Intro to Data Science - Lab 8

Week 8 - Linear Models

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this lab assignment by myself, with help from the book and the professor.
```

Linear modeling, also referred to as **regression analysis** or multiple regression **bold text**, is a technique for fitting a line, plane, or higher order linear object to data. In their simplest form, linear models have one metric **outcome variable** and one or more **predictor variables** (any combination of metric values, ordered scales such as ratings, or dummy codes).

Make sure to library the **MASS** and **ggplot2** packages before running the following:

```
ggplot(data=Boston) + aes(x=rm, y=medv) + geom_point() +
 geom_smooth(method="lm", se=FALSE)
```

```
# install.packages("MASS")
# install.packages("ggplot2")

library(MASS)
library(ggplot2)
```

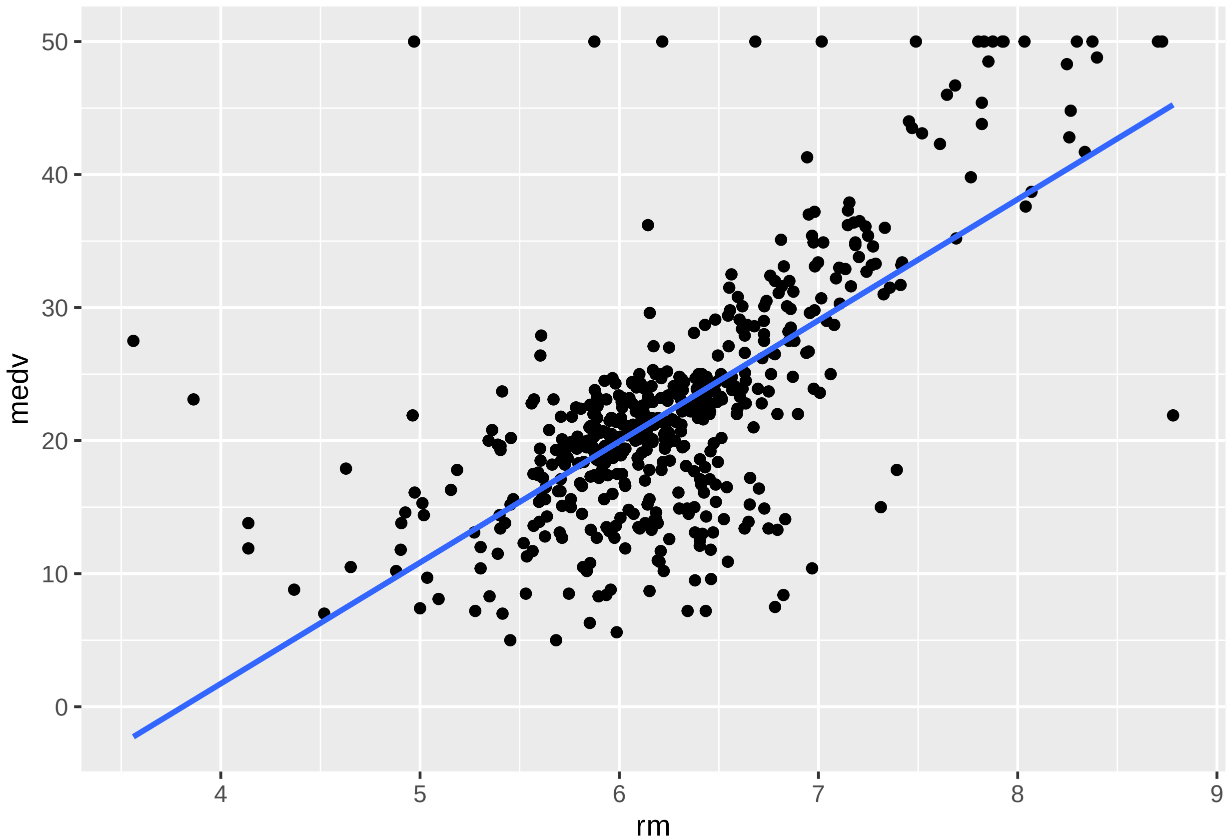
1. Explore this dataset descrption by typing `?Boston` in a code cell.

