Course Project 2

Devin Romines and Joseph Audras

May 13, 2019

I. Introduction

Dataset Source: https://www.kaggle.com/lava18/google-play-store-apps (https://www.kaggle.com/lava18/google-play-store-apps)

Description

The Google Play Store Data Set is a data set containing data on over 10,000 apps in the Google Play Store for Android products. The data set looks at things such as rating, number of installations, and genre to produce a plethora of information on each app. For this project, we would like to see if the rating of the app can be accurately predicted by the variables provided in the data set.

The data set was gathered from https://www.kaggle.com/lava18/google-play-store-apps (https://www.kaggle.com/lava18/google-play-store-apps) which was last updated 2 months ago, giving us Version 6, with which we are working. The data was posted by Lavanya Gupta, a software engineer at HSBC Software Development in India. More information on her can be found here https://www.kaggle.com/lava18 (https://www.kaggle.com/lava18). The information in the data set was scraped from the Google Play Store by Lavanya Gupta, in an effort to provide similar information on its apps as is publically available from the Apple App Store so that developers may be more inclined to work in the Android market.

```
library(tidyverse)
library(lubridate)
library(ggplot2)
```

II. Data Dictionary

- App – Factor, 9660 levels – The name of the app.
- Category – Factor, 34 levels – The general type of app, such as Dating, Cooking, or Art and Design
- Rating – Numerical, range from 0-5 – The rating of the app on a scale from 0-5.
- Reviews – Number, very wide range – The number of reviews that the app received.
- Size (Removed in Cleaning) – Factor, 462 levels – The size in megabytes of the app.
- Installs – Number, given as a factor of 10 – The number of installations an app received.
- Type (Removed in Cleaning) – Factor, 4 levels – Whether or not the app is free.
- Price – Number, mostly 0 but some go very high – The Price of the app.
- Content.Rating – Factor, 4 levels – The rating for the app, Everyone, Everyone 10+, Mature 17+, and Teen.
- Genres – Factor, 120 levels – The genre of the app, Action, Action and Adventure, etc.

- Last.Updated – Factor, 1378 levels – The date on which the app was last updated, given in multiple formats.
- Current.Ver (Removed in Cleaning) – Factor, 2834 levels – The current version of the app, 0.0.0.2, 0.0.1, etc.
- Android.Ver (Removed in Cleaning) – Factor, 35 levels – The version of Android phone required to use the app, given in the form, “version and up”

III. Data Cleaning

Here we load the data set.

```
RawData <- read.csv("googleplaystore.csv")
```

This next code removes a problematic row that we found to have severe errors when it was inputed.

```
GooglePlayStore <- RawData[-c(10473), ]
```

Some of the values in the data set for the Rating value, which is the one we are looking at, read as NaN, so we will remove all entries that contain that value.

```
GooglePlayStore <- GooglePlayStore[-grep("NaN", GooglePlayStore$Rating), ]
```

The Reviews variable was initially given as a factor variable. Here we change that to be a numeric as it should be.

```
GooglePlayStore$Reviews <- as.numeric(as.character(GooglePlayStore$Reviews))
```

The size variable was deemed unnecessary for our goals for this project so here we remove it.

```
GooglePlayStore <- GooglePlayStore[, -5]
```

The Installs variable was initially given as a factor, with undesirable formatting. Here we remove the characters in the entries we do not want and then convert the Installs variable to a numeric as desired.

```
GooglePlayStore$Installs <- gsub("\\D", "", GooglePlayStore$Installs)
GooglePlayStore$Installs <- as.numeric(as.character(GooglePlayStore$Installs))
```

The Type variable merely indicates whether or not the app is free, which is also given by the better variable, Price, so here we remove it.

```
GooglePlayStore <- GooglePlayStore[, -6]
```

The Price variable is interesting, here we had to remove the dollar sign as we wanted to convert it to a numeric variable; however, when removing the dollar sign we also noticed that the code also removed the decimal point. To recover it, we simply divided every price by 100. This solved the problem.

```
GooglePlayStore$Price <- gsub("[[:punct:]]", "", GooglePlayStore$Price)
GooglePlayStore$Price <- as.numeric(GooglePlayStore$Price)
GooglePlayStore$Price <- GooglePlayStore$Price/100
```

We determined that the Current.Ver variable was irrelevant to our goals, so here we remove it.

```
GooglePlayStore <- GooglePlayStore[, -10]
```

We also determined that the Android.Ver variable was irrelevant to our goals, so here we remove that as well.

```
GooglePlayStore <- GooglePlayStore[, -10]
```

Here we write the cleaned data set to disk.

```
write.csv(GooglePlayStore,"Prepared_Google_Play_Store_App_Data.csv",row.names=FALSE)
```

Because we intend on making a predictive model, here we split the data into a testing and training data set.

```
set.seed(42)

GooglePlayStoreTemp <- GooglePlayStore %>% mutate(id=row_number())
Train <- GooglePlayStoreTemp %>% sample_frac(0.6)
Test <- GooglePlayStoreTemp %>% anti_join(Train,by="id")
Train$id <- NULL
Test$id <- NULL
write.csv(Test,"Test.csv",row.names = FALSE)
rm(Test,GooglePlayStoreTemp)
```

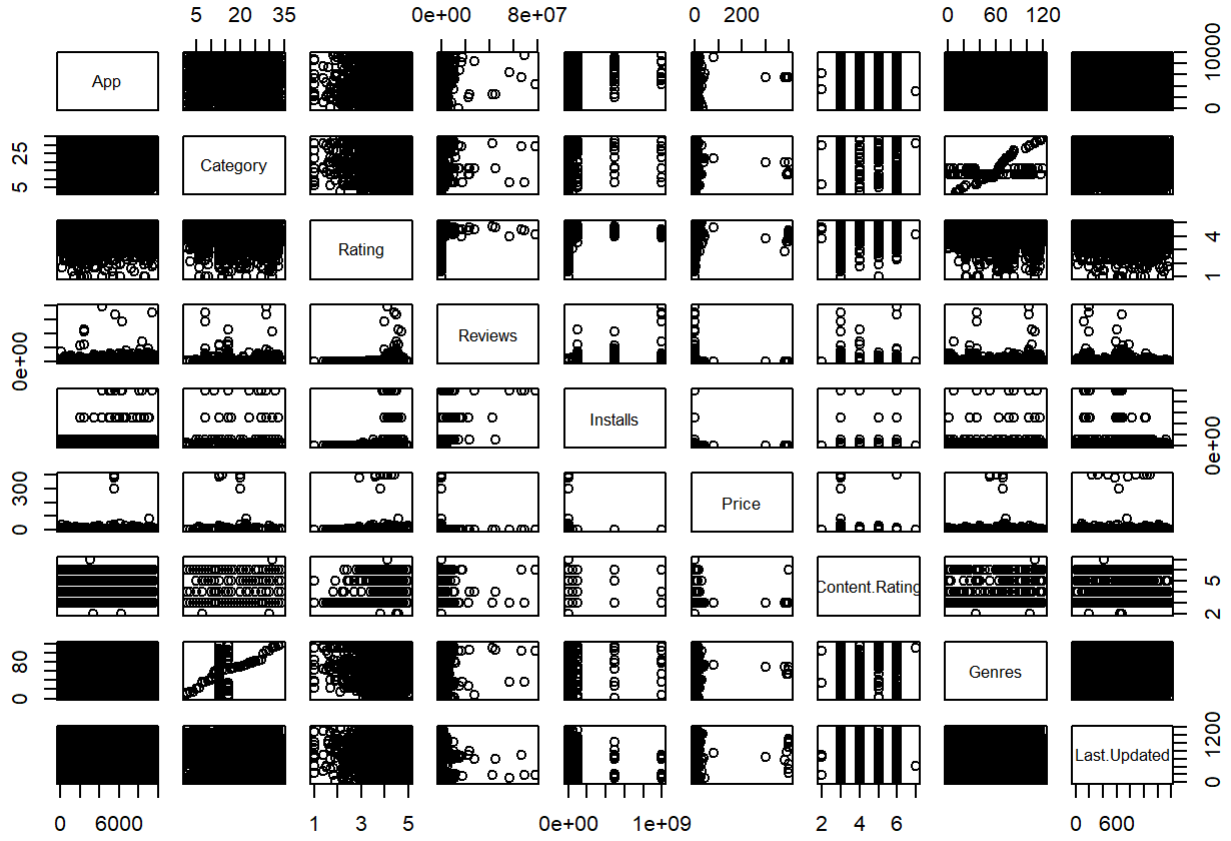
IV. Exploratory Data Analysis

To begin the exploratory data analysis, let’s look at the data set as a whole.

```
summary(Train)
```

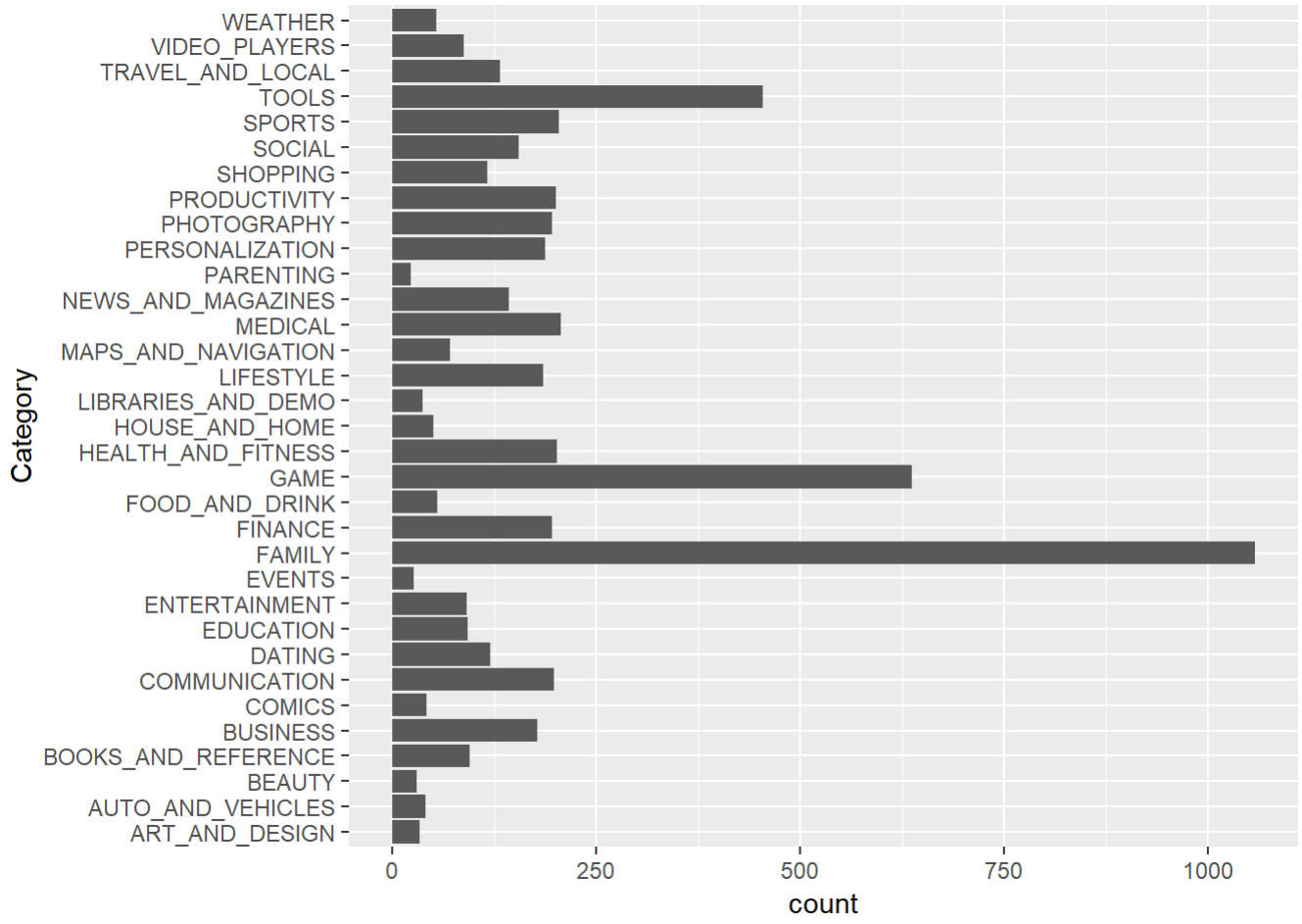
```
##                                                 App
##  Candy Crush Saga                                 :   6
##  ROBLOX                                           :   6
##  Bleacher Report: sports news, scores, & highlights:   5
##  Duolingo: Learn Languages Free                   :   5
##  ESPN                                             :   5
##  8 Ball Pool                                      :   4
##  (Other)                                          :5589
##                Category        Rating         Reviews
##  FAMILY            :1058   Min.   :1.000   Min.   :       1
##  GAME              : 638   1st Qu.:4.000   1st Qu.:     174
##  TOOLS             : 455   Median :4.300   Median :    5417
##  MEDICAL           : 207   Mean   :4.183   Mean   :  503129
##  SPORTS            : 205   3rd Qu.:4.500   3rd Qu.:   77868
##  HEALTH_AND_FITNESS: 203   Max.   :5.000   Max.   :78158306
##  (Other)           :2854
##     Installs             Price                 Content.Rating
##  Min.   :1.000e+00   Min.   :  0.000                 :   0
##  1st Qu.:1.000e+04   1st Qu.:  0.000   Adults only 18+:   3
##  Median :5.000e+05   Median :  0.000   Everyone       :4434
##  Mean   :1.716e+07   Mean   :  1.109   Everyone 10+   : 253
##  3rd Qu.:5.000e+06   3rd Qu.:  0.000   Mature 17+     : 276
##  Max.   :1.000e+09   Max.   :400.000   Teen           : 653
##                                        Unrated        :   1
##            Genres              Last.Updated
##  Tools        : 455   August 1, 2018: 179
##  Entertainment: 336   August 3, 2018: 178
##  Education    : 281   July 31, 2018 : 164
##  Action       : 213   August 2, 2018: 157
##  Sports       : 211   July 30, 2018 : 114
##  Medical      : 207   August 6, 2018:  99
##  (Other)      :3917   (Other)       :4729
```

```
pairs(Train)
```

