*ASSIGNMENT- 4(Faithful Data)*

# Install and load the caret package

library(caret)

data("faithful")

#Performing the sub-sampling

set.seed(3569)

trainfaithful = rbinom(272, size = 1, prob = 0.5)

table(trainfaithful)

trainset = faithful[ trainfaithful == 1,]

testset = faithful[ trainfaithful == 0,]

*Output:*

# Install and load the caret package

> library(caret)

> data("faithful")

> #Performing the sub-sampling

> set.seed(3569)

> trainfaithful = rbinom(272, size = 1, prob = 0.5)

> table(trainfaithful)

trainfaithful

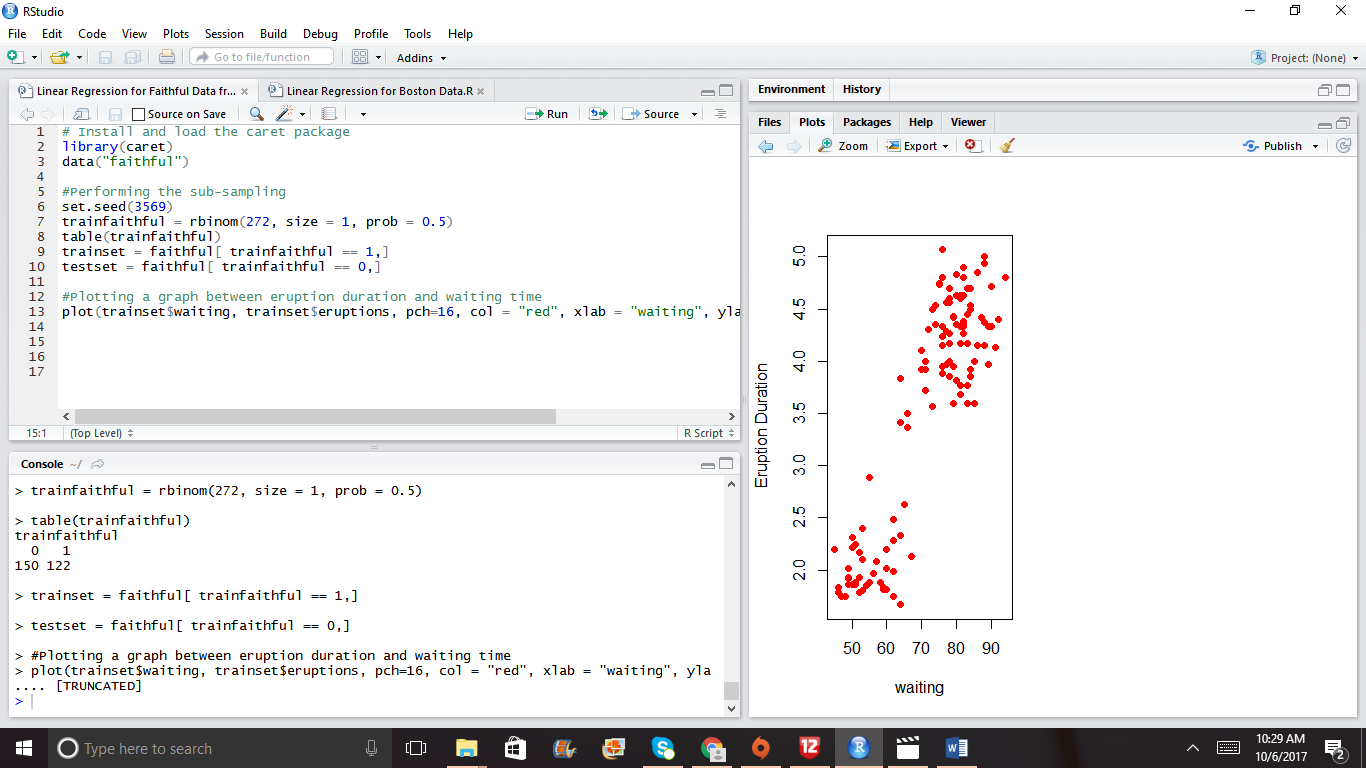
0 1

150 122

#Plotting a graph between eruption duration and waiting time

plot(trainset$waiting, trainset$eruptions, pch=16, col = "red", xlab = "waiting", ylab = " Eruption Duration")

*Output:*



#In order to linearly re-produce the data, Fit a linear model

fit1 <- lm(eruptions ~ waiting, data = trainset)

summary(fit1)

*Output:*

Call:

lm(formula = eruptions ~ waiting, data = trainset)

Residuals:

Min 1Q Median 3Q Max

-1.29122 -0.39601 0.04915 0.35541 1.21344

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.816919 0.250362 -7.257 4.26e-11 \*\*\*

waiting 0.074612 0.003457 21.582 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5128 on 120 degrees of freedom

