# R Codes for Chapter-10
# Model Diagnostics for Regression in R

After fitting a regression model, it is important to determine whether all the necessary model assumptions are valid before performing inference. If there are any violations, subsequent inferential procedures may be invalid resulting in faulty conclusions. Therefore, it is crucial to perform appropriate model diagnostics.

Model diagnostic procedures involve both graphical methods and formal statistical tests. These procedures allow us to explore whether the assumptions of the regression model are valid and decide whether we can trust subsequent inference results.

**Ex.** Problem 6.9, "Grocery Retailer" Y=Hours, $X_1$ = Cases, $X_2$ = Costs, and $X_3$ = Holiday.

To read in the data and fit a multiple linear regression model with Hours as the response variable and Cases and Costs as the explanatory variables use the commands:

```
> data=read.table("R:\\Teaching\\2016\\MA 542\\Class preperation\\Ex.6.9.csv"
,header = FALSE)
> X1=data[,2]
> X2=data[,3]
> Y=data[,1]

> Hoursd=data.frame(Y,X1,X2)

> fit<-lm(Y~X1+X2, data=Hoursd)
```

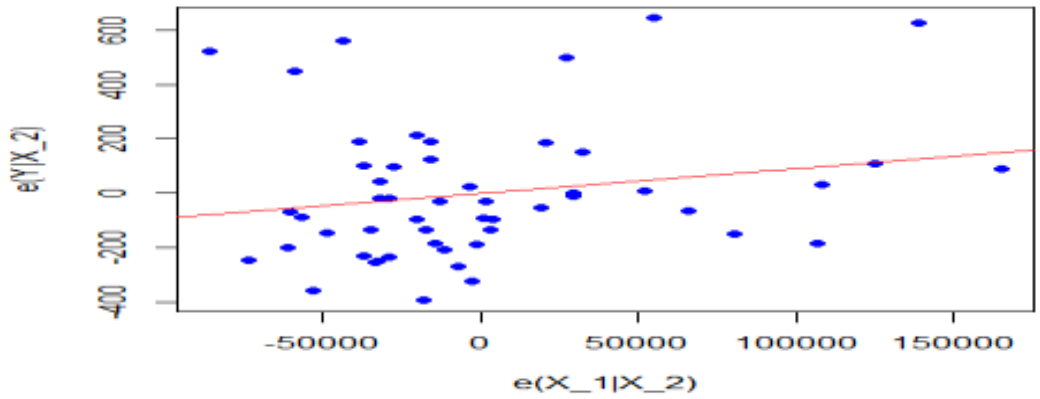
All information about the fitted regression model is now contained in fit.

## Added Variable Plots

Added Variable Plots can be drawn using one of the following commands.

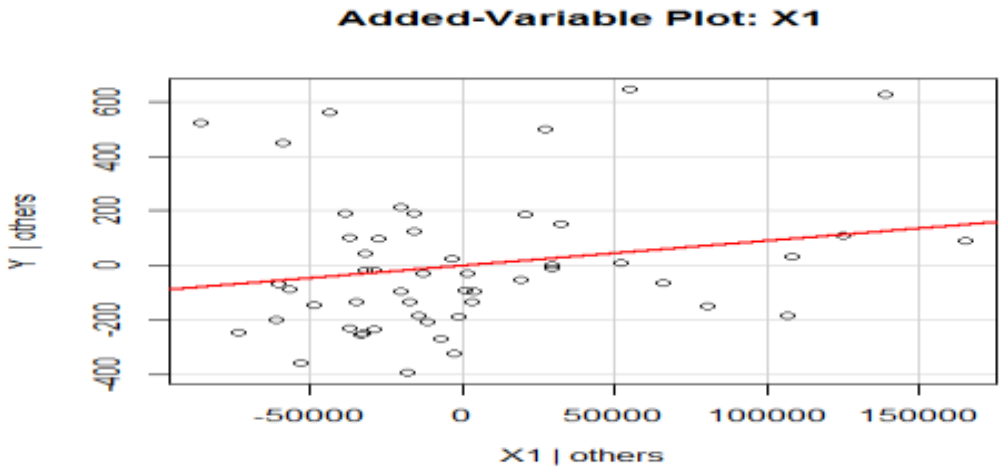
To draw added variable plot for X1: (first method)

```
> plot(resid(lm(Y~X2)) ~ resid(lm(X1~X2)),col="blue",pch=16,
+ xlab="e(X_1|X_2)", ylab="e(Y|X_2)")
> abline(lm(resid(lm(Y~X2))~resid(lm(X1~X2))),col="red")
```



