PRACTICAL 6: Practical of Simple/Multiple Linear Regression

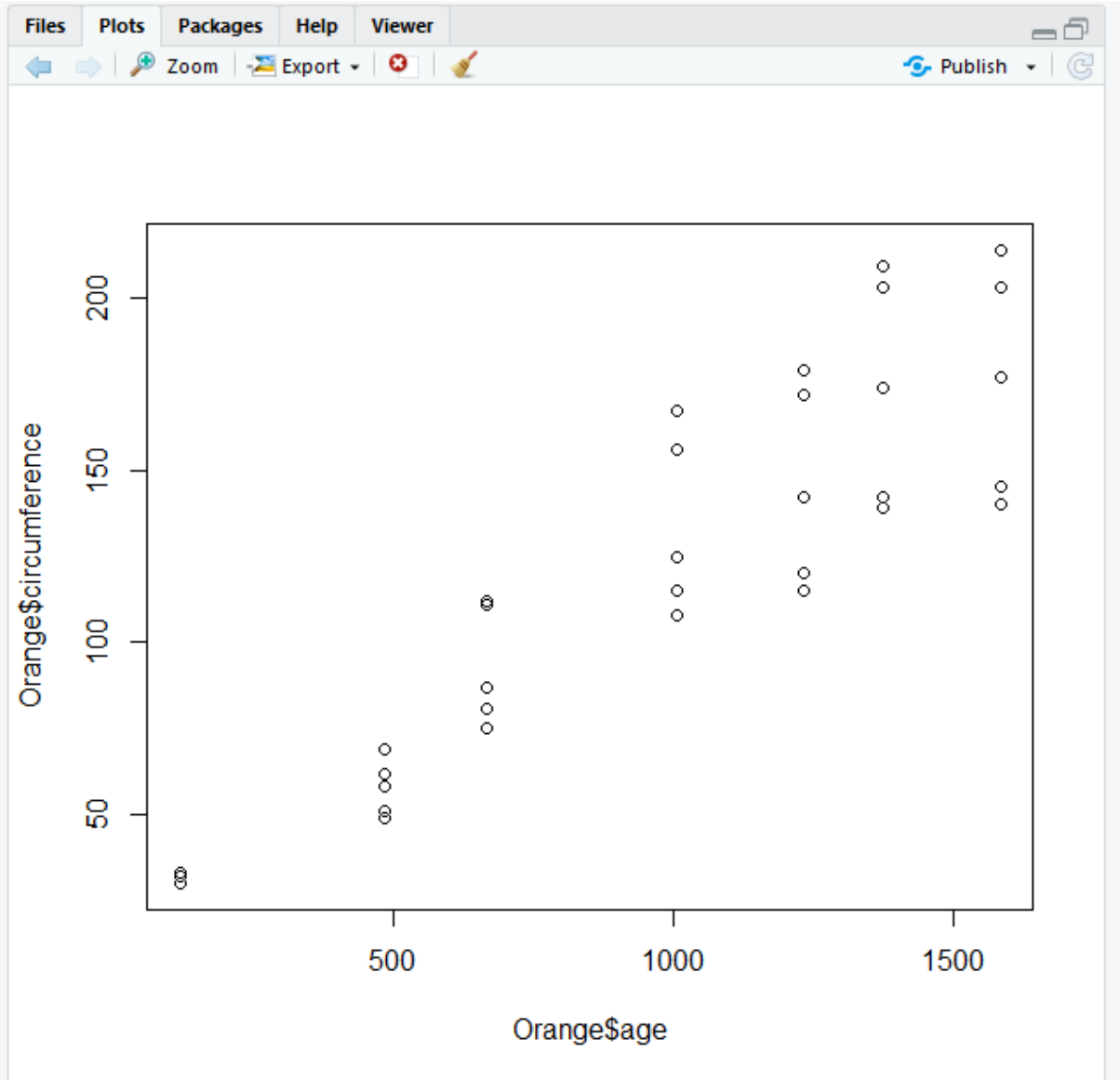
Aim: Practical of Simple and Multiple Regression

SIMPLE LINEAR REGRESSION

We make use of inbuilt dataset Orange and plot Orange$age and Orange$circumference.

```
> library(datasets)
> library(help='datasets')
> View(Orange)
> plot(Orange$age,Orange$circumference)
```

