Group- 12

Nikita Pai & Abhilash Hemaraj

03/01/2020

Problem 1

Problem 1 (25points): Gradient Descent Algorithm for Multiple Linear Regression The file concrete.csv includes 1,030 types of concrete with numerical features indicating characteristics of the concrete. The variable "strength" is treated as the response variable. • Standardize all variables (including the response variable "strength"). Split the data set into a training set (60%) and a validation set (40%). • Implement the gradient descent algorithm in R with the ordinary least square cost function. • Fit the multiple linear regression model using the gradient descent algorithm and the training set. Try out different learning rates: alpha = 0.01,0.1,0.3,0.5 and compare the speed of convergence by plotting the cost function. Determine the number of iterations needed for each alpha value. • Apply the fitted regression model to the validation set and evaluate the model performance (ME, RMSE, MAE, MPE, MPAE). Calculate the correlation between the predicted strength and the actual strength. Create a lift chart to show model performance

```
concrete <- read.csv("C:/Users/abhil/Downloads/concrete.csv", stringsAsFactors=FALSE)
View(concrete)
```

```
#Standardize all variables (including the response variable "strength").

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

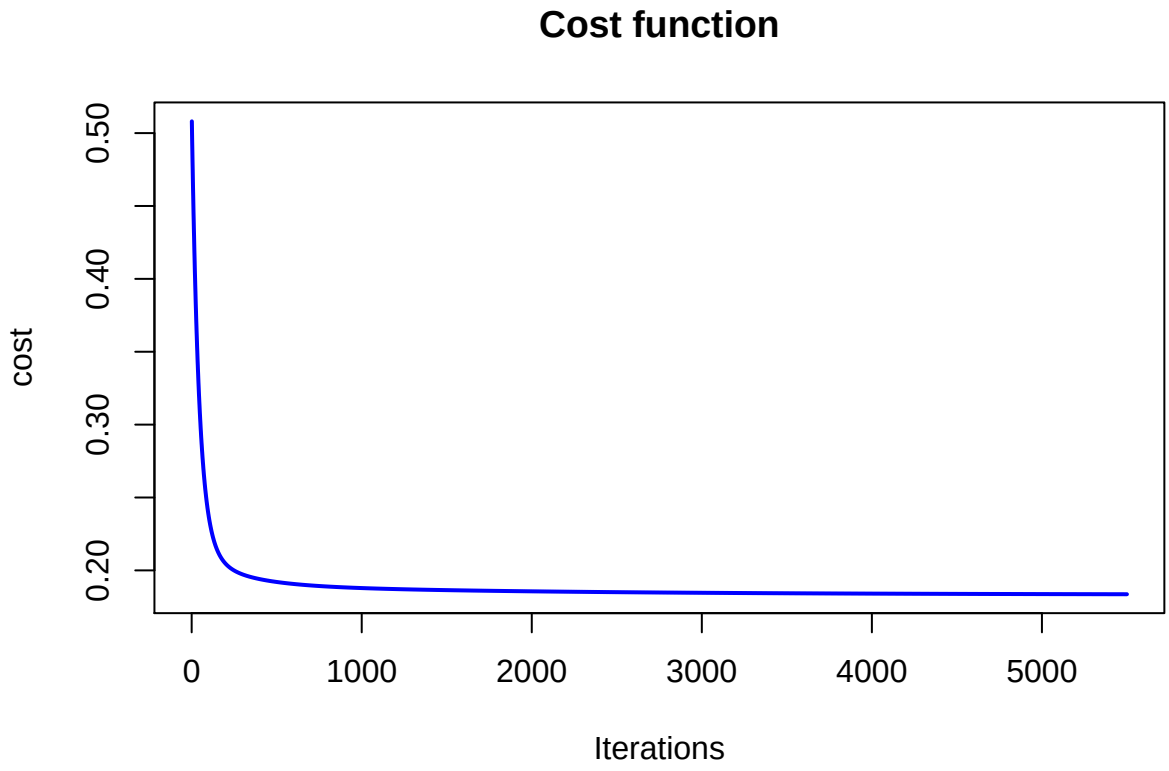
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
concrete_std <- concrete %>% mutate_all(scale)

# Split the data set into a training set (60%) and a validation set (40%).
size_ <- floor(0.60 * nrow(concrete_std))

set.seed(50)

concrete_std$x0 = 1
train_ind <- sample(seq_len(nrow(concrete_std)), size = size_)
train_concrete <- concrete[train_ind,]
train_concrete_std <- concrete_std[train_ind,] # Training set
test_concrete <- concrete[-train_ind,]
test_concrete_std <- concrete_std[-train_ind,] # Validation set
```

```r
library(Metrics)
library(psych)

X_validation <- as.matrix(test_concrete_std[c(10,1:8)])
Y_validation <- as.matrix(test_concrete_std[,9])
#Implement the gradient descent algorithm in R with the ordinary least square cost function.

#Gradient Descent Algorithm:
Jcost <- function(X, y, b){
  m <- length(y)
  return((t(X%*%b-y))%*%(X%*%b-y)/(2*m))
}
concrete_std$x0 = 1

X <- as.matrix(train_concrete_std[c(10,1:8)])

Y <- as.matrix(train_concrete_std[,9])


# When alpha = 0.01, we have to run over 5000 iterations to reach optimal value
alpha_ = 0.01
niter = 5500
J_history <- c()
J_history[1:niter] = 0
# Fit the multiple linear regression model using the gradient descent algorithm and the training set.
#Try out different learning rates: alpha = 0.01,0.1,0.3,0.5 and compare the speed of convergence
# by plotting the cost function. Determine the number of iterations needed.
b0 <- as.matrix(c(0,0,0,0,0,0,0,0,0))
brun <- b0
m = length(Y)

for (iter in 1:niter){
  brun = b0 + alpha_*t(t(Y-X%*%b0)%*%X)/m
  J_history[iter] <- Jcost(X, Y, brun)
  #J_history[iter] <<- Jcost(X,Y,brun)
  #print(J_history[iter])
  b0 <<- brun # making b0 a global variable
}

plot(J_history, type='line', col='blue', lwd=2, main='Cost function', ylab='cost', xlab='Iterations')
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```




```
#compare with normal equations

b1 <- solve(t(X)%*%X)%*%(t(X)%*%Y)
print(head(b0))
```

```
##                    [,1]
## x0           0.02452355
## cement       0.70007141
## slag         0.47557018
## ash          0.28306118
## water       -0.27771072
## superplastic 0.09091275
```

```
print(head(b1))
```

```
##                    [,1]
## x0           0.02443147
## cement       0.76736711
## slag         0.54076185
## ash          0.34035926
## water       -0.21894856
## superplastic 0.09884737
```

```
y_predict <- X%*%b0

print(head(y_predict))
```

```
##           [,1]
```

```
## 11   0.29943911
## 820 -0.77934580
## 863 -0.08546577
## 882 -0.04381692
## 814 -0.15645021
## 707 -0.27799655
```

```
y_validation_predict <- X_validation%*%b0
# Evaluate the model performance
# ME, RMSE, MAE, MPE, MAPE

Metrics::rmse(Y_validation, y_validation_predict)
```

```
## [1] 0.6445699
```

```
Metrics::mae(Y_validation, y_validation_predict)
```

```
## [1] 0.5141353
```

```
Metrics::mape(Y_validation, y_validation_predict)
```

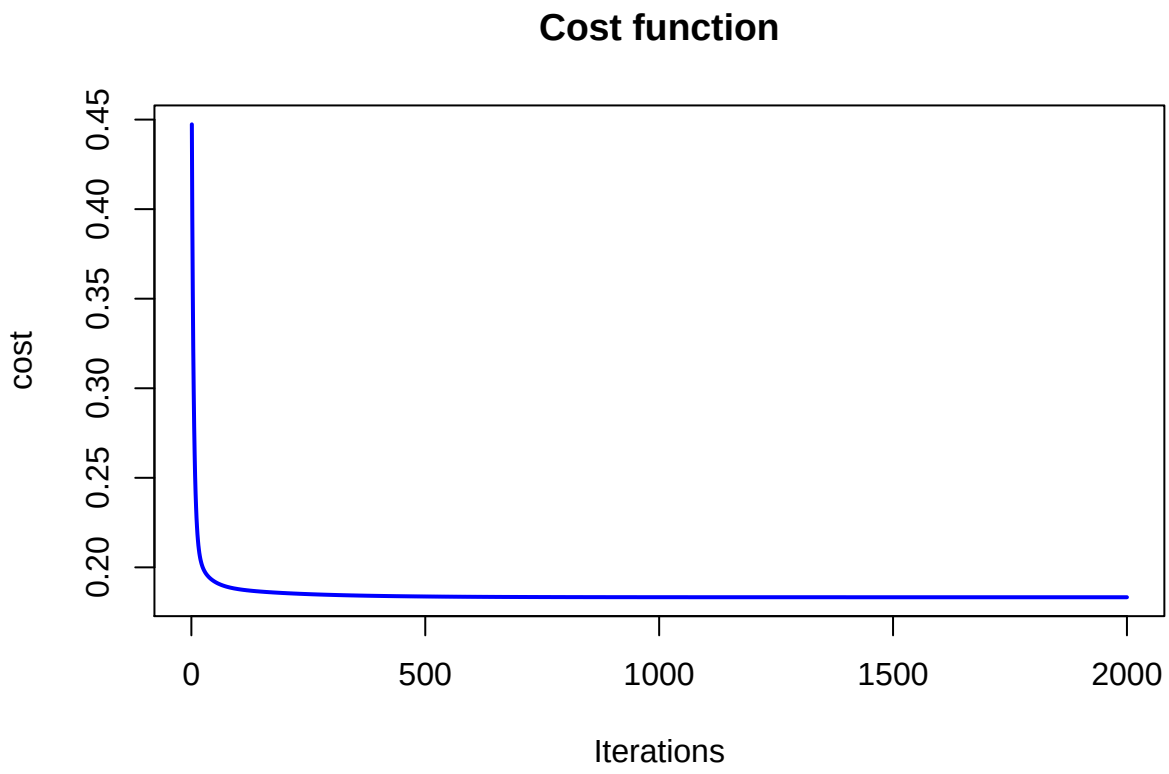
```
## [1] 2.13642
```

```
# correlation between actual and predicted values of strength
print(cor(Y_validation, y_validation_predict))
```

```
##           [,1]
## [1,] 0.7558538
```

```
# When alpha = 0.1, we have to run about 2000 iterations to reach optimal value
alpha_ = 0.1
niter = 2000
J_history <- c()
J_history[1:niter] = 0
# Fit the multiple linear regression model using the gradient descent algorithm and the training set.
#Try out different learning rates: alpha = 0.01,0.1,0.3,0.5 and compare the speed of convergence
# by plotting the cost function. Determine the number of iterations needed.
b0 <- as.matrix(c(0,0,0,0,0,0,0,0,0))
brun <- b0
m = length(Y)

for (iter in 1:niter){
  brun = b0 + alpha_*t(t(Y-X%*%b0)%*%X)/m
  J_history[iter] <- Jcost(X, Y, brun)
  #J_history[iter] <<- Jcost(X,Y,brun)
  #print(J_history[iter])
  b0 <<- brun # making b0 a global variable
}

plot(J_history, type='line', col='blue', lwd=2, main='Cost function', ylab='cost', xlab='Iterations')
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```




```
#compare with normal equations

b1 <- solve(t(X)%*%X)%*%(t(X)%*%Y)
print(head(b0))
```

```
##                     [,1]
## x0            0.02443298
## cement        0.76626472
## slag          0.53969393
## ash           0.33942068
## water        -0.21991133
## superplastic  0.09871724
```

```
print(head(b1))
```

```
##                     [,1]
## x0            0.02443147
## cement        0.76736711
## slag          0.54076185
## ash           0.34035926
## water        -0.21894856
## superplastic  0.09884737
```

```
y_predict <- X%*%b0

print(head(y_predict))
```

```
##           [,1]
```

```
## 11   0.26457719
## 820 -0.79422047
## 863 -0.07634989
## 882 -0.03234272
## 814 -0.16205371
## 707 -0.28176913
```

```
y_validation_predict <- X_validation%*%b0
# Evaluate the model performance
# ME, RMSE, MAE, MPE, MAPE

Metrics::rmse(Y_validation, y_validation_predict)
```

```
## [1] 0.6440725
```

```
Metrics::mae(Y_validation, y_validation_predict)
```

```
## [1] 0.5129379
```

```
Metrics::mape(Y_validation, y_validation_predict)
```

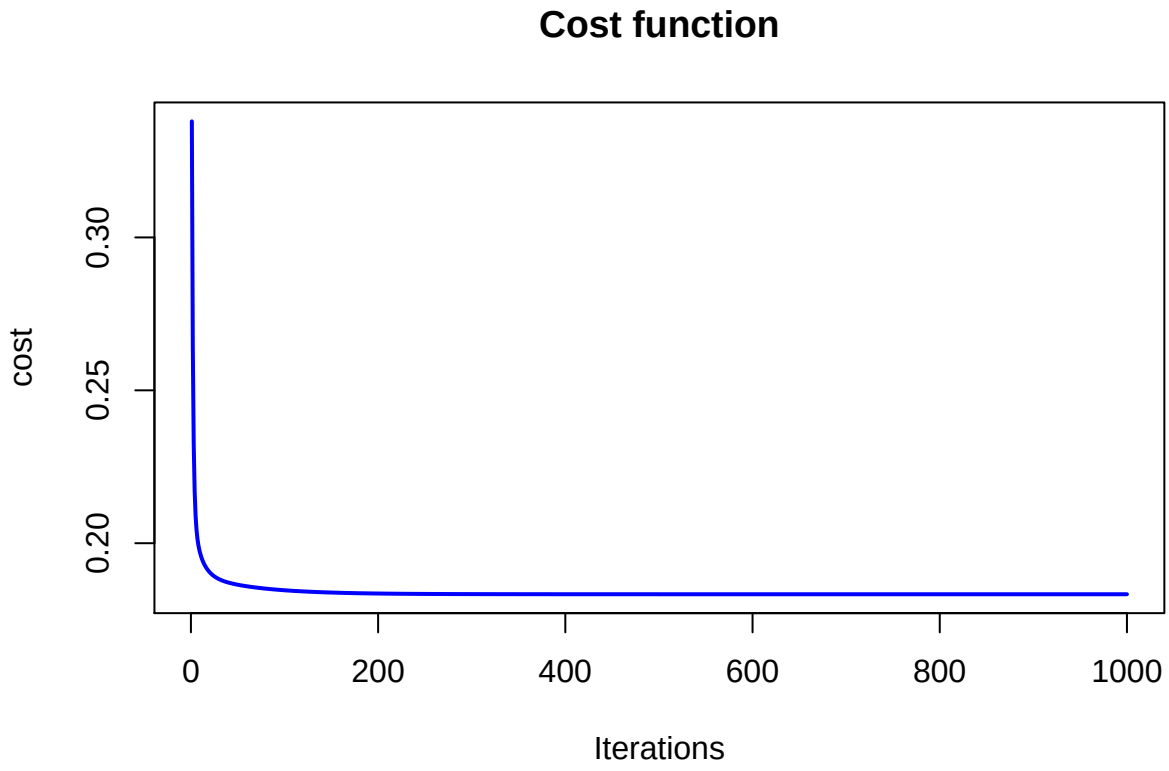
```
## [1] 2.092632
```

```
# correlation between actual and predicted values of strength
print(cor(Y_validation, y_validation_predict))
```

```
##           [,1]
## [1,] 0.7566735
```

```
# When alpha = 0.3, we have to run about 1000 iterations to reach optimal value
alpha_ = 0.3
niter = 1000
J_history <- c()
J_history[1:niter] = 0
# Fit the multiple linear regression model using the gradient descent algorithm and the training set.
#Try out different learning rates: alpha = 0.01,0.1,0.3,0.5 and compare the speed of convergence
# by plotting the cost function. Determine the number of iterations needed.
b0 <- as.matrix(c(0,0,0,0,0,0,0,0,0))
brun <- b0
m = length(Y)

for (iter in 1:niter){
  brun = b0 + alpha_*t(t(Y-X%*%b0)%*%X)/m
  J_history[iter] <- Jcost(X, Y, brun)
  #J_history[iter] <<- Jcost(X,Y,brun)
  #print(head(J_history[iter]))
  b0 <<- brun # making b0 a global variable
}

plot(J_history, type='line', col='blue', lwd=2, main='Cost function', ylab='cost', xlab='Iterations')
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```




```
#compare with normal equations

b1 <- solve(t(X)%*%X)%*%(t(X)%*%Y)
print(head(b0))
```

```
##                     [,1]
## x0            0.02443156
## cement        0.76730388
## slag          0.54070060
## ash           0.34030542
## water        -0.21900379
## superplastic  0.09883991
```

```
print(head(b1))
```

```
##                     [,1]
## x0            0.02443147
## cement        0.76736711
## slag          0.54076185
## ash           0.34035926
## water        -0.21894856
## superplastic  0.09884737
```

```
y_predict <- X%*%b0

print(head(y_predict))
```

```
##           [,1]
```

```
## 11   0.26402996
## 820 -0.79445401
## 863 -0.07620676
## 882 -0.03216254
## 814 -0.16214175
## 707 -0.28182830
```

```
y_validation_predict <- X_validation%*%b0
# Evaluate the model performance
# ME, RMSE, MAE, MPE, MAPE

Metrics::rmse(Y_validation, y_validation_predict)
```

```
## [1] 0.6440735
```

```
Metrics::mae(Y_validation, y_validation_predict)
```

```
## [1] 0.5129229
```

```
Metrics::mape(Y_validation, y_validation_predict)
```

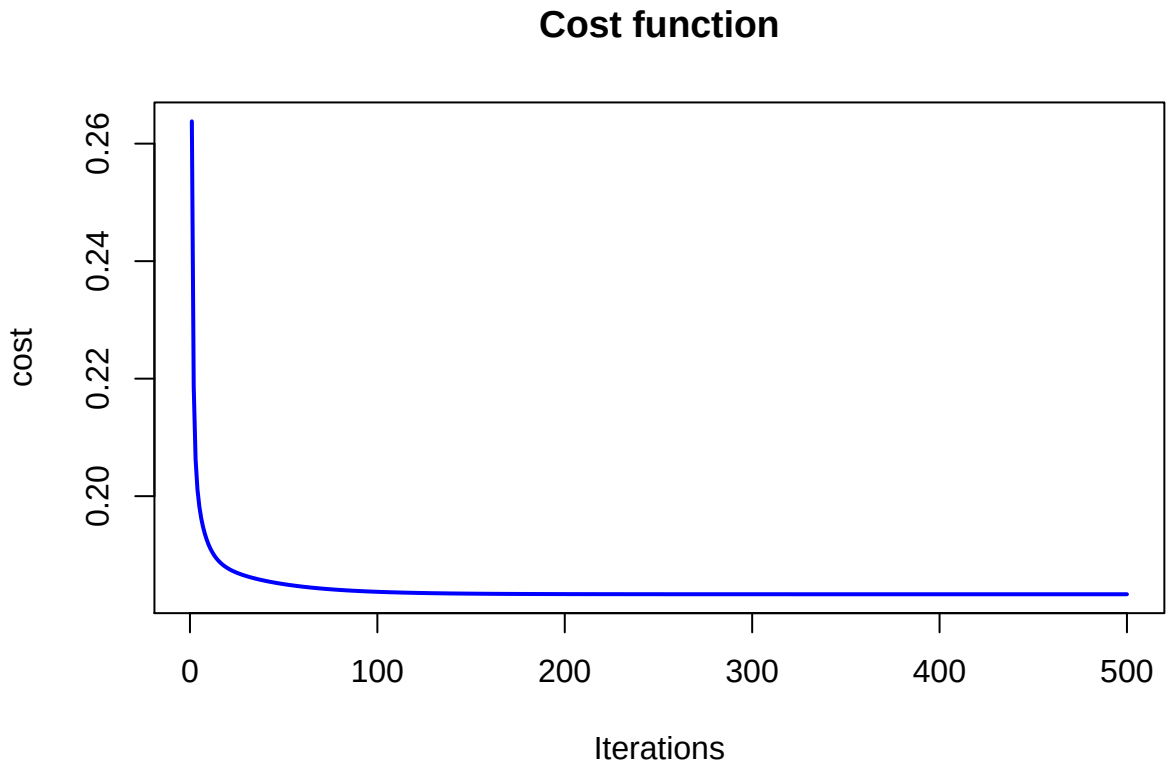
```
## [1] 2.091949
```

```
# correlation between actual and predicted values of strength
print(cor(Y_validation, y_validation_predict))
```

```
##         [,1]
## [1,] 0.75668
```

```
# When alpha = 0.5, we have to run about 500 iterations to reach optimal value
alpha_ = 0.5
niter = 500
J_history <- c()
J_history[1:niter] = 0
# Fit the multiple linear regression model using the gradient descent algorithm and the training set.
#Try out different learning rates: alpha = 0.01,0.1,0.3,0.5 and compare the speed of convergence
# by plotting the cost function. Determine the number of iterations needed.
b0 <- as.matrix(c(0,0,0,0,0,0,0,0,0,0))
brun <- b0
m = length(Y)

for (iter in 1:niter){
  brun = b0 + alpha_*t(t(Y-X%*%b0)%*%X)/m
  J_history[iter] <- Jcost(X, Y, brun)
  #J_history[iter] <<- Jcost(X,Y,brun)
  #print(head(J_history[iter]))
  b0 <<- brun # making b0 a global variable
}

plot(J_history, type='line', col='blue', lwd=2, main='Cost function', ylab='cost', xlab='Iterations')
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```




```
#compare with normal equations

b1 <- solve(t(X)%*%X)%*%(t(X)%*%Y)
print(head(b0))
```

```
##                     [,1]
## x0            0.02443182
## cement        0.76711048
## slag          0.54051325
## ash           0.34014076
## water        -0.21917269
## superplastic  0.09881708
```

```
print(head(b1))
```

```
##                     [,1]
## x0            0.02443147
## cement        0.76736711
## slag          0.54076185
## ash           0.34035926
## water        -0.21894856
## superplastic  0.09884737
```

```
y_predict <- X%*%b0

print(head(y_predict))
```

```
##           [,1]
```

```
## 11   0.26413180
## 820 -0.79441054
## 863 -0.07623340
## 882 -0.03219608
## 814 -0.16212536
## 707 -0.28181729
```

```
y_validation_predict <- X_validation%*%b0

# Evaluate the model performance
# ME, RMSE, MAE, MPE, MAPE

Metrics::rmse(Y_validation, y_validation_predict)
```

```
## [1] 0.6440733
```

```
Metrics::mae(Y_validation, y_validation_predict)
```

```
## [1] 0.5129257
```

```
Metrics::mape(Y_validation, y_validation_predict)
```

```
## [1] 2.092076
```

```
# correlation between actual and predicted values of strength
print(cor(Y_validation, y_validation_predict))
```

```
##           [,1]
## [1,] 0.7566788
```

```
View(y_validation_predict)
describe(test_concrete$strength)
```

```
##    vars   n  mean    sd median trimmed   mad  min   max range skew kurtosis  se
## X1    1 412 35.34 16.31  34.73   34.59 16.82 2.33 81.75 79.42 0.36    -0.37 0.8
```

```
y_validation_predict_dstd <- data.frame(y_validation_predict * sd(test_concrete$strength) + mean(test_c

y_validation_predict_dstd
```

```
##      y_validation_predict...sd.test_concrete.strength....mean.test_concrete.strength.
## 6                                                                          23.984269
## 7                                                                          23.914958
## 8                                                                          31.010527
## 12                                                                         15.139558
## 15                                                                         39.235676
## 16                                                                         36.041334
## 19                                                                         25.034800
## 25                                                                         25.893120
## 27                                                                         26.889326
## 28                                                                         64.345542
## 29                                                                         23.932505
## 35                                                                         33.157243
## 36                                                                         32.299372
## 38                                                                         38.114098
## 39                                                                         24.011364
## 46                                                                         53.897841
## 53                                                                         35.659881
```

```
## 54     30.264131
## 63     34.511802
## 64     58.844622
## 66     27.724587
## 70     50.833866
## 71     19.448975
## 74     25.604932
## 76     40.877506
## 77     31.367488
## 78     30.836743
## 81     28.434883
## 82     24.335864
## 85     12.747081
## 90     34.578059
## 94     47.788973
## 98     51.544737
## 100    44.455463
## 101    32.458568
## 102    54.281019
## 106    28.564003
## 107    17.624264
## 108    19.002598
## 109    16.793564
## 110    34.254572
## 111    54.194341
## 118    29.841243
## 119    26.112945
## 120    35.677377
## 124    35.950675
## 125    57.121825
## 127    23.393224
## 129    53.294295
## 133    12.094350
## 134    54.038802
## 136    54.626334
## 138    51.237837
## 142    26.001507
## 143    35.577523
## 145    37.345469
## 146    24.485600
## 147    40.057364
## 148    40.873201
## 153    39.400200
## 154    30.958855
## 156    27.311362
## 158    61.678587
## 164    51.078193
## 165    31.291054
## 166    30.946882
## 167    53.841253
## 168    39.029180
## 169    31.244454
## 170    39.567908
## 174    39.415193
```

```
## 178 60.163006
## 179 33.952648
## 183 56.894112
## 187 28.135158
## 188 16.029938
## 192 44.153772
## 193 47.899924
## 194 34.615496
## 195 23.918350
## 196 21.463027
## 199 58.442524
## 203 30.950787
## 204 37.557956
## 205 22.033452
## 206 16.129101
## 208 35.433914
## 214 27.875183
## 218 22.821505
## 220 25.044981
## 223 31.982839
## 227 37.034910
## 229 30.577427
## 231  9.890915
## 232 32.803215
## 239 33.056182
## 241 39.983925
## 245 37.716272
## 246 61.217902
## 249 14.585284
## 250 20.084120
## 253 26.109970
## 255 27.050449
## 256 19.951096
## 257 75.706056
## 261 37.998104
## 265 46.019529
## 267 44.379315
## 268 23.721741
## 269 18.684421
## 271 70.871304
## 278 36.786350
## 279 54.035174
## 282 39.881230
## 283 31.779844
## 288 32.140711
## 289 30.773684
## 291 16.897739
## 293 76.660259
## 294 48.511891
## 295 30.625104
## 297 29.939866
## 298 59.167045
## 299 54.035174
## 301 49.183490
```

```
## 303 38.100722
## 305 26.106593
## 306 21.630439
## 308 51.269961
## 309 33.944177
## 310 28.135383
## 312 18.450367
## 313 32.363596
## 315 20.085489
## 322 26.827655
## 323 50.684459
## 324 63.406036
## 330 19.625203
## 331 32.114053
## 332 30.468625
## 337 43.380120
## 338 58.660806
## 339 30.419987
## 341 33.980128
## 342 36.409819
## 349 38.224936
## 350 17.625097
## 351 25.332763
## 352 33.021594
## 353 34.569721
## 354 36.856413
## 356 38.812833
## 357 33.004298
## 362 58.116953
## 364 32.402633
## 365 47.145353
## 366 22.122736
## 368 16.944651
## 369 25.761994
## 372 29.557471
## 374 50.513897
## 375 19.218747
## 376 17.535543
## 378 27.503271
## 379 17.557516
## 383 25.933172
## 384 46.174013
## 390 28.754222
## 396 40.449686
## 397 54.005166
## 401 50.907534
## 405 23.397443
## 406 34.918016
## 412 45.942896
## 415 39.186982
## 417 32.057641
## 419 57.895183
## 422 19.898797
## 423 23.334525
```

```
## 429 37.507140
## 430 23.446305
## 435 36.672141
## 444 30.682852
## 446 36.410901
## 448 55.714217
## 455 24.970224
## 459 33.472156
## 460 33.037844
## 462 41.525268
## 463 46.424134
## 465 33.407311
## 469 18.490366
## 473 36.545288
## 476 31.427275
## 479 26.323596
## 480 43.005476
## 483 64.345542
## 485 66.130669
## 490 60.244827
## 491 48.809532
## 494 61.217902
## 495 36.825302
## 498 34.444172
## 503 39.354384
## 504 51.376187
## 507 46.404664
## 514 25.763083
## 517 32.457587
## 518 60.244827
## 521 37.124293
## 523 53.841253
## 525 47.899788
## 526 53.836616
## 530 54.027088
## 535 37.850292
## 536 21.876925
## 539 61.616302
## 541 54.616667
## 549 33.735533
## 552 37.849047
## 553 25.920036
## 555 25.141731
## 558 32.267016
## 559 27.010784
## 563 44.892115
## 565 45.333795
## 568 23.717886
## 570 58.726380
## 571 45.978609
## 573 50.424762
## 574 23.224495
## 576 65.180427
## 578 56.964255
```

