# Project - Part1

Anusha Dasari

11/26/2021

*Loading the required libraries:*

```
library(readxl)
library(olsrr)
```

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

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

```
library(GGally)
```

```
## Loading required package: ggplot2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggplot2)
```

*Loading LINTHALL data for analysis:*

```
data=read_excel("/Users/anusha_dasari/Downloads/LINTHALL.xlsx")
data=data[-1:-3]
data
```

```
## # A tibble: 45 x 15
##      BIO   H2S   SAL   Eh7    pH   BUF     P     K    Ca    Mg     Na    Mn
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>
##  1   676  -610    33  -290  5     2.34  20.2 1442. 2150  5169. 35184. 14.3
##  2   516  -570    35  -268  4.75  2.66  15.6 1299. 1845. 4358. 28170.  7.73
##  3  1052  -610    32  -282  4.2   4.18  18.7 1154. 1750. 4041. 26455  17.8
##  4   868  -560    30  -232  4.4   3.6   22.8 1045. 1674. 3966. 25073. 49.2
##  5  1008  -610    33  -318  5.55  1.9   37.8  522. 3360. 4609. 31664. 30.5
##  6   436  -620    33  -308  5.05  3.22  27.4 1273. 1811. 4390. 25492.  9.76
##  7   544  -590    36  -264  4.25  4.5   21.3 1346. 1907. 4579. 20877. 25.7
##  8   680  -610    30  -340  4.45  3.5   16.5 1254. 1860. 3983. 25621. 10.0
##  9   640  -580    38  -252  4.75  2.62  18.2 1243. 1799. 4142. 27587.  9.01
## 10   492  -610    30  -288  4.6   3.04  19.3 1282. 1797. 4264. 26512. 12.7
## # ... with 35 more rows, and 3 more variables: Zn <dbl>, Cu <dbl>, NH4 <dbl>
```

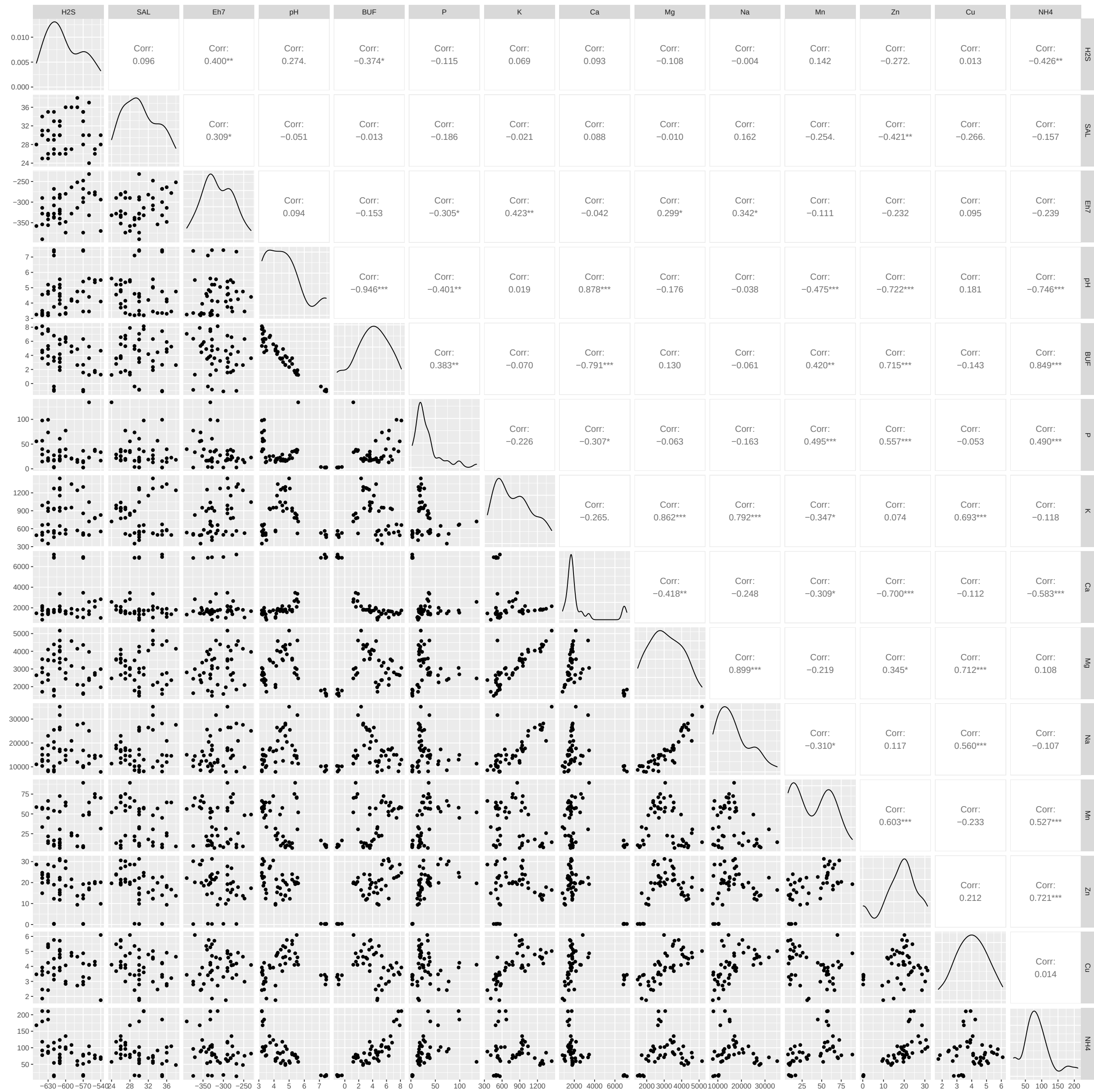
*Performing Ordinary Least Square Regression and shown below are the estimated regression coefficients:*

```
model_ols<-lm(BIO~H2S+SAL+Eh7+pH+BUF+P+K+Ca+Mg+Na+Mn+Zn+Cu+NH4, data=data)
coef(model_ols)
```

```
##   (Intercept)           H2S           SAL           Eh7            pH
##  2.909934e+03  4.289992e-01 -2.398072e+01  2.553224e+00  2.425278e+02
##           BUF             P             K            Ca            Mg
## -6.902268e+00 -1.701511e+00 -1.046591e+00 -1.160706e-01 -2.802284e-01
##            Na            Mn            Zn            Cu           NH4
##  4.451049e-03 -1.678760e+00 -1.879452e+01  3.451628e+02 -2.705172e+00
```