Output:



We fit the model using lm command
We are using linear regression to find out correlation/linear relationship between the two variables.

```
> ?lm
> lm_result<-lm(Orange$circumference~Orange$age)
> lm_result

Call:
lm(formula = Orange$circumference ~ Orange$age)

Coefficients:
(Intercept)   Orange$age
    17.3997       0.1068
```

```
> summary(lm_result)

Call:
lm(formula = Orange$circumference ~ Orange$age)

Residuals:
    Min      1Q  Median      3Q     Max
-46.310 -14.946  -0.076  19.697  45.111

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.399650   8.622660   2.018   0.0518 .
Orange$age   0.106770   0.008277  12.900 1.93e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 23.74 on 33 degrees of freedom
Multiple R-squared:  0.8345,    Adjusted R-squared:  0.8295
F-statistic: 166.4 on 1 and 33 DF,  p-value: 1.931e-14
```

```
> attributes(lm_result)
$`names`
 [1] "coefficients"  "residuals"     "effects"       "rank"          "fitted.values"
 [6] "assign"        "qr"            "df.residual"   "xlevels"       "call"
[11] "terms"         "model"

$class
[1] "lm"
```

Analysis:

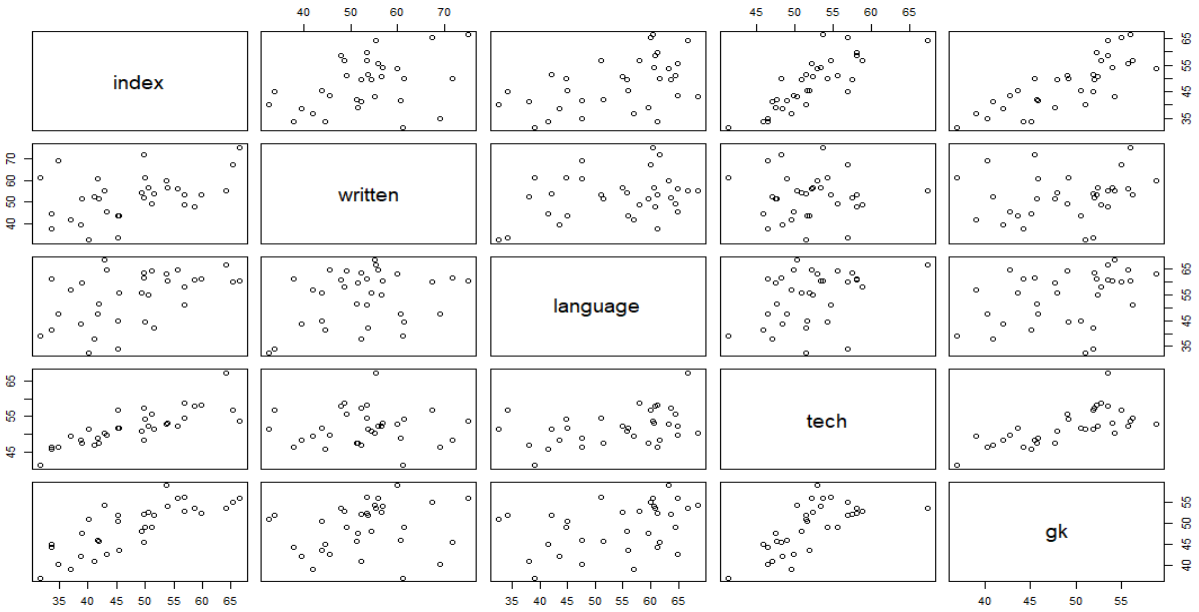
1. The adjusted R-squared value is 0.8295. This means the model explains 82.95% of the variance (Informally, it measures how far a set of (random) numbers are spread out from their average value).
2. The p-value is 1.93e-14 for Orange$age which is very low.
3. As the p-value is much less than 0.05, we reject the null hypothesis that β = 0. Hence there is a significant relationship between the variables in the linear regression model of the data set Orange.

