Google Play Store Apps Academic Project

Andrea Bradshaw, Bing-Je Wu, Barbara A. Jaehn, Phillip A. Garver

June 9, 2019

Data Loading & Cleansing

Load the Google Play Store .CSV File

```
urlToRead <- "https://raw.githubusercontent.com/bing020815/Syracuse-University/master/Google%20Play%20S
appData<- read.csv(url(urlToRead))
nrow(appData)
```

```
## [1] 10841
```

Remove Erroneous Row Import and any Duplicates

```
appData <- appData[-which(appData$Installs=="Free"),]
paste0("Before: ", nrow(appData))
```

```
## [1] "Before: 10840"
```

```
appData <- unique(appData)
paste0("After: ", nrow(appData))
```

```
## [1] "After: 10357"
```

Rename All Fields to Use Underscores Between Words Instead of Dots (.)

```
colnames(appData) <- c("App", "Category", "Rating", "Reviews", "Size", "Installs",
                       "Type","Price", "Content_Rating", "Genre", "Last_Updated",
                       "Current_Ver","Android_Ver")

#Renumber rows
row.names(appData) <- NULL
paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 10357"
```

Identify and remove all N/A and NAN fields

```r
colSums(is.na(appData))
```

```
##            App       Category         Rating        Reviews           Size
##              0              0           1465              0              0
##       Installs           Type          Price Content_Rating          Genre
##              0              0              0              0              0
##   Last_Updated    Current_Ver    Android_Ver
##              0              0              0
```

```r
appData <- na.omit(appData)
appData <- appData[-which(appData$Android_Ver =="NaN"),]

#Remove the unused Factor Levels in all columns
appData <- droplevels(appData)

#Confirm N/As have been removed
paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8890"
```

```r
colSums(is.na(appData))
```

```
##            App       Category         Rating        Reviews           Size
##              0              0              0              0              0
##       Installs           Type          Price Content_Rating          Genre
##              0              0              0              0              0
##   Last_Updated    Current_Ver    Android_Ver
##              0              0              0
```

Convert Size Field to Kilobytes

```r
# Split dataset into three group SizeM SizeK NotSize
SizeM <- appData[grepl(pattern = "M",appData$Size),]
NotSize <- appData[!grepl(pattern = "M",appData$Size),]
SizeK <- NotSize[grepl(pattern = "k",NotSize$Size),]
NotSize <-NotSize[!grepl(pattern = "k",NotSize$Size),]
# verify the number of records
ifelse (nrow(appData) == (nrow(SizeM) + nrow(NotSize) + nrow(SizeK)),
        "Record Counts Match", "Record Count Discrepancy")
```

```
## [1] "Record Counts Match"
```

```r
# Converting everything into KB
SizeM$Size <- gsub("\\M","",SizeM$Size)
SizeM$Size <- as.numeric(SizeM$Size)
SizeM$Size <- SizeM$Size*1024
SizeK$Size <- gsub("\\k","",SizeK$Size)
SizeK$Size <- as.numeric(SizeK$Size)
```

```r
appData <- data.frame(rbind(SizeM,SizeK,NotSize))
#Remove + from Size field data

paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8890"
```

Remove + in Installs and change into a number (so the buckets to sort numerically)

```r
options("scipen"=100, "digits"=4)
appData$Installs <- gsub("\\+","",appData$Installs)
appData$Installs <- gsub("\\,", "", appData$Installs)
appData$Installs <- as.numeric(appData$Installs)
#appData <- within(appData, {Installs <- as.numeric(as.character(Installs))} )

paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8890"
```

Convert Reviews to a number

```r
appData$Reviews <- as.numeric(as.character(appData$Reviews))
#appData <- within(appData, {Reviews <- as.numeric(as.character(Reviews))})

paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8890"
```

Remove Dollar Sign from Price Field and change to numeric data for Price field

```r
#Put Zeroes into double format and remove $
appData$Price <- gsub("\\$", "", appData$Price)
appData$Price <- as.numeric(appData$Price)

paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8890"
```