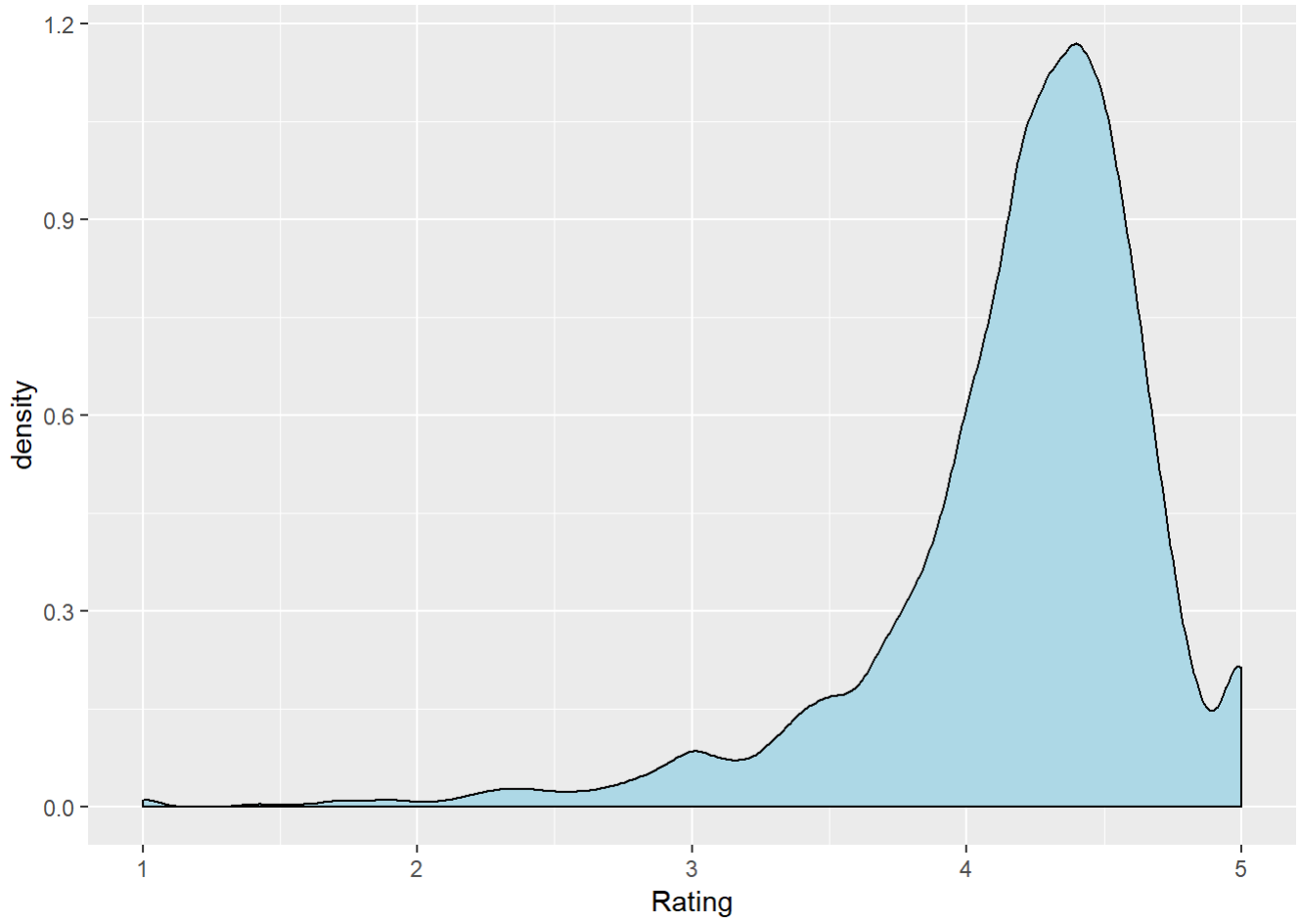


Now, let's look at the individual variables.

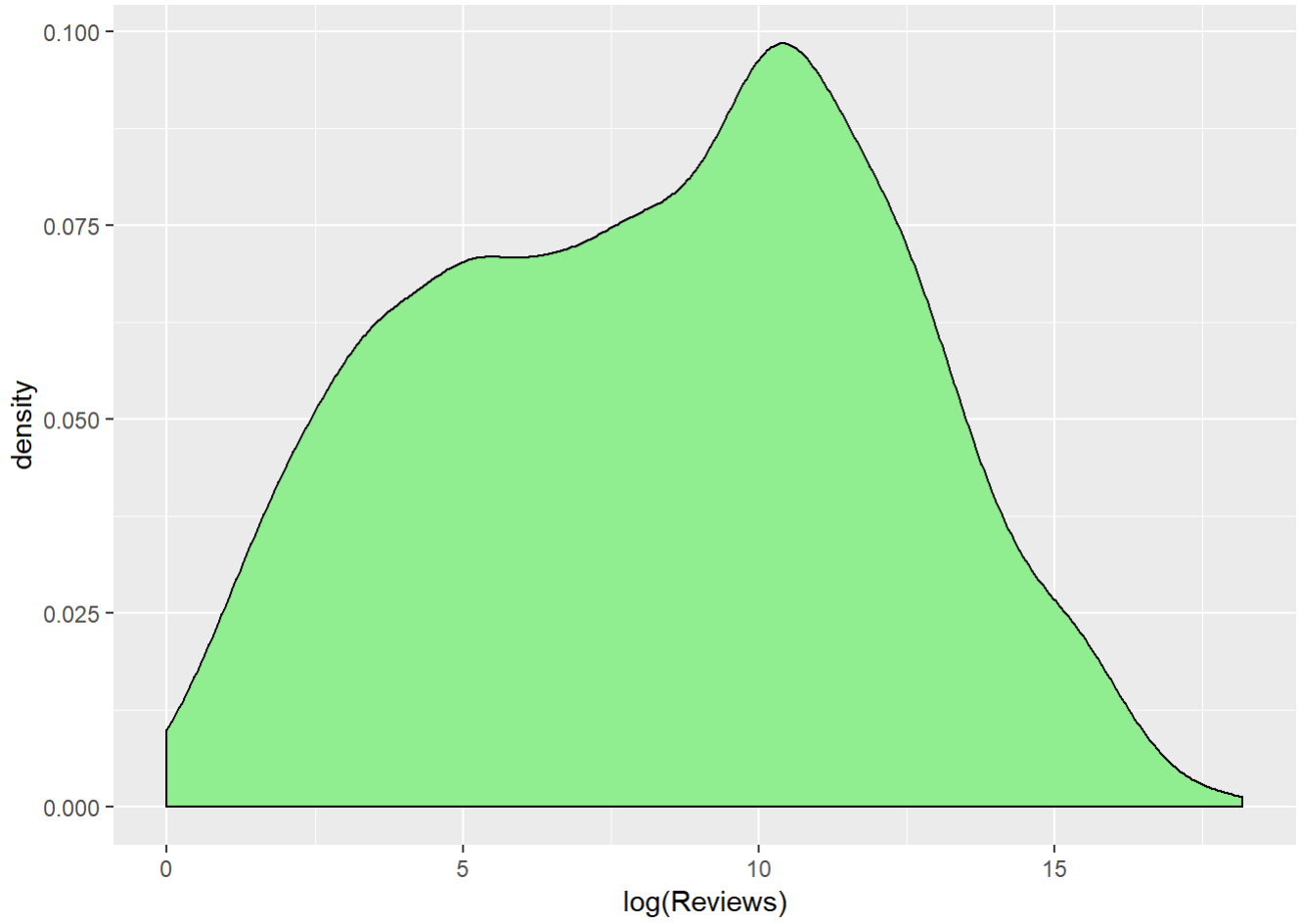
```
ggplot(Train) + geom_bar(aes(x=Category)) + coord_flip()
```



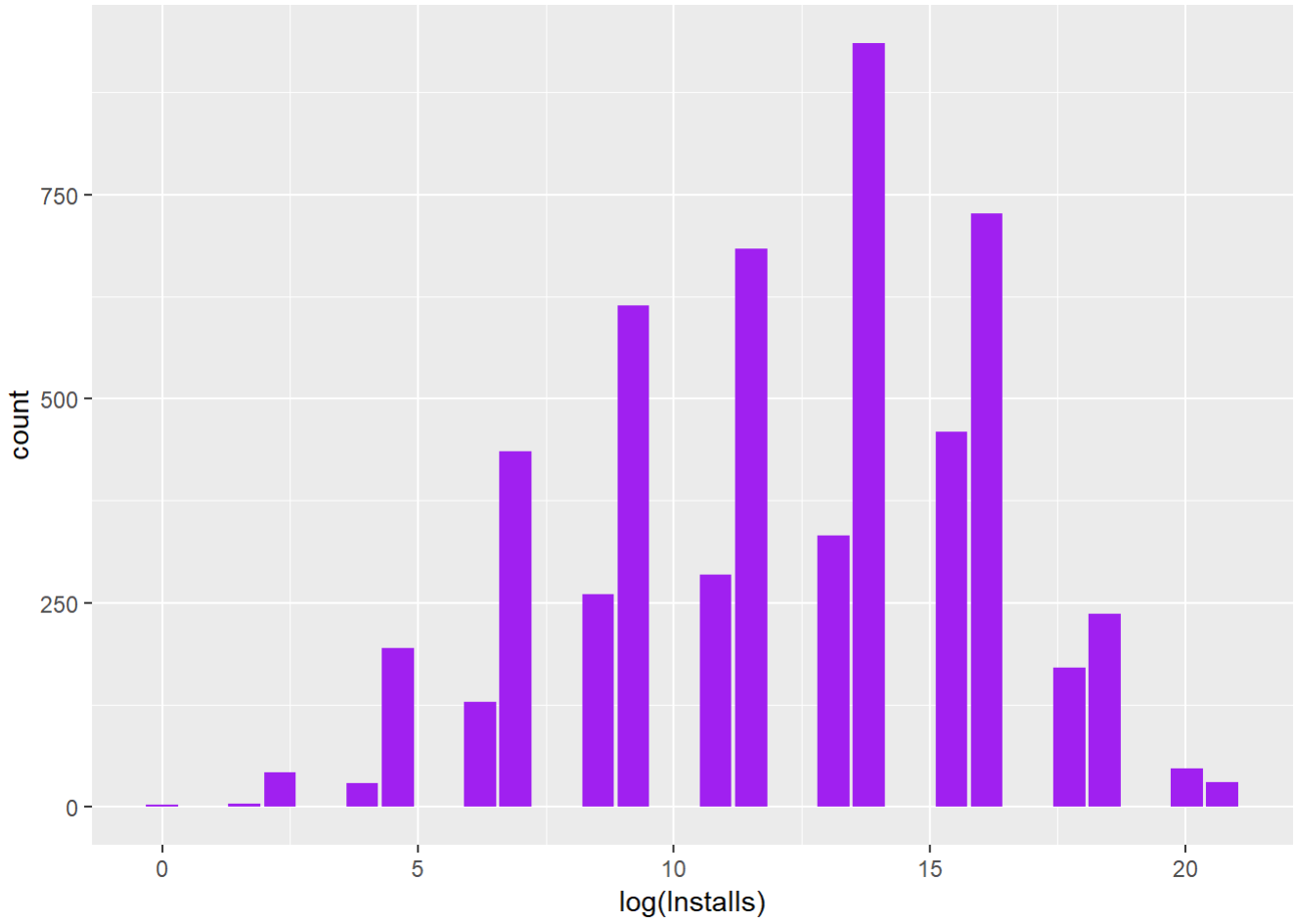

```
ggplot(Train) + geom_density(aes(x=Rating),fill="light blue")
```



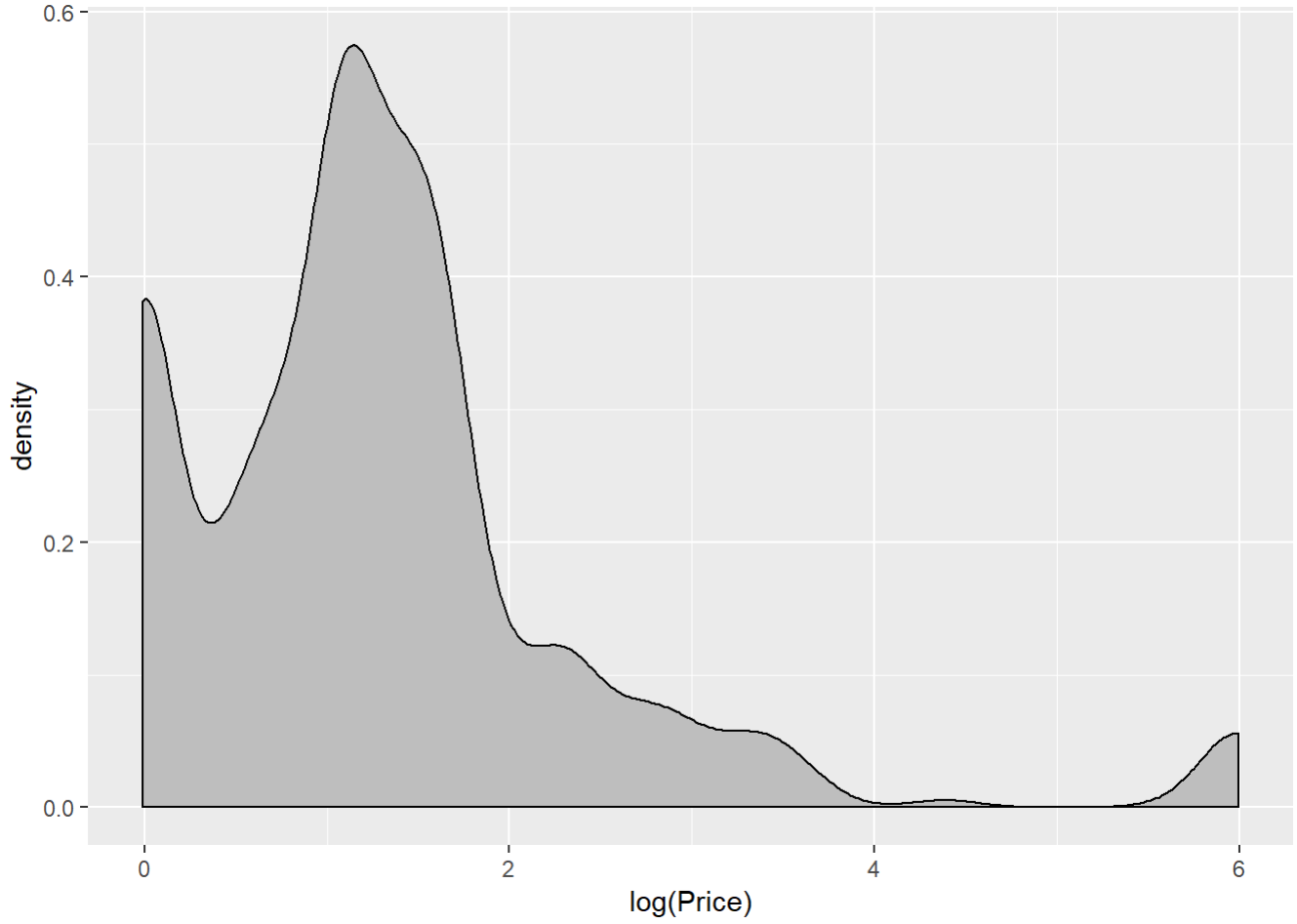

```
ggplot(Train) + geom_density(aes(x=log(Reviews)),fill="light green")
```