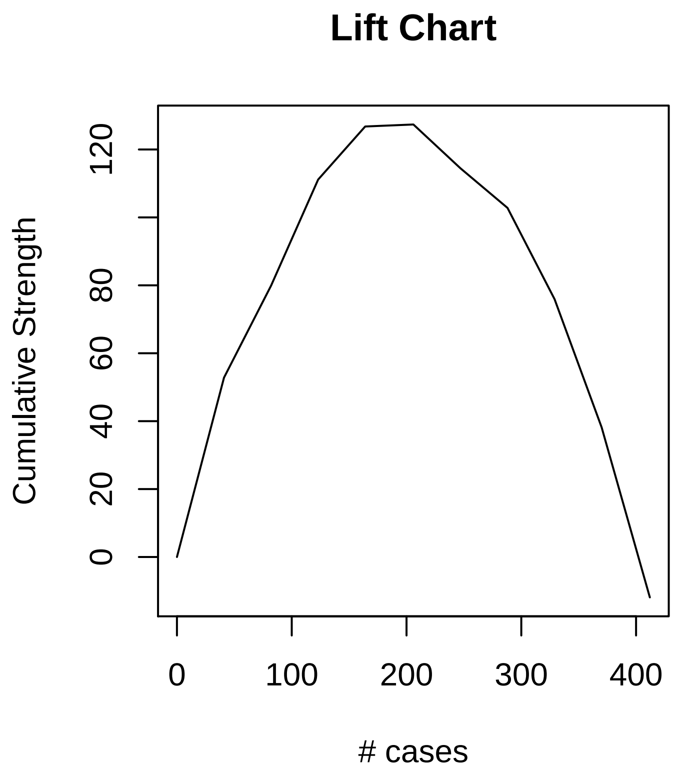
```
## 585 61.455218
## 586 41.034896
## 588 35.059297
## 591 45.753419
## 593 14.501728
## 596 22.156682
## 598 25.477683
## 603 27.989208
## 607 33.832742
## 616 54.550532
## 626 38.375621
## 627 29.642180
## 632 50.912171
## 633 47.978715
## 634 20.384222
## 639 19.800746
## 640 51.380824
## 644 31.268129
## 645 32.929435
## 650 36.952149
## 653 34.092011
## 655 51.874162
## 658 35.883751
## 660 16.804720
## 664 48.885334
## 667 22.920199
## 669 26.794597
## 671 33.521827
## 674 12.124855
## 675 22.953403
## 677 55.338712
## 678 28.086578
## 683 17.483494
## 685 29.008148
## 692 26.686349
## 694 22.990954
## 695 30.804264
## 698 34.553134
## 699 15.989939
## 705 23.759443
## 708 32.349886
## 711 31.152507
## 713 30.941220
## 715 28.624879
## 716 30.567618
## 717 24.444015
## 718 27.757501
## 721 68.630801
## 725 33.751260
## 729 25.367828
## 735 35.732297
## 736 17.778719
## 737 12.819997
## 738 41.305804
```

```
## 740 21.694090
## 741 18.721445
## 742 59.091190
## 743 24.092397
## 744 26.041159
## 746 44.020218
## 748 25.445938
## 754 51.380824
## 757 32.015283
## 759 28.345975
## 760 26.145373
## 761 36.450909
## 768 39.928253
## 769 27.906366
## 772 30.686583
## 773 23.837397
## 780 37.314319
## 781 35.486205
## 782 24.126899
## 784 11.929531
## 785 47.312204
## 786 35.838036
## 788 28.860308
## 789 50.695996
## 791 27.519727
## 792 32.599140
## 798 20.319952
## 799 19.903292
## 802 44.520214
## 804 31.111929
## 805 48.957697
## 807 28.313400
## 808 29.146778
## 818 26.804791
## 819 19.265149
## 823 29.450604
## 826 33.511080
## 835 56.019875
## 837 39.240020
## 838 31.254635
## 841 19.536652
## 843 25.852972
## 844 21.484585
## 845 45.709507
## 846 48.053468
## 848 21.490164
## 849 40.455414
## 850 25.464519
## 852 31.509431
## 855 47.688187
## 857 51.024510
## 859 49.308956
## 861 40.077549
## 862 40.729016
```

```
## 867 31.098769
## 868 39.012868
## 870 50.669015
## 871 14.611313
## 872 23.432691
## 873 33.913438
## 875 50.805471
## 876 43.387170
## 877 15.890620
## 878 33.661141
## 884 23.088431
## 889 34.156311
## 890 51.643919
## 891 23.355230
## 894 33.677790
## 896 25.614347
## 898 48.350438
## 899 30.296915
## 900 38.900985
## 902 70.297556
## 909 52.634968
## 914 57.285738
## 915 36.348093
## 917 28.758254
## 920 61.818406
## 922 30.184322
## 925 18.910355
## 926 32.302711
## 928 23.924628
## 930 36.774970
## 933 40.694596
## 934 51.376187
## 941 21.962162
## 944 50.907534
## 945 14.858613
## 947 34.179995
## 949 28.408973
## 950 48.156840
## 953 24.889220
## 954 40.787712
## 956 66.349891
## 957 46.232866
## 961 34.388581
## 962 29.450227
## 964 49.524773
## 969 25.206076
## 971 29.122201
## 972 45.972533
## 973 30.825257
## 979 39.453147
## 981 35.170489
## 984 48.308402
## 986 42.252833
## 992 61.115839
```

```
## 993                                                                        56.964255
## 996                                                                        60.442630
## 1000                                                                       33.754339
## 1001                                                                       32.855002
## 1010                                                                       27.641793
## 1011                                                                       24.473198
## 1012                                                                       27.788279
## 1014                                                                       50.746516
## 1015                                                                       30.705726
## 1016                                                                       30.965233
## 1017                                                                       22.676918
## 1018                                                                       42.316646
## 1021                                                                       14.761048
## 1023                                                                       19.002773
## 1025                                                                       33.155376
## 1029                                                                       48.645417
## 1030                                                                       49.366639
```

```r
library(gains)
gain <- gains(test_concrete_std$strength[!is.na(y_validation_predict_dstd)], y_validation_predict[!is.na
options(scipen=999)
strength <- test_concrete_std$strength[!is.na(test_concrete_std$strength)]

par(pty="s")
plot(c(0,gain$cume.pct.of.total*sum(strength))~c(0,gain$cume.obs),
     xlab = "# cases", ylab = "Cumulative Strength", main = "Lift Chart", type = "l")
```



Problem 2

Problem 2 (25points): Multiple Linear Regression Model for Concrete Slump Test Data • Read the included research article "Modeling Slump Flow Concrete". It is sufficient to consider "Slump Flow" as the response variable in this problem just as in the included article. • Create a scatterplot matrix of "Concrete Slump Test Data" and select an initial set of predictor variables. • Build a few potential regression models using "Concrete Slump Test Data" • Perform regression diagnostics using both typical approach and enhanced approach • Identify unusual observations and take corrective measures • Select the best regression model • Fine tune the selection of predictor variables • Interpret the prediction results

```
library(car)
```

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

```
##
## Attaching package: 'car'
```

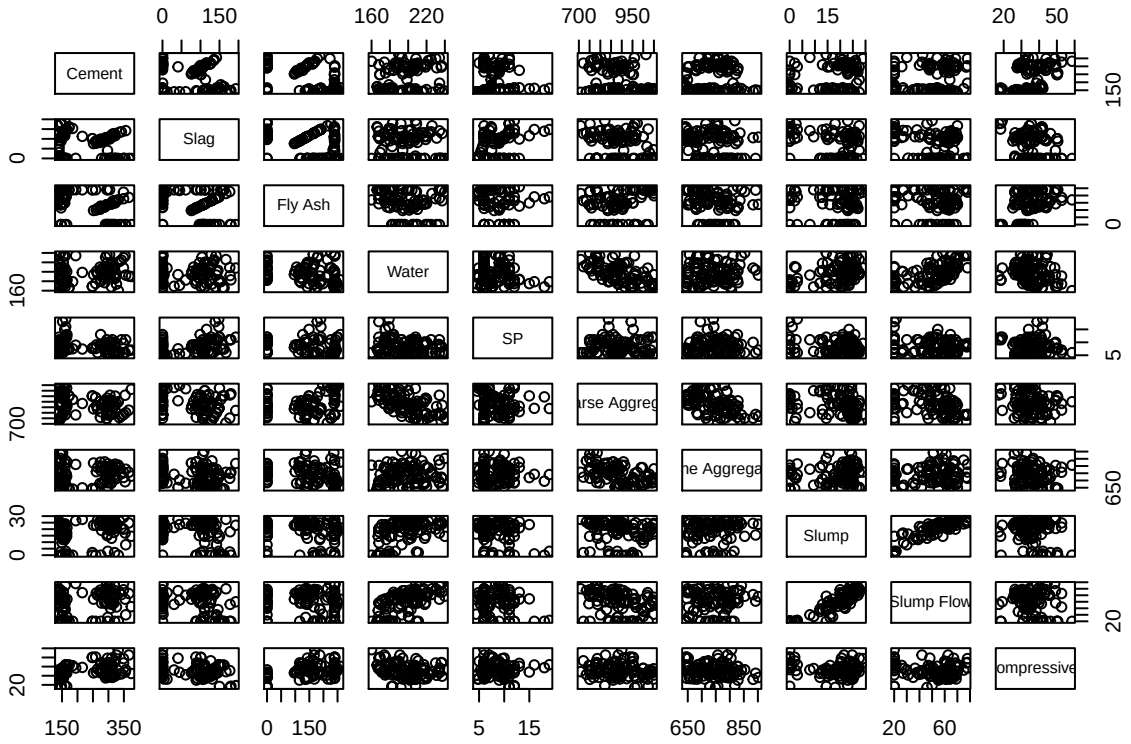
```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(readxl)
cstd <- read_excel("C:/Users/abhil/Downloads/Concrete Slump Test Data(1).xlsx")
View(cstd)
colnames(cstd)
```

```
##  [1] "No"                          "Cement"
##  [3] "Slag"                        "Fly Ash"
##  [5] "Water"                       "SP"
##  [7] "Coarse Aggregate"            "Fine Aggregate"
##  [9] "Slump"                       "Slump Flow"
## [11] "28-day Compressive Strength"
```

```
#scatterplotMatrix(~Cement+Slag+`Fly Ash`+Water+SP+`Coarse Aggregate`+`Fine Aggregate`+Slump+`Slump Flo

pairs(~Cement+Slag+`Fly Ash`+Water+SP+`Coarse Aggregate`+`Fine Aggregate`+Slump+`Slump Flow`+`28-day Co
```

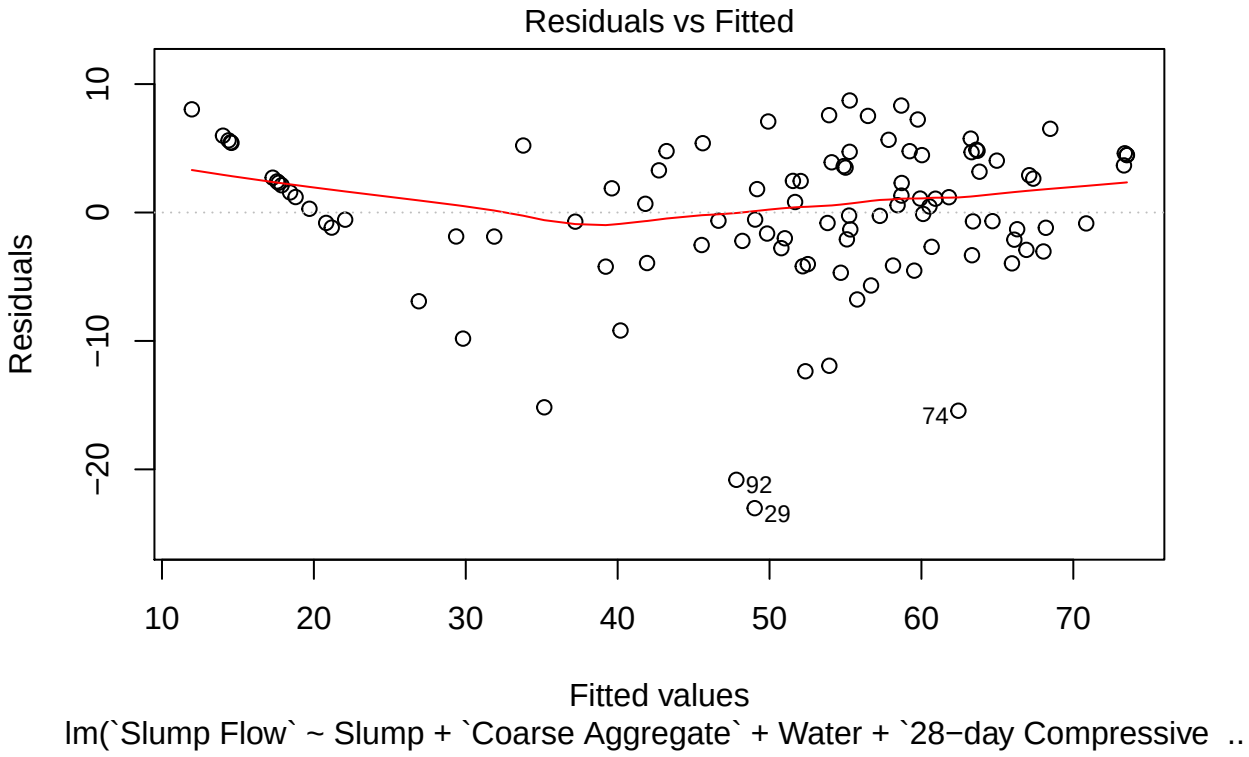


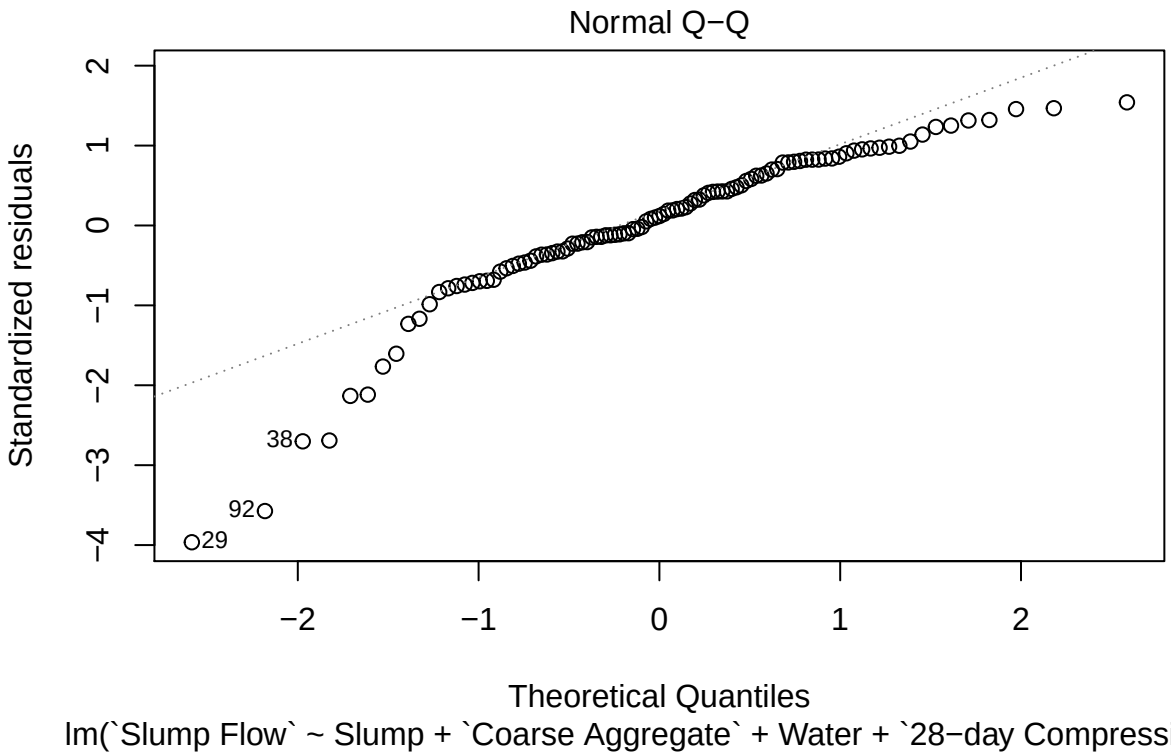

```
# Model 1

fit_1 <- lm(`Slump Flow` ~ Slump + `Coarse Aggregate` + Water + `28-day Compressive Strength`, data = cs
summary(fit_1)
```

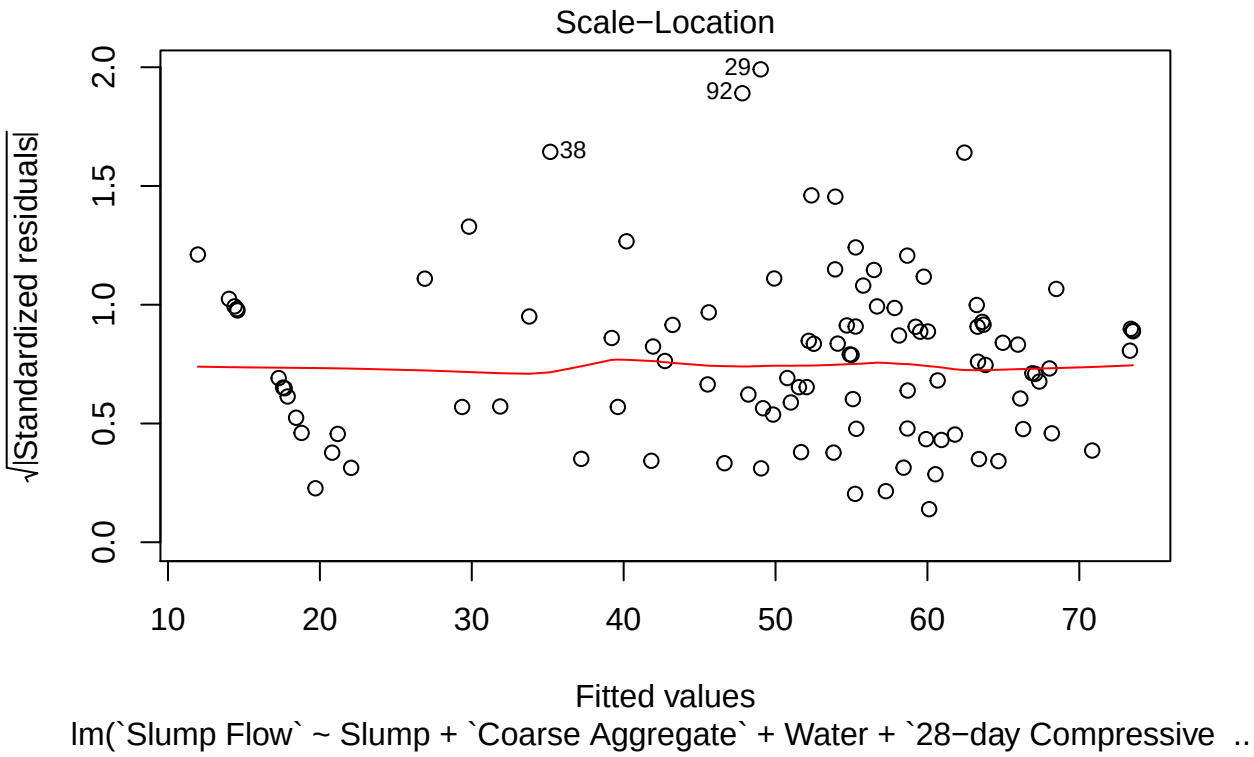
```
##
## Call:
## lm(formula = `Slump Flow` ~ Slump + `Coarse Aggregate` + Water +
##     `28-day Compressive Strength`, data = cstd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.0211  -2.1639   0.6763   4.2524   8.7220
##
## Coefficients:
##                                  Estimate Std. Error t value
## (Intercept)                    -52.674918  16.342213  -3.223
## Slump                            1.597223   0.075834  21.062
## `Coarse Aggregate`               0.007931   0.009035   0.878
## Water                            0.278970   0.043551   6.406
## `28-day Compressive Strength`    0.317515   0.084006   3.780
##                                           Pr(>|t|)
## (Intercept)                                0.00172 **
## Slump                          < 0.0000000000000002 ***
## `Coarse Aggregate`                         0.38217
## Water                                 0.00000000522 ***
```

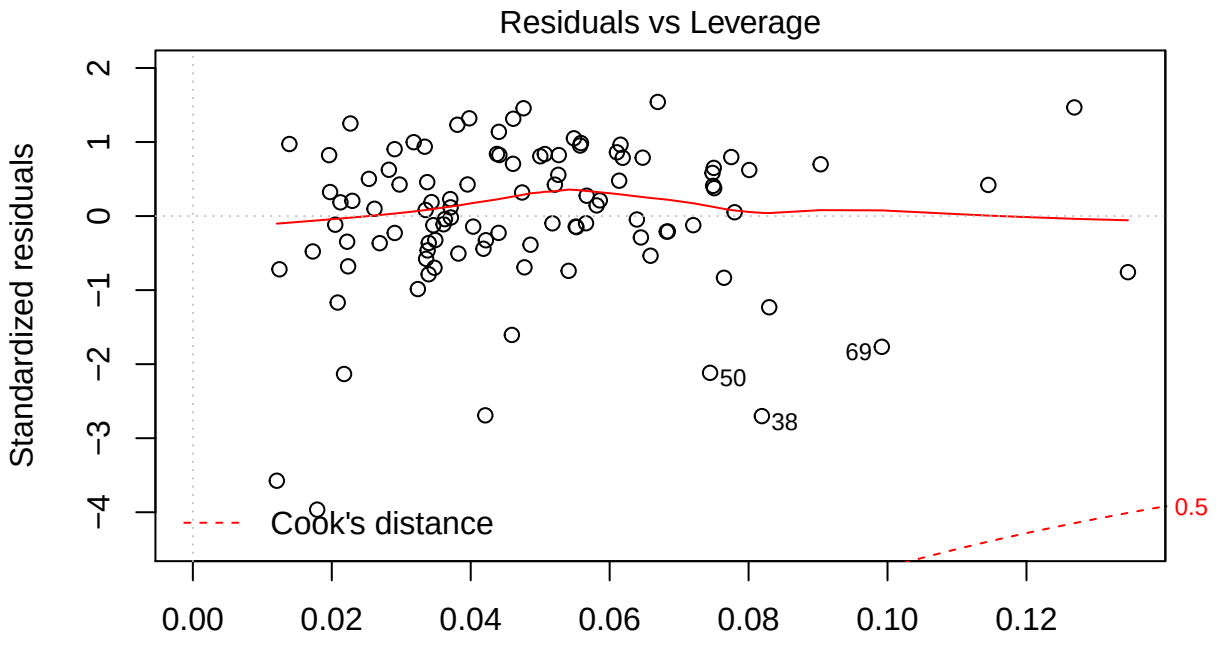
```
## `28-day Compressive Strength`              0.00027 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.859 on 98 degrees of freedom
## Multiple R-squared:  0.8931, Adjusted R-squared:  0.8888
## F-statistic: 204.7 on 4 and 98 DF,  p-value: < 0.00000000000000022
```

```
plot(fit_1)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(`Slump Flow` ~ Slump + `Coarse Aggregate` + Water + `28−day Compressive ..
38
92
29
2
1
0
−1
−2
−3
−4
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
29
92
38
10
20
30
40
50
60
70
Fitted values
lm(`Slump Flow` ~ Slump + `Coarse Aggregate` + Water + `28−day Compressive ..

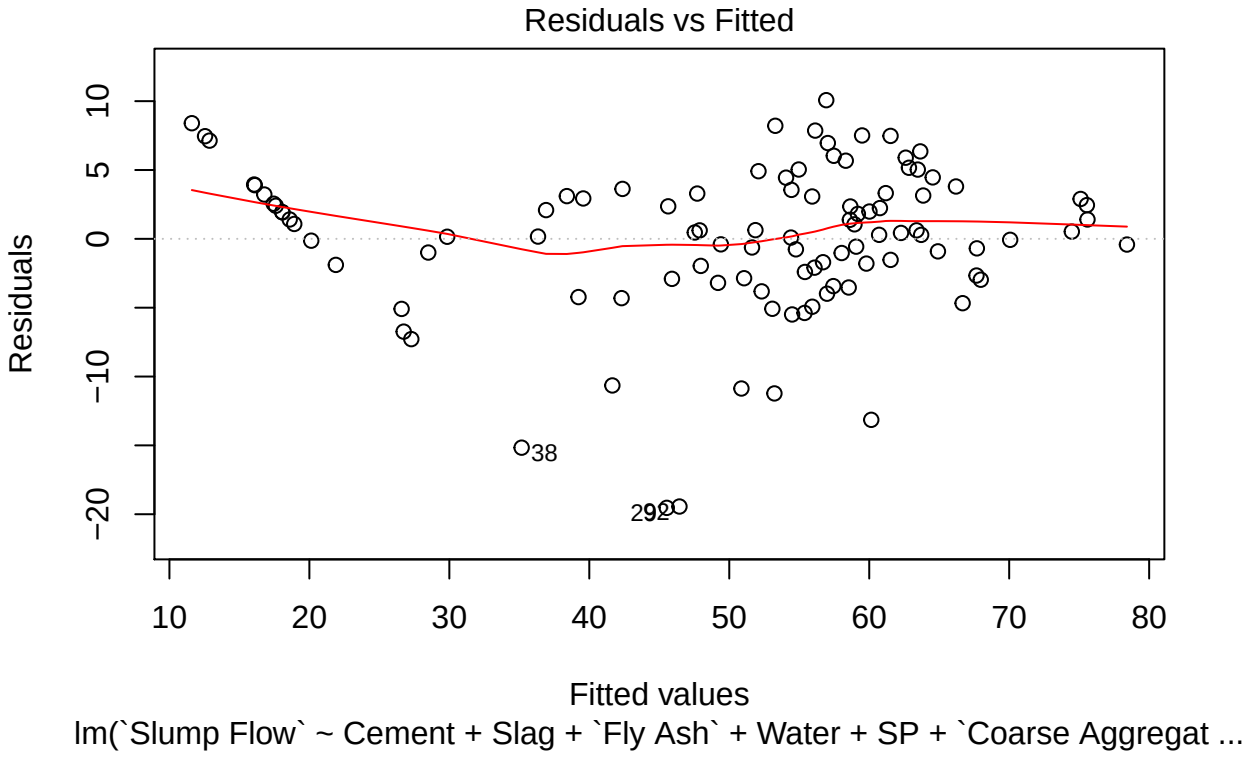


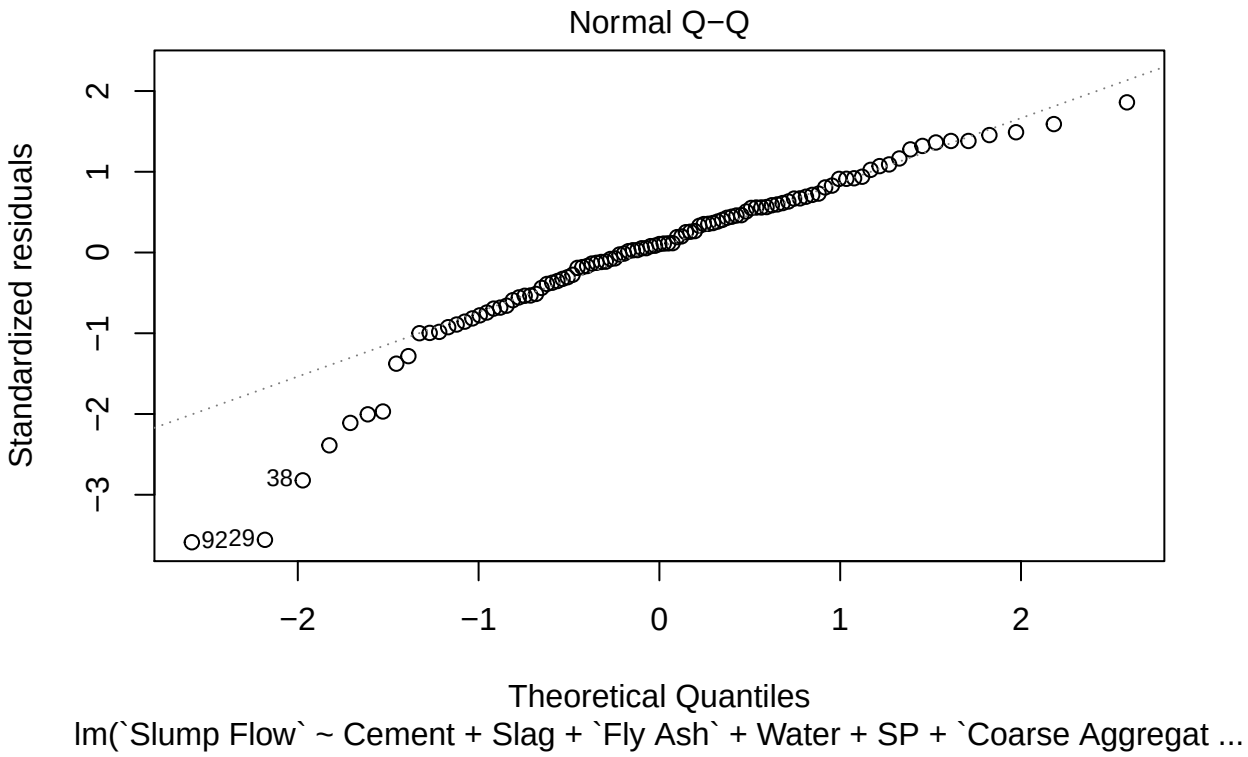

