R codes Chapter -11
Remedial measures in R

When the fitted regression model is not appropriate or one or several cases are very influential, remedial measures may be used to fix the model.

Weighted Least Square Method

E.x. Machine Speed example (7:11): Y-Productivity, X-Machine speed setting

To read data:

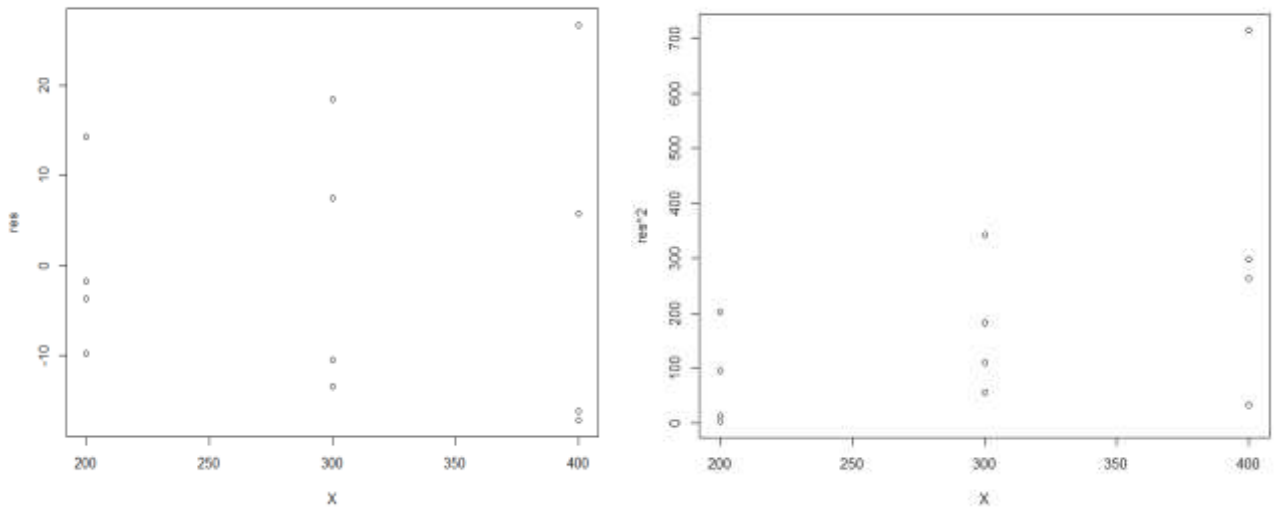
```
> data=read.table("R:\\Teaching\\2016\\MA 542\\Class preperation\\Ex.11.7.csv
",header = FALSE)
> X=data[,2]
> Y=data[,1]
```

To fit a linear regression model and extract residuals:

```
> linfit=lm(Y~X)
res=residuals(linfit)
```

The following are the plot of residuals vs X and squared residuals vs X

```
> plot(X, res)
> plot(X, res^2)
```



To estimate the variance function regress squared residuals on X.

```
> varf=lm(res^2~X)
```

So estimated weights are given by

```
> wi=1/fitted.values(varf)
```

```
> wi
          1           2           3           4           5           6
7
0.014563107 0.003150433 0.005180229 0.003150433 0.014563107 0.005180229 0.005
180229
          8           9          10          11          12
0.003150433 0.014563107 0.003150433 0.014563107 0.005180229
```

To fit the weighted least square regression model use "**lm**" with weights as follows

```
> newdata=data.frame(Y,X,wi)
> wlsq=lm(Y~X,weights=wi)
```

Summary of the Least Square fit:

```
> summary(linfit)

Call:
lm(formula = Y ~ X)

Residuals:
    Min      1Q  Median      3Q     Max
-17.250 -11.250  -2.750   9.188  26.750

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.75000   16.73052  -0.344  0.73820
X            0.18750    0.05381   3.484  0.00588 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.22 on 10 degrees of freedom
Multiple R-squared:  0.5484,   Adjusted R-squared:  0.5032
F-statistic: 12.14 on 1 and 10 DF,  p-value: 0.005878
```