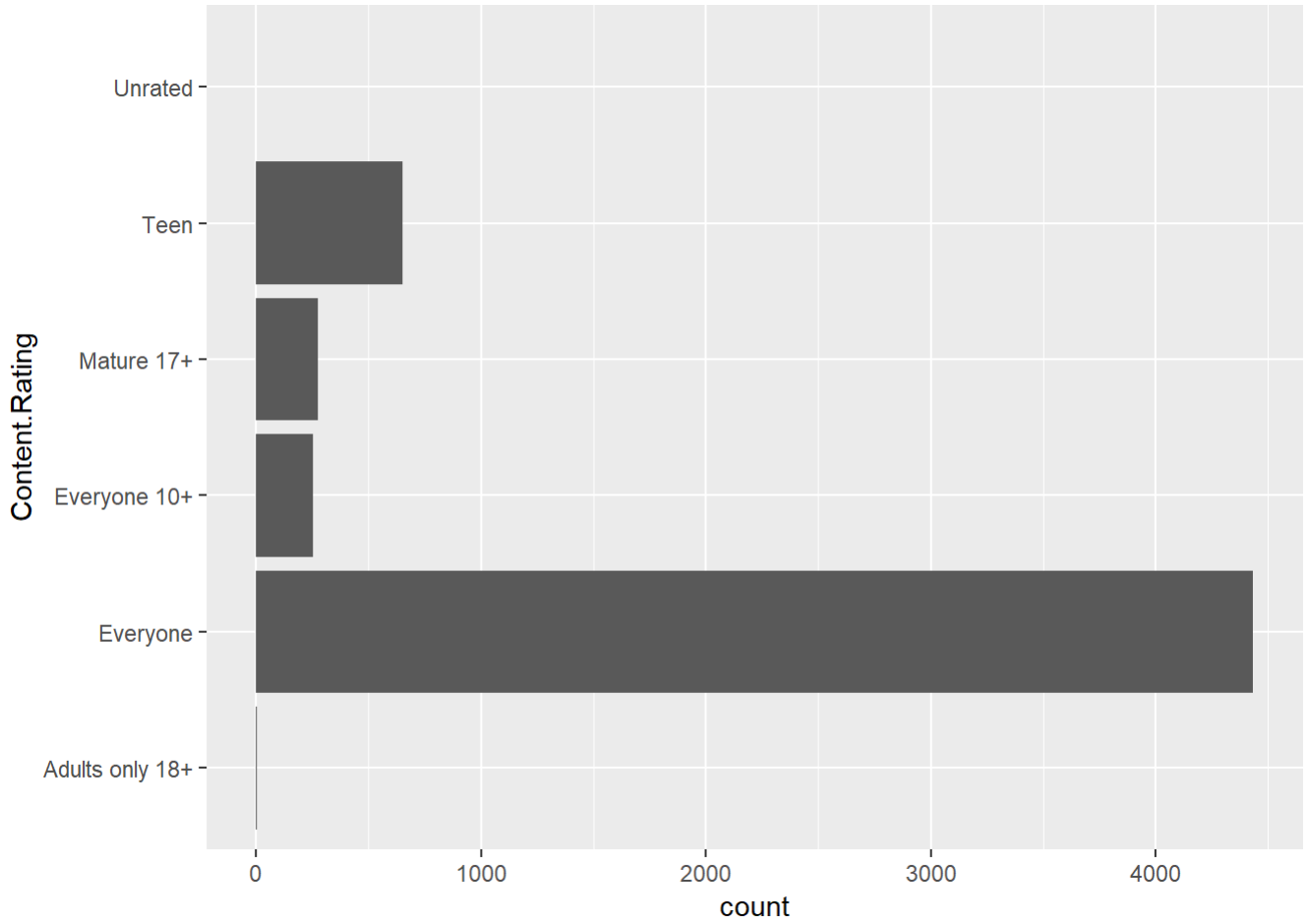

```
ggplot(Train) + geom_bar(aes(x=log(Installs)),fill="purple")
```




```
ggplot(Train) + geom_density(aes(x=log(Price)),fill="gray")
```

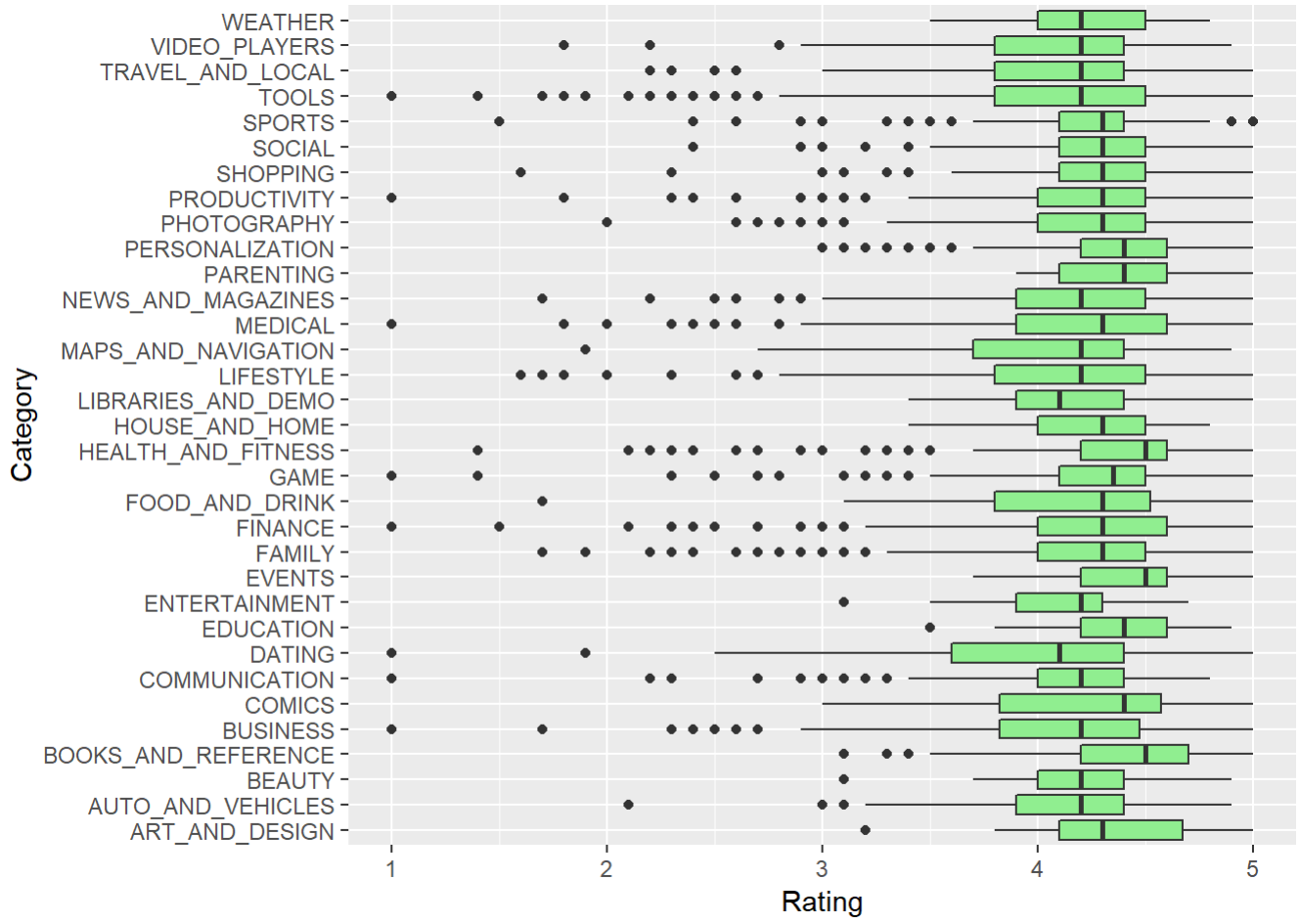



```
ggplot(Train) + geom_bar(aes(x=Content.Rating)) + coord_flip()
```

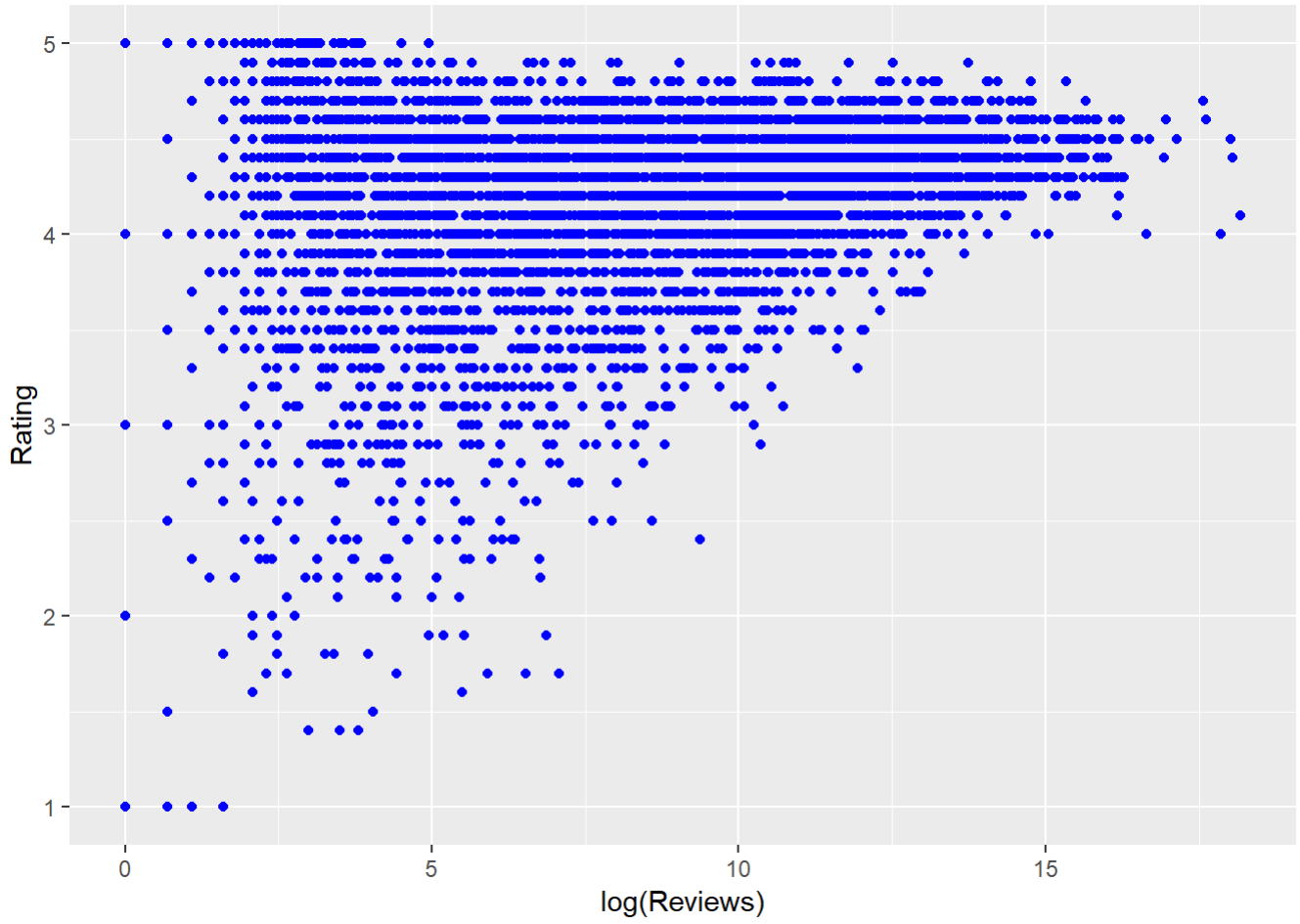


Let’s look at some of the relationships between the Rating variable, our response, and some of the possible predictors.