```
# Model 2

fit_2 <- lm(`Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` + `Fine Aggregat
summary(fit_2)
```

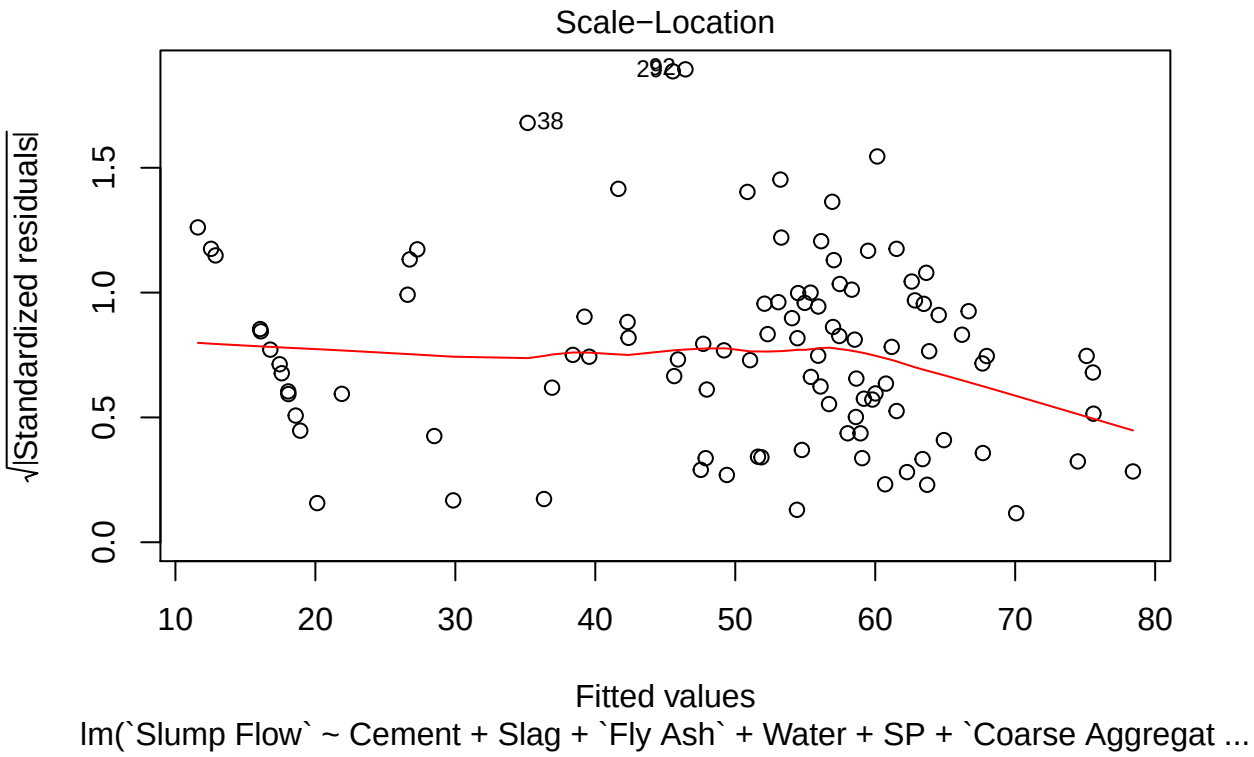
```
##
## Call:
## lm(formula = `Slump Flow` ~ Cement + Slag + `Fly Ash` + Water +
##     SP + `Coarse Aggregate` + `Fine Aggregate` + Slump + `28-day Compressive Strength`,
##     data = cstd)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.536  -2.534   0.525   3.254  10.071
##
## Coefficients:
##                                 Estimate  Std. Error t value
## (Intercept)                  -175.303251  158.104676  -1.109
## Cement                          0.009729    0.051860   0.188
## Slag                            0.028296    0.069614   0.406
## `Fly Ash`                       0.028562    0.051881   0.551
## Water                           0.425401    0.164688   2.583
## SP                              0.543975    0.294332   1.848
## `Coarse Aggregate`              0.051510    0.061101   0.843
## `Fine Aggregate`                0.050360    0.063425   0.794
## Slump                           1.588185    0.082020  19.363
```

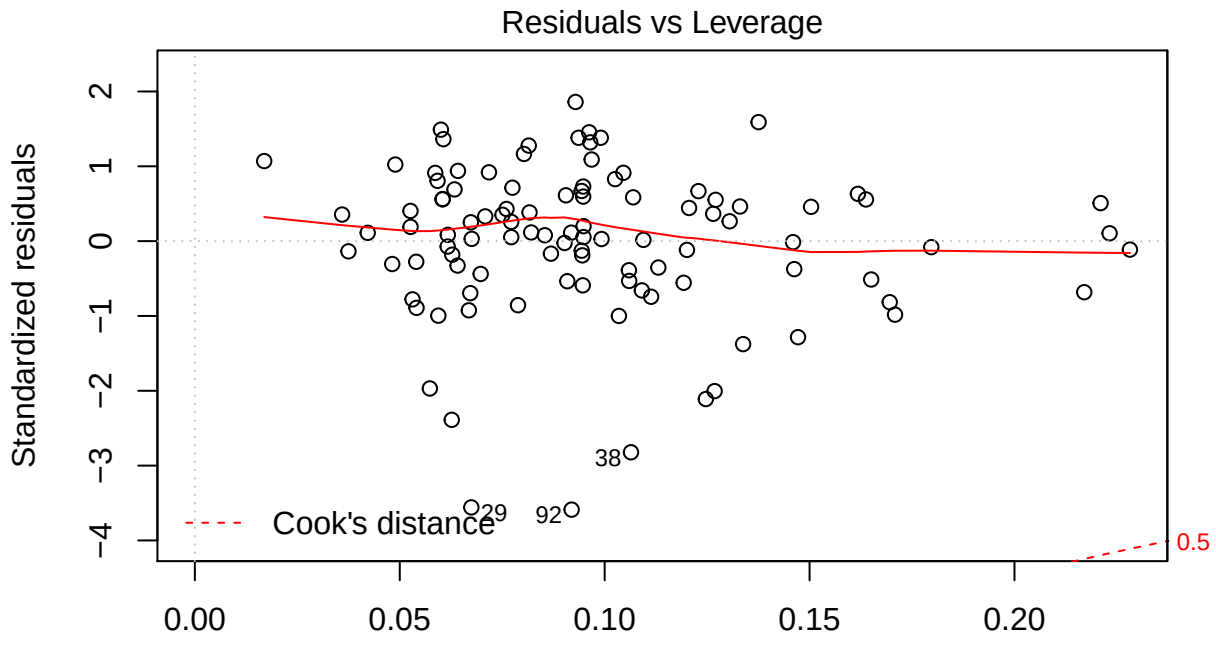
```
## `28-day Compressive Strength`    0.450866    0.234524   1.922
##                                          Pr(>|t|)
## (Intercept)                                0.2704
## Cement                                     0.8516
## Slag                                       0.6853
## `Fly Ash`                                  0.5833
## Water                                      0.0114 *
## SP                                         0.0678 .
## `Coarse Aggregate`                         0.4014
## `Fine Aggregate`                           0.4292
## Slump                          <0.0000000000000002 ***
## `28-day Compressive Strength`              0.0576 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.685 on 93 degrees of freedom
## Multiple R-squared:  0.9045, Adjusted R-squared:  0.8953
## F-statistic: 97.89 on 9 and 93 DF,  p-value: < 0.00000000000000022
```

```
plot(fit_2)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(`Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregat ...
38
92
29
−3
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
10
20
30
40
50
60
70
80
Fitted values
lm(`Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregat ...



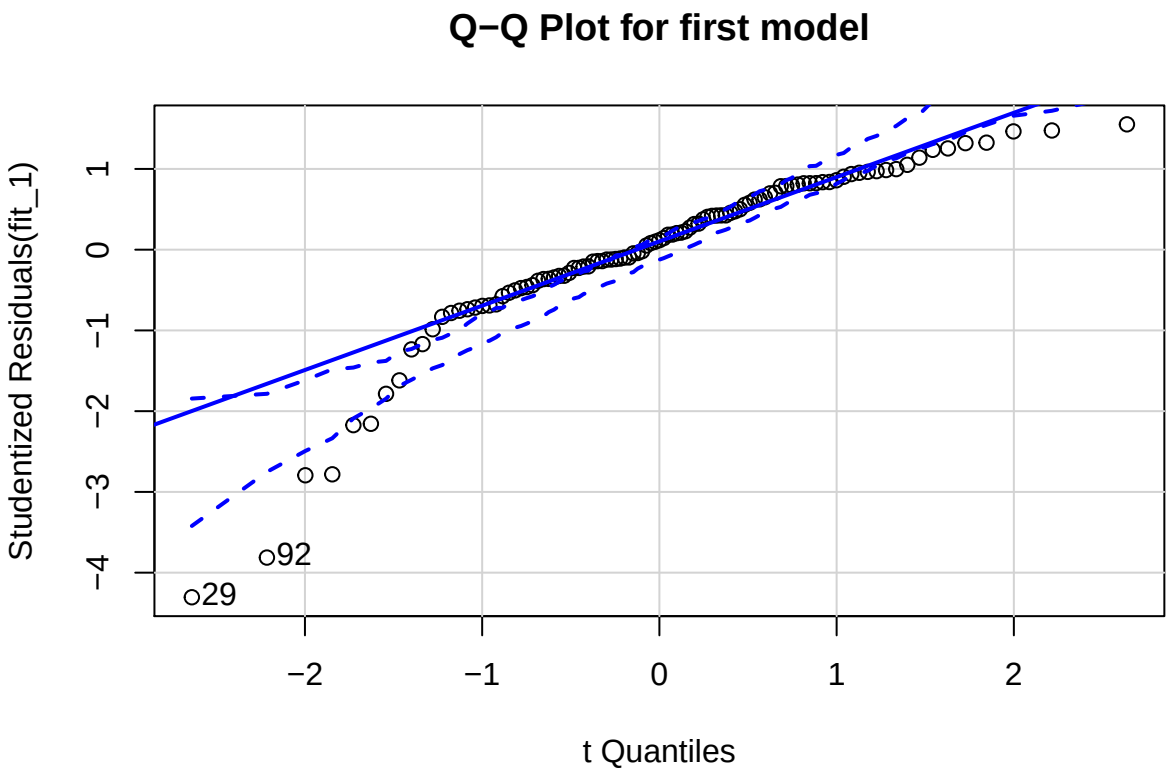

```
# Perform regression diagnostics using both typical approach and enhanced approach
confint(fit_1)
```

```
##                                    2.5 %      97.5 %
## (Intercept)                  -85.105508526 -20.2443269
## Slump                          1.446733853   1.7477127
## `Coarse Aggregate`            -0.009998071   0.0258606
## Water                          0.192543682   0.3653962
## `28-day Compressive Strength`  0.150808389   0.4842209
```

```
confint(fit_2)
```

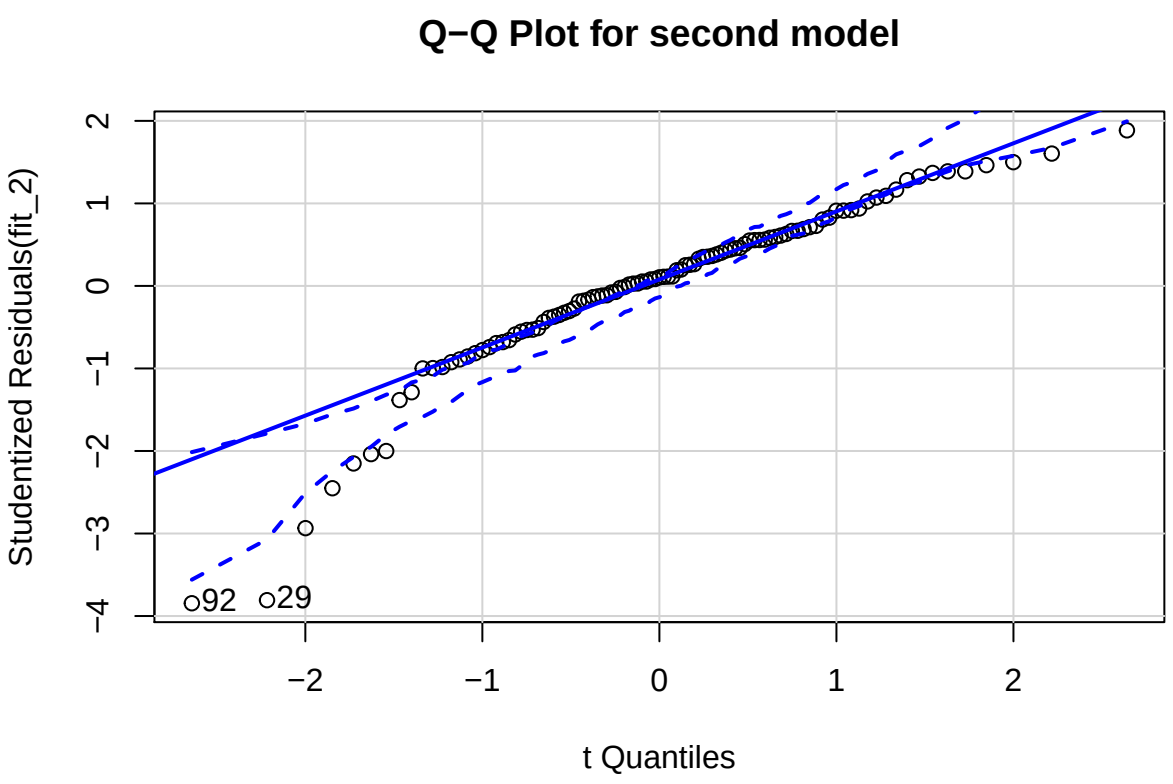
```
##                                    2.5 %      97.5 %
## (Intercept)                  -489.26780249 138.6613012
## Cement                         -0.09325374   0.1127127
## Slag                           -0.10994428   0.1665364
## `Fly Ash`                      -0.07446362   0.1315886
## Water                           0.09836267   0.7524397
## SP                             -0.04051119   1.1284608
## `Coarse Aggregate`             -0.06982321   0.1728439
## `Fine Aggregate`               -0.07558950   0.1763089
## Slump                           1.42530952   1.7510606
## `28-day Compressive Strength`  -0.01485154   0.9165834
```

```
#library(car)
# Enhanced Approach
qqPlot(fit_1, labels = row.names(cstd), id.method = "identify", simulate = TRUE, main = "Q-Q Plot for fi
```




```
## [1] 29 92
```

```
qqPlot(fit_2, labels = row.names(cstd), id.method = "identify", simulate = TRUE, main = "Q-Q Plot for s
```




```
## [1] 29 92
```

```
cstd[92,]
```

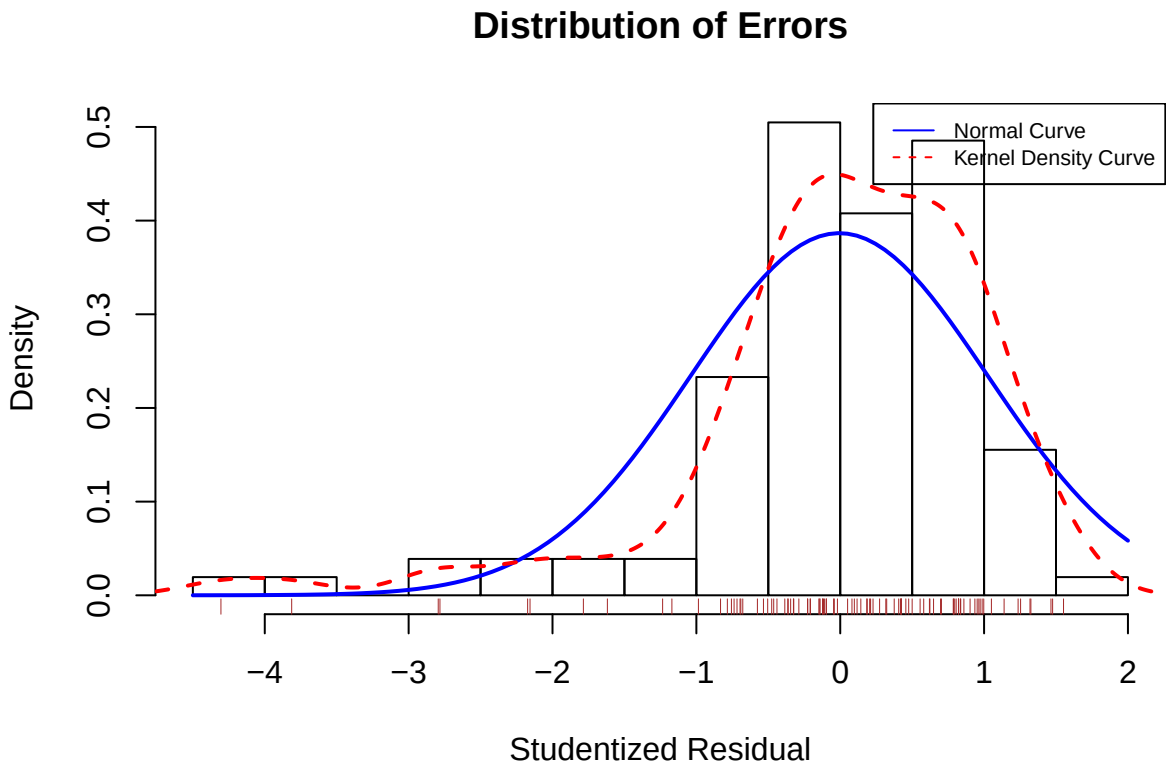
```
## # A tibble: 1 x 11
##      No Cement  Slag `Fly Ash` Water    SP `Coarse Aggrega... `Fine Aggregate`
##   <dbl>  <dbl> <dbl>     <dbl> <dbl> <dbl>             <dbl>             <dbl>
## 1    92   165.  143.      238.  200.   7.1              883.              653.
## # ... with 3 more variables: Slump <dbl>, `Slump Flow` <dbl>, `28-day Compressive
## #   Strength` <dbl>
```

```
cstd[29,]
```

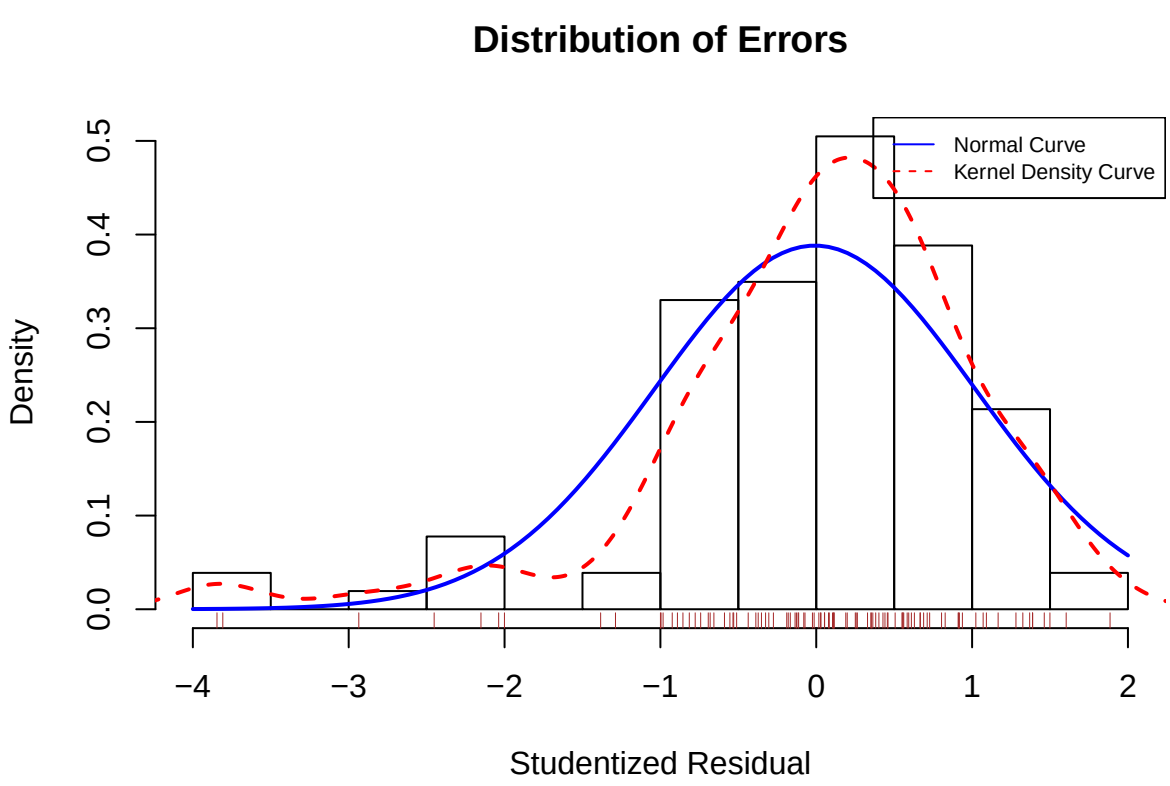
```
## # A tibble: 1 x 11
##      No Cement  Slag `Fly Ash` Water    SP `Coarse Aggrega... `Fine Aggregate`
##   <dbl>  <dbl> <dbl>     <dbl> <dbl> <dbl>             <dbl>             <dbl>
## 1    29    298   107       137   201     6               878               655
## # ... with 3 more variables: Slump <dbl>, `Slump Flow` <dbl>, `28-day Compressive
## #   Strength` <dbl>
```

```
residPlot = function(fit, nbreaks = 10){
  z = rstudent(fit)
  hist(z, breaks = nbreaks, freq = FALSE, xlab = "Studentized Residual", main = "Distribution of Errors
  rug(jitter(z), col="brown")
  curve(dnorm(x, mean=mean(z), sd=sd(z)), add=TRUE, col="blue", lwd=2)
  lines(density(z)$x, density(z)$y, col="red", lwd=2, lty=2)
  legend("topright", legend = c("Normal Curve", "Kernel Density Curve"), lty = 1:2, col=c("blue", "red"
}
```

```
residPlot(fit_1)
```




```
residPlot(fit_2)
```



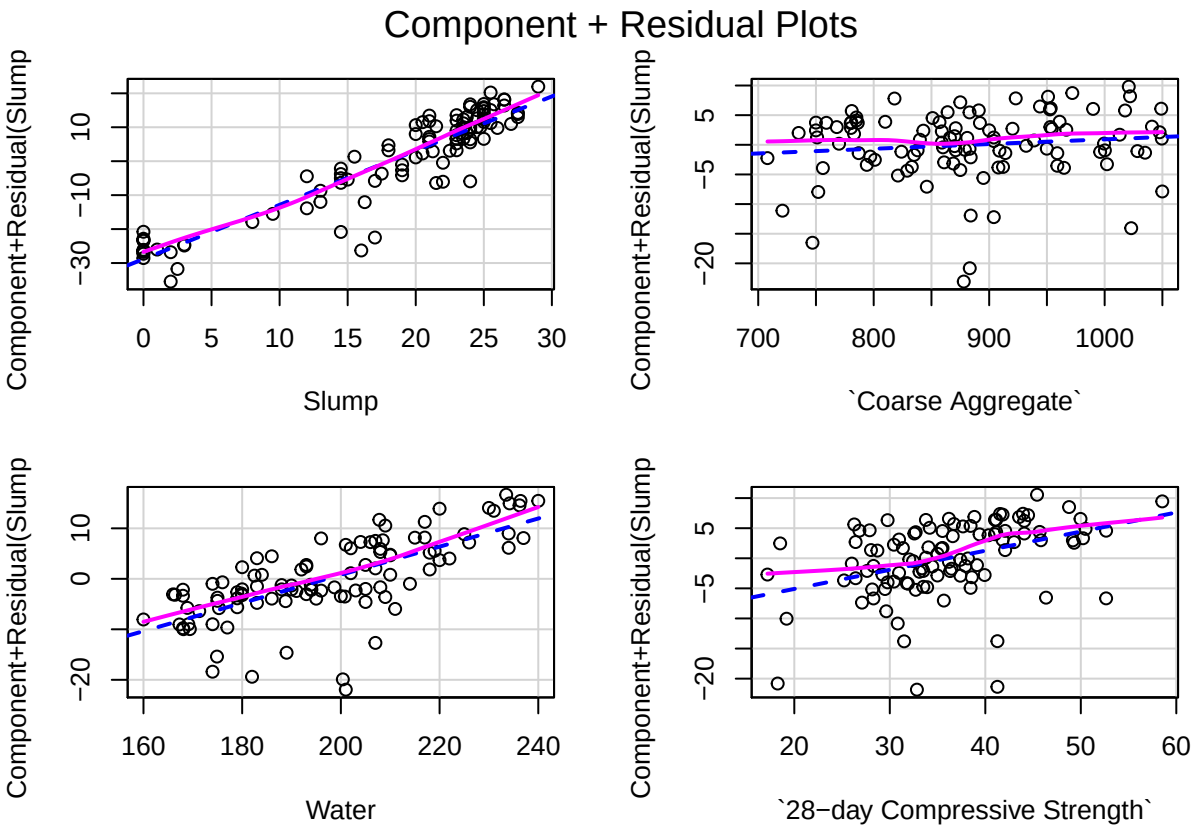

```
# Independence of Errors
durbinWatsonTest(fit_1)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.1429789      2.279677   0.194
##  Alternative hypothesis: rho != 0
```

