Exam 2 - Question 3

Adam McQuistan

Tuesday, April 05, 2016

Problem 3 - Do problem 6.18 on page 252.

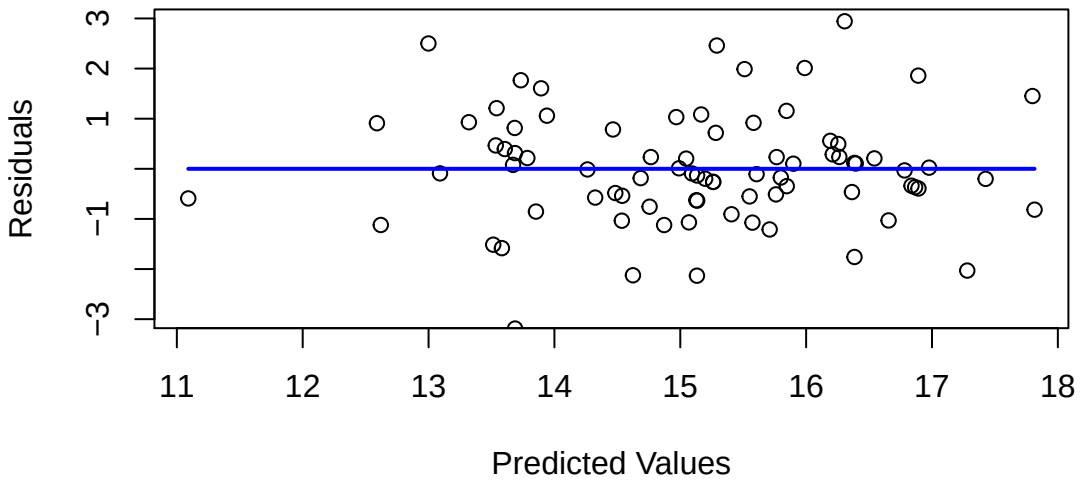
- Only do parts (e-g).
- For part (g) do either Levene or Pagan test

Part E. Plot Residuals vs Predicted Values, Vs each explanatory variable, vs each two factor ineraction term on separate plots. Prepare a normal probability plot of the residuals. Analyze and state your findings.

```
df <- read.csv("data/6.18.csv")
result <- lm(Rental ~ Age + Expense + Vacancy + Footage, data=df)
result_smry <- summary(result)
df_model <- result$model[, 1:5]
df_model$Residuals <- result_smry$residuals
df_model$PredictedVals <- result$fitted.values
df_model$AgeExpense <- df_model$Age * df_model$Expense
df_model$AgeVacancy <- df_model$Age * df_model$Vacancy
df_model$AgeFootage <- df_model$Age * df_model$Footage
df_model$ExpenseVacancy <- df_model$Expense * df_model$Vacancy
df_model$ExpenseFootage <- df_model$Expense * df_model$Footage
df_model$VacancyFootage <- df_model$Vacancy * df_model$Footage
```