MULTIPLE LINEAR REGRESSION

Step 1:- Import file index.csv
Scatter plot

```
[Workspace loaded from ~/.RData]

> index<-read.csv(file.choose(),sep = ",",header = T)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> pairs(~index+written+language+tech+gk,data=index)
>
```



Step 2:- Fit a model (elements are stored)

```
> model1<-lm(index~.,data=index)
> model1

Call:
lm(formula = index ~ ., data = index)

Coefficients:
(Intercept)        empid      written     language         tech           gk
  -56.37329     -0.12830      0.33206      0.04794      1.17174      0.51787

> summary(model1)

Call:
lm(formula = index ~ ., data = index)

Residuals:
    Min      1Q  Median      3Q     Max
-5.5382 -2.4528  0.0266  2.2774  5.4622

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -56.37329    7.12537  -7.912 1.67e-08 ***
empid        -0.12830    0.06542  -1.961  0.06025 .
written       0.33206    0.06472   5.131 2.14e-05 ***
language      0.04794    0.06828   0.702  0.48859
tech          1.17174    0.17714   6.615 4.26e-07 ***
gk            0.51787    0.15123   3.424  0.00198 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.382 on 27 degrees of freedom
Multiple R-squared:  0.8922,    Adjusted R-squared:  0.8722
F-statistic: 44.67 on 5 and 27 DF,  p-value: 3.188e-12
```

Index= -56.3724+(-0.12830)+written(0.33206)+language(0.04794)+tech(1.17174)+gk(0.57787)

Conclusion:-

As value of multiple r(square) is 0.8922

So 89% of variation in index is explained by the model and 11% is not explained by the model

Step 3 :- Write down the equation and check for global testing

- **Fitted Value:** A **fitted value** is simply another name for a predicted **value** as it describes where a particular x-**value** fits the line of best fit given by the fitted function.
- **Residual value:** In regression analysis, the difference between the observed value of the dependent variable (y) and the predicted value (ŷ) is called the residual (e) given by the residuals function.

```
> index$pred<-fitted(model1)
> head(index)
  empid index written language  tech    gk     pred
1     1 45.52   43.83    55.92 51.82 43.58 44.02220
2     2 40.10   32.71    32.56 51.49 51.03 42.55279
3     3 50.61   56.64    54.84 52.29 52.47 53.12213
4     4 38.97   51.53    59.69 47.48 47.69 43.41803
5     5 41.87   51.35    51.50 47.59 45.77 41.97188
6     6 38.71   39.60    43.63 48.34 42.06 36.52202
> index$res<-residuals(model1)
> head(index)
  empid index written language  tech    gk     pred        res
1     1 45.52   43.83    55.92 51.82 43.58 44.02220  1.4978042
2     2 40.10   32.71    32.56 51.49 51.03 42.55279 -2.4527900
3     3 50.61   56.64    54.84 52.29 52.47 53.12213 -2.5121304
4     4 38.97   51.53    59.69 47.48 47.69 43.41803 -4.4480298
5     5 41.87   51.35    51.50 47.59 45.77 41.97188 -0.1018831
6     6 38.71   39.60    43.63 48.34 42.06 36.52202  2.1879775
>
```

Step 4:- To check the multicollinearity.

- In statistics, **multicollinearity** (also collinearity) is a phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.
- If the VIF is equal to 1 there is no multicollinearity among factors, but if the VIF is greater than 1, the predictors may be moderately correlated. The output above shows that the VIF for the Publication and Years factors are about 1.5, which indicates some correlation, but not enough to be overly concerned about. A VIF between 5 and 10 indicates high correlation that may be problematic. And if the VIF goes above 10, you can assume that the regression coefficients are poorly estimated due to multicollinearity.

```
> library(car)
> vif(model1)
    empid   written language      tech        gk
1.119815 1.185225 1.344122 2.178955 2.033284
>
```

Conclusion:

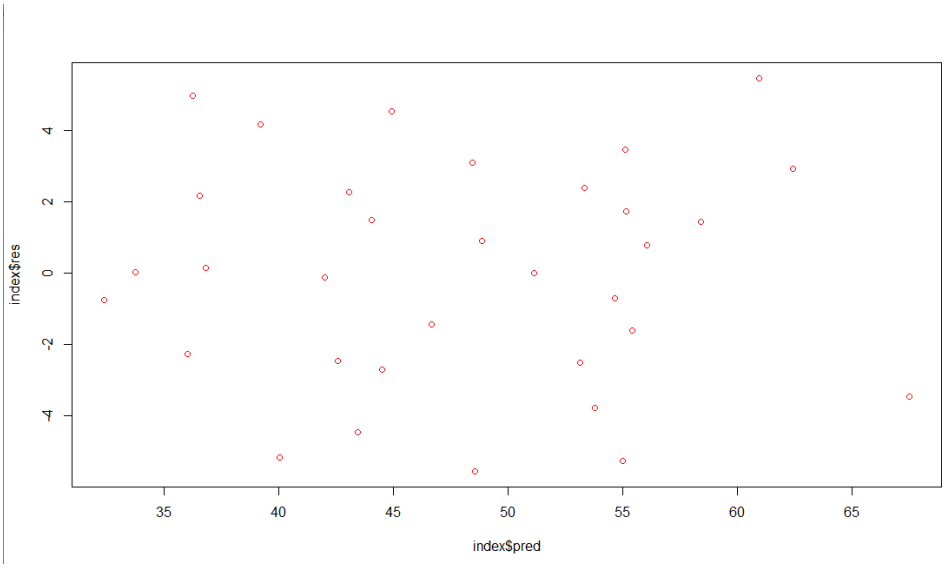
As all VIF are less than 5

Multicollinearity is not present.

Step 5:- to check heteroscedasticity

- *Heteroscedasticity* means unequal scatter.
- **Heteroscedasticity** (the violation of **homoscedasticity**) is present when the size of the error term differs across values of an independent variable

```
> plot(index$pred,index$res,col="red")
```



Conclusion:-

Since errors are generated randomly. There is no heteroscedasticity.

Step 6 :- Perform Shapiro test to check normality of errors.

- **shapiro.test** tests the *Null hypothesis* that "the samples come from a Normal distribution" **against** the *alternative hypothesis* "the samples do not come from a Normal distribution".

```
> shapiro.test(index$res)

        Shapiro-Wilk normality test

data:  index$res
W = 0.97172, p-value = 0.5293
```

Conclusion:-

H0: the samples come from a Normal distribution

H1: the samples do not come from a Normal distribution

As p value is greater than 0.05 accept Ho.

Step 7 :- Detecting heteroscedasticity of data using NCV test.

- homoscedasticity means that the variance around the **regression** line is the same for all values of the predictor variable.

```
> library(car)
> ncvTest(model1,~written+language+tech+gk)
Non-constant Variance Score Test
Variance formula: ~ written + language + tech + gk
Chisquare = 2.146914, Df = 4, p = 0.70876
```

Conclusion:-

Ho: There is homoscedasticity.
H1: there is no constant variance (there is heteroscedasticity).
As P-value is greater than 0.05 we accept Ho.

Step 8 :- Detecting autocorrelation using Durbin Watson Test d=2(1-r)

- **Autocorrelation** is a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals. It is the same as calculating the correlation between two different time series, except that the same time series is actually used twice: once in its original form and once lagged one or more time periods.
- The Durbin Watson (DW) statistic is a test for autocorrelation in the residuals from a statistical regression analysis.