```
?Boston

# The dataset is about Housing Values in Suburbs of Boston.
# 506 rows and 14 columns

ggplot(data=Boston) + aes(x=rm, y=medv) + geom_point() +
 geom_smooth(method="lm", se=FALSE)
```

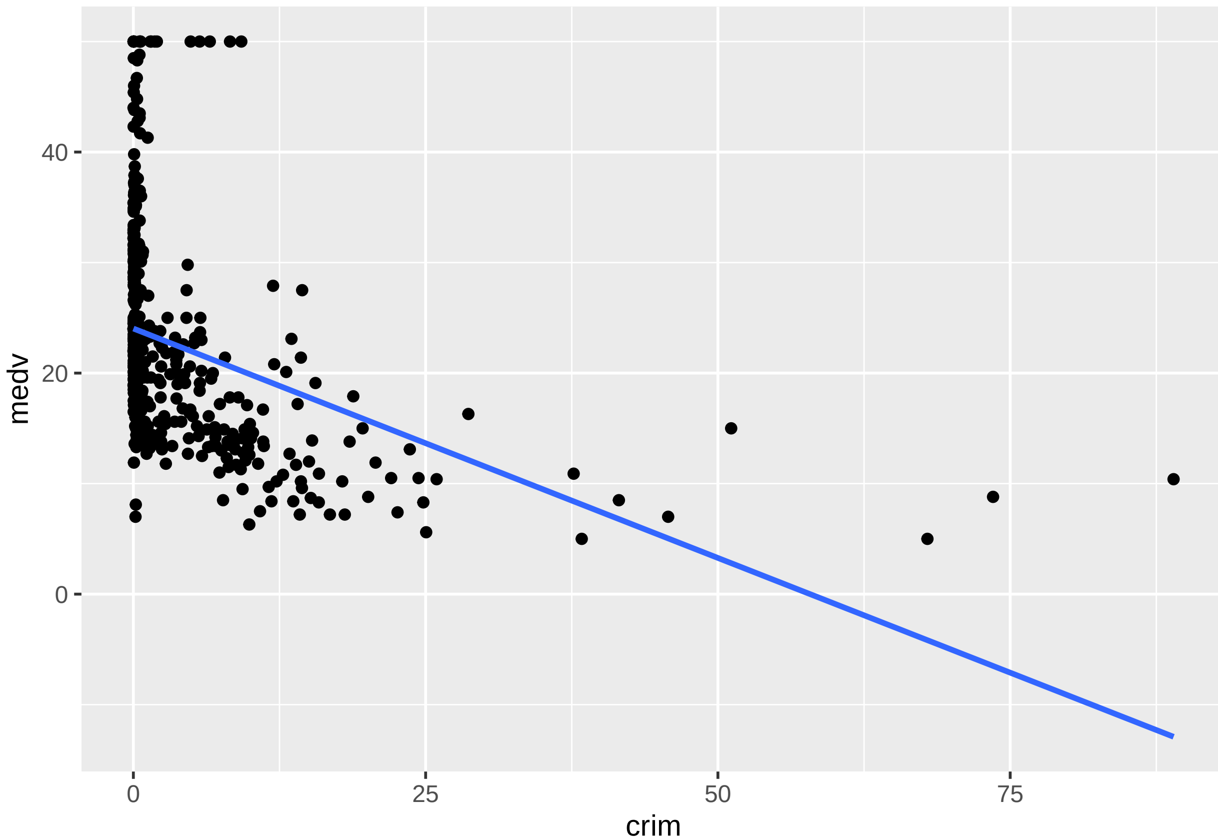
```
## `geom_smooth()` using formula 'y ~ x'
```



2. The graphic you just created fits a best line to a cloud of points. Copy and modify the code to produce a plot where ** crim ** is the x variable instead of ** rm**.