Summary of the Weighted Least Square fit:

```
> summary(wlsq)

Call:
lm(formula = Y ~ X, weights = wi)

Weighted Residuals:
    Min      1Q  Median      3Q     Max
-1.1572 -0.9338 -0.3124  0.7377  1.7391

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.23322   13.16843  -0.473  0.64613
X            0.18911    0.05056   3.740  0.00385 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.109 on 10 degrees of freedom
Multiple R-squared:  0.5831,   Adjusted R-squared:  0.5414
F-statistic: 13.99 on 1 and 10 DF,  p-value: 0.003846
```

Ridge Regression:

E.x. Patient Satisfaction Example (11.22 or 6.15): Y- Patient Satisfaction, X1-age, X2-severity of illness, X3-anxiety level.

To read data:

```
> data=read.table("R:\\Teaching\\2016\\MA 542\\Class preperation\\Ex.6.15.csv
",header = FALSE)
> X1=data[,2]
> X2=data[,3]
> X3=data[,4]
> Y=data[,1]
```

To fit ridge regression model follow the steps:

The following packages need to be installed and uploaded.

```
> library(MASS)
> library(genridge)
```

To fit OLS model:

```
> lmod=lm(Y~X1+X2+X3,data=data2)
```

Define a new data frame.

```
> X=data.matrix(data2[,c(2:4,1)])
```

Define a sequence for values of c.

```
> cvals= seq(0,500,by= 5)
```

To fit ridge regression model:

```
>  lridge <- ridge(y, X, lambda=cvals)

>  coef(lridge)
                X1             X2             X3           Y
  0  0.000000000  4.857226e-17 -4.163336e-17 0.14744196
  5 -0.017848516 -4.719311e-03 -7.907858e-03 0.11315705
 10 -0.024654181 -7.954232e-03 -1.187392e-02 0.09495335
 15 -0.027616232 -1.017464e-02 -1.417804e-02 0.08328336
 20 -0.028899576 -1.171889e-02 -1.560244e-02 0.07498143
 25 -0.029349592 -1.280162e-02 -1.650169e-02 0.06867316
```

This is only a part of the output.

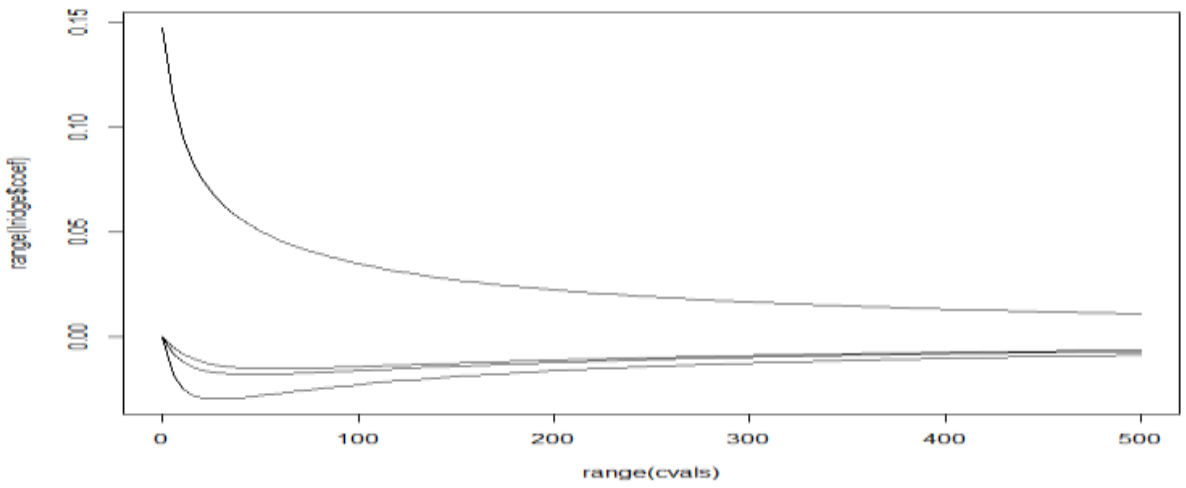
To calculate VIF for each model:

```
> vridge <- vif(lridge)
>  vridge
          X1       X2       X3        Y
  0 2.730096 2.041736 2.181254 3.146577
  5 1.717338 1.559815 1.608050 1.845630
 10 1.354710 1.323612 1.342121 1.397854
 15 1.188397 1.192571 1.199135 1.198456
 20 1.103891 1.114835 1.116295 1.099567
 25 1.060202 1.067265 1.066735 1.049739
```

This is only a part of the output.