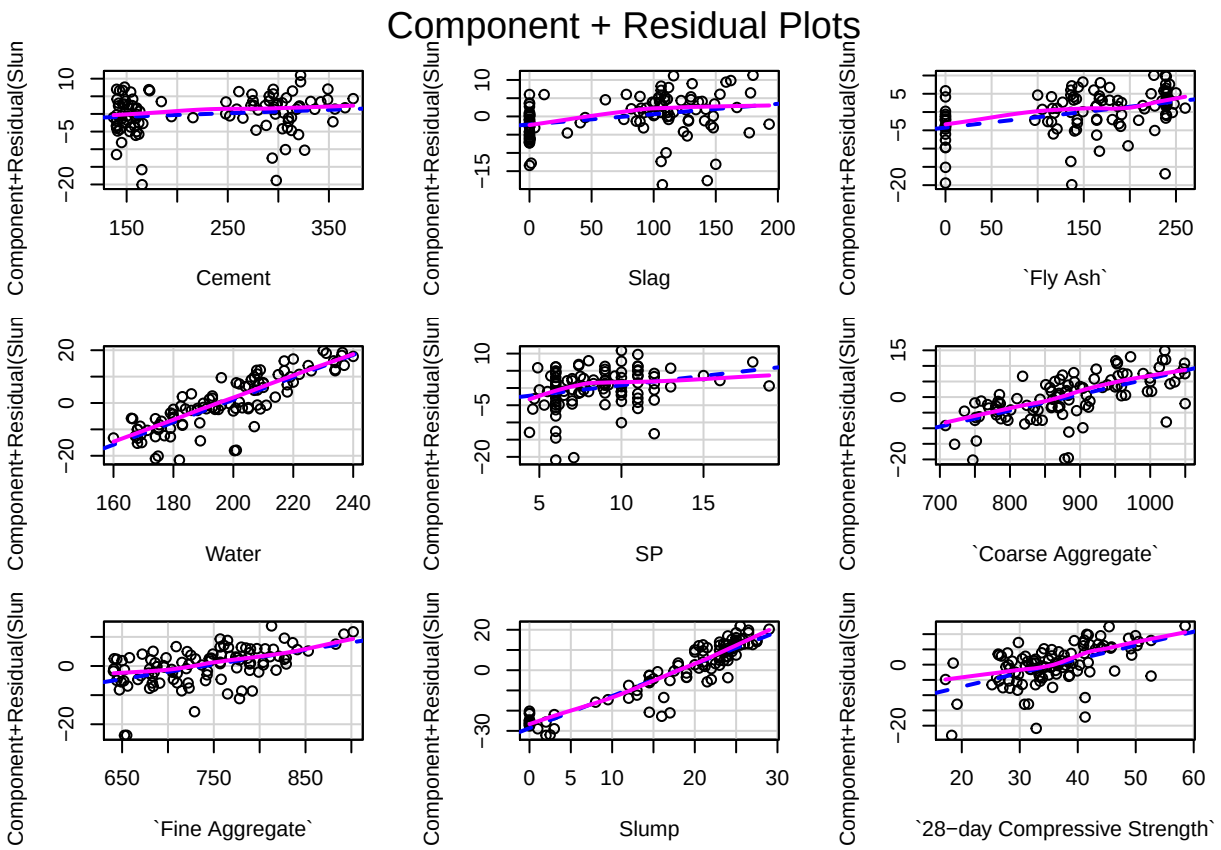
```
durbinWatsonTest(fit_2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.1756961      2.347293   0.124
##  Alternative hypothesis: rho != 0
```

```
# Linearity
library(car)
crPlots(fit_1)
```




```
crPlots(fit_2)
```



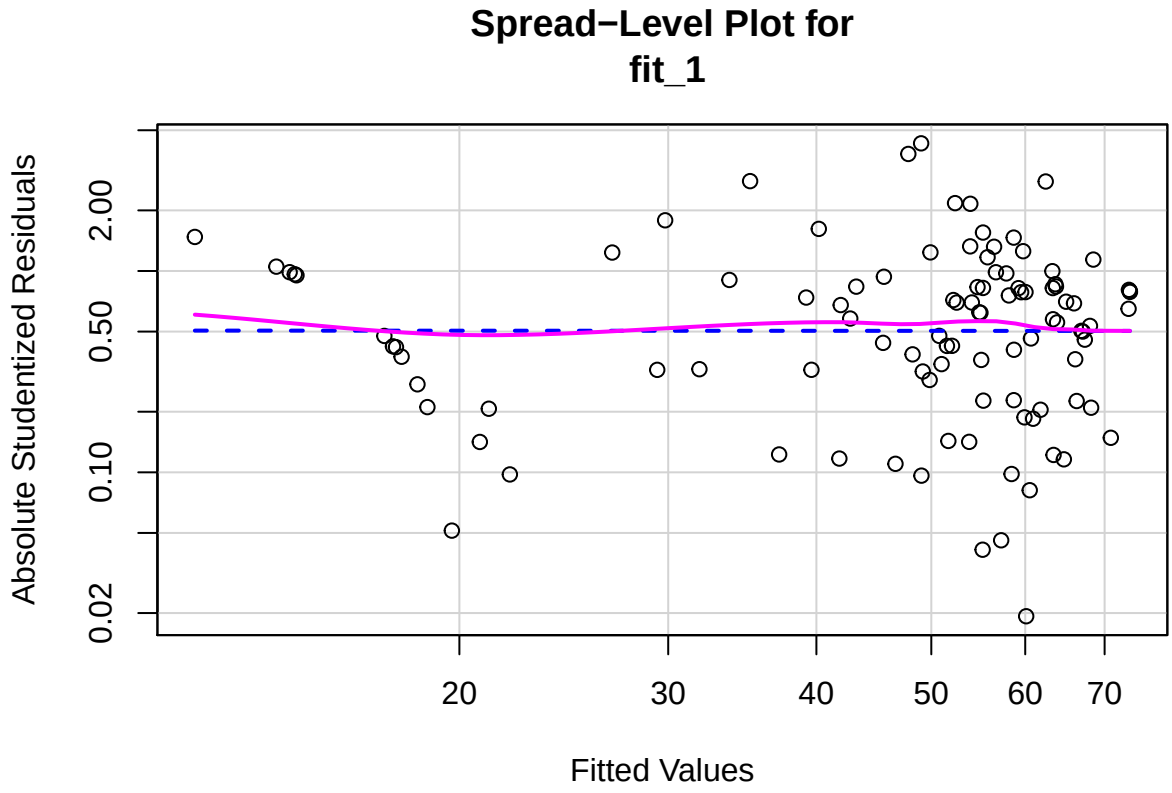

```
# Homoscedacity
ncvTest(fit_1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.1579589, Df = 1, p = 0.69104
```

```
ncvTest(fit_2)
```

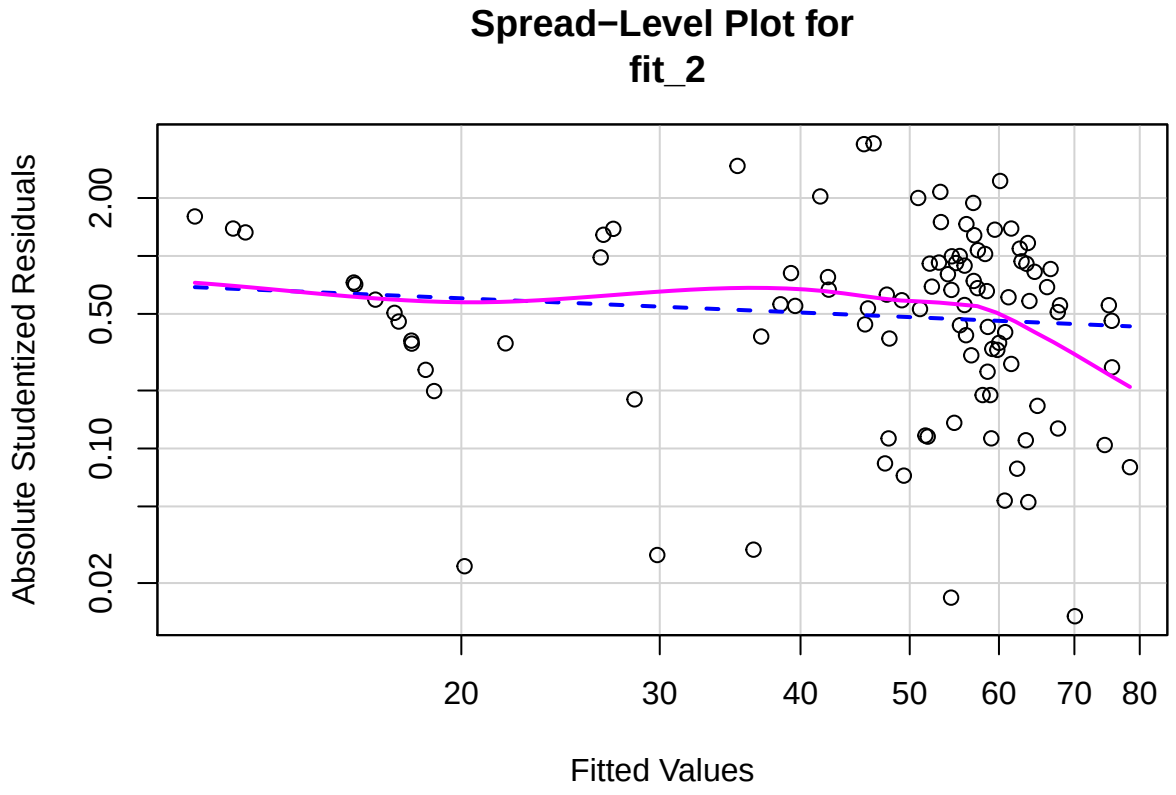
```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.33931, Df = 1, p = 0.24716
```

```
# Assessing Homoscedacity
spreadLevelPlot(fit_1)
```




```
##
## Suggested power transformation:  1.001973
```

```
spreadLevelPlot(fit_2)
```




```
##
## Suggested power transformation:  1.245239
```

```
# Suggested Power Transformation is 1.25
# Which is very close to 1 hence no transformation required
```

```
library(gvlma)
par("mar")
```

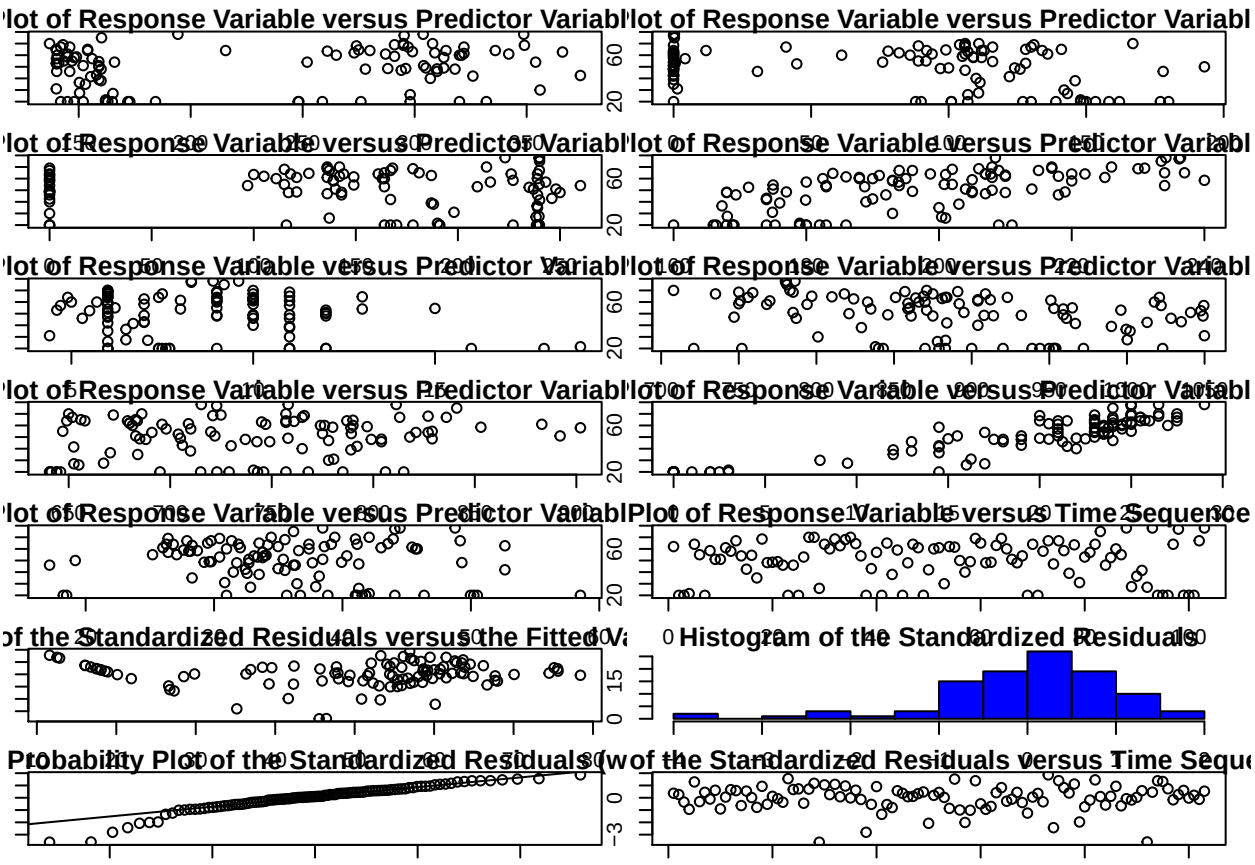
```
## [1] 5.1 4.1 4.1 2.1
```

```
par(mar=c(1,1,1,1))
gvmodel <- gvlma(fit_2)
gvmodel
```

```
##
## Call:
## lm(formula = `Slump Flow` ~ Cement + Slag + `Fly Ash` + Water +
##     SP + `Coarse Aggregate` + `Fine Aggregate` + Slump + `28-day Compressive Strength`,
##     data = cstd)
##
## Coefficients:
##                   (Intercept)                        Cement
##                   -175.303251                      0.009729
##                          Slag                     `Fly Ash`
##                      0.028296                      0.028562
##                         Water                            SP
##                      0.425401                      0.543975
```

```
##            `Coarse Aggregate`                `Fine Aggregate`
##                      0.051510                        0.050360
##                         Slump  `28-day Compressive Strength`
##                      1.588185                        0.450866
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = fit_2)
##
##                       Value              p-value                   Decision
## Global Stat         71.9120 0.000000000000008993 Assumptions NOT satisfied!
## Skewness            25.2687 0.000000498747101729 Assumptions NOT satisfied!
## Kurtosis            24.6425 0.000000690139391413 Assumptions NOT satisfied!
## Link Function       21.1060 0.000004345624428526 Assumptions NOT satisfied!
## Heteroscedasticity  0.8949 0.344160254273815691    Assumptions acceptable.
```

```r
plot(gvmodel)
```



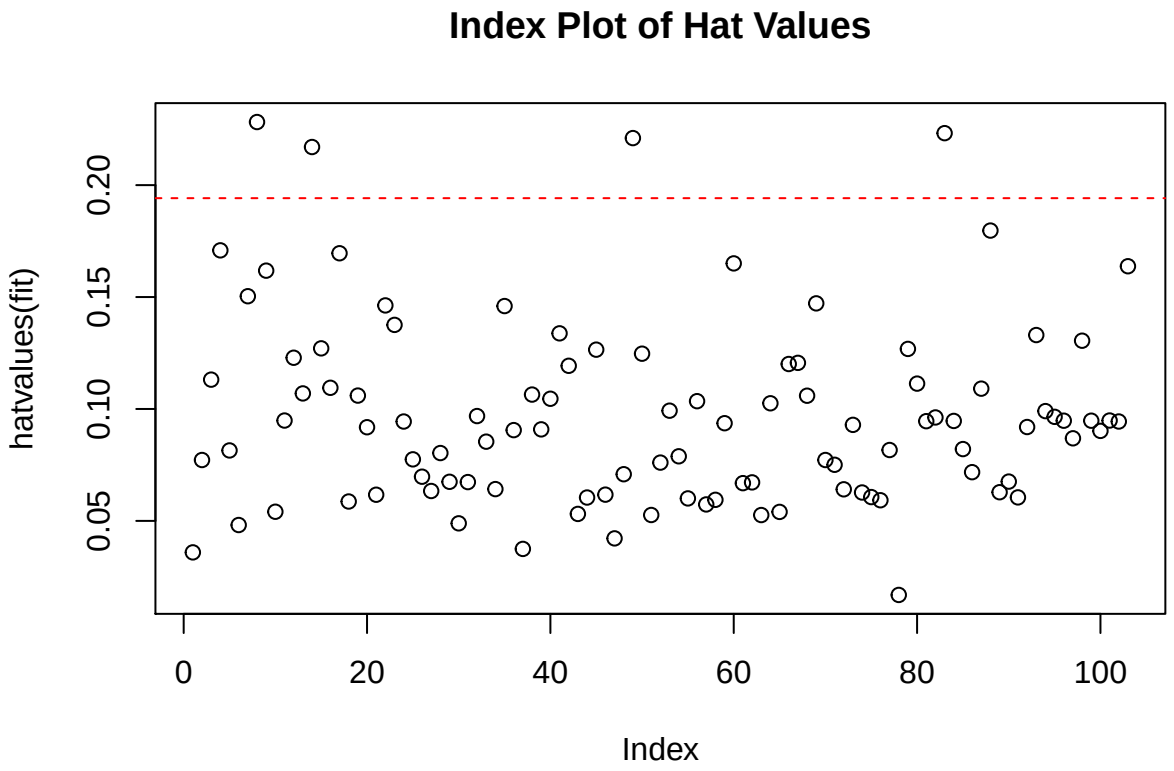

```r
#### Unusual observations

#Outliers
outlierTest(fit_2)
```

```
##     rstudent unadjusted p-value Bonferroni p
## 92 -3.844966         0.00022186     0.022852
## 29 -3.807811         0.00025267     0.026025
```

There is no significant outlier in this dataset. Though it is to be noted that points 92, 29 are outliers of the dataset

#High leverage points

```
hat.plot <- function(fit) {
  p <- length(coefficients(fit))
  n <- length(fitted(fit))
  plot(hatvalues(fit), main="Index Plot of Hat Values")
  abline(h = c(2, 3) * p / n, col = "red", lty = 2)
  identify(1:n, hatvalues(fit), names(hatvalues(fit)))
  }
hat.plot(fit_2)
```

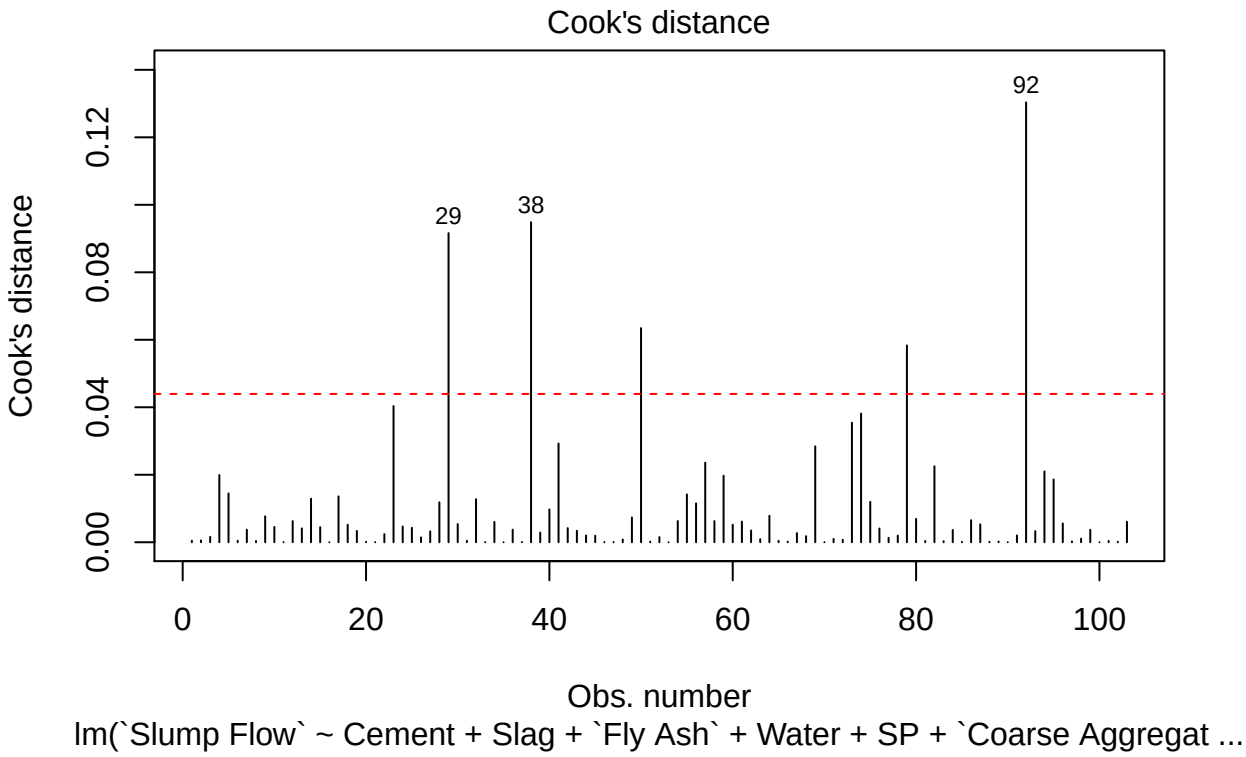



```
## integer(0)
```

We found the 4th, 83th and 88th observations are above the line.

- Influential observations

```
cutoff <- 4 / (nrow(cstd) - length(fit_2$coefficients) - 2)
plot(fit_2, which = 4, cook.levels = cutoff)
abline(h = cutoff, lty = 2, col = "red")
```



The graph identifies 29, 38, 50 and 92 as influential observations.

Corrective measures

- Transforming variables

```
summary(powerTransform(cstd$`Slump Flow`))
```

```
## bcPower Transformation to Normality
##                    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## cstd$`Slump Flow`     1.4678           1       0.9342       2.0015
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df           pval
## LR test, lambda = (0) 31.06187  1 0.000000024993
##
## Likelihood ratio test that no transformation is needed
##                             LRT df     pval
## LR test, lambda = (1) 3.036391  1 0.081417
```

it is suggested that no transformation is needed.

5. Select the best regression model. #### Comparing models

```
fit_1 <- lm(`Slump Flow` ~ Slump + `Coarse Aggregate` + Water + `28-day Compressive Strength`, data = cs

fit_2 <- lm(`Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` + `Fine Aggregate
anova(fit_2, fit_1)
```

```
## Analysis of Variance Table
##
## Model 1: `Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` +
##     `Fine Aggregate` + Slump + `28-day Compressive Strength`
## Model 2: `Slump Flow` ~ Slump + `Coarse Aggregate` + Water + `28-day Compressive Strength`
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     93 3006.2
## 2     98 3364.6 -5   -358.43 2.2177 0.05896 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test is nonsignificant ($p = 0.059$), the p-value is not sufficiently large for us to drop them from our model

variable selection

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

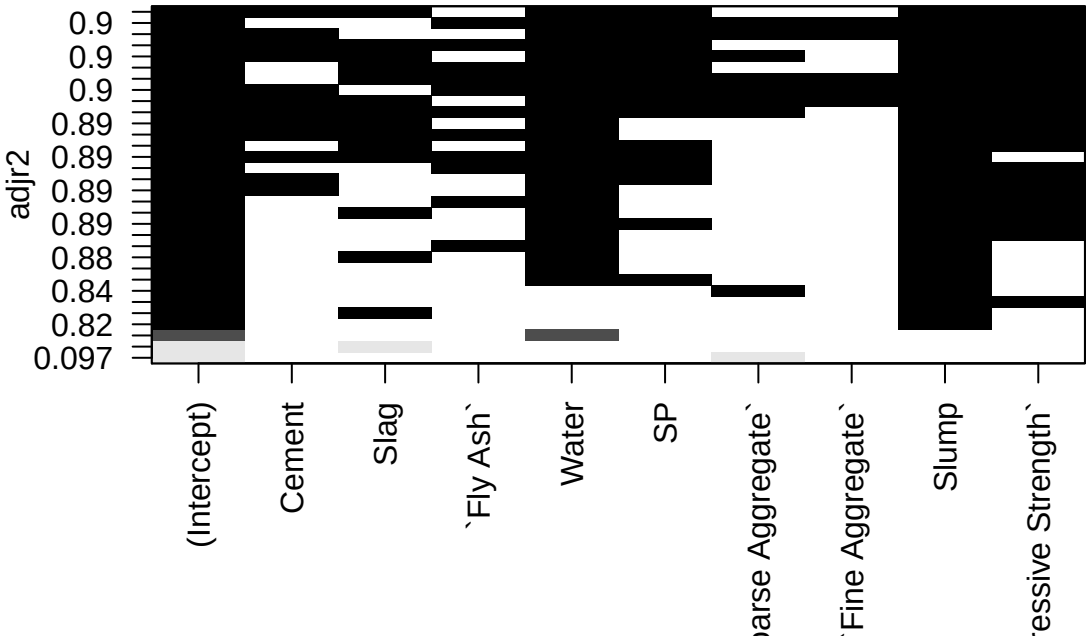
```
stepAIC(fit_2, direction = "backward")
```

```
## Start:  AIC=367.49
## `Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` +
##     `Fine Aggregate` + Slump + `28-day Compressive Strength`
##
##                                 Df Sum of Sq     RSS    AIC
## - Cement                         1       1.1  3007.3 365.53
## - Slag                           1       5.3  3011.5 365.67
## - `Fly Ash`                      1       9.8  3016.0 365.83
## - `Fine Aggregate`               1      20.4  3026.5 366.19
## - `Coarse Aggregate`             1      23.0  3029.1 366.27
## <none>                                        3006.2 367.49
## - SP                             1     110.4  3116.6 369.21
## - `28-day Compressive Strength`  1     119.5  3125.6 369.50
## - Water                          1     215.7  3221.8 372.63
## - Slump                          1   12119.7 15125.9 531.91
##
## Step:  AIC=365.53
## `Slump Flow` ~ Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` +
##     `Fine Aggregate` + Slump + `28-day Compressive Strength`
##
##                                 Df Sum of Sq     RSS    AIC
## - Slag                           1      10.2  3017.5 363.88
## <none>                                        3007.3 365.53
## - `Coarse Aggregate`             1      59.1  3066.4 365.53
## - `Fine Aggregate`               1      62.0  3069.3 365.63
## - `28-day Compressive Strength`  1     137.1  3144.4 368.12
## - `Fly Ash`                      1     148.6  3155.9 368.50
## - SP                             1     169.6  3176.9 369.18
## - Water                          1     514.7  3522.0 379.80
## - Slump                          1   12265.6 15272.9 530.91
```

```
##
## Step:  AIC=363.88
## `Slump Flow` ~ `Fly Ash` + Water + SP + `Coarse Aggregate` +
##     `Fine Aggregate` + Slump + `28-day Compressive Strength`
##
##                                 Df Sum of Sq     RSS    AIC
## <none>                                       3017.5 363.88
## - `Fine Aggregate`               1     131.6  3149.1 366.27
## - `Coarse Aggregate`             1     139.0  3156.5 366.52
## - `Fly Ash`                      1     147.3  3164.8 366.79
## - SP                             1     163.7  3181.2 367.32
## - `28-day Compressive Strength`  1     391.9  3409.4 374.45
## - Water                          1    1524.1  4541.6 403.99
## - Slump                          1   13458.4 16475.9 536.72