```
ggplot(data=Boston) + aes(x=crim, y=medv) + geom_point() +
 geom_smooth(method="lm", se=FALSE)
```

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




```
# as we can see from the plot, the crime rate is mostly between 0-25%
```

3. Produce a histogram and descriptive statistics for **Boston$crim**. Write a comment describing any anomalies or oddities.

```
summary(Boston$crim)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00632  0.08204  0.25651  3.61352  3.67708 88.97620
```

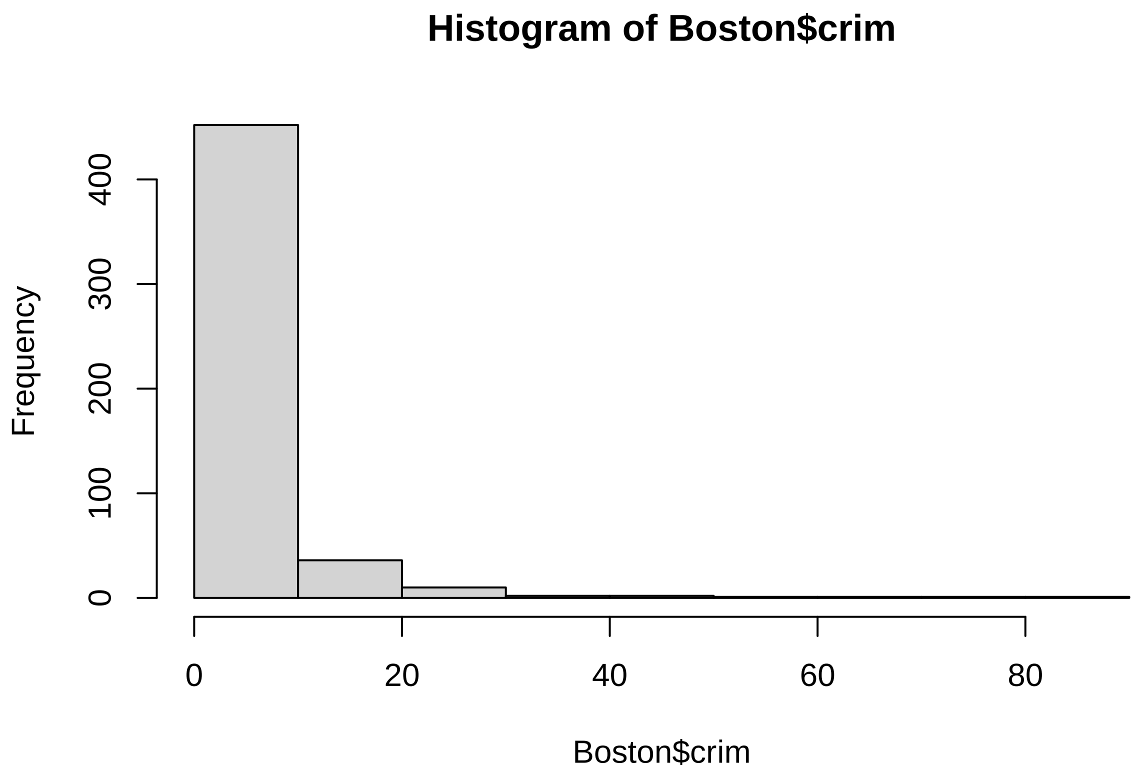
```
sd(Boston$crim) # standart deviation
```

```
## [1] 8.601545
```

```
var(Boston$crim) # variance
```

```
## [1] 73.98658
```

```
hist(Boston$crim)
```




```
# crime rate is at 0-25% where it occurs mostly
```

4. Produce a linear model, using the **lm( )** function where **crim** predicts **medv**. Remember that in R s formula language, the **outcome variable** comes first and is separated from the predictors by a **tilde**, like this: `medv ~ crim` Try to get in the habit of storing the output object that is produced by lm and other analysis procedures. For example, I often use **lmOut <- lm(. . .)**