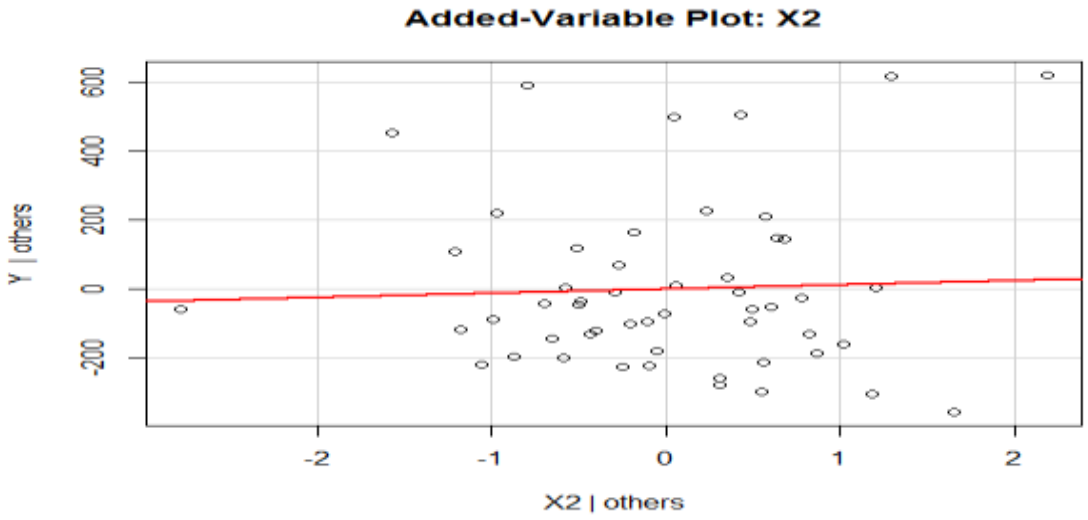
To draw added variable plot for X1: (second method)

```
> avPlot( model=fit, variable=X1 )
```




To draw added variable plot for X2:

```
> avPlot( model=fit, variable=X2 )
```




**Leverage values**

We can compute and plot the leverage of each point using the following commands:

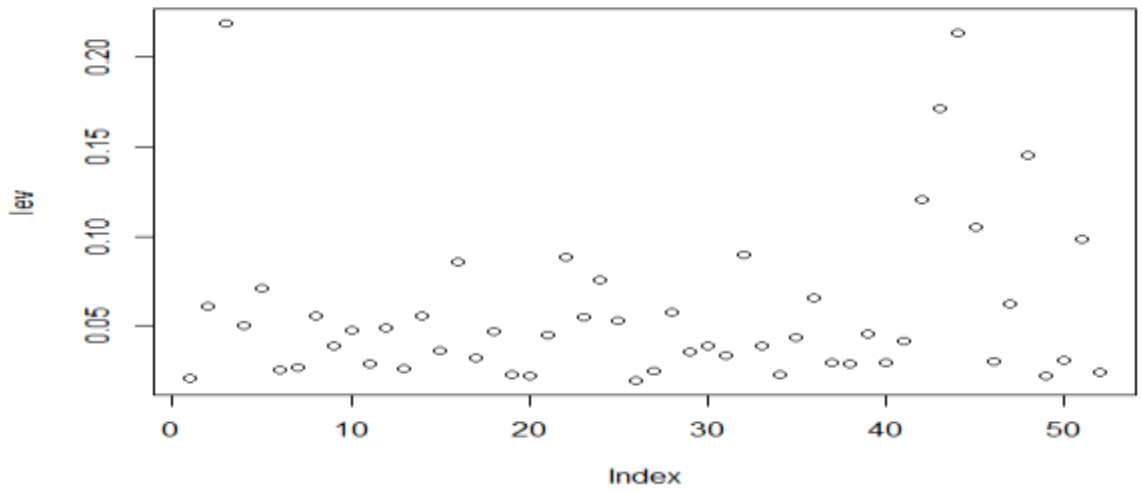
```
> lev = hat(model.matrix(fit))
> lev

[1] 0.02036085 0.06077066 0.21886918 0.05045737 0.07076303 0.02538487
 [7] 0.02660716 0.05558623 0.03899906 0.04734818 0.02838109 0.04888199
[13] 0.02593238 0.05530379 0.03587426 0.08531532 0.03181562 0.04649028
[19] 0.02242901 0.02187055 0.04475752 0.08843281 0.05510316 0.07530940
[25] 0.05275399 0.01954423 0.02465772 0.05755721 0.03553655 0.03887357
[31] 0.03340839 0.08984069 0.03849089 0.02273225 0.04326558 0.06560725
[37] 0.02962408 0.02856752 0.04519518 0.02958955 0.04148349 0.12037885
[43] 0.17120389 0.21348398 0.10528259 0.03008173 0.06187934 0.14524287
[49] 0.02169993 0.03056711 0.09806373 0.02434407
```

