REGRESSION ANALYSIS

Bogdan Tanasa

11. REGRESSION ANALYSIS : using SPLINES

12. REGRESSION ANALYSIS : using GLM

13. REGRESSION ANALYSIS : MODEL SELECTION based on BIC

14. REGRESSION ANALYSIS : MODEL SELECTION based on AIC

15. REGRESSION ANALYSIS : using RIDGE REGRESSION

16. REGRESSION ANALYSIS : using LASSO REGRESSION

0. DATA INFORMATION

We are using the data that we had from **UCI** a while ago in the file : https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.

The aim is to use the 8 features of the concrete in order to predict the concrete compressive strength.

According to the web page at UC Irvine, the concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate.

Data Characteristics:

The actual concrete compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory. Data is in raw form (not scaled).

Summary Statistics:

Number of instances (observations): 1030 Number of Attributes: 9 Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable Missing Attribute Values: None

Variable Information:

Name – Data Type – Measurement – Description

Cement (component 1) – quantitative – kg in a m3 mixture – Input Variable

Blast Furnace Slag (component 2) – quantitative – kg in a m3 mixture – Input Variable

Fly Ash (component 3) – quantitative – kg in a m3 mixture – Input Variable

Water (component 4) – quantitative – kg in a m3 mixture – Input Variable

Superplasticizer (component 5) – quantitative – kg in a m3 mixture – Input Variable

Coarse Aggregate (component 6) – quantitative – kg in a m3 mixture – Input Variable

Fine Aggregate (component 7) – quantitative – kg in a m3 mixture – Input Variable

Age – quantitative – Day (1~365) – Input Variable

Concrete compressive strength – quantitative – MPa – Output Variable

```
options(warn=-1)
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(reshape2))
suppressPackageStartupMessages(library(readxl))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(tidyr))
suppressPackageStartupMessages(library(purrr))
```

```r
suppressPackageStartupMessages(library(ggpubr))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(tibble))
suppressPackageStartupMessages(library(class))
suppressPackageStartupMessages(library(gmodels))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(e1071))
suppressPackageStartupMessages(library(ISLR))
suppressPackageStartupMessages(library(pROC))
suppressPackageStartupMessages(library(lattice))
suppressPackageStartupMessages(library(kknn))
suppressPackageStartupMessages(library(multiROC))
suppressPackageStartupMessages(library(MLeval))
suppressPackageStartupMessages(library(AppliedPredictiveModeling))
suppressPackageStartupMessages(library(corrplot))
suppressPackageStartupMessages(library(Hmisc))
suppressPackageStartupMessages(library(rattle))
suppressPackageStartupMessages(library(Hmisc))
suppressPackageStartupMessages(library(broom)) # to add : AUGMENT
suppressPackageStartupMessages(library(rattle))
suppressPackageStartupMessages(library(quantmod))
suppressPackageStartupMessages(library(nnet))
suppressPackageStartupMessages(library(NeuralNetTools))
suppressPackageStartupMessages(library(neuralnet))
suppressPackageStartupMessages(library(klaR))
suppressPackageStartupMessages(library(kernlab))
suppressPackageStartupMessages(library(gridExtra))
suppressPackageStartupMessages(library(cluster))
suppressPackageStartupMessages(library(factoextra))
suppressPackageStartupMessages(library(magrittr))
suppressPackageStartupMessages(library(fpc))
suppressPackageStartupMessages(library(gplots))
suppressPackageStartupMessages(library(pheatmap))
# suppressPackageStartupMessages(library(d3heatmap))
suppressPackageStartupMessages(library(clValid))
suppressPackageStartupMessages(library(clustertend))
suppressPackageStartupMessages(library(factoextra))
suppressPackageStartupMessages(library(ggfortify))
suppressPackageStartupMessages(library(splines))
suppressPackageStartupMessages(library(mgcv))
suppressPackageStartupMessages(library(leaps))
suppressPackageStartupMessages(library(MASS))
suppressPackageStartupMessages(library(glmnet))
suppressPackageStartupMessages(library(car))
suppressPackageStartupMessages(library(MASS))

# suppressPackageStartupMessages(library(ggstatsplot))
#########################################################
#########################################################
FILE1="Concrete_Data.csv"
#########################################################
#########################################################
file = read.delim(FILE1, sep = ",", header=TRUE, stringsAsFactors=F)
```

```
#######################################################
#######################################################
str(file)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ Cement..component.1..kg.in.a.m.3.mixture.            : num  540 540 332 332 199 ...
##  $ Blast.Furnace.Slag..component.2..kg.in.a.m.3.mixture.: num  0 0 142 142 132 ...
##  $ Fly.Ash..component.3..kg.in.a.m.3.mixture.           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water...component.4..kg.in.a.m.3.mixture.            : num  162 162 228 228 192 228 228 228 228 2
##  $ Superplasticizer..component.5..kg.in.a.m.3.mixture.  : num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ Coarse.Aggregate...component.6..kg.in.a.m.3.mixture. : num  1040 1055 932 932 978 ...
##  $ Fine.Aggregate..component.7..kg.in.a.m.3.mixture.    : num  676 676 594 594 826 ...
##  $ Age..day.                                            : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ Concrete.compressive.strength.MPa..megapascals..     : num  80 61.9 40.3 41 44.3 ...
```

```
class(file)
```

```
## [1] "data.frame"
```

```
# summary(file)
# all the FEATURES are NUMERICAL
#######################################################
#######################################################
# we choose shorter NAMES for the FEATURES

names(file)[1] <- "Cement"
names(file)[2] <- "Blast.Furnace.Slag"
names(file)[3] <- "Fly.Ash"
names(file)[4] <- "Water"
names(file)[5] <- "Superplasticizer"
names(file)[6] <- "Coarse.Aggregate"
names(file)[7] <- "Fine.Aggregate"
names(file)[8] <- "Age"
names(file)[9] <- "CCS" #### Concrete.compressive.strength.MPa..megapascals

summary(file)
```

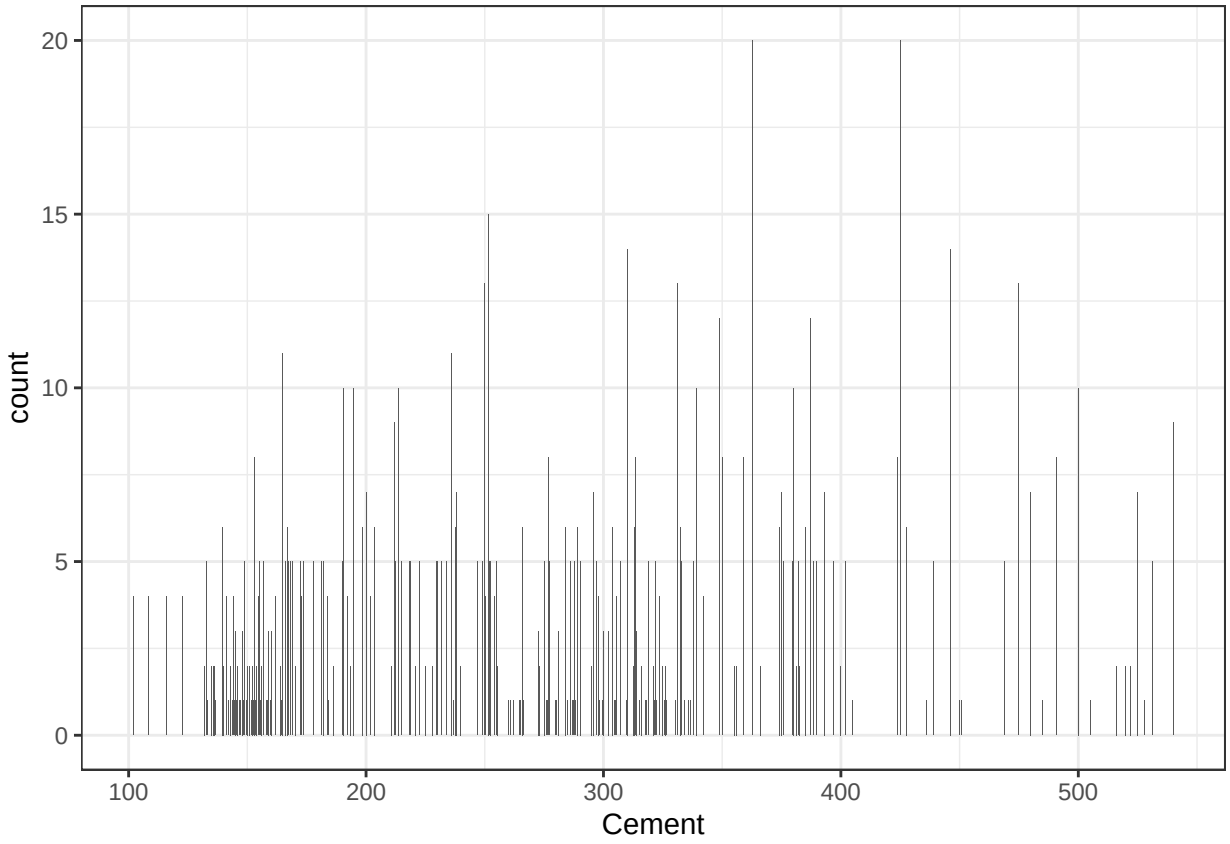
```
##      Cement      Blast.Furnace.Slag    Fly.Ash           Water
##  Min.   :102.0   Min.   :  0.0      Min.   :  0.00   Min.   :121.8
##  1st Qu.:192.4   1st Qu.:  0.0      1st Qu.:  0.00   1st Qu.:164.9
##  Median :272.9   Median : 22.0      Median :  0.00   Median :185.0
##  Mean   :281.2   Mean   : 73.9      Mean   : 54.19   Mean   :181.6
##  3rd Qu.:350.0   3rd Qu.:142.9      3rd Qu.:118.30   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Coarse.Aggregate Fine.Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.400   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.205   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.200   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##       CCS
##  Min.   : 2.33
##  1st Qu.:23.71
##  Median :34.45
```

```
##  Mean   :35.82
##  3rd Qu.:46.13
##  Max.   :82.60
```

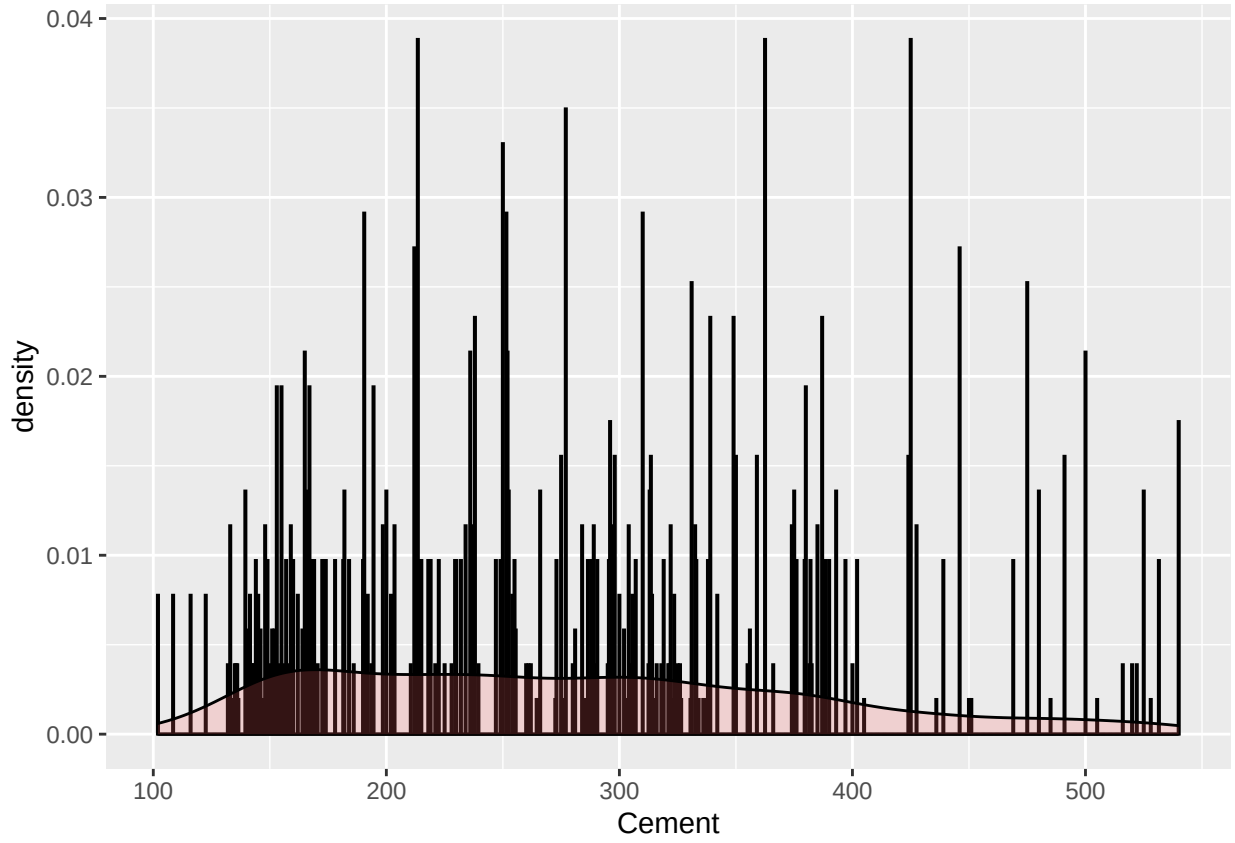
1. DATA EXPLORATION

```
# exploring each feature in the data
```

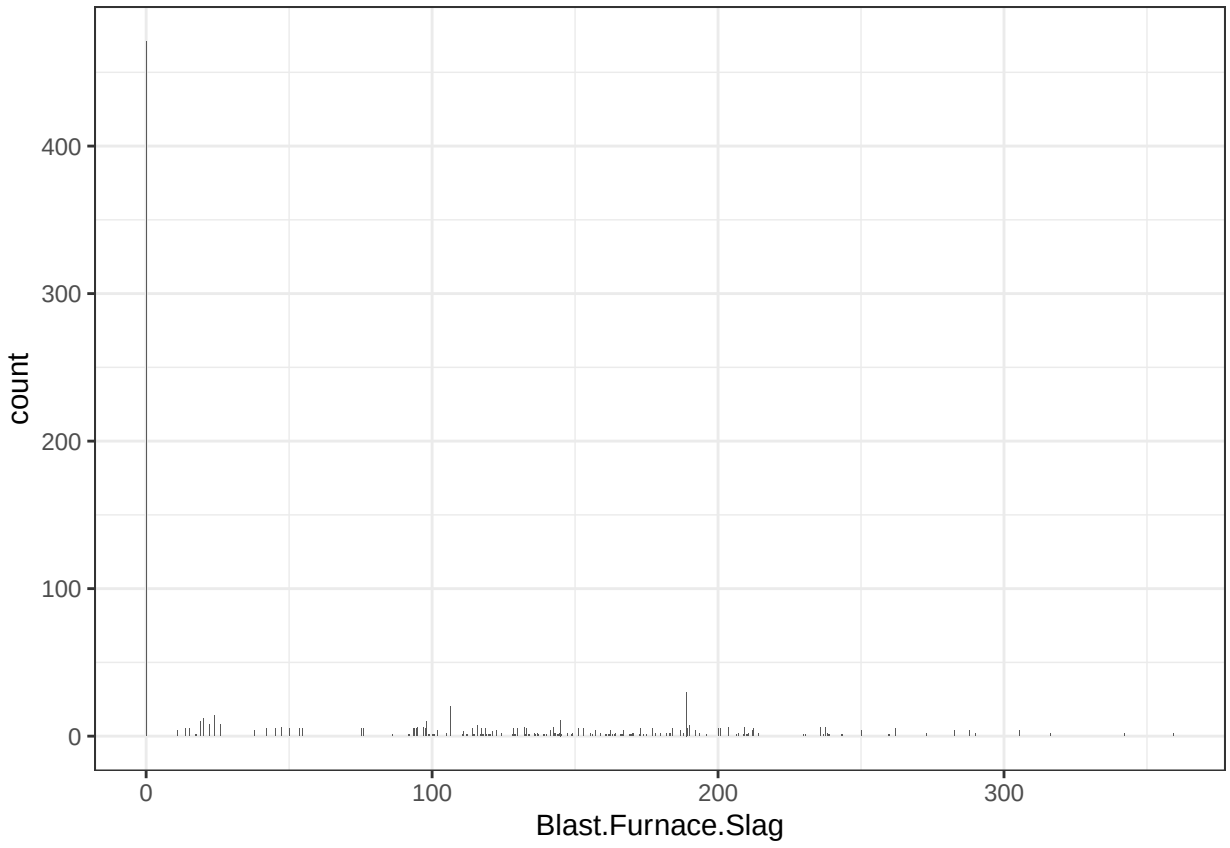
```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Cement, fill=Cement)) + theme_bw()
```



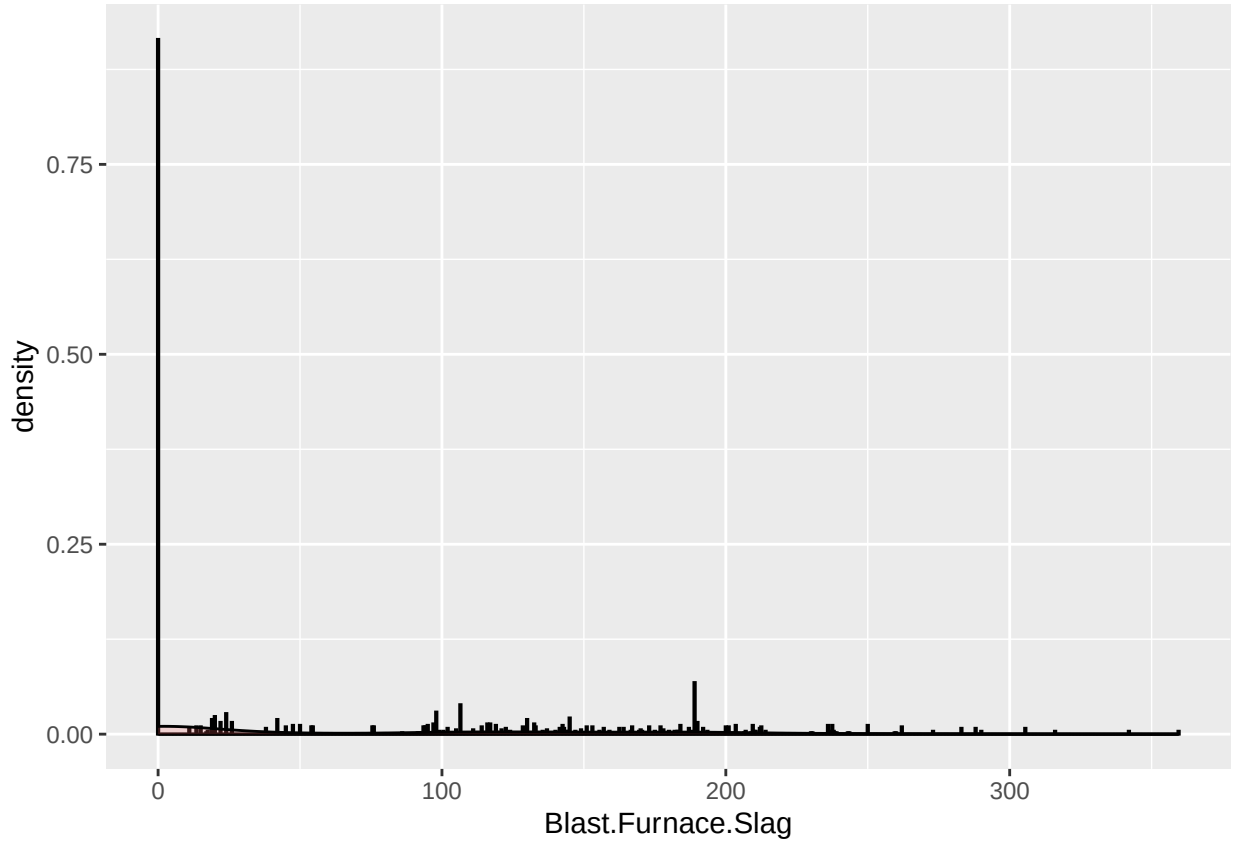

```
ggplot(file, aes(x=Cement)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



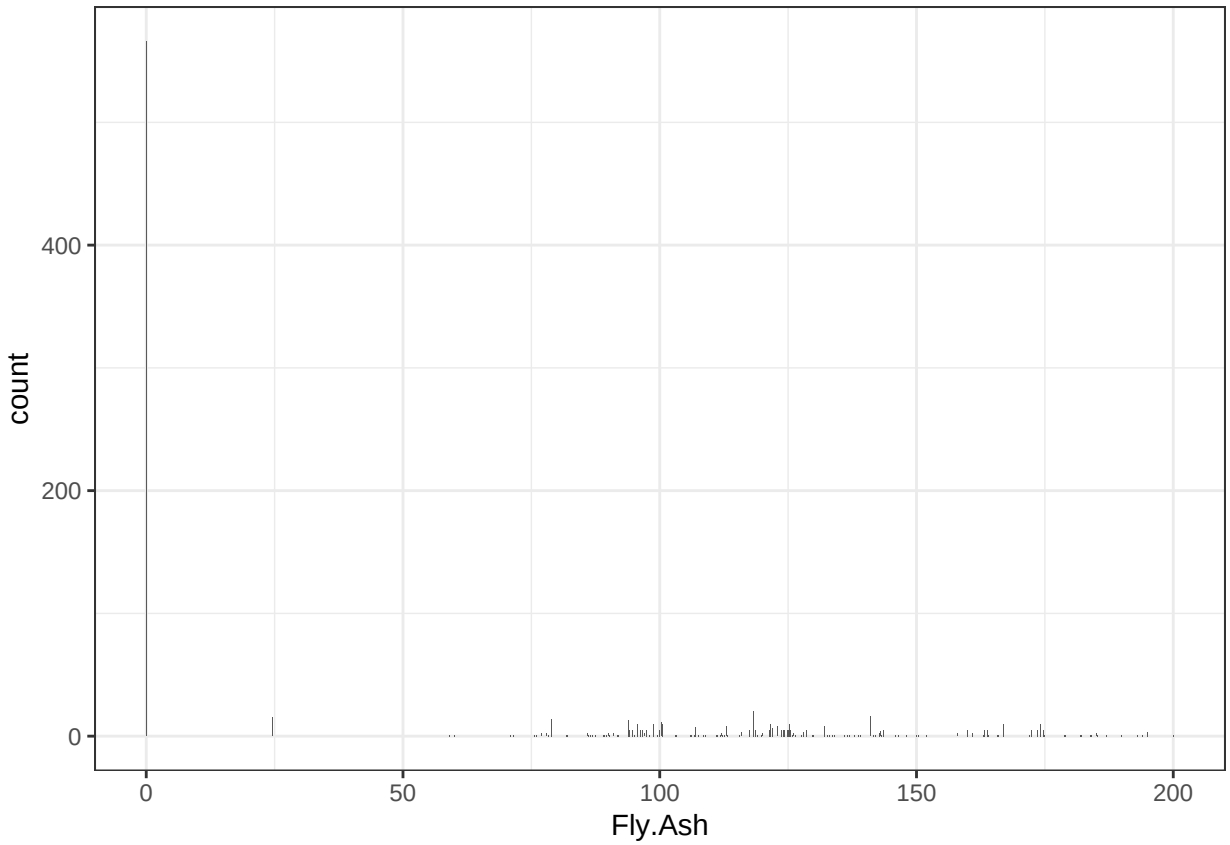

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Blast.Furnace.Slag, fill=Blast.Furnace.Slag)) + theme_bw()
```