To draw the trace plot:

```
> plot(range(cvals), range(lridge$coef),type="n")
> for(i in 1:ncol(lridge$coef)){ lines(cvals,lridge$coef[,i])}
```

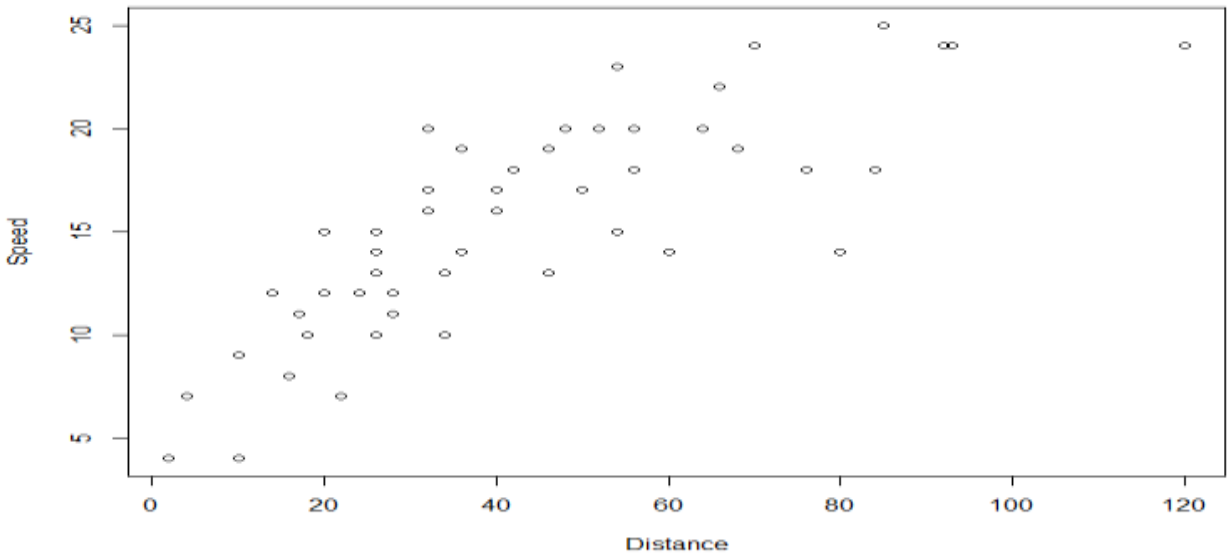


Nonparametric Regression

Lowess Method:

E.x. Cars example: (this is in build in R), Y-speed, X-distance.

```
> data(cars) # this is in build in R, Y-speed, X-distance.
> Speed=cars$speed
> Distance=cars$dist

> plot(Distance,Speed)
```