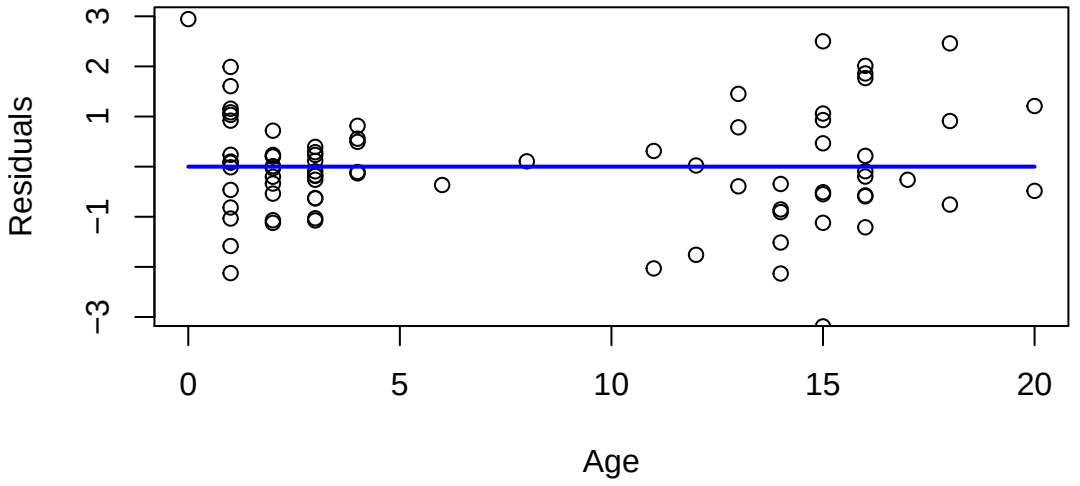
```
with(df_model, {
     plot(x=PredictedVals, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Predicted Values", ylab="Residuals", main="")

     points(c(min(PredictedVals), max(PredictedVals)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



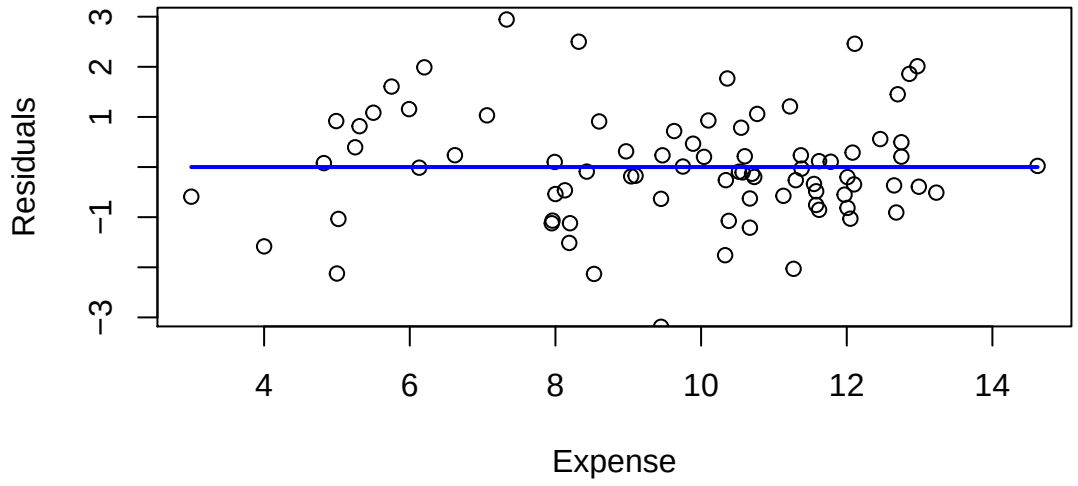

```
with(df_model, {
     plot(x=Age, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Age", ylab="Residuals", main="")

     points(c(min(Age), max(Age)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



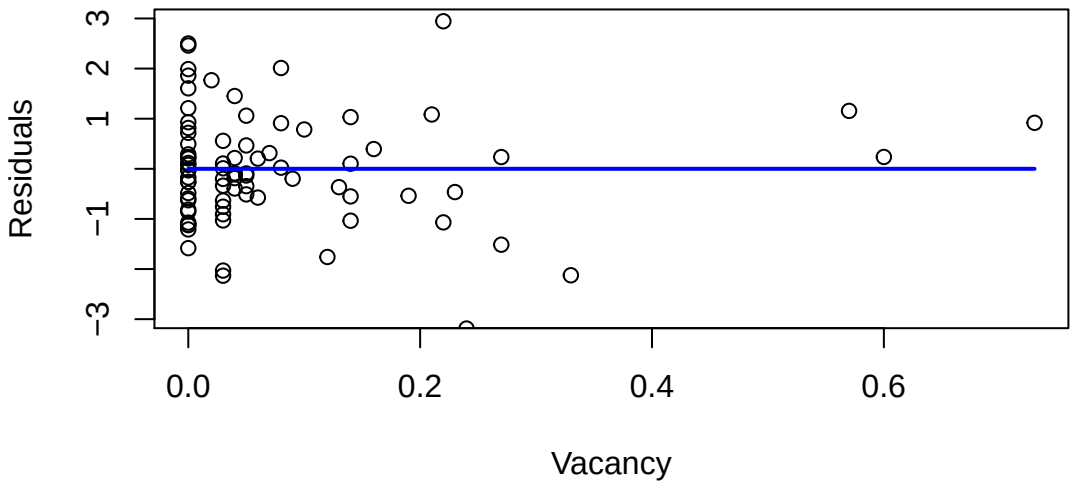
```r
with(df_model, {
     plot(x=Expense, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Expense", ylab="Residuals", main="")

     points(c(min(Expense), max(Expense)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



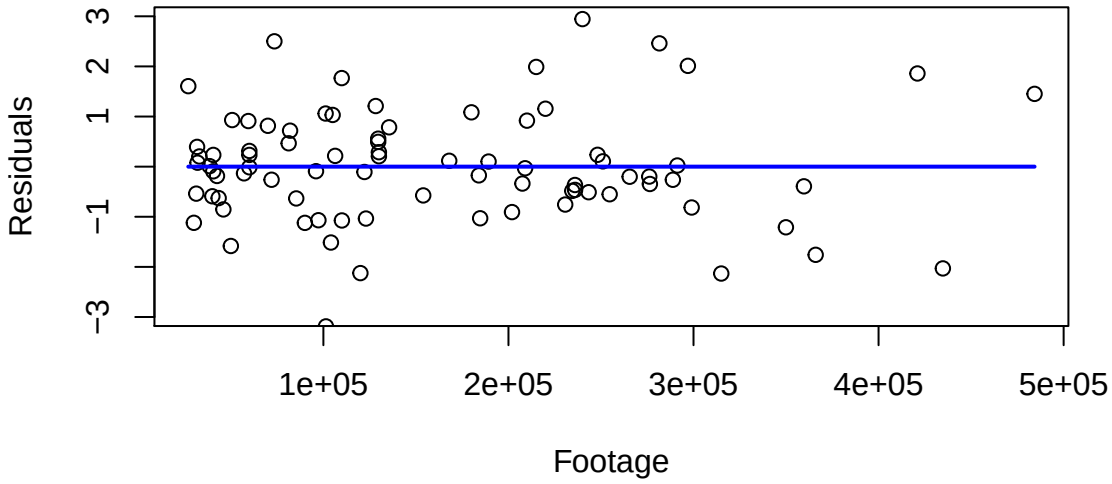