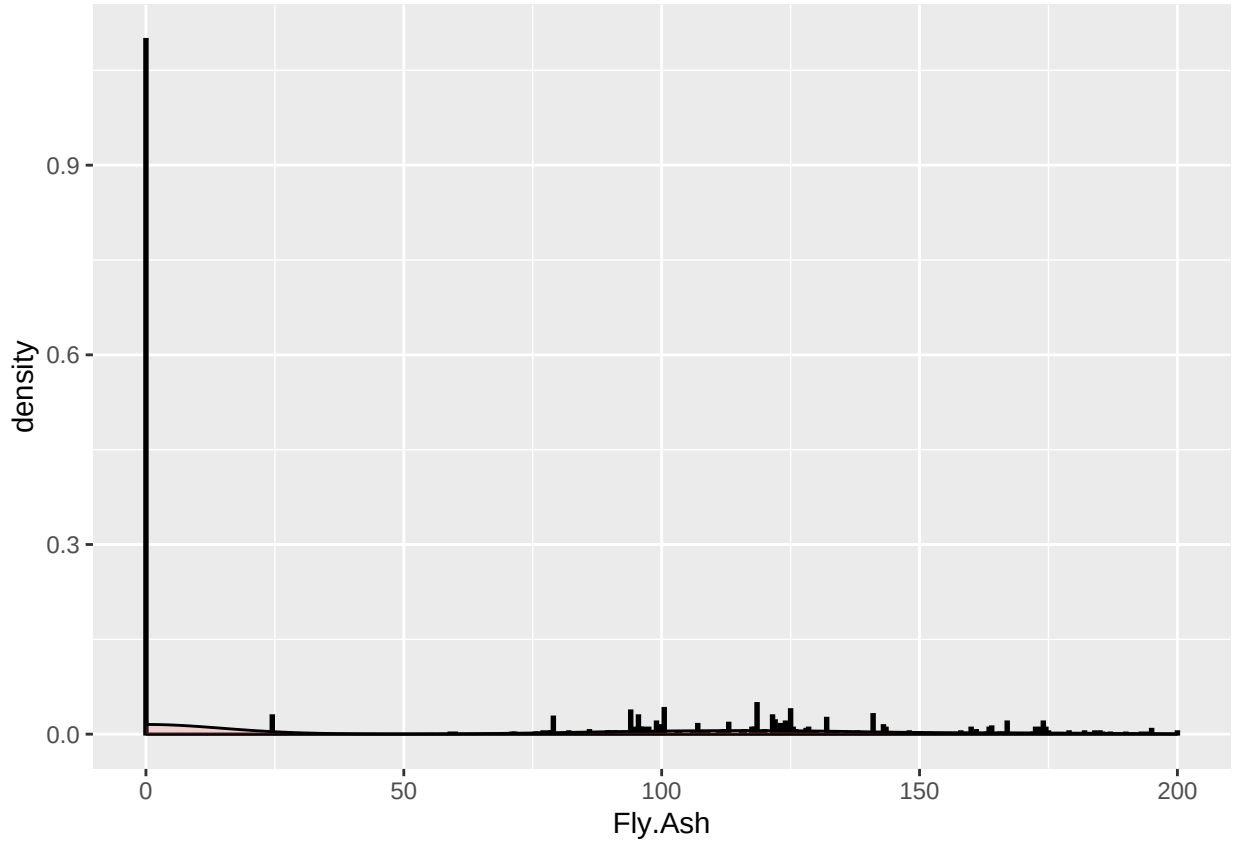

```
ggplot(file, aes(x=Blast.Furnace.Slag)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



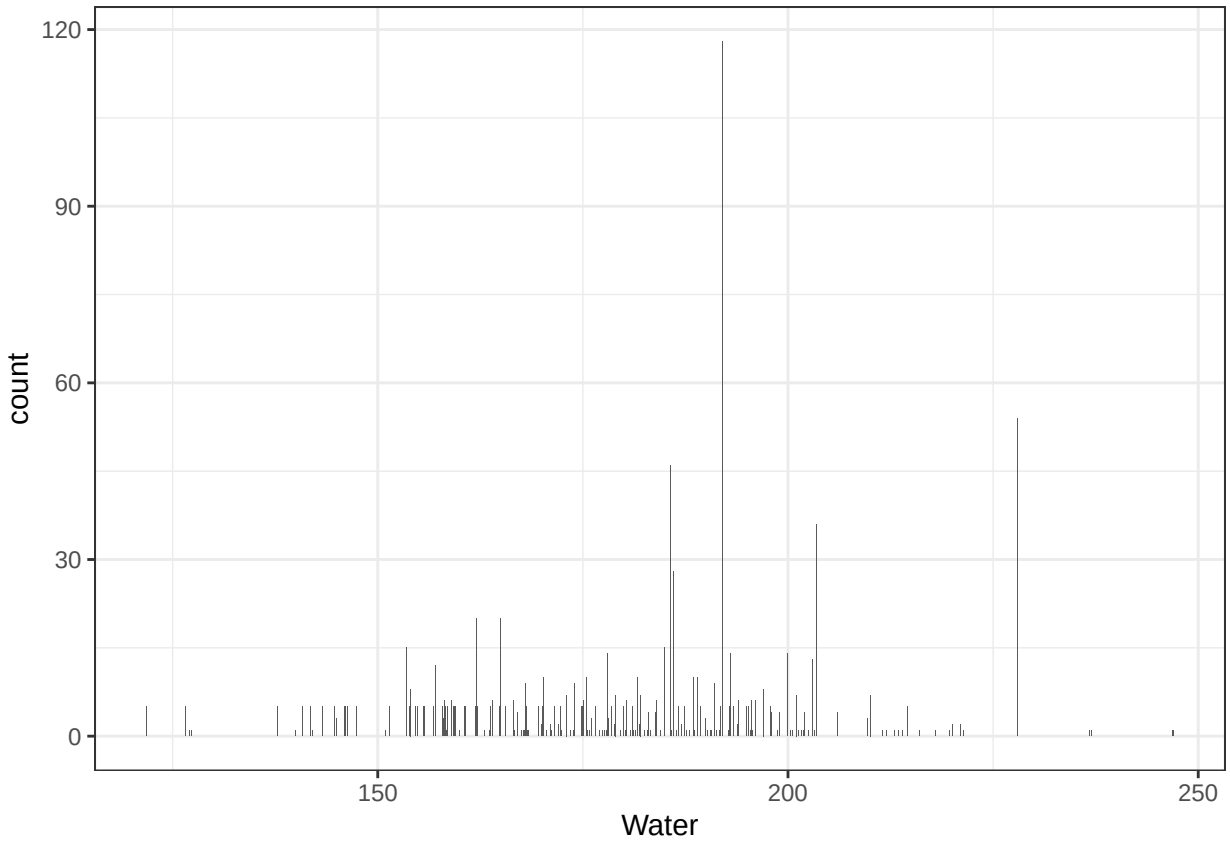

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Fly.Ash, fill=Fly.Ash)) + theme_bw()
```



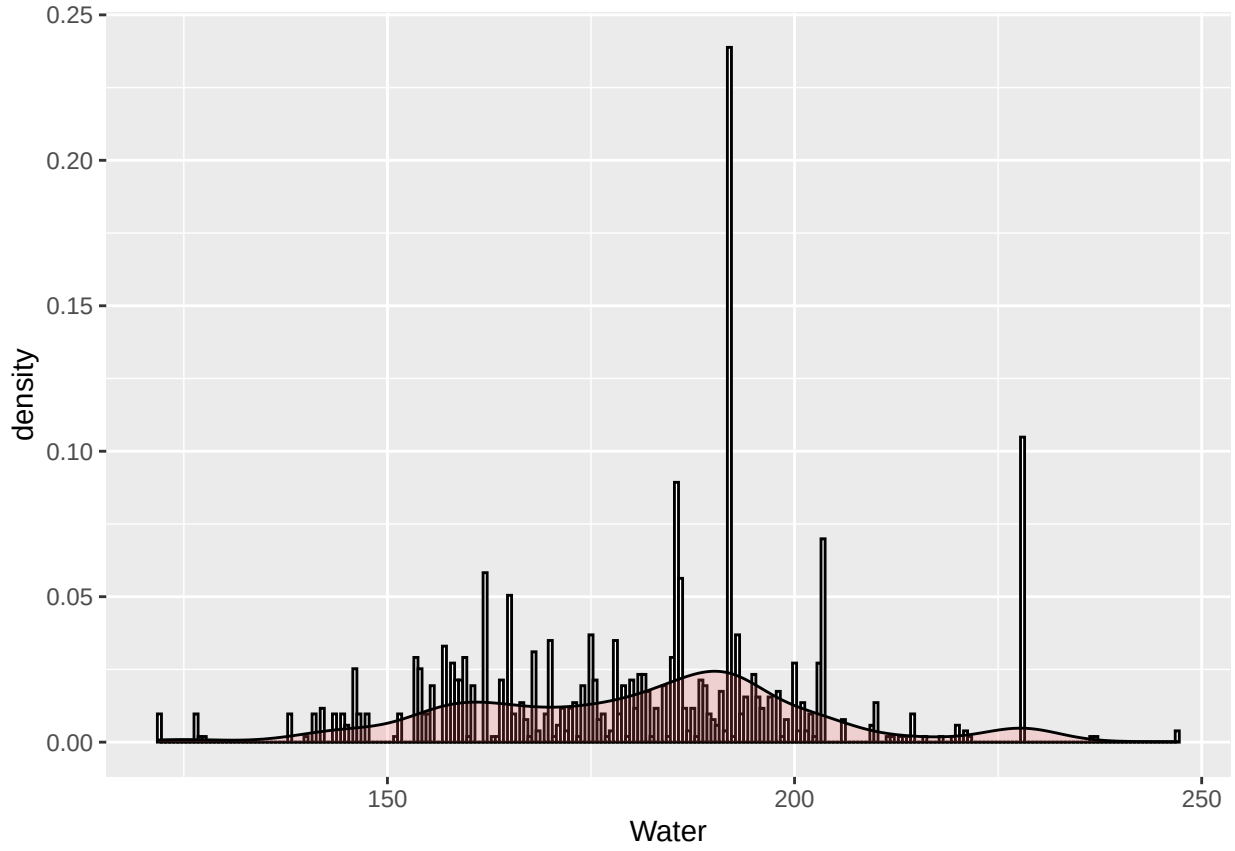

```
ggplot(file, aes(x=Fly.Ash)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



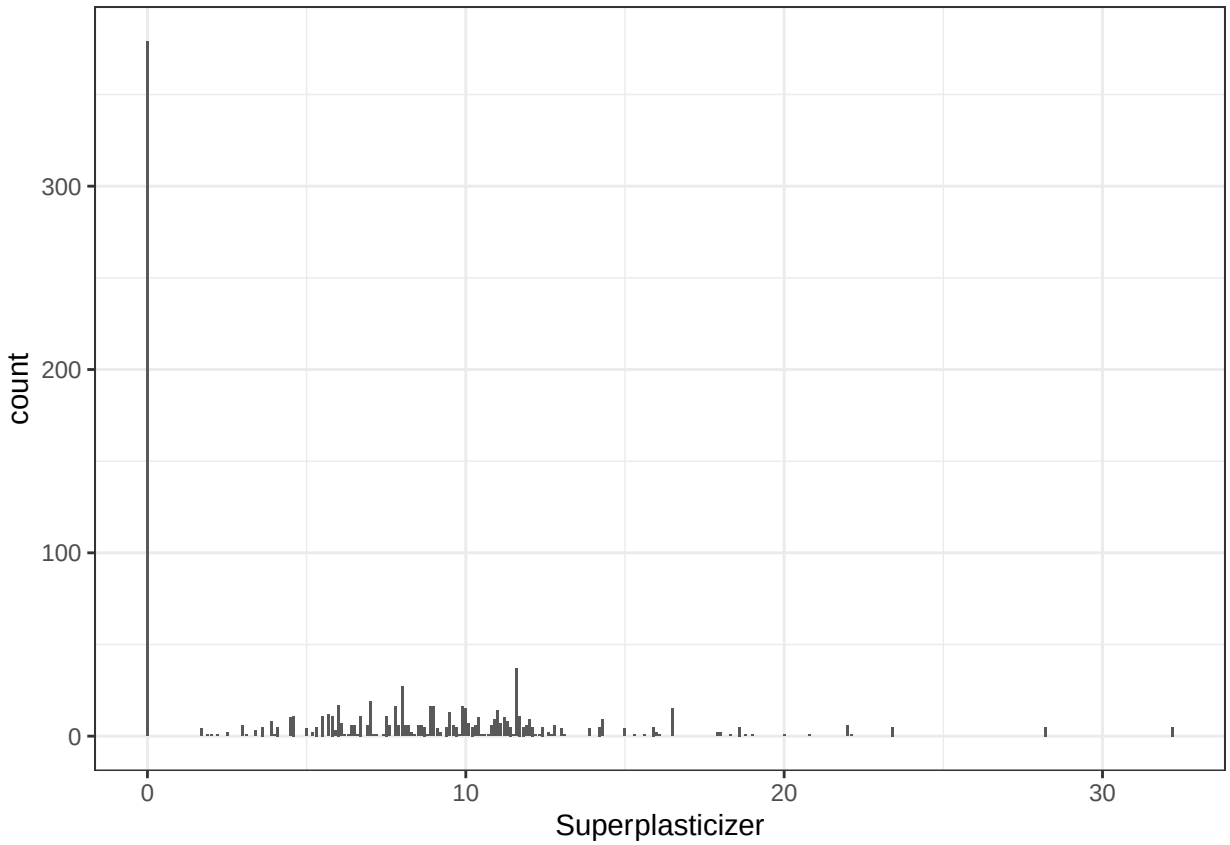

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Water, fill=Water)) + theme_bw()
```



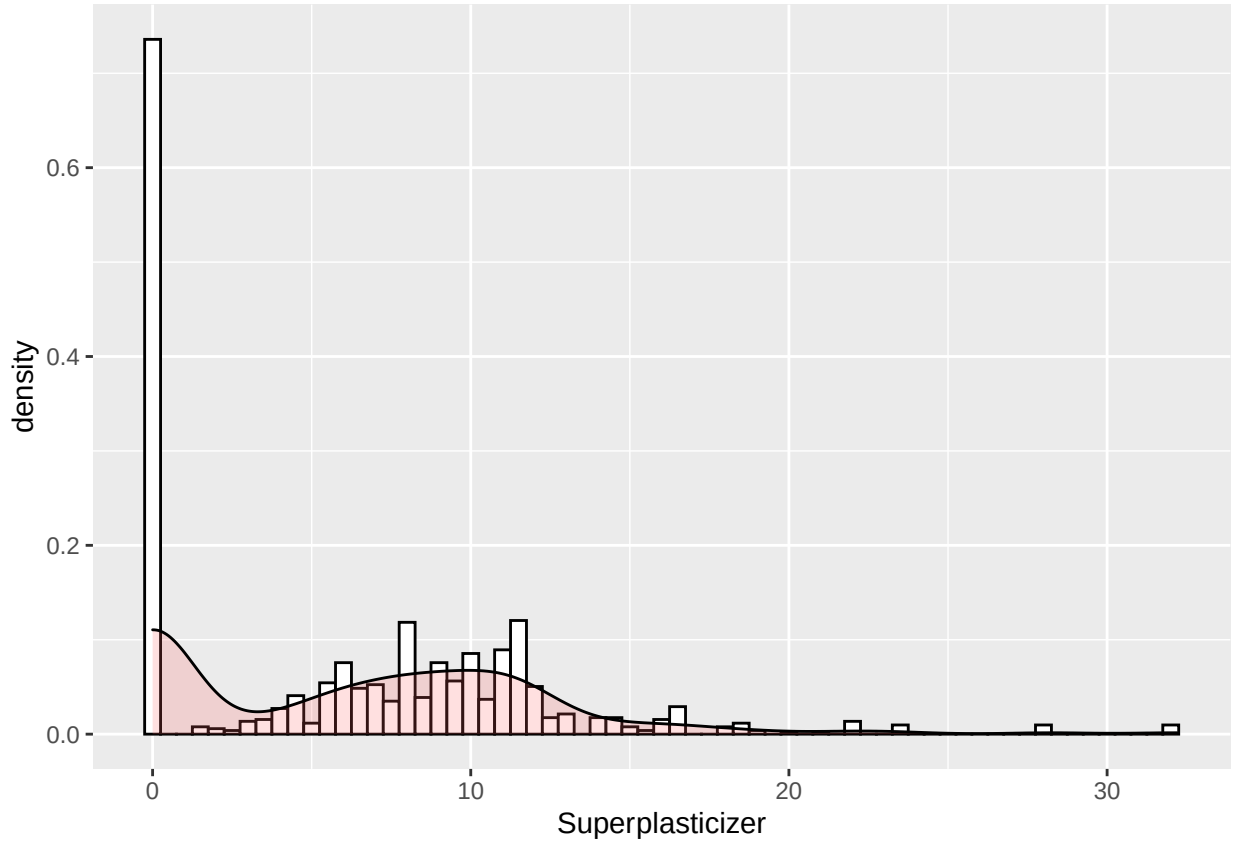

```
ggplot(file, aes(x=Water)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



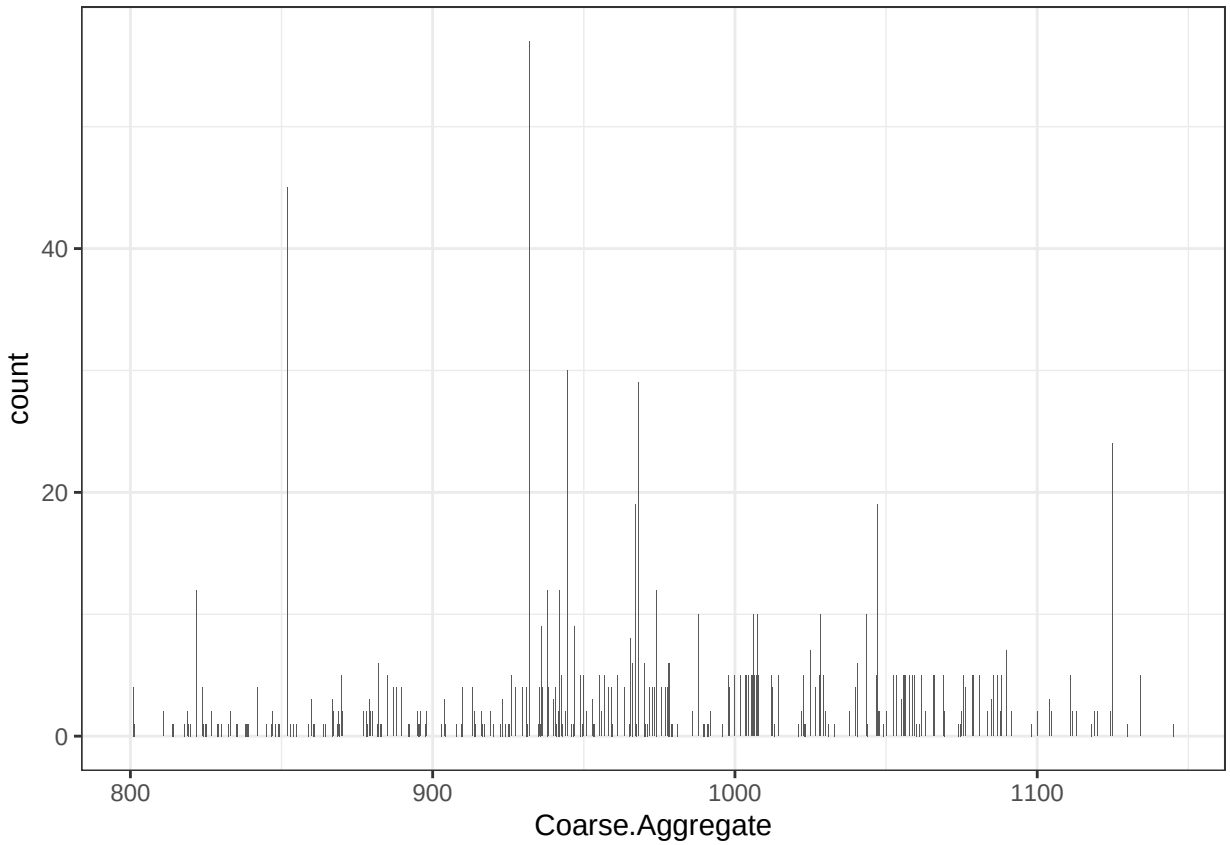

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Superplasticizer, fill=Superplasticizer)) + theme_bw()
```



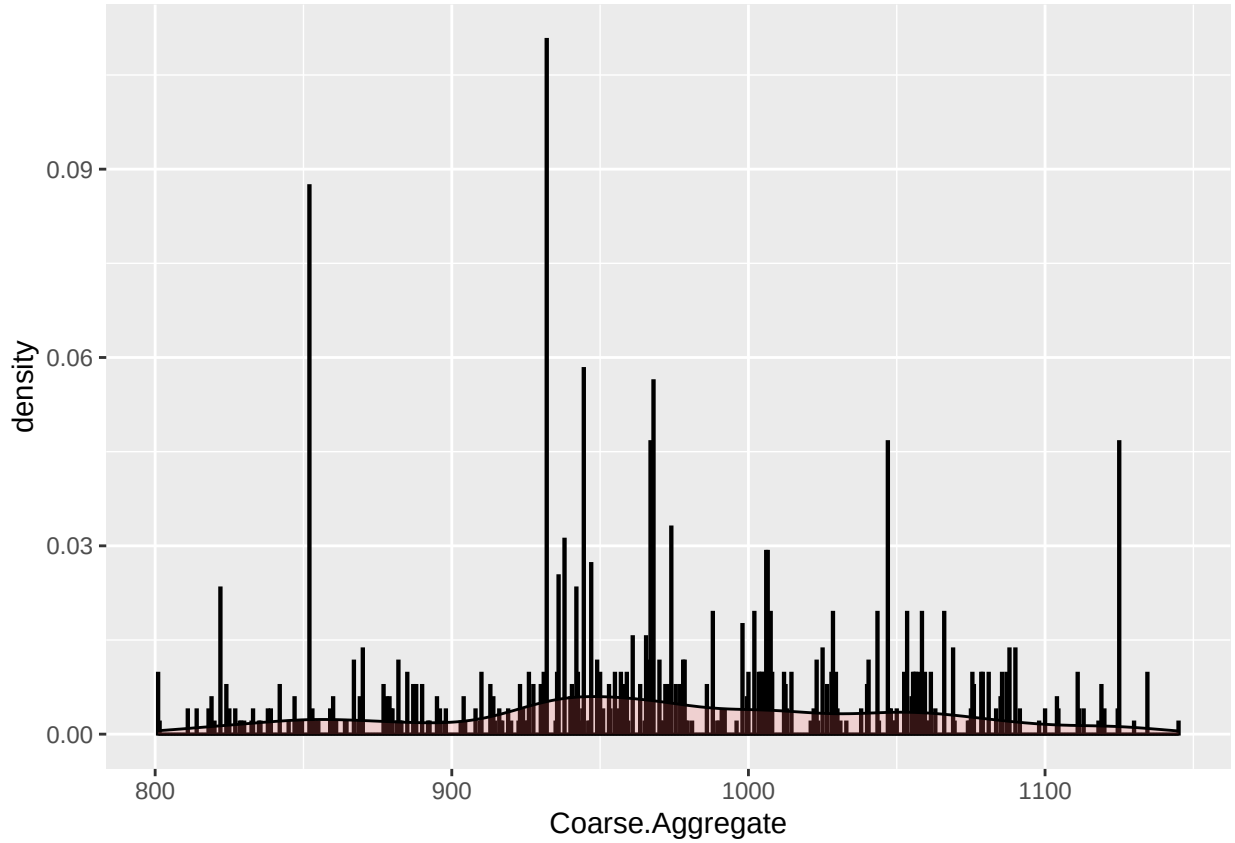

```
ggplot(file, aes(x=Superplasticizer)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



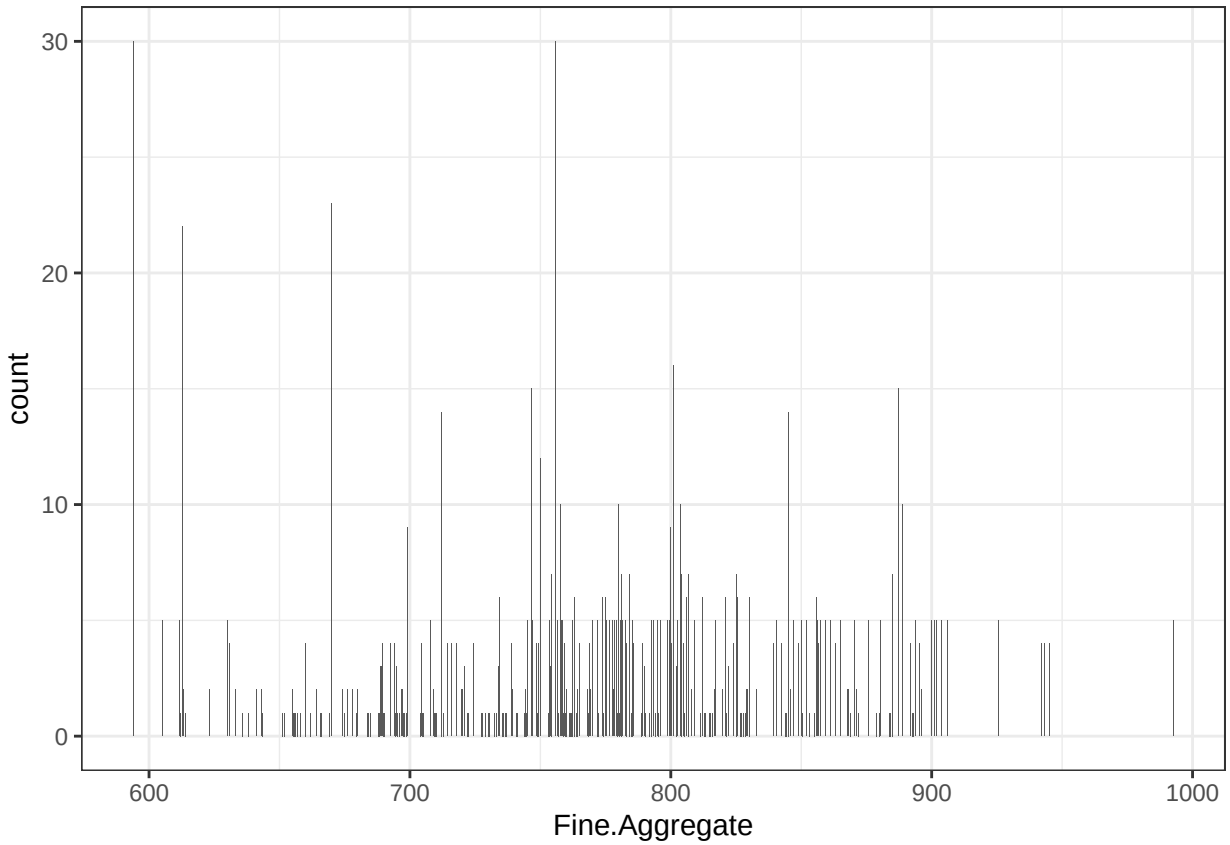

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Coarse.Aggregate, fill=Coarse.Aggregate)) + theme_bw()
```