*Performing collinearity check:*

```
ggpairs(data[-1])
```



```
corMat=cor(data[-1])
corMat
```

```
##              H2S         SAL         Eh7          pH         BUF           P
## H2S  1.000000000  0.09580885  0.39965489  0.27352903 -0.37383137 -0.11539415
## SAL  0.095808846  1.00000000  0.30929922 -0.05133280 -0.01253342 -0.18567773
## Eh7  0.399654889  0.30929922  1.00000000  0.09401821 -0.15308284 -0.30543134
## pH   0.273529029 -0.05133280  0.09401821  1.00000000 -0.94637154 -0.40137180
## BUF -0.373831370 -0.01253342 -0.15308284 -0.94637154  1.00000000  0.38293556
## P   -0.115394148 -0.18567773 -0.30543134 -0.40137180  0.38293556  1.00000000
## K    0.068962897 -0.02063286  0.42261051  0.01922804 -0.07024704 -0.22647271
## Ca   0.093307112  0.08797761 -0.04212099  0.87797818 -0.79107974 -0.30669171
## Mg  -0.107821769 -0.01004276  0.29850251 -0.17614751  0.13045912 -0.06323688
## Na  -0.003762818  0.16226567  0.34246269 -0.03771997 -0.06071412 -0.16322762
```

```
## Mn   0.141540958 -0.25358394 -0.11125483 -0.47514280  0.42035664  0.49541023
## Zn  -0.272397809 -0.42083353 -0.23200548 -0.72216711  0.71468318  0.55740692
## Cu   0.012719279 -0.26600362  0.09454352  0.18135418 -0.14315285 -0.05313661
## NH4 -0.426213001 -0.15683469 -0.23896605 -0.74595877  0.84948759  0.48973876
##               K          Ca          Mg           Na         Mn         Zn
## H2S  0.06896290  0.09330711 -0.10782177 -0.003762818  0.1415410 -0.2723978
## SAL -0.02063286  0.08797761 -0.01004276  0.162265666 -0.2535839 -0.4208335
## Eh7  0.42261051 -0.04212099  0.29850251  0.342462687 -0.1112548 -0.2320055
## pH   0.01922804  0.87797818 -0.17614751 -0.037719968 -0.4751428 -0.7221671
## BUF -0.07024704 -0.79107974  0.13045912 -0.060714117  0.4203566  0.7146832
## P   -0.22647271 -0.30669171 -0.06323688 -0.163227625  0.4954102  0.5574069
## K    1.00000000 -0.26520634  0.86224465  0.792095939 -0.3474548  0.0736092
## Ca  -0.26520634  1.00000000 -0.41844612 -0.248186547 -0.3089848 -0.6998662
## Mg   0.86224465 -0.41844612  1.00000000  0.899469916 -0.2193897  0.3452170
## Na   0.79209594 -0.24818655  0.89946992  1.000000000 -0.3100614  0.1170469
## Mn  -0.34745478 -0.30898479 -0.21938970 -0.310061415  1.0000000  0.6033230
## Zn   0.07360920 -0.69986624  0.34521697  0.117046932  0.6033230  1.0000000
## Cu   0.69305055 -0.11224676  0.71206868  0.560069457 -0.2334684  0.2121025
## NH4 -0.11758057 -0.58260897  0.10822612 -0.107024306  0.5270207  0.7206793
##              Cu         NH4
## H2S  0.01271928 -0.42621300
## SAL -0.26600362 -0.15683469
## Eh7  0.09454352 -0.23896605
## pH   0.18135418 -0.74595877
## BUF -0.14315285  0.84948759
## P   -0.05313661  0.48973876
## K    0.69305055 -0.11758057
## Ca  -0.11224676 -0.58260897
## Mg   0.71206868  0.10822612
## Na   0.56006946 -0.10702431
## Mn  -0.23346835  0.52702068
## Zn   0.21210248  0.72067927
## Cu   1.00000000  0.01365659
## NH4  0.01365659  1.00000000
```

From the above scatter plot and correlation matrix result, we can say that there is collinearity between following predictors:

1. BUF and pH,NH4
2. pH and Ca
3. Mg and K, Na

We can confirm this by further analysis:

```r
ev<-eigen(corMat)
eigenvalues=ev$values
sum(1/eigenvalues)>(5*14)
```

```
## [1] TRUE
```

There is a condition that proves that collinearity exists in the data, i.e. if the sum of reciprocals of eigen values is greater than five times the number of predictor variables there exists collinearity. Since it is true in this case, collinearity exists! Lets do further diagnostics.

*Performing collinearity diagnostics to know more about collinearity:*

```
collDiag=ols_coll_diag(model_ols)
collDiag
```

```
## Tolerance and Variance Inflation Factor
## ---------------------------------------
##    Variables  Tolerance       VIF
## 1        H2S 0.33031035  3.027456
## 2        SAL 0.29519293  3.387615
## 3        Eh7 0.50570254  1.977447
## 4         pH 0.01610803 62.080846
## 5        BUF 0.02904296 34.431748
## 6          P 0.52748079  1.895804
## 7          K 0.13573843  7.367110
## 8         Ca 0.06001628 16.662146
## 9         Mg 0.04208005 23.764229
## 10        Na 0.09660862 10.351043
## 11        Mn 0.16166507  6.185628
## 12        Zn 0.08601057 11.626479
## 13        Cu 0.20707349  4.829203
## 14       NH4 0.11938152  8.376506
##
##
## Eigenvalue and Condition Index
## ------------------------------
##      Eigenvalue Condition Index    intercept          H2S          SAL
## 1  1.294944e+01        1.000000 1.379015e-06 4.927295e-06 2.469209e-05
## 2  1.026131e+00        3.552418 1.631652e-06 4.370603e-06 4.697219e-05
## 3  4.716343e-01        5.239898 7.335283e-07 1.753249e-06 3.198041e-06
## 4  2.229969e-01        7.620371 9.452610e-06 4.299452e-05 2.995299e-04
## 5  1.440425e-01        9.481568 4.799439e-06 8.111528e-05 5.811243e-04
## 6  6.169318e-02       14.487949 1.513221e-04 2.456443e-04 1.530679e-02
## 7  4.589855e-02       16.796779 1.546383e-04 4.312801e-04 5.640359e-04
## 8  3.353922e-02       19.649385 3.397399e-05 1.938328e-05 1.473366e-03
## 9  1.905222e-02       26.070702 6.704742e-05 4.812828e-05 7.248382e-03
## 10 1.028171e-02       35.488920 2.238372e-04 3.012610e-03 5.334881e-02
## 11 6.736775e-03       43.842917 9.367487e-04 2.413048e-03 6.852299e-02
## 12 4.070296e-03       56.404335 2.217502e-03 1.826069e-03 3.382145e-01
## 13 2.937430e-03       66.395965 1.331403e-03 2.815446e-08 1.435834e-02
## 14 1.376530e-03       96.991276 1.623828e-03 4.041484e-01 8.634098e-02
## 15 1.706479e-04      275.470413 9.932417e-01 5.877202e-01 4.136663e-01
##             Eh7           pH          BUF           P            K           Ca
## 1  3.920588e-05 5.694387e-06 3.856747e-05 0.000853382 8.657391e-05 7.178141e-05
## 2  2.302797e-05 1.021964e-04 1.348424e-03 0.027792284 2.972575e-04 5.431099e-03
## 3  1.331277e-04 4.723385e-05 4.239444e-04 0.077382658 4.659263e-03 1.039615e-02
## 4  2.725798e-04 1.675990e-08 5.332433e-03 0.562630856 1.977456e-03 1.117032e-03
## 5  8.919115e-04 1.301510e-08 7.970950e-03 0.088197485 1.951416e-03 1.652319e-03
## 6  8.909108e-04 1.731546e-05 5.889086e-03 0.035962329 2.300726e-04 1.862385e-02
## 7  1.703572e-02 1.022095e-03 8.973475e-06 0.019755495 5.145114e-04 4.481887e-02
## 8  3.144923e-03 5.691860e-04 3.835266e-04 0.039686539 1.238293e-01 3.357062e-02
## 9  1.658112e-02 1.341888e-03 6.036681e-02 0.020673561 2.683803e-01 1.636426e-01
## 10 1.222603e-01 2.218735e-03 7.090866e-02 0.008219367 1.334528e-01 4.957589e-02
## 11 3.304381e-01 1.603343e-02 7.641439e-02 0.008742983 8.126657e-03 5.380778e-02
```

```
## 12 1.487581e-01 7.338025e-02 2.614792e-01 0.083150327 1.517736e-01 1.888912e-02
## 13 1.051393e-01 2.667761e-03 1.929310e-03 0.014164395 1.764394e-01 4.466032e-03
## 14 2.142043e-01 1.608579e-01 1.035191e-01 0.007854784 2.341179e-02 1.844663e-02
## 15 4.018732e-02 7.417362e-01 4.039866e-01 0.004933554 1.048695e-01 5.754903e-01
##              Mg           Na           Mn           Zn           Cu
## 1  1.975017e-05 7.263437e-05 0.0002095668 7.942743e-05 7.272789e-05
## 2  1.482384e-05 2.329789e-04 0.0037360369 1.048976e-03 1.625697e-04
## 3  7.162789e-04 4.435346e-03 0.0128373136 1.147278e-04 5.963907e-04
## 4  3.087602e-04 3.251828e-03 0.0097439574 6.116354e-05 8.371869e-04
## 5  1.887260e-04 2.284242e-03 0.1887316690 4.137156e-03 3.977878e-04
## 6  1.871163e-04 9.123351e-03 0.0181583305 2.249175e-02 3.698115e-02
## 7  9.155913e-04 6.507452e-02 0.0209474881 4.968857e-02 2.035728e-02
## 8  1.053039e-03 7.717812e-02 0.0354239361 1.381533e-01 2.678695e-02
## 9  6.661018e-05 4.159672e-02 0.0010608205 4.596880e-02 5.604222e-02
## 10 1.886127e-03 3.723324e-03 0.0042532942 1.888128e-04 4.252986e-01
## 11 2.565401e-02 1.070023e-01 0.0999603306 2.045457e-01 1.867808e-01
## 12 1.281858e-03 1.045043e-01 0.0384957074 1.743962e-01 1.178785e-03
## 13 8.529493e-01 4.931528e-01 0.0905489110 2.560661e-01 4.415575e-02
## 14 9.919277e-04 1.123096e-03 0.0168126066 9.053418e-02 1.840632e-01
## 15 1.137661e-01 8.724451e-02 0.4590800312 1.252505e-02 1.628870e-02
##             NH4
## 1  1.331770e-04
## 2  2.629707e-03
## 3  5.739337e-06
## 4  8.346513e-03
## 5  1.136779e-02
## 6  1.137318e-01
## 7  1.162930e-01
## 8  2.299290e-02
## 9  1.204446e-01
## 10 1.863258e-01
## 11 1.099935e-01
## 12 8.643857e-02
## 13 4.151738e-02
## 14 1.791581e-02
## 15 1.618637e-01
```

```r
#sets of collinearity
sum(collDiag$eig_cindex$`Condition Index`>15)
```

```
## [1] 9
```