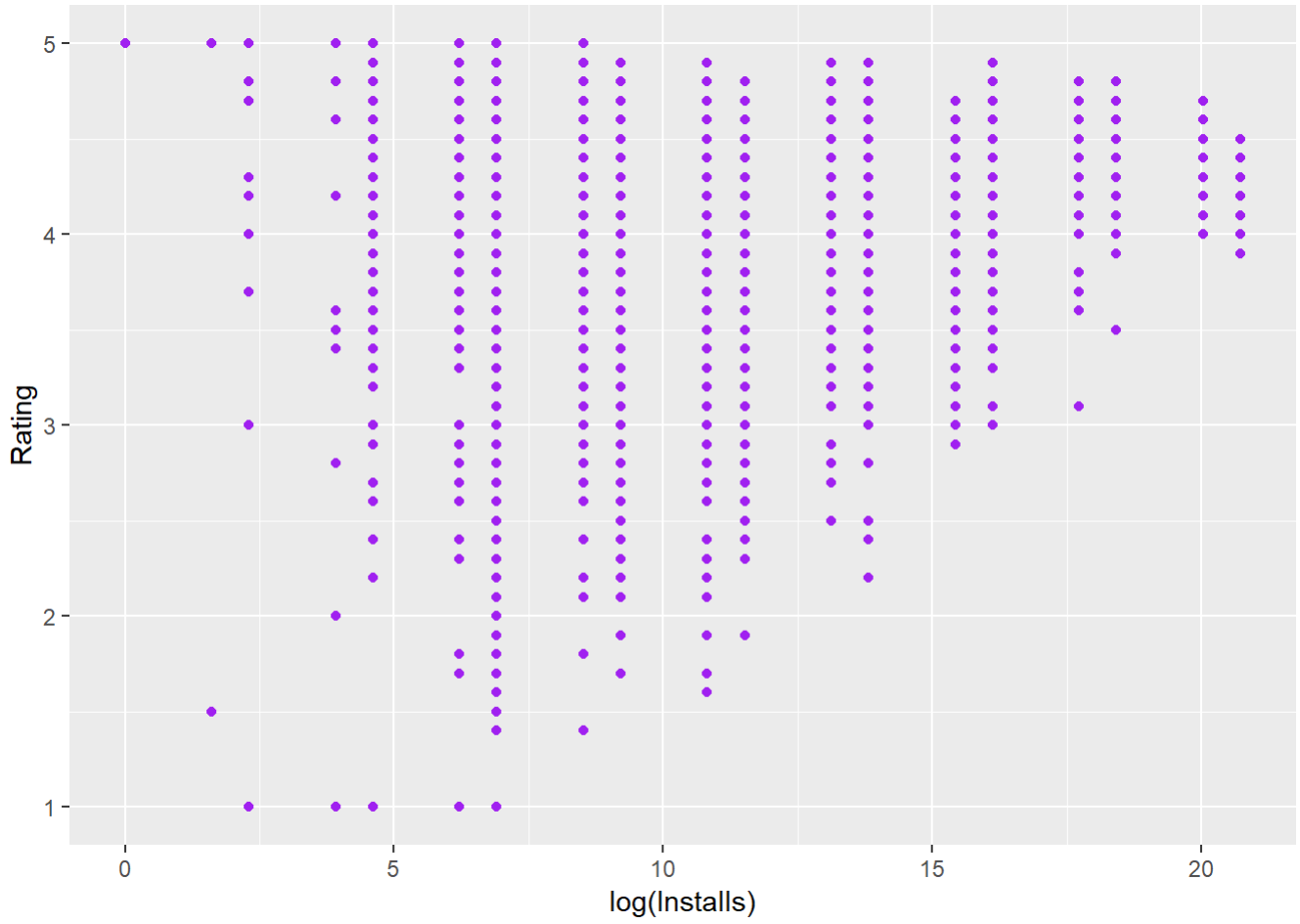
```
ggplot(Train) + geom_boxplot(aes(x=Category,y=Rating), fill="light green") + coord_flip()
```



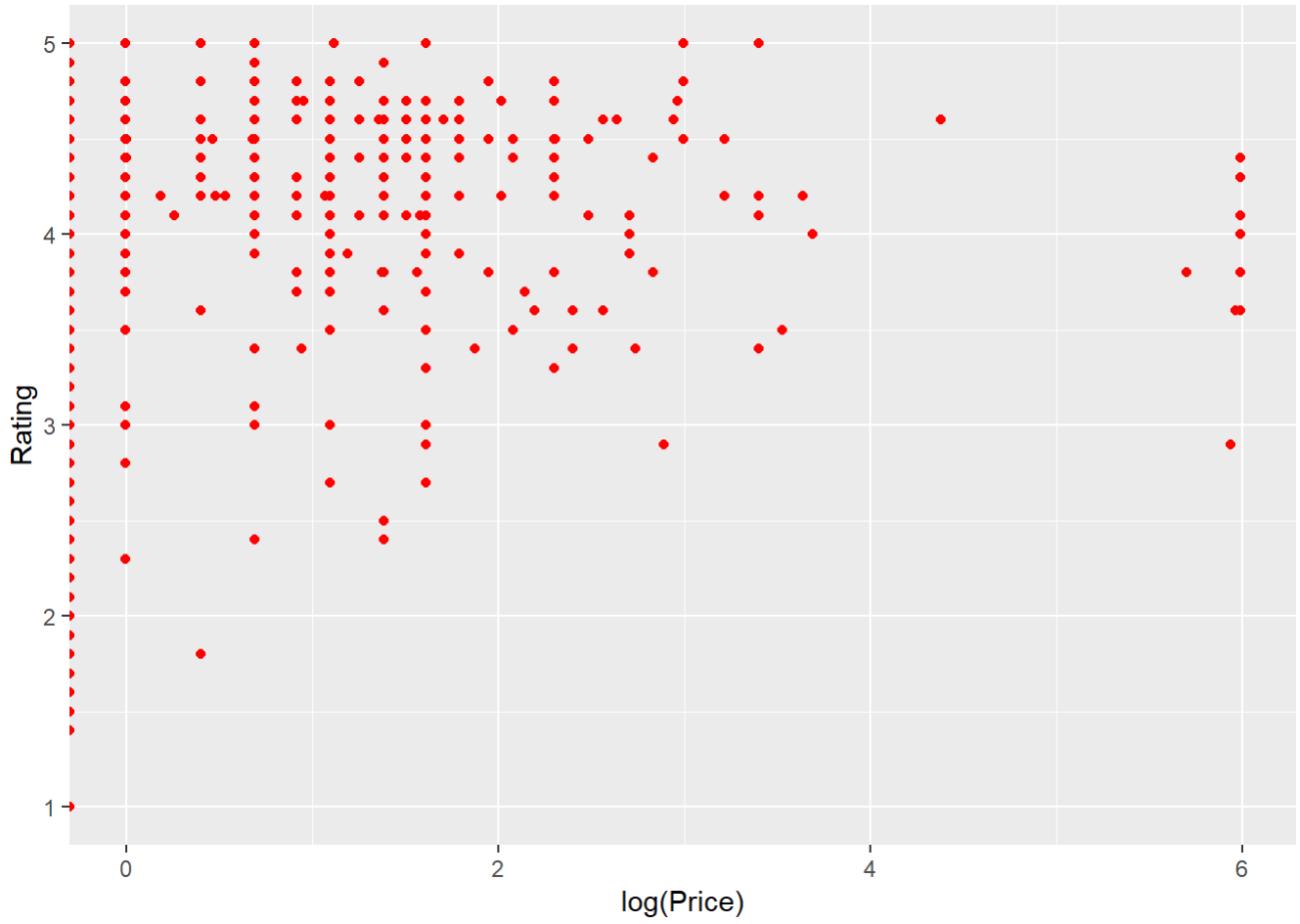

```
ggplot(Train) + geom_point(aes(x=log(Reviews),y=Rating), color="blue")
```



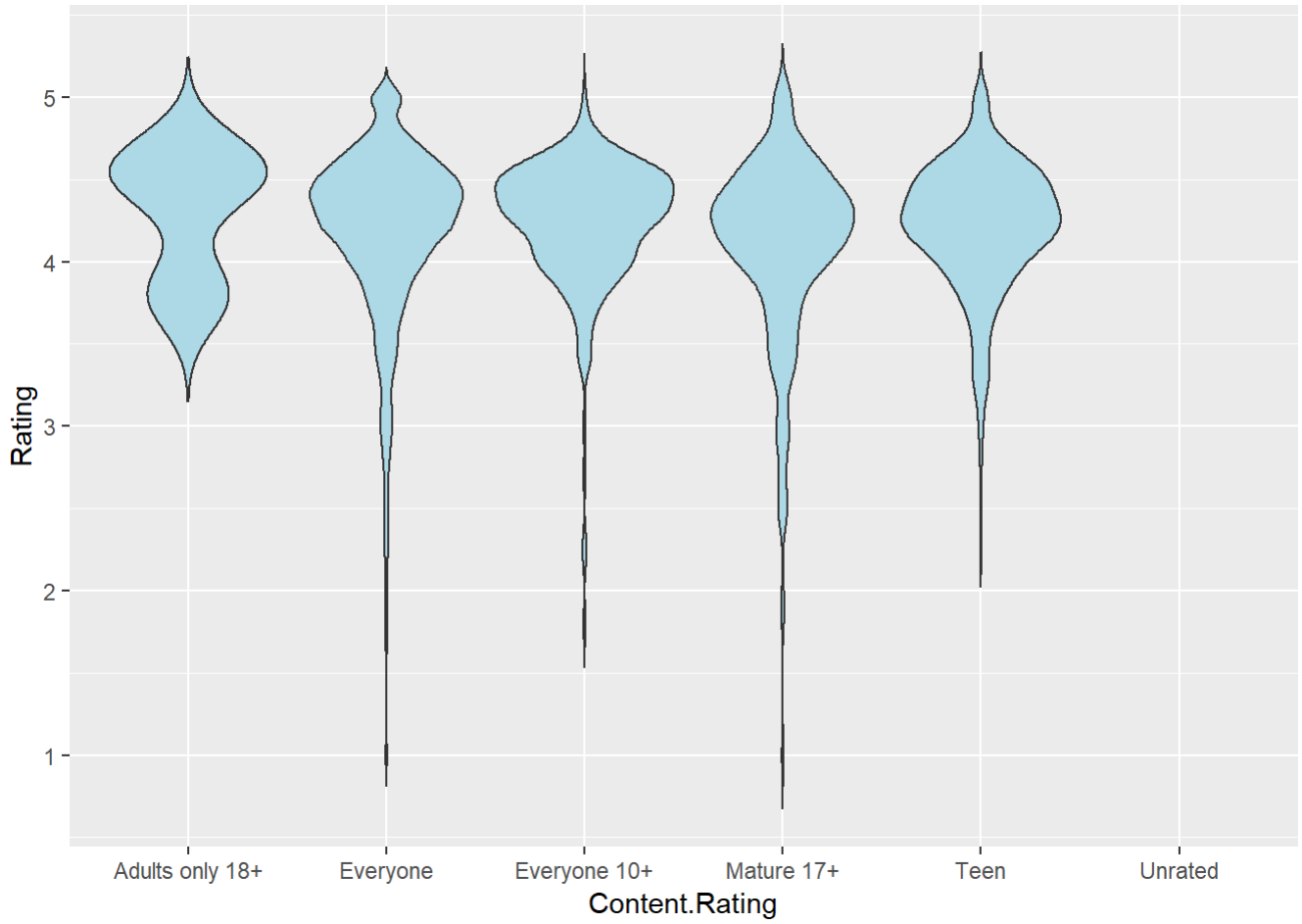

```
ggplot(Train) + geom_point(aes(x=log(Installs),y=Rating), color="purple")
```




```
ggplot(Train) + geom_point(aes(x=log(Price),y=Rating), color="red")
```

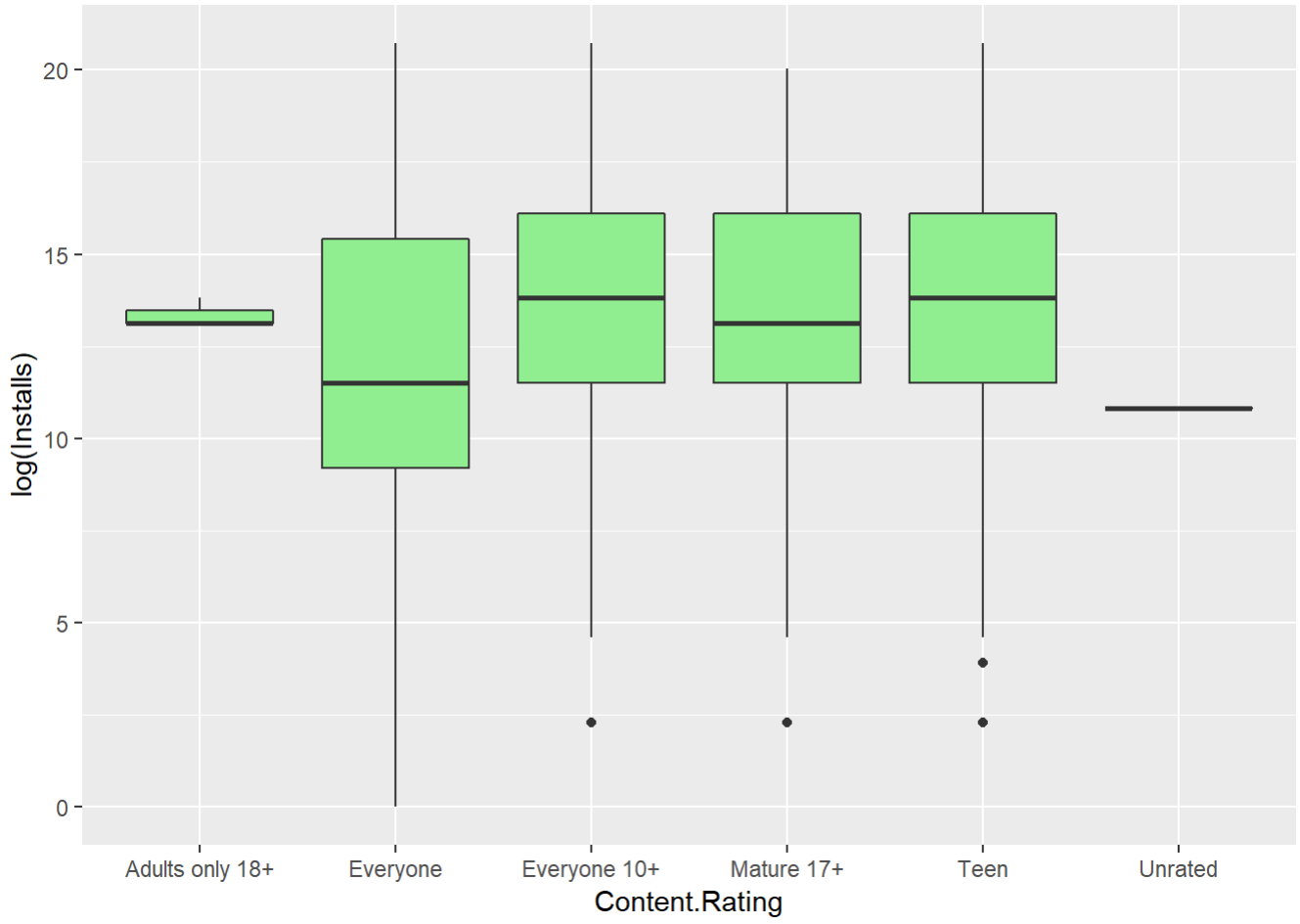



```
ggplot(Train) + geom_violin(aes(x=Content.Rating,y=Rating), trim=FALSE, fill="light blue")
```