Identify duplicate entries for Apps and take the Max Reviews and all remaining data as all other columns are the same for these enries.

```r
library(dplyr)
appData <- appData %>%

            group_by(App, Rating) %>%

            arrange(desc(Reviews)) %>%

            slice(1) %>%

            ungroup()

#appData <- appData %>% distinct(App,Rating, .keep_all = TRUE)

##reference:  https://stackoverflow.com/questions/24237399/how-to-select-the-rows-with-maximum-values-i
##reference:  https://www.datanovia.com/en/lessons/identify-and-remove-duplicate-data-in-r/
```

Remove Last_Updated, Current_Ver and Android_Ver Fields

```r
appData <- appData[, -11:-13]
paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8211"
```

Break out Sub-Genre's Where Multiple Genre Entries Exist in the Same Field, then set NAs to None

```r
library(tidyr)
appData <- separate(appData, Genre, into = c("Genre", "SubGenre"), sep = ";")

library(sqldf)
#Show Number of apps with a SubGenre:
sqldf('select count(*) from appData where SubGenre is NOT NULL')
```

```
##   count(*)
## 1      381
```

```r
#Replace SubGenre NA values with None
appData$SubGenre[is.na(appData$SubGenre)] = "None"

paste0("Count After: ", nrow(appData))
```

```
## [1] "Count After: 8211"
```

```r
str(appData)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    8211 obs. of  11 variables:
##  $ App           : Factor w/ 8194 levels "- Free Comics - Comic Apps",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 6 30 7 12 29 30 30 24 24 24 ...
##  $ Rating        : num  3.5 4.5 4.7 3.6 3.2 3.9 4.2 4 4.5 4.4 ...
##  $ Reviews       : num  115 259 573 21433 4 ...
##  $ Size          : chr  "9318.4" "203" "54272" "21504" ...
##  $ Installs      : num  10000 10000 10000 1000000 100 500000 1000000 100 1000 5000 ...
##  $ Type          : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 2 1 1 ...
##  $ Price         : num  0 0 0 0 0 0 0 0.99 0 0 ...
##  $ Content_Rating: Factor w/ 6 levels "Adults only 18+",..: 4 2 4 4 2 2 2 2 2 2 ...
##  $ Genre         : chr  "Comics" "Tools" "Communication" "Entertainment" ...
##  $ SubGenre      : chr  "None" "None" "None" "None" ...
```

Preliminary Prep of RandomForsest Data Set, and Adding SizeCategories

```r
#Remove App column from target dataset for randomForest analysis
rfappData <- data.frame(appData[,-1])

#Create SizeCategory column
rfappData$SizeCategory <- NULL

#Switch to SizeCategories to include Varies by device
rfappData$SizeCategory[rfappData$Size == "Varies with device"] = "Varies with device"
rfappData$SizeCategory[as.numeric(is.na(rfappData$Size))] = "Varies with Device"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 0 &
                         as.numeric(rfappData$Size) <= 1024] = "0-1MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 1024 &
                         as.numeric(rfappData$Size) <= 5120] = "1MB-5MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 5120 &
                         as.numeric(rfappData$Size) <= 10240] = "5MB-10MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 10240 &
                         as.numeric(rfappData$Size) <= 25600] = "10MB-25MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 25600 &
                         as.numeric(rfappData$Size) <= 51200] = "25MB-50MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 51200 &
                         as.numeric(rfappData$Size) <= 76800] = "50MB-75MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 76800 &
                         as.numeric(rfappData$Size) <= 102400] = "75MB-100MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 102400 &
                         as.numeric(rfappData$Size) <= 128000] = "100MB-125MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 128000 &
                         as.numeric(rfappData$Size) <= 153600] = "125MB-150MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 153600 &
                         as.numeric(rfappData$Size) <= 179000] = "150MB-175MB"
rfappData$SizeCategory[as.numeric(rfappData$Size) > 179000 &
                         as.numeric(rfappData$Size) <= 204800] = "175MB-200MB"

str(rfappData)
```

```
## 'data.frame':    8211 obs. of  11 variables:
##  $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 6 30 7 12 29 30 30 24 24 24 ...
##  $ Rating        : num  3.5 4.5 4.7 3.6 3.2 3.9 4.2 4 4.5 4.4 ...
```

```
##  $ Reviews       : num  115 259 573 21433 4 ...
##  $ Size          : chr  "9318.4" "203" "54272" "21504" ...
##  $ Installs      : num  10000 10000 10000 1000000 100 500000 1000000 100 1000 5000 ...
##  $ Type          : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 2 1 1 ...
##  $ Price         : num  0 0 0 0 0 0 0 0.99 0 0 ...
##  $ Content_Rating: Factor w/ 6 levels "Adults only 18+",..: 4 2 4 4 2 2 2 2 2 2 ...
##  $ Genre         : chr  "Comics" "Tools" "Communication" "Entertainment" ...
##  $ SubGenre      : chr  "None" "None" "None" "None" ...
##  $ SizeCategory  : chr  "5MB-10MB" "0-1MB" "50MB-75MB" "10MB-25MB" ...
```

Display initial descriptive statistics & visualizations of all elements.