##
## Call:
## lm(formula = `Slump Flow` ~ `Fly Ash` + Water + SP + `Coarse Aggregate` +
##     `Fine Aggregate` + Slump + `28-day Compressive Strength`,
##     data = cstd)
##
## Coefficients:
##                   (Intercept)                       `Fly Ash`
##                     -109.13335                         0.01660
##                         Water                               SP
##                       0.35242                         0.49704
##            `Coarse Aggregate`                 `Fine Aggregate`
##                       0.02581                         0.02525
##                         Slump  `28-day Compressive Strength`
##                       1.57483                         0.35033
```

all subsets selection

```
library(leaps)
leaps <- regsubsets(`Slump Flow` ~ Cement + Slag + `Fly Ash` + Water + SP + `Coarse Aggregate` + `Fine
plot(leaps, scale = "adjr2")
```



#6. Fine tune the selection of predictor variables. #### Cross-validation

```
shrinkage <- function(fit, k=10){
require(bootstrap)
theta.fit <- function(x,y){lsfit(x,y)}
theta.predict <- function(fit,x){cbind(1,x)%*%fit$coef}
x <- fit$model[,2:ncol(fit$model)]
y <- fit$model[,1]
results <- crossval(x, y, theta.fit, theta.predict, ngroup=k)
r2 <- cor(y, fit$fitted.values)^2
r2cv <- cor(y, results$cv.fit)^2
cat("Original R-square =", r2, "\n")
cat(k, "Fold Cross-Validated R-square =", r2cv, "\n")
cat("Change =", r2-r2cv, "\n")
}
shrinkage(fit_2)
```

```
## Loading required package: bootstrap
```

```
## Original R-square = 0.9045143
## 10 Fold Cross-Validated R-square = 0.8849932
## Change = 0.0195211
```

There is some shrinkage in the adjust R squared but it is still acceptable

```
#Relative Importance
library(dplyr)
cstd_std <- cstd %>% mutate_all(scale)
```

```
rel_fit <- lm(`Slump Flow` ~ Cement + Slag + Slag:Water + `Fly Ash` + Water + SP + Slag:`Coarse Aggrega
coef(rel_fit)
```

```
##                   (Intercept)                        Cement
##                   0.004417232                  -0.154340574
##                          Slag                     `Fly Ash`
##                  -0.102880427                  -0.082316476
##                         Water                            SP
##                   0.312883204                   0.051584040
##               `Fine Aggregate`                         Slump
##                  -0.013608720                   0.816454022
## `28-day Compressive Strength`                    Slag:Water
##                   0.202305536                  -0.043131997
##       Slag:`Coarse Aggregate`
##                   0.025091919
```

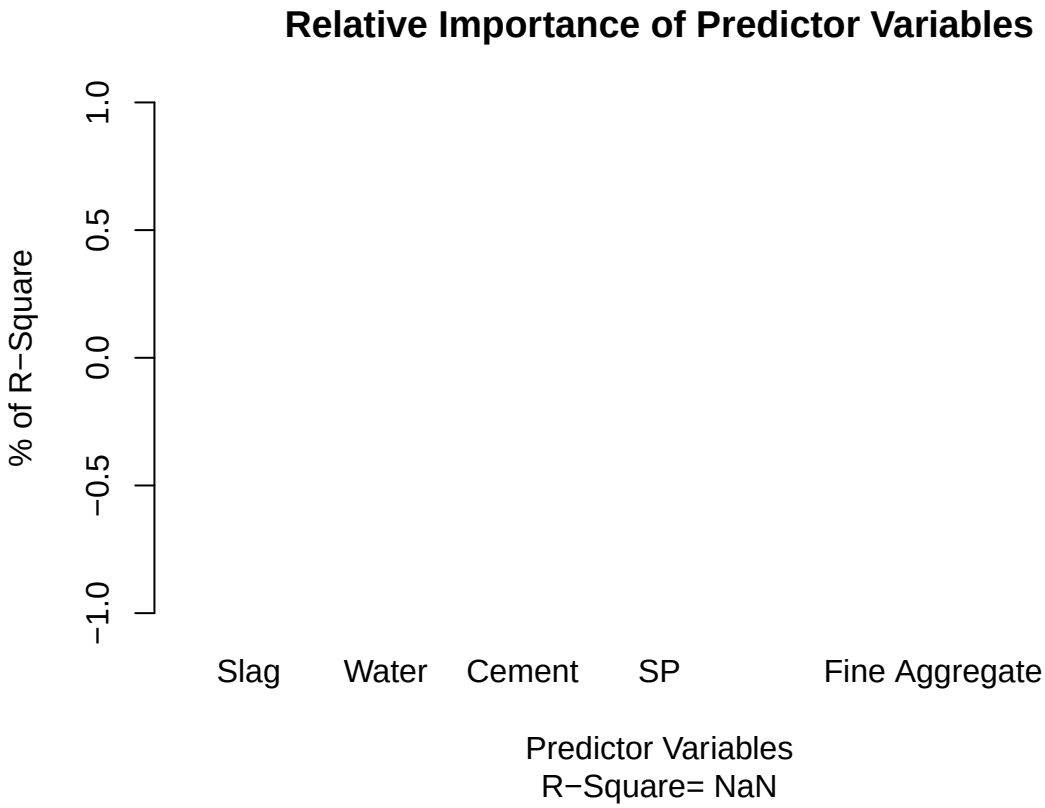
```
relweights<-function(fit,...){
  R<-cor(fit$model)
  nvar<-ncol(R)
  rxx<-R[2:nvar,2:nvar]
  rxy<-R[2:nvar,1]
  svd<-eigen(rxx)
  evec<-svd$vectors
  ev<-svd$values
  delta<-diag(sqrt(ev))
  lambda<-evec%*%delta%*%t(evec)
  lambdasq<-lambda^2
  beta<-solve(lambda)%*%rxy
  rsquare<-colSums(beta^2)
  rawwgt<-lambdasq%*%beta^2
  import<-(rawwgt/rsquare)*100
  lbls<-names(fit$model[2:nvar])
  rownames(import)<-lbls
  colnames(import)<-"Weights"
  barplot(t(import),names.arg=lbls,
          ylab="% of R-Square",
          xlab="Predictor Variables",
          main="Relative Importance of Predictor Variables",
          sub=paste("R-Square=",round(rsquare,digits=3)),
          ...)
  return(import)
}
```

```
fit_final <- lm(`Slump Flow` ~ Slag + Water + Cement:SP + Slag:Water + Slag:`Coarse Aggregate` + Slag:`
summary(fit_final)
```

```
##
## Call:
## lm(formula = `Slump Flow` ~ Slag + Water + Cement:SP + Slag:Water +
##     Slag:`Coarse Aggregate` + Slag:`Fine Aggregate` + `Fly Ash`:`Coarse Aggregate` +
##     Water:SP + Cement + SP + `Coarse Aggregate` + `Fine Aggregate` +
##     `Fly Ash` + `Coarse Aggregate`, data = cstd)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.0785  -5.0220  -0.0481   6.8206  23.6384
##
## Coefficients:
##                                 Estimate   Std. Error t value    Pr(>|t|)
## (Intercept)                 -279.9876088  384.8914409  -0.727      0.4689
## Slag                          -3.8306428    0.6827114  -5.611 0.000000225 ***
## Water                          0.8949831    0.4537782   1.972      0.0517 .
## Cement                         0.0192096    0.1406699   0.137      0.8917
## SP                             5.2239264    4.9876815   1.047      0.2978
## `Coarse Aggregate`             0.0514686    0.1506457   0.342      0.7334
## `Fine Aggregate`               0.0763249    0.1507533   0.506      0.6139
## `Fly Ash`                      0.0522079    0.2146951   0.243      0.8084
## Cement:SP                      0.0167330    0.0067635   2.474      0.0153 *
## Slag:Water                     0.0066826    0.0013413   4.982 0.000003066 ***
## Slag:`Coarse Aggregate`        0.0014820    0.0003220   4.602 0.000013820 ***
## Slag:`Fine Aggregate`          0.0018027    0.0004115   4.381 0.000032210 ***
## `Coarse Aggregate`:`Fly Ash`   0.0001673    0.0001821   0.919      0.3606
## Water:SP                      -0.0401600    0.0264220  -1.520      0.1321
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.72 on 89 degrees of freedom
## Multiple R-squared:  0.675,  Adjusted R-squared:  0.6275
## F-statistic: 14.22 on 13 and 89 DF,  p-value: < 0.00000000000000022
```

```
relweights(fit_final, col = "blue")
```




```
##                   Weights
## Slag                  NaN
## Water                 NaN
## Cement                NaN
## SP                    NaN
## Coarse Aggregate      NaN
## Fine Aggregate        NaN
## Fly Ash               NaN
```

```
# The final regression model will look like:
# Slump Flow = -3.83*Slag + 0.895*Water + 0.019*Cemet + 5.22*SP + 0.051*`Coarse Aggregate` + 0.076325*`
```

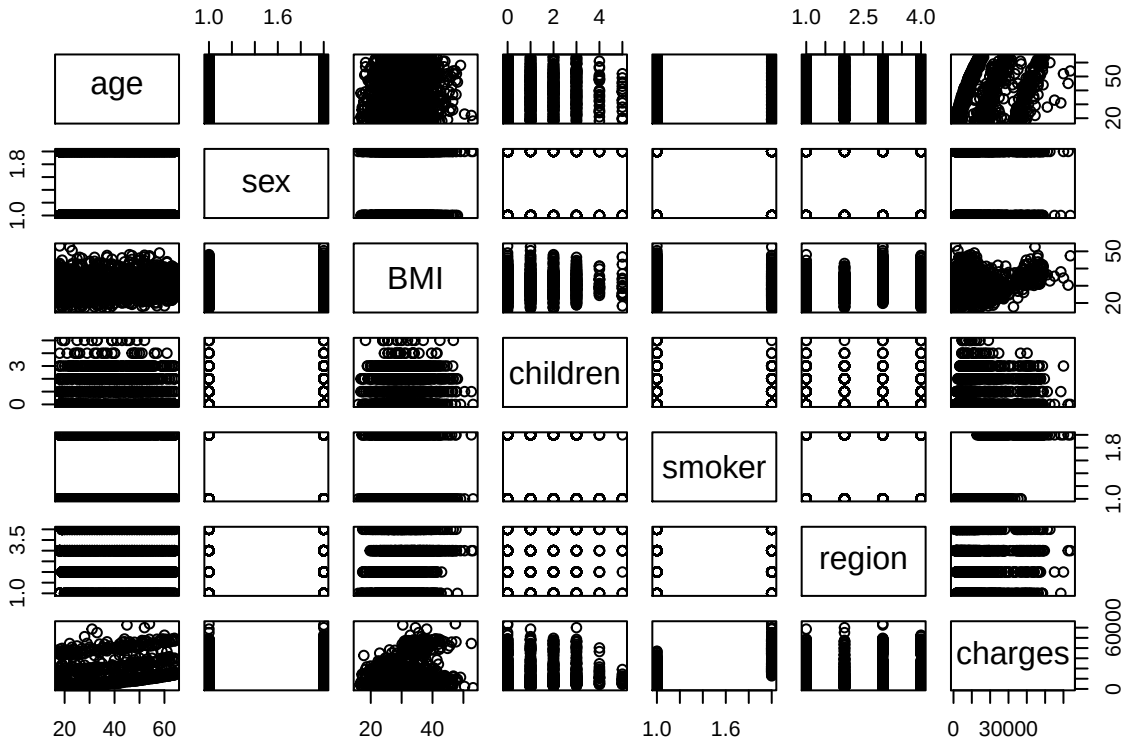
Problem 3

```
library(car)
lic <- read.csv("C:/Users/abhil/Downloads/insurance.csv", stringsAsFactors = TRUE)
#Prior to building a regression model, it is often helpful to check for normality. Although linear regr

summary(lic)
```

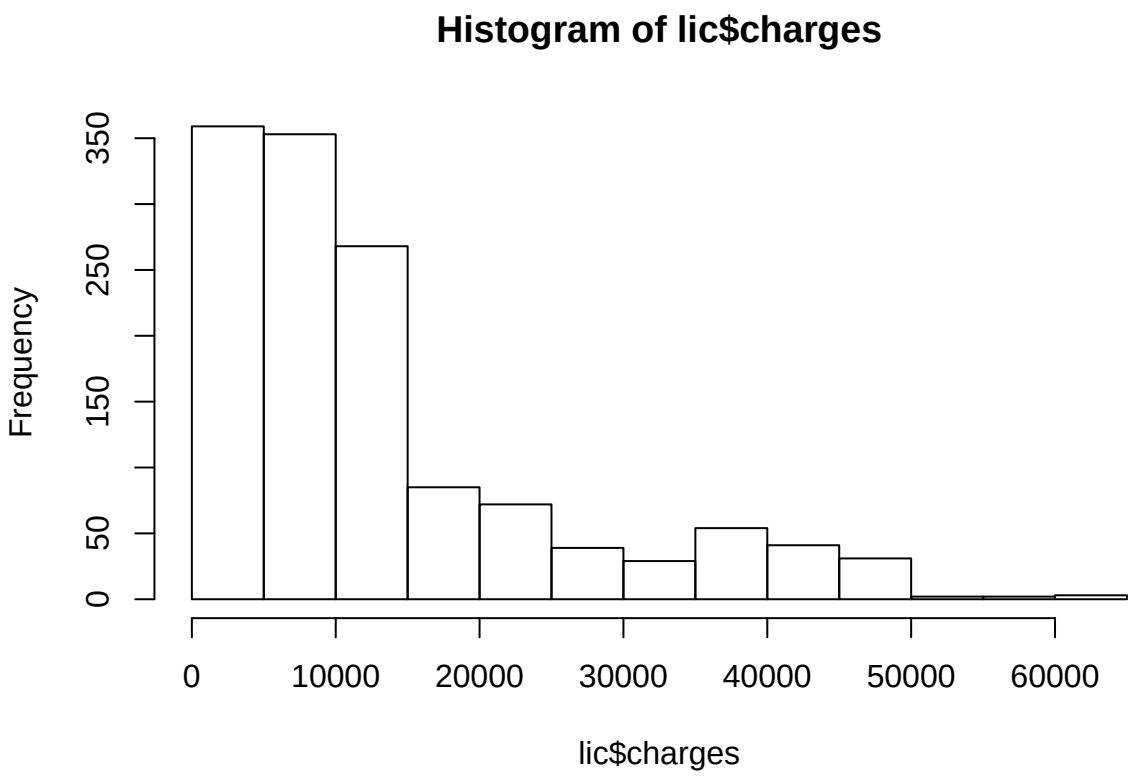
```
##       age            sex           BMI           children     smoker
##  Min.   :18.00   female:662   Min.   :15.96   Min.   :0.000   no :1064
##  1st Qu.:27.00   male  :676   1st Qu.:26.30   1st Qu.:0.000   yes: 274
##  Median :39.00                Median :30.40   Median :1.000
##  Mean   :39.21                Mean   :30.66   Mean   :1.095
##  3rd Qu.:51.00                3rd Qu.:34.69   3rd Qu.:2.000
```

```
##  Max.   :64.00                Max.   :53.13   Max.   :5.000
##        region       charges
##  northeast:324   Min.   : 1122
##  northwest:325   1st Qu.: 4740
##  southeast:364   Median : 9382
##  southwest:325   Mean   :13270
##                  3rd Qu.:16640
##                  Max.   :63770
```

```
plot(lic)
```




```
hist(lic$charges)
```

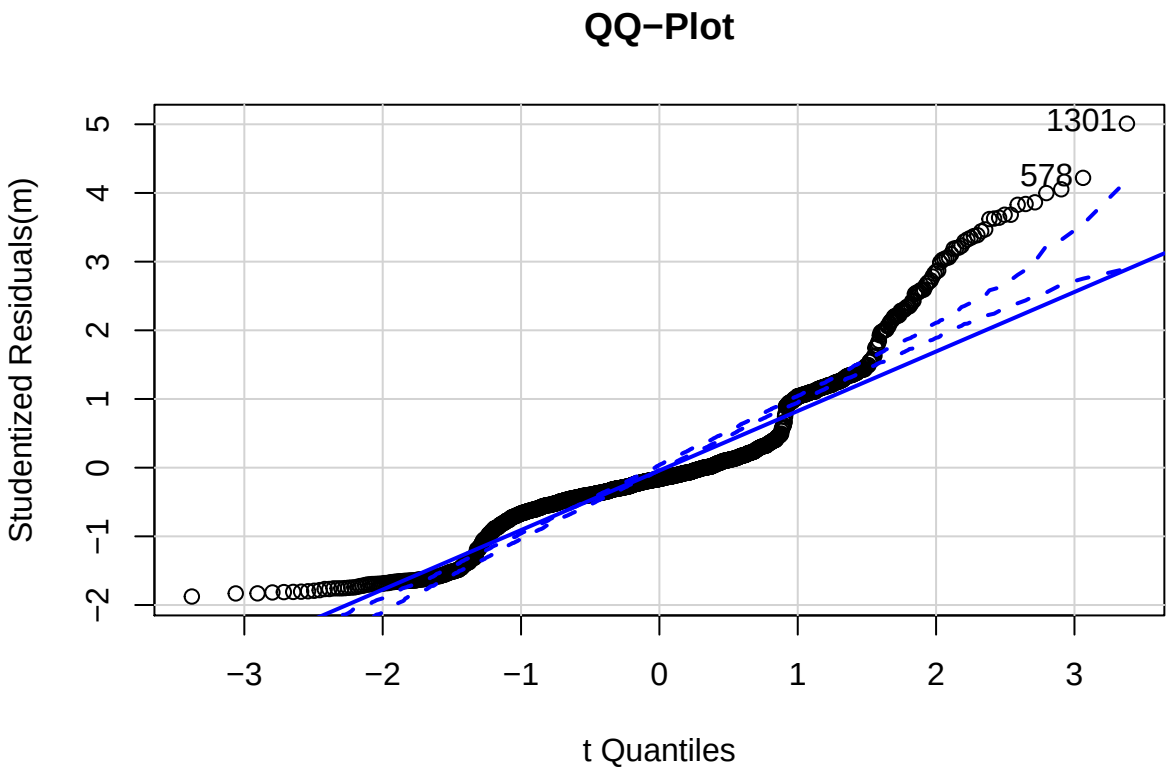



```
m<- lm( formula = charges ~ . - charges, data = lic)
summary(m)
```

```
##
## Call:
## lm(formula = charges ~ . - charges, data = lic)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                  Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     -11938.5      987.8 -12.086 < 0.0000000000000002 ***
## age                256.9       11.9  21.587 < 0.0000000000000002 ***
## sexmale           -131.3      332.9  -0.394             0.693348
## BMI                339.2       28.6  11.860 < 0.0000000000000002 ***
## children           475.5      137.8   3.451             0.000577 ***
## smokeryes        23848.5      413.1  57.723 < 0.0000000000000002 ***
## regionnorthwest   -353.0      476.3  -0.741             0.458769
## regionsoutheast  -1035.0      478.7  -2.162             0.030782 *
## regionsouthwest   -960.0      477.9  -2.009             0.044765 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6062 on 1329 degrees of freedom
```

```
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 0.00000000000000022
```

```
qqPlot(m, labels = row.names(lic), id.method = "identify", simulate = TRUE, main = "QQ-Plot")
```




```
## [1]  578 1301
```

```
#the data is not normally distribued according as the analysis of QQ plot suggests that the numbers don
#from the given data, we can see that BMI is diversely varied and can't be treated for any relation wit
#Create a correlation matrix and a scatterplot matrix for the four numeric variables in the insurance d
cor(lic[-c(2,5,6)])
```

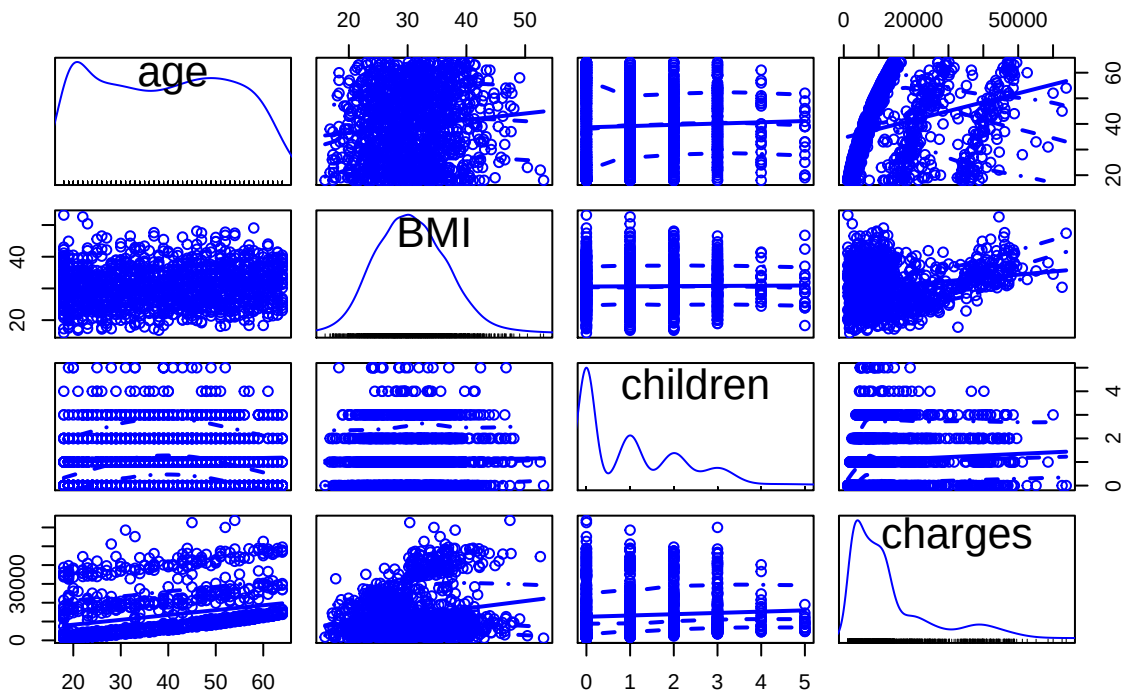
```
##                age       BMI   children    charges
## age      1.0000000 0.1092719 0.04246900 0.29900819
## BMI      0.1092719 1.0000000 0.01275890 0.19834097
## children 0.0424690 0.0127589 1.00000000 0.06799823
## charges  0.2990082 0.1983410 0.06799823 1.00000000
```

```
lic1 <- lic[-c(2,5,6)]
library(car)
scatterplotMatrix(lic1, spread = FALSE, main = "Scatterplot Matrix")
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
```

```
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter
```

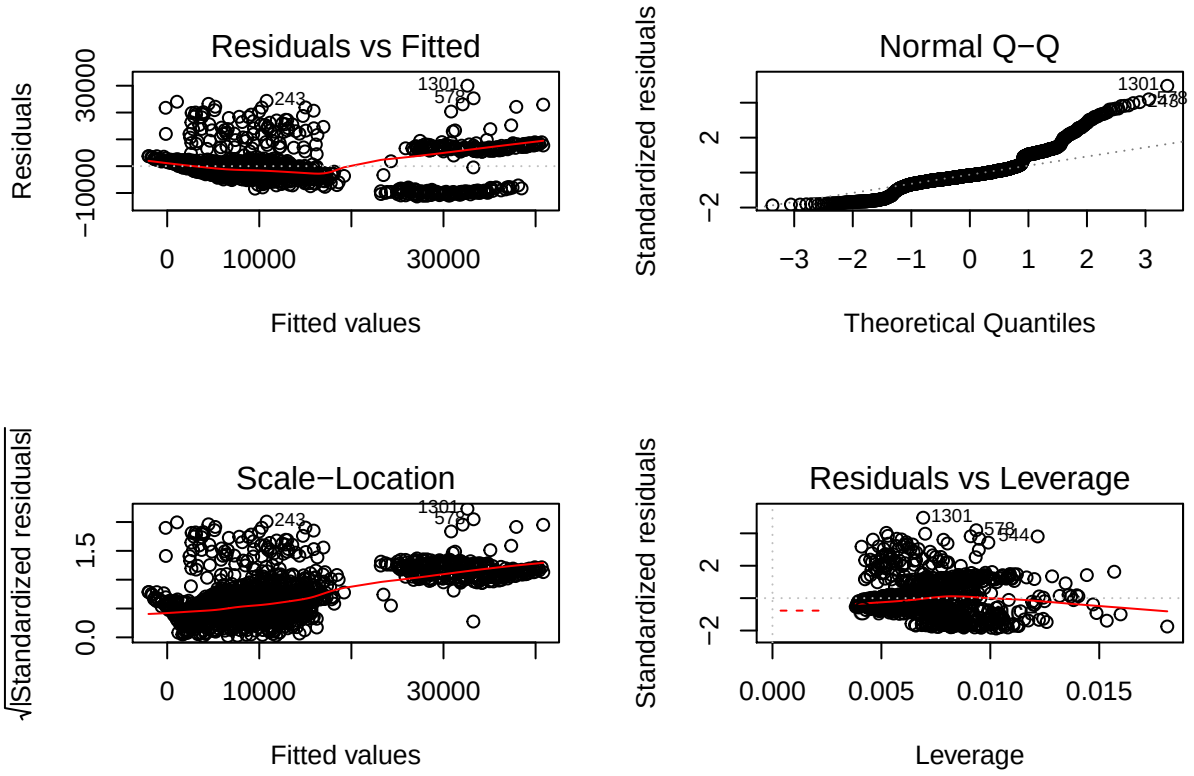



```
#The correlation matrix and scatterplot matrix shows that the charges and age is slightly correlated fo
#Thus, they can be considered as independent variable for our following regression model along with sex
#Build a regression model using the independent variables, then evaluate the model performance.
model <- lm( formula = charges ~ . + age + BMI + children + sex, data = lic)
summary(model)
```

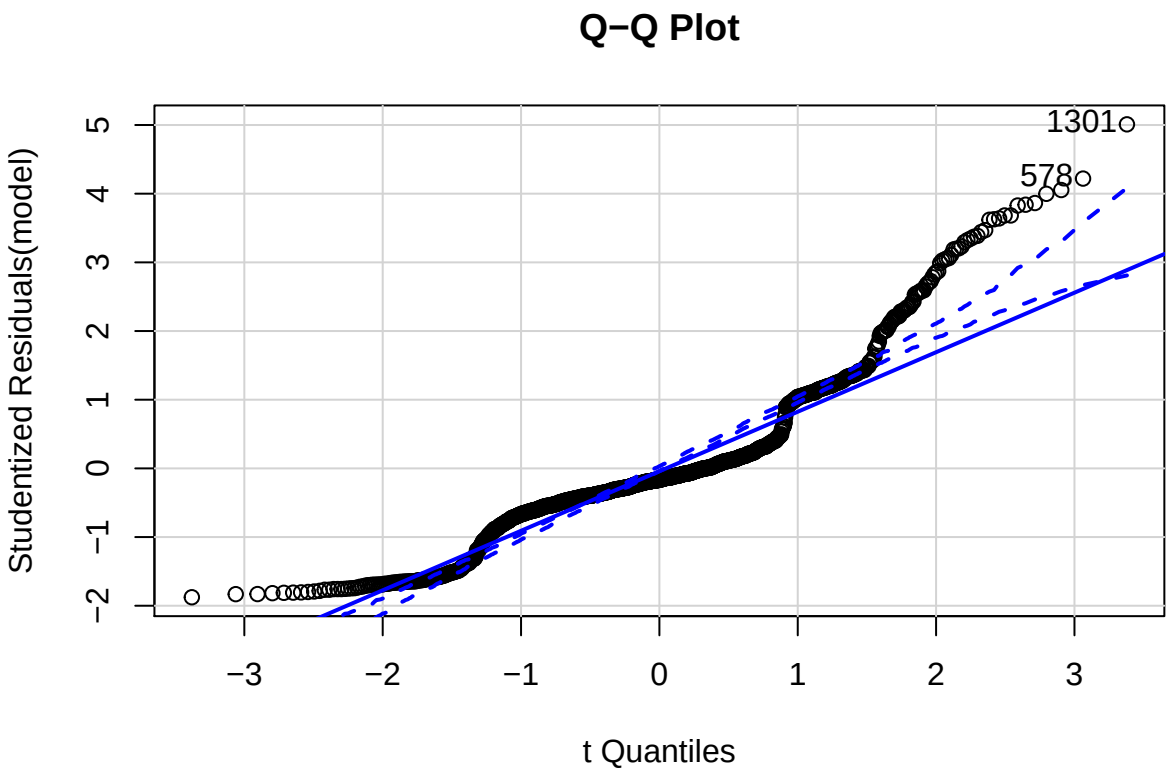
```
##
## Call:
## lm(formula = charges ~ . + age + BMI + children + sex, data = lic)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11304.9  -2848.1   -982.1   1393.9  29992.8
##
## Coefficients:
##                  Estimate Std. Error t value             Pr(>|t|)
## (Intercept)     -11938.5      987.8 -12.086 < 0.0000000000000002 ***
## age                256.9       11.9  21.587 < 0.0000000000000002 ***
## sexmale           -131.3      332.9  -0.394             0.693348
## BMI                339.2       28.6  11.860 < 0.0000000000000002 ***
## children           475.5      137.8   3.451             0.000577 ***
## smokeryes        23848.5      413.1  57.723 < 0.0000000000000002 ***
## regionnorthwest   -353.0      476.3  -0.741             0.458769
## regionsoutheast  -1035.0      478.7  -2.162             0.030782 *
## regionsouthwest   -960.0      477.9  -2.009             0.044765 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6062 on 1329 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 0.00000000000000022
```