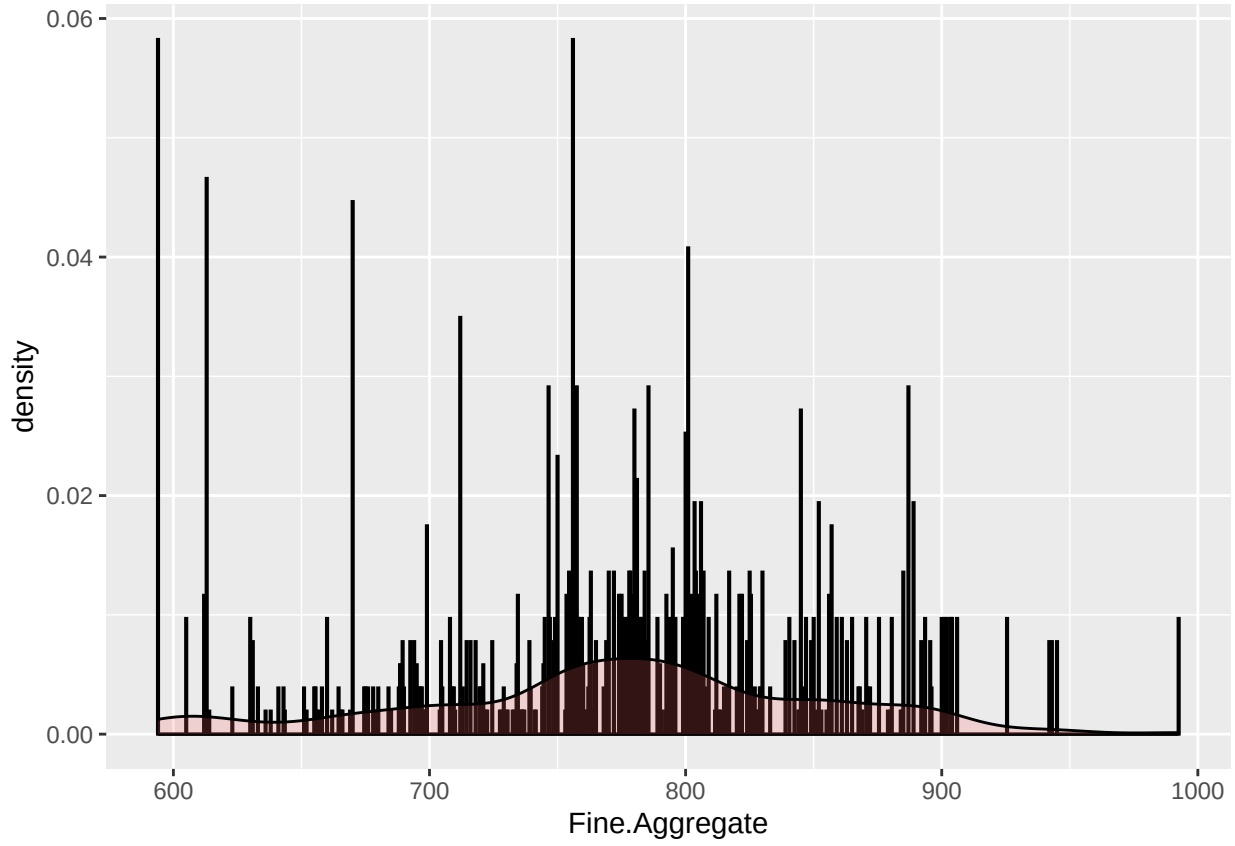

```
ggplot(file, aes(x=Coarse.Aggregate)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



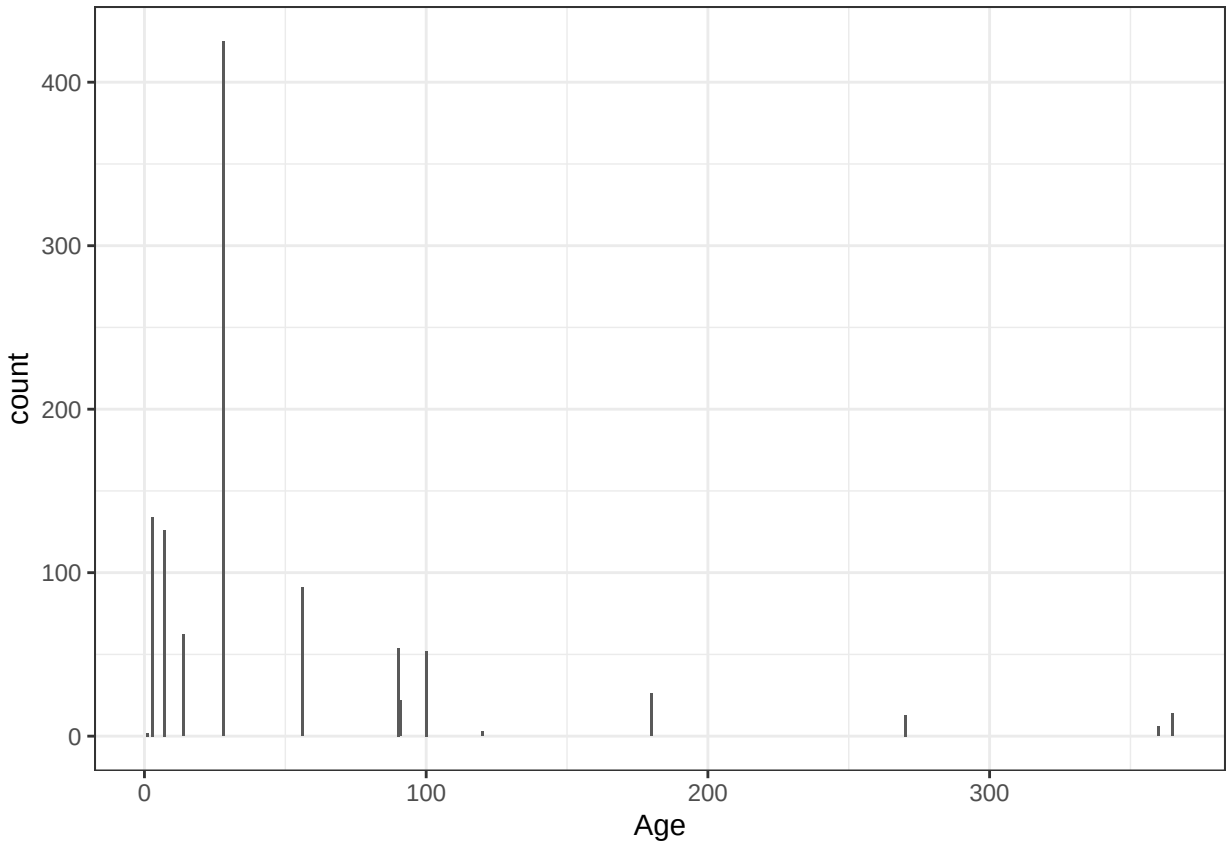

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Fine.Aggregate, fill=Fine.Aggregate)) + theme_bw()
```



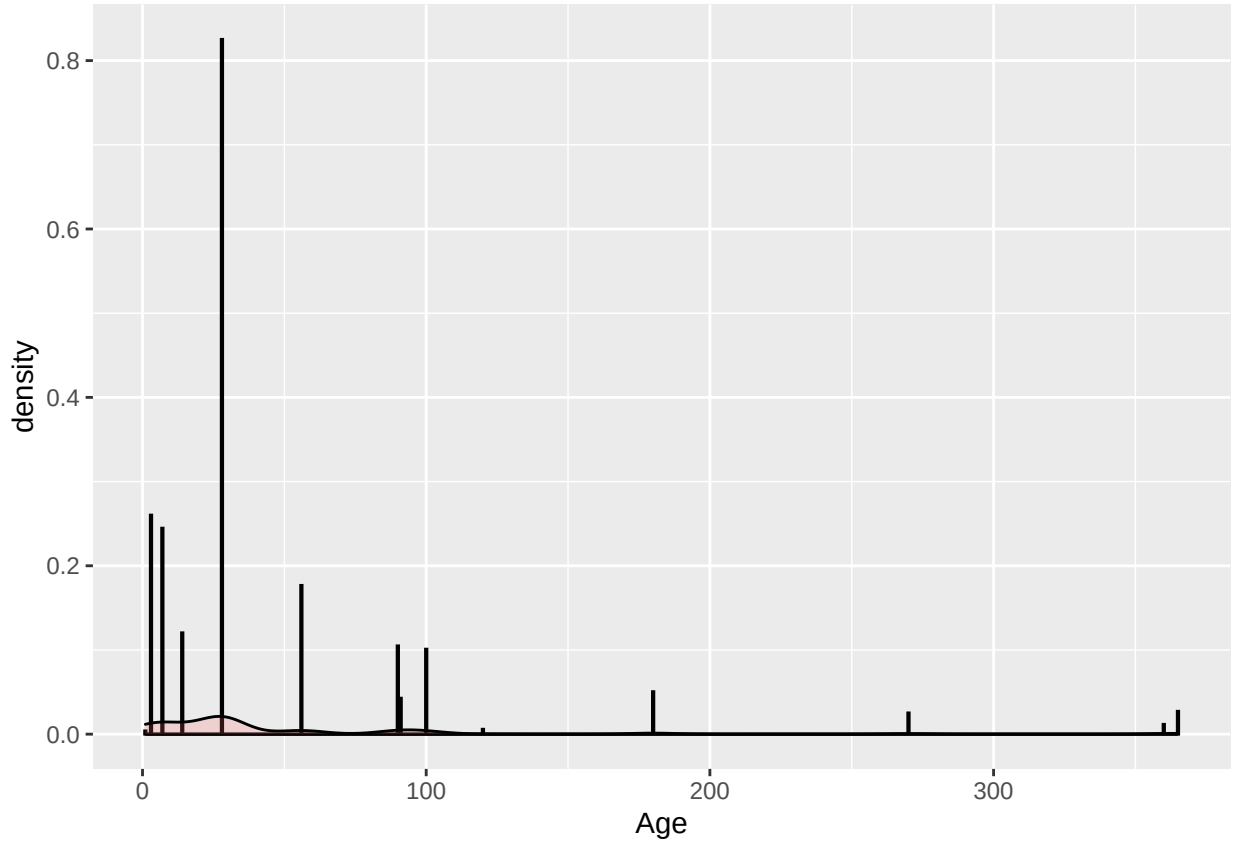

```
ggplot(file, aes(x=Fine.Aggregate)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```



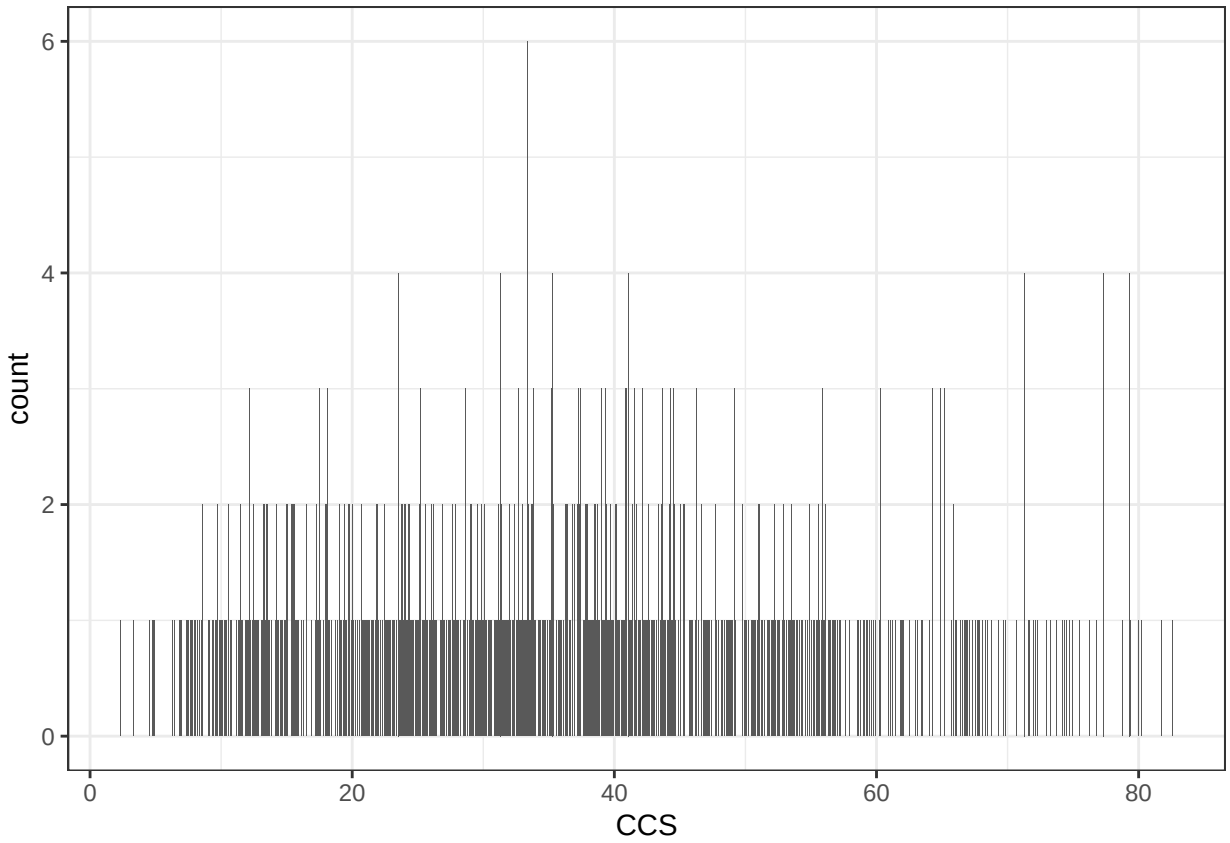

```
ggplot(data = file) +
       geom_bar(mapping = aes(x=Age, fill=Age)) + theme_bw()
```



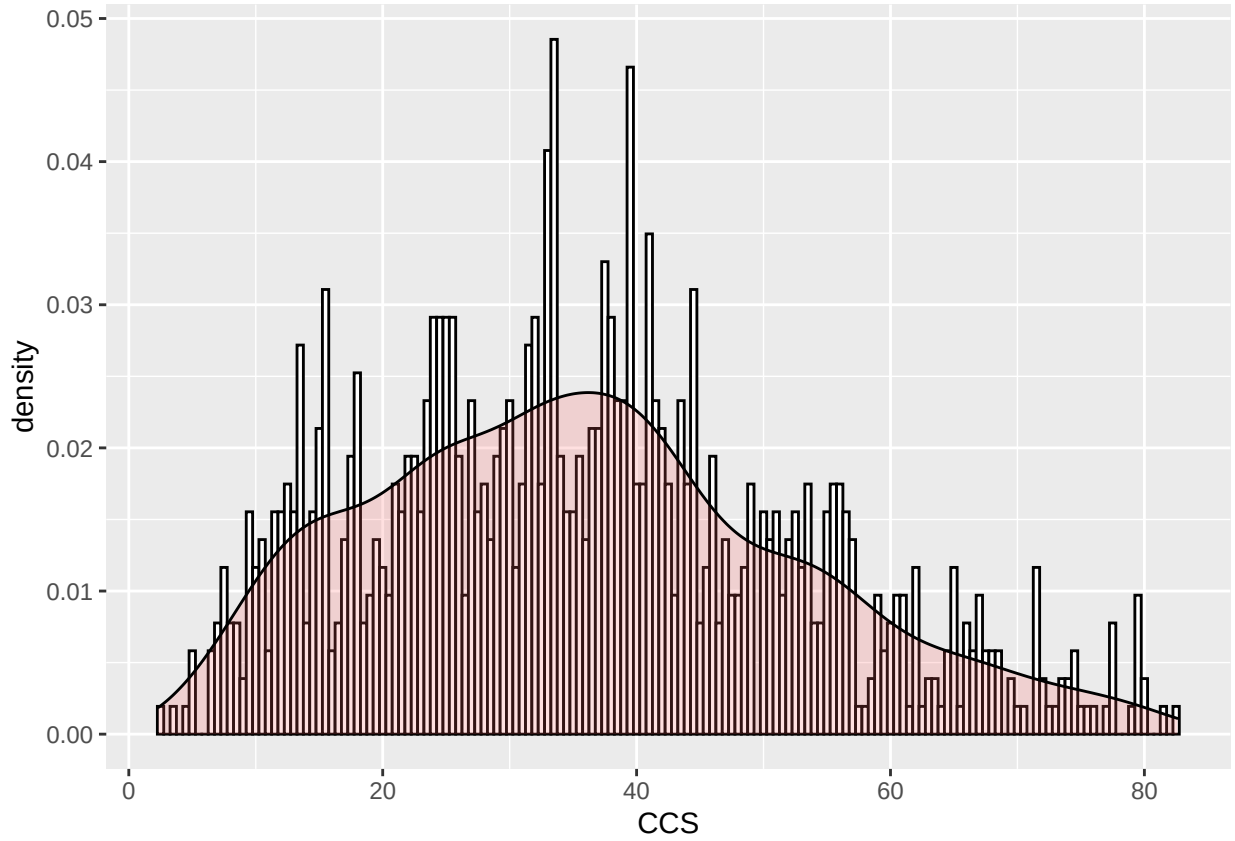

```
ggplot(file, aes(x=Age)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```




```
ggplot(data = file) +
       geom_bar(mapping = aes(x=CCS, fill=CCS)) + theme_bw()
```




```
ggplot(file, aes(x=CCS)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=.5,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")  # Overlay with transparent density plot
```


0.05
0.04
0.03
0.02
0.01
0.00
density
0
20
40
60
80
CCS

MORE DATA VISUALIZATION

```
featurePlot(x = file[c("Cement", "Blast.Furnace.Slag", "Fly.Ash", "Water",
                       "Superplasticizer", "Coarse.Aggregate", "Fine.Aggregate", "Age")],
            y = file$CCS,
            plot = "density",
            ## Pass in options to xyplot() to make it prettier
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(4, 2),
            auto.key = list(columns = 5))

featurePlot(x = file[c(1,2,3,4,5,6,7,8)],
            y = file$CCS,
            plot = "density",
            ## Pass in options to xyplot() to make it prettier
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(4, 2),
            auto.key = list(columns = 5))
```

```
varlist <- c("Cement", "Blast.Furnace.Slag", "Fly.Ash", "Water",
             "Superplasticizer", "Coarse.Aggregate", "Fine.Aggregate", "Age")

customPlot <- function(varName) {

file %>%
group_by_("CCS") %>%
select_("CCS", varName) %>%
ggplot(aes_string("CCS", varName, fill="CCS")) +
       geom_boxplot() +
       # scale_fill_manual(values=c("#999999", "#E69F00")) +
       facet_wrap(~CCS)
}

lapply(varlist, customPlot)
```

2. REGRESSION ANALYSIS by each FEATURE

2.1 one FEATURE

```r
##############################################################################################
reg_model <- lm(CCS~Cement, data = file)

# Linear Regression Model Summary
reg_model
summary(reg_model)
summary(reg_model)$coefficient

# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Cement, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)

# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)

# Diagnostic Plots
autoplot(reg_model)

# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```

2.2 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Blast.Furnace.Slag, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Blast.Furnace.Slag, data = file)
##
## Coefficients:
##        (Intercept)  Blast.Furnace.Slag
##           33.88882             0.02611
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Blast.Furnace.Slag, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.798 -12.116  -1.599  10.241  46.101
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        33.888824   0.679565  49.868  < 2e-16 ***
## Blast.Furnace.Slag  0.026106   0.005984   4.363 1.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.56 on 1028 degrees of freedom
## Multiple R-squared:  0.01818,    Adjusted R-squared:  0.01722
## F-statistic: 19.03 on 1 and 1028 DF,  p-value: 1.414e-05
```

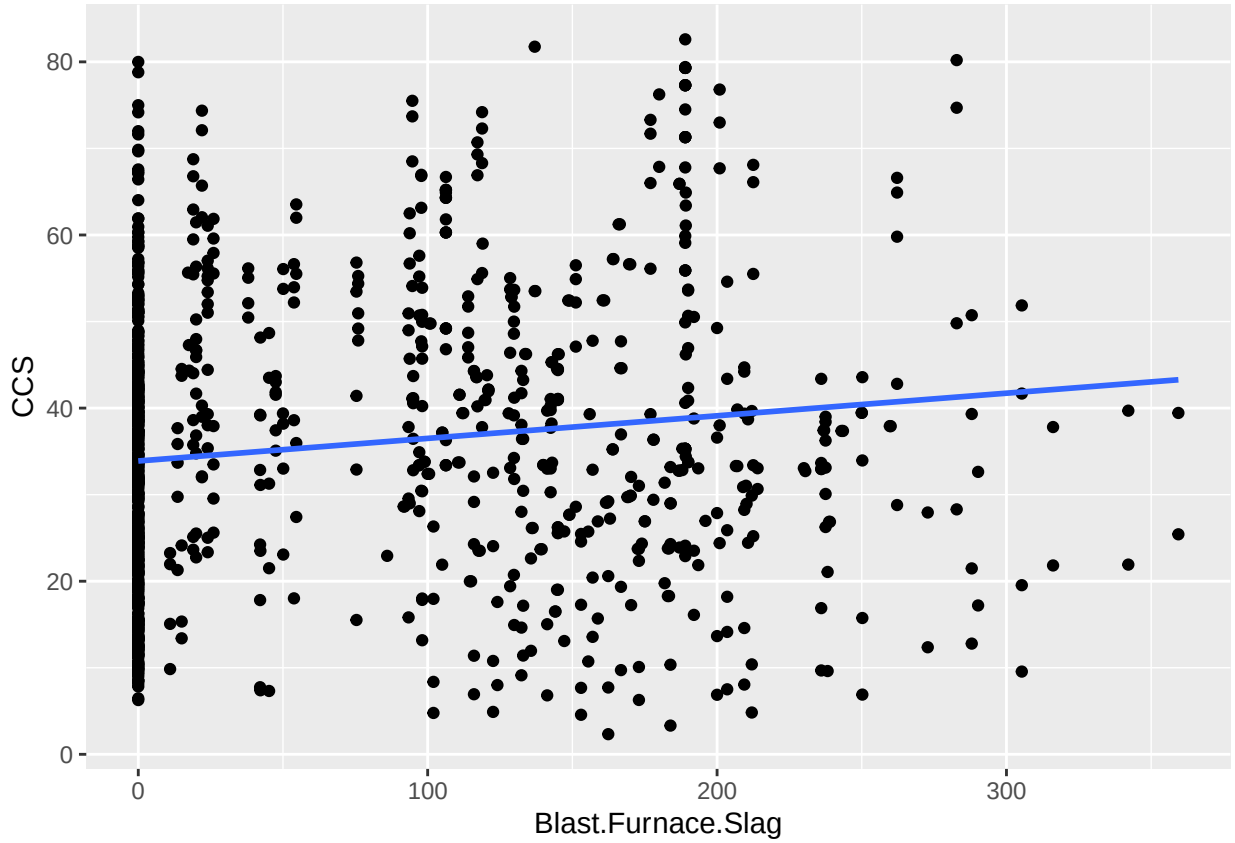
```
summary(reg_model)$coefficient
```

```
##                       Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept)        33.88882446 0.679565262 49.868388 1.093374e-276
## Blast.Furnace.Slag  0.02610617 0.005983821  4.362792  1.413567e-05
```

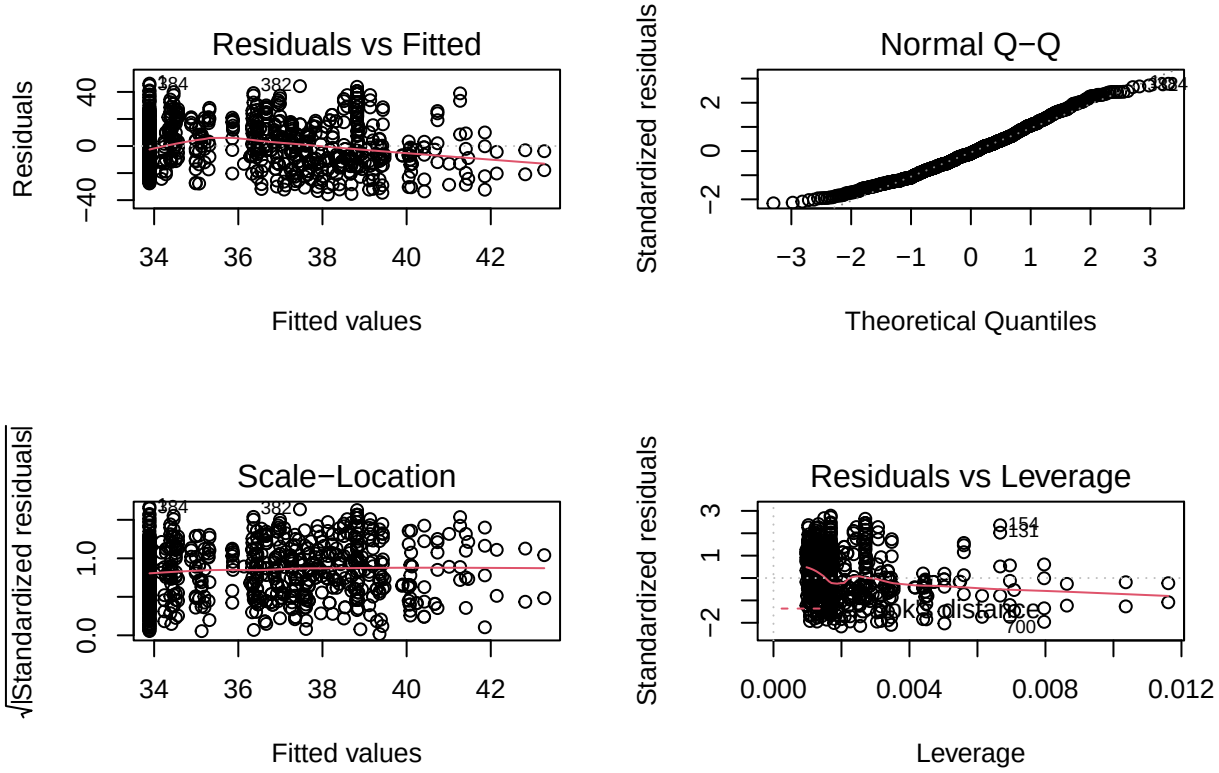
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Blast.Furnace.Slag, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