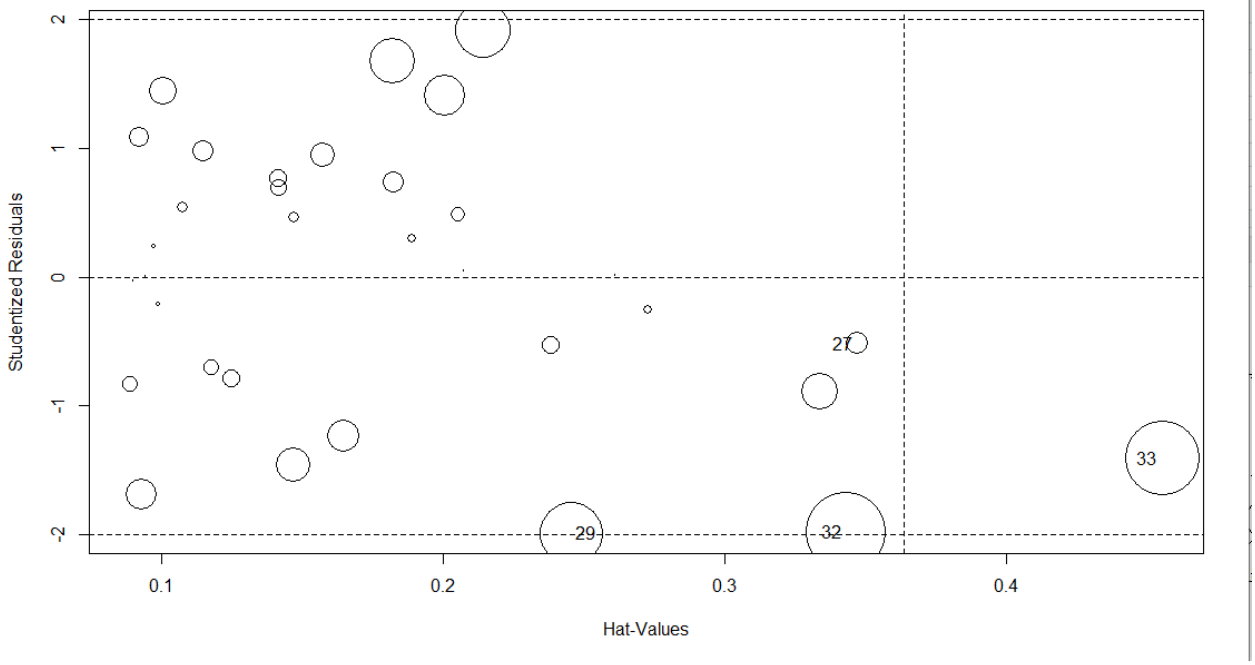
```
> library(car)
> durbinWatsonTest(model1)
 lag Autocorrelation D-W Statistic p-value
   1       0.2268414      1.500571   0.084
 Alternative hypothesis: rho != 0
>
```

Conclusion:-

Ho : Auto correlation is not present among errors
H1 : not Ho
As P-value is greater than 0.05 we accept Ho.

Step 9 :- influence plot

```
> influencePlot(model1)
      StudRes       Hat      CookD
27 -0.5173889 0.3473188 0.02440349
29 -1.9854091 0.2453822 0.19264149
32 -1.9789285 0.3433424 0.30800324
33 -1.4060295 0.4554469 0.26594943
```



Step 10 :-

```
> library("lattice")
> library("ggplot2")
> library("caret")
> index<-read.csv(file.choose(),sep = ",",header=T)
> summary(index)
     empid          index          written         language          tech
 Min.   : 1   Min.   :31.64   Min.   :32.71   Min.   :32.56   Min.   :41.25
 1st Qu.: 9   1st Qu.:41.19   1st Qu.:45.59   1st Qu.:44.89   1st Qu.:48.34
 Median :17   Median :49.45   Median :53.38   Median :57.04   Median :51.64
 Mean   :17   Mean   :47.87   Mean   :52.66   Mean   :53.99   Mean   :52.02
 3rd Qu.:25   3rd Qu.:53.92   3rd Qu.:56.75   3rd Qu.:61.28   3rd Qu.:54.68
 Max.   :33   Max.   :66.39   Max.   :75.03   Max.   :68.53   Max.   :67.27
       gk
 Min.   :37.00
 1st Qu.:45.07
 Median :50.53
 Mean   :49.04
 3rd Qu.:53.50
 Max.   :58.90
> data<-createDataPartition(index$empid,p=0.8,list = F)
> head(data)
     Resample1
[1,]         1
[2,]         2
[3,]         4
[4,]         5
[5,]         6
[6,]         7
> dim(data)
[1] 29  1
>
```

Step 11 :-

Partitioned the data into training data and testing data.

```
> traindata<-index[data,]
> testdata<-index[-data,]
> dim(traindata)
[1] 29  6
> dim(testdata)
[1] 4 6
>
```