```r
with(df_model, {
     plot(x=Vacancy, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Vacancy", ylab="Residuals", main="")

     points(c(min(Vacancy), max(Vacancy)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



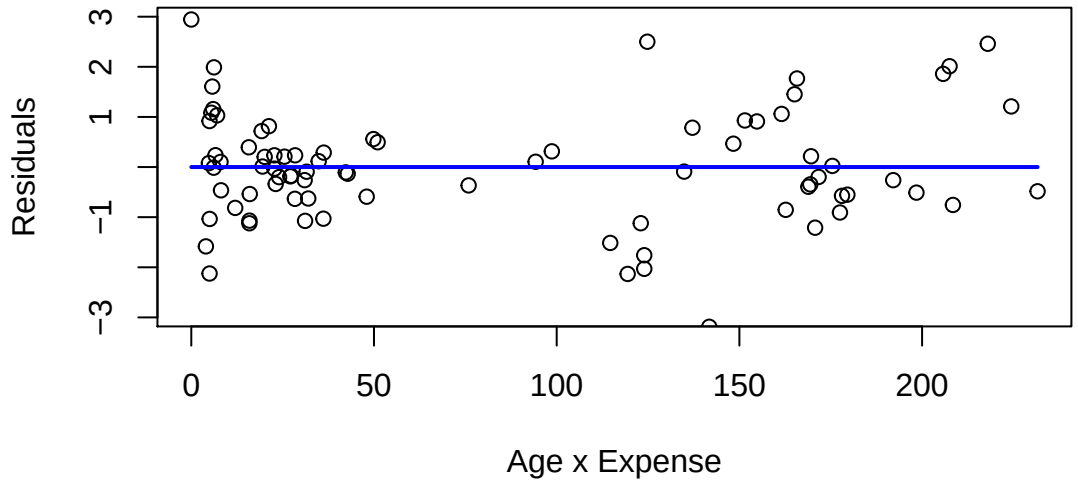

```
with(df_model, {
     plot(x=Footage, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Footage", ylab="Residuals", main="")

     points(c(min(Footage), max(Footage)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



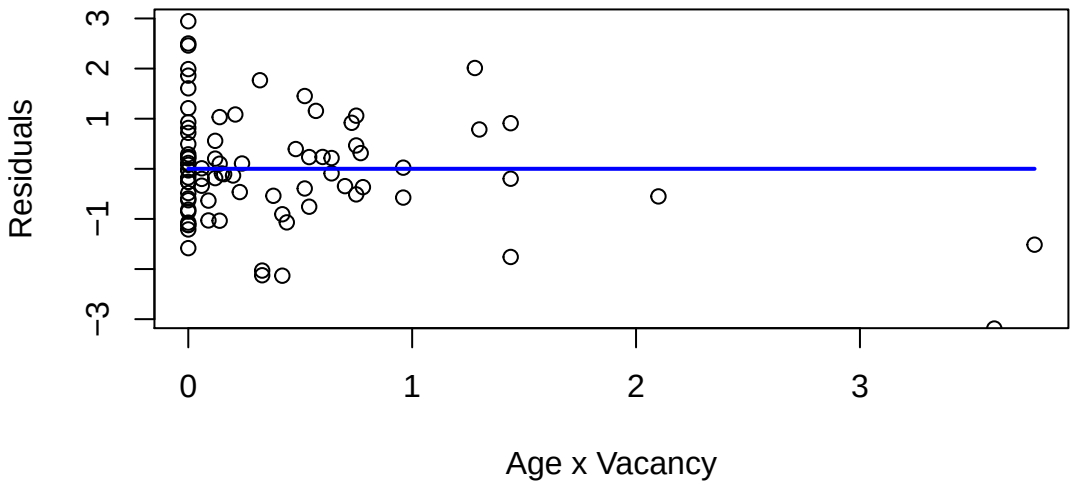
```r
with(df_model, {
     plot(x=AgeExpense, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Age x Expense", ylab="Residuals", main="")