Summarize our data

```r
summary(appData[,2:9])
```

```
##          Category        Rating         Reviews                Size
##  FAMILY      :1652   Min.   :1.00   Min.   :       1   Length:8211
##  GAME        : 900   1st Qu.:4.00   1st Qu.:     127   Class :character
##  TOOLS       : 721   Median :4.30   Median :    3031   Mode  :character
##  FINANCE     : 302   Mean   :4.17   Mean   :  255124
##  LIFESTYLE   : 301   3rd Qu.:4.50   3rd Qu.:   44044
##  PRODUCTIVITY: 301   Max.   :5.00   Max.   :78158306
##  (Other)     :4034
##     Installs             Type          Price              Content_Rating
##  Min.   :         1   Free:7608   Min.   :  0   Adults only 18+:   3
##  1st Qu.:     10000   Paid: 603   1st Qu.:  0   Everyone       :6633
##  Median :    100000               Median :  0   Everyone 10+   : 305
##  Mean   :   9179208               Mean   :  1   Mature 17+     : 357
##  3rd Qu.:   1000000               3rd Qu.:  0   Teen           : 912
##  Max.   :1000000000               Max.   :400   Unrated        :   1
##
```

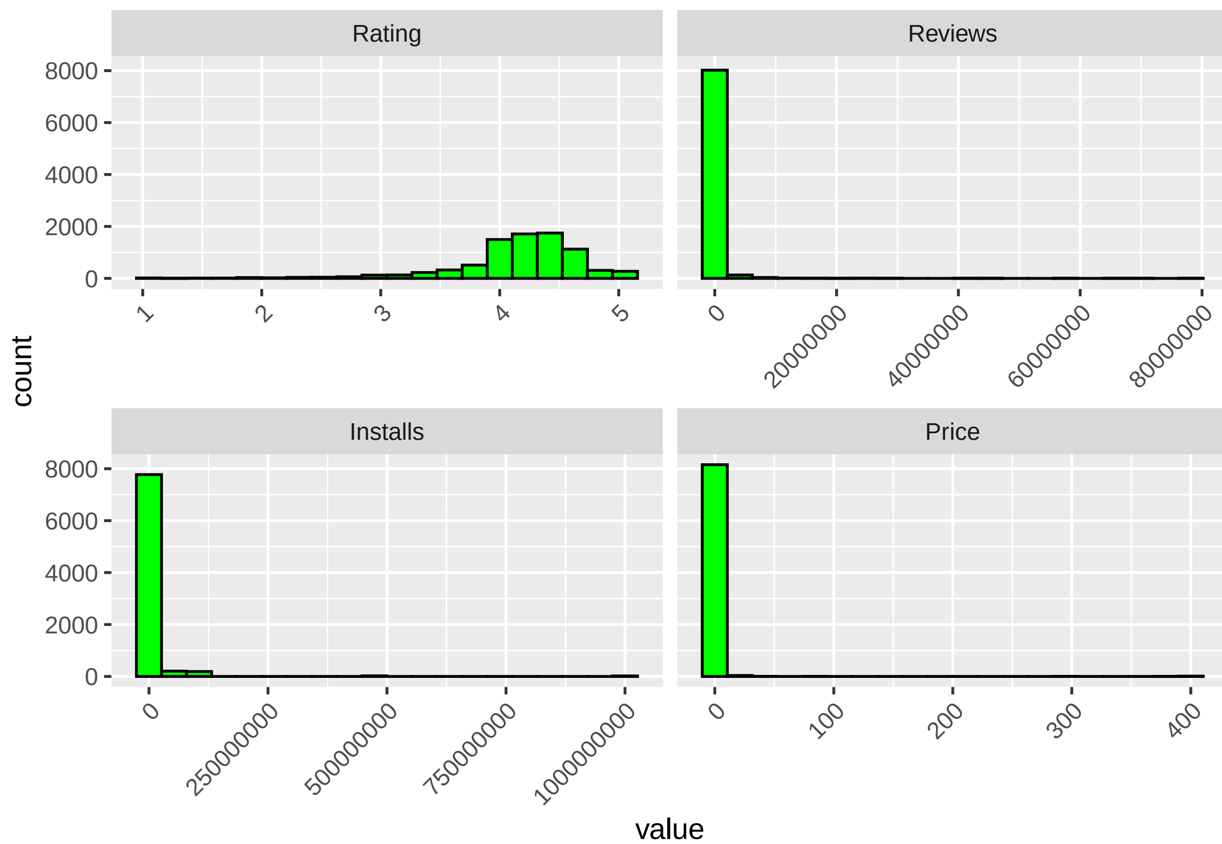
Show Summary of App Reviews

```r
summary(appData$Reviews)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        1      127     3031   255124    44044 78158306
```