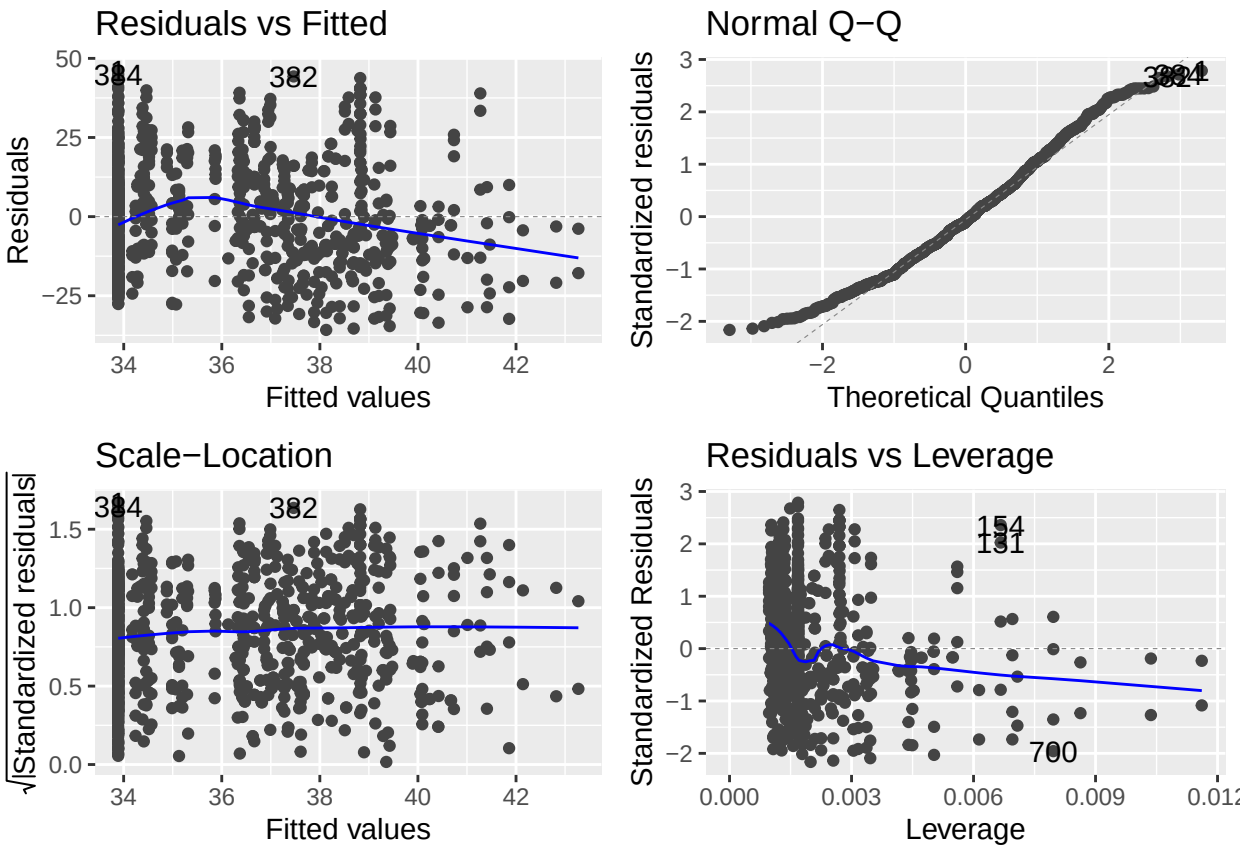
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

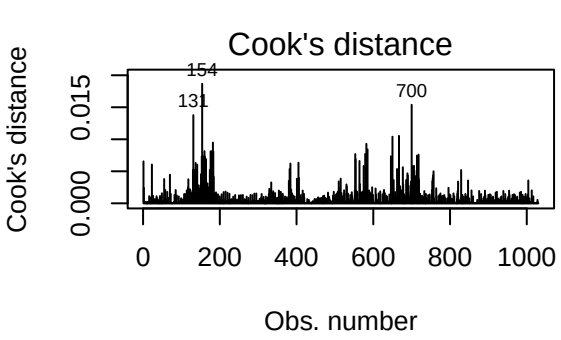


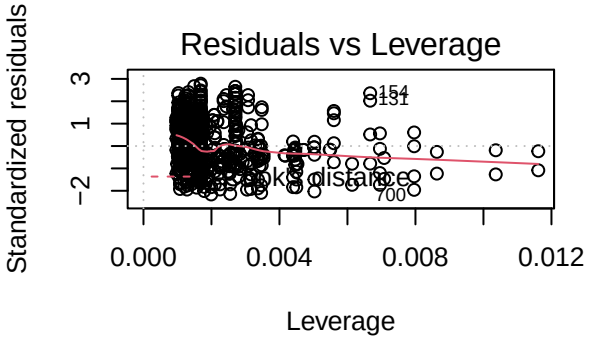

```r
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.000
0.015
154
131
700
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−2
1
3
154
131
Cook's distance
700
0.000
0.004
0.008
0.012
Leverage

2.3 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Fly.Ash, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Fly.Ash, data = file)
##
## Coefficients:
## (Intercept)      Fly.Ash
##    37.31390     -0.02761
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Fly.Ash, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.984 -12.134  -0.568  10.486  45.286
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.313896   0.678709   54.98  < 2e-16 ***
## Fly.Ash     -0.027606   0.008096   -3.41 0.000675 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.62 on 1028 degrees of freedom
## Multiple R-squared:  0.01118,    Adjusted R-squared:  0.01022
## F-statistic: 11.63 on 1 and 1028 DF,  p-value: 0.0006752
```

```
summary(reg_model)$coefficient
```

```
##               Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 37.3138958 0.67870915 54.977741 2.298371e-308
## Fly.Ash     -0.0276062 0.00809594 -3.409882  6.751584e-04
```

```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)
head(reg_model.diagnostics)
```

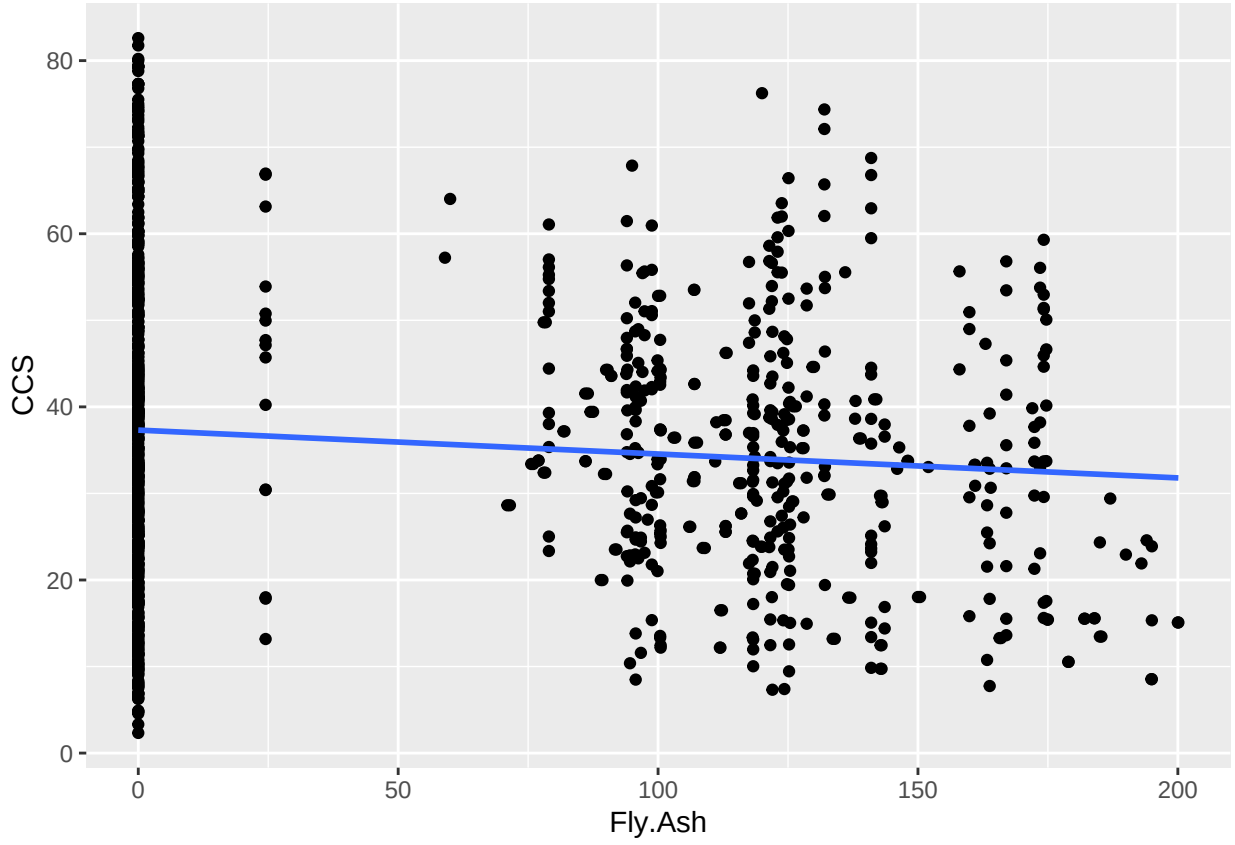
```
## # A tibble: 6 x 8
##     CCS Fly.Ash .fitted .resid    .hat .sigma   .cooksd .std.resid
##   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl>     <dbl>      <dbl>
## 1  80.0       0    37.3  42.7  0.00167   16.6 0.00552        2.57
## 2  61.9       0    37.3  24.6  0.00167   16.6 0.00183        1.48
## 3  40.3       0    37.3   2.96 0.00167   16.6 0.0000265      0.178
## 4  41.0       0    37.3   3.74 0.00167   16.6 0.0000423      0.225
## 5  44.3       0    37.3   6.99 0.00167   16.6 0.000148       0.421
## 6  47.0       0    37.3   9.72 0.00167   16.6 0.000286       0.585
```

```
tail(reg_model.diagnostics)
```

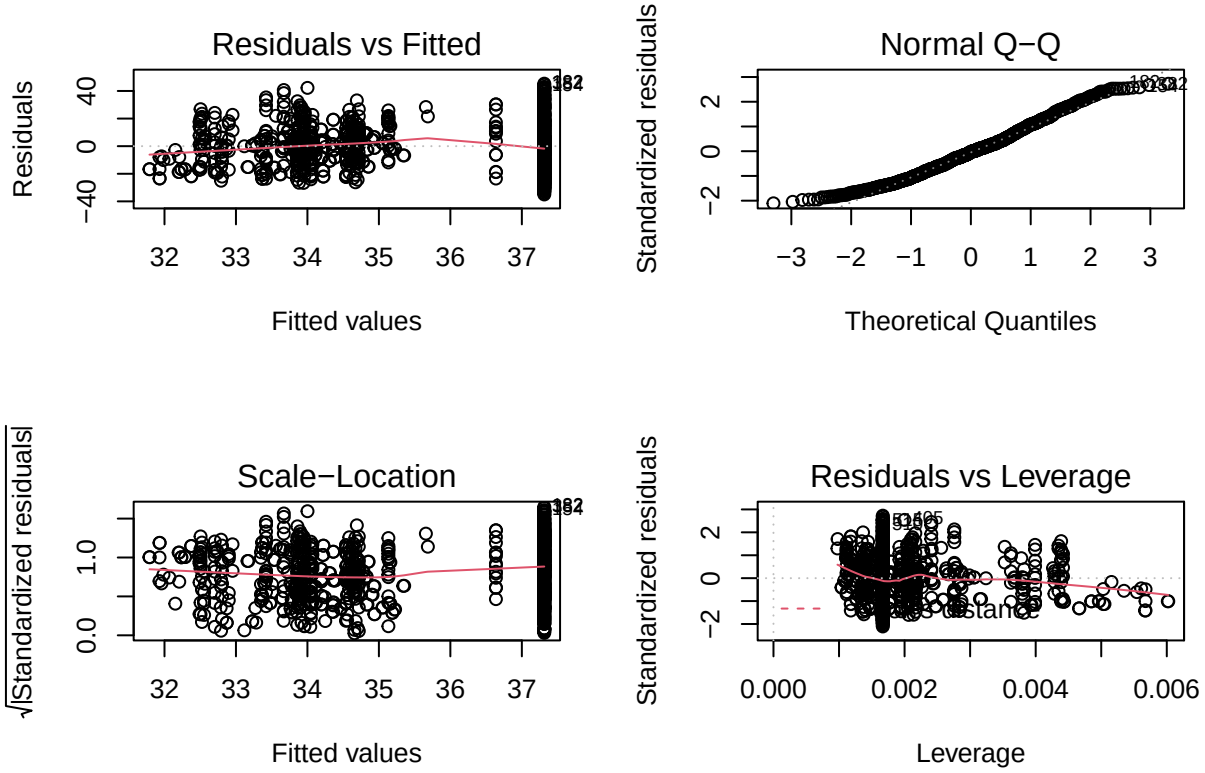
```
## # A tibble: 6 x 8
##     CCS Fly.Ash .fitted  .resid    .hat .sigma    .cooksd .std.resid
##   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>      <dbl>      <dbl>
## 1  37.9     0      37.3   0.606 0.00167   16.6 0.00000111     0.0365
## 2  44.3    90.3    34.8   9.46  0.00128   16.6 0.000208       0.569
## 3  31.2   116.     34.1  -2.94  0.00187   16.6 0.0000294     -0.177
## 4  23.7   109.     34.3 -10.6   0.00167   16.6 0.000342      -0.639
## 5  32.8     0      37.3  -4.54  0.00167   16.6 0.0000625     -0.274
## 6  32.4    78.3    35.2  -2.75  0.00111   16.6 0.0000152     -0.166
```

```
# Displaying:
ggplot(reg_model.diagnostics, aes(Fly.Ash, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

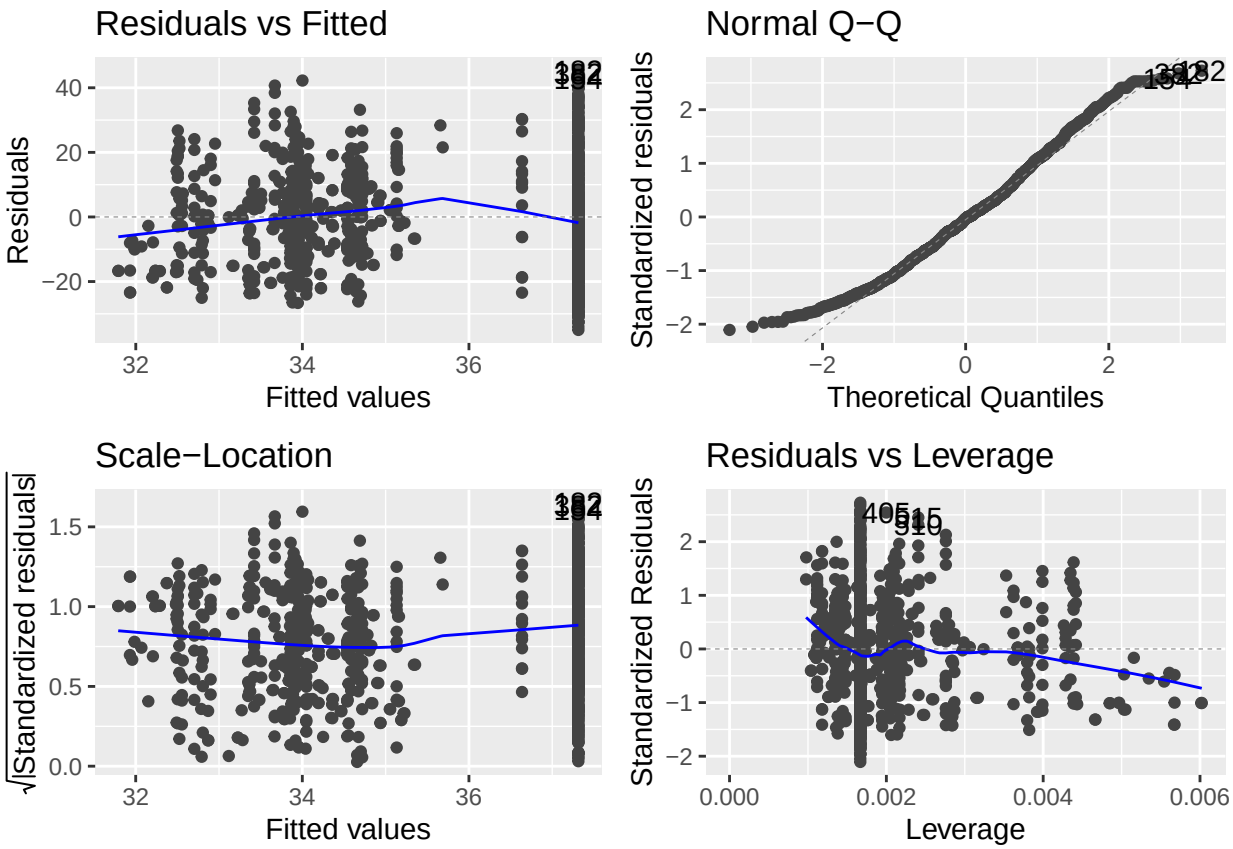
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

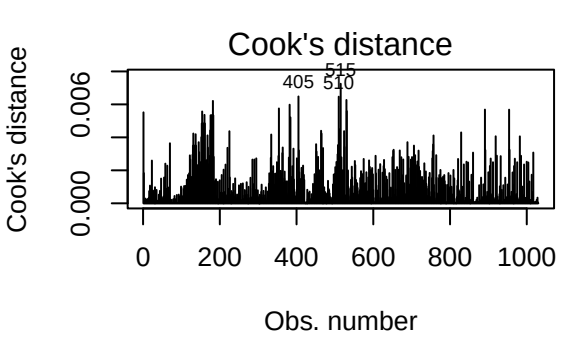


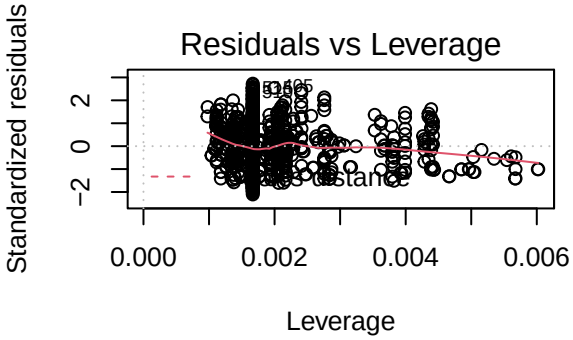

```
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
405
515
510
Cook's distance
0.000
0.006
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−2
0
2
Cook's distance
0.000
0.002
0.004
0.006
Leverage

2.4 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Water, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Water, data = file)
##
## Coefficients:
## (Intercept)        Water
##     76.9583      -0.2266
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Water, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.610 -11.804  -0.843  10.533  44.888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 76.95834    4.26951  18.025   <2e-16 ***
## Water       -0.22658    0.02335  -9.702   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16 on 1028 degrees of freedom
## Multiple R-squared:  0.08389,    Adjusted R-squared:  0.083
## F-statistic: 94.13 on 1 and 1028 DF,  p-value: < 2.2e-16
```

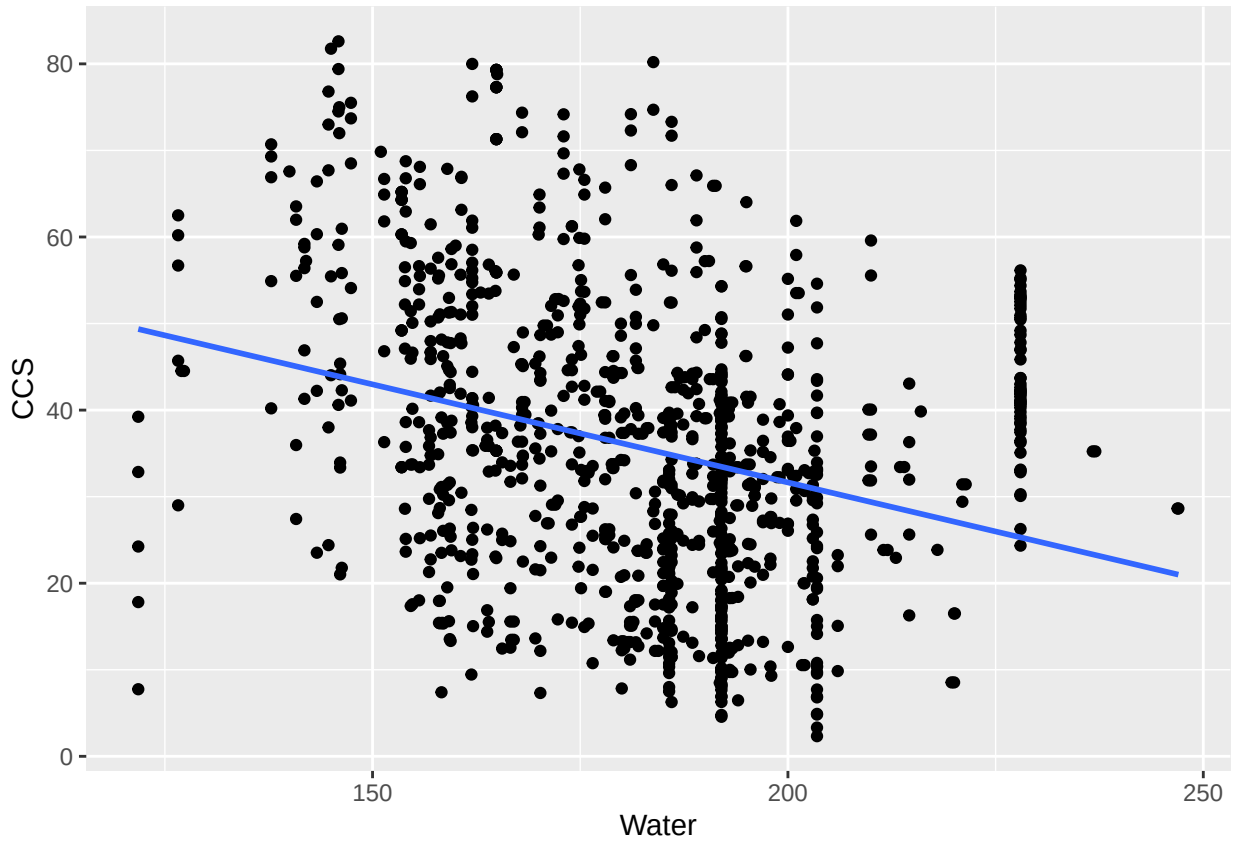
```
summary(reg_model)$coefficient
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 76.9583410 4.26950665 18.025113 2.496810e-63
## Water       -0.2265848 0.02335393 -9.702212 2.350655e-21
```

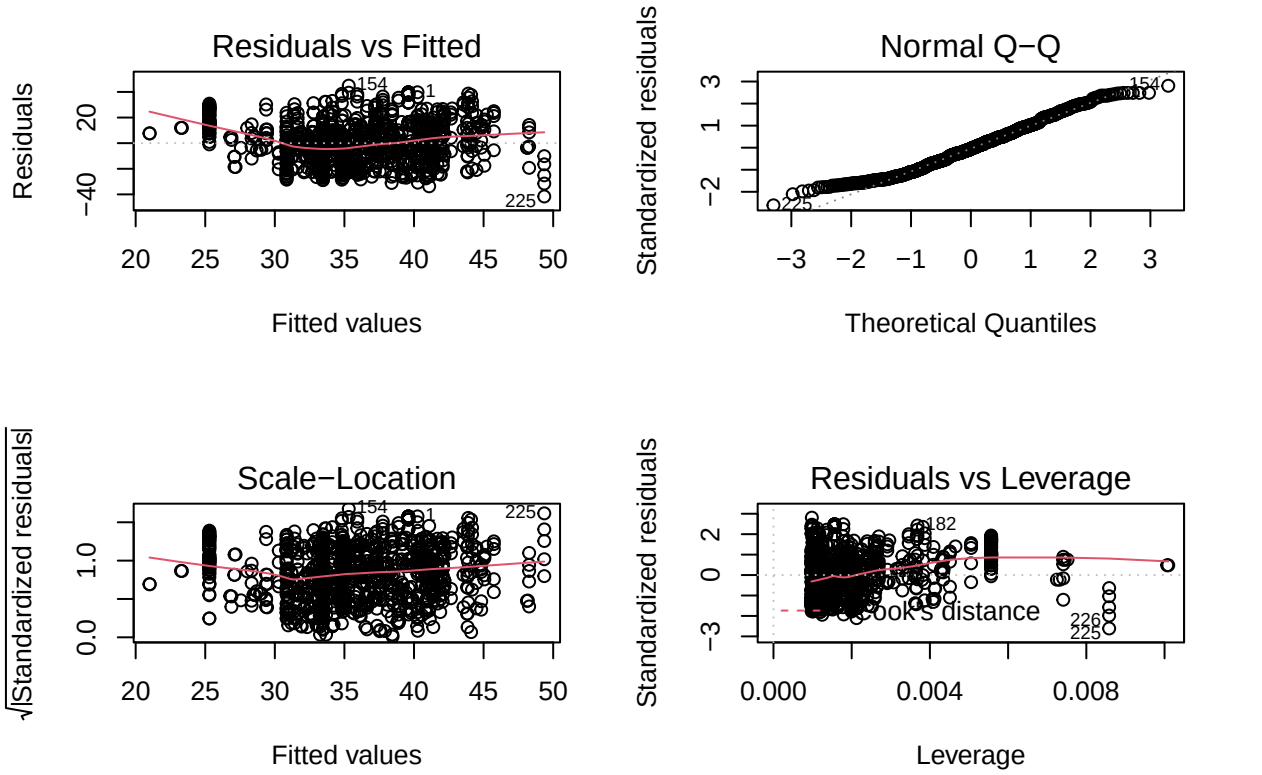
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Water, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

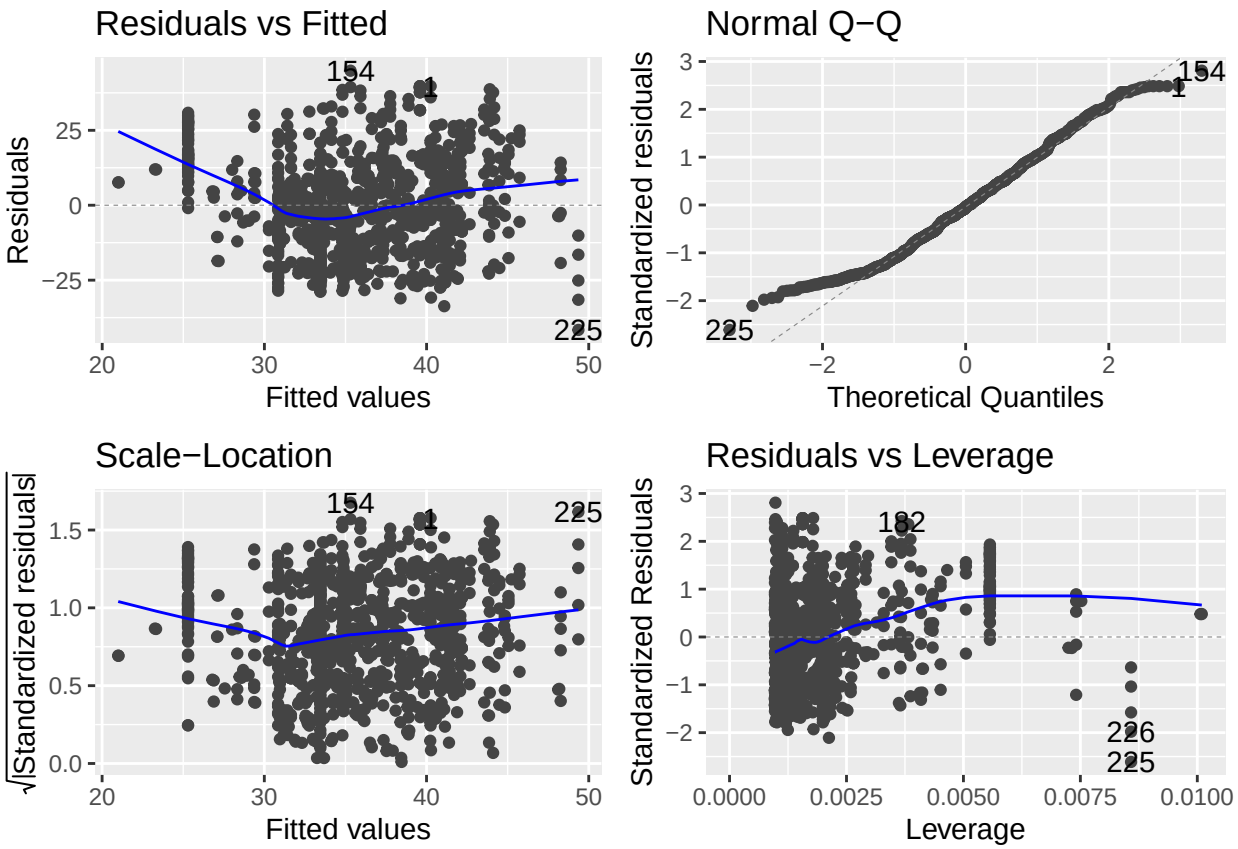
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

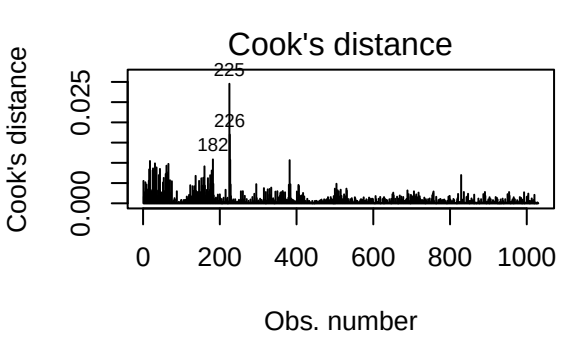


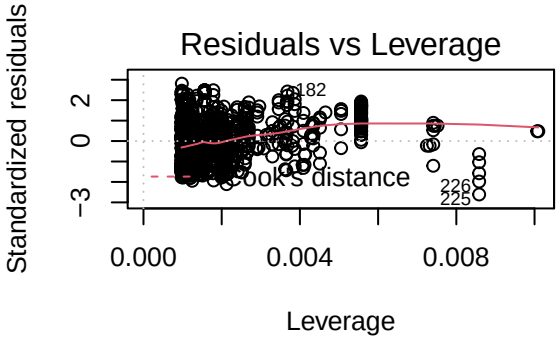