Step 12 :- Validation using k fold method

- cross validation predicts how well the model will perform in testing data set on the basis of training dataset.
- The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.
- The general procedure is as follows:
  - Shuffle the dataset randomly.
  - Split the dataset into k groups
  - For each unique group:
    - Take the group as a hold out or test data set
    - Take the remaining groups as a training data set
    - Fit a model on the training set and evaluate it on the test set
    - Retain the evaluation score and discard the model
  - Summarize the skill of the model using the sample of model evaluation scores
- the trainControl function takes parameter method as cv which means cross validation and number as 4 which means 4-fold cross validation will be performed. trainControl controls the computational nuances of the train function
- **RMSE:** Root Mean Squared Error
- The root-mean-squared error (RMSE) is a measure of how well your model performed. It does this by measuring ***difference between predicted values and the actual values.***

```
> kfolds<-trainControl(method = "cv" , number = 4)
> modelkfold<-train(index~written+language+tech+gk,data = index,method="lm",trControl=kfolds)
> modelkfold
Linear Regression

33 samples
 4 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 24, 25, 25, 25
Resampling results:

  RMSE      Rsquared   MAE
  4.237887  0.8787614  3.595394

Tuning parameter 'intercept' was held constant at a value of TRUE
>
```

Conclusion :-

As the value of RMSE is sufficiently large the model is stable.

Step 13 :- Model selection forward method.

```
> index<-read.csv(file.choose(),sep = ",",header = T)
> null<-lm(index~1,data = index)
> full<-lm(index~.,data = index)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> step(null,scope = list(lower=null,upper=full),direction = "forward")
Start:  AIC=149.28
index ~ 1

           Df Sum of Sq     RSS    AIC
+ tech      1   1867.81  994.92 116.40
+ gk        1   1787.03 1075.69 118.98
+ language  1    660.54 2202.19 142.62
+ written   1    479.64 2383.09 145.23
<none>                  2862.73 149.28
+ empid     1     62.42 2800.31 150.55

Step:  AIC=116.4
index ~ tech

           Df Sum of Sq    RSS     AIC
+ written   1    490.24 504.68  96.005
+ gk        1    302.78 692.14 106.428
+ language  1     99.24 895.68 114.936
<none>                  994.92 116.403
+ empid     1     24.53 970.39 117.579

Step:  AIC=96
index ~ tech + written

           Df Sum of Sq    RSS    AIC
+ gk        1   149.196 355.48 86.440
+ empid     1    49.957 454.72 94.565
<none>                  504.68 96.005
+ language  1     7.276 497.40 97.526

Step:  AIC=86.44
index ~ tech + written + gk

           Df Sum of Sq    RSS    AIC
+ empid     1    41.105 314.38 84.385
<none>                  355.48 86.440
+ language  1     2.764 352.72 88.183

Step:  AIC=84.39
index ~ tech + written + gk + empid

           Df Sum of Sq    RSS    AIC
<none>                  314.38 84.385
+ language  1    5.6376 308.74 85.788

Call:
lm(formula = index ~ tech + written + gk + empid, data = index)

Coefficients:
(Intercept)         tech      written           gk        empid
   -56.4681       1.1988       0.3456       0.5276      -0.1233
```

Conclusion:

The model selects variables correctly in the order of importance: tech, written then gk then empid. This shows tech score plays highest role in index interpretation.

Step 14:- Model selection backword method.

```
> index<-read.csv(file.choose(),sep = ",",header = T)
> null<-lm(index~1,data = index)
> full<-lm(index~.,data = index)
> names(index)
[1] "empid"    "index"    "written"  "language" "tech"     "gk"
> step(full,scope = list(lower=null,upper=full),direction = "backward")
Start:  AIC=85.79
index ~ empid + written + language + tech + gk

           Df Sum of Sq    RSS     AIC
- language  1      5.64 314.38  84.385
<none>                308.74  85.788
- empid     1     43.98 352.72  88.183
- gk        1    134.09 442.83  95.691
- written   1    300.99 609.74 106.245
- tech      1    500.35 809.10 115.581

Step:  AIC=84.39
index ~ empid + written + tech + gk

          Df Sum of Sq    RSS     AIC
<none>                314.38  84.385
- empid    1     41.11 355.48  86.440
- gk       1    140.34 454.72  94.565
- written  1    357.94 672.32 107.469
- tech     1    549.77 864.15 115.753

Call:
lm(formula = index ~ empid + written + tech + gk, data = index)

Coefficients:
(Intercept)        empid      written         tech           gk
   -56.4681      -0.1233       0.3456       1.1988       0.5276

> |
```