If the condition indices is small (<15), then predictor variables are not collinear. But from this results we can say that there are 9 sets of predictors shows collinearity as their condition indices>15.

```r
#predictors effected by collinearity:
vifvals=collDiag$vif_t$VIF
which(vifvals>10)
```

```
## [1]  4  5  8  9 10 12
```

From the above results, we can see that for variables pH, BUF, Ca, Mg, Na, Zn VIF> 10. This indicates that these predictors are effected by collinearity.

Therefore we proved that collinearity exists and as per the analysis the variables that contribute to collinearity are pH, BUF, Ca, Mg, Na, Zn.

# Project - Part2

Anusha Dasari

*Loading the required libraries:*

```
library(readxl)
```

*Loading LINTHALL data for analysis:*

```
data_2=read_excel("/Users/anusha_dasari/Downloads/LINTHALL.xlsx")
data_2=data_2[-1:-3]
data_2
```

```
## # A tibble: 45 x 15
##      BIO   H2S   SAL   Eh7    pH   BUF     P     K    Ca    Mg     Na    Mn
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>
##  1   676  -610    33  -290  5      2.34  20.2 1442. 2150  5169. 35184. 14.3
##  2   516  -570    35  -268  4.75   2.66  15.6 1299. 1845. 4358. 28170.  7.73
##  3  1052  -610    32  -282  4.2    4.18  18.7 1154. 1750. 4041. 26455  17.8
##  4   868  -560    30  -232  4.4    3.6   22.8 1045. 1674. 3966. 25073. 49.2
##  5  1008  -610    33  -318  5.55   1.9   37.8  522. 3360. 4609. 31664. 30.5
##  6   436  -620    33  -308  5.05   3.22  27.4 1273. 1811. 4390. 25492.  9.76
##  7   544  -590    36  -264  4.25   4.5   21.3 1346. 1907. 4579. 20877. 25.7
##  8   680  -610    30  -340  4.45   3.5   16.5 1254. 1860. 3983. 25621. 10.0
##  9   640  -580    38  -252  4.75   2.62  18.2 1243. 1799. 4142. 27587.  9.01
## 10   492  -610    30  -288  4.6    3.04  19.3 1282. 1797. 4264. 26512. 12.7
## # ... with 35 more rows, and 3 more variables: Zn <dbl>, Cu <dbl>, NH4 <dbl>
```

```
# Correlation Matrix of LINTHALL data
df=data_2[-1]
df_cor=cor(df)
```

```
# Eigen vectors
ev<-eigen(df_cor)
V=ev$vectors
K=ev$values
```

```
# Standardizing the data
data_scale<-scale(data_2)
data_scale = as.data.frame(data_scale)
```

*Performing linear regression on standardized data:*

```r
std_model <- lm(BIO~H2S+SAL+Eh7+pH+BUF+P+K+Ca+Mg+Na+Mn+Zn+Cu+NH4, data=data_scale)
coef(std_model)
```

```
##   (Intercept)           H2S           SAL           Eh7            pH 
##  1.272632e-16  1.994975e-02 -1.351380e-01  1.429480e-01  4.581740e-01 
##           BUF             P             K            Ca            Mg 
## -2.620829e-02 -7.111132e-02 -4.718649e-01 -3.021569e-01 -3.988137e-01 
##            Na            Mn            Zn            Cu           NH4 
##  4.640963e-02 -6.226077e-02 -2.357522e-01  5.422551e-01 -1.937359e-01
```

Above coefficients are estimates of theta . Regression coefficients of original data can be directly calculated using the formula:

$$\hat{\beta}_j = \frac{S_y}{S_j} * \hat{\theta}_j$$

Sy = Standard Deviation of Y. Sj = Standard Deviation of Xj

(This will be calculated later in this report.)

```r
# Finding principle components
PC= as.data.frame(as.matrix(data_scale[-1])%*%V)
PC$Y=data_scale$BIO
names(PC)=c('C1','C2','C3','C4','C5','C6','C7','C8','C9','C10','C11','C12','C13','C14','Y_tilde')
PC
```

```
##             C1          C2          C3          C4           C5           C6
## 1  -1.09972452  4.22794724 -0.18489241 -0.60595684  0.809830297 -0.024336458
## 2  -1.58206552  3.00784250 -1.58593536  0.09343406  0.439111749 -0.503446963
## 3  -0.44044374  2.54017472 -0.63623392 -0.56337290 -0.007631024  0.010728665
## 4  -0.33877226  2.04146668 -1.43605833  1.75116920 -0.359000935  0.607685674
## 5  -0.70015659  1.57752709  0.45756251 -0.40280601  1.627458933  0.246837864
## 6  -1.03344165  2.56512187 -0.20270445 -1.09131025  0.563248838 -0.356308649
## 7  -0.30989245  2.86946918 -1.46522861 -0.03361302  0.379626930  0.385672062
## 8  -0.71033954  2.42207953  0.63083495 -0.93115532 -0.049522904 -1.058096464
## 9  -1.68668206  3.02454532 -2.03907970 -0.11856906  0.838500169 -0.093512932
## 10 -0.97823307  2.73648518 -0.17012854 -0.52048487 -0.145033993 -0.165672150
## 11 -0.63688432 -2.09896433 -2.10568959  1.04519565 -1.286654788 -1.332068306
## 12  0.54591319 -1.08504011 -2.78170734  0.92474545  0.130218375  0.010746951
## 13  0.11388682 -1.27282213 -2.75849697  0.80051267 -1.160825991  0.101004676
## 14 -0.72794752 -1.38657189 -2.22905770 -0.60295044 -0.276543857 -1.187791230
## 15  0.58522133 -2.42533788 -0.99612712 -1.44535091 -1.208943699 -0.525363010
## 16  2.31524533 -0.40293388  0.01950434  0.15449274  0.502003906 -0.188978288
## 17  2.62321654 -1.93257950  0.80344025  0.45077546  0.523975286 -1.067103822
## 18  2.50563743 -2.65147832  0.24825597 -0.12221857  0.484489054 -0.421681112
## 19  1.38777362 -2.19275621 -0.62467934 -1.11573367  0.918166622 -0.229738196
## 20  1.43112336 -1.35563277 -1.50735887 -0.40679730  0.708185985 -0.637196316
## 21  2.92972396 -1.39111128  0.49455851 -1.50652003  0.009049219 -0.234566156
## 22  3.44594205 -1.64970263  0.70101740 -1.04657258 -0.170420514  0.421793958
## 23  3.04257410 -1.47947758 -0.41539201 -0.92233435 -0.527233167  0.890130675
## 24  3.74196277 -0.81630340 -0.04814267 -0.54607373  1.304555775  1.379991348
## 25  3.02985371 -0.72273663 -0.43581742 -0.88781778  2.038293463  0.557584081
## 26 -4.28297996 -2.33652757  1.30336264 -0.99446135 -0.202242397  0.495397177
## 27 -5.03237284 -1.93060169 -0.36808876 -0.89289547  0.288943920  1.702715835
## 28 -5.20881826 -2.19545416 -0.43204992 -0.11966569  0.491714825  0.411579418
```

```
## 29 -4.64267457 -2.72408764  1.02090043 -1.07250177 -0.163131550  0.632219524
## 30 -4.74678002 -2.84657751  1.07468463 -0.39054082  0.083727602 -0.870416317
## 31  0.03117391  1.75893918  1.69625432 -0.74824756 -0.734819227 -0.409860230
## 32  0.42406702  0.73135155  1.13427775 -1.03022641 -0.708657825 -0.286384797
## 33 -0.02982551  1.42277325  2.04274352 -0.55491059 -0.766379058 -0.086833412
## 34  0.30792243  2.02798075  1.55286958 -0.91520640 -0.650691641 -0.107743244
## 35 -0.09018252  1.01382654  1.73015989 -0.17094229 -0.594387025 -0.351601033
## 36  1.03631331  0.57929819  0.84834394  0.11292241 -0.892678076  1.334977556
## 37  2.16980532  0.08051310  1.07719279  0.46843786 -0.796948848 -0.004793546
## 38  1.98086677  0.73383737 -0.05635194  0.94641463 -1.181420131  0.938659838
## 39  1.79761508  0.72357802 -0.04872047  0.11011092 -1.230368696  0.749053424
## 40  0.69888193  0.80467704  0.58125255  0.01095357 -0.939364329 -0.652270773
## 41  0.18324549 -1.40851242  2.05015992  2.66199327  2.075526157 -0.736663476
## 42 -0.48334093 -0.34163662  0.11404758  2.74793702 -0.304975544  0.503762313
## 43 -0.53910138 -0.37572054  0.41758543  2.79214440 -0.421115858  0.183870579
## 44 -0.57424723  0.05180724  1.86898688  2.26492679  0.387235260 -1.026227653
## 45 -0.45305900  0.08132519  0.65994568  2.42306991  0.175128708  0.994242915
##              C7           C8          C9          C10         C11         C12
## 1  -0.428797283 -0.724855198 -0.35962882  0.014290841 -0.05337831  0.05353735
## 2  -0.242896851 -0.009475874 -0.30190519  0.150968627  0.31541655  0.23273668
## 3  -0.224386117  0.067509860  0.03485376 -0.045621061 -0.15006781 -0.42322190
## 4  -0.475246322 -0.112029250 -0.16867616  0.512035205 -0.12925029 -0.11806625
## 5  -0.115971101 -1.454563747  1.06140002  1.381029822 -0.10188499  0.19475221
## 6  -0.426400224  0.052965117 -0.34694758 -0.417645830 -0.14552998  0.16501899
## 7   0.531232112  0.387601744 -0.09293220 -0.675952852  0.56912486  0.29754411
## 8   0.064325496 -0.112856744 -0.34575103 -0.348412944 -0.11762469 -0.41277073
## 9   0.067379660  0.206325632  0.27915343 -0.161748557  0.12202995  0.10464988
## 10 -0.636303572  0.140893634 -0.47874075 -0.115202913 -0.33313064 -0.16273353
## 11 -0.486899493  0.640685328 -0.52437893  0.534141043  0.09719685  0.31263278
## 12  0.903903000 -0.302888364  0.41270276  0.156881807 -0.21602009 -0.08984171
## 13  0.016045710  0.676312498  0.48065642  0.034361123 -0.19090096 -0.45679750
## 14  0.271235172  0.334319304  0.17409191 -0.043565811 -0.17332092  0.06028779
## 15 -1.102430867 -0.185109846  0.28456411 -0.119539767 -0.61584598  0.10569482
## 16 -0.381683584 -0.304258591  0.58435297  0.078178518  0.61853566 -0.31353448
## 17 -0.641233235 -0.344159728  0.05474492  0.080321539  0.60051458 -0.28460917
## 18 -1.075996805 -0.457132999  0.61580852 -0.415761244  0.17487204  0.18877326
## 19 -0.519198511 -0.672232378  0.17148043 -0.696101937 -0.25464040  0.08664471
## 20  0.500476820 -0.899576758 -0.09661267 -0.401963013 -0.09011081  0.06354695
## 21  0.780933052 -0.661878863 -0.62173666  0.332486226 -0.39372091  0.08143976
## 22  0.072214009 -0.060555229  0.03080742  0.145718579  0.11752345 -0.05191727
## 23  1.124399877  0.382075832 -0.39086244  0.263240863 -0.06144626  0.39598864
## 24  0.164114246  1.077031626 -0.53404393  0.307349474 -0.05187306 -0.07127233
## 25  0.632364231  1.121228922 -0.23790208  0.095011620 -0.11707476 -0.19015139
## 26 -0.154071929 -0.222794488 -0.25634759 -0.108018807 -0.10234191 -0.34922797
## 27 -0.068624679  0.383994995  0.45343248 -0.447824663  0.16958252  0.04600080
## 28  0.565613296  0.169383557  0.02773349  0.304586752  0.37602922  0.05120930
## 29 -0.254520900 -0.021578506 -0.07496422 -0.006160226 -0.04157124 -0.11648905
## 30  0.358805450 -0.470677856 -0.82464123  0.453269907  0.39212068  0.06782038
## 31  0.414311696  0.487953581  0.32600730  0.371314683 -0.34449227 -0.08721158
## 32  0.710610554  0.517871734  0.61015677 -0.555844166  0.12104503  0.09507239
## 33 -0.153769909  0.409904115  0.63387671  0.130627482 -0.08830458  0.13585183
## 34  0.002197292  0.090466375 -0.06293667  0.345756726 -0.25197287  0.01001423
## 35  0.245163963  1.431298822  0.73804899 -0.058753371  0.18726136  0.30000995
## 36 -1.055961159 -0.431599104 -0.35937099 -0.228371857 -0.18853866  0.35051607
```

```