     points(c(min(AgeExpense), max(AgeExpense)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



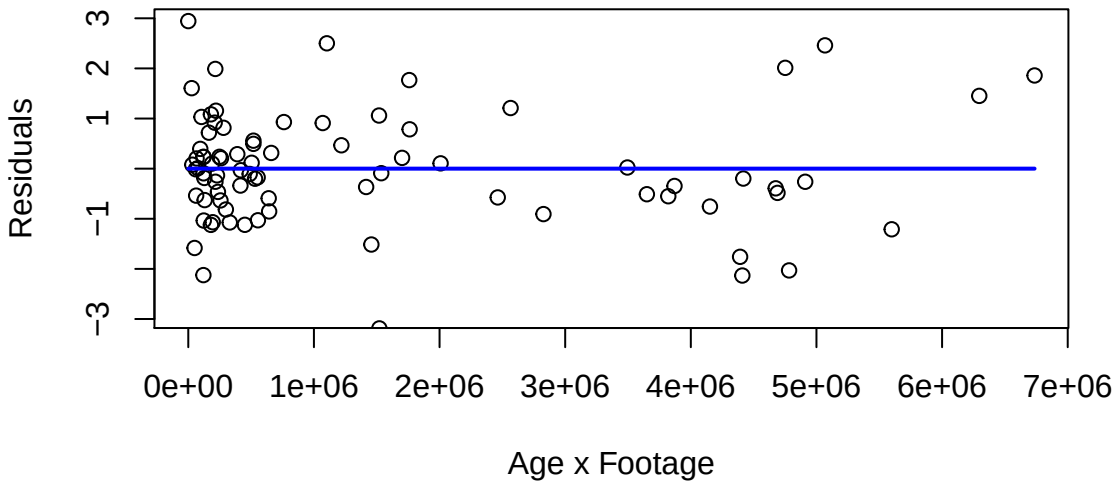

```r
with(df_model, {
     plot(x=AgeVacancy, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Age x Vacancy", ylab="Residuals", main="")

     points(c(min(AgeVacancy), max(AgeVacancy)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



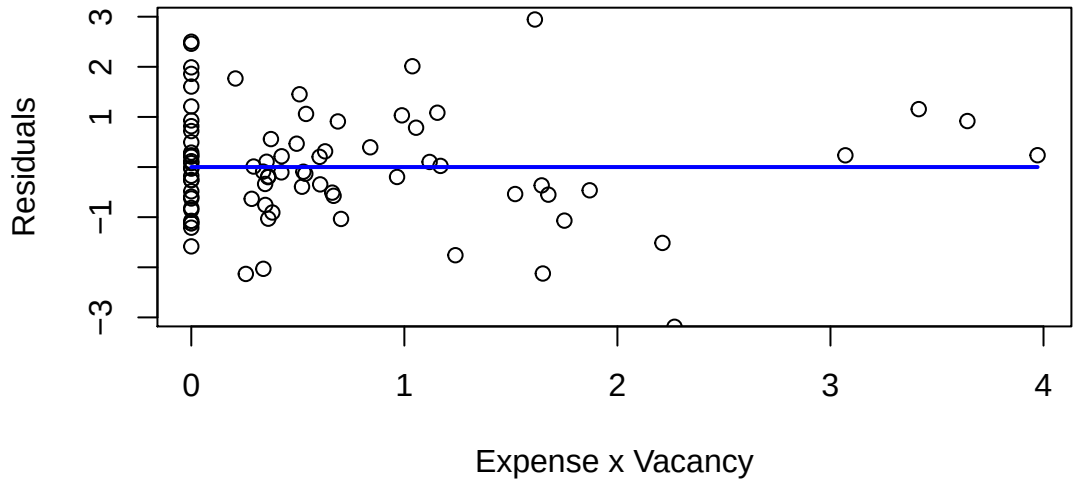

