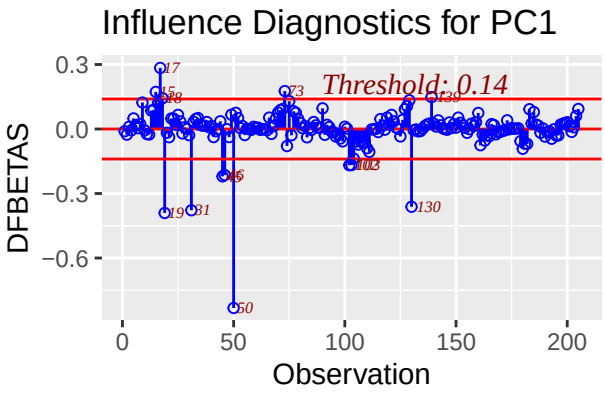
```
lev3=ols_plot_dffits(fit_good) # Computing DFFFITS statistics
```

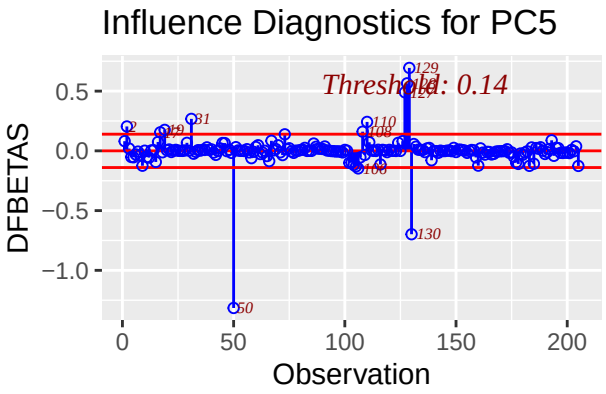


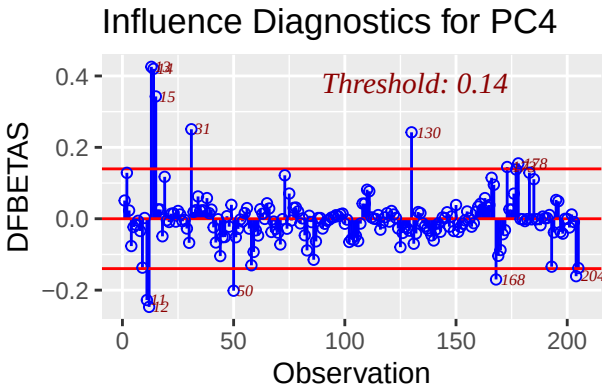

```
which(lev3$data$color=="outlier")
```

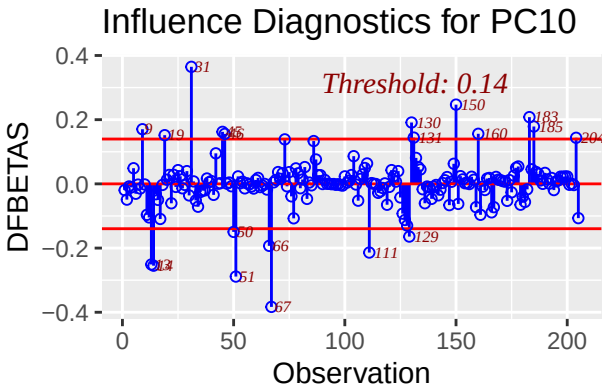
```
##  [1]   9  13  14  15  17  19  31  45  50  51  67  73 127 128 129 130 183
```

```
ols_plot_dfbetas(fit_good)# Computing DFBETAS statistics
```


Influence Diagnostics for PC1
Threshold: 0.14
DFBETAS
Observation


Influence Diagnostics for PC5
Threshold: 0.14
DFBETAS
Observation


Influence Diagnostics for PC4
Threshold: 0.14
DFBETAS
Observation


Influence Diagnostics for PC10
Threshold: 0.14
DFBETAS
Observation