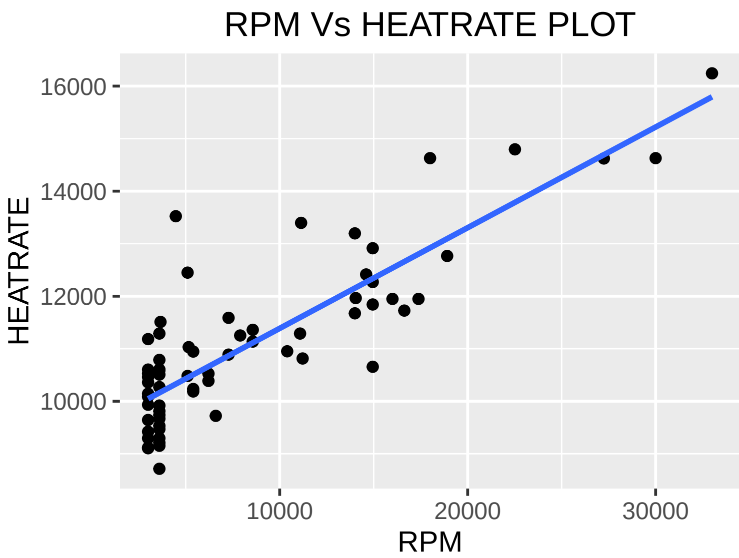
CSC 423 Homework 4

Akhil Kumar Ramasagaram

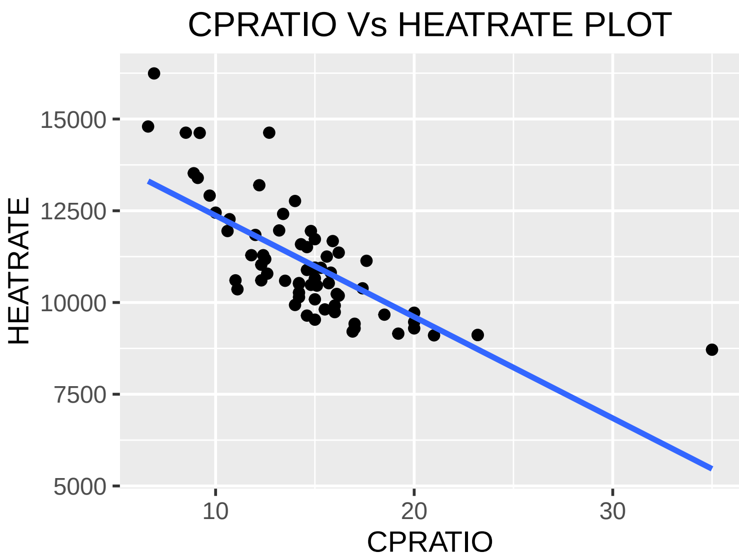
April 30, 2016

5.8 Cooling method for gas turbines.

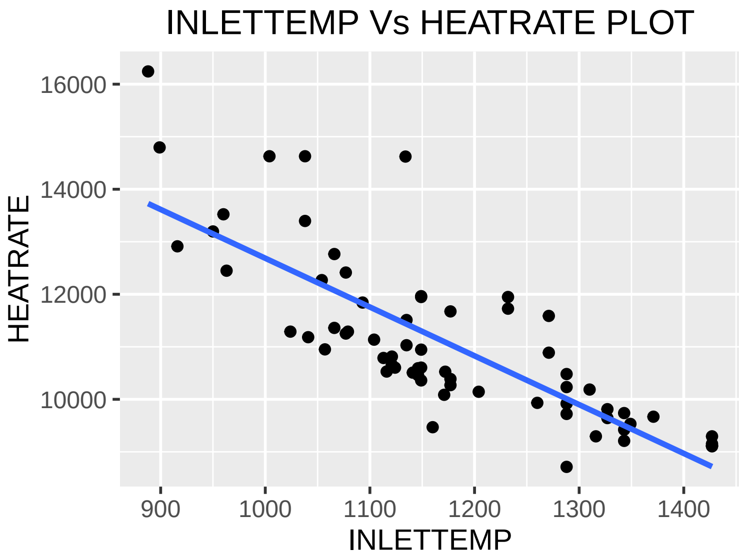
```
library(ggplot2)
library(gridExtra)
load("rdata/GASTURBINE.Rdata")
ggplot(GASTURBINE, aes( x = RPM, y = HEATRATE)) + geom_point() +
  geom_smooth(method = "lm", se = F) + ggtitle("RPM Vs HEATRATE PLOT")
```