Multiple R-squared: 0.7951, Adjusted R-squared: 0.7934

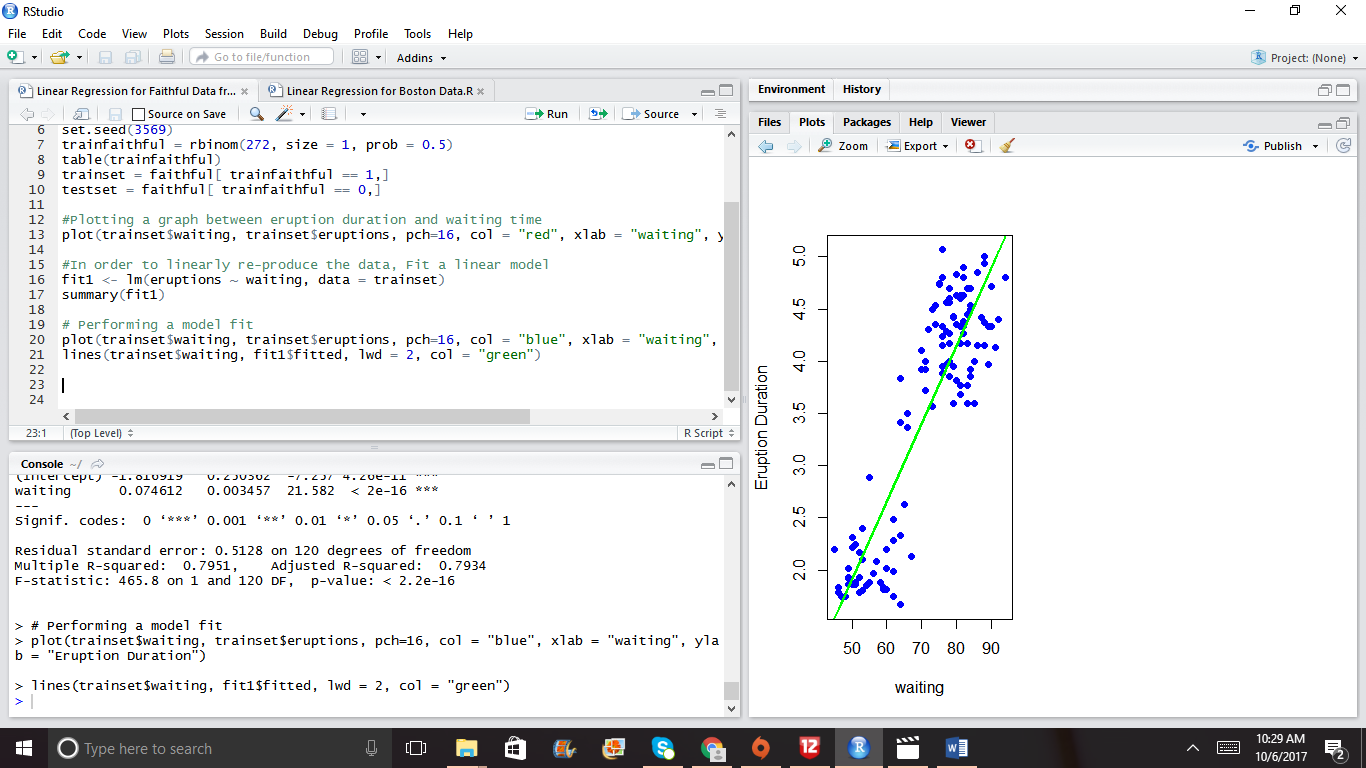
F-statistic: 465.8 on 1 and 120 DF, p-value: < 2.2e-16

# Performing a model fit

plot(trainset$waiting, trainset$eruptions, pch=16, col = "blue", xlab = "waiting", ylab = "Eruption Duration")

lines(trainset$waiting, fit1$fitted, lwd = 2, col = "green")

*Output:*



#Predicting a new value with help of intercept and slope coefficient

coef(fit1)[1] + coef(fit1)[2] \*30

#Predicting a value with the help of the predict function

Ndata <- data.frame(waiting = 30)

predict(fit1, Ndata)

*Output:*

#Predicting a new value with help of intercept and slope coefficient

> coef(fit1)[1] + coef(fit1)[2] \*30

(Intercept)

0.4214299

> #Predicting a value with the help of the predict function

> Ndata <- data.frame(waiting = 30)

> predict(fit1, Ndata)

1

0.4214299

#Plot predictions for the training set and test set having one row and two columns

par(mfrow=c(1,2))