```
#Perform regression diagnostics using both typical approach and enhanced approach for the regression mo

#typical approach

library(MASS)
par(mfrow = c(2,2))
plot(model)
```




```
#enhanced approach
#normality
par(mfrow = c(1,1))
qqPlot(model, labels = row.names(ff), id.method = "identify", simulate = TRUE, main = "Q-Q Plot")
```

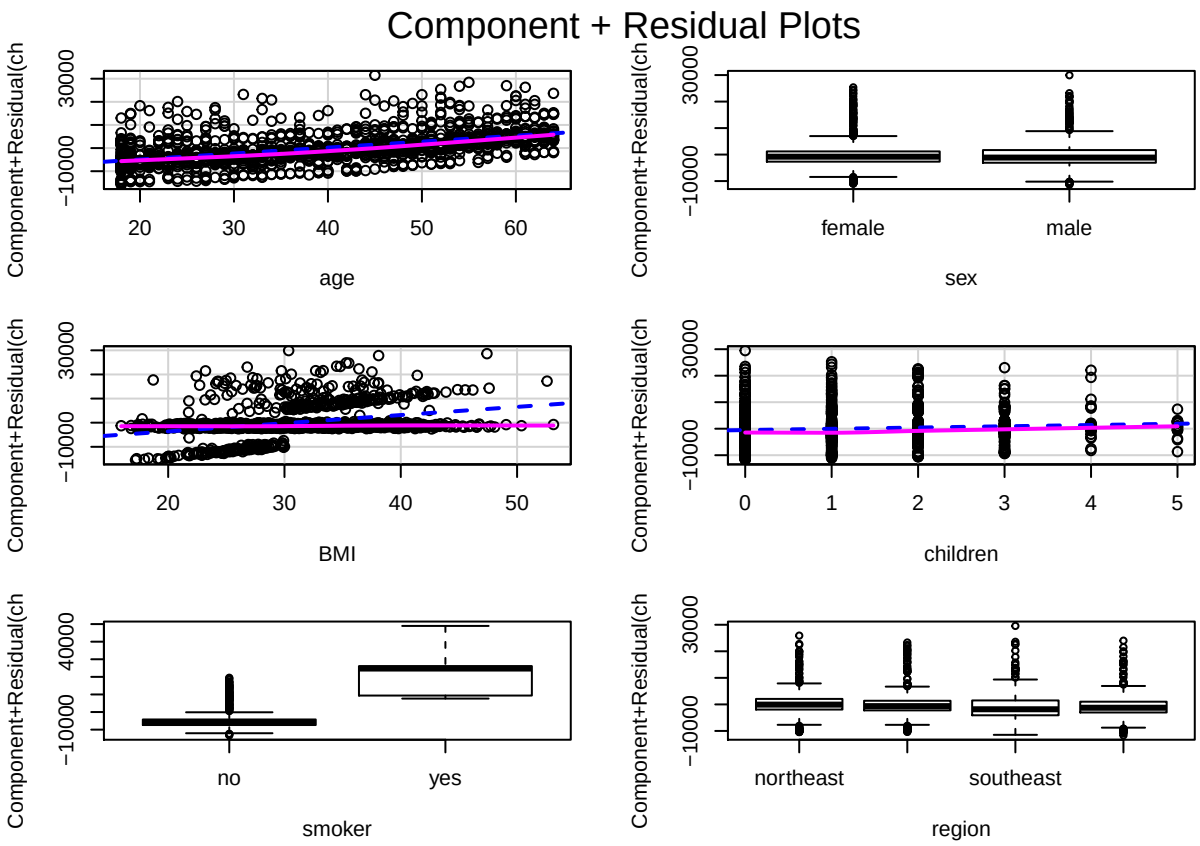



```
## [1]  578 1301
```

```
#Based on this graph, we can see the points fall close to the line only upto a certain extent, otherwis

#Independence
durbinWatsonTest(model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.04558149      2.088423   0.102
##  Alternative hypothesis: rho != 0
```

```
#Linearity
crPlots(model)
```




```
#Homoscedasticity

ncvTest(model)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 236.1255, Df = 1, p = < 0.000000000000000222
```

```
#Improve the regression model by adding a non-linear term for age and creating an indicator for obesity

#adding non linear age to the model
lic$age1 <- lic$age^2

#converting numeric value to a binary indicator
lic$BMI30 <-ifelse(lic$BMI >= 30, 1, 0)

#adding non-linear variables to the new model along with the independent variables
model2 <- lm( formula = charges ~ . + age + age1 + BMI + children + sex + region + BMI30*smoker, data =
summary(model2)
```

```
##
## Call:
## lm(formula = charges ~ . + age + age1 + BMI + children + sex +
##     region + BMI30 * smoker, data = lic)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -17296.4  -1656.0  -1263.3   -722.1  24160.2
##
## Coefficients:
##                   Estimate Std. Error t value             Pr(>|t|)
## (Intercept)       134.2509  1362.7511   0.099             0.921539
## age               -32.6851    59.8242  -0.546             0.584915
## sexmale          -496.8245   244.3659  -2.033             0.042240 *
## BMI               120.0196    34.2660   3.503             0.000476 ***
## children          678.5612   105.8831   6.409       0.000000000204 ***
## smokeryes       13404.6866   439.9491  30.469 < 0.0000000000000002 ***
## regionnorthwest  -279.2038   349.2746  -0.799             0.424212
## regionsoutheast  -828.5467   351.6352  -2.356             0.018604 *
## regionsouthwest -1222.6437   350.5285  -3.488             0.000503 ***
## age1                3.7316     0.7463   5.000       0.000000649662 ***
## BMI30           -1000.1403   422.8402  -2.365             0.018159 *
## smokeryes:BMI30 19810.7533   604.6567  32.764 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4445 on 1326 degrees of freedom
## Multiple R-squared:  0.8664, Adjusted R-squared:  0.8653
## F-statistic: 781.7 on 11 and 1326 DF,  p-value: < 0.00000000000000022
```

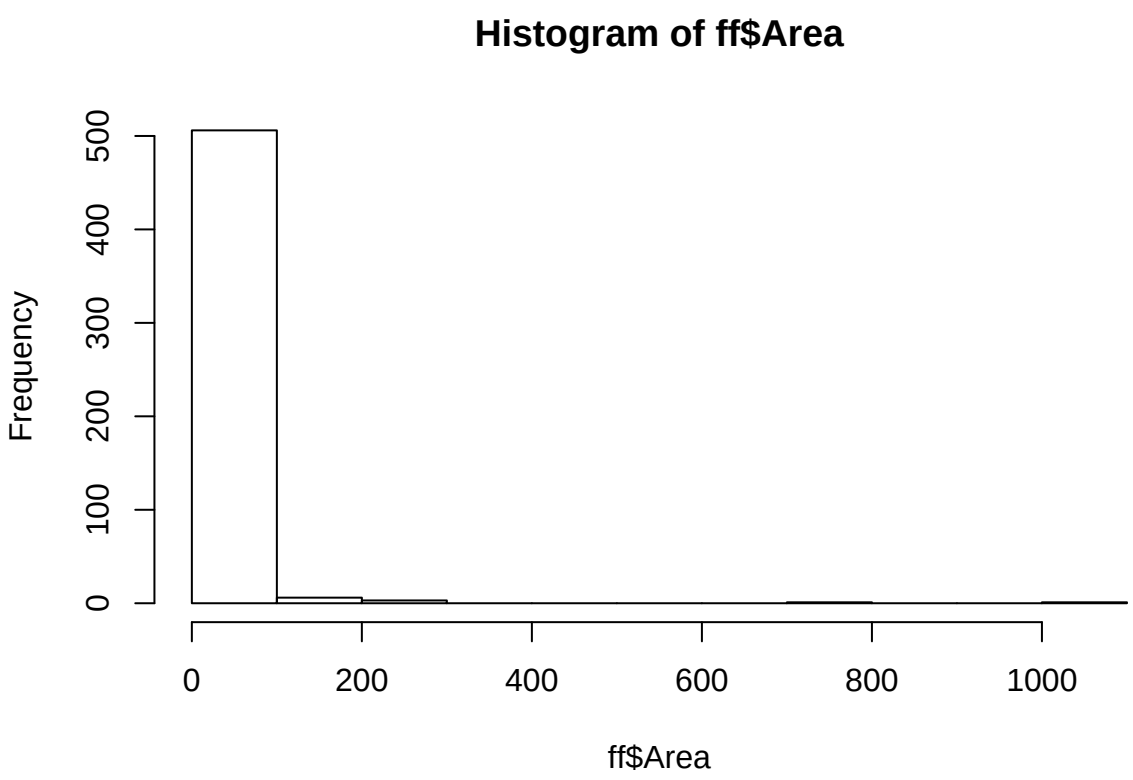
```
#Hence, model 2 seems like a better regression model.
```

Problem 4

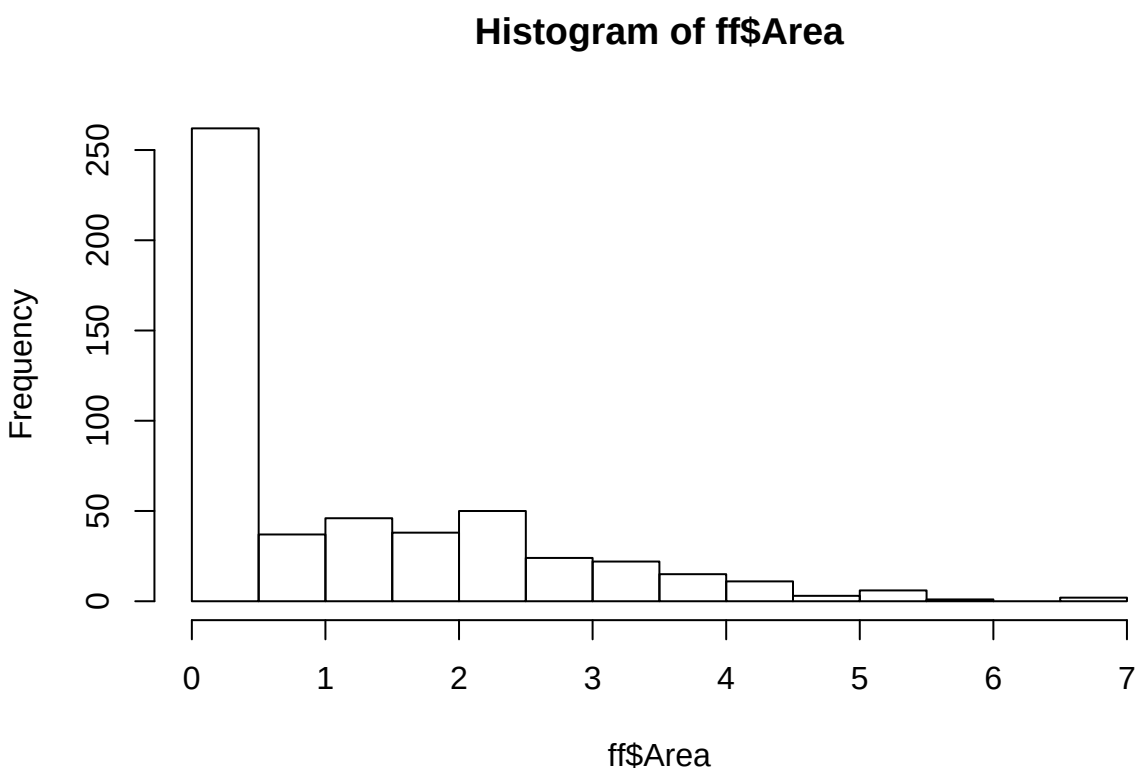
```
library(readxl)
ff <- read_excel("C:/Users/abhil/Downloads/Forest Fires Data(1).xlsx")
ff <- ff[complete.cases(ff),]

str(ff)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    517 obs. of  13 variables:
##  $ X    : num  7 7 7 8 8 8 8 8 8 7 ...
##  $ Y    : num  5 4 4 6 6 6 6 6 6 5 ...
##  $ Month: chr  "mar" "oct" "oct" "mar" ...
##  $ Day  : chr  "fri" "tue" "sat" "fri" ...
##  $ FFMC : num  86.2 90.6 90.6 91.7 89.3 92.3 92.3 91.5 91 92.5 ...
##  $ DMC  : num  26.2 35.4 43.7 33.3 51.3 ...
##  $ DC   : num  94.3 669.1 686.9 77.5 102.2 ...
##  $ ISI  : num  5.1 6.7 6.7 9 9.6 14.7 8.5 10.7 7 7.1 ...
##  $ Temp : num  8.2 18 14.6 8.3 11.4 22.2 24.1 8 13.1 22.8 ...
##  $ RH   : num  51 33 33 97 99 29 27 86 63 40 ...
##  $ Wind : num  6.7 0.9 1.3 4 1.8 5.4 3.1 2.2 5.4 4 ...
##  $ Rain : num  0 0 0 0.2 0 0 0 0 0 0 ...
##  $ Area : num  0 0 0 0 0 0 0 0 0 0 ...
```

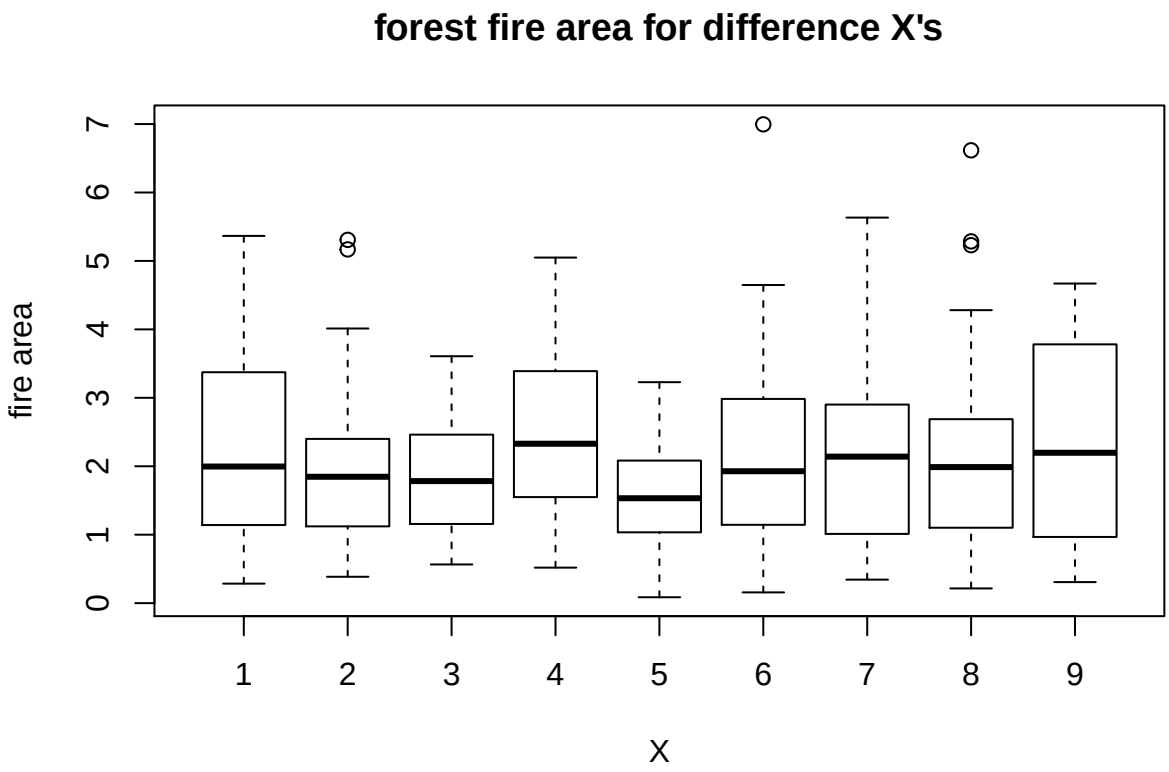
```
hist(ff$Area)
```



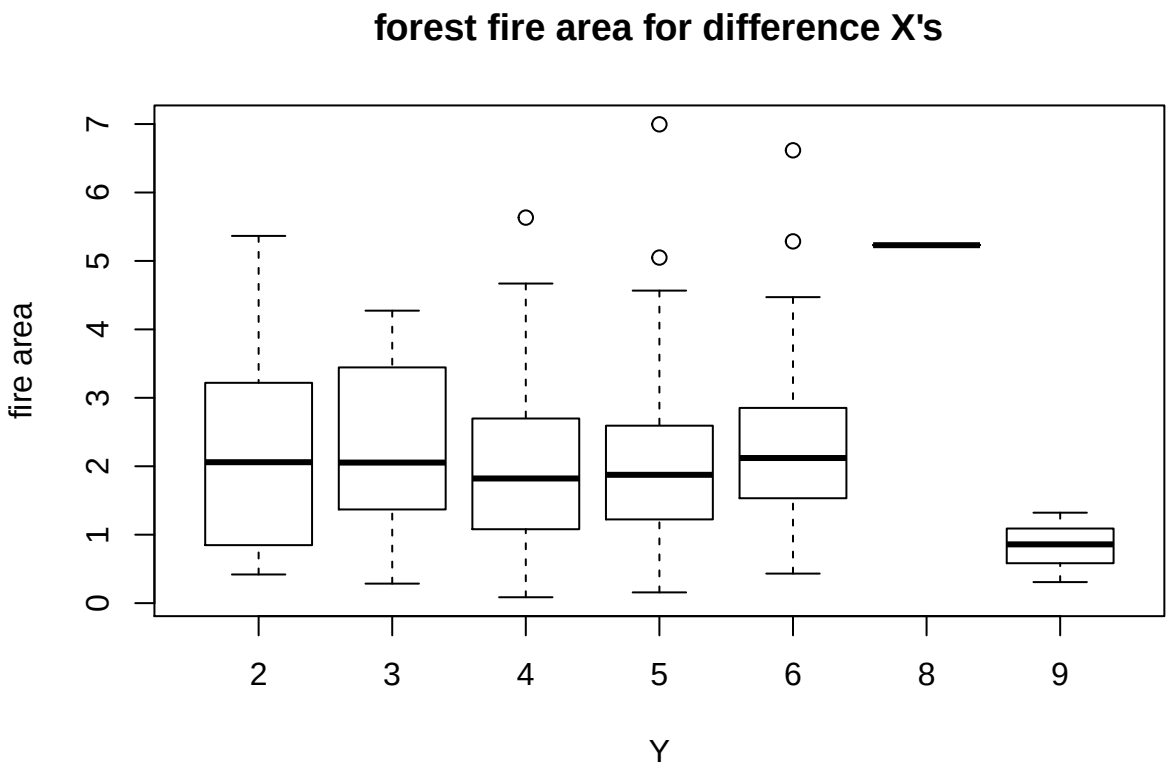

```
ff$Area <- ff$Area + 1
ff$Area <- log(ff$Area)
hist(ff$Area)
```



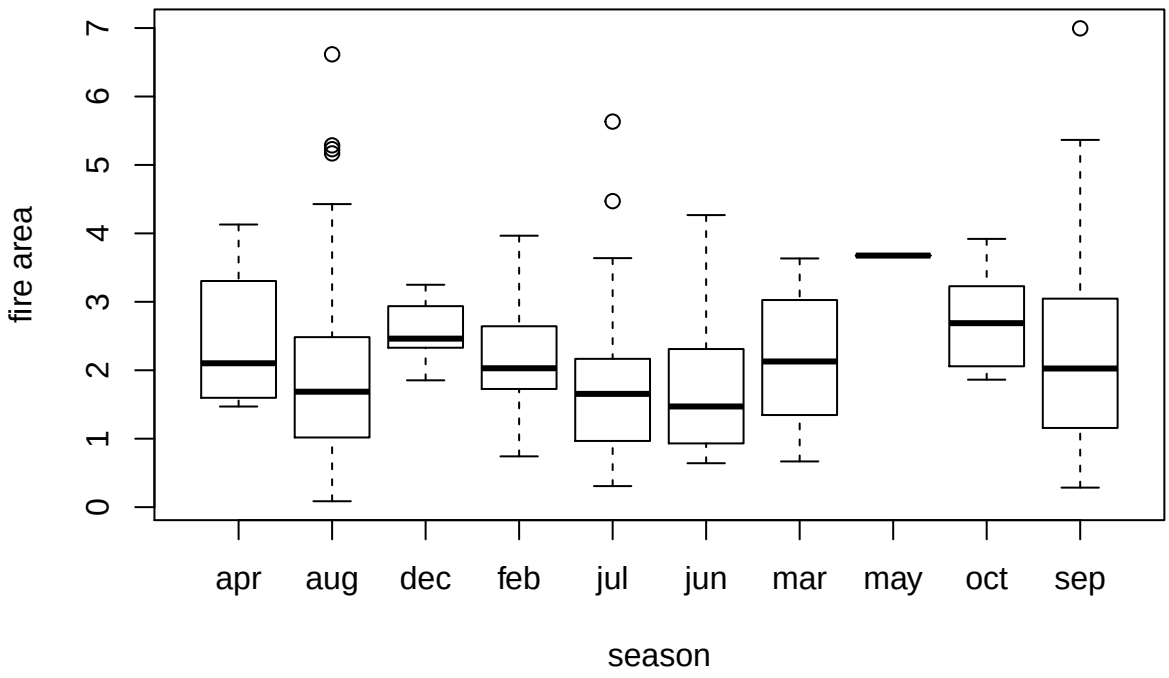

```
#Data exploration---Find the relationship between the Forestfires and Spatial Variables (X & Y)
par(mfrow = c(1,1))
ff <- ff[ff$Area>0, ]
boxplot(ff$Area ~ as.factor(X), data = ff, xlab = "X", ylab = "fire area", main = "forest fire area for
```




```
boxplot(ff$Area ~ as.factor(Y), data = ff, xlab = "Y", ylab = "fire area", main = "forest fire area for
```




```
#Based on the boxplots, we can see that it doesn't show any obvious relationship between the spatial lo
#Data exploration---Find the relationship between the Forestfires and Temporal Variables (Months & Days
boxplot(ff$Area ~ ff$Month, data = ff, xlab = "season", ylab = "fire area", main = "forest fire area fo
```




```
#Let's first examine the bivariate relationships with the *cor()* function, and then generate the scatt
#The aim of this task is to understand the how the burned are of forest fires, in the northeast region

#Create a scatterplot matrix of "Forest Fire Data" and select an initial set of predictor variables
#We set an initia; set of predictors as FWI and Meteorological data
scatterplotMatrix(ff[-c(3,4)], spread = FALSE, main = "Scatter plot matrix")
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
```

```
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

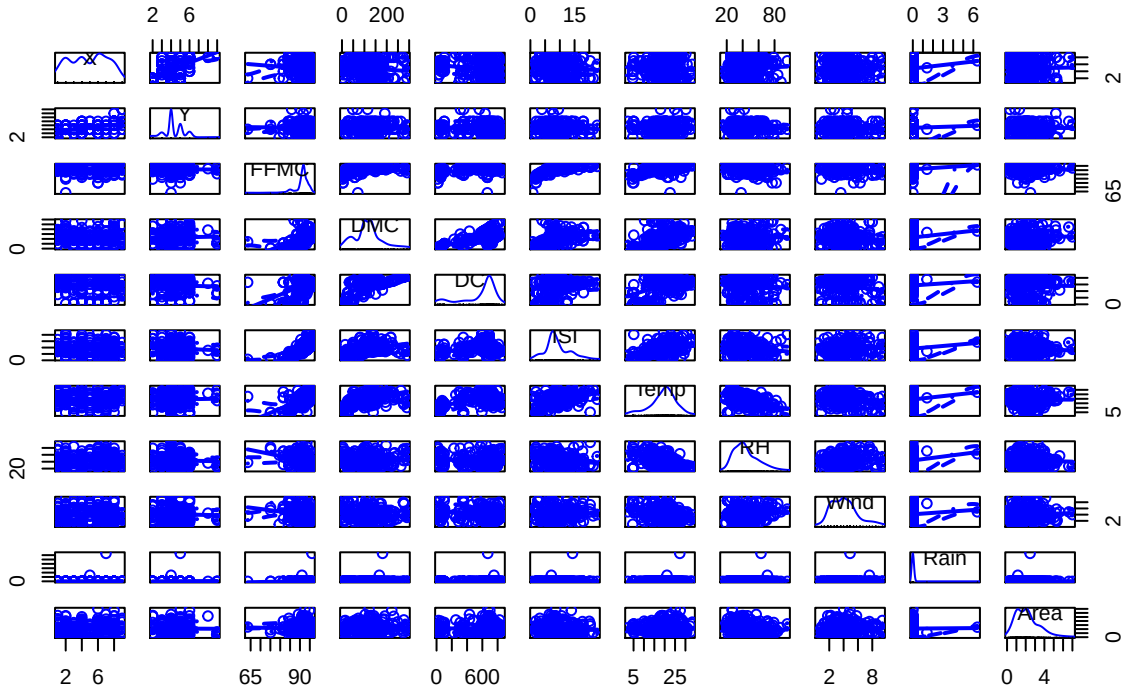
```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread
```

```
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit negative part of the spread
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter
## Warning in title(...): "spread" is not a graphical parameter
## Warning in plot.window(...): "spread" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in smoother(x[subs], y[subs], col = smoother.args$col[i], log.x =
## FALSE, : could not fit smooth

## Warning in plot.window(...): "spread" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "spread" is not a graphical parameter

## Warning in title(...): "spread" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "spread" is not a
## graphical parameter
```