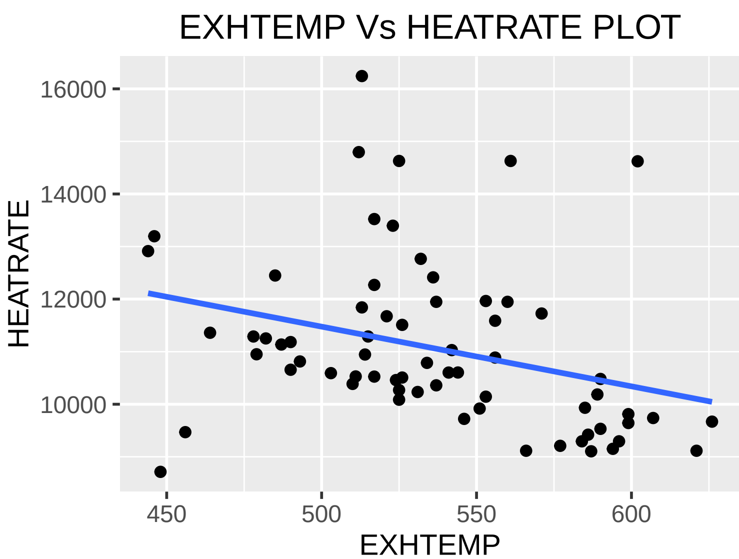

```
ggplot(GASTURBINE, aes( x = CPRATIO, y = HEATRATE)) + geom_point() +
  geom_smooth(method = "lm", se = F) + ggtitle("CPRATIO Vs HEATRATE PLOT")
```



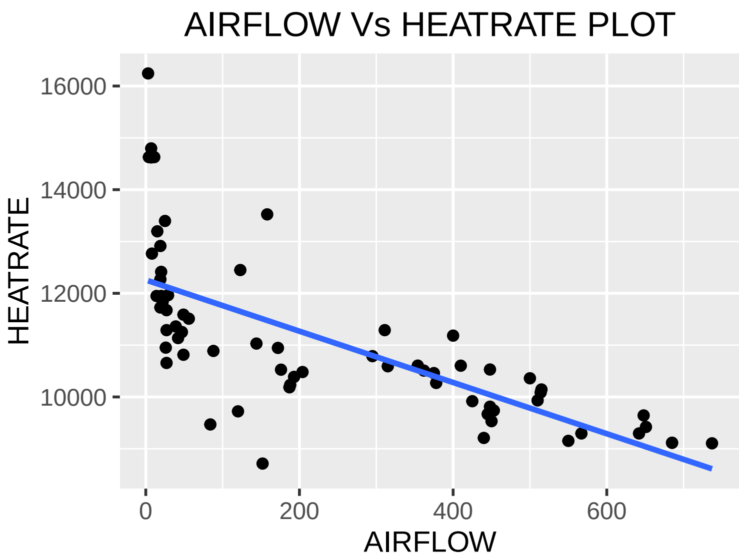
```r
ggplot(GASTURBINE, aes( x = INLETTEMP, y = HEATRATE)) + geom_point() +
  geom_smooth(method = "lm", se = F) + ggtitle("INLETTEMP Vs HEATRATE PLOT")
```




```r
ggplot(GASTURBINE, aes( x = EXHTEMP, y = HEATRATE)) + geom_point() +
  geom_smooth(method = "lm", se = F) + ggtitle("EXHTEMP Vs HEATRATE PLOT")
```




```r
ggplot(GASTURBINE, aes( x = AIRFLOW, y = HEATRATE)) + geom_point() +
  geom_smooth(method = "lm", se = F) + ggtitle("AIRFLOW Vs HEATRATE PLOT")
```



A linear model can be used to predict the heat rate using RPM but for other varialbes the linear model doesn't capture the pattern. I think a polynomial model will be a better option for the remaining variables.