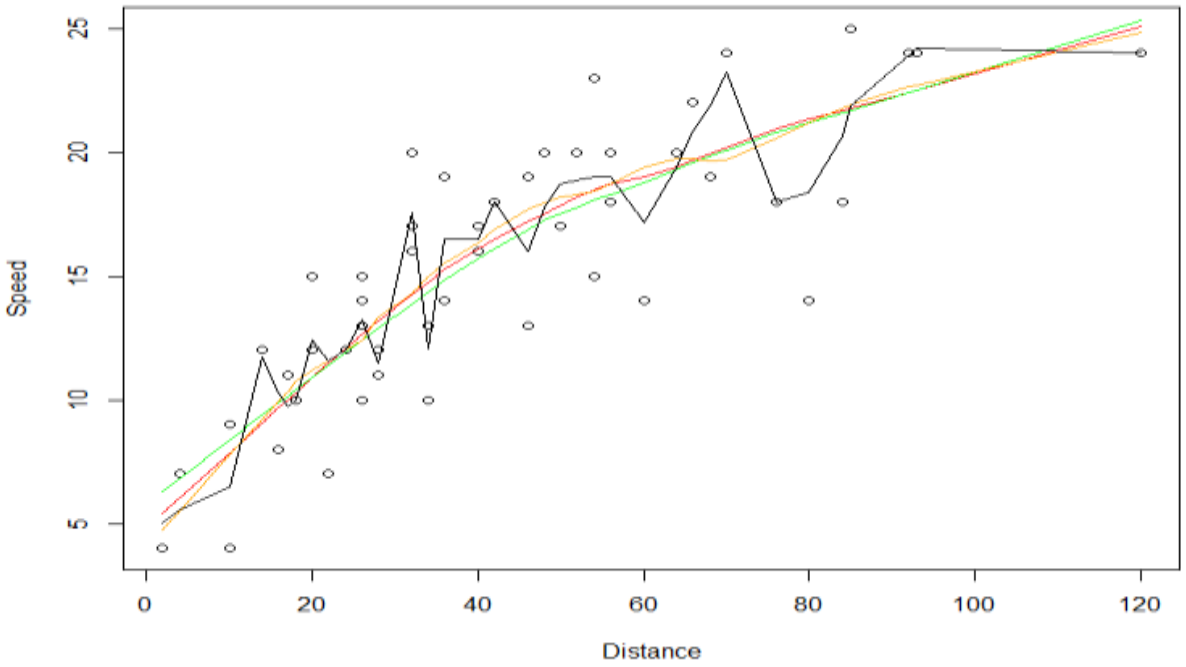
Fitting the lowess curve:

```
> lowess(Speed~Distance)
$x
 [1]   2   4  10  10  14  16  17  18  20  20  22  24  26  26  26  26  28  28
32  32  32  34
[23]  34  34  36  36  40  40  42  46  46  48  50  52  54  54  56  56  60  64
66  68  70  76
[45]  80  84  85  92  93 120

$y
 [1]  5.419833  6.034063  7.861475  7.861475  9.069588  9.673301  9.975386 10
.277712 10.883150
[10] 10.883150 11.488354 12.085713 12.660596 12.660596 12.660596 12.660596 13
.201558 13.201558
[19] 14.237556 14.237556 14.237556 14.737713 14.737713 14.737713 15.251018 15
.251018 16.103033
[28] 16.103033 16.481524 17.210519 17.210519 17.508543 17.853728 18.173957 18
.483517 18.483517
[37] 18.704807 18.704807 19.028842 19.416744 19.664883 19.923202 20.182120 20
.935868 21.346516
[46] 21.696448 21.781930 22.396404 22.487343 25.050362
```

Here ys are the fitted value for each value of X. This is the optimal fit

We also can change the value of q (here it is f in R). To see the effect of "f" value and identify the optimal value, we can draw several plots as follows.

```
> plot(Distance,Speed)
> lines(lowess(Speed~Distance,f=2/3),col="red")
> lines(lowess(Speed~Distance,f=.9),col="green")
> lines(lowess(Speed~Distance,f=1/3),col="orange")
> lines(lowess(Speed~Distance,f=0.1),col="black")
```



Here note that the model is over fit for small c-values and model is under fit for large c-values.

Regression Trees

E.x. Patient Satisfaction Example (11.22 or 6.15):

The following packages need to be installed and uploaded.

```
> library(rpart)
> library(MASS)
```

To divide data into sub regions (grow tree)

```
> fit <- rpart(Y~X1+X2+X3, method="anova", data=data2)
```

To display the results

```
> print(fit)
n= 46

node), split, n, deviance, yval
      * denotes terminal node

1) root 46 13369.300 61.56522
  2) X1>=36.5 24  3513.833 50.08333
    4) X1>=46 10   713.600 40.80000 *
    5) X1< 46 14  1322.857 56.71429 *
  3) X1< 36.5 22  3239.818 74.09091
    6) X1>=29.5 12  1254.000 67.00000 *
    7) X1< 29.5 10   658.400 82.60000 *
```

Summary:

```
> summary(fit)
```

Bootstrapping

E.x. Cars example.