Scatter plot matrix




```
#Build a few potential regression models using "Forest Fire Data"

#We start from the simplest model, without any interaction and quadratic terms

m0 <- lm(Area ~ X + Y + FFMC + DMC + DC + ISI + Temp + RH + Wind + Rain, data = ff)
summary(m0)
```

```
##
## Call:
## lm(formula = Area ~ X + Y + FFMC + DMC + DC + ISI + Temp + RH +
##     Wind + Rain, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0878 -0.9142 -0.1657  0.6505  4.7245
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.2002522  3.0343182   0.725   0.4690
## X            0.0322701  0.0379328   0.851   0.3957
## Y           -0.0649455  0.0767913  -0.846   0.3985
## FFMC         0.0072518  0.0340149   0.213   0.8313
## DMC          0.0033127  0.0019075   1.737   0.0836 .
## DC          -0.0003593  0.0004724  -0.761   0.4476
## ISI         -0.0456458  0.0284349  -1.605   0.1097
## Temp        -0.0062486  0.0206059  -0.303   0.7619
```

```
## RH          -0.0100595  0.0066331  -1.517   0.1306
## Wind         0.0483358  0.0460989   1.049   0.2954
## Rain         0.0439902  0.1975839   0.223   0.8240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.259 on 259 degrees of freedom
## Multiple R-squared:  0.03451,    Adjusted R-squared:  -0.002767
## F-statistic: 0.9258 on 10 and 259 DF,  p-value: 0.5099
```

```r
#we can see that the R square is pretty small, which means the model is terrible. It might simply becau
#Therefore, we will apply the four meterological indices (FFMC, DMCM, DC, ISI) since the influnce of th
m1 <- lm(formula = Area ~ .-FFMC - DMC - DC - ISI, data = ff)
summary(m1)
```

```
##
## Call:
## lm(formula = Area ~ . - FFMC - DMC - DC - ISI, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1941 -0.8426 -0.1332  0.5882  4.5409
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.566912   1.022042   1.533   0.1265
## X            0.034492   0.039210   0.880   0.3799
## Y           -0.032288   0.078937  -0.409   0.6829
## Monthaug    -0.942789   0.734881  -1.283   0.2007
## Monthdec     0.360014   0.805755   0.447   0.6554
## Monthfeb    -0.243610   0.762802  -0.319   0.7497
## Monthjul    -1.016788   0.788929  -1.289   0.1987
## Monthjun    -1.057832   0.840862  -1.258   0.2096
## Monthmar    -0.401117   0.706622  -0.568   0.5708
## Monthmay     1.207434   1.452398   0.831   0.4066
## Monthoct     0.092771   0.882548   0.105   0.9164
## Monthsep    -0.450635   0.706883  -0.637   0.5244
## Daymon       0.062920   0.296184   0.212   0.8319
## Daysat       0.612647   0.283047   2.164   0.0314 *
## Daysun       0.409369   0.277486   1.475   0.1414
## Daythu       0.145987   0.312953   0.466   0.6413
## Daytue       0.340182   0.291989   1.165   0.2451
## Daywed       0.055894   0.309948   0.180   0.8570
## Temp         0.036174   0.028393   1.274   0.2038
## RH           0.001264   0.008099   0.156   0.8761
## Wind         0.039662   0.046799   0.848   0.3975
## Rain         0.002653   0.201668   0.013   0.9895
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.258 on 248 degrees of freedom
## Multiple R-squared:  0.07741,    Adjusted R-squared:  -0.0007114
## F-statistic: 0.9909 on 21 and 248 DF,  p-value: 0.4751
```

```r
m2<- lm( formula = Area ~ . - Temp - RH - Wind - Rain, data = ff)
summary(m2)
```

```
##
## Call:
## lm(formula = Area ~ . - Temp - RH - Wind - Rain, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1576 -0.9006 -0.1046  0.6202  4.4273
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.584007   2.882753   0.549 0.583173
## X            0.049142   0.038520   1.276 0.203238
## Y           -0.075072   0.078005  -0.962 0.336791
## Monthaug     1.075513   1.099485   0.978 0.328930
## Monthdec     1.582308   0.915745   1.728 0.085252 .
## Monthfeb    -0.321698   0.746355  -0.431 0.666824
## Monthjul     0.441656   0.918615   0.481 0.631093
## Monthjun    -0.044469   0.861588  -0.052 0.958879
## Monthmar    -0.330384   0.698724  -0.473 0.636744
## Monthmay     1.450597   1.408560   1.030 0.304087
## Monthoct     3.004121   1.363778   2.203 0.028530 *
## Monthsep     2.109197   1.247331   1.691 0.092099 .
## Daymon       0.080785   0.288705   0.280 0.779849
## Daysat       0.559118   0.275695   2.028 0.043626 *
## Daysun       0.455919   0.268794   1.696 0.091110 .
## Daythu       0.195276   0.300741   0.649 0.516735
## Daytue       0.371903   0.284328   1.308 0.192083
## Daywed       0.129056   0.296662   0.435 0.663921
## FFMC         0.009613   0.033850   0.284 0.776648
## DMC          0.008155   0.002296   3.552 0.000457 ***
## DC          -0.004551   0.001695  -2.686 0.007729 **
## ISI         -0.011099   0.029026  -0.382 0.702507
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.232 on 248 degrees of freedom
## Multiple R-squared:  0.1146, Adjusted R-squared:  0.03959
## F-statistic: 1.528 on 21 and 248 DF,  p-value: 0.06862
```

```r
m3 <- lm( formula = Area ~ . + (FFMC + DMC + DC + ISI)^2, data = ff)
summary(m3)
```

```
##
## Call:
## lm(formula = Area ~ . + (FFMC + DMC + DC + ISI)^2, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3344 -0.8487 -0.0889  0.6421  3.8913
##
## Coefficients:
```

```
##                 Estimate   Std. Error t value Pr(>|t|)
## (Intercept)  0.025083039  7.769016540   0.003   0.9974
## X            0.058121518  0.038910093   1.494   0.1366
## Y           -0.089989159  0.079559799  -1.131   0.2592
## Monthaug    -1.019193769  1.478217830  -0.689   0.4912
## Monthdec     0.321528017  1.096554788   0.293   0.7696
## Monthfeb     0.000020467  0.775983595   0.000   1.0000
## Monthjul    -1.381747896  1.302473009  -1.061   0.2898
## Monthjun    -1.737034150  1.198743471  -1.449   0.1486
## Monthmar    -0.959346103  0.776475192  -1.236   0.2179
## Monthmay     1.019206865  1.442173418   0.707   0.4804
## Monthoct     1.156137457  1.640343407   0.705   0.4816
## Monthsep     0.263000560  1.580112707   0.166   0.8679
## Daymon      -0.040794830  0.297791187  -0.137   0.8912
## Daysat       0.547536282  0.280436938   1.952   0.0521 .
## Daysun       0.350641357  0.275435459   1.273   0.2042
## Daythu       0.113808816  0.312163554   0.365   0.7157
## Daytue       0.287523780  0.294062635   0.978   0.3292
## Daywed       0.094834661  0.306650473   0.309   0.7574
## FFMC         0.020611132  0.092411100   0.223   0.8237
## DMC         -0.101070707  0.096152866  -1.051   0.2943
## DC           0.020419577  0.017514510   1.166   0.2448
## ISI         -0.058627322  0.781752416  -0.075   0.9403
## Temp         0.008116885  0.030453630   0.267   0.7901
## RH          -0.002131796  0.008203085  -0.260   0.7952
## Wind         0.043072906  0.049209372   0.875   0.3823
## Rain        -0.026142283  0.199077683  -0.131   0.8956
## FFMC:DMC     0.001420127  0.001073270   1.323   0.1870
## FFMC:DC     -0.000236990  0.000203556  -1.164   0.2455
## FFMC:ISI     0.000787480  0.008340830   0.094   0.9249
## DMC:DC      -0.000026532  0.000013921  -1.906   0.0579 .
## DMC:ISI     -0.000005988  0.000698895  -0.009   0.9932
## DC:ISI      -0.000078807  0.000201064  -0.392   0.6954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.222 on 238 degrees of freedom
## Multiple R-squared:  0.1643, Adjusted R-squared:  0.05549
## F-statistic:  1.51 on 31 and 238 DF,  p-value: 0.04722
```

```
m4 <- lm(formula = Area ~ (.^2), data = ff)
summary(m4)
```

```
##
## Call:
## lm(formula = Area ~ (.^2), data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1736 -0.3220  0.0000  0.2793  2.6599
##
## Coefficients: (86 not defined because of singularities)
##                       Estimate    Std. Error t value Pr(>|t|)
## (Intercept)     -543.52731264  322.35857618  -1.686  0.09563 .
## X                  6.06421853    5.43025903   1.117  0.26740
```

```
## Y              -0.06256072  16.55649692  -0.004  0.99699
## Monthaug        47.69963725  62.57816585   0.762  0.44813
## Monthdec        46.07523231  72.06862646   0.639  0.52442
## Monthfeb       987.74886437 952.75175075   1.037  0.30295
## Monthjul       146.22250688 159.92272640   0.914  0.36326
## Monthjun       434.20669165 656.08282273   0.662  0.50997
## Monthmar       391.62626094 230.89748398   1.696  0.09371 .
## Monthmay       -20.09480833  15.93919372  -1.261  0.21103
## Monthoct       100.12422181  46.45753106   2.155  0.03412 *
## Monthsep         2.85482191  34.93649863   0.082  0.93508
## Daymon          -2.68858646  44.85494448  -0.060  0.95235
## Daysat         112.66009427  43.08022654   2.615  0.01063 *
## Daysun          41.20264792  42.67449076   0.966  0.33716
## Daythu          78.87008068  63.02473928   1.251  0.21439
## Daytue          16.41787455  36.15883384   0.454  0.65101
## Daywed          14.70732520  57.25375523   0.257  0.79792
## FFMC             6.59031928   3.38288003   1.948  0.05486 .
## DMC             -0.83048231   0.35839624  -2.317  0.02302 *
## DC               0.52762892   0.33948638   1.554  0.12404
## ISI             -3.55747294   3.37307761  -1.055  0.29471
## Temp             5.83045810   5.92128296   0.985  0.32772
## RH               2.07935620   1.63679014   1.270  0.20758
## Wind           -10.05261317   7.89773449  -1.273  0.20671
## Rain             8.69703880   7.19026021   1.210  0.22997
## X:Y              0.03858603   0.05699378   0.677  0.50032
## X:Monthaug      -0.28121966   0.25542344  -1.101  0.27416
## X:Monthdec      -1.15380646   1.66880780  -0.691  0.49129
## X:Monthfeb      -4.82350097   4.27604664  -1.128  0.26264
## X:Monthjul      -0.66622441   1.40724489  -0.473  0.63718
## X:Monthjun       5.19401119   3.91680068   1.326  0.18854
## X:Monthmar      -2.20541063   1.14755805  -1.922  0.05814 .
## X:Monthmay               NA           NA      NA       NA
## X:Monthoct     -14.82384611   4.67470558  -3.171  0.00215 **
## X:Monthsep               NA           NA      NA       NA
## X:Daymon         0.18085891   0.23698447   0.763  0.44758
## X:Daysat         0.12586608   0.23151382   0.544  0.58816
## X:Daysun         0.17642763   0.22300948   0.791  0.43118
## X:Daythu         0.18715374   0.27143091   0.690  0.49247
## X:Daytue         0.24035504   0.22510105   1.068  0.28880
## X:Daywed         0.23117651   0.22755970   1.016  0.31270
## X:FFMC          -0.04696192   0.05337853  -0.880  0.38158
## X:DMC            0.00146553   0.00202492   0.724  0.47131
## X:DC            -0.00225194   0.00174731  -1.289  0.20113
## X:ISI            0.00535729   0.03394400   0.158  0.87499
## X:Temp          -0.01262891   0.02679792  -0.471  0.63872
## X:RH            -0.00970648   0.00940719  -1.032  0.30523
## X:Wind           0.01385336   0.04868232   0.285  0.77670
## X:Rain          -1.21756142   1.02706015  -1.185  0.23929
## Y:Monthaug      -5.05552074   9.73517957  -0.519  0.60496
## Y:Monthdec     -10.48696503  14.25477999  -0.736  0.46405
## Y:Monthfeb      -3.19240421  10.60444760  -0.301  0.76415
## Y:Monthjul      -5.55619033  10.14499765  -0.548  0.58542
## Y:Monthjun       7.79662782  12.95605231   0.602  0.54900
## Y:Monthmar      -2.41904366   9.54271840  -0.253  0.80053
```

```
## Y:Monthmay                NA           NA      NA       NA
## Y:Monthoct        -2.42358300    9.86601242  -0.246  0.80657
## Y:Monthsep        -5.89137840    9.81447987  -0.600  0.55000
## Y:Daymon          -1.31373044    0.55027978  -2.387  0.01930 *
## Y:Daysat          -0.75992880    0.54160940  -1.403  0.16441
## Y:Daysun          -0.53742710    0.58119633  -0.925  0.35787
## Y:Daythu          -1.17946495    0.56730469  -2.079  0.04078 *
## Y:Daytue          -0.77325216    0.51193198  -1.510  0.13482
## Y:Daywed          -1.11152065    0.58363165  -1.904  0.06040 .
## Y:FFMC             0.01487561    0.13483525   0.110  0.91243
## Y:DMC             -0.01012851    0.00408478  -2.480  0.01523 *
## Y:DC               0.00277892    0.00308185   0.902  0.36989
## Y:ISI             -0.09810312    0.06641460  -1.477  0.14352
## Y:Temp             0.14338858    0.07412208   1.934  0.05654 .
## Y:RH               0.03780360    0.02382060   1.587  0.11641
## Y:Wind             0.21295705    0.11301114   1.884  0.06310 .
## Y:Rain                    NA           NA      NA       NA
## Monthaug:Daymon   33.83152255   21.38118918   1.582  0.11748
## Monthdec:Daymon   27.18485668   22.29951974   1.219  0.22635
## Monthfeb:Daymon    6.24667492   18.54426691   0.337  0.73710
## Monthjul:Daymon   36.35337616   21.04826246   1.727  0.08795 .
## Monthjun:Daymon   27.73556841   24.53050851   1.131  0.26154
## Monthmar:Daymon   23.55930997   21.46650030   1.097  0.27568
## Monthmay:Daymon           NA           NA      NA       NA
## Monthoct:Daymon   25.89673926   20.53290183   1.261  0.21085
## Monthsep:Daymon   32.67402639   21.46941530   1.522  0.13193
## Monthaug:Daysat   -0.56458862    1.84614290  -0.306  0.76053
## Monthdec:Daysat           NA           NA      NA       NA
## Monthfeb:Daysat  -93.00336126   62.22852156  -1.495  0.13892
## Monthjul:Daysat    0.60866630    7.01998909   0.087  0.93112
## Monthjun:Daysat  -12.70335472   13.68994700  -0.928  0.35620
## Monthmar:Daysat  -14.94490238    7.47976520  -1.998  0.04907 *
## Monthmay:Daysat           NA           NA      NA       NA
## Monthoct:Daysat  -51.70774907   16.81280058  -3.075  0.00287 **
## Monthsep:Daysat           NA           NA      NA       NA
## Monthaug:Daysun    1.02986755    2.24015479   0.460  0.64694
## Monthdec:Daysun    9.70563488    7.51541977   1.291  0.20023
## Monthfeb:Daysun  -41.74295314   24.58010407  -1.698  0.09330 .
## Monthjul:Daysun    0.26084253    6.23561984   0.042  0.96674
## Monthjun:Daysun   -0.53387324   12.75143139  -0.042  0.96671
## Monthmar:Daysun   -4.63204254    9.10893447  -0.509  0.61247
## Monthmay:Daysun           NA           NA      NA       NA
## Monthoct:Daysun           NA           NA      NA       NA
## Monthsep:Daysun           NA           NA      NA       NA
## Monthaug:Daythu   -0.11951115    2.74261294  -0.044  0.96535
## Monthdec:Daythu   -1.83461660    9.14405266  -0.201  0.84149
## Monthfeb:Daythu           NA           NA      NA       NA
## Monthjul:Daythu    1.71165629    7.15303759   0.239  0.81148
## Monthjun:Daythu   45.42368200   33.05117302   1.374  0.17313
## Monthmar:Daythu   -8.32818208   11.58513595  -0.719  0.47429
## Monthmay:Daythu           NA           NA      NA       NA
## Monthoct:Daythu           NA           NA      NA       NA
## Monthsep:Daythu           NA           NA      NA       NA
## Monthaug:Daytue    2.13599825    2.61346770   0.817  0.41615
```

```
## Monthdec:Daytue            NA            NA      NA       NA
## Monthfeb:Daytue  -89.88281584   96.13471389  -0.935  0.35258
## Monthjul:Daytue    3.24566253    6.85186232   0.474  0.63699
## Monthjun:Daytue            NA            NA      NA       NA
## Monthmar:Daytue            NA            NA      NA       NA
## Monthmay:Daytue            NA            NA      NA       NA
## Monthoct:Daytue            NA            NA      NA       NA
## Monthsep:Daytue            NA            NA      NA       NA
## Monthaug:Daywed    2.87501199    2.37649255   1.210  0.22989
## Monthdec:Daywed   11.64435451   10.80060633   1.078  0.28418
## Monthfeb:Daywed  -11.39505179   12.03864508  -0.947  0.34669
## Monthjul:Daywed   10.87832302    8.25656400   1.318  0.19137
## Monthjun:Daywed            NA            NA      NA       NA
## Monthmar:Daywed   -3.22243119   10.13976287  -0.318  0.75145
## Monthmay:Daywed            NA            NA      NA       NA
## Monthoct:Daywed            NA            NA      NA       NA
## Monthsep:Daywed            NA            NA      NA       NA
## Monthaug:FFMC     -0.54926611    0.50273601  -1.093  0.27783
## Monthdec:FFMC              NA            NA      NA       NA
## Monthfeb:FFMC    -11.27445757   10.90781708  -1.034  0.30439
## Monthjul:FFMC     -1.92280834    1.61214699  -1.193  0.23647
## Monthjun:FFMC     -5.83844902    7.72522580  -0.756  0.45198
## Monthmar:FFMC     -4.11043231    2.39577651  -1.716  0.09004 .
## Monthmay:FFMC              NA            NA      NA       NA
## Monthoct:FFMC              NA            NA      NA       NA
## Monthsep:FFMC              NA            NA      NA       NA
## Monthaug:DMC       0.01457739    0.01543078   0.945  0.34762
## Monthdec:DMC               NA            NA      NA       NA
## Monthfeb:DMC       2.94231895    3.09649546   0.950  0.34483
## Monthjul:DMC       0.05005329    0.05907341   0.847  0.39932
## Monthjun:DMC               NA            NA      NA       NA
## Monthmar:DMC       1.18457834    0.37291275   3.177  0.00211 **
## Monthmay:DMC               NA            NA      NA       NA
## Monthoct:DMC               NA            NA      NA       NA
## Monthsep:DMC               NA            NA      NA       NA
## Monthaug:DC       -0.00343144    0.01071129  -0.320  0.74952
## Monthdec:DC                NA            NA      NA       NA
## Monthfeb:DC                NA            NA      NA       NA
## Monthjul:DC       -0.00716278    0.02207915  -0.324  0.74646
## Monthjun:DC                NA            NA      NA       NA
## Monthmar:DC       -0.67635743    0.21760280  -3.108  0.00260 **
## Monthmay:DC                NA            NA      NA       NA
## Monthoct:DC                NA            NA      NA       NA
## Monthsep:DC                NA            NA      NA       NA
## Monthaug:ISI       0.20368658    0.32495789   0.627  0.53255
## Monthdec:ISI               NA            NA      NA       NA
## Monthfeb:ISI               NA            NA      NA       NA
## Monthjul:ISI       1.09782784    1.47324736   0.745  0.45832
## Monthjun:ISI               NA            NA      NA       NA
## Monthmar:ISI      -0.30092724    1.82834946  -0.165  0.86968
## Monthmay:ISI               NA            NA      NA       NA
## Monthoct:ISI               NA            NA      NA       NA
## Monthsep:ISI               NA            NA      NA       NA
## Monthaug:Temp     -0.05578774    0.25826039  -0.216  0.82952
```

```
## Monthdec:Temp             NA           NA      NA       NA
## Monthfeb:Temp             NA           NA      NA       NA
## Monthjul:Temp      0.51309801   0.82219569   0.624  0.53434
## Monthjun:Temp             NA           NA      NA       NA
## Monthmar:Temp     -0.03349530   1.06660897  -0.031  0.97502
## Monthmay:Temp             NA           NA      NA       NA
## Monthoct:Temp             NA           NA      NA       NA
## Monthsep:Temp             NA           NA      NA       NA
## Monthaug:RH        0.00761688   0.07627374   0.100  0.92070
## Monthdec:RH               NA           NA      NA       NA
## Monthfeb:RH               NA           NA      NA       NA
## Monthjul:RH        0.12365800   0.16466853   0.751  0.45486
## Monthjun:RH               NA           NA      NA       NA
## Monthmar:RH       -0.25314309   0.30108389  -0.841  0.40295
## Monthmay:RH               NA           NA      NA       NA
## Monthoct:RH               NA           NA      NA       NA
## Monthsep:RH               NA           NA      NA       NA
## Monthaug:Wind     -0.09351636   0.34443479  -0.272  0.78669
## Monthdec:Wind             NA           NA      NA       NA
## Monthfeb:Wind             NA           NA      NA       NA
## Monthjul:Wind      0.03128830   2.11716867   0.015  0.98825
## Monthjun:Wind             NA           NA      NA       NA
## Monthmar:Wind      1.38324768   1.63925225   0.844  0.40125
## Monthmay:Wind             NA           NA      NA       NA
## Monthoct:Wind             NA           NA      NA       NA
## Monthsep:Wind             NA           NA      NA       NA
## Monthaug:Rain             NA           NA      NA       NA
## Monthdec:Rain             NA           NA      NA       NA
## Monthfeb:Rain             NA           NA      NA       NA
## Monthjul:Rain             NA           NA      NA       NA
## Monthjun:Rain             NA           NA      NA       NA
## Monthmar:Rain             NA           NA      NA       NA
## Monthmay:Rain             NA           NA      NA       NA
## Monthoct:Rain             NA           NA      NA       NA
## Monthsep:Rain             NA           NA      NA       NA
## Daymon:FFMC       -0.41659310   0.48426547  -0.860  0.39219
## Daysat:FFMC       -1.11536782   0.49631351  -2.247  0.02734 *
## Daysun:FFMC       -0.43095127   0.46133957  -0.934  0.35301
## Daythu:FFMC       -0.96811436   0.66213678  -1.462  0.14758
## Daytue:FFMC       -0.14298070   0.42373292  -0.337  0.73666
## Daywed:FFMC       -0.25084040   0.58850988  -0.426  0.67107
## Daymon:DMC        -0.02465682   0.02438057  -1.011  0.31487
## Daysat:DMC         0.01643812   0.01325760   1.240  0.21859
## Daysun:DMC         0.00424869   0.01464943   0.290  0.77254
## Daythu:DMC         0.02975890   0.02299194   1.294  0.19923
## Daytue:DMC        -0.00787034   0.02666601  -0.295  0.76864
## Daywed:DMC         0.00306651   0.01779501   0.172  0.86361
## Daymon:DC          0.01087074   0.01398719   0.777  0.43931
## Daysat:DC         -0.00596480   0.01112643  -0.536  0.59336
## Daysun:DC          0.00845265   0.01519443   0.556  0.57954
## Daythu:DC          0.00926184   0.01661058   0.558  0.57866
## Daytue:DC          0.00711711   0.01369232   0.520  0.60463
## Daywed:DC          0.02090233   0.01432541   1.459  0.14840
## Daymon:ISI        -0.37945741   0.32906770  -1.153  0.25225
```

```
## Daysat:ISI         0.35891670   0.30472814   1.178  0.24231
## Daysun:ISI         0.21786899   0.26728618   0.815  0.41740
## Daythu:ISI         0.33120383   0.31362197   1.056  0.29408
## Daytue:ISI         0.00859787   0.22391119   0.038  0.96946
## Daywed:ISI        -0.19512671   0.31590676  -0.618  0.53852
## Daymon:Temp        0.37055000   0.30476607   1.216  0.22757
## Daysat:Temp       -0.26281956   0.25602973  -1.027  0.30770
## Daysun:Temp       -0.21431467   0.27778899  -0.772  0.44265
## Daythu:Temp        0.04264391   0.26641419   0.160  0.87323
## Daytue:Temp       -0.16243404   0.25368046  -0.640  0.52378
## Daywed:Temp       -0.08333271   0.28374473  -0.294  0.76975
## Daymon:RH         -0.00339175   0.10525330  -0.032  0.97437
## Daysat:RH         -0.10352771   0.06261142  -1.653  0.10210
## Daysun:RH         -0.09226287   0.06662947  -1.385  0.16994
## Daythu:RH         -0.01305121   0.07113000  -0.183  0.85488
## Daytue:RH         -0.05697769   0.07334264  -0.777  0.43950
## Daywed:RH         -0.03779627   0.07397298  -0.511  0.61078
## Daymon:Wind        1.15676913   0.71849018   1.610  0.11129
## Daysat:Wind        0.44194407   0.28234561   1.565  0.12142
## Daysun:Wind       -0.02175680   0.27789766  -0.078  0.93779
## Daythu:Wind        0.15236402   0.48055733   0.317  0.75202
## Daytue:Wind        0.08562364   0.32884200   0.260  0.79523
## Daywed:Wind        0.33737910   0.32090381   1.051  0.29623
## Daymon:Rain                NA           NA      NA       NA
## Daysat:Rain                NA           NA      NA       NA
## Daysun:Rain                NA           NA      NA       NA
## Daythu:Rain                NA           NA      NA       NA
## Daytue:Rain                NA           NA      NA       NA
## Daywed:Rain                NA           NA      NA       NA
## FFMC:DMC           0.01052074   0.00378462   2.780  0.00676 **
## FFMC:DC           -0.00646666   0.00369525  -1.750  0.08391 .
## FFMC:ISI           0.00016163   0.02647076   0.006  0.99514
## FFMC:Temp         -0.07505406   0.06154516  -1.219  0.22620
## FFMC:RH           -0.02404065   0.01712445  -1.404  0.16418
## FFMC:Wind          0.06736224   0.08114005   0.830  0.40887
## FFMC:Rain                  NA           NA      NA       NA
## DMC:DC             0.00003410   0.00007245   0.471  0.63913
## DMC:ISI           -0.00254057   0.00280253  -0.907  0.36735
## DMC:Temp          -0.00268603   0.00284039  -0.946  0.34714
## DMC:RH            -0.00044992   0.00080367  -0.560  0.57714
## DMC:Wind          -0.00364756   0.00351994  -1.036  0.30317
## DMC:Rain                   NA           NA      NA       NA
## DC:ISI             0.00278555   0.00226906   1.228  0.22314
## DC:Temp            0.00045321   0.00162003   0.280  0.78038
## DC:RH             -0.00004568   0.00052450  -0.087  0.93081
## DC:Wind            0.00352747   0.00253575   1.391  0.16800
## DC:Rain                    NA           NA      NA       NA
## ISI:Temp           0.06313608   0.03570054   1.768  0.08074 .
## ISI:RH             0.01521950   0.01051065   1.448  0.15147
## ISI:Wind           0.01619649   0.04450185   0.364  0.71684
## ISI:Rain                   NA           NA      NA       NA
## Temp:RH           -0.00278541   0.00327044  -0.852  0.39690
## Temp:Wind          0.02150262   0.04150480   0.518  0.60582
## Temp:Rain                  NA           NA      NA       NA
```

```
## RH:Wind            0.00549365    0.01287335   0.427  0.67070
## RH:Rain                  NA            NA      NA       NA
## Wind:Rain                NA            NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.179 on 81 degrees of freedom
## Multiple R-squared:  0.7353, Adjusted R-squared:  0.1209
## F-statistic: 1.197 on 188 and 81 DF,  p-value: 0.1796
```

```
#choosing m3 as the best fit model because of acceptable F-statistics
#Perform regression diagnostics using both typical approach and enhanced approach
#typical approach
fit_4<- lm( formula = Area ~ (.^2), data = ff)
summary(m3)
```