```r
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.000
0.025
225
226
182
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−3
0
2
182
Cook's distance
226
225
0.000
0.004
0.008
Leverage

2.5 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Superplasticizer, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Superplasticizer, data = file)
##
## Coefficients:
##      (Intercept)  Superplasticizer
##           29.466             1.024
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Superplasticizer, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.122 -11.577  -1.071  10.056  47.965
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      29.46605    0.69892   42.16   <2e-16 ***
## Superplasticizer  1.02373    0.08117   12.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.55 on 1028 degrees of freedom
## Multiple R-squared:  0.134,  Adjusted R-squared:  0.1332
## F-statistic: 159.1 on 1 and 1028 DF,  p-value: < 2.2e-16
```

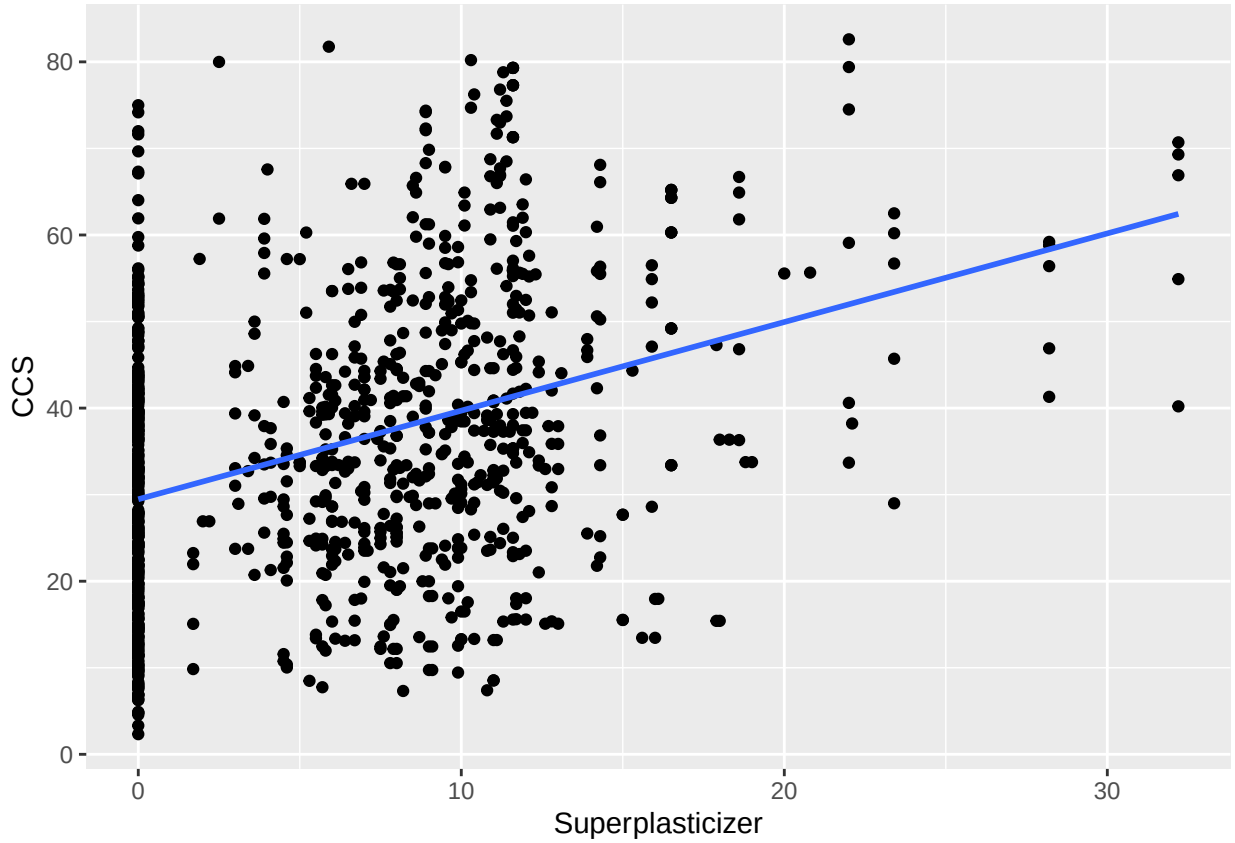
```
summary(reg_model)$coefficient
```

```
##                   Estimate Std. Error  t value      Pr(>|t|)
## (Intercept)      29.466046 0.69891834 42.15950 2.443375e-226
## Superplasticizer  1.023733 0.08116542 12.61292  5.131485e-34
```

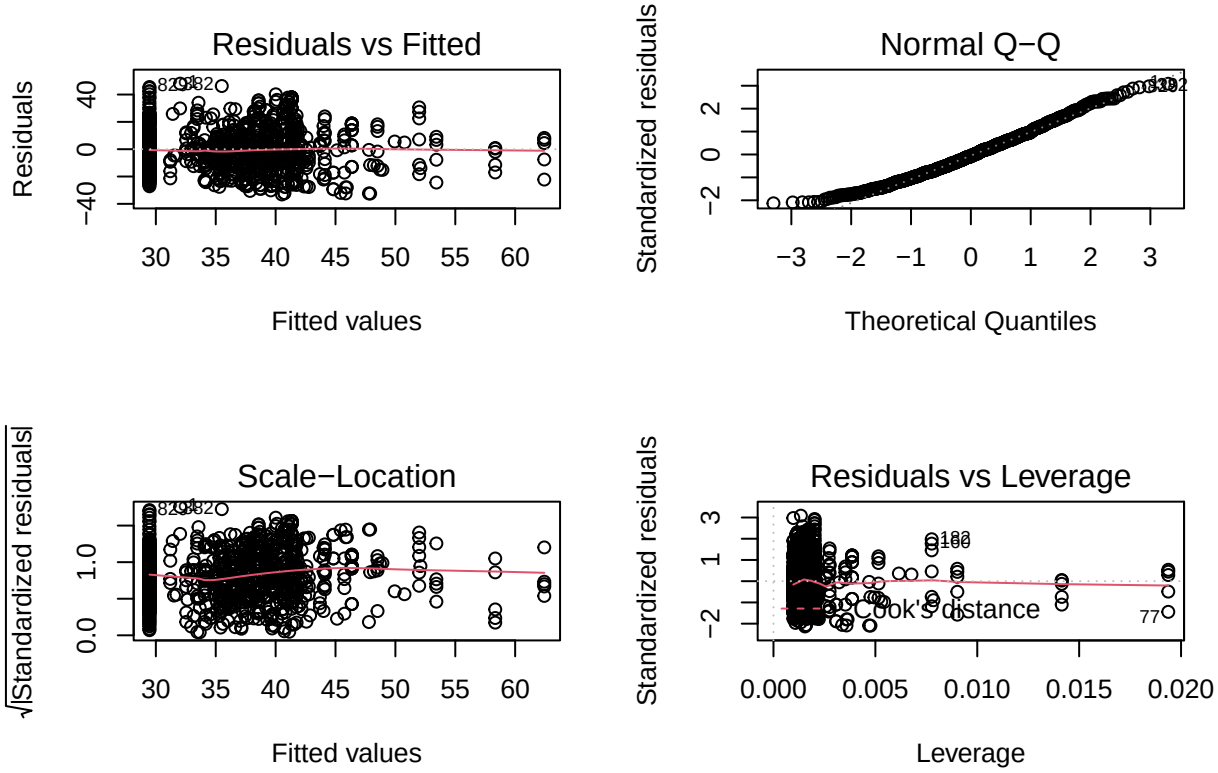
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Superplasticizer, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

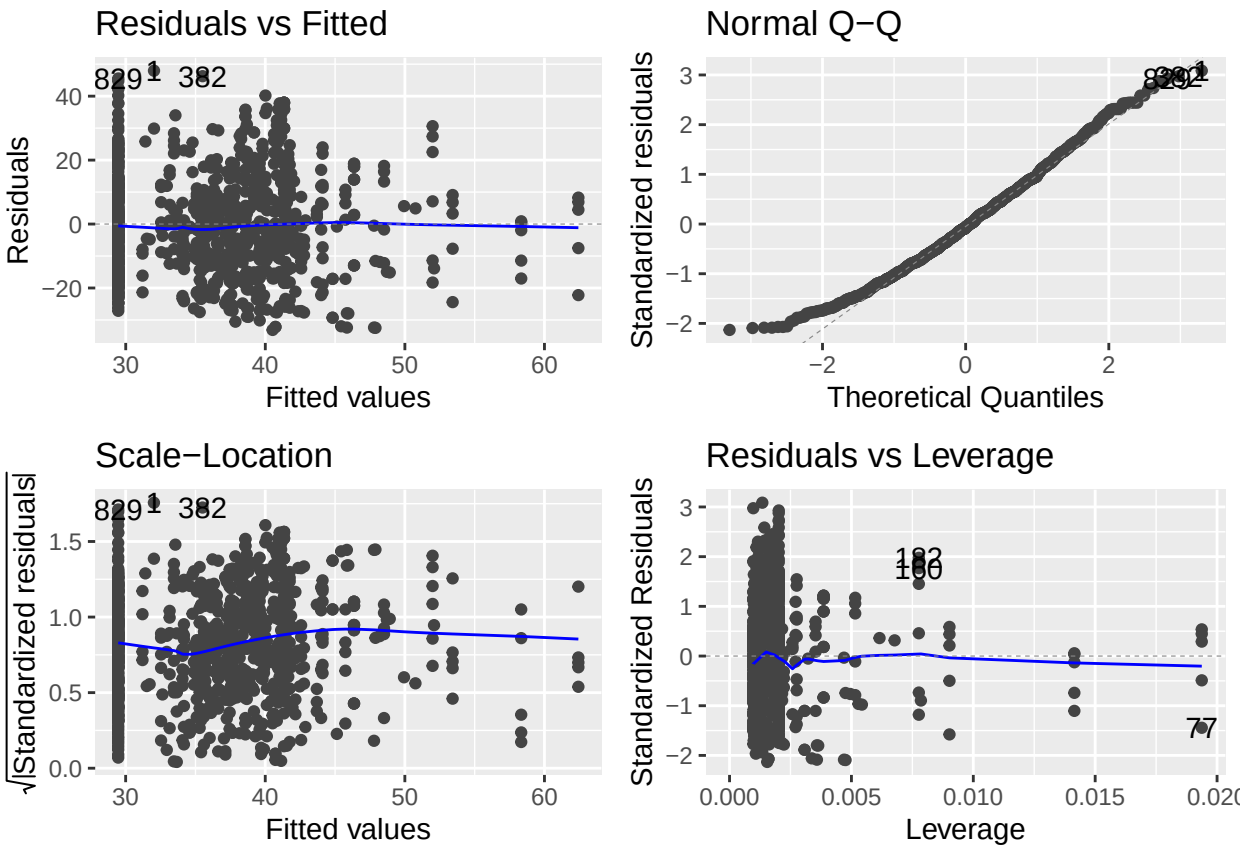
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

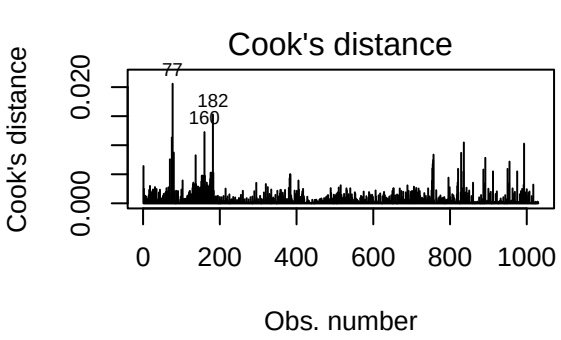


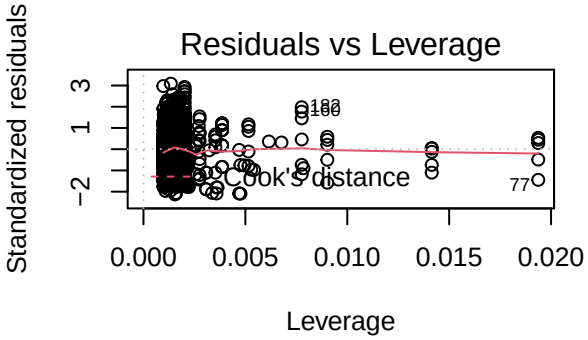

```
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.000
0.020
77
182
160
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−2
1
3
Cook's distance
77
0.000
0.005
0.010
0.015
0.020
Leverage

2.6 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Coarse.Aggregate, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Coarse.Aggregate, data = file)
##
## Coefficients:
##      (Intercept)  Coarse.Aggregate
##         70.29514          -0.03544
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Coarse.Aggregate, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.718 -12.226  -1.395  10.516  51.498
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      70.295142   6.450829  10.897  < 2e-16 ***
## Coarse.Aggregate -0.035437   0.006609  -5.362 1.02e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.48 on 1028 degrees of freedom
## Multiple R-squared:  0.0272, Adjusted R-squared:  0.02626
## F-statistic: 28.75 on 1 and 1028 DF,  p-value: 1.018e-07
```

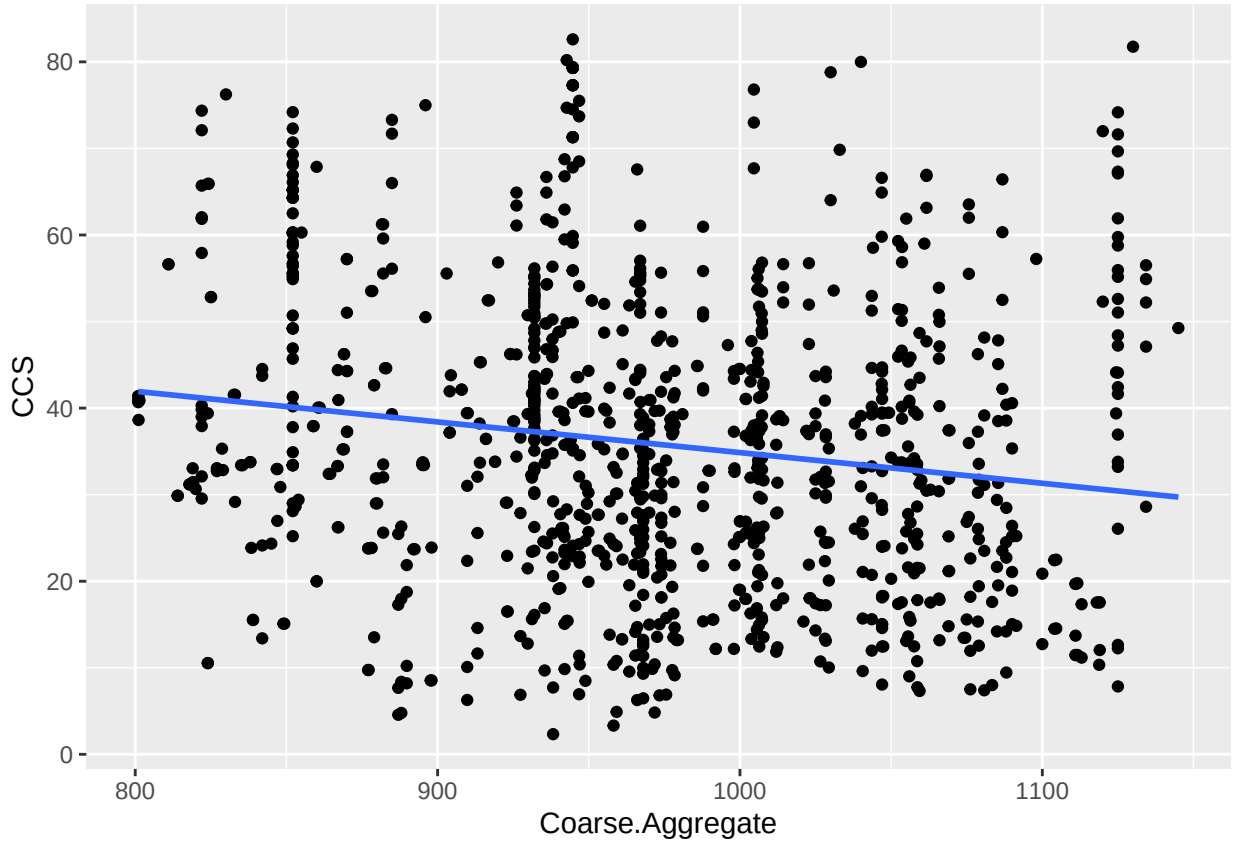
```
summary(reg_model)$coefficient
```

```
##                     Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)      70.29514225 6.450829351 10.897071 3.040601e-26
## Coarse.Aggregate -0.03543685 0.006609335 -5.361636 1.018351e-07
```

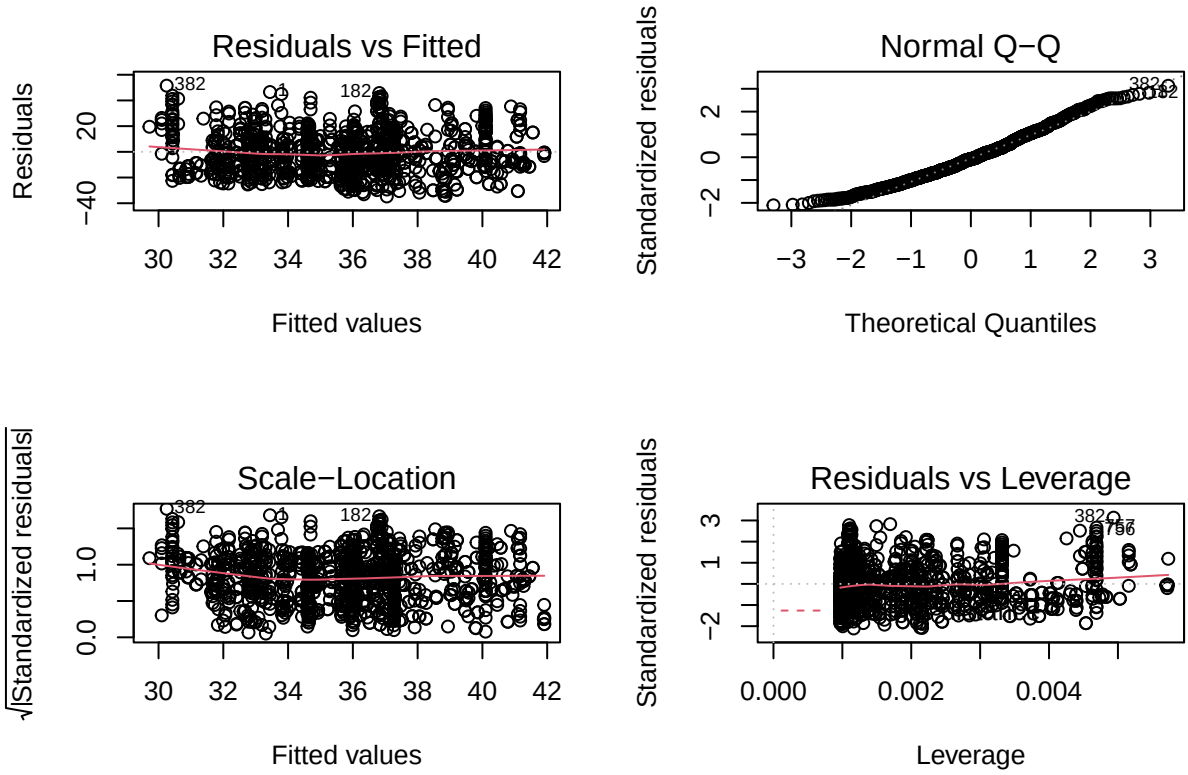
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Coarse.Aggregate, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

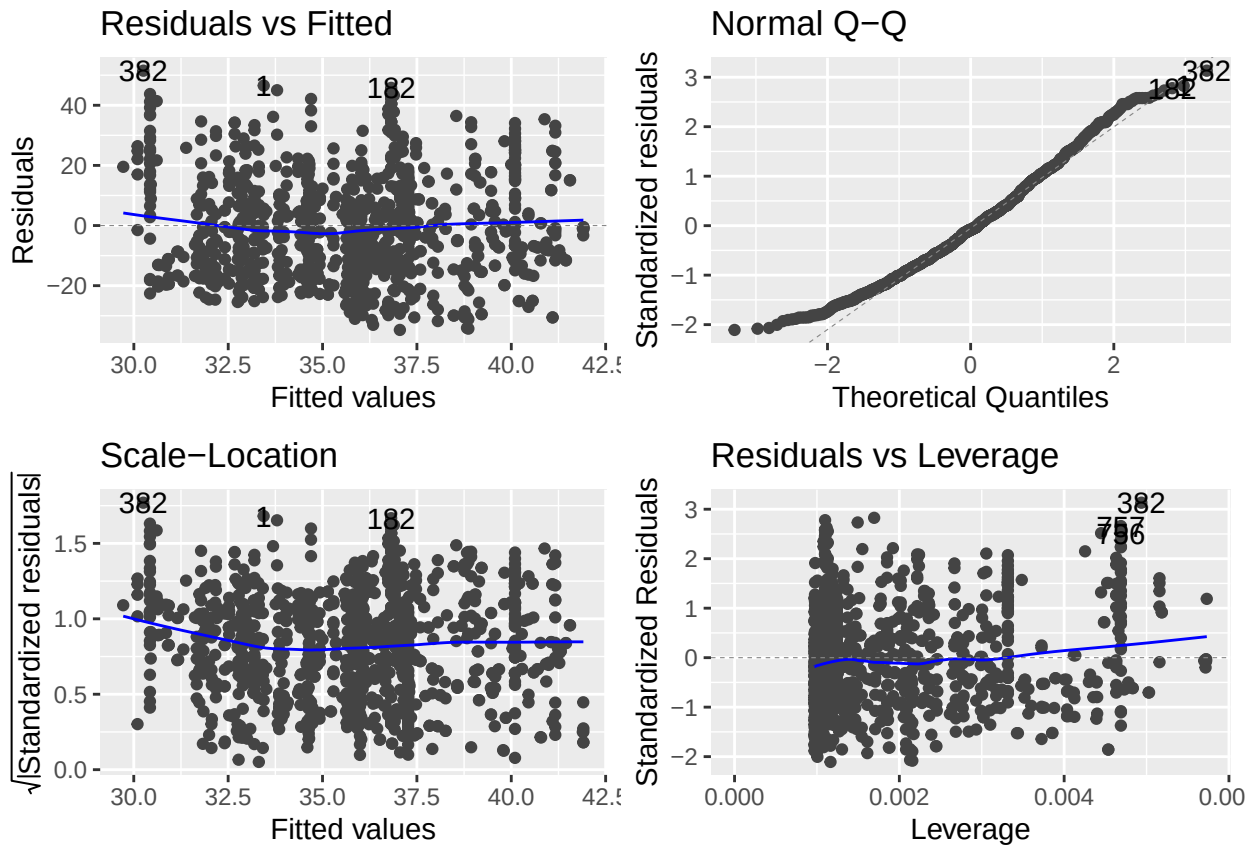
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

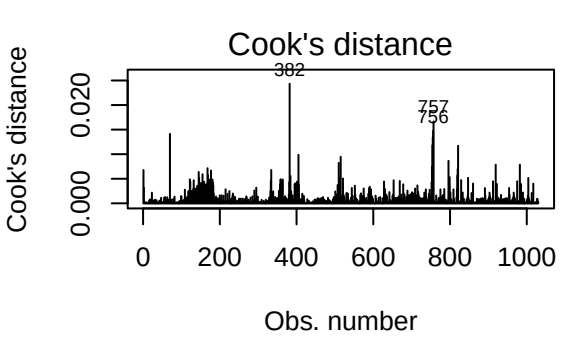


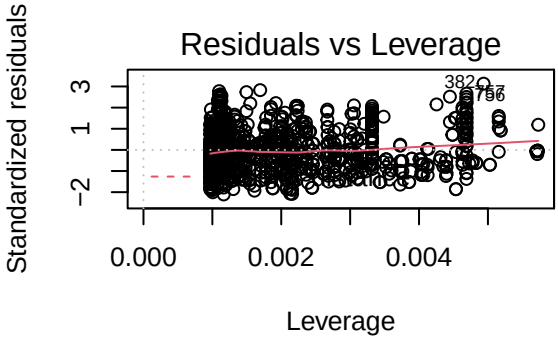