Preliminary Data Visualization

```r
library(ggplot2)
library(reshape2)
ggplot(data = melt(appData), mapping = aes(x = value)) + geom_histogram(bins=20, color = "black", fill =
```

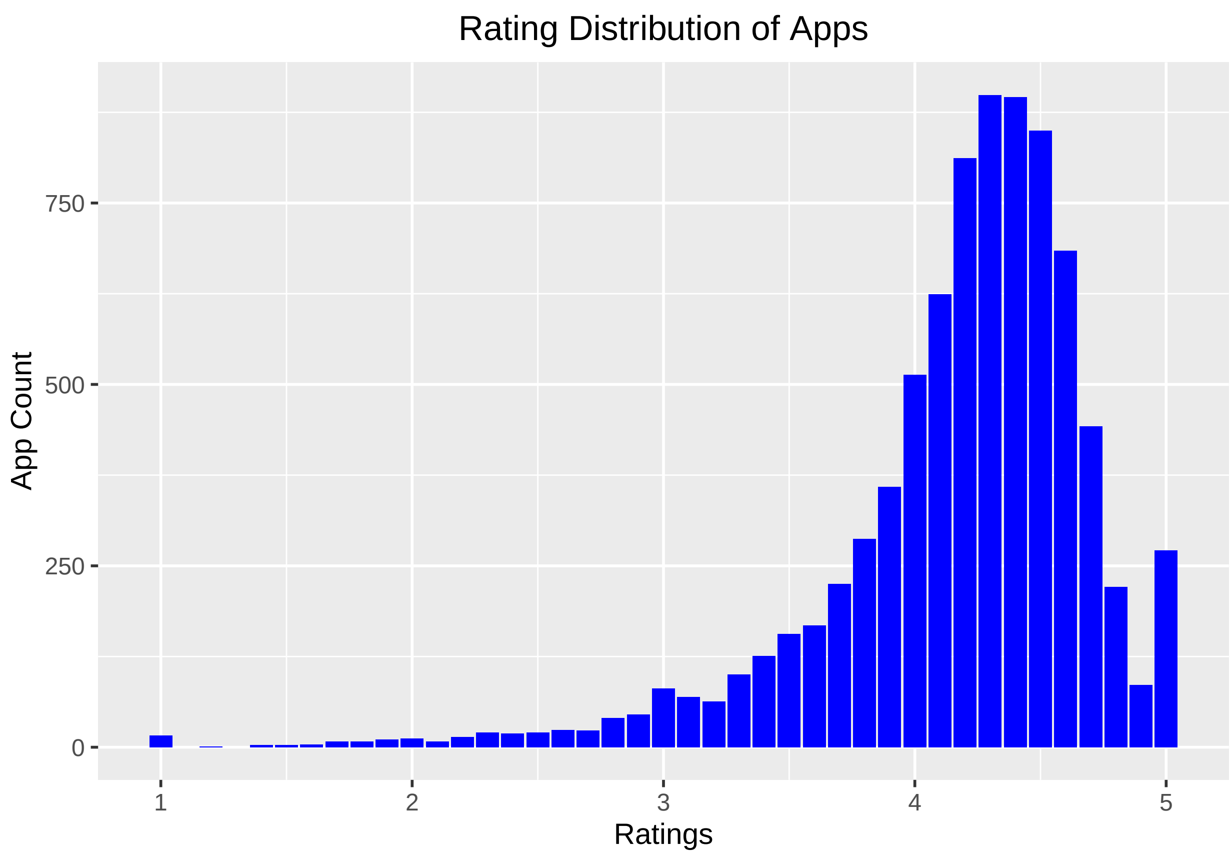


Average Rating of all Apps

```
mr <- mean(appData$Rating)
mr
```

```
## [1] 4.173
```

```
mrPlot <- ggplot(appData, aes(x=Rating)) +
  geom_bar(fill = "blue")  +
  ggtitle("Rating Distribution of Apps") +
  xlab("Ratings") + ylab("App Count") +
  theme(plot.title = element_text(hjust = 0.5))
mrPlot
```