## 37  0.157284158 -1.123155865 -0.43326118 -0.369365598  0.46449745  0.14805000
## 38 -0.141791462 -0.160991338 -0.27016920  0.171455879  0.35780263 -0.17778979
## 39 -0.385736454 -0.352796927  0.05493159 -0.009501176  0.49296725 -0.22568799
## 40  0.427604076  0.521703071 -0.18732897  0.317458987  0.28948170 -0.16921669
## 41 -1.355534002  1.456565273 -0.29888078 -0.030476595 -0.16793084  0.03467077
## 42 -0.019700430 -0.247322032 -0.00120972  0.035247211 -0.12219210  0.15143691
## 43 -0.082145185  0.048101333  0.13621656 -0.042743355 -0.34604181  0.19003674
## 44  1.951378689 -0.507623873  0.12304912 -0.438384380 -0.22848659 -0.05361333
## 45  0.467707513 -0.764078796 -0.01884069 -0.478772792 -0.38830802 -0.16978462
##             C13          C14      Y_tilde
## 1   0.049107894 -0.052409266 -0.492062640
## 2   0.179603178  0.065113533 -0.734458029
## 3   0.015278894  0.057235174  0.077566524
## 4  -0.083249195  0.095333865 -0.201188173
## 5  -0.303222202 -0.038276432  0.010907793
## 6  -0.242637720 -0.255423250 -0.855655723
## 7  -0.245647017  0.126560803 -0.692038836
## 8   0.027386844  0.074550438 -0.486002755
## 9   0.320849821 -0.039052321 -0.546601602
## 10 -0.122145027  0.117614023 -0.770817337
## 11 -0.185381526  0.043444162 -0.025451516
## 12  0.120854318  0.063971830  0.604776496
## 13 -0.263133256 -0.035783360  0.416920069
## 14  0.015925925 -0.222374096  1.113806812
## 15  0.275829550  0.013741850  0.004847908
## 16  0.191372237  0.082793314 -0.916254570
## 17  0.011714951 -0.015185986 -0.982913302
## 18 -0.160319881  0.162366584 -1.019272611
## 19  0.145703880  0.171750215 -0.922314455
## 20 -0.242616485 -0.067022461 -1.158649959
## 21 -0.015723694 -0.001515507 -0.922314455
## 22 -0.038597497 -0.116790435 -1.110170882
## 23  0.006747101  0.138538993 -1.134410420
## 24 -0.126668818  0.028263088 -1.158649959
## 25  0.283889405 -0.135464827 -1.001092957
## 26 -0.202060527  0.044167953  2.174286639
## 27 -0.001792742  0.042713555  1.840992979
## 28  0.249568729  0.002948017  1.659196438
## 29  0.014235939 -0.082228047  0.998669003
## 30 -0.039399219  0.024015230  1.925831366
## 31 -0.028598079  0.073514842 -0.267846905
## 32 -0.247802118 -0.096359425  0.295722375
## 33  0.393670370 -0.030042431  1.453160357
## 34 -0.011744912  0.043006982  1.634956899
## 35 -0.203493968  0.033798516  1.156226005
## 36  0.100320232  0.044412630 -0.892015031
## 37  0.130719754 -0.132299340 -0.885955147
## 38 -0.120175773  0.007231405 -0.752637683
## 39 -0.017754720 -0.177732265 -0.770817337
## 40  0.244182021  0.066921585 -0.552661487
## 41 -0.076180413 -0.019858474  1.144106236
## 42  0.129104721 -0.039005771  0.350261337
## 43  0.123353140 -0.152276199  0.604776496
## 44  0.007936801  0.089107957  0.938070155
```

```
## 45 -0.059010912 -0.004016648  0.847171885
```

```
# Performing principle component regression
pcr_full=lm (Y_tilde~C1+C2+C3+C4+C5+C6+C7+C8+C9+C10+C11+C12+C13+C14,data=PC)
alpha=coef(pcr_full)
alpha
```

```
##   (Intercept)            C1            C2            C3            C4
##  1.248943e-16 -3.274690e-01 -1.195167e-01  2.017575e-01  1.391803e-01
##            C5            C6            C7            C8            C9
## -1.275601e-01 -3.111161e-02  2.058651e-01  2.945784e-01  5.013478e-01
##           C10           C11           C12           C13           C14
##  2.028432e-01 -5.096238e-01 -2.170568e-01  6.097098e-02 -4.357941e-01
```

```
theta= V %*% alpha[-1]
theta
```

```
##              [,1]
##  [1,]  0.01994975
##  [2,] -0.13513800
##  [3,]  0.14294802
##  [4,]  0.45817397
##  [5,] -0.02620829
##  [6,] -0.07111132
##  [7,] -0.47186494
##  [8,] -0.30215695
##  [9,] -0.39881373
## [10,]  0.04640963
## [11,] -0.06226077
## [12,] -0.23575222
## [13,]  0.54225511
## [14,] -0.19373594
```

```
s = 'Y_tilde~C1'
p = length(theta)
theta_mat = matrix(nrow=p, ncol=(2+p))
i=1
while(i<=p){
  pcr = lm(as.formula(paste(s,'-1')),data=PC)
  alpha = as.matrix(coef(pcr))
  theta= V[,1:i]%*%alpha
  r2 = summary(pcr)$r.squared
  theta_mat[i,1] = i
  theta_mat[i,2] = r2
  theta_mat[i,3:(p+2)]= theta
  s = paste(s,sprintf('+C%d',i+1))
  i=i+1}
theta_mat = as.data.frame(theta_mat)
names(theta_mat) = c('ncomp', 'R^2', paste('theta',1:p,sep=''))
theta_mat
```

```
##    ncomp       R^2      theta1      theta2       theta3    theta4      theta5
```

```
## 1       1 0.5280199 0.053586123  0.03533178  0.040544897 0.1336784 -0.13481258
## 2       2 0.5808034 0.052500201  0.03326122  0.013624100 0.1369612 -0.13476937
## 3       3 0.6462232 0.005759263 -0.08894865 -0.078831413 0.1939921 -0.17611332
## 4       4 0.6720818 0.101754937 -0.12658142 -0.036894631 0.2053667 -0.19919370
## 5       5 0.6833352 0.099919868 -0.19147659 -0.015622934 0.1936798 -0.17843798
## 6       6 0.6838197 0.112966460 -0.19179007 -0.034185708 0.1877296 -0.17768941
## 7       7 0.7001547 0.174745358 -0.11278514 -0.095300218 0.1994427 -0.15358578
## 8       8 0.7331969 0.196471970 -0.14250010 -0.003173178 0.2081439 -0.10863005
## 9       9 0.7749131 0.112093899 -0.05473123  0.110199485 0.1961529 -0.15646669
## 10     10 0.7807966 0.172102945 -0.10090249  0.127188343 0.2259625 -0.13595551
## 11     11 0.8033574 0.058494036 -0.14596579  0.138953316 0.2047304 -0.31166487
## 12     12 0.8054911 0.055149749 -0.17987214  0.150982769 0.1328516 -0.21280436
## 13     13 0.8056021 0.054731263 -0.17408734  0.153024803 0.1312701 -0.23422905
## 14     14 0.8074049 0.019949754 -0.13513800  0.142948017 0.4581740 -0.02620829
##          theta6       theta7       theta8      theta9      theta10      theta11
## 1  -0.08946321  0.010952364  0.117417811 -0.02588088  0.005609677 -0.090735829
## 2  -0.07616372 -0.047358304  0.138984054 -0.08547820 -0.050615620 -0.068964232
## 3  -0.04377292 -0.042736707  0.180666107 -0.07548813 -0.060819431 -0.072968954
## 4  -0.01594177 -0.036751892  0.173096796 -0.08057668 -0.068385005 -0.005734027
## 5  -0.11124380 -0.028843379  0.146800056 -0.09381650 -0.098938123 -0.010696026
## 6  -0.11068680 -0.028327336  0.133497391 -0.09487995 -0.097057725 -0.020014304
## 7  -0.18004862 -0.042206966  0.155102692 -0.10397814 -0.134455354  0.005629722
## 8  -0.06261141 -0.008302201  0.100343918 -0.15434984 -0.266997790 -0.150999484
## 9  -0.10163011 -0.288598484  0.006886607 -0.14868804 -0.311201669 -0.194173803
## 10 -0.10521746 -0.401177340  0.021848854 -0.12605447 -0.222112896 -0.267531435
## 11 -0.08761385 -0.512221035 -0.238726866 -0.18659715 -0.111915363 -0.129977341
## 12 -0.07354378 -0.518798001 -0.163500682 -0.27294029 -0.033038816 -0.113084609
## 13 -0.07753536 -0.503584282 -0.168350639 -0.31501801 -0.016197959 -0.123626067
## 14 -0.07111132 -0.471864943 -0.302156946 -0.39881373  0.046409628 -0.062260768
##         theta12      theta13     theta14
## 1  -0.13236142  0.003532769 -0.13057956
## 2  -0.14297724 -0.043282766 -0.12747592
## 3  -0.10739263  0.032727332 -0.12533586
## 4  -0.08650899  0.046926940 -0.13982271
## 5  -0.08551815  0.038835332 -0.13907561
## 6  -0.08658688  0.036408832 -0.15095049
## 7  -0.10159585  0.152224561 -0.06958182
## 8  -0.16302293  0.233844704 -0.02686051
## 9   0.05729689  0.422705439 -0.23747653
## 10  0.06021906  0.396499043 -0.15761383
## 11 -0.22957079  0.494791321 -0.09123691
## 12 -0.27792021  0.561012429 -0.15668171
## 13 -0.25375552  0.561035103 -0.14260988
## 14 -0.23575222  0.542255110 -0.19373594
```