```
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
382
757
756
Cook's distance
0.000
0.020
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
382
757
756
Standardized residuals
−2
1
3
0.000
0.002
0.004
Leverage

2.7 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Fine.Aggregate, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Fine.Aggregate, data = file)
##
## Coefficients:
##    (Intercept)  Fine.Aggregate
##       62.77489        -0.03485
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Fine.Aggregate, data = file)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31.86 -12.08  -1.64  10.37  46.16
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    62.774889   4.982970  12.598  < 2e-16 ***
## Fine.Aggregate -0.034847   0.006407  -5.439  6.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.48 on 1028 degrees of freedom
## Multiple R-squared:  0.02797,    Adjusted R-squared:  0.02702
## F-statistic: 29.58 on 1 and 1028 DF,  p-value: 6.704e-08
```

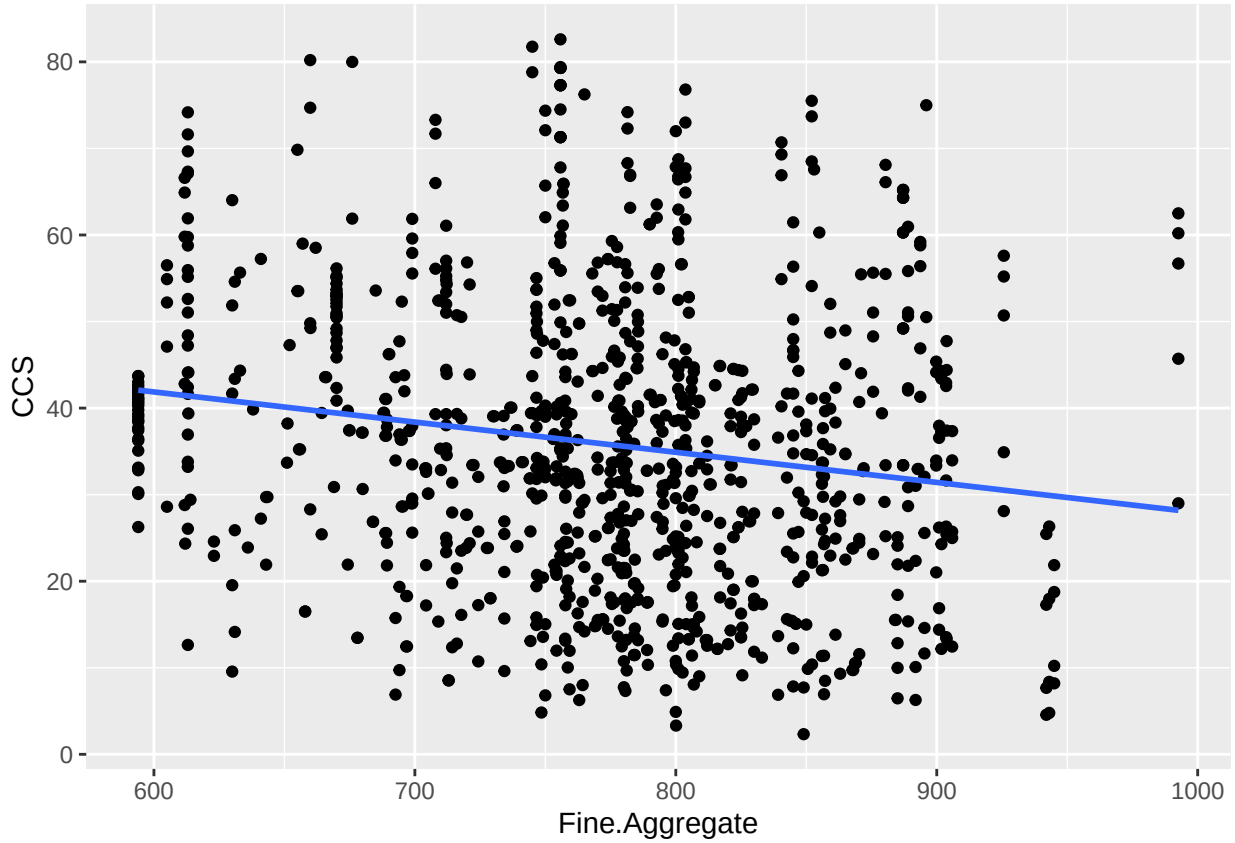
```
summary(reg_model)$coefficient
```

```
##                   Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)    62.77488891 4.982969521 12.597887 6.052935e-34
## Fine.Aggregate -0.03484696 0.006407149 -5.438762 6.704114e-08
```

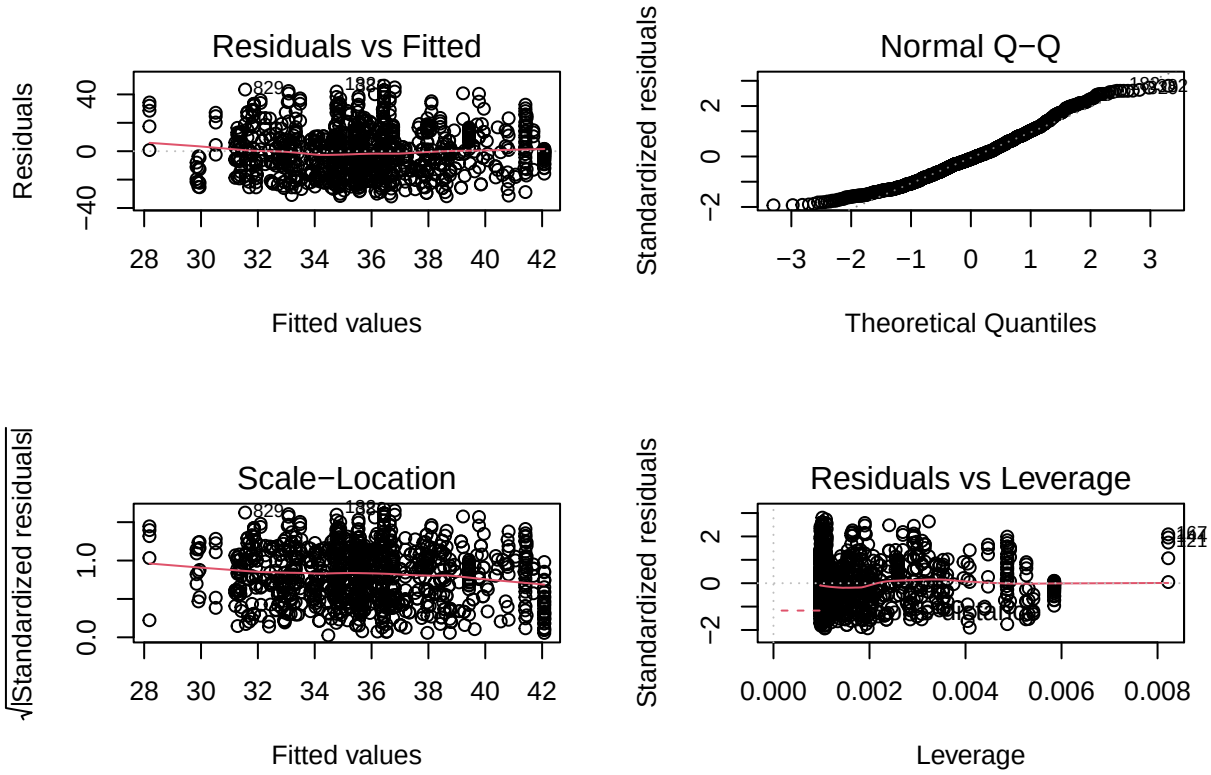
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Fine.Aggregate, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

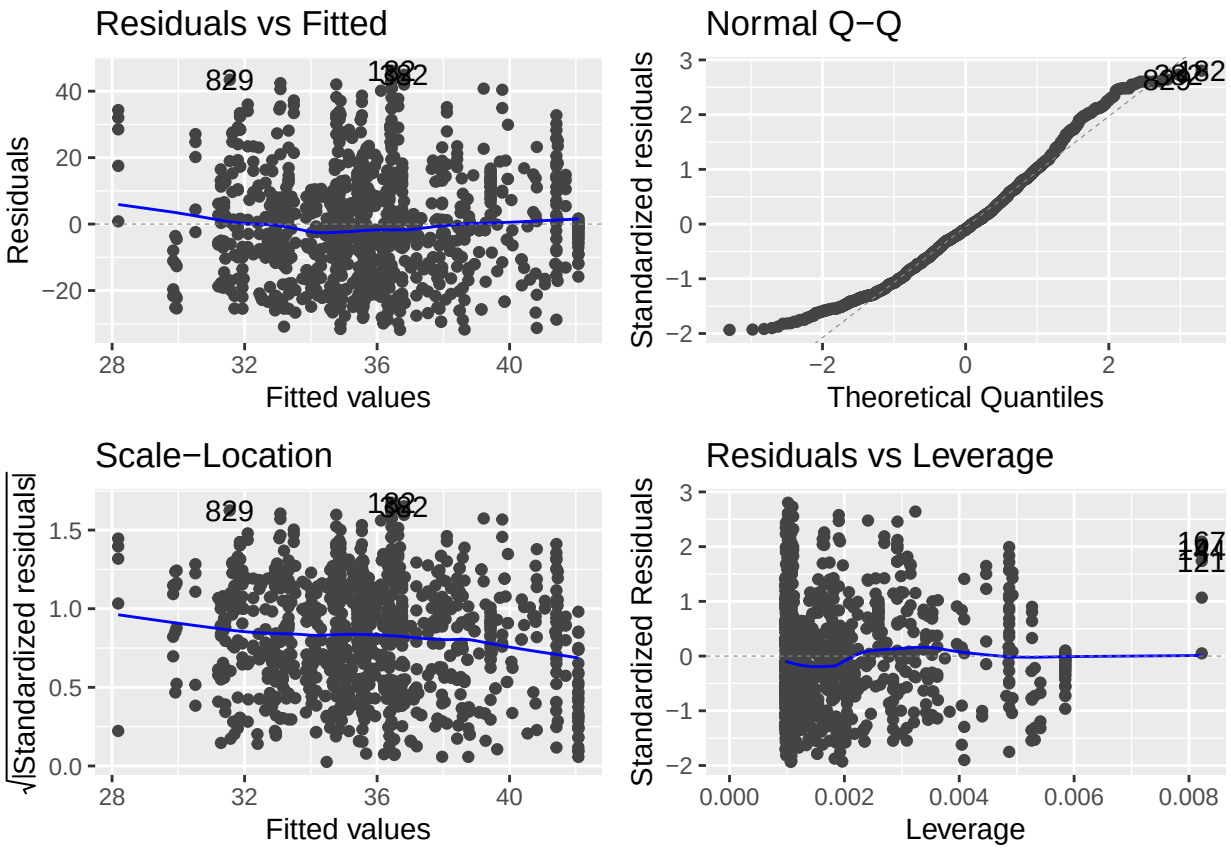
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

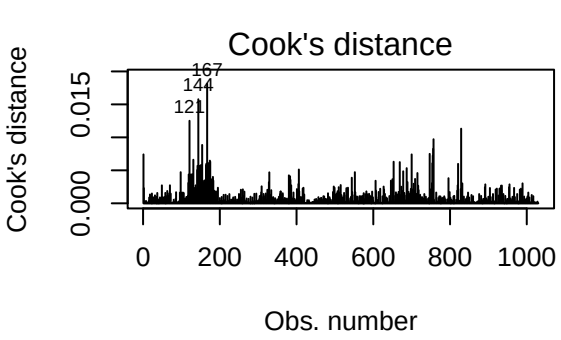


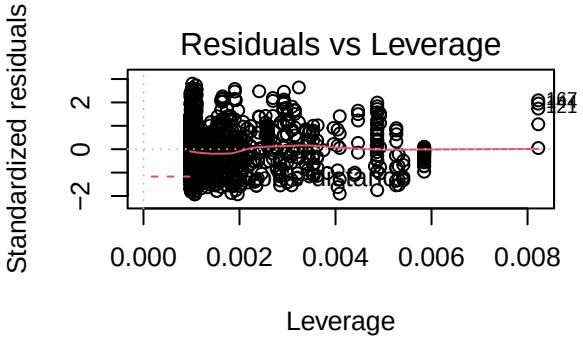

```
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.000
0.015
167
144
121
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−2
0
2
167
144
121
0.000
0.002
0.004
0.006
0.008
Leverage

2.8 one FEATURE

```
#############################################################################################
reg_model <- lm(CCS~Age, data = file)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Age, data = file)
##
## Coefficients:
## (Intercept)          Age
##    31.84659      0.08697
```

```
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Age, data = file)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.512 -11.290  -1.517   9.424  47.468
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 31.846595   0.606952   52.47   <2e-16 ***
## Age          0.086973   0.007789   11.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.78 on 1028 degrees of freedom
## Multiple R-squared:  0.1082, Adjusted R-squared:  0.1073
## F-statistic: 124.7 on 1 and 1028 DF,  p-value: < 2.2e-16
```

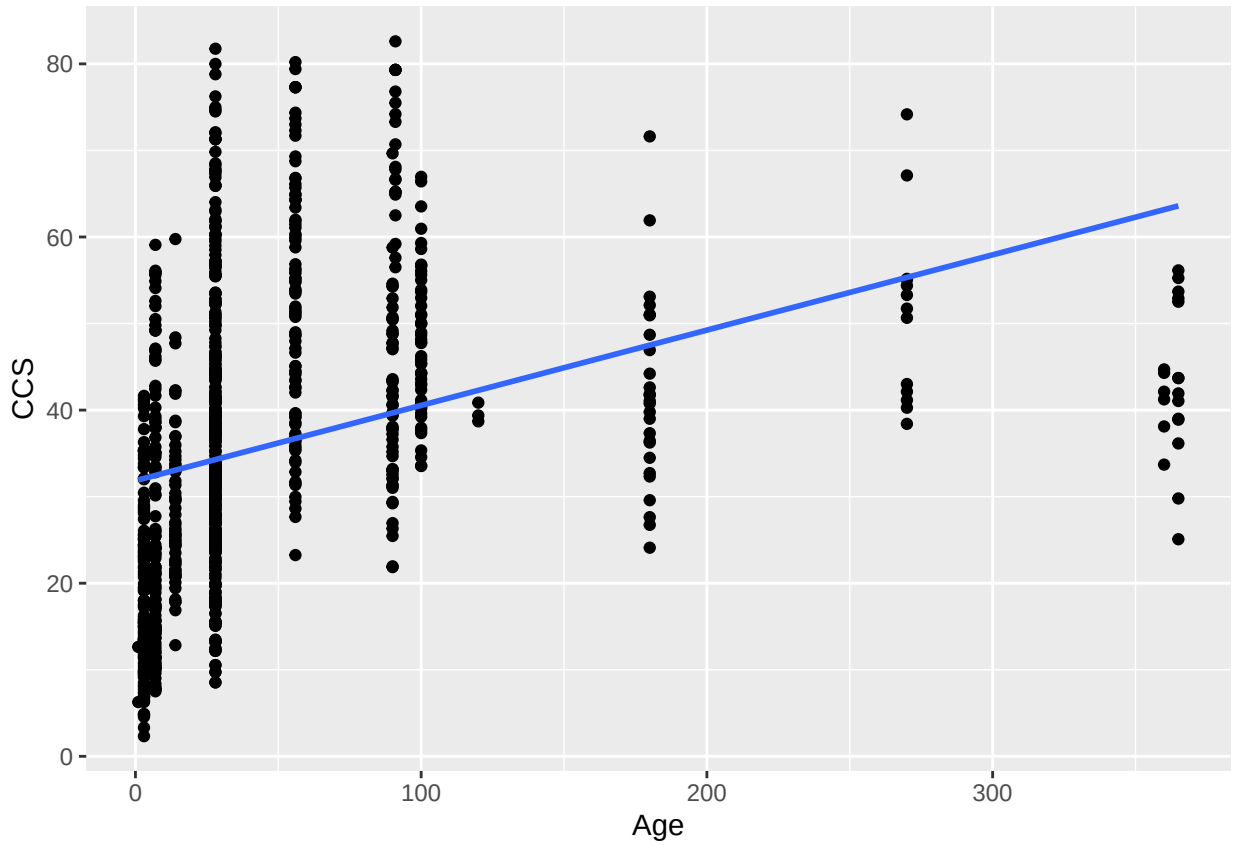
```
summary(reg_model)$coefficient
```

```
##                Estimate  Std. Error  t value      Pr(>|t|)
## (Intercept) 31.84659489 0.606952081 52.46970 5.433711e-293
## Age          0.08697285 0.007789383 11.16556  2.106341e-27
```

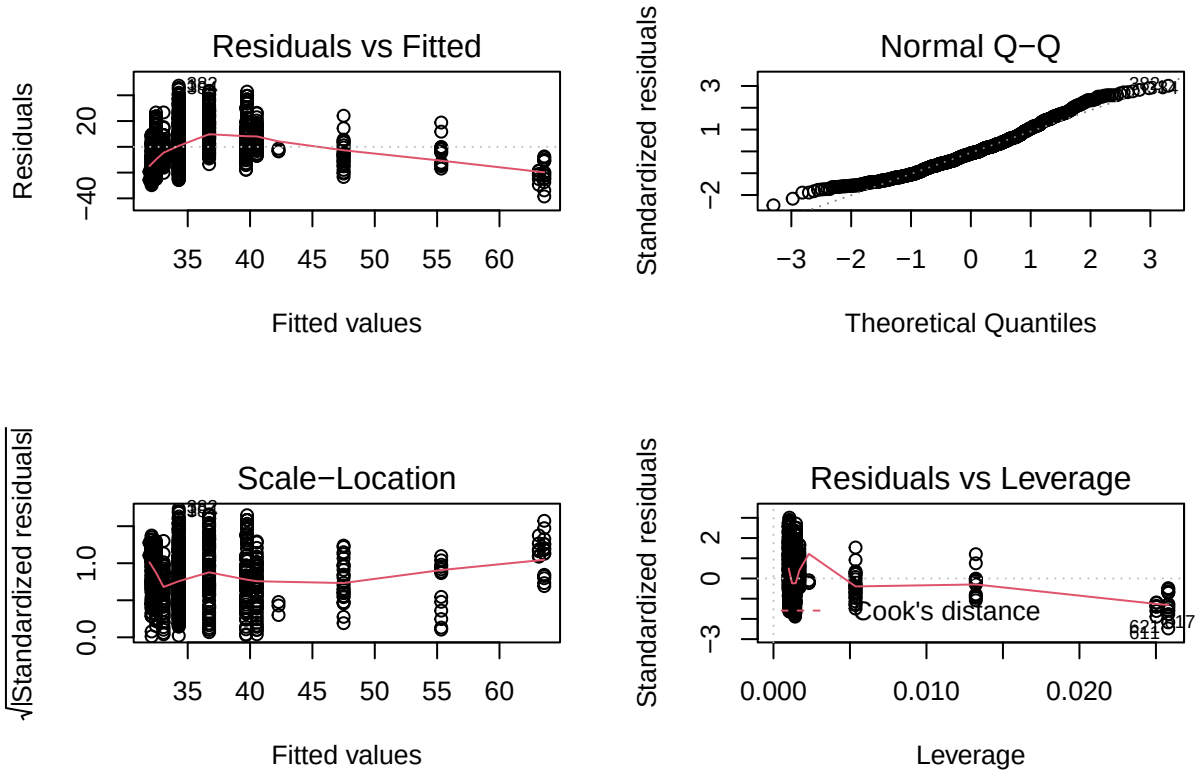
```
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Displaying:
ggplot(reg_model.diagnostics, aes(Age, CCS)) +
geom_point() +
stat_smooth(method = lm, se = FALSE)
```

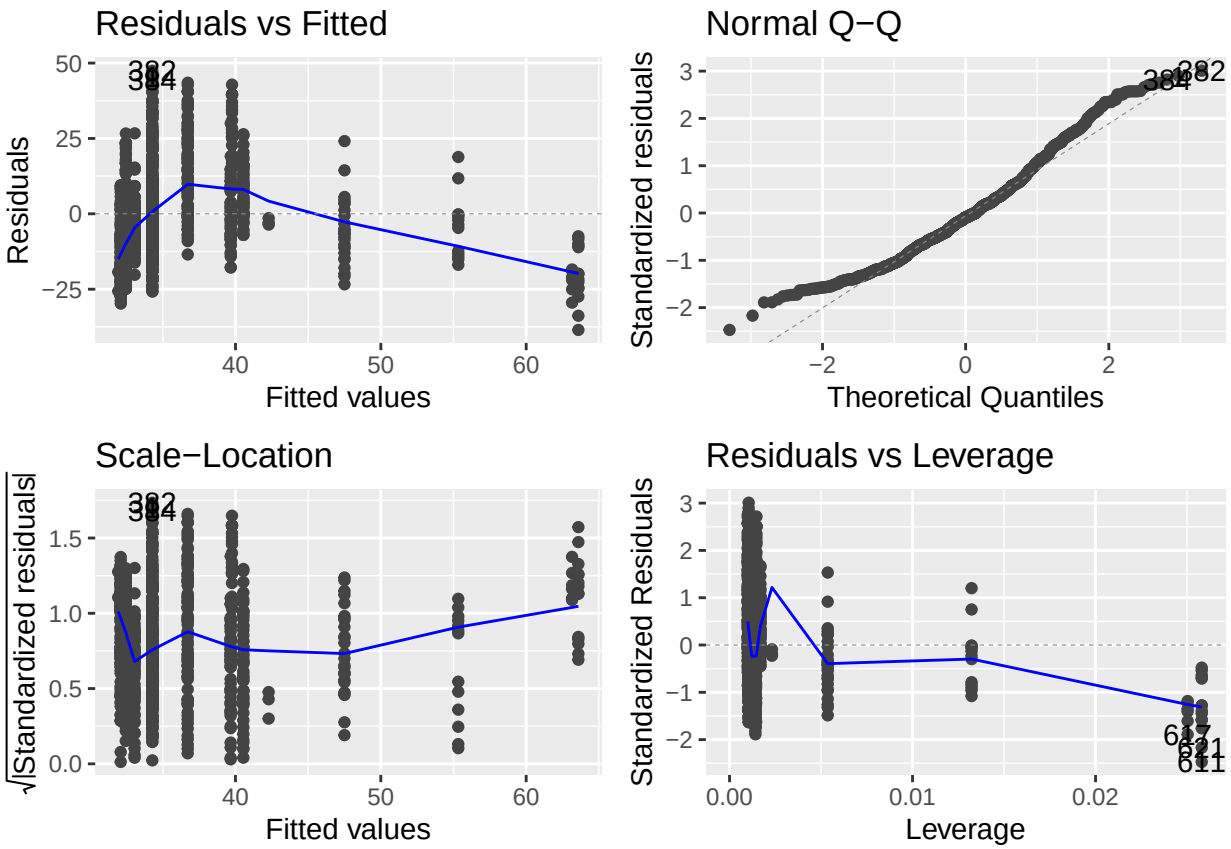
```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

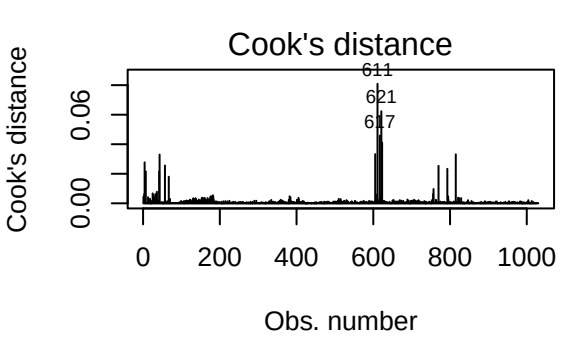


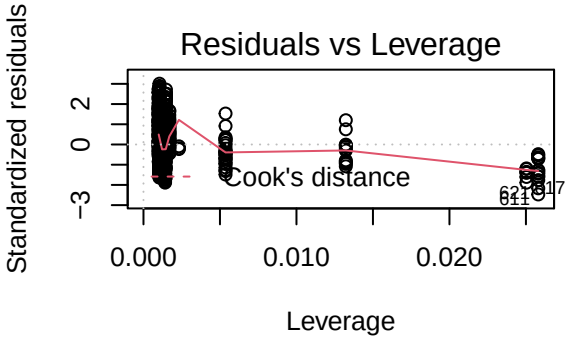

```r
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.00
0.06
611
621
617
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−3
0
2
Cook's distance
0.000
0.010
0.020
Leverage

3. REGRESSION ANALYSIS : the correlations of the features

```
transparentTheme(trans = .4)

correlation_r <- rcorr(as.matrix(file))

corr_graph <- corrplot(correlation_r$r,
                       type = "upper",
                       order = "hclust",
                       p.mat = correlation_r$P,
                       sig.level = 0.05)

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corsig <- corrplot(correlation_r$r,
                   method = "color",
                   col = col(200),
                   type = "upper",
                   order = "hclust",
                   addCoef.col = "black",             # Add coefficient of correlation
                   tl.col = "darkblue", tl.srt = 45, # Text label color and rotation
                   # Combine with significance level
                   p.mat = correlation_r$P,
                   sig.level = 0.05,
                   insig = "blank",
                   # hide correlation coefficient on the principal diagonal
                   diag = FALSE,
                   title = "Correlations between the Features",
                   mar=c(0,0,1,0))
```

Considering the correlation coefficients, we may consider all the features in the LM excepting "Superplasticizer" that is anti-correlated with "Water".