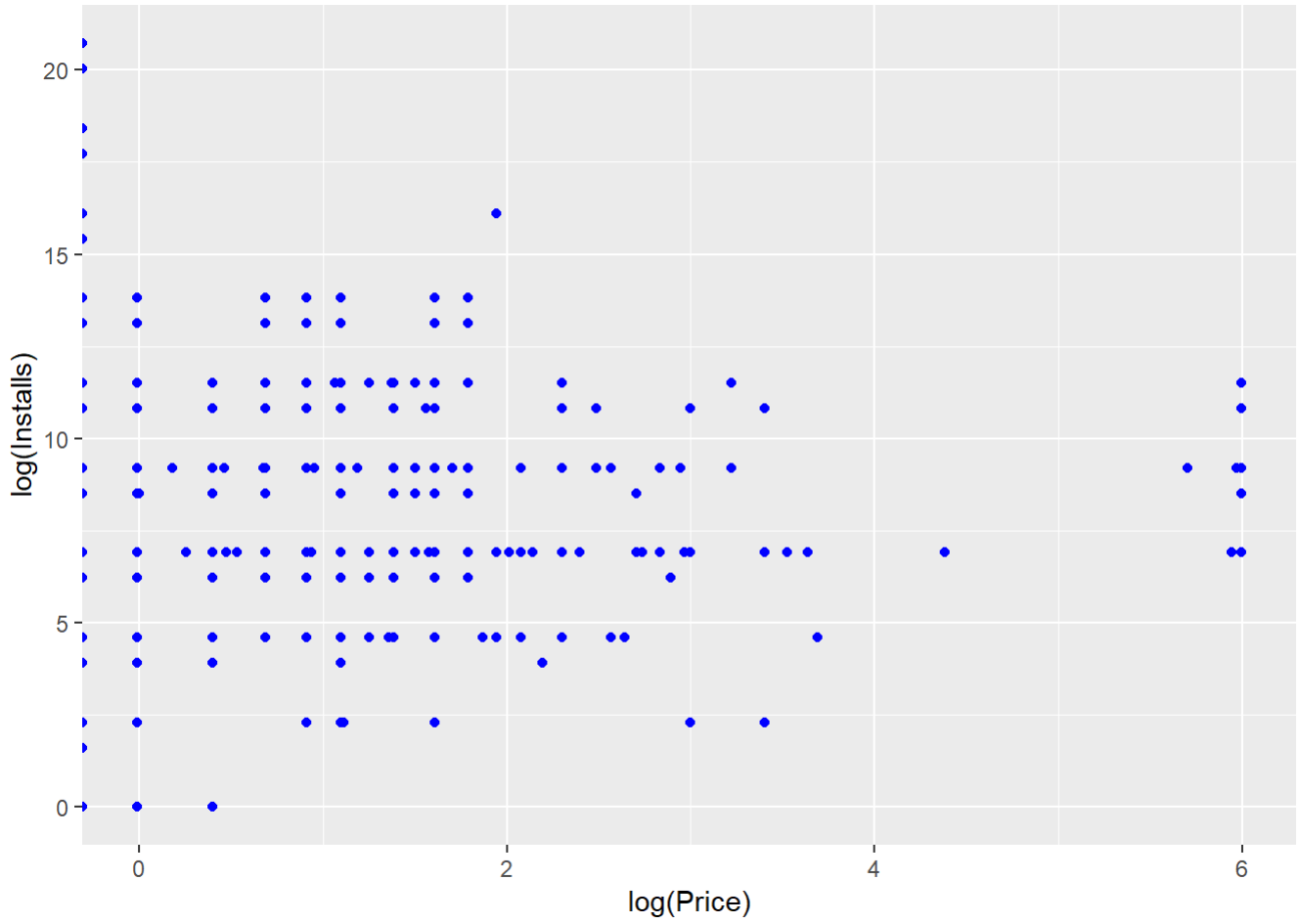


Now, let's look at some of the relationships between the other variables.

```
ggplot(Train) + geom_boxplot(aes(x=Content.Rating,y=log(Installs)), fill="light green")
```




```
ggplot(Train) + geom_point(aes(x=log(Price),y=log(Installs)), color="blue")
```



IV. Project 1: Conclusion

It appears from the data that the Reviews variable seems quite positively related to the Ratings variable. While ratings of 5 can be achieved across all Review amounts, they seem to be more common with higher Review amounts. The same can be said for the Installs variable, in that the more installations a particular app receives could correlate to the rating it is given. The Price appears to have essentially the opposite effect. With the exception of free apps, lower priced apps appear to earn higher rating scores thant more expensive apps. Content rating does not appear to affect an app's rating.

With regard to the other relationships explored, Content Rating seemd very equal across app installations, with apps rated as Everyone had the most variance, probably due to the fact that most of the apps in the data set are rated Everyone, as seen by the Content Rating bar plot earlier. When price was compared to installations, an almost bubble appeared in the bottom right portion of the plot, indicating that less expensive apps might warrent more installations from users.

Moving forward, we feel that the model we are aiming to create should include the Review, Installs, and Price variables, possibly Genres and Category as well.

V. Questions to answer in this study

Broad Questions

What makes a good app?

We are interested in seeing what attributes or characteristics, if any, of a particular app affect how “good” it is. Now, “good” is a highly subjective term and with the variety of apps that exist out there, we cannot generalize them into a category of “good” and “bad”. However, for the purposes of our look into this matter, we will be defining the “good”-ness of an app by its rating on a 5 point scale. This allows us to quantify, to some degree of accuracy, how installers felt about their experience with the app.

Can we know in advance if an app will be good?

This question is the other side of the coin to the previous question. Basically, if we know certain characteristics about an app, like genre, price, or content rating, then we can determine how well it will be rated by installers. We would like to know if this can be done with any of the variables in our data set.

Narrow Questions

Can we create a model that accurately predicts an app’s user rating based on the number of reviews, number of installations, price, and category?

After exploring the data set, we found the number of reviews, number of installations, and price to have a somewhat apparent correlation with the app’s user rating. Now, we would like to see if a model can accurately predict that rating while using those variables.

Can we create a model that accurately predicts and app’s user rating based purely on variables that would be known before the app is released, such as content rating, price, and category?

This question is an attempt to work with the second broad question, looking purely at information about an app that does not require user data. Such a model would be useful for app developers that wish to have some idea at an app’s success before it hits the market. This could guide their decisions on what apps to support, develop, and market.

Discussion

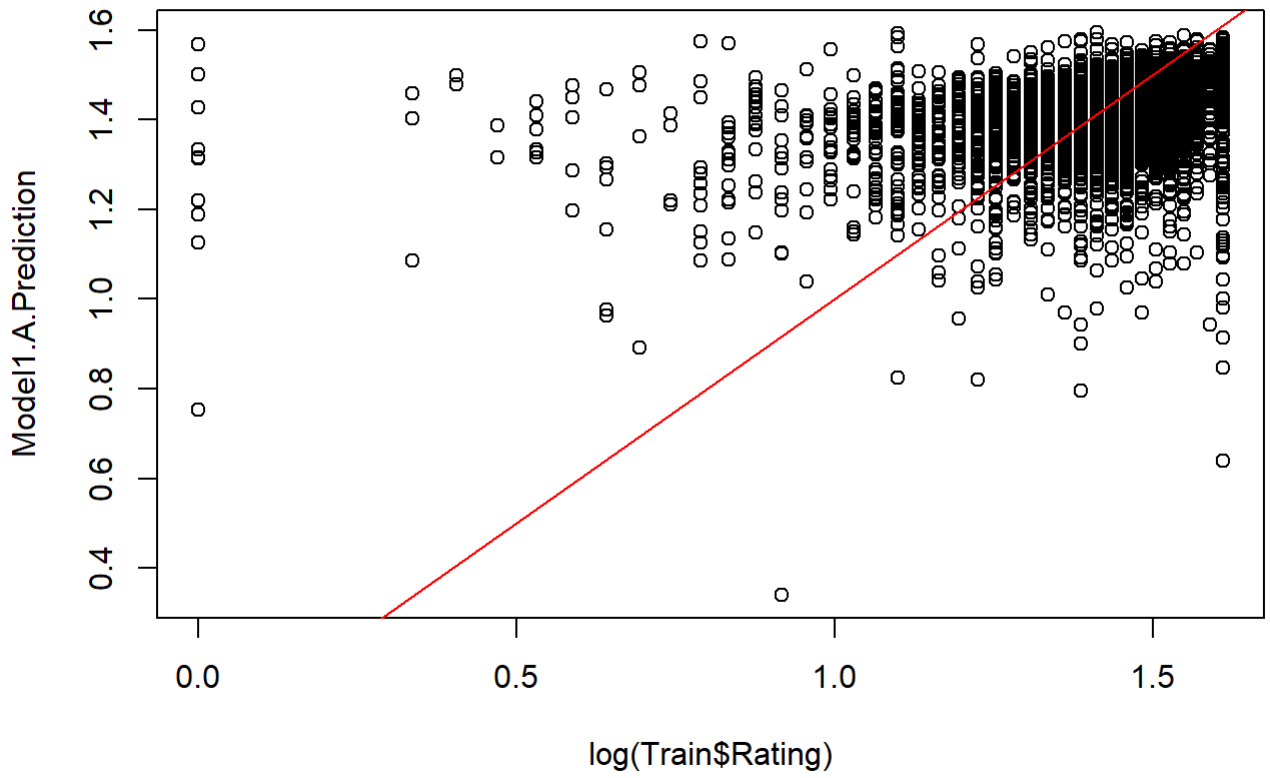
We will be creating two predictive models for our data set to hopefully predict the user rating. The first will be based on the number of reviews, number of installations, price, and category, and is hoping to just get a model that is accurate for both the train and test data. The second model will be based solely on non-user information, such as content rating, category, and price, and will hopefully be able to provide accurate predictions for both the train and test data.

VI. Modeling/Hypothesis Testing

Model 1 Unconditional Model Building

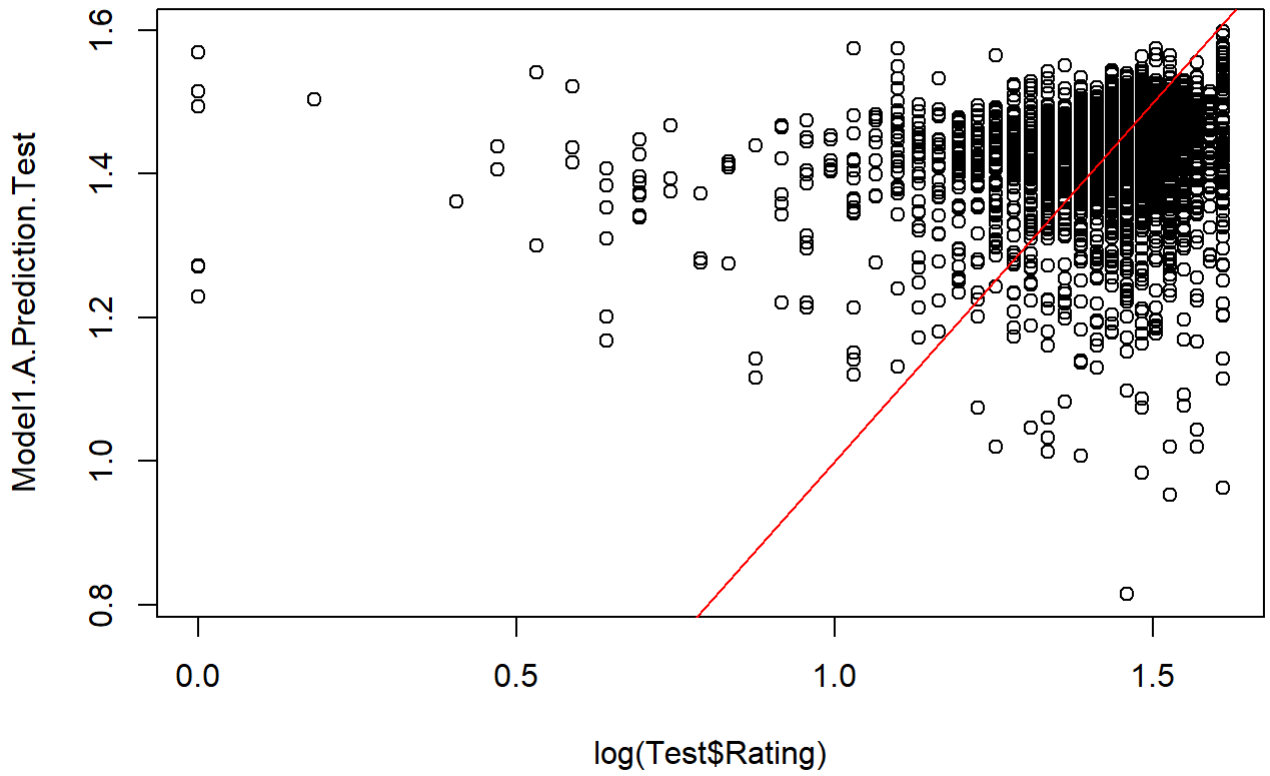
Model 1.A Random Forest

```
library(randomForest)
set.seed(42)
Model1.A <- randomForest(I(log(Rating)) ~ Reviews + Installs + Price + Category, data = Train, m
try = 4)
Model1.A.Prediction <- predict(Model1.A)
plot(log(Train$Rating),Model1.A.Prediction)
abline(0,1,col="red")
```



This random forest model incorporates the number of reviews, the price, the number of installations, and the category of the app in an effort to predict the rating. The plot seen above shows the predictions with the line in red showing where the prediction and the rating are equal. As a note, for the creation of the model, the log(Rating) was used so the plot reflects that, as doing that more evenly distributed the scores. As can be seen by the plot, the model over estimates lower user rating scores.

```
Test <- read.csv("Test.csv")
levels(Test$Category) <- levels(Train$Category)
Model1.A.Prediction.Test <- predict(Model1.A,newdata=Test)
plot(log(Test$Rating),Model1.A.Prediction.Test)
abline(0,1,col="red")
```




```
rm(Test)
```