```
with(df_model, {
     plot(x=AgeFootage, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Age x Footage", ylab="Residuals", main="")

     points(c(min(AgeFootage), max(AgeFootage)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



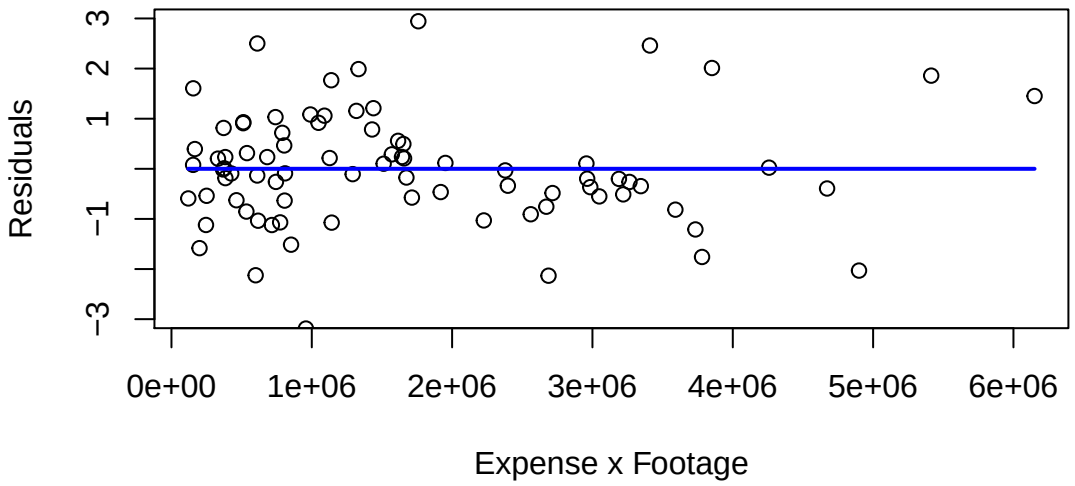
```
with(df_model, {
     plot(x=ExpenseVacancy, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Expense x Vacancy", ylab="Residuals", main="")

     points(c(min(ExpenseVacancy), max(ExpenseVacancy)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



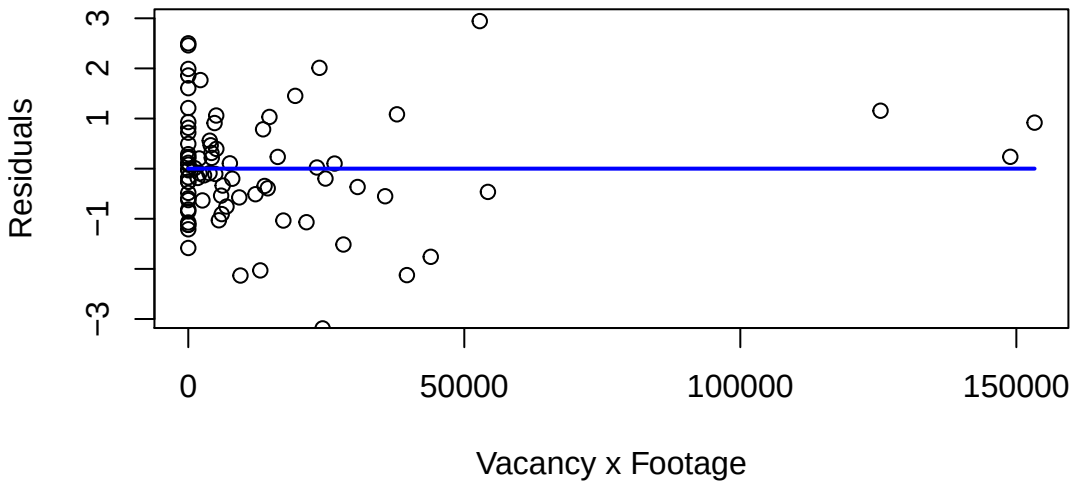