4. REGRESSION ANALYSIS : selecting the independent features

```r
str(file)
```

```
## 'data.frame':    1030 obs. of  9 variables:
##  $ Cement            : num  540 540 332 332 199 ...
##  $ Blast.Furnace.Slag: num  0 0 142 142 132 ...
##  $ Fly.Ash           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water             : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Superplasticizer  : num  2.5 2.5 0 0 0 0 0 0 0 0 ...
##  $ Coarse.Aggregate  : num  1040 1055 932 932 978 ...
##  $ Fine.Aggregate    : num  676 676 594 594 826 ...
##  $ Age               : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ CCS               : num  80 61.9 40.3 41 44.3 ...
```

```r
file2 = subset(file, select = -c(Superplasticizer))
str(file2)
```

```
## 'data.frame':    1030 obs. of  8 variables:
##  $ Cement            : num  540 540 332 332 199 ...
##  $ Blast.Furnace.Slag: num  0 0 142 142 132 ...
##  $ Fly.Ash           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Water             : num  162 162 228 228 192 228 228 228 228 228 ...
##  $ Coarse.Aggregate  : num  1040 1055 932 932 978 ...
##  $ Fine.Aggregate    : num  676 676 594 594 826 ...
##  $ Age               : num  28 28 270 365 360 90 365 28 28 28 ...
##  $ CCS               : num  80 61.9 40.3 41 44.3 ...
```

5. REGRESSION ANALYSIS : the initial MULTI-LINEAR REGRESSION

```r
reg_model <- lm(CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
                      Coarse.Aggregate + Fine.Aggregate + Age,
                      data = file2)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
##     Coarse.Aggregate + Fine.Aggregate + Age, data = file2)
##
## Coefficients:
##        (Intercept)              Cement  Blast.Furnace.Slag             Fly.Ash
##           4.551840            0.119551            0.103540            0.092722
##              Water    Coarse.Aggregate      Fine.Aggregate                 Age
##          -0.216302            0.007713            0.014872            0.114798
```

```r
summary(reg_model)
```

```
##
## Call:
## lm(formula = CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
##     Coarse.Aggregate + Fine.Aggregate + Age, data = file2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.199  -6.318   0.879   6.623  33.783
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       4.551840  25.153883   0.181    0.856
## Cement            0.119551   0.008525  14.024  < 2e-16 ***
## Blast.Furnace.Slag  0.103540   0.010179  10.172  < 2e-16 ***
## Fly.Ash           0.092722   0.012543   7.392 3.00e-13 ***
## Water            -0.216302   0.034261  -6.313 4.06e-10 ***
## Coarse.Aggregate  0.007713   0.008825   0.874    0.382
## Fine.Aggregate    0.014872   0.010611   1.402    0.161
## Age               0.114798   0.005447  21.074  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.44 on 1022 degrees of freedom
## Multiple R-squared:  0.6118, Adjusted R-squared:  0.6092
## F-statistic: 230.1 on 7 and 1022 DF,  p-value: < 2.2e-16
```

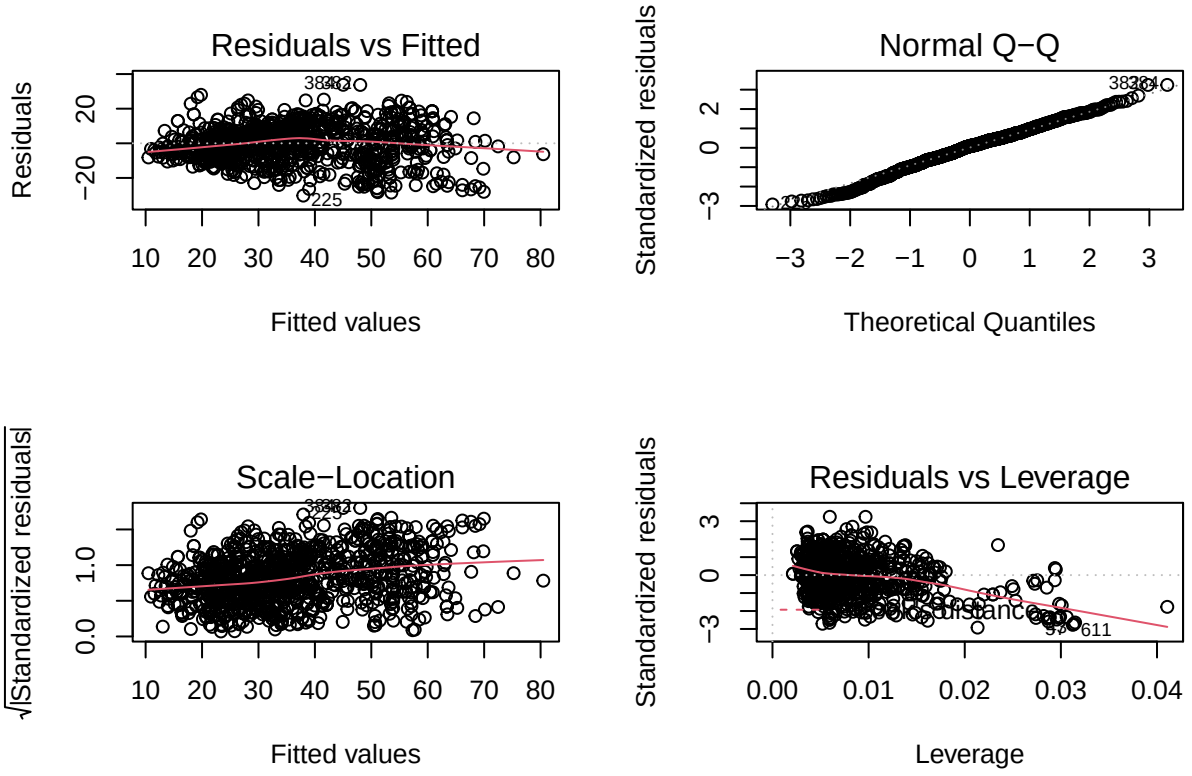
```r
summary(reg_model)$coefficient
```

```
##                       Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)         4.55184041 25.153883453  0.1809597 8.564350e-01
## Cement              0.11955094  0.008525032 14.0235190 5.427547e-41
## Blast.Furnace.Slag  0.10353998  0.010178729 10.1721914 3.223711e-23
## Fly.Ash             0.09272218  0.012543431  7.3920909 3.002851e-13
## Water              -0.21630215  0.034260581 -6.3134407 4.062038e-10
## Coarse.Aggregate    0.00771250  0.008824895  0.8739481 3.823518e-01
```

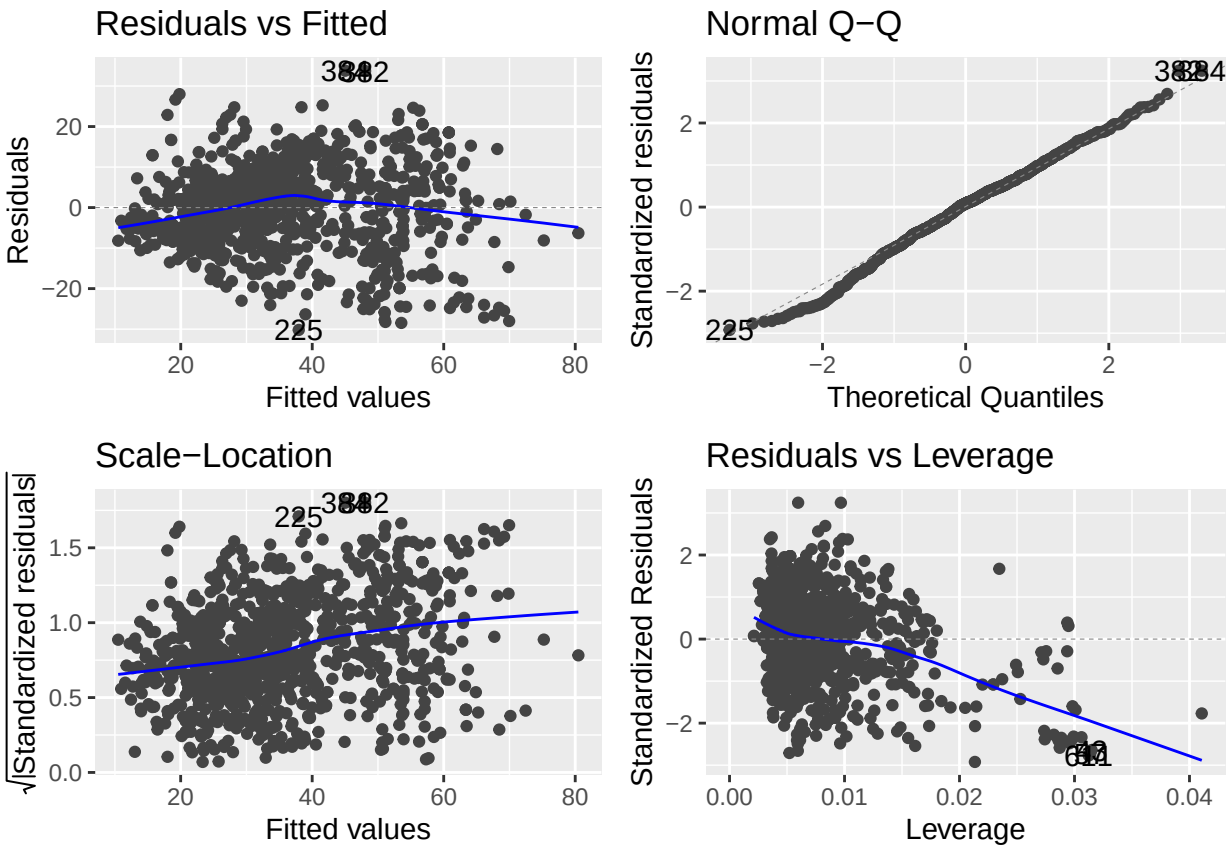
```
## Fine.Aggregate      0.01487168  0.010611045  1.4015282 1.613598e-01
## Age                 0.11479794  0.005447284 21.0743440 3.724317e-82
```

```r
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```

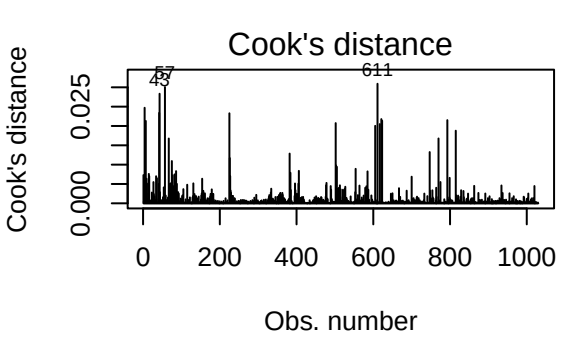


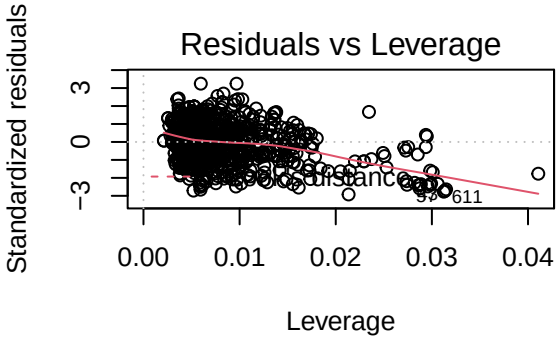

```r
# Diagnostic Plots
autoplot(reg_model)
```




```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)
```


Cook's distance
Cook's distance
0.000
0.025
57
43
611
0
200
400
600
800
1000
Obs. number


Residuals vs Leverage
Standardized residuals
−3
0
3
Cook's distance
611
0.00
0.01
0.02
0.03
0.04
Leverage

6. REGRESSION ANALYSIS : the subsequent MULTI-LINEAR REGRESSION

After examining the F-statistics and the associated p-values, we will keep only the PREDICTOR VARIABLES that are significantly related to the outcome variables.

In our data, the changes in "Coarse.Aggregate" and "Fine.Aggregate" are not associated with the outcome.

```
reg_model <- lm(CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water + Age,
                data = file2)

# Linear Regression Model Summary
reg_model
```

```
##
## Call:
## lm(formula = CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
##     Age, data = file2)
##
## Coefficients:
##        (Intercept)              Cement  Blast.Furnace.Slag             Fly.Ash
##           34.82572             0.11005             0.09236             0.07962
##              Water                 Age
##           -0.25498             0.11402
```

```
summary(reg_model)
```

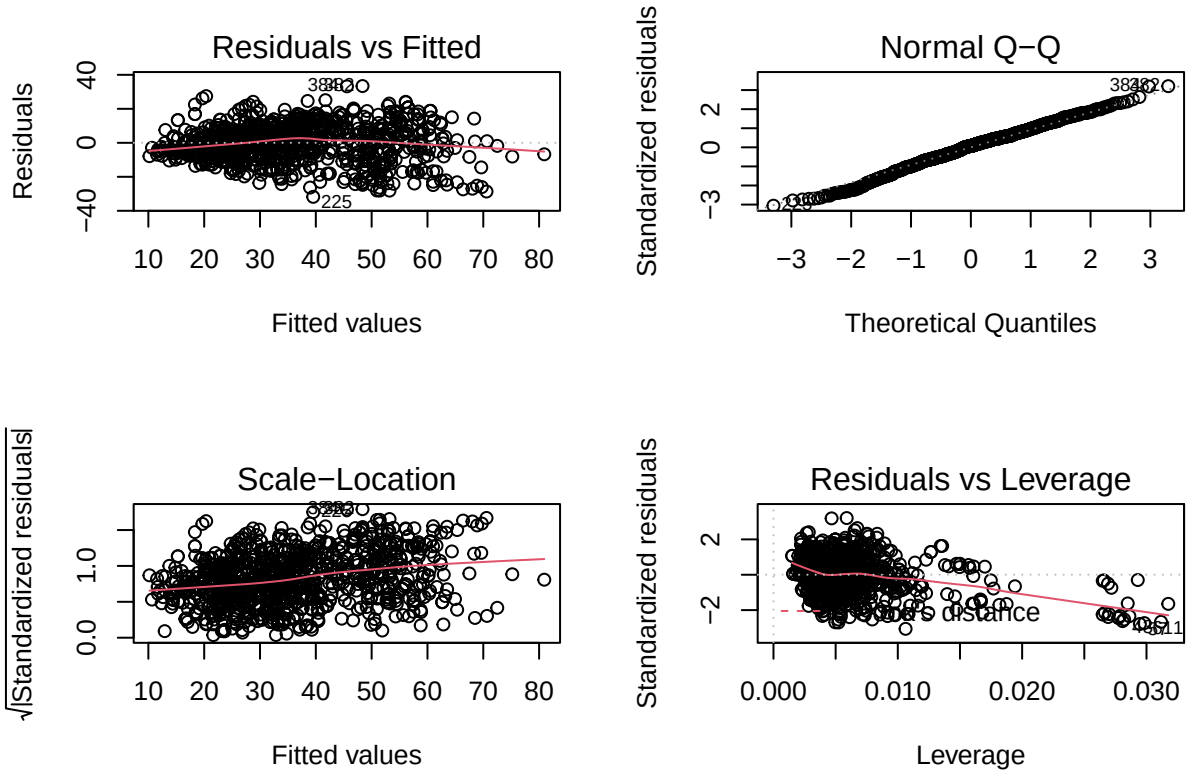
```
##
## Call:
## lm(formula = CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
##     Age, data = file2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.780  -6.440   0.763   6.441  33.385
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        34.825722   3.692092   9.433   <2e-16 ***
## Cement              0.110049   0.003934  27.974   <2e-16 ***
## Blast.Furnace.Slag  0.092365   0.004536  20.361   <2e-16 ***
## Fly.Ash             0.079625   0.006727  11.836   <2e-16 ***
## Water              -0.254982   0.016728 -15.243   <2e-16 ***
## Age                 0.114017   0.005423  21.025   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.45 on 1024 degrees of freedom
## Multiple R-squared:  0.611,  Adjusted R-squared:  0.6091
## F-statistic: 321.6 on 5 and 1024 DF,  p-value: < 2.2e-16
```

```
summary(reg_model)$coefficient
```

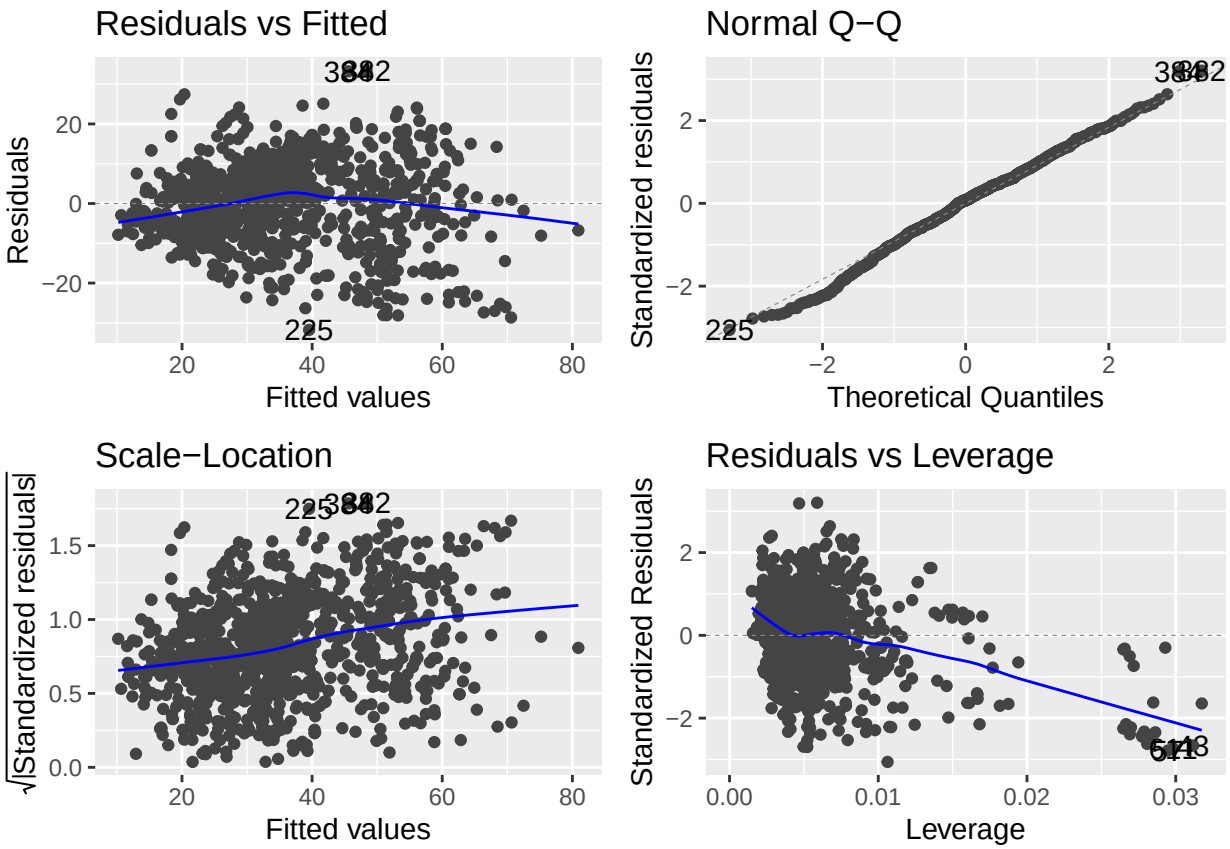
```
##                      Estimate  Std. Error    t value      Pr(>|t|)
## (Intercept)        34.82572232 3.692091539   9.432519  2.594910e-20
## Cement              0.11004918 0.003933942  27.974274 2.202696e-128
## Blast.Furnace.Slag  0.09236487 0.004536385  20.360895  1.200411e-77
## Fly.Ash             0.07962474 0.006727070  11.836467  2.185723e-30
## Water              -0.25498231 0.016727527 -15.243276  1.945008e-47
## Age                 0.11401747 0.005422944  21.025014  7.265453e-82
```

```r
# Making Diagnostic Plots:
reg_model.diagnostics <- augment(reg_model)

# Diagnostic Plots
par(mfrow = c(2, 2))
plot(reg_model)
```




```r
# Diagnostic Plots
autoplot(reg_model)
```

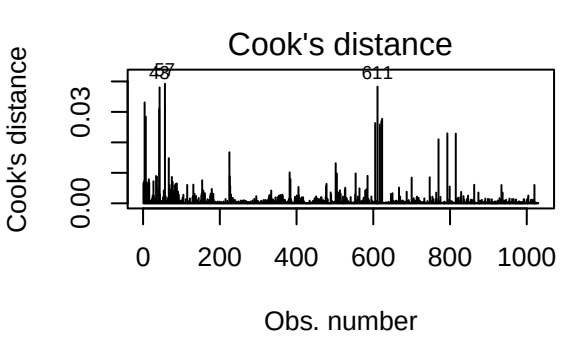


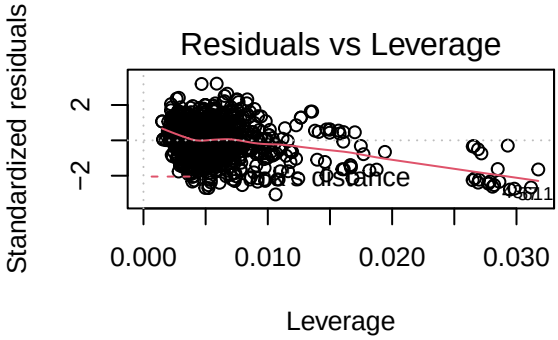

```
# Cook's Distance is used to evaluate the Influential Points that will alter the Regression Analysis
# or the coefficents values.
# By default Cook's Distance more than 4/( n - p - 1) defines an influential value.

plot(reg_model, 4)
# Residuals vs Leverage
plot(reg_model, 5)

# The confidence interval of the model coefficient can be extracted as it follows:
confint(reg_model)
```

```
##                         2.5 %      97.5 %
## (Intercept)        27.58079257 42.07065207
## Cement              0.10232967  0.11776868
## Blast.Furnace.Slag  0.08346319  0.10126654
## Fly.Ash             0.06642433  0.09282516
## Water              -0.28780646 -0.22215817
## Age                 0.10337612  0.12465882
```





Considering **the QQ plots**, we can see that the data is normally distributed (i.e. we do not have to apply additional transformations), and we can write the formula:

```
CCS = 34.82 + 0.11 * Cement + 0.09 * Blast.Furnace.Slag + 0.07 * Fly.Ash - 0.25 * Water + 0.11 *Age
```

We have obtained *Adjusted R-squared : 0.6091* i.e "60% of the variance in the measure of CCS can be predicted by *Cement*, *Blast.Furnace.Slag*, *Fly.Ash*, *Water*, and *Age*".

In contrast, a simple linear model only on *Cement*, *Blast.Furnace.Slag*, *Fly.Ash*, *Water*, and *Age*", provides the following numerical values of *Adjusted R-squared* (as we have noted above, on the previous pages):

"Cement" : Adjusted R-squared: 0.2471

"Blast.Furnace.Slag" : Adjusted R-squared: 0.01722

"Fly.Ash" : Adjusted R-squared: 0.01022

"Water" : Adjusted R-squared: 0.083

"Superplasticizer" : Adjusted R-squared: 0.1332

"Coarse.Aggregate" : Adjusted R-squared: 0.02626

"Fine.Aggregate" : Adjusted R-squared: 0.02702

"Age" : Adjusted R-squared: 0.1073

We can also compute **the Residual Standard Error (RSE)**:

```
sigma(reg_model)/mean(file2$CCS)
```

```
## [1] 0.2916158
```

The RSE estimate gives a measure of error of prediction.

The lower the RSE, the more accurate the model.

7. REGRESSION ANALYSIS by CARET : the TRAINING and the TESTING datasets

```r
set.seed(123)
head(file2)
```

```
##   Cement Blast.Furnace.Slag Fly.Ash Water Coarse.Aggregate Fine.Aggregate Age
## 1  540.0                0.0       0   162           1040.0          676.0  28
## 2  540.0                0.0       0   162           1055.0          676.0  28
## 3  332.5              142.5       0   228            932.0          594.0 270
## 4  332.5              142.5       0   228            932.0          594.0 365
## 5  198.6              132.4       0   192            978.4          825.5 360
## 6  266.0              114.0       0   228            932.0          670.0  90
##     CCS
## 1 79.99
## 2 61.89
## 3 40.27
## 4 41.05
## 5 44.30
## 6 47.03
```

```r
tail(file2)
```

```
##      Cement Blast.Furnace.Slag Fly.Ash Water Coarse.Aggregate Fine.Aggregate
## 1025  166.0              259.7     0.0 183.2            858.8          826.8
## 1026  276.4              116.0    90.3 179.6            870.1          768.3
## 1027  322.2                0.0   115.6 196.0            817.9          813.4
## 1028  148.5              139.4   108.6 192.7            892.4          780.0
## 1029  159.1              186.7     0.0 175.6            989.6          788.9
## 1030  260.9              100.5    78.3 200.6            864.5          761.5
##      Age   CCS
## 1025  28 37.92
## 1026  28 44.28
## 1027  28 31.18
## 1028  28 23.70
## 1029  28 32.77
## 1030  28 32.40
```

```r
trainIndex <- createDataPartition(file2$CCS, p = 0.8, list=FALSE, times=1)
subTrain <- file2[trainIndex,]
subTest <- file2[-trainIndex,]
```

```r
# setup cross validation and control parameters
control <- trainControl(method="repeatedcv", number=3, repeats = 3, verbose = TRUE, search = "grid")
metric <- "RMSE"
tuneLength <- 10
```

8. REGRESSION ANALYSIS by CARET to build a LM considering one predictor

We have chosen here the predictor "Cement" as it provided an "Adjusted R-squared: 0.2471".