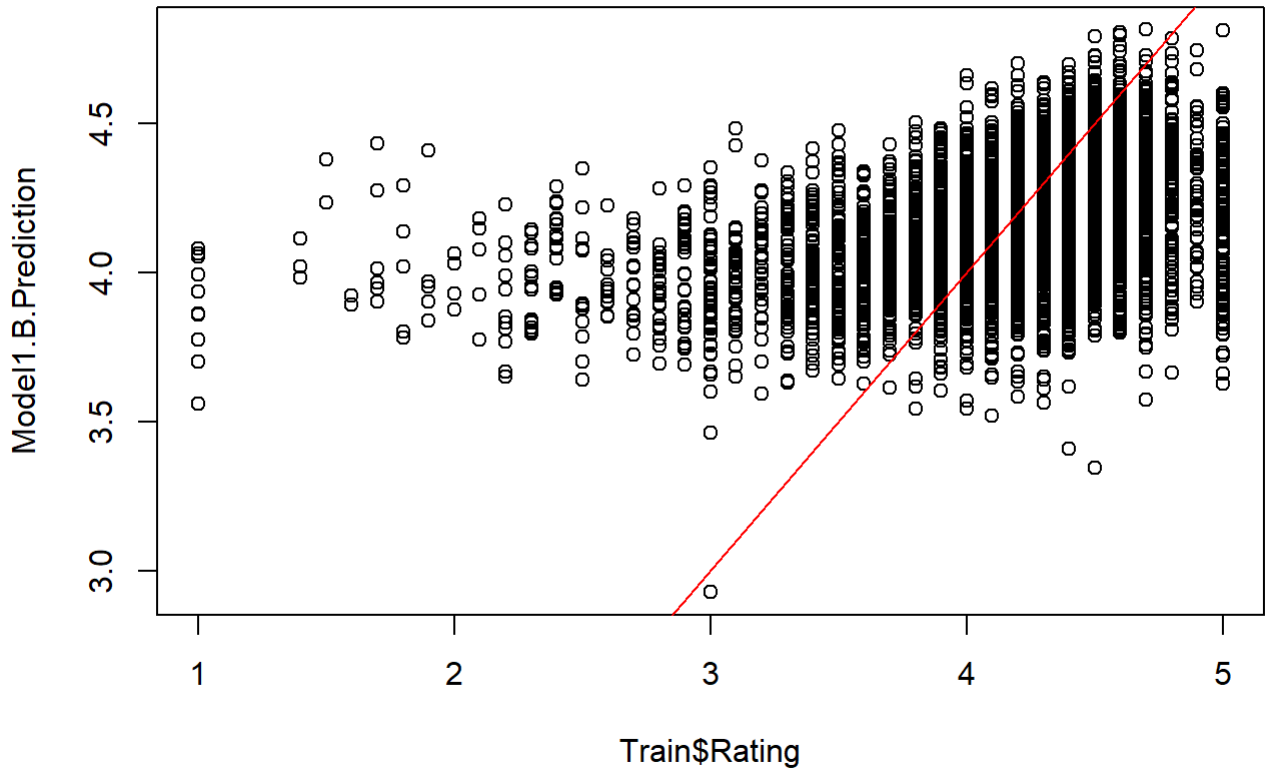
Oddly enough, at first glance, the model appears to perform slightly better with the test data, but still overestimates low user rating scores; however, it can be seen this model, while not glowingly accurate, does not perform diffently with never before seen data.

Model 1.B Linear Model

```
Model1.B <- lm(Rating ~ I(log(Reviews)) + I(log(Installs)) + Price + Category, data=Train)
Model1.B.Prediction <- predict(Model1.B)
summary(Model1.B)
```

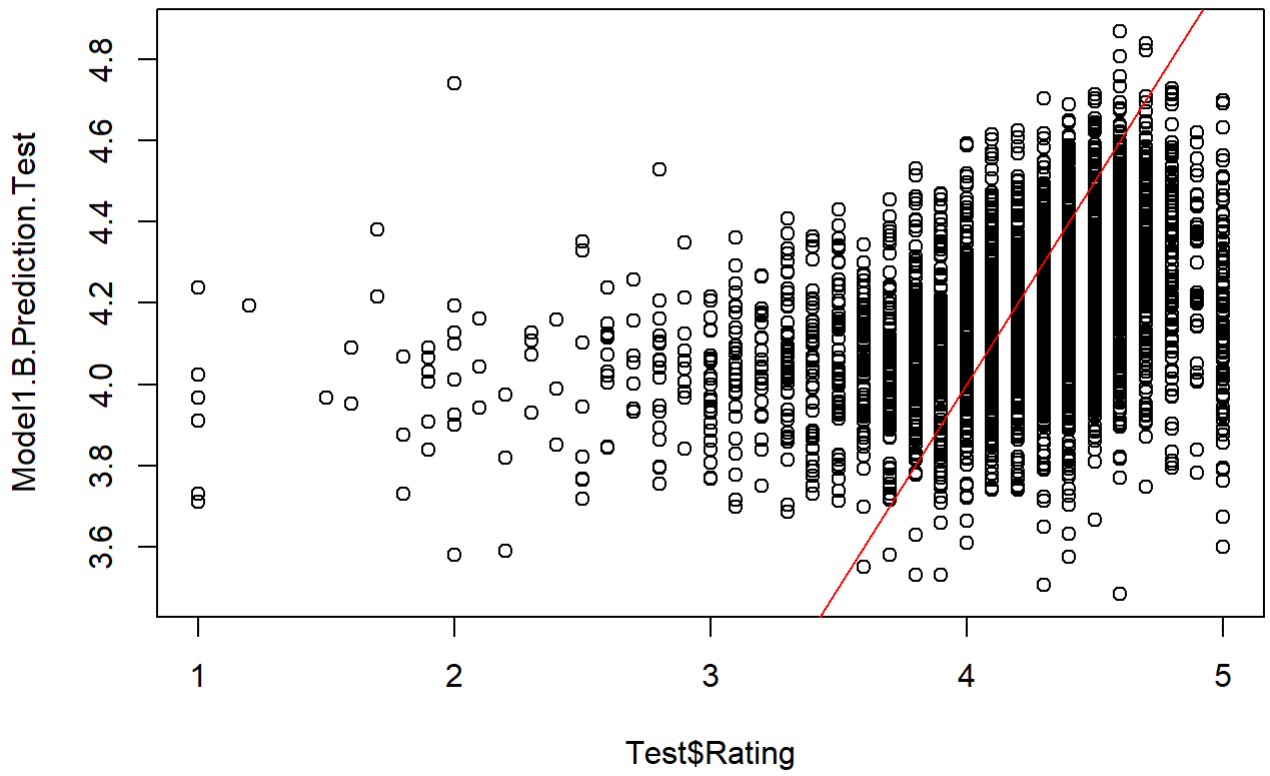
```
##
## Call:
## lm(formula = Rating ~ I(log(Reviews)) + I(log(Installs)) + Price +
##     Category, data = Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0780 -0.1784  0.0500  0.2682  1.3745
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.7824237  0.0887919  53.861  < 2e-16 ***
## I(log(Reviews))              0.1590634  0.0059847  26.578  < 2e-16 ***
## I(log(Installs))            -0.1351149  0.0060001 -22.519  < 2e-16 ***
## Price                       -0.0010287  0.0003742  -2.749 0.005988 **
## CategoryAUTO_AND_VEHICLES   -0.2597760  0.1117275  -2.325 0.020103 *
## CategoryBEAUTY              -0.0836898  0.1196155  -0.700 0.484171
## CategoryBOOKS_AND_REFERENCE -0.0441487  0.0962096  -0.459 0.646338
## CategoryBUSINESS            -0.2985174  0.0902341  -3.308 0.000945 ***
## CategoryCOMICS              -0.2554503  0.1111951  -2.297 0.021638 *
## CategoryCOMMUNICATION       -0.3639872  0.0896132  -4.062 4.94e-05 ***
## CategoryDATING              -0.4795769  0.0935859  -5.124 3.08e-07 ***
## CategoryEDUCATION           -0.1116429  0.0967430  -1.154 0.248544
## CategoryENTERTAINMENT       -0.3733013  0.0969719  -3.850 0.000120 ***
## CategoryEVENTS               0.0969359  0.1242322   0.780 0.435259
## CategoryFAMILY              -0.2292887  0.0840470  -2.728 0.006390 **
## CategoryFINANCE             -0.2579701  0.0896630  -2.877 0.004029 **
## CategoryFOOD_AND_DRINK      -0.2572679  0.1047934  -2.455 0.014119 *
## CategoryGAME                -0.2909103  0.0852579  -3.412 0.000649 ***
## CategoryHEALTH_AND_FITNESS  -0.2187885  0.0894152  -2.447 0.014440 *
## CategoryHOUSE_AND_HOME      -0.2125849  0.1067025  -1.992 0.046385 *
## CategoryLIBRARIES_AND_DEMO  -0.1725551  0.1137155  -1.517 0.129215
## CategoryLIFESTYLE           -0.3032571  0.0899269  -3.372 0.000751 ***
## CategoryMAPS_AND_NAVIGATION -0.4123675  0.1002973  -4.111 3.99e-05 ***
## CategoryMEDICAL             -0.1924384  0.0893484  -2.154 0.031299 *
## CategoryNEWS_AND_MAGAZINES  -0.3592561  0.0919499  -3.907 9.45e-05 ***
## CategoryPARENTING            0.0861430  0.1300361   0.662 0.507707
## CategoryPERSONALIZATION     -0.1156563  0.0898918  -1.287 0.198282
## CategoryPHOTOGRAPHY         -0.2624715  0.0896624  -2.927 0.003433 **
## CategoryPRODUCTIVITY        -0.2243243  0.0893940  -2.509 0.012122 *
## CategorySHOPPING            -0.2590566  0.0939940  -2.756 0.005869 **
## CategorySOCIAL              -0.2580975  0.0914772  -2.821 0.004798 **
## CategorySPORTS              -0.2977191  0.0894411  -3.329 0.000878 ***
## CategoryTOOLS               -0.3426418  0.0856729  -3.999 6.43e-05 ***
## CategoryTRAVEL_AND_LOCAL    -0.3573351  0.0926315  -3.858 0.000116 ***
## CategoryVIDEO_PLAYERS       -0.3702765  0.0973459  -3.804 0.000144 ***
## CategoryWEATHER             -0.2606247  0.1056110  -2.468 0.013625 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4816 on 5584 degrees of freedom
## Multiple R-squared:  0.1559, Adjusted R-squared:  0.1506
## F-statistic: 29.47 on 35 and 5584 DF,  p-value: < 2.2e-16
```

```
plot(Train$Rating,Model1.B.Prediction)
abline(0,1,col="red")
```



This linear model works to predict the rating score with the same predictor variables as the previous model, and it appears to perform quite poorly, only fitting the data around 14%.

```
Test <- read.csv("Test.csv")
Model1.B.Prediction.Test <- predict(Model1.B,newdata=Test)
plot(Test$Rating,Model1.B.Prediction.Test)
abline(0,1,col="red")
```



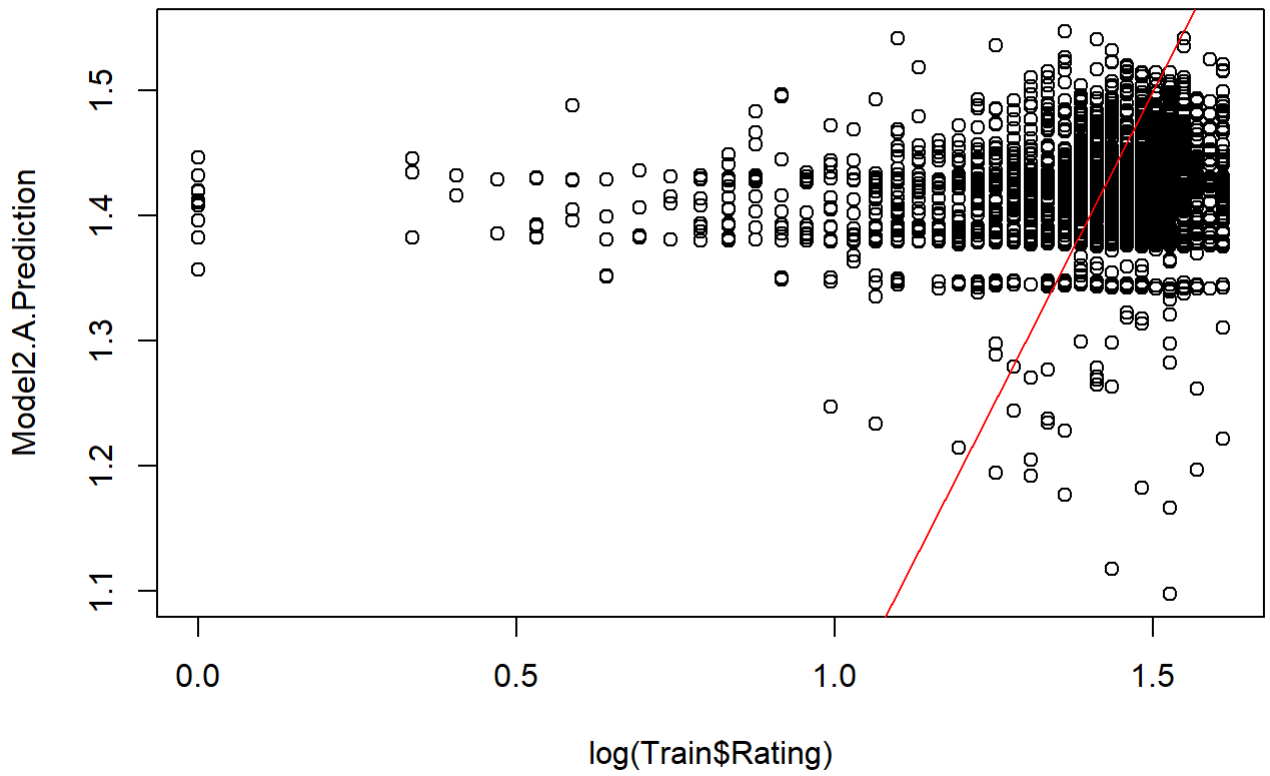

```
rm(Test)
```

The data seems to perform worse for the test data set, indicating that the linear model slightly over fits the training data and cannot consistently perform with never before seen data. Overall, this model is worse than the random forest model discussed previously.