We use principal components C1,C2,C3,C4,C5,C6,C7,C8,C9,C10 and C11 in the model as per the Rsquare results in above table.

Now lets calculate the regression coefficients of the original data using theta values generated previously. Formula used:

$$\hat{\beta}_j = \frac{S_y}{S_j} * \hat{\theta}_j$$

Regression Coefficients of the original model:

```
beta1=(sd(data_2$BIO)/sd(data_2$H2S))*theta[1]
beta2=(sd(data_2$BIO)/sd(data_2$SAL))*theta[2]
beta3=(sd(data_2$BIO)/sd(data_2$Eh7))*theta[3]
beta4=(sd(data_2$BIO)/sd(data_2$pH))*theta[4]
beta5=(sd(data_2$BIO)/sd(data_2$BUF))*theta[5]
beta6=(sd(data_2$BIO)/sd(data_2$P))*theta[6]
beta7=(sd(data_2$BIO)/sd(data_2$K))*theta[7]
beta8=(sd(data_2$BIO)/sd(data_2$Ca))*theta[8]
beta9=(sd(data_2$BIO)/sd(data_2$Mg))*theta[9]
beta10=(sd(data_2$BIO)/sd(data_2$Na))*theta[10]
beta11=(sd(data_2$BIO)/sd(data_2$Mn))*theta[11]
beta12=(sd(data_2$BIO)/sd(data_2$Zn))*theta[12]
beta13=(sd(data_2$BIO)/sd(data_2$Cu))*theta[13]
beta14=(sd(data_2$BIO)/sd(data_2$NH4))*theta[14]

betas=c(beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9,beta10,beta11,beta12,beta13,beta14)

betas
```

```
##  [1]   0.428999215 -23.980715733   2.553223782 242.527810058  -6.902267789
##  [6]  -1.701510693  -1.046591019  -0.116070623  -0.280228359   0.004451049
## [11]  -1.678759799 -18.794521173 345.162813094  -2.705172439
```

Intercept of the original model:

```
bi<-c()
for(x in 1:14){
  bi[x]=betas[x]*(sum(data_2[x+1])/44)
}
interceptEsti = mean(data_2$BIO) - sum(bi)
interceptEsti
```

```
## [1] 2953.324
```

# Project - Part 3

**1) Stepwise Regression:**

```
library(readxl)
library(leaps)
library(car)
```

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

```
linth_df=read_excel("/Users/anusha_dasari/Downloads/LINTH-5.xlsx") #loading the data to R
linth_df=linth_df[-1:-3] #removing unnecessary variables
linth_df # displaying the content of the data
```

```
## # A tibble: 45 x 6
##      BIO   SAL    pH     K     Na    Zn
##    <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>
##  1   676    33  5    1442. 35184.  16.5
##  2   516    35  4.75 1299. 28170.  14.0
##  3  1052    32  4.2  1154. 26455   15.3
##  4   868    30  4.4  1045. 25073.  17.3
##  5  1008    33  5.55  522. 31664.  22.3
##  6   436    33  5.05 1273. 25492.  12.3
##  7   544    36  4.25 1346. 20877.  17.8
##  8   680    30  4.45 1254. 25621.  14.4
##  9   640    38  4.75 1243. 27587.  13.7
## 10   492    30  4.6  1282. 26512.  11.8
## # ... with 35 more rows
```

```
model1<-lm(BIO~SAL,data=linth_df)
model2<-lm(BIO~pH,data=linth_df)
model3<-lm(BIO~K,data=linth_df)
model4<-lm(BIO~Na,data=linth_df)
model5<-lm(BIO~Zn,data=linth_df)

summary(model1)$coefficients
```

```
##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept) 1554.90670  820.68191  1.8946521 0.06487582
## SAL          -18.30749   26.91701 -0.6801458 0.50005789
```

```
summary(model2)$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -885.2105  243.44073 -3.636247 7.349989e-04
## pH           409.8043   51.09422  8.020561 4.433213e-10
```

```r
summary(model3)$coefficients
```

```
##                 Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept) 1362.8413219 281.4780668  4.841732 1.696924e-05
## K             -0.4539004   0.3310816 -1.370962 1.775003e-01
```

```r
summary(model4)$coefficients
```

```
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept) 1433.86803474 252.45875717  5.679613 1.067247e-06
## Na            -0.02609361   0.01407409 -1.854017 7.060418e-02
```

```r
summary(model5)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 1890.60677 186.703828 10.126235 5.890805e-13
## Zn           -49.77873   9.496139 -5.241997 4.566092e-06
```

From the above results, we have to choose the one with smallest p-value and p-value less than alphaE=0.15 is *pH*. pH is entered to the equation. In next step, we will check the next predictor that can be added.

```r
model6<-lm(BIO~pH+SAL,data=linth_df)
model7<-lm(BIO~pH+K,data=linth_df)
model8<-lm(BIO~pH+Na,data=linth_df)
model9<-lm(BIO~pH+Zn,data=linth_df)

summary(model6)$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -535.69792  588.25824 -0.9106509 3.676763e-01
## pH           408.07631   51.50591  7.9229032 7.171988e-10
## SAL          -11.28502   17.26675 -0.6535695 5.169516e-01
```

```r
summary(model7)$coefficients
```

```
##                 Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept) -506.9773515 279.7713769 -1.812113 7.712215e-02
## pH           412.0395455  48.4975260  8.496094 1.148936e-10
## K             -0.4870976   0.2032112 -2.397001 2.105660e-02
```

```r
summary(model8)$coefficients
```

```
##                  Estimate   Std. Error   t value     Pr(>|t|)
## (Intercept) -475.72558011 2.735227e+02 -1.739255 8.931536e-02
## pH           404.94818933 4.776984e+01  8.477068 1.220284e-10
## Na            -0.02332607 8.655196e-03 -2.695036 1.007761e-02
```

```
summary(model9)$coefficients
```

```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -450.5213  506.92861 -0.8887274 3.792115e-01
## pH           357.6212   73.90344  4.8390330 1.792288e-05
## Zn           -10.8827   11.13035 -0.9777496 3.337967e-01
```

On observing the above results, predictor variable 'Na' has smaller p-value and is less than alphaE=0.15, hence 'Na' enters the equation. But, now we have to check if this affects 'pH'. The p-value of estimated coefficient of 'pH' is less than alphaR=0.15, therefore it does not have any negative effect and 'Na' is added to the regression equation.