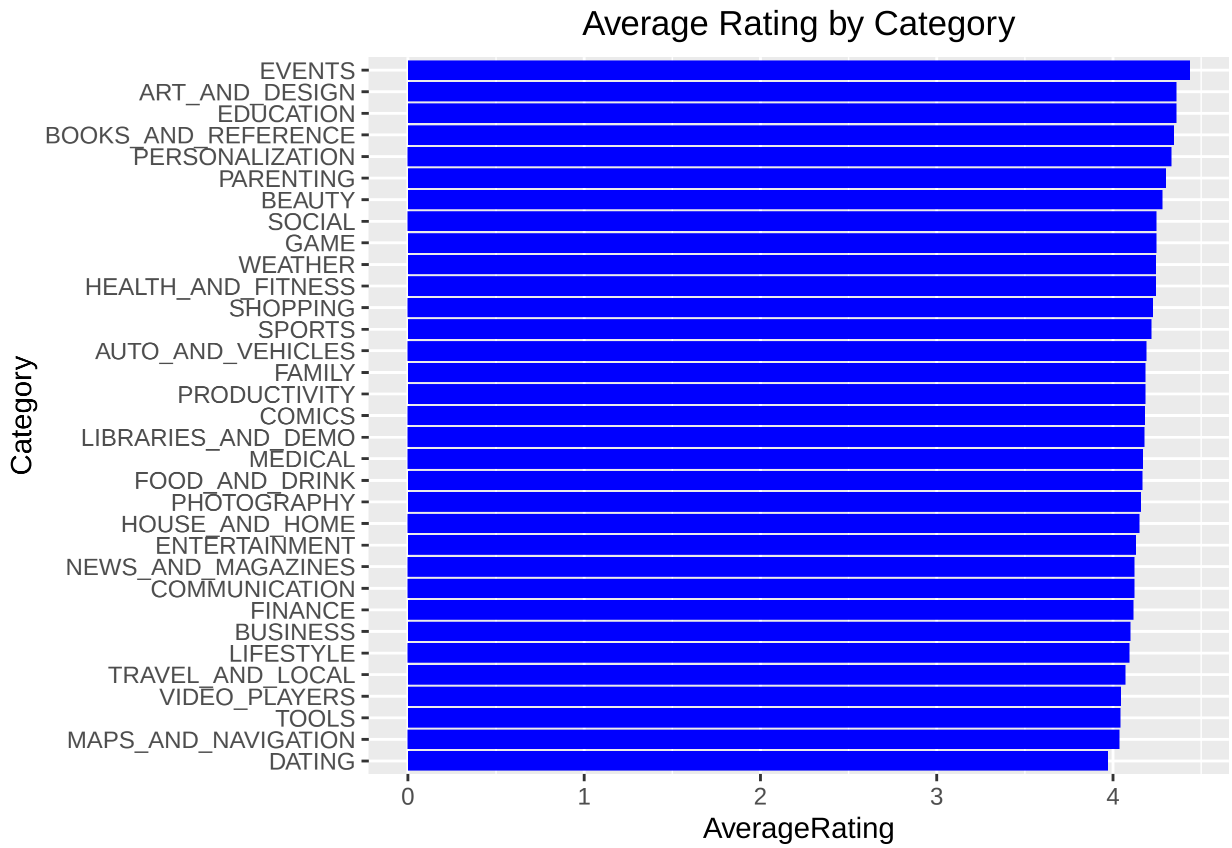


Average Rating by Category

```
mr1 <- data.frame(tapply(appData$Rating, appData$Category, mean))
mr1 <- cbind(rownames(mr1), data.frame(mr1, row.names = NULL))
colnames(mr1) <- c("Category", "AverageRating")


mrPlot1 <- ggplot(mr1, aes(x=reorder(Category,AverageRating), y = AverageRating)) +
  geom_bar(stat="identity", fill = "blue") +
  ggtitle("Average Rating by Category") + xlab("Category") +
  coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))

maxCatRating <- mr1[which.max(mr1$AverageRating),]

paste0("Category with Maximum Average Rating: ", maxCatRating$Category, "   Average Rating: ", maxCatRa
```

```
## [1] "Category with Maximum Average Rating: EVENTS   Average Rating: 4.43555555555556"
```

```
mrPlot1
```