```
# Training process
# Fit / train a Linear Regression model to the data

fit.LR <- caret::train(CCS ~ Cement,
                       data=file2,
                       method="lm",
                       metric=metric,
                       preProc=c("center", "scale"),
                       trControl=control,
                       tuneLength = tuneLength)
```

```
## + Fold1.Rep1: intercept=TRUE
## - Fold1.Rep1: intercept=TRUE
## + Fold2.Rep1: intercept=TRUE
## - Fold2.Rep1: intercept=TRUE
## + Fold3.Rep1: intercept=TRUE
## - Fold3.Rep1: intercept=TRUE
## + Fold1.Rep2: intercept=TRUE
## - Fold1.Rep2: intercept=TRUE
## + Fold2.Rep2: intercept=TRUE
## - Fold2.Rep2: intercept=TRUE
## + Fold3.Rep2: intercept=TRUE
## - Fold3.Rep2: intercept=TRUE
## + Fold1.Rep3: intercept=TRUE
## - Fold1.Rep3: intercept=TRUE
## + Fold2.Rep3: intercept=TRUE
## - Fold2.Rep3: intercept=TRUE
## + Fold3.Rep3: intercept=TRUE
## - Fold3.Rep3: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

```
summary(fit.LR)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.593 -10.952  -0.569   9.990  43.240
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.8180     0.4517    79.3   <2e-16 ***
## Cement        8.3167     0.4519    18.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.5 on 1028 degrees of freedom
## Multiple R-squared:  0.2478, Adjusted R-squared:  0.2471
## F-statistic: 338.7 on 1 and 1028 DF,  p-value: < 2.2e-16
```

We have obtained similar results as before: "Adjusted R-squared: 0.2471".

```r
predictions <- predict(fit.LR, newdata = subTest)

rmse <- RMSE(predictions, subTest$CCS)

rmse
```

```
## [1] 14.27825
```

```r
# R2
R2(predictions,  subTest$CCS)
```

```
## [1] 0.2478188
```

```r
# Error Rate
error.rate = rmse/mean(subTest$CCS)
error.rate
```

```
## [1] 0.3952875
```

9. REGRESSION ANALYSIS by CARET to build a LM using multiple predictors

We have chosen now multiple predictors “Cement”, “Blast.Furnace.Slag”, “Fly.Ash”, “Water”, “Age”, as all these provided an “Adjusted R-squared: 0.6091”.

```
# Training process
# Fit / train a Linear Regression model to the data

fit.LR <- caret::train(CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water + Age,
                       data=file2,
                       method="lm",
                       metric=metric,
                       preProc=c("center", "scale"),
                       trControl=control,
                       tuneLength = tuneLength)
```

```
## + Fold1.Rep1: intercept=TRUE
## - Fold1.Rep1: intercept=TRUE
## + Fold2.Rep1: intercept=TRUE
## - Fold2.Rep1: intercept=TRUE
## + Fold3.Rep1: intercept=TRUE
## - Fold3.Rep1: intercept=TRUE
## + Fold1.Rep2: intercept=TRUE
## - Fold1.Rep2: intercept=TRUE
## + Fold2.Rep2: intercept=TRUE
## - Fold2.Rep2: intercept=TRUE
## + Fold3.Rep2: intercept=TRUE
## - Fold3.Rep2: intercept=TRUE
## + Fold1.Rep3: intercept=TRUE
## - Fold1.Rep3: intercept=TRUE
## + Fold2.Rep3: intercept=TRUE
## - Fold2.Rep3: intercept=TRUE
## + Fold3.Rep3: intercept=TRUE
## - Fold3.Rep3: intercept=TRUE
## Aggregating results
## Fitting final model on full training set
```

```
summary(fit.LR)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.780  -6.440   0.763   6.441  33.385
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         35.8180     0.3255  110.05   <2e-16 ***
## Cement              11.5008     0.4111   27.97   <2e-16 ***
## Blast.Furnace.Slag   7.9692     0.3914   20.36   <2e-16 ***
## Fly.Ash              5.0957     0.4305   11.84   <2e-16 ***
## Water               -5.4449     0.3572  -15.24   <2e-16 ***
## Age                  7.2025     0.3426   21.02   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 10.45 on 1024 degrees of freedom
## Multiple R-squared:  0.611,  Adjusted R-squared:  0.6091
## F-statistic: 321.6 on 5 and 1024 DF,  p-value: < 2.2e-16
```

We have obtained similar results as before: "Adjusted R-squared: 0.6091".

```
predictions <- predict(fit.LR, newdata = subTest)

rmse <- RMSE(predictions, subTest$CCS)
rmse
```

```
## [1] 10.26203
```

```
# R2
r2 = R2(predictions,  subTest$CCS)
r2
```

```
## [1] 0.6114459
```

```
# Error Rate
error.rate = rmse/mean(subTest$CCS)
error.rate
```

```
## [1] 0.2841001
```

Indeed, as we have noted before, the **ERROR RATE of MULTIPLE LINEAR REGRESSION** is lower than for **SIMPLE LINEAR REGRESSION**.

We can vary the potential successful models by introducing combinations of **INTERACTION TERMS**.

```
# library(car)
# to compute VIF
# model <- lm(CCS ~., data = subTrain)
# car:: vif(fit.LR)
```

10. REGRESSION ANALYSIS : using POLYNOMIAL REGRESSION

Here we are using another mathematical model based on **POLYNOMIAL REGRESSION**, having a degree 2 polynomial on "Cement". We have chosen this model for illustrative purposes.

```
set.seed(200)
poly_reg <- lm(CCS ~ poly(Cement, 2), data = subTrain)

poly_reg
```

```
##
## Call:
## lm(formula = CCS ~ poly(Cement, 2), data = subTrain)
##
## Coefficients:
##      (Intercept)  poly(Cement, 2)1  poly(Cement, 2)2
##            35.74            239.75            -10.14
```

```
summary(poly_reg)
```

```
##
## Call:
## lm(formula = CCS ~ poly(Cement, 2), data = subTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.784 -11.170  -0.765  10.103  40.744
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        35.7431     0.5064  70.587   <2e-16 ***
## poly(Cement, 2)1 239.7529    14.5531  16.474   <2e-16 ***
## poly(Cement, 2)2 -10.1384    14.5531  -0.697    0.486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.55 on 823 degrees of freedom
## Multiple R-squared:  0.2483, Adjusted R-squared:  0.2465
## F-statistic: 135.9 on 2 and 823 DF,  p-value: < 2.2e-16
```

```
predictions <- poly_reg %>% predict(subTest)

rmse <- RMSE(predictions, subTest$CCS)
rmse
```

```
## [1] 14.30219
```

```
# R2
r2 = R2(predictions,  subTest$CCS)
r2
```

```
## [1] 0.2454989
```

```
# Error Rate
error.rate = rmse/mean(subTest$CCS)
error.rate
```

```
## [1] 0.3959503
```

Shall we compare the **RMSE**, **R2** and **ERROR RATE** in the models that we have buit, we could note that **“one linear predictor”** on “Cement” has the same performance as **“the polynomial predictor”** on “Cement”, while **“the multiple linear predictor”** performs better.

one linear predictor (on “Cement”):

RMSE : 14

R2 : 0.24

error.rate : 0.39

multiple linear predictor (on “Cement”, “Blast.Furnace.Slag”,“Fly.Ash”, “Water”, “Age”):

RMSE : 10.26

R2 : 0.61

error.rate 0.28

polynomial predictor (on “Cement”):

RMSE : 14.3

R2 : 0.24

error.rate : 0.39

11. REGRESSION ANALYSIS : using SPLINES

```r
set.seed(200)
library(splines)

knots <- quantile(subTrain$CCS, p = c( 0.25, 0.5, 0.75))
knots
```

```
##    25%    50%    75%
## 23.710 34.445 46.135
```

```r
splinemodel <- lm(CCS ~ bs(Cement, knots = knots), data = subTrain)

splinemodel
```

```
##
## Call:
## lm(formula = CCS ~ bs(Cement, knots = knots), data = subTrain)
##
## Coefficients:
##                 (Intercept)  bs(Cement, knots = knots)1
##                      54.467                          NA
## bs(Cement, knots = knots)2  bs(Cement, knots = knots)3
##                          NA                     -33.217
## bs(Cement, knots = knots)4  bs(Cement, knots = knots)5
##                     -22.124                      -7.305
## bs(Cement, knots = knots)6
##                          NA
```

```r
summary(splinemodel)
```

```
##
## Call:
## lm(formula = CCS ~ bs(Cement, knots = knots), data = subTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.647 -11.171  -0.684  10.210  40.794
##
## Coefficients: (3 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   54.467      2.893  18.828  < 2e-16 ***
## bs(Cement, knots = knots)1       NA         NA      NA       NA
## bs(Cement, knots = knots)2       NA         NA      NA       NA
## bs(Cement, knots = knots)3  -33.217      4.352  -7.632 6.40e-14 ***
## bs(Cement, knots = knots)4  -22.124      4.311  -5.132 3.57e-07 ***
## bs(Cement, knots = knots)5   -7.305      7.320  -0.998    0.319
## bs(Cement, knots = knots)6       NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.56 on 822 degrees of freedom
## Multiple R-squared:  0.2484, Adjusted R-squared:  0.2457
## F-statistic: 90.57 on 3 and 822 DF,  p-value: < 2.2e-16
```

```r
# Make predictions
predictions <- splinemodel %>% predict(subTest)
```

```r
# Model performance
rmse = RMSE(predictions, subTest$CCS)
rmse
```

```
## [1] 14.31504
```

```r
# R2
r2 = R2(predictions, subTest$CCS)
r2
```

```
## [1] 0.2441709
```

```r
# Error Rate
error.rate = rmse/mean(subTest$CCS)
error.rate
```

```
## [1] 0.396306
```

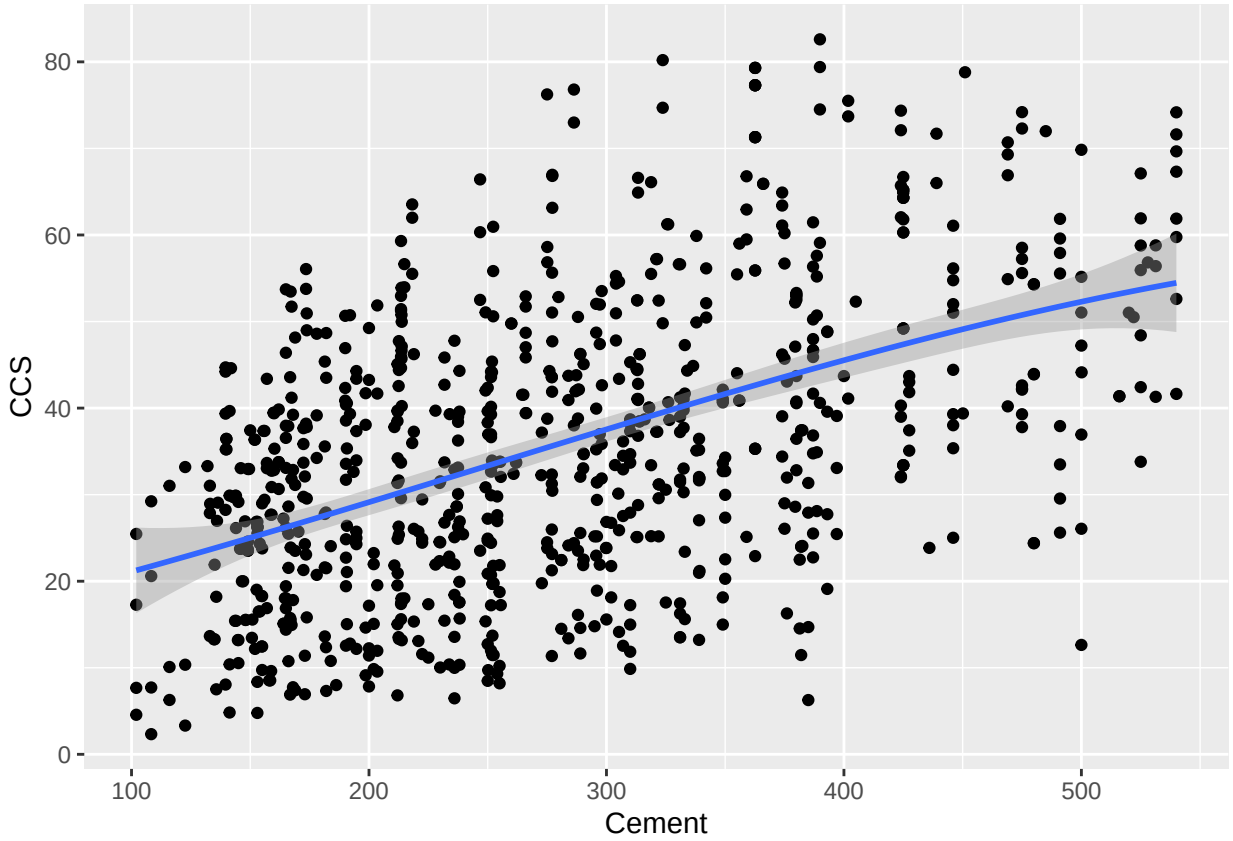
As we can note above, **the REGRESSION MODEL using the SPLINES on one variable** ("Cement") performs as well as **a SIMPLE LINEAR REGRESION MODEL on the same variable ("Cement")**, and as well as **a POLYNOMIAL MODEL on the same variable ("Cement")**. More precisely:

RMSE : 14.31

R2 : 0.244

error.rate : 0.39

```r
ggplot(subTrain, aes(Cement, CCS)) +
                 geom_point() +
                 stat_smooth(method = lm, formula = y ~ splines:: bs(x, df = 3))
```


80
60
40
20
0
CCS
100
200
300
400
500
Cement

12. REGRESSION ANALYSIS : using GAM

```r
set.seed(200)
library(mgcv)

gmmodel <- gam(CCS ~ s(Cement), data = subTrain)

gmmodel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## CCS ~ s(Cement)
##
## Estimated degrees of freedom:
## 4.41  total = 5.41
##
## GCV score: 212.279
```

```r
summary(gmmodel)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## CCS ~ s(Cement)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.7431     0.5053   70.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F p-value
## s(Cement) 4.41  5.454 50.68  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.25   Deviance explained = 25.4%
## GCV = 212.28  Scale est. = 210.89    n = 826
```

```r
gampredictions <- gmmodel %>% predict(subTest)

# Model performance
rmse = RMSE(gampredictions, subTest$CCS)
rmse
```

```
## [1] 14.32231
```

```r
# R2
r2 = R2(gampredictions, subTest$CCS)
r2
```

```
## [1] 0.2433835
```

```r
# Error Rate
error.rate = rmse/mean(subTest$CCS)
error.rate
```

```
## [1] 0.3965073
```

As we could note also for **the GAM-based approach**, the performance of the model on one feature "Cement" is the same as **the SPLINE-based approach**, or as **the POLYNOMIAL-based approach**, namely:

RMSE : 14.32

R2 : 0.24

error.rate : 0.39

13. REGRESSION ANALYSIS : MODEL SELECTION based on BIC

As we know from the class, for example, the lower the AIC, the better the model is.

Here, we compare the models by considering a metrics based on Adj.R2, CP and BIC.

We use the function **"regsubsets"** in the package "leaps" that gives the information :

rsq : The r-squared for each model

rss : Residual sum of squares for each model

adjr2 : Adjusted r-squared

cp : Mallows' Cp

bic : Schwartz's information criterion, BIC

```
checkmodels <- regsubsets(CCS ~., data = file2, nvmax = 13)

summary(checkmodels)
```

```
## Subset selection object
## Call: regsubsets.formula(CCS ~ ., data = file2, nvmax = 13)
## 7 Variables  (and intercept)
##                     Forced in Forced out
## Cement                  FALSE      FALSE
## Blast.Furnace.Slag      FALSE      FALSE
## Fly.Ash                 FALSE      FALSE
## Water                   FALSE      FALSE
## Coarse.Aggregate        FALSE      FALSE
## Fine.Aggregate          FALSE      FALSE
## Age                     FALSE      FALSE
## 1 subsets of each size up to 7
## Selection Algorithm: exhaustive
##          Cement Blast.Furnace.Slag Fly.Ash Water Coarse.Aggregate
## 1  ( 1 ) "*"    " "                " "     " "   " "
## 2  ( 1 ) "*"    " "                " "     " "   " "
## 3  ( 1 ) "*"    " "                " "     "*"   " "
## 4  ( 1 ) "*"    "*"                " "     "*"   " "
## 5  ( 1 ) "*"    "*"                "*"     "*"   " "
## 6  ( 1 ) "*"    "*"                "*"     "*"   " "
## 7  ( 1 ) "*"    "*"                "*"     "*"   "*"
##          Fine.Aggregate Age
## 1  ( 1 ) " "            " "
## 2  ( 1 ) " "            "*"
## 3  ( 1 ) " "            "*"
## 4  ( 1 ) " "            "*"
## 5  ( 1 ) " "            "*"
## 6  ( 1 ) "*"            "*"
## 7  ( 1 ) "*"            "*"
```

```
str(checkmodels)
```

```
## List of 28
##  $ np       : int 8
```

```
##  $ nrbar    : int 28
##  $ d        : num [1:8] 1030 6221002 453634 4806435 3906306 ...
##  $ rbar     : num [1:28] 972.9 181.6 773.6 54.2 73.9 ...
##  $ thetab   : num [1:8] 35.818 -0.0354 -0.2587 -0.1022 -0.0615 ...
##  $ first    : int 2
##  $ last     : int 8
##  $ vorder   : int [1:8] 1 6 5 7 4 3 8 2
##  $ tol      : num [1:8] 1.60e-08 1.79e-05 3.40e-06 1.54e-05 2.56e-06 ...
##  $ rss      : num [1:8] 287175 279363 249003 198784 183992 ...
##  $ bound    : num [1:8] 287175 216003 192009 157410 127003 ...
##  $ nvmax    : int 8
##  $ ress     : num [1:8, 1] 287175 216003 192009 157410 127003 ...
##  $ ir       : int 8
##  $ nbest    : int 1
##  $ lopt     : int [1:36, 1] 1 1 2 1 2 8 1 2 5 8 ...
##  $ il       : int 36
##  $ ier      : int 0
##  $ xnames   : chr [1:8] "(Intercept)" "Cement" "Blast.Furnace.Slag" "Fly.Ash" ...
##  $ method   : chr "exhaustive"
##  $ force.in : Named logi [1:8] TRUE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:8] "" "Cement" "Blast.Furnace.Slag" "Fly.Ash" ...
##  $ force.out: Named logi [1:8] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:8] "" "Cement" "Blast.Furnace.Slag" "Fly.Ash" ...
##  $ sserr    : num 111471
##  $ intercept: logi TRUE
##  $ lindep   : logi [1:8] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ nullrss  : num 287175
##  $ nn       : int 1030
##  $ call     : language regsubsets.formula(CCS ~ ., data = file2, nvmax = 13)
##  - attr(*, "class")= chr "regsubsets"
```

```
res.sum <- summary(checkmodels)

data.frame(Adj.R2 = which.max(res.sum$adjr2),
           CP = which.min(res.sum$cp),
           BIC = which.min(res.sum$bic))
```

```
##   Adj.R2 CP BIC
## 1      6  5   5
```

As we can note again, the model that works the best based on BIC and CP is **the model 5** that includes all the features, except "Coarse.Aggregate" and "Fine.Aggregate", and that is the model that we have chosen to work with in the section on *MULTI-LINEAR REGRESSION*.

```
##          Cement Blast.Furnace.Slag Fly.Ash Water Coarse.Aggregate
## 5  ( 1 ) "*"    "*"                "*"     "*"   " "
## 6  ( 1 ) "*"    "*"                "*"     "*"   " "
## 7  ( 1 ) "*"    "*"                "*"     "*"   "*"
##          Fine.Aggregate Age
## 5  ( 1 ) " "            "*"
## 6  ( 1 ) "*"            "*"
## 7  ( 1 ) "*"            "*"
```

14. REGRESSION ANALYSIS : MODEL SELECTION based on AIC

```r
# the best AIC model:

full.model <- lm(CCS ~., data = file2)

step.model <- stepAIC(full.model, direction = "both", trace = FALSE)

summary(step.model)
```

```
##
## Call:
## lm(formula = CCS ~ Cement + Blast.Furnace.Slag + Fly.Ash + Water +
##     Age, data = file2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.780  -6.440   0.763   6.441  33.385
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        34.825722   3.692092   9.433   <2e-16 ***
## Cement              0.110049   0.003934  27.974   <2e-16 ***
## Blast.Furnace.Slag  0.092365   0.004536  20.361   <2e-16 ***
## Fly.Ash             0.079625   0.006727  11.836   <2e-16 ***
## Water              -0.254982   0.016728 -15.243   <2e-16 ***
## Age                 0.114017   0.005423  21.025   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.45 on 1024 degrees of freedom
## Multiple R-squared:  0.611,  Adjusted R-squared:  0.6091
## F-statistic: 321.6 on 5 and 1024 DF,  p-value: < 2.2e-16
```

It is indeed an additional confirmation that we shall include in the model the following attributes (features) : “Cement”, “Blast.Furnace.Slag”, “Fly.Ash”, “Water”, “Age” (as we have already done above in the previous sections).

And, the adjusted R-squared: 0.6091.

15. REGRESSION ANALYSIS : using RIDGE REGRESSION

```r
# in this section, we are resuming the MULTI LINEAR MODEL
# fit.LR
# summary(fit.LR)
# head(subTrain)
# tail(subTrain)

# x <- model.matrix(CCS ~., subTrain)[,-8]
x <- model.matrix(CCS ~., subTrain)
y <- subTrain$CCS

# Alpha 0 for Ridge ,
# Alpha 1 for Lasso .

cv <- cv.glmnet(x, y, alpha = 0)
# Display the best lambda value which signifies the best shrinkage
cv$lambda.min ##  Fit the final model on the training data
```

```
## [1] 0.8342069
```

```r
model <- glmnet(x, y, alpha = 0, lambda = cv$lambda.min) # Display regression coefficients coef (model)

coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)        89.60114215
## (Intercept)         .
## Cement              0.08382335
## Blast.Furnace.Slag  0.06330766
## Fly.Ash             0.04422134
## Water              -0.30288268
## Coarse.Aggregate   -0.01939776
## Fine.Aggregate     -0.02024226
## Age                 0.11068502
```

```r
# Predictions on the test data

x.test <- model.matrix(CCS ~., subTest)

predictions <- model %>% predict(x.test) %>% as.vector()

# Model performance metrics

data.frame( rmse = RMSE(predictions, subTest$CCS),
            Rsquare = R2(predictions, subTest$CCS))
```

```
##       rmse   Rsquare
## 1 10.41408 0.6012527
```

The results are similar to the typical multi-linear regression model that we have applied above that has showed :

RMSE : 10.26

R2 : 0.61

16. REGRESSION ANALYSIS : using LASSO REGRESSION

```r
# in this section, we are resuming the MULTI LINEAR MODEL
# fit.LR
# summary(fit.LR)
# head(subTrain)
# tail(subTrain)

# x <- model.matrix(CCS ~., subTrain)[,-8]
x <- model.matrix(CCS ~., subTrain)
y <- subTrain$CCS

# Alpha 0 for Ridge ,
# Alpha 1 for Lasso .

cv <- cv.glmnet(x, y, alpha = 1)
# Display the best lambda value which signifies the best shrinkage
cv$lambda.min ##  Fit the final model on the training data
```

```
## [1] 0.06023648
```

```r
model <- glmnet(x, y, alpha = 1, lambda = cv$lambda.min) # Display regression coefficients coef (model)

coef(model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)        36.2916051478
## (Intercept)         .
## Cement              0.1064563002
## Blast.Furnace.Slag  0.0897901185
## Fly.Ash             0.0750924519
## Water              -0.2520642067
## Coarse.Aggregate   -0.0015020444
## Fine.Aggregate      0.0008735112
## Age                 0.1180685974
```

```r
# Predictions on the test data

x.test <- model.matrix(CCS ~., subTest)

predictions <- model %>% predict(x.test) %>% as.vector()

# Model performance metrics

data.frame( rmse = RMSE(predictions, subTest$CCS),
            Rsquare = R2(predictions, subTest$CCS))
```

```
##       rmse   Rsquare
## 1 10.32261 0.6068977
```

The results are similar to the typical multi-linear regression model that we have applied above that has showed :

RMSE : 10.26

R2 : 0.61