2nd method

```
> hatvalues(fit)
```

The output is same as the previous one.

```
plot(lev)
```




Note there are several points that with higher leverage than all the other points. To identify these points, type:

```
> Hoursd[lev>0.2,] # do not forget  the "," .

       Y     X1   X2
3   4317 317164 4.61
44  4469 472476 8.20
```

**Studentized residuals:**

Studentized residuals can be computed using the command:

```
> r = rstudent(fit)
> r
          1           2           3           4           5           6
-0.40076892  0.50920430 -0.11675549 -0.51554298  2.66338613 -0.12702850
          7           8           9          10          11          12
-0.89917566 -0.85341872 -0.75904759  0.95871406  0.29406943 -0.31120971
         13          14          15          16          17          18
-0.47623486 -1.32845616 -0.14484006  2.04831262  0.50599527 -0.70658129
         19          20          21          22          23          24
 0.11454910 -1.06994251  2.13939159  2.67966259 -0.73643140 -0.05289125
         25          26          27          28          29          30
-0.13973385 -0.37364519  0.04133542 -0.58428701  0.03926593 -0.40600028
         31          32          33          34          35          36
-0.89954206 -1.61421152  0.55088855  0.67020142 -1.26989804 -0.42616648
         37          38          39          40          41          42
-0.78942878  0.83221679 -0.80907949  0.91528179 -0.15709601 -0.02019478
         43          44          45          46          47          48
 2.79817285 -0.27059851 -0.27549432 -0.55220862 -0.92969128  2.26016357
         49          50          51          52
-0.29167031  0.56756048 -1.20518197 -0.06510943
```

**Influence of an observation:**

- **DFFITS**

```
> dffits(fit)
     1           2           3           4           5
-0.057777524  0.129524884 -0.061802739 -0.118842005  0.734977625
     6           7           8           9          10
-0.020500860 -0.148661910 -0.207044931 -0.152909391  0.213734018
```

- **Cook's distance**

```
> cook = cooks.distance(fit)
> cook

           1           2           3           4           5
1.132141e-03 5.678065e-03 1.299349e-03 4.779422e-03 1.601481e-01
           6           7           8           9          10
1.429657e-04 7.395689e-03 1.436887e-02 7.861764e-03 1.525258e-02
```

- **DFBETAS**

```
> dfbetas(fit)
    (Intercept)            X1            X2
1  -0.014885843 -0.0041556877  0.0132679366
2   0.071236669  0.0428063357 -0.1014395801
3  -0.047358967 -0.0098194810  0.0588261954
4   0.010557866 -0.0886809297  0.0370236599
5  -0.282096200 -0.3077127418  0.5706858496
6  -0.010219267 -0.0006299925  0.0100915191
7  -0.067734769  0.0755569391  0.0139592630
```

To get a summary of all the above influential measures use:

```
> influence.measures(fit)

Influence measures of
         lm(formula = Y ~ X1 + X2, data = Hoursd) :

     dfb.1_    dfb.X1    dfb.X2    dffit cov.r   cook.d    hat inf
1  -0.01489 -0.004156  0.013268 -0.05778 1.075 1.13e-03 0.0204
2   0.07124  0.042806 -0.101440  0.12952 1.114 5.68e-03 0.0608
3  -0.04736 -0.009819  0.058826 -0.06180 1.361 1.30e-03 0.2189   *
4   0.01056 -0.088681  0.037024 -0.11884 1.102 4.78e-03 0.0505
5  -0.28210 -0.307713  0.570686  0.73498 0.757 1.60e-01 0.0708   *
6  -0.01022 -0.000630  0.010092 -0.02050 1.090 1.43e-04 0.0254
7  -0.06773  0.075557  0.013959 -0.14866 1.039 7.40e-03 0.0266
```

**Multicollinearity**

- **variance inflation factors**

```
vif(fit)
   X1      X2
1.00726 1.00726
```