5.19 Cooling method for gas turbines.

a) A second order model using RPM & CPRATIO can be written as. $Heatrate = \beta_0 + \beta_1 * RPM + \beta_2 * CPRATIO + \beta_3 * (RPM)^2 + \beta_4 * (CPRATIO)^2$

b) Below is the fit.

```
full_lm_model <- lm(HEATRATE ~ RPM + CPRATIO, data = GASTURBINE)
reduced_lm_model <- lm(HEATRATE ~ poly(RPM, 2) + poly(CPRATIO, 2), data = GASTURBINE)
summary(reduced_lm_model)
```

```
##
## Call:
## lm(formula = HEATRATE ~ poly(RPM, 2) + poly(CPRATIO, 2), data = GASTURBINE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1126.51  -293.22   -42.15   291.84  1932.16
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        11066.43      68.64 161.220  < 2e-16 ***
## poly(RPM, 2)1       6941.98     711.56   9.756 3.84e-14 ***
## poly(RPM, 2)2       -744.40     582.93  -1.277    0.206
## poly(CPRATIO, 2)1 -6134.29     662.54  -9.259 2.68e-13 ***
## poly(CPRATIO, 2)2  2776.16     638.09   4.351 5.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 561.9 on 62 degrees of freedom
## Multiple R-squared:  0.8834, Adjusted R-squared:  0.8759
## F-statistic: 117.5 on 4 and 62 DF,  p-value: < 2.2e-16
```

c)

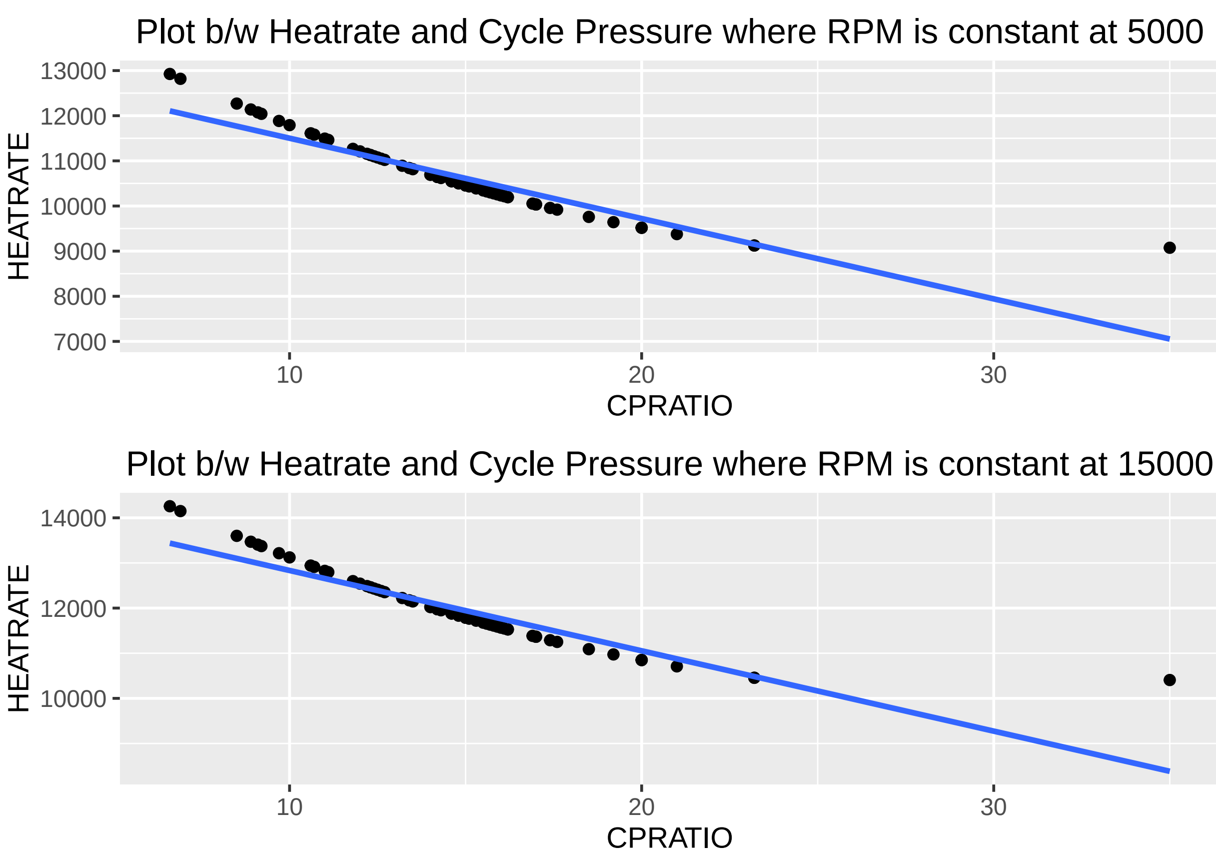
```r
anova(full_lm_model, reduced_lm_model)
```

```
## Analysis of Variance Table
##
## Model 1: HEATRATE ~ RPM + CPRATIO
## Model 2: HEATRATE ~ poly(RPM, 2) + poly(CPRATIO, 2)
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1     64 25553200
## 2     62 19572351  2   5980848 9.4729 0.0002571 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the p-value is significant, we will reject the H_o that $\beta_3 = \beta_4 = 0$ and conclude that the quadratic terms are significant.

d), e) & f)