```
with(df_model, {
     plot(x=ExpenseFootage, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Expense x Footage", ylab="Residuals", main="")

     points(c(min(ExpenseFootage), max(ExpenseFootage)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```

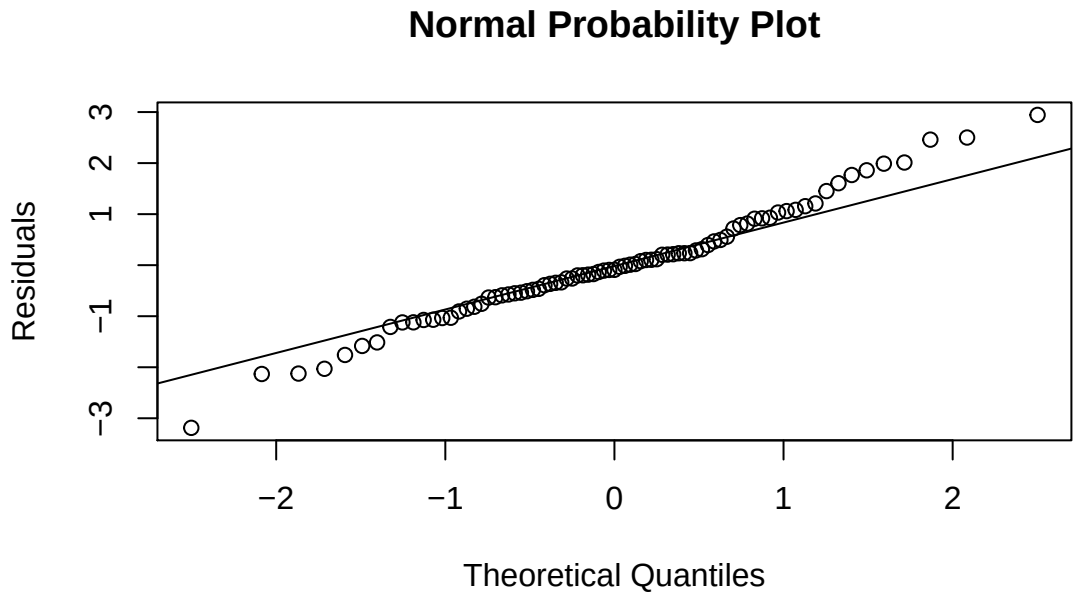



```
with(df_model, {
     plot(x=VacancyFootage, y=Residuals,
          ylim=c(-max(Residuals), max(Residuals)),
          xlab="Vacancy x Footage", ylab="Residuals", main="")

     points(c(min(VacancyFootage), max(VacancyFootage)),
            c(0,0), type="l", lwd="2", col="blue")
     })
```