Next, we will check for other predictors that can be added.

```
model10<-lm(BIO~pH+Na+SAL,data=linth_df)
model11<-lm(BIO~pH+Na+K,data=linth_df)
model12<-lm(BIO~pH+Na+Zn,data=linth_df)

summary(model10)$coefficients
```

```
##                 Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) -344.32124135 5.569748e+02 -0.6181990 5.398634e-01
## pH           404.34552789 4.835625e+01  8.3618050 2.122008e-10
## Na            -0.02293885 8.867412e-03 -2.5868706 1.333140e-02
## SAL           -4.46224518 1.641690e+01 -0.2718081 7.871337e-01
```

```
summary(model11)$coefficients
```

```
##                 Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) -447.90979141 282.01696242 -1.5882371 1.199157e-01
## pH           406.82620617  48.36933385  8.4108292 1.820207e-10
## Na            -0.01783237   0.01435492 -1.2422480 2.212038e-01
## K             -0.16002059   0.33180045 -0.4822796 6.321722e-01
```

```
summary(model12)$coefficients
```

```
##                 Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) -195.42611063 4.865500e+02 -0.4016568 6.900227e-01
## pH           369.78510921 6.959675e+01  5.3132529 4.073773e-06
## Na            -0.02252891 8.782881e-03 -2.5650931 1.407138e-02
## Zn            -7.36781014 1.054677e+01 -0.6985843 4.887556e-01
```

As you can see from the above result, the p-value for estimated coefficients of SAL,K and Zn are greater than alphaE=0.15, none of the predictors enters the regression equation. Hence, our final model will be **BIO~pH+Na**.

**2)Best Subset Selection:**

```
#Identify all possible models
subsets <- regsubsets(BIO~SAL+pH+K+Na+Zn, method="exhaustive", nbest=2, data=linth_df )
summary(subsets)
```

```
## Subset selection object
## Call: regsubsets.formula(BIO ~ SAL + pH + K + Na + Zn, method = "exhaustive",
##     nbest = 2, data = linth_df)
## 5 Variables  (and intercept)
##     Forced in Forced out
## SAL     FALSE      FALSE
## pH      FALSE      FALSE
## K       FALSE      FALSE
## Na      FALSE      FALSE
## Zn      FALSE      FALSE
## 2 subsets of each size up to 5
## Selection Algorithm: exhaustive
##          SAL pH  K   Na  Zn
## 1  ( 1 ) " " "*" " " " " " "
## 1  ( 2 ) " " " " " " " " "*"
## 2  ( 1 ) " " "*" " " "*" " "
## 2  ( 2 ) " " "*" "*" " " " "
## 3  ( 1 ) " " "*" " " "*" "*"
## 3  ( 2 ) " " "*" "*" "*" " "
## 4  ( 1 ) "*" "*" "*" " " "*"
## 4  ( 2 ) "*" "*" " " "*" "*"
## 5  ( 1 ) "*" "*" "*" "*" "*"
```

```
summary(subsets)$adjr2
```

```
## [1] 0.5900471 0.3756964 0.6421676 0.6307938 0.6377518 0.6355077 0.6423444
## [8] 0.6389476 0.6359404
```

```
summary(subsets)$cp
```

```
## [1]  7.420574 32.738066  2.281592  3.593736  3.796000  4.048722  4.296370
## [8]  4.669587  6.000000
```

Two-variable models with small Cp values are pH+Na(2.281592) and pH+K(3.593736).

```
model_13=lm(BIO~pH+Na, data=linth_df)
model_14=lm(BIO~pH+K, data=linth_df)

vif(model_13)
```

```
##       pH       Na
## 1.001425 1.001425
```

```
vif(model_14)
```

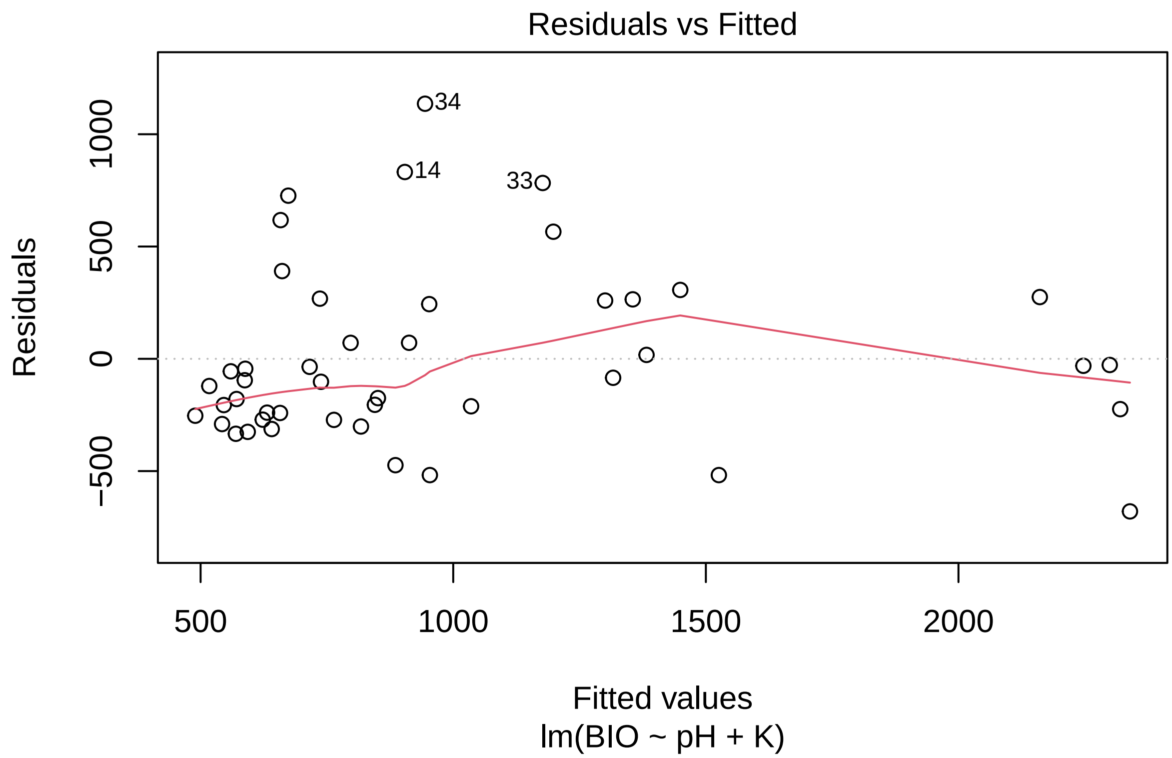
```
##      pH       K
## 1.00037 1.00037
```

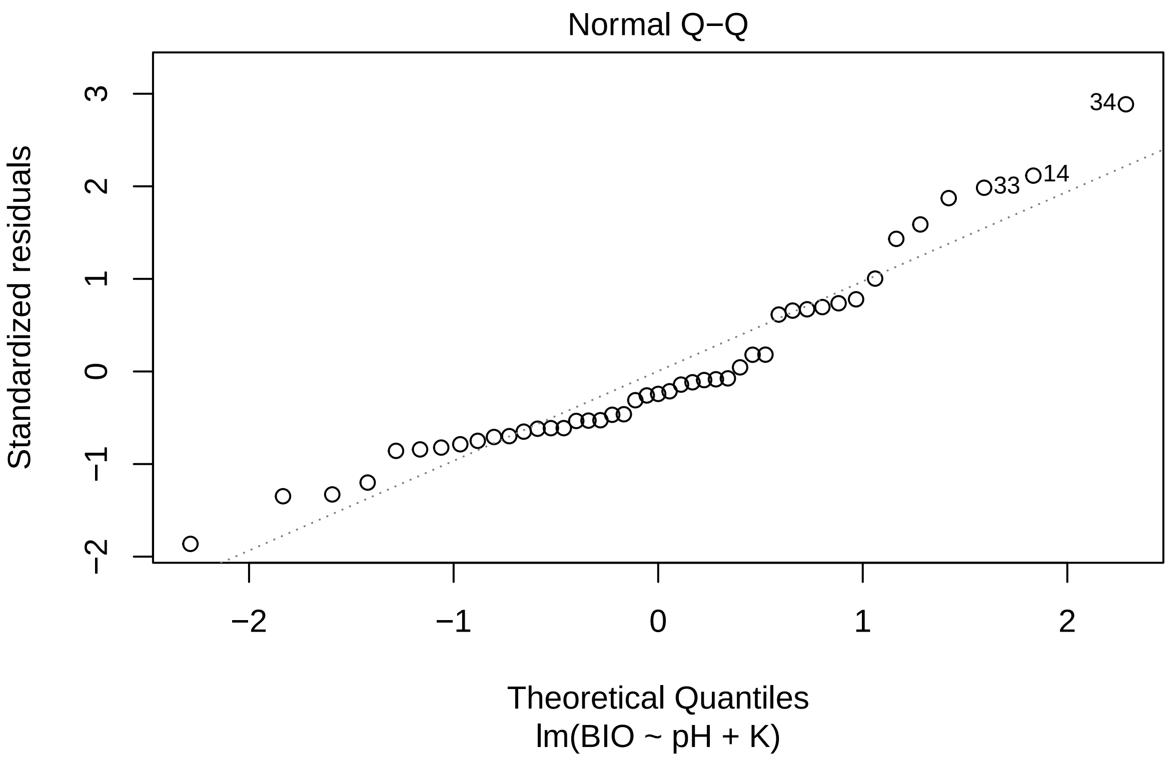
pH+K model has lower VIF value.