```r
graph_prediction <- function(x){
  predictions = predict(reduced_lm_model, newdata = data.frame(RPM = x, CPRATIO = GASTURBINE$CPRATIO))
  new_data = data.frame("HEATRATE" = predictions, "CPRATIO" = GASTURBINE$CPRATIO)
  ggplot(new_data, aes(x = CPRATIO, y = HEATRATE)) + geom_point() + geom_smooth(method = "lm", se = F) +
}
grid.arrange(graph_prediction(5000), graph_prediction(15000), ncol = 1)
```



```
predict(reduced_lm_model, newdata = data.frame(RPM = 15000, CPRATIO = GASTURBINE$CPRATIO)) -
predict(reduced_lm_model, newdata = data.frame(RPM = 5000, CPRATIO = GASTURBINE$CPRATIO))
```

```
##        1        2        3        4        5        6        7        8
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##        9       10       11       12       13       14       15       16
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       17       18       19       20       21       22       23       24
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       25       26       27       28       29       30       31       32
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       33       34       35       36       37       38       39       40
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       41       42       43       44       45       46       47       48
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       49       50       51       52       53       54       55       56
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       57       58       59       60       61       62       63       64
## 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631 1331.631
##       65       66       67
## 1331.631 1331.631 1331.631
```

Although the graphs are similar the predictions are off by 1331.631. This can be interpreted for 10,000 increase in RPM the heat rate is expected to increase by 1331.631.

5.20 Tire wear and pressure.

```
load("rdata/TIRES2.Rdata")
```

a) The equation using coding system for pressure is $u = (x - \bar{x})/s_x$

b)

```
TIRES2$U <- (TIRES2$X_PSI - mean(TIRES2$X_PSI))/sd(TIRES2$X_PSI)
TIRES2$U
```

```
## [1] -1.3887301 -0.9258201 -0.4629100  0.0000000  0.4629100  0.9258201
## [7]  1.3887301
```

```
TIRES2$U2 <- TIRES2$U^2
TIRES2$U2
```

```
## [1] 1.9285714 0.8571429 0.2142857 0.0000000 0.2142857 0.8571429 1.9285714
```

c) & d)

```
cor(TIRES2$X_PSI, TIRES2$X_PSI^2)
```

```
## [1] 0.9996558
```

```r
cor(TIRES2$U, TIRES2$U2)
```

```
## [1] 0
```

The variable are highly correlated in part c where as in part d it is 0 as they are standardized.

e)