The following package need to be installed and uploaded.

```
> library(boot)

> speed=data.frame(Y,X)
```

Here suppose we want to estimate the precision of **b1** when **weighted lease square model is fit.** To calculate and return value of b1, a function like the following need to be written.

```
> b1=function(data,indices){
+   d = data[indices, ]
+   lfit=lm(Y~X,data=d)
+   res=residuals(lfit)
+   sdf=lm(abs(res)~X)
+   wi=1/fitted.values(sdf)
+   wls=lm(Y~X,weights=wi)
+   b_1=coef(wls)[2]
+ return(b_1)
+ }
```

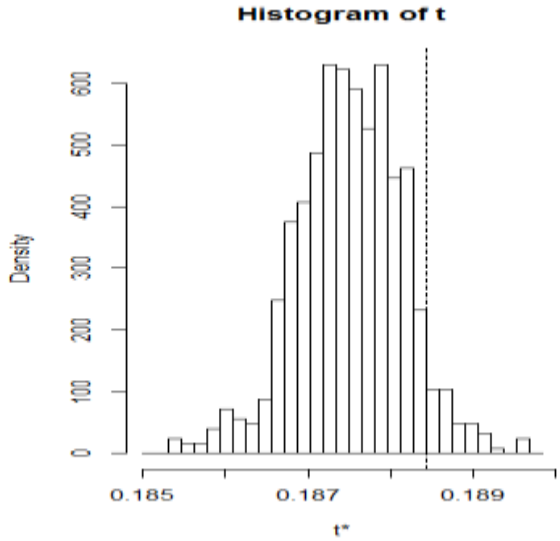
```
b1=function(data,indices){
  d = data[indices, ]
  lfit=lm(Y~X,data=d)
  res=residuals(lfit)
  sdf=lm(abs(res)~X)
  wi=1/fitted.values(sdf)
  wls=lm(Y~X,weights=wi)
  b_1=coef(wls)[2]
return(b_1)
}
```

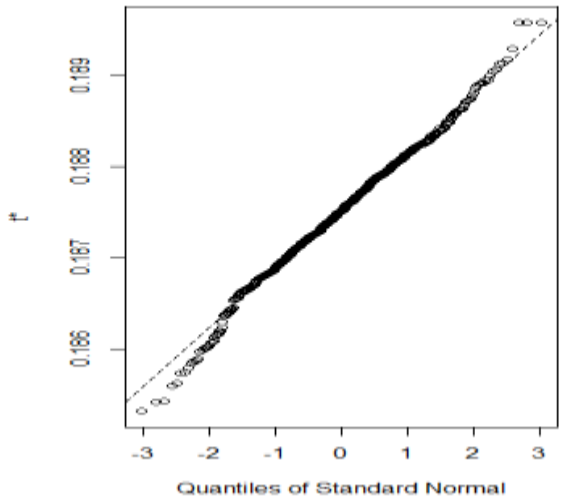
To get the bootstrap estimates

```
> results = boot(data =speed, statistic = b1, R=800)
> print(results)
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = speed, statistic = b1, R = 800)


Bootstrap Statistics :
     original        bias     std. error
t1* 0.1884407 -0.0009261067 0.0006441258
```

We can see the distribution of the b1* drawing following plots.

```
> plot(results)
```





Bootstrap confidence intervals

```
boot.ci(results,conf=0.95, type="all")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 800 bootstrap replicates

CALL :
boot.ci(boot.out = results, conf = 0.95, type = "all")

Intervals :
Level      Normal              Basic
95%   ( 0.1881,  0.1906 )   ( 0.1881,  0.1908 )

Level     Percentile            BCa
95%   ( 0.1861,  0.1888 )   ( 0.1882,  0.1896 )
```

Here output shows confidence intervals based on several criteria. The one for percentile is the one we learnt in the class.

We also can use:

```
> boot.ci(results,conf=0.95,type="perc")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 800 bootstrap replicates

CALL :
boot.ci(boot.out = results, conf = 0.95, type = "perc")

Intervals :
Level     Percentile
95%   ( 0.1861,  0.1888 )
```

Compare with the confidence interval for the Least Square Fit and the WLS fit.

```
> confint(linfit)
                   2.5 %     97.5 %
(Intercept) -43.02792943 31.5279294
X             0.06760159  0.3073984


> confint(wlsq)
                   2.5 %     97.5 %
(Intercept) -35.57431132 23.1078749
X             0.07644931  0.3017721
```