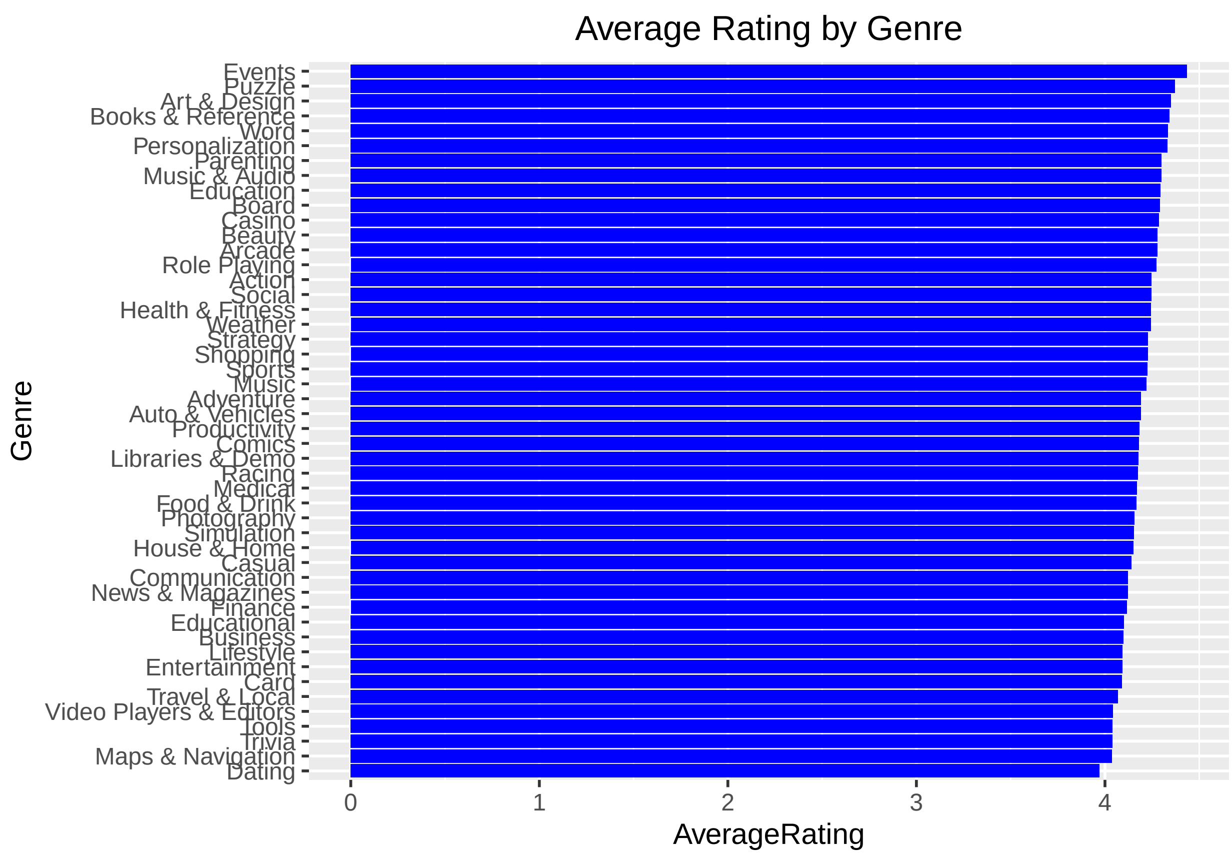


Average Rating by Genre

```
mr2 <- data.frame(tapply(appData$Rating, appData$Genre, mean))
mr2 <- cbind(rownames(mr2), data.frame(mr2, row.names = NULL))
colnames(mr2) <- c("Genre", "AverageRating")


mrPlot2 <- ggplot(mr2, aes(x=reorder(Genre,AverageRating), y = AverageRating)) +
  geom_bar(stat="identity", fill = "blue") +
  ggtitle("Average Rating by Genre") +