```r
lm_model <- lm(Y_THOUS ~ U + U2, TIRES2)
summary(lm_model)
```

```
##
## Call:
## lm(formula = Y_THOUS ~ U + U2, data = TIRES2)
##
## Residuals:
##       1       2       3       4       5       6       7
##  1.0714 -1.4286 -0.6429  0.4286  0.7857  0.4286 -0.6429
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.5714     0.6455  58.205 5.22e-07 ***
## U            -0.4629     0.4564  -1.014 0.367854
## U2           -5.3333     0.5693  -9.369 0.000723 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.118 on 4 degrees of freedom
## Multiple R-squared:  0.9569, Adjusted R-squared:  0.9353
## F-statistic:  44.4 on 2 and 4 DF,  p-value: 0.001858
```

The equation can be written as $E(y) = 37.5714 - 0.4629 * U - 5.333 * U^2$. So for a unit increase in U mileage decreases by 0.46 units.

5.27 Quality of Bordeaux wine.

lets create a dummy dataset with 100 observations a)

```r
set.seed(48)
wine_data <- data.frame("quality" = round(runif(100,0,10),2),
                        "method" = sample(c("auto","manual"),100,T),
                        "soil" = sample(c("sand","clay","gravel"),100,T))
wine_data <- within(wine_data, soil <- relevel(wine_data$soil, ref = "sand"))
lm_model <- lm(quality ~  method * soil, data = wine_data)
summary(lm_model)
```

```
##
## Call:
## lm(formula = quality ~ method * soil, data = wine_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9160 -1.9917  0.0256  1.8275  5.6033
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.39800    0.70562   6.233 1.29e-08 ***
## methodmanual           -0.28133    0.95542  -0.294  0.76905
## soilclay                2.70800    0.99790   2.714  0.00792 **
## soilgravel             -0.01133    0.89950  -0.013  0.98997
## methodmanual:soilclay  -0.69252    1.39434  -0.497  0.62058
## methodmanual:soilgravel  0.92681    1.32570   0.699  0.48621
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.733 on 94 degrees of freedom
## Multiple R-squared:  0.1379, Adjusted R-squared:  0.09207
## F-statistic: 3.008 on 5 and 94 DF,  p-value: 0.01457
```

b) when a wine is made from grape which are picked automatically and grown on sand soil have a score of 4.39.
c) β_0 is the intercept, β_1 is manual$, β_2 is clay, β_3 is gravel, β_4 is manual&clay interaction & β_5 is manual&soil interaction. Since our base levels are automatic and sand, if the picking method is manual instead of automatic the wine quality is expected to decrease by 0.28 when everything is held at constant. If grapes are grown on a clay soil instead of sand soil we expect an 2.7 increase in wine quality when everything is held at constant.
d) Its 0.28 units lower. This can be concluded from the β_1 estimate and from the below table.