**On the basis of Cp, BIO~pH+K is the best two-variable model!**

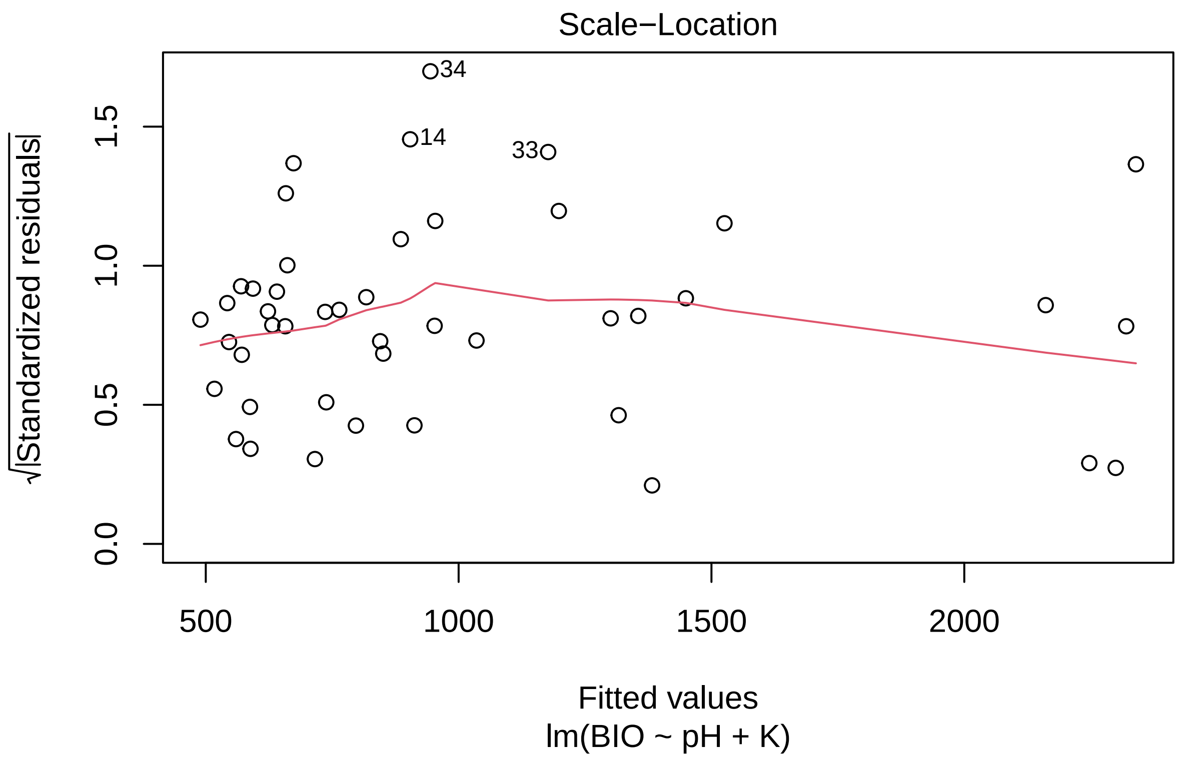
Below are the fitted plots for the final model.

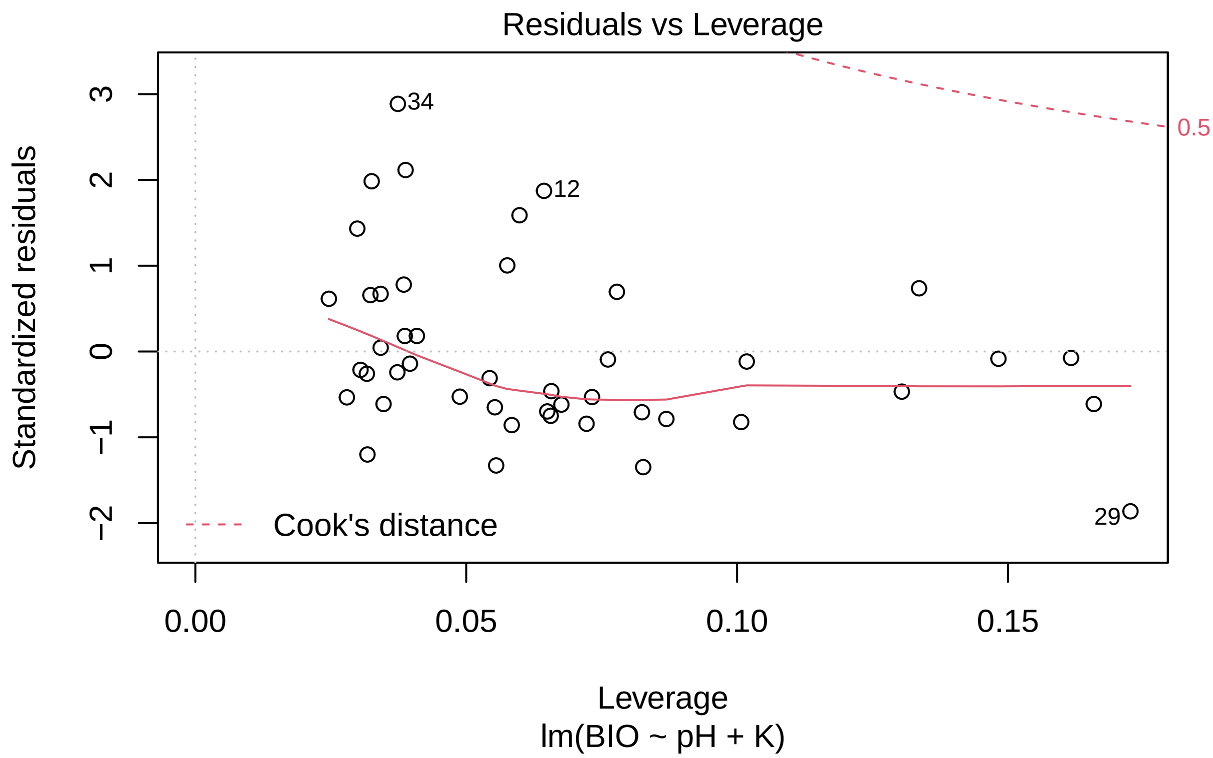
```
plot(lm(BIO~pH+K,data=linth_df))
```





Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(BIO ~ pH + K)
34
33
14
3
2
1
0
−1
−2
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
34
14
33
500
1000
1500
2000
Fitted values
lm(BIO ~ pH + K)




As you observe the residual plot, it is horizontally distributed along line 0, there is no trend of linear relationship. And in Normal probability plot, the distribution around both sides of linear line is roughly equal. This shows that all necessary predictors were included in the model to predict Y (i.e BIO).

```
k=coef(model_14)
k
```

```
##   (Intercept)            pH             K
## -506.9773515  412.0395455   -0.4870976
```

Final best model is BIO = -506.9773515 + (412.0395455)pH - (0.4870976)K