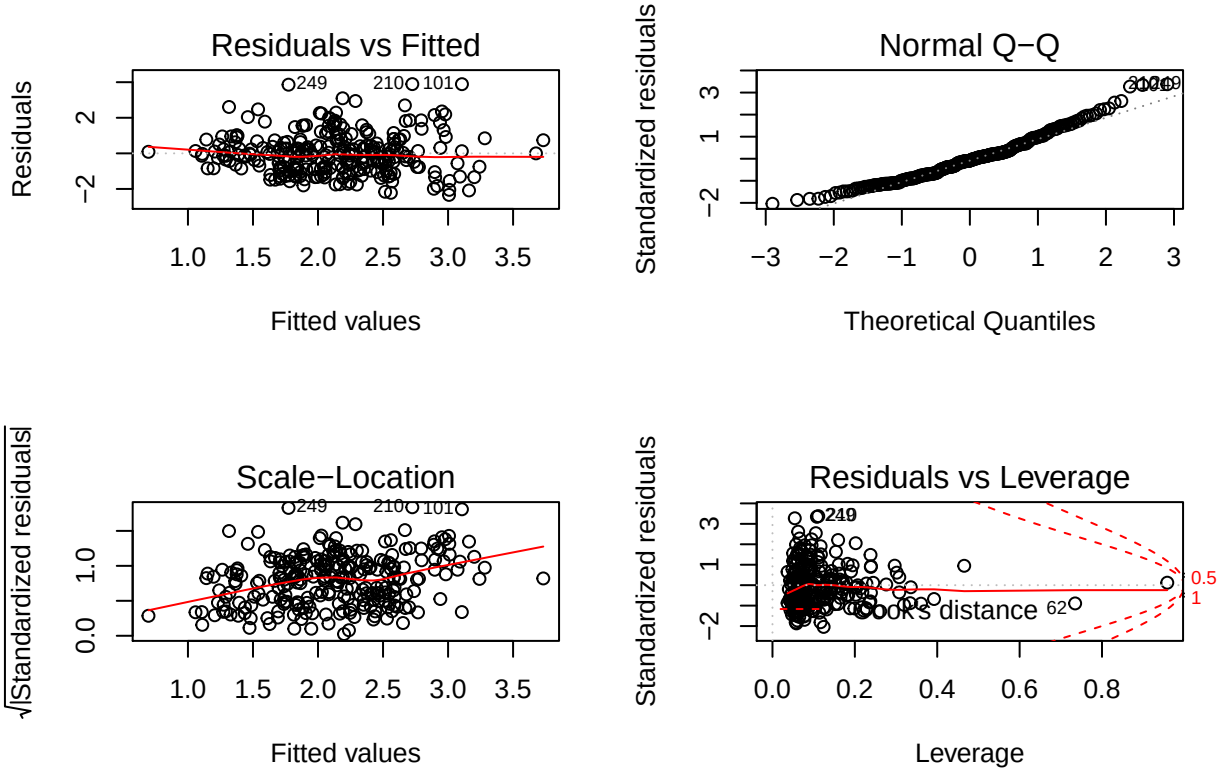
```
##
## Call:
## lm(formula = Area ~ . + (FFMC + DMC + DC + ISI)^2, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3344 -0.8487 -0.0889  0.6421  3.8913
##
## Coefficients:
##                 Estimate   Std. Error t value Pr(>|t|)
## (Intercept)  0.025083039  7.769016540   0.003   0.9974
## X            0.058121518  0.038910093   1.494   0.1366
## Y           -0.089989159  0.079559799  -1.131   0.2592
## Monthaug    -1.019193769  1.478217830  -0.689   0.4912
## Monthdec     0.321528017  1.096554788   0.293   0.7696
## Monthfeb     0.000020467  0.775983595   0.000   1.0000
## Monthjul    -1.381747896  1.302473009  -1.061   0.2898
## Monthjun    -1.737034150  1.198743471  -1.449   0.1486
## Monthmar    -0.959346103  0.776475192  -1.236   0.2179
## Monthmay     1.019206865  1.442173418   0.707   0.4804
## Monthoct     1.156137457  1.640343407   0.705   0.4816
## Monthsep     0.263000560  1.580112707   0.166   0.8679
## Daymon      -0.040794830  0.297791187  -0.137   0.8912
## Daysat       0.547536282  0.280436938   1.952   0.0521 .
## Daysun       0.350641357  0.275435459   1.273   0.2042
## Daythu       0.113808816  0.312163554   0.365   0.7157
## Daytue       0.287523780  0.294062635   0.978   0.3292
## Daywed       0.094834661  0.306650473   0.309   0.7574
## FFMC         0.020611132  0.092411100   0.223   0.8237
## DMC         -0.101070707  0.096152866  -1.051   0.2943
## DC           0.020419577  0.017514510   1.166   0.2448
## ISI         -0.058627322  0.781752416  -0.075   0.9403
## Temp         0.008116885  0.030453630   0.267   0.7901
## RH          -0.002131796  0.008203085  -0.260   0.7952
## Wind         0.043072906  0.049209372   0.875   0.3823
## Rain        -0.026142283  0.199077683  -0.131   0.8956
## FFMC:DMC     0.001420127  0.001073270   1.323   0.1870
## FFMC:DC     -0.000236990  0.000203556  -1.164   0.2455
## FFMC:ISI     0.000787480  0.008340830   0.094   0.9249
```

```
## DMC:DC      -0.000026532  0.000013921  -1.906   0.0579 .
## DMC:ISI     -0.000005988  0.000698895  -0.009   0.9932
## DC:ISI      -0.000078807  0.000201064  -0.392   0.6954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.222 on 238 degrees of freedom
## Multiple R-squared:  0.1643, Adjusted R-squared:  0.05549
## F-statistic:  1.51 on 31 and 238 DF,  p-value: 0.04722
```

```r
library(MASS)
par(mfrow = c(2,2))
plot(m3)
```

```
## Warning: not plotting observations with leverage one:
##   241
```

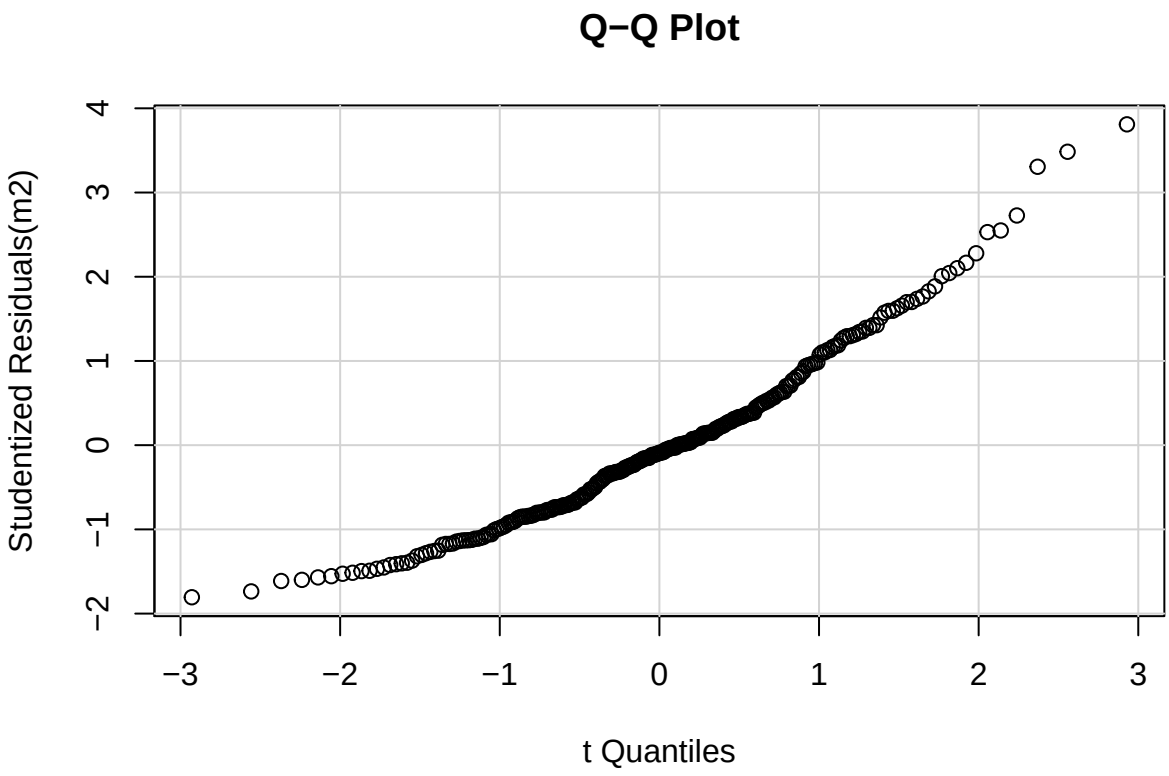
```
## Warning: not plotting observations with leverage one:
##   241
```




```r
#enhanced approach
#normality
par(mfrow = c(1,1))
qqPlot(m2, labels = row.names(ff), id.method = "identify", simulate = TRUE, main = "Q-Q Plot")
```

```
## Warning in matrix(yhat, n, reps): data length [270] is not a sub-multiple or
## multiple of the number of rows [269]
```

```
## Error in model.frame.default(formula = Y ~ X - 1, drop.unused.levels = TRUE): variable lengths diffe
```




```
#Based on this graph, we can see the points fall close to the line but a little bit beyond the confiden

#Independence
durbinWatsonTest(m3)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.4367686      1.123331       0
##  Alternative hypothesis: rho != 0
```

```
#Linearity
crPlots(m3)
```

```
## Error in crPlots(m3): C+R plots not available for models with interactions.
```

```
return()
```

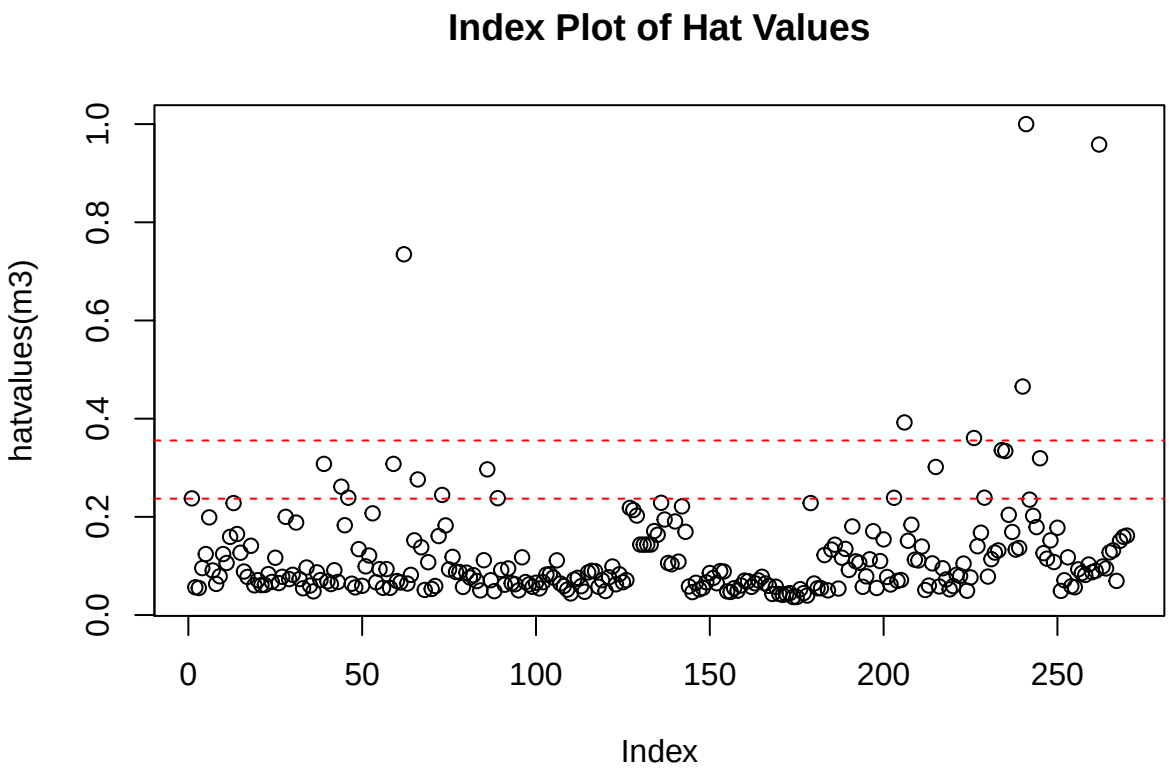
```
## NULL
```

```
#Homoscedasticity

ncvTest(m3)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 12.90031, Df = 1, p = 0.00032853
```

```
#4. Identify unusual observations and take corrective measures.
#High Leverage pointa
hat.plot<-function(m3) {
  p<-length(coefficients(m3))
```

```
  n<-length(fitted(m3))
  plot(hatvalues(m3), main="Index Plot of Hat Values")
  abline(h=c(2,3)*p/n,col="red", lty=2)
  identify(1:n,hatvalues(m3), names(hatvalues(m3)))
}
hat.plot(m3)
```

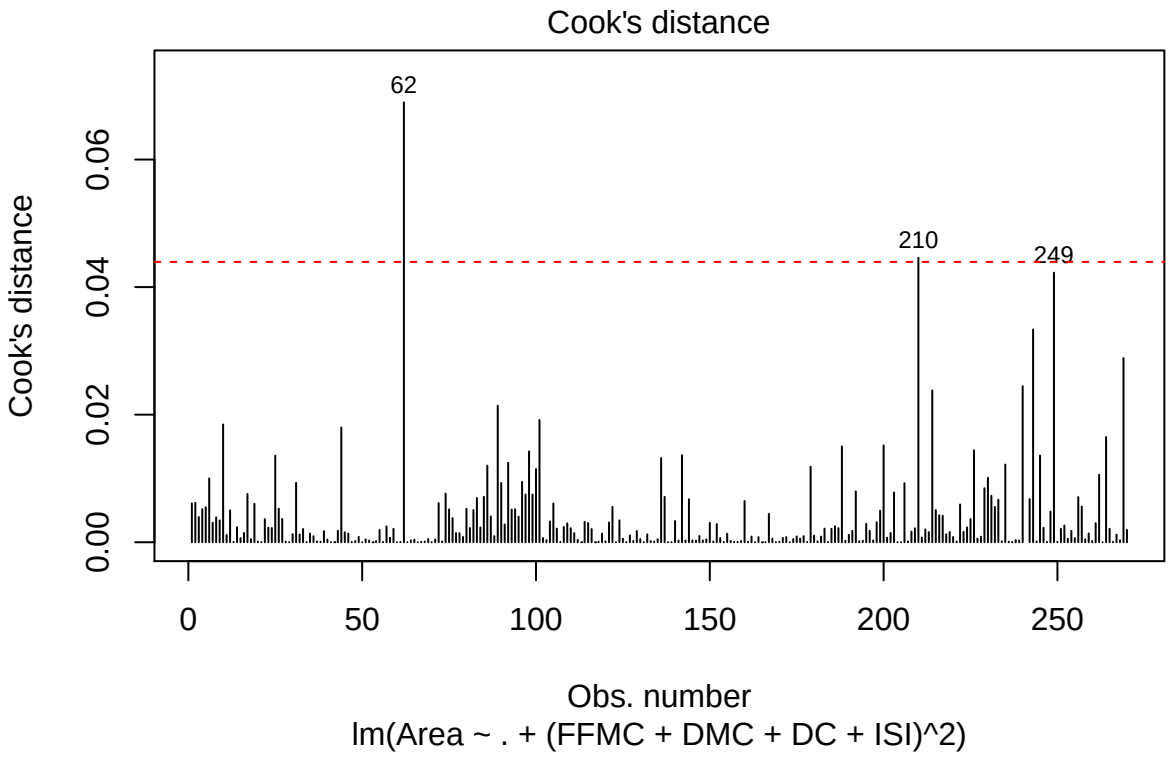



```
## integer(0)
```

```
#There are three obvious outliers above the red line (54, 240, 256 respectively).
#Influential observation
cutoff<-4/(nrow(ff)-length(reg1$coefficients)-2)
```

```
## Error in eval(expr, envir, enclos): object 'reg1' not found
```

```
plot(m3,which=4,cook.levels=cutoff)
abline(h=cutoff,lty=2,col="red")
```




```
# 5. Select the best regression model.
#Comparing nested models using the anova() function

anova(m2, m3)
```

```
## Analysis of Variance Table
##
## Model 1: Area ~ (X + Y + Month + Day + FFMC + DMC + DC + ISI + Temp +
##     RH + Wind + Rain) - Temp - RH - Wind - Rain
## Model 2: Area ~ X + Y + Month + Day + FFMC + DMC + DC + ISI + Temp + RH +
##     Wind + Rain + (FFMC + DMC + DC + ISI)^2
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    248 376.57
## 2    238 355.40 10    21.166 1.4174 0.1731
```

```
#Since the p value is comparatively, we can conclude that the new model add linear predicion and we wil
#6. Fine tune the selection of predictor variables.
m3 <- step(m0, direction = "backward")
```

```
## Start:  AIC=135.19
## Area ~ X + Y + FFMC + DMC + DC + ISI + Temp + RH + Wind + Rain
##
##        Df Sum of Sq    RSS    AIC
## - FFMC  1    0.0721 410.69 133.24
## - Rain  1    0.0786 410.69 133.24
## - Temp  1    0.1458 410.76 133.29
## - DC    1    0.9172 411.53 133.79
```

```
## - Y     1    1.1340 411.75 133.94
## - X     1    1.1474 411.76 133.95
## - Wind  1    1.7430 412.36 134.34
## <none>              410.61 135.19
## - RH    1    3.6463 414.26 135.58
## - ISI   1    4.0854 414.70 135.87
## - DMC   1    4.7819 415.40 136.32
##
## Step:  AIC=133.24
## Area ~ X + Y + DMC + DC + ISI + Temp + RH + Wind + Rain
##
##        Df Sum of Sq    RSS    AIC
## - Rain  1    0.0906 410.78 131.30
## - Temp  1    0.1439 410.83 131.33
## - DC    1    0.8914 411.58 131.82
## - Y     1    1.1380 411.82 131.99
## - X     1    1.1561 411.84 132.00
## - Wind  1    1.6710 412.36 132.34
## <none>              410.69 133.24
## - RH    1    4.1046 414.79 133.93
## - DMC   1    5.4157 416.10 134.78
## - ISI   1    5.7045 416.39 134.96
##
## Step:  AIC=131.3
## Area ~ X + Y + DMC + DC + ISI + Temp + RH + Wind
##
##        Df Sum of Sq    RSS    AIC
## - Temp  1    0.1156 410.89 129.38
## - DC    1    0.9015 411.68 129.89
## - Y     1    1.1344 411.91 130.04
## - X     1    1.1865 411.96 130.08
## - Wind  1    1.7449 412.52 130.44
## <none>              410.78 131.30
## - RH    1    4.0145 414.79 131.93
## - DMC   1    5.4049 416.18 132.83
## - ISI   1    5.7094 416.49 133.03
##
## Step:  AIC=129.38
## Area ~ X + Y + DMC + DC + ISI + RH + Wind
##
##        Df Sum of Sq    RSS    AIC
## - DC    1    1.0413 411.93 128.06
## - Y     1    1.1476 412.04 128.13
## - X     1    1.1684 412.06 128.14
## - Wind  1    2.2479 413.14 128.85
## <none>              410.89 129.38
## - RH    1    4.7619 415.65 130.49
## - DMC   1    5.4870 416.38 130.96
## - ISI   1    7.5352 418.43 132.28
##
## Step:  AIC=128.06
## Area ~ X + Y + DMC + ISI + RH + Wind
##
##        Df Sum of Sq    RSS    AIC
```

```
## - Y     1    1.0746 413.01 126.76
## - X     1    1.3612 413.29 126.95
## - Wind  1    2.9655 414.90 128.00
## <none>               411.93 128.06
## - RH    1    4.3901 416.32 128.92
## - DMC   1    4.7721 416.71 129.17
## - ISI   1    7.8666 419.80 131.17
##
## Step:  AIC=126.76
## Area ~ X + DMC + ISI + RH + Wind
##
##        Df Sum of Sq    RSS    AIC
## - X     1    0.5499 413.56 125.12
## - Wind  1    3.0631 416.07 126.76
## <none>               413.01 126.76
## - RH    1    4.0145 417.02 127.37
## - DMC   1    4.2747 417.28 127.54
## - ISI   1    7.3632 420.37 129.53
##
## Step:  AIC=125.12
## Area ~ DMC + ISI + RH + Wind
##
##        Df Sum of Sq    RSS    AIC
## <none>               413.56 125.12
## - Wind  1    3.1192 416.68 125.15
## - RH    1    3.8498 417.41 125.62
## - DMC   1    4.0326 417.59 125.74
## - ISI   1    7.4192 420.98 127.92
```

```
summary(m3)
```

```
##
## Call:
## lm(formula = Area ~ DMC + ISI + RH + Wind, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0840 -0.9176 -0.2069  0.6584  4.6994
##
## Coefficients:
##             Estimate Std. Error t value        Pr(>|t|)
## (Intercept)  2.395547   0.344855   6.947 0.0000000000289 ***
## DMC          0.002144   0.001334   1.607          0.1091
## ISI         -0.043542   0.019970  -2.180          0.0301 *
## RH          -0.008167   0.005200  -1.571          0.1175
## Wind         0.059001   0.041733   1.414          0.1586
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.249 on 265 degrees of freedom
## Multiple R-squared:  0.02759,    Adjusted R-squared:  0.01291
## F-statistic:  1.88 on 4 and 265 DF,  p-value: 0.1143
```

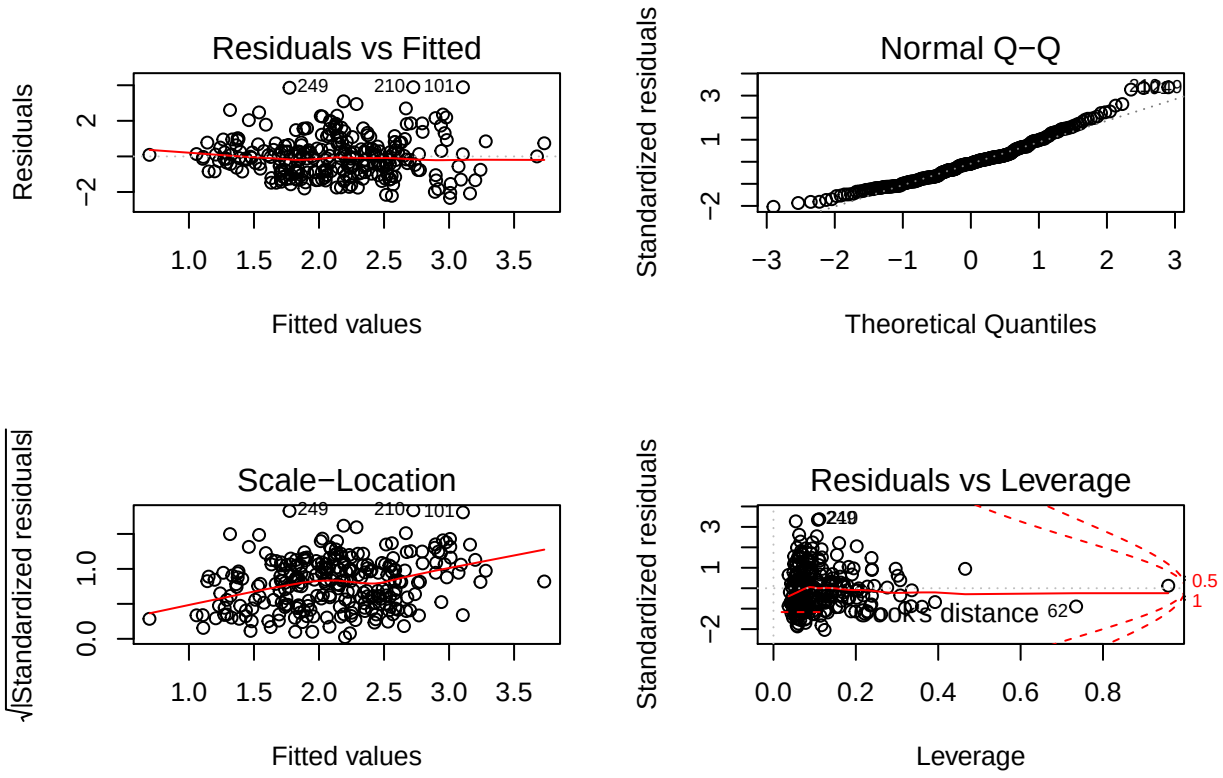
```
#7. Interpret the prediction results.
#Based on the previous the output of previous question.
```

```r
model_final <- lm( formula = Area ~ . + (FFMC + DMC + DC + ISI)^2, data = ff)
summary(model_final)
```

```
##
## Call:
## lm(formula = Area ~ . + (FFMC + DMC + DC + ISI)^2, data = ff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3344 -0.8487 -0.0889  0.6421  3.8913
##
## Coefficients:
##                 Estimate   Std. Error t value Pr(>|t|)
## (Intercept)  0.025083039  7.769016540   0.003   0.9974
## X            0.058121518  0.038910093   1.494   0.1366
## Y           -0.089989159  0.079559799  -1.131   0.2592
## Monthaug    -1.019193769  1.478217830  -0.689   0.4912
## Monthdec     0.321528017  1.096554788   0.293   0.7696
## Monthfeb     0.000020467  0.775983595   0.000   1.0000
## Monthjul    -1.381747896  1.302473009  -1.061   0.2898
## Monthjun    -1.737034150  1.198743471  -1.449   0.1486
## Monthmar    -0.959346103  0.776475192  -1.236   0.2179
## Monthmay     1.019206865  1.442173418   0.707   0.4804
## Monthoct     1.156137457  1.640343407   0.705   0.4816
## Monthsep     0.263000560  1.580112707   0.166   0.8679
## Daymon      -0.040794830  0.297791187  -0.137   0.8912
## Daysat       0.547536282  0.280436938   1.952   0.0521 .
## Daysun       0.350641357  0.275435459   1.273   0.2042
## Daythu       0.113808816  0.312163554   0.365   0.7157
## Daytue       0.287523780  0.294062635   0.978   0.3292
## Daywed       0.094834661  0.306650473   0.309   0.7574
## FFMC         0.020611132  0.092411100   0.223   0.8237
## DMC         -0.101070707  0.096152866  -1.051   0.2943
## DC           0.020419577  0.017514510   1.166   0.2448
## ISI         -0.058627322  0.781752416  -0.075   0.9403
## Temp         0.008116885  0.030453630   0.267   0.7901
## RH          -0.002131796  0.008203085  -0.260   0.7952
## Wind         0.043072906  0.049209372   0.875   0.3823
## Rain        -0.026142283  0.199077683  -0.131   0.8956
## FFMC:DMC     0.001420127  0.001073270   1.323   0.1870
## FFMC:DC     -0.000236990  0.000203556  -1.164   0.2455
## FFMC:ISI     0.000787480  0.008340830   0.094   0.9249
## DMC:DC      -0.000026532  0.000013921  -1.906   0.0579 .
## DMC:ISI     -0.000005988  0.000698895  -0.009   0.9932
## DC:ISI      -0.000078807  0.000201064  -0.392   0.6954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.222 on 238 degrees of freedom
## Multiple R-squared:  0.1643, Adjusted R-squared:  0.05549
## F-statistic:  1.51 on 31 and 238 DF,  p-value: 0.04722
```

```r
par(mfrow=c(2,2))
plot(model_final)
```

```
## Warning: not plotting observations with leverage one:
##   241

## Warning: not plotting observations with leverage one:
##   241
```




```
#The final formula will be: Area = (2.261e-02)FFMC + (-1.292e-01)DMC + (2.610e-02)DC + (1.654e-03)FFMC:
```