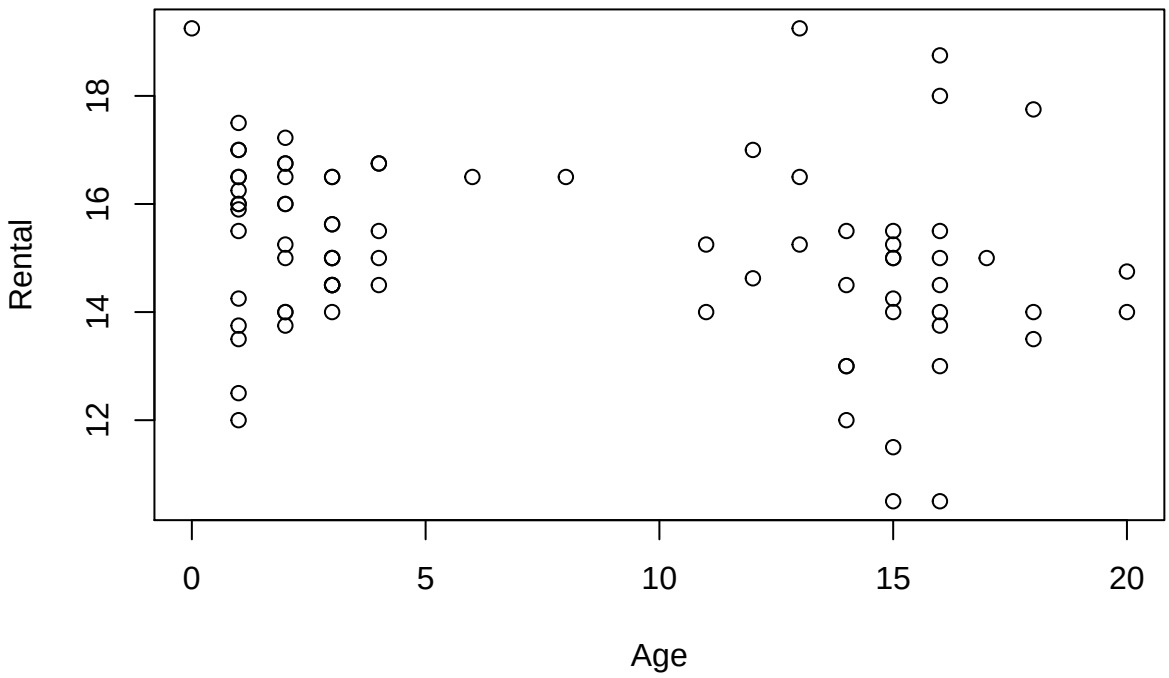
```
qqnorm(result$residuals, ylab="Residuals", main="Normal Probability Plot")
qqline(result$residuals)
```



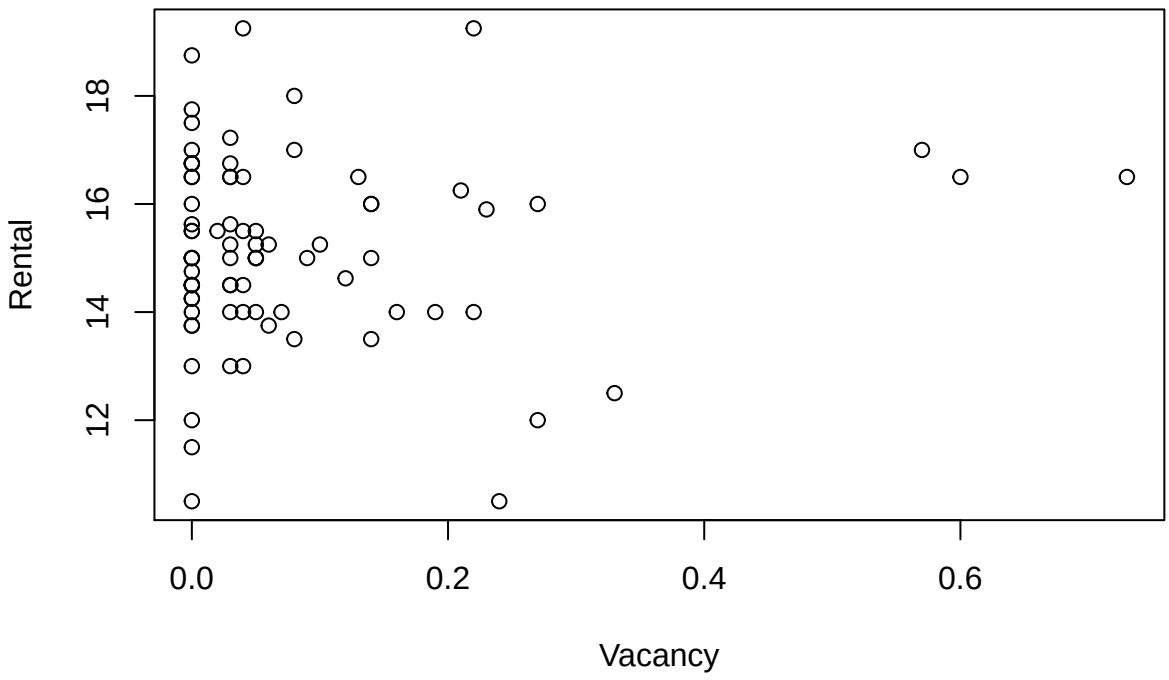
As a whole the model appears to have constant variance of the error terms but, individually some of the predictors do not have constant variance across all their values (the domain of predictor values). For example, Age and Vacancy appear to have a non-constant variance in their error terms.