Model 2 Non-User Data Model Building

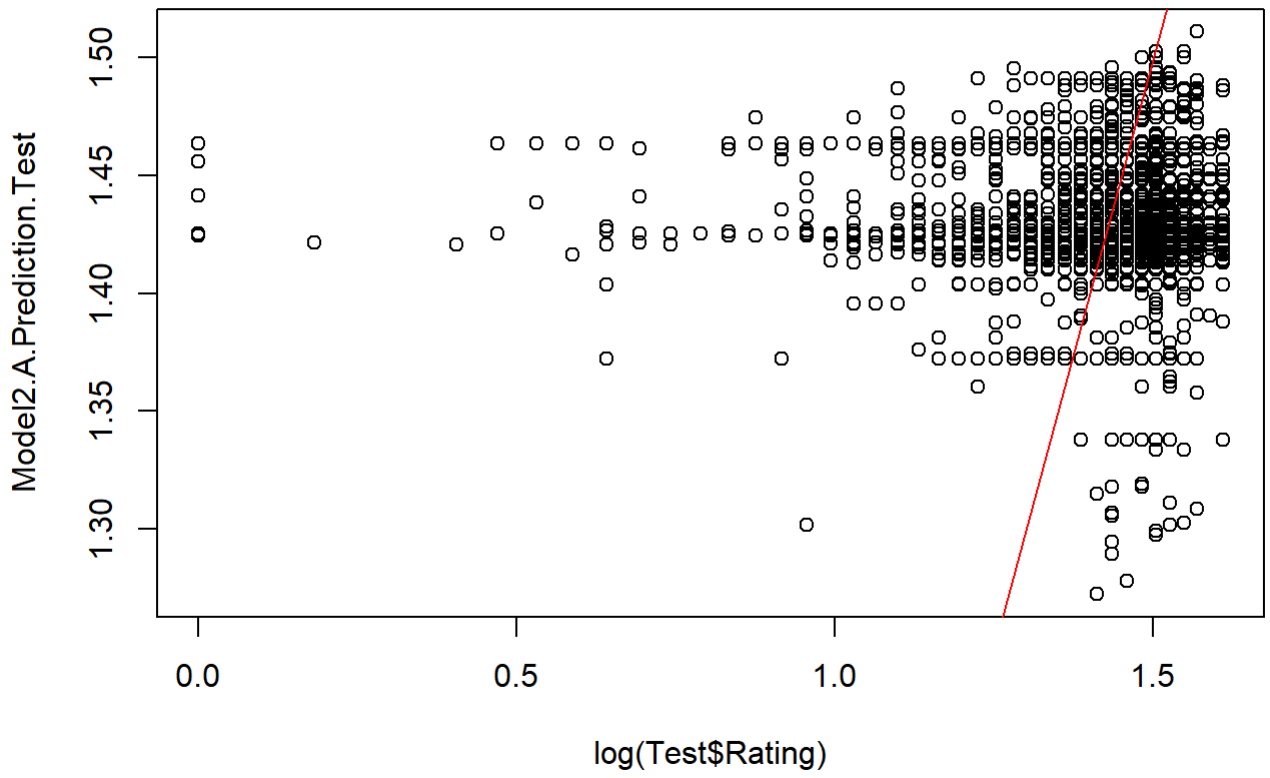
Model 2.A Random Forest

```
library(randomForest)
set.seed(42)
Model2.A <- randomForest(I(log(Rating)) ~ Price + Category + Content.Rating, data = Train, mtry
 = 3)
Model2.A.Prediction <- predict(Model2.A)
plot(log(Train$Rating),Model2.A.Prediction)
abline(0,1,col="red")
```



Here, we have a random forest model similar to that of Model 1.A, with the only exception being that it is based upon the data that can be gathered before the app is relased to the public, price, category, and content rating. Much like in all of the other models, this one also overestimates lower user ratings.

```
Test <- read.csv("Test.csv")
levels(Test$Category) <- levels(Train$Category)
levels(Test$Content.Rating) <- levels(Train$Content.Rating)
Model2.A.Prediction.Test <- predict(Model2.A,newdata=Test)
plot(log(Test$Rating),Model2.A.Prediction.Test)
abline(0,1,col="red")
```




```
rm(Test)
```

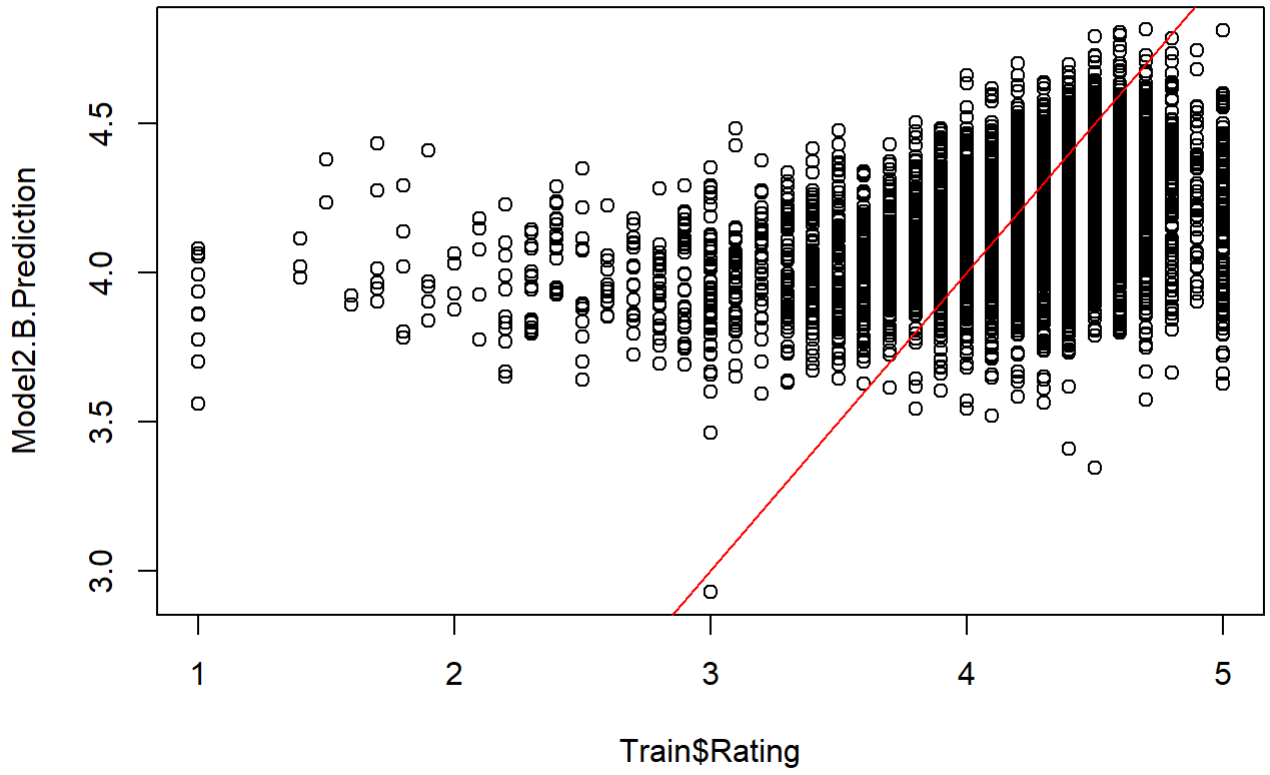
Surprisingly, the model appears to perform better on the test data than it did on the training data. While it still overestimates, lower user rating scores, it is not as dramatic as with the training data, which is interesting.

Model 2.B Linear Model

```
Model2.B <- lm(Rating ~ I(log(Reviews)) + I(log(Installs)) + Price + Category, data=Train)
Model2.B.Prediction <- predict(Model2.B)
summary(Model2.B)
```

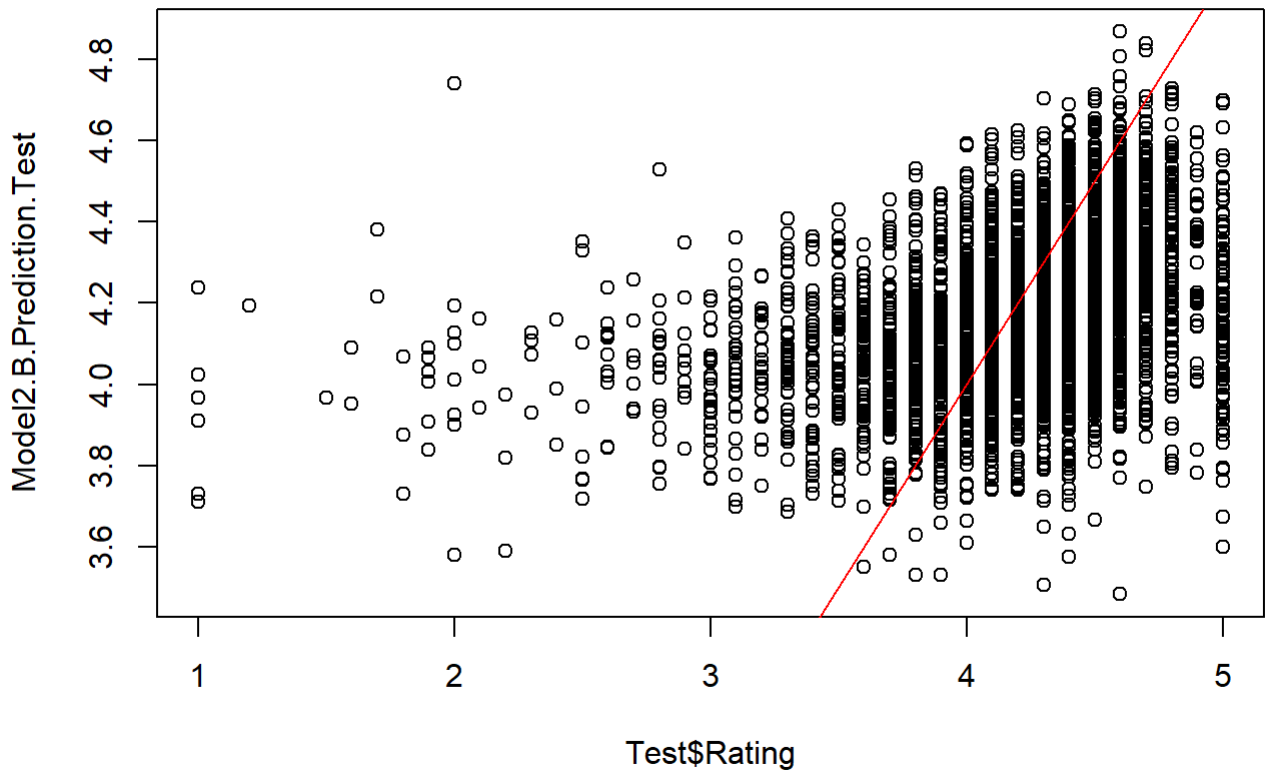
```
##
## Call:
## lm(formula = Rating ~ I(log(Reviews)) + I(log(Installs)) + Price +
##     Category, data = Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0780 -0.1784  0.0500  0.2682  1.3745
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.7824237  0.0887919  53.861  < 2e-16 ***
## I(log(Reviews))              0.1590634  0.0059847  26.578  < 2e-16 ***
## I(log(Installs))            -0.1351149  0.0060001 -22.519  < 2e-16 ***
## Price                       -0.0010287  0.0003742  -2.749 0.005988 **
## CategoryAUTO_AND_VEHICLES   -0.2597760  0.1117275  -2.325 0.020103 *
## CategoryBEAUTY              -0.0836898  0.1196155  -0.700 0.484171
## CategoryBOOKS_AND_REFERENCE -0.0441487  0.0962096  -0.459 0.646338
## CategoryBUSINESS            -0.2985174  0.0902341  -3.308 0.000945 ***
## CategoryCOMICS              -0.2554503  0.1111951  -2.297 0.021638 *
## CategoryCOMMUNICATION       -0.3639872  0.0896132  -4.062 4.94e-05 ***
## CategoryDATING              -0.4795769  0.0935859  -5.124 3.08e-07 ***
## CategoryEDUCATION           -0.1116429  0.0967430  -1.154 0.248544
## CategoryENTERTAINMENT       -0.3733013  0.0969719  -3.850 0.000120 ***
## CategoryEVENTS               0.0969359  0.1242322   0.780 0.435259
## CategoryFAMILY              -0.2292887  0.0840470  -2.728 0.006390 **
## CategoryFINANCE             -0.2579701  0.0896630  -2.877 0.004029 **
## CategoryFOOD_AND_DRINK      -0.2572679  0.1047934  -2.455 0.014119 *
## CategoryGAME                -0.2909103  0.0852579  -3.412 0.000649 ***
## CategoryHEALTH_AND_FITNESS  -0.2187885  0.0894152  -2.447 0.014440 *
## CategoryHOUSE_AND_HOME      -0.2125849  0.1067025  -1.992 0.046385 *
## CategoryLIBRARIES_AND_DEMO  -0.1725551  0.1137155  -1.517 0.129215
## CategoryLIFESTYLE           -0.3032571  0.0899269  -3.372 0.000751 ***
## CategoryMAPS_AND_NAVIGATION -0.4123675  0.1002973  -4.111 3.99e-05 ***
## CategoryMEDICAL             -0.1924384  0.0893484  -2.154 0.031299 *
## CategoryNEWS_AND_MAGAZINES  -0.3592561  0.0919499  -3.907 9.45e-05 ***
## CategoryPARENTING            0.0861430  0.1300361   0.662 0.507707
## CategoryPERSONALIZATION     -0.1156563  0.0898918  -1.287 0.198282
## CategoryPHOTOGRAPHY         -0.2624715  0.0896624  -2.927 0.003433 **
## CategoryPRODUCTIVITY        -0.2243243  0.0893940  -2.509 0.012122 *
## CategorySHOPPING            -0.2590566  0.0939940  -2.756 0.005869 **
## CategorySOCIAL              -0.2580975  0.0914772  -2.821 0.004798 **
## CategorySPORTS              -0.2977191  0.0894411  -3.329 0.000878 ***
## CategoryTOOLS               -0.3426418  0.0856729  -3.999 6.43e-05 ***
## CategoryTRAVEL_AND_LOCAL    -0.3573351  0.0926315  -3.858 0.000116 ***
## CategoryVIDEO_PLAYERS       -0.3702765  0.0973459  -3.804 0.000144 ***
## CategoryWEATHER             -0.2606247  0.1056110  -2.468 0.013625 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4816 on 5584 degrees of freedom
## Multiple R-squared:  0.1559, Adjusted R-squared:  0.1506
## F-statistic: 29.47 on 35 and 5584 DF,  p-value: < 2.2e-16
```

```
plot(Train$Rating,Model2.B.Prediction)
abline(0,1,col="red")
```



This model is depressingly inaccurate, and drastically overestimates lower user rating scores. It is a linear model based on the same predictors as Model2.A. Much like Model1.A, it only fits the data about 15%.

```
Test <- read.csv("Test.csv")
Model2.B.Prediction.Test <- predict(Model2.B,newdata=Test)
plot(Test$Rating,Model2.B.Prediction.Test)
abline(0,1,col="red")
```