```
lmOut <- lm(medv ~ crim, data = Boston)
summary(lmOut)
```

```
##
## Call:
## lm(formula = medv ~ crim, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## crim        -0.41519    0.04389   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.484 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

5. Run a **multiple regression** where you use **rm**, **crim**, and **dis** (distance to Boston employment centers). You will use all three predictors in one model with this formula: `medv ~ crim + rm + dis` Now run

three separate models for each independent variable separate.

```
lmOut2 <- lm(medv ~ crim + rm + dis, data = Boston)
summary(lmOut2)
```

```
##
## Call:
## lm(formula = medv ~ crim + rm + dis, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.247  -2.930  -0.572   2.390  39.072
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -29.45838    2.60010 -11.330  < 2e-16 ***
## crim         -0.25405    0.03532  -7.193 2.32e-12 ***
## rm            8.34257    0.40870  20.413  < 2e-16 ***
## dis           0.12627    0.14382   0.878     0.38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.238 on 502 degrees of freedom
## Multiple R-squared:  0.5427, Adjusted R-squared:  0.5399
## F-statistic: 198.6 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
lmOut3 <- lm(medv ~ crim, data = Boston)
summary(lmOut3)
```

```
##
## Call:
## lm(formula = medv ~ crim, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## crim        -0.41519    0.04389   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.484 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lmOut4 <- lm(medv ~ rm, data = Boston)
summary(lmOut4)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = Boston)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
lmOut5 <- lm(medv ~ dis, data = Boston)
summary(lmOut5)
```

```
##
## Call:
## lm(formula = medv ~ dis, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.016  -5.556  -1.865   2.288  30.377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.3901     0.8174  22.499  < 2e-16 ***
## dis           1.0916     0.1884   5.795 1.21e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.914 on 504 degrees of freedom
## Multiple R-squared:  0.06246,    Adjusted R-squared:  0.0606
## F-statistic: 33.58 on 1 and 504 DF,  p-value: 1.207e-08
```

6. Interpret the results of your analysis in a comment. Make sure to mention the **p-value**, the **adjusted R-squared**, the list of **significant predictors** and the **coefficient** for each significant predictor.

```
# lmOut 3 - adjusted R-squared is 0.54 which means that crime variable
# accounts for 54% of median value of owner-occupied homes.
# Coeffiecients - one unit change in crim, have an impact of -0.4 in median
# value of owner-occupied homes. p-value is small so we can say that
# crim and medv are related with each other since p-value < 0.05
# lmOut 4 - adjusted R-squared is 0.48 which means that rm variable
# accounts for 48% of median value of owner-occupied homes.
# Coeffiecients - one unit change in rm, have an impact of 9.1 in median
# value of owner-occupied homes. p-value is small so we can say that
# rm and medv are related with each other since p-value < 0.05
# lmOut 5 - adjusted R-squared is 0.06 which means that dis variable
# accounts for 6% of median value of owner-occupied homes.
# Coeffiecients - one unit change in dis, have an impact of 1 in median
# value of owner-occupied homes. p-value is small so we can say that
# dis and medv are not related with each other since p-value < 0.05
# dis is not a sifnificant predictor
```

7. Create a one-row **data frame** that contains some plausible values for the predictors. For example, this data frame contains the median values for each predictor: `predDF <- data.frame(crim = 0.26, dis=3.2, rm=6.2)` The numbers used here were selected randomly by looking at min and max data of the variables.

```
predDF <- data.frame(crim = 0.72580, dis=4.0900, rm=6.575)
```

8. Use the **predict( )** command to predict a new value of **medv** from the one-row data frame. If you stored the output of your lm model in **lmOut**, the command would look like this: `predict(lmOut, predDF)`

```
predict(lmOut2, predDF)
```

```
##        1
## 25.72606
```

```
# predict medv for the values provided from 1 row dataframe.
# The predicted median value of owner-occupied homes in $1000s will be round 26000 when the predictors
# values are as shown above.
```