Also, it does not appear that the relationship between Age or Vacancy against Rental is linear as shown below.

```
with(df, plot(x=Age, y=Rental, type="p"))
```




```
with(df, plot(x=Vacancy, y=Rental, type="p"))
```



Part F. Can you conduct a formal lack of fit test?

Again, I will have to assess whether there are enough naturally occuring repeating vectors of the combination of explanatory variables.

```
df_model$CommonLevels <- with(df_model,
                              paste(Age,
                                    Expense,
                                    Vacancy,
                                    Footage, sep=""))
```

Total Data Instances vs Unique Instances

```
n <- dim(df_model)[1]
m <- length(unique(df_model$CommonLevels))
txt = paste("Total Instances ", n, "  Unique Instances ", m, "\n", sep="")
cat(txt, sep="")
```

```
## Total Instances 81  Unique Instances 81
```

A smoothing of predictor variables will need to be conducted so that repeat combinations are available for the lack of fit test.

```
smoother <- function(x){
  groups <- quantile(x, probs=seq(0,1,0.2))
  groups <- as.numeric(groups)
  for(i in 1:length(x)){
    lower_idx <- max(which(as.numeric(groups) <= x[i]))
    upper_idx <- lower_idx + 1
    mp <- ifelse(lower_idx == length(groups),
              (groups[lower_idx-1] + groups[lower_idx]) / 2,
              (groups[lower_idx] + groups[upper_idx]) / 2)
    x[i] <- mp
  }

  return(x)
}

df_model$SmoothAge <- smoother(df_model$Age)
df_model$SmoothExpense <- smoother(df_model$Expense)
df_model$SmoothVacancy <- smoother(df_model$Vacancy)
df_model$SmoothFootage <- smoother(df_model$Footage)

df_model$CommonLevels <- with(df_model,
                              paste(SmoothAge,
                                    SmoothExpense,
                                    SmoothVacancy,
                                    SmoothFootage, sep=""))
m <- length(unique(df_model$CommonLevels))
txt = paste("Total Instances ", n, "  Unique Instances ", m, "\n", sep="")
cat(txt, sep="")
```

```
## Total Instances 81  Unique Instances 66
```

This should give enough repeating groups to allow for a lack of fit test.

```
reduced <- lm(Rental ~ SmoothAge + SmoothExpense + SmoothVacancy + SmoothFootage
              , data=df_model)

full <- lm(Rental ~ 0 + factor(SmoothAge) + factor(SmoothExpense) + factor(SmoothVacancy) + factor(Smoo

anova(reduced, full)
```

```
## Analysis of Variance Table
##
## Model 1: Rental ~ SmoothAge + SmoothExpense + SmoothVacancy + SmoothFootage
## Model 2: Rental ~ 0 + factor(SmoothAge) + factor(SmoothExpense) + factor(SmoothVacancy) +
##     factor(SmoothFootage)
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1     76 118.427
## 2     65  95.675 11    22.753 1.4052 0.1921
```

The anova of the reduced verse full model is used to assess the appropriateness (fittness) of the model:

$H_o : E\{Y\} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 x_3 + \beta_4 x_4$ * Concludes that the regression function is linear where p-value > 0.05

$H_a : E\{Y\} \neq \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 x_3 + \beta_4 x_4$ * Concludes that there is a lack of linear fit where p-value <= 0.05

Since p-value > 0.05 we do not reject the null hypothesis, the model appears adequate

Part G. Conduct a levene test for constant variance of the different groups.

```
# order the df_model datastructure lowest to highest
# for the fitted (predicted) values
df_model <- df_model[order(df_model$PredictedVals),]
case40Prediction <- df_model$PredictedVals[40]
df_model$LeveneGrps <- ifelse(df_model$PredictedVals <= case40Prediction, "Low", "High")
library(lawstat)
with(df_model, levene.test(Residuals, as.factor(LeveneGrps), location="mean"))
```

```
##
##  classical Levene's test based on the absolute deviations from the
##  mean ( none not applied because the location is not set to median
##  )
##
## data:  Residuals
## Test Statistic = 0.2438, p-value = 0.6228
```

The Leven test assumes the two population's variances are equal

H_o: no difference in population variances

H_o: there are differences in the populations variances

Since p-value > 0.05 we do not reject the null hypothsis and conclude equal variance of the error terms between the groups.