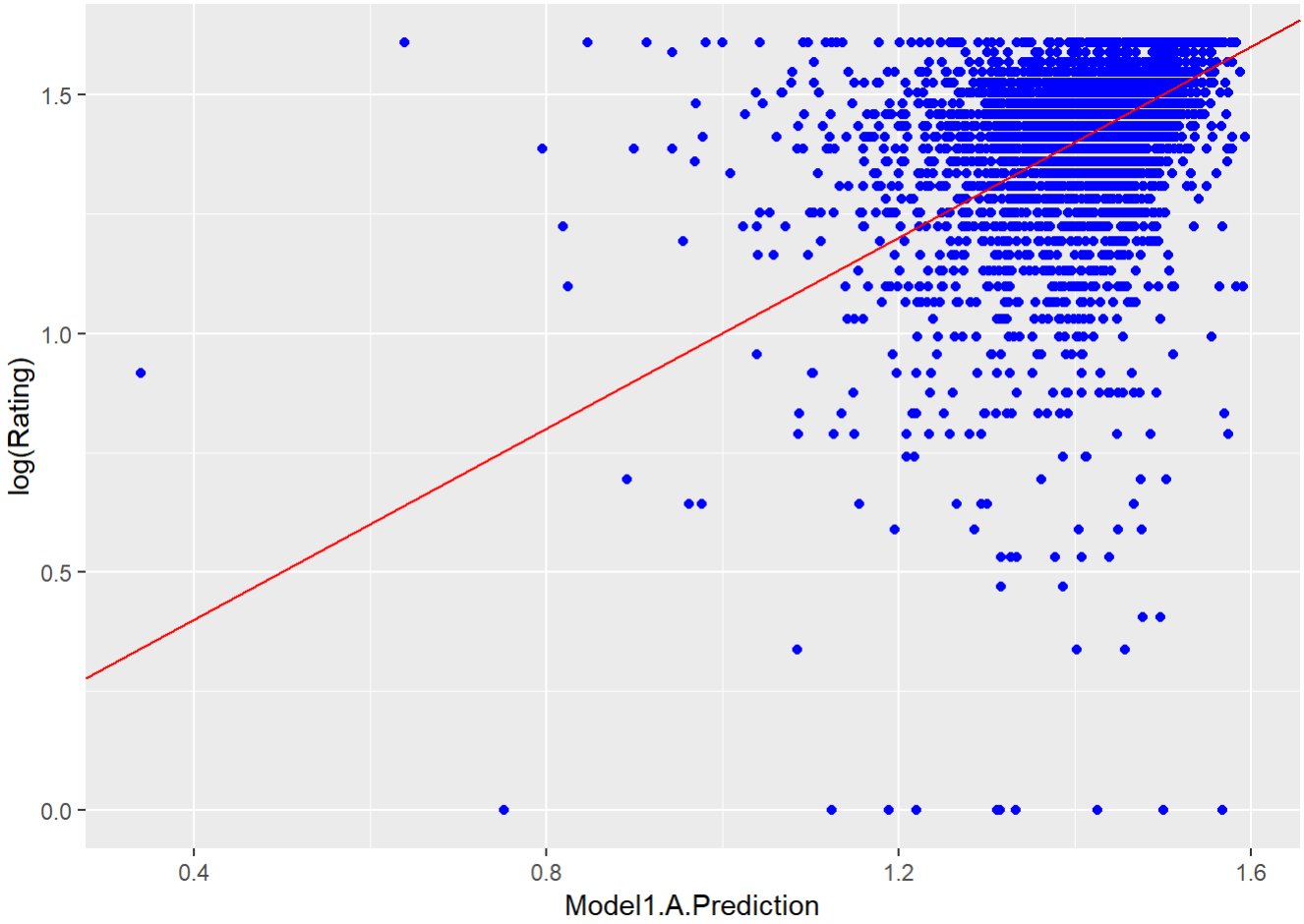

```
rm(Test)
```

The model performs worse when tested against the test data than to the training data. This indicates that, as poor as the model fit the training data, it is overfitting the data and is not generalizing all of the data well. It overestimates the lower user rating scores quite a bit.

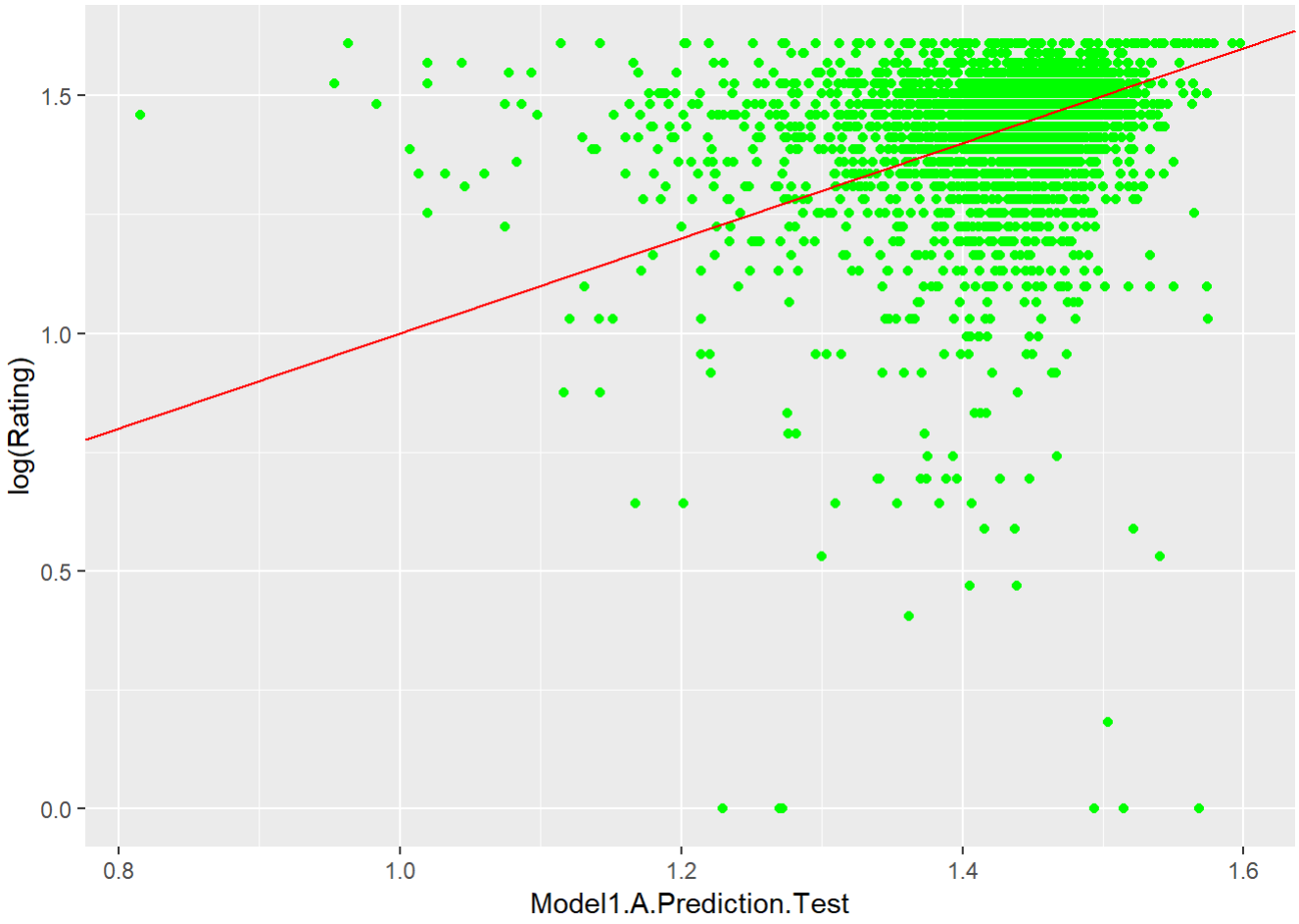
VII. Results

It was determined when the models were built and tested that the Random Forest models were superior to the linear models. This determination was based on how close the data points were to the y=x line on the plots, and how many drastic points there were. This decision was also based on the performance of the models on the test data, in which the Random Forest models performed as well, if not, better, than on the training dat while the linear model was not able to generalize the data to a satisfactory degree. There were two final models that were developed. Model1A, a random forest model, incorporated the Category, Reviews, Installs, and Price variables in order to predict the log of the Rating variable. Its purpose was to try and unconditionally predict the user rating score. Model2A, a random forest model, incorporated the Category, Price, and Content.Rating variables in order to predict the log of the Rating variable. Its purpose was to try and conditionally predict the user rating score, with the condition being that only non-user related data could be used. In other words, only data that could be collected before the app was released to the market could be used. Overall, both models did not accurately predict the user rating score very well.

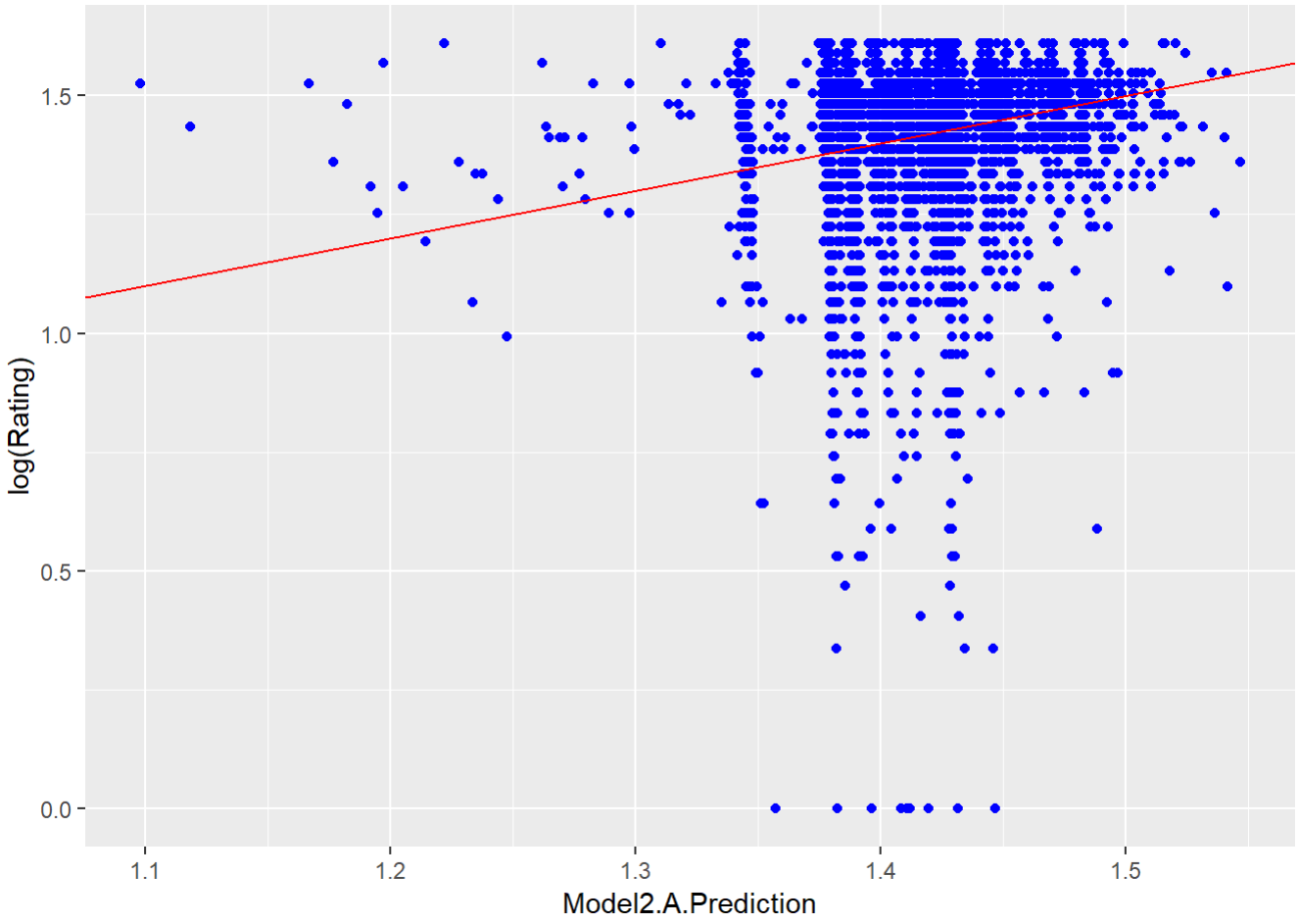
```
library(ggplot2)
Test <- read.csv("Test.csv")
ggplot(Train) + geom_point(aes(x=Model1.A.Prediction,y=log(Rating)),color = "blue") +
  geom_abline(intercept = 0, color = "red")
```




```
ggplot(Test) + geom_point(aes(x=Model1.A.Prediction.Test,y=log(Rating)),color = "green") +
  geom_abline(intercept = 0, color = "red")
```




```
ggplot(Train) + geom_point(aes(x=Model2.A.Prediction,y=log(Rating)),color = "blue") +
  geom_abline(intercept = 0, color = "red")
```




```
ggplot(Test) + geom_point(aes(x=Model2.A.Prediction.Test,y=log(Rating)),color = "green") +
  geom_abline(intercept = 0, color = "red")
```


```
rm(Test)
```

These plots are the results of the models and their predictions. Blue models are for training, green models are for testing. Model 1A is the unconditional model. Model 2A is the conditional model.

VIII. Discussion and Conclusions

Overall, our models were not able to generate very accurate cases. In both conditions, the models tended to overestimate the user rating scores, especially for those who earned very low rating scores. While we would like to believe this is due to the model's inherent optimism, we feel that this is due to the majority of the user rating scores being higher values, resulting in a predictive model which tends to moreso pick higher ratings as its predictions. Also, it should be noted that having the number of installations and the number of reviews greatly aided in the predictive power of the model. This indicates that there is a real correlation, however difficult to discern, between user-related info and the user rating score. The models are not at all decent at what they do; meaning that future work should strive to collect more data from different variable sources to try and more accurately determine a user rating score in advance. This would in turn force us to understand what makes a good app and what people want in their apps. Provisionally from this study, we can discern that, on average, a free, highly downloaded, highly reviewed, rated E for Everyone app would most likely lead to a high user rating score. Our research is important because it shows that there is something to being able to predict a user rating score. This is because the models, though sad as they might be, are not completely incapable, suggesting some form of correlation somewhere. Being able to predict a user rating score of an app before it launches could be very helpful to businesses and corporations in deciding which apps they will support, and which they will cast off to the side in favor an app with a better predicted outcome on the market.