```r
aggregate(quality ~ method + soil,wine_data, mean)
```

```
##   method   soil  quality
## 1   auto   sand 4.398000
## 2 manual   sand 4.116667
## 3   auto   clay 7.106000
## 4 manual   clay 6.132143
## 5   auto gravel 4.386667
## 6 manual gravel 5.032143
```

5.29 Impact of flavor name on consumer choice.

a) $E(y) = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * (x_1 * x_2)$

b) β_0 will be the intercept which is the number of jelly's taken when flavour is ambiguous and cognitive load is high β_1 is the unit increase in jelly's taken when flavor is common and everything is held constant. β_2 is unit increase in jelly's taken when cognitive load is low and everything is held constant.

c) We can find out the impact by just building a model with only flavor and then build another model by introducing it and compare the adjusted R squared, if it changed then there is an impact.

5.35 Lead in fern moss

a)

The equation for first order model can be written as $E(y) = \beta_0 + \beta_1 * ELEVATION + \beta_2 * SLOPE + \beta_3 * (ELEVATION * SLOPE)$

b)

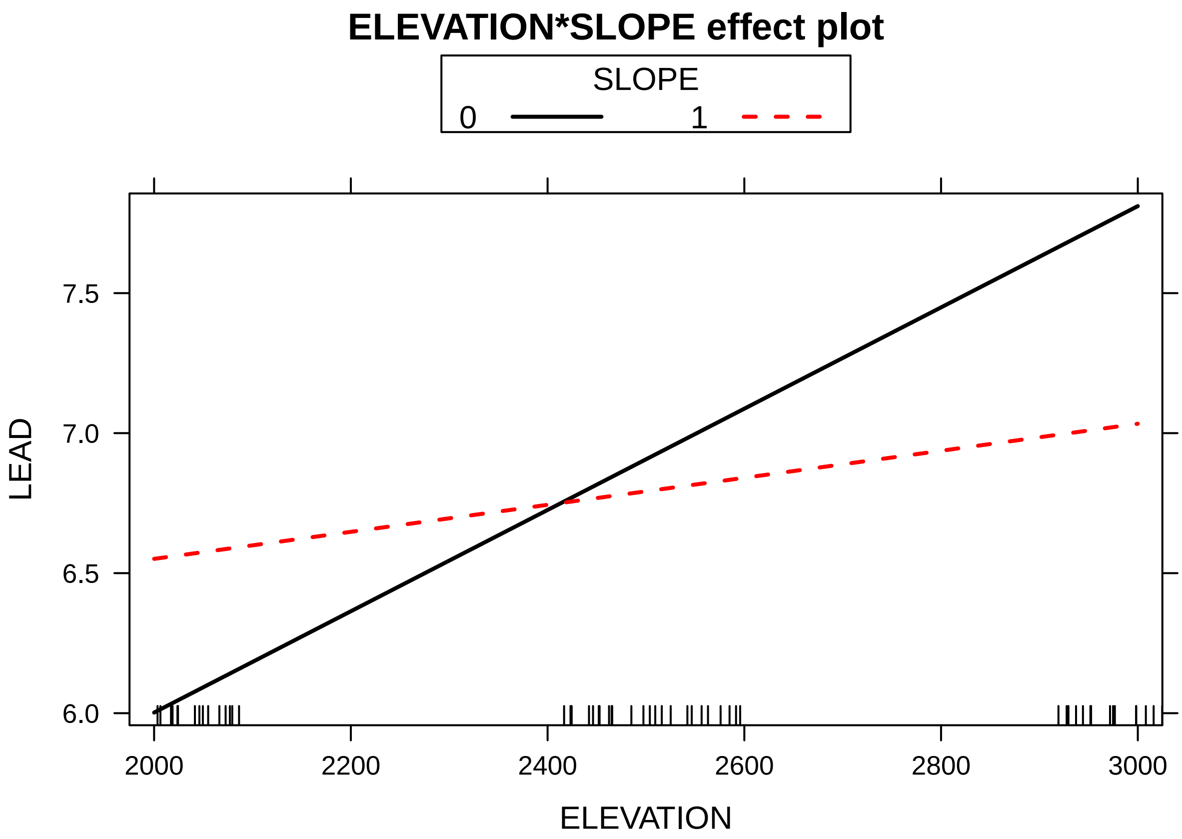
```
library(effects)
```

```
## Warning: package 'effects' was built under R version 3.2.4
```

```
load("rdata/LEADMOSS.Rdata")
lm_model <- lm(LEAD ~ ELEVATION*SLOPE, data = LEADMOSS)
summary(lm_model)
```

```
##
## Call:
## lm(formula = LEAD ~ ELEVATION * SLOPE, data = LEADMOSS)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8108 -2.8437 -1.2154  0.4023 22.3417
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.384866   5.393140   0.442    0.660
## ELEVATION         0.001809   0.002141   0.845    0.401
## SLOPE             3.201458   7.669876   0.417    0.678
## ELEVATION:SLOPE -0.001326   0.003028  -0.438    0.663
##
## Residual standard error: 5.132 on 66 degrees of freedom
## Multiple R-squared:  0.01151,    Adjusted R-squared:  -0.03342
## F-statistic: 0.2562 on 3 and 66 DF,  p-value: 0.8567
```

```
mean = mean(LEADMOSS$SLOPE)
sd = sd(LEADMOSS$SLOPE)
plot(effect("ELEVATION:SLOPE", lm_model,, list(SLOPE=c(0,1))), multiline=TRUE)
```



c) For every one foot increase in elevation the lead level increase by 0.001809.

d) As our p-value is 0.85 which is very high, at $\alpha = 0.1$ we conclude that the above model is not significant to predict the lead level

e) $E(y) = \beta_0 + \beta_1 * elevation + \beta_2 * elevation^2 + \beta_3 * slope + \beta_4 * slope^2$