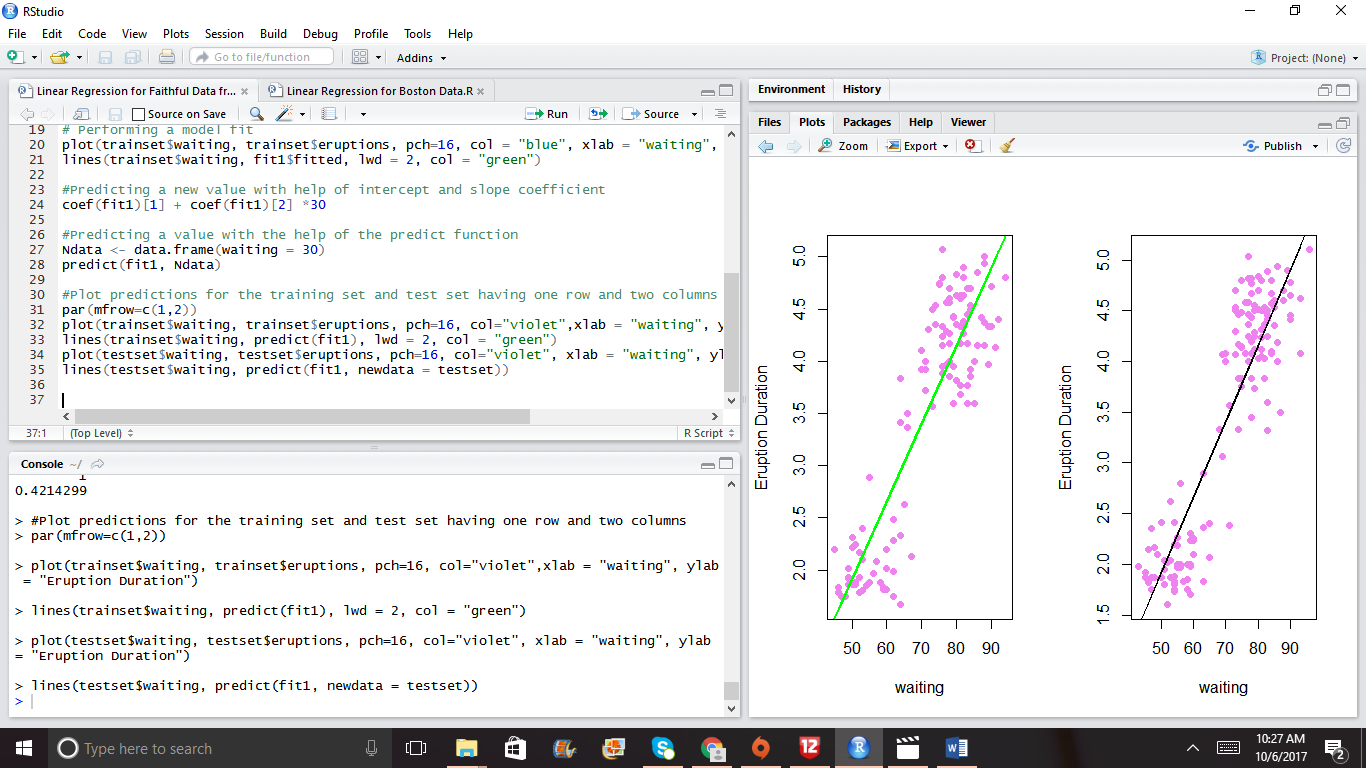
plot(trainset$waiting, trainset$eruptions, pch=16, col="violet",xlab = "waiting", ylab = "Eruption Duration")

lines(trainset$waiting, predict(fit1), lwd = 2, col = "green")

plot(testset$waiting, testset$eruptions, pch=16, col="violet", xlab = "waiting", ylab = "Eruption Duration")

lines(testset$waiting, predict(fit1, newdata = testset))

Output:



#Calculate Root Mean Square Error value on train set (where the distance between the fitted and the original)

sqrt(sum((fit1$fitted - trainset$eruptions)^2))

#Calculate Root Mean Square Error value on test set

sqrt(sum((predict(fit1,newdata = testset) - testset$eruptions)^2))

Output:

|  |
| --- |
| > sqrt(sum((fit1$fitted - trainset$erup .... [TRUNCATED]  [1] 5.61694  > #Calculate Root Mean Square Error value on test set  > sqrt(sum((predict(fit1,newdata = testset) - testset$eruptions)^2))  [1] 5.92658 |
|  |
| |  | | --- | | #Prediction Intervals  prediction <- predict(fit1, newdata = testset, interval = "predict")  ord <- order(testset$waiting)  plot(testset$waiting, testset$eruptions, pch=16,col = "blue")  matlines(testset$waiting[ord],prediction[ord,], type = "l", col = c(1,2,2), lty = c(1,1,1), lwd=2) | |