  xlab("Genre") +
  coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))

maxGenreRating <- mr2[which.max(mr2$AverageRating),]

paste0("Genre with Maximum Average Rating: ", maxGenreRating$Genre, "   Average Rating: ", maxGenreRati
```

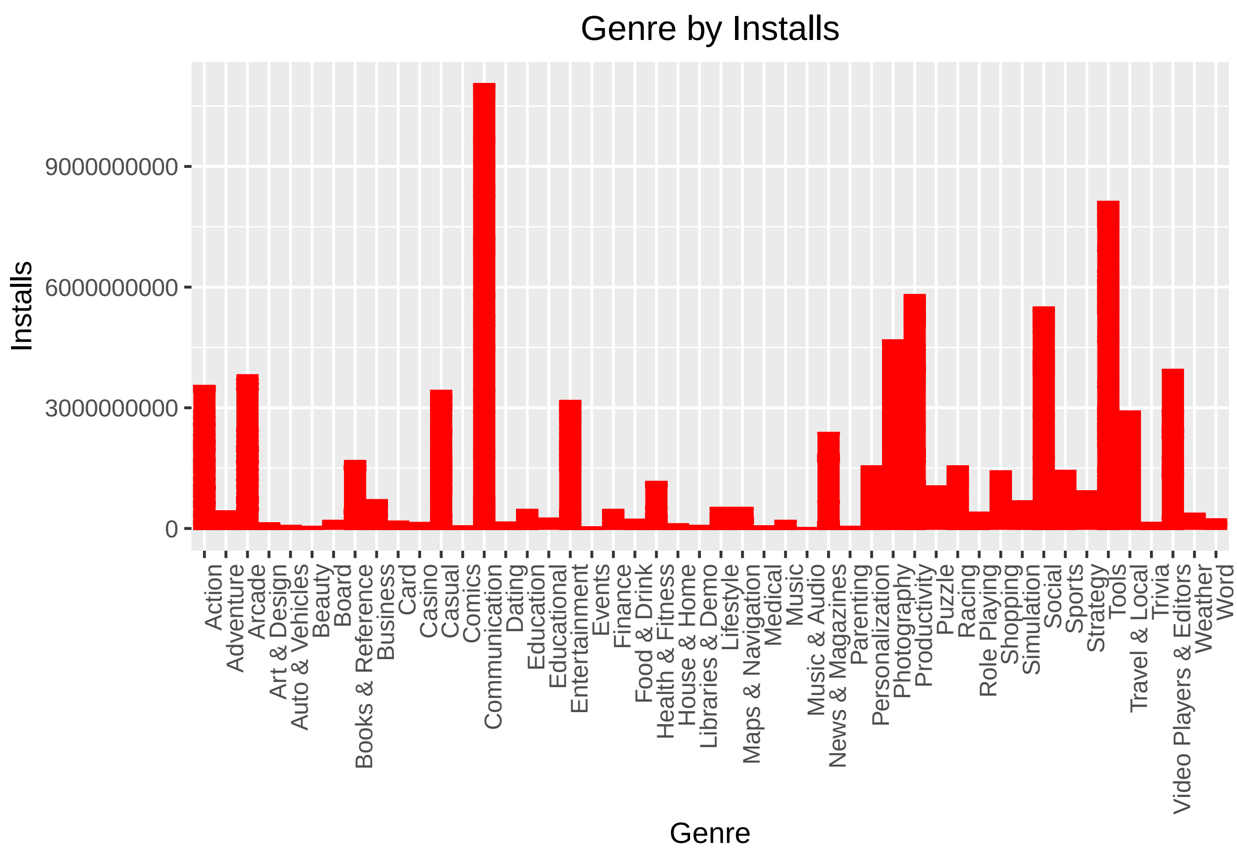
```
## [1] "Genre with Maximum Average Rating: Events   Average Rating: 4.43555555555556"
```

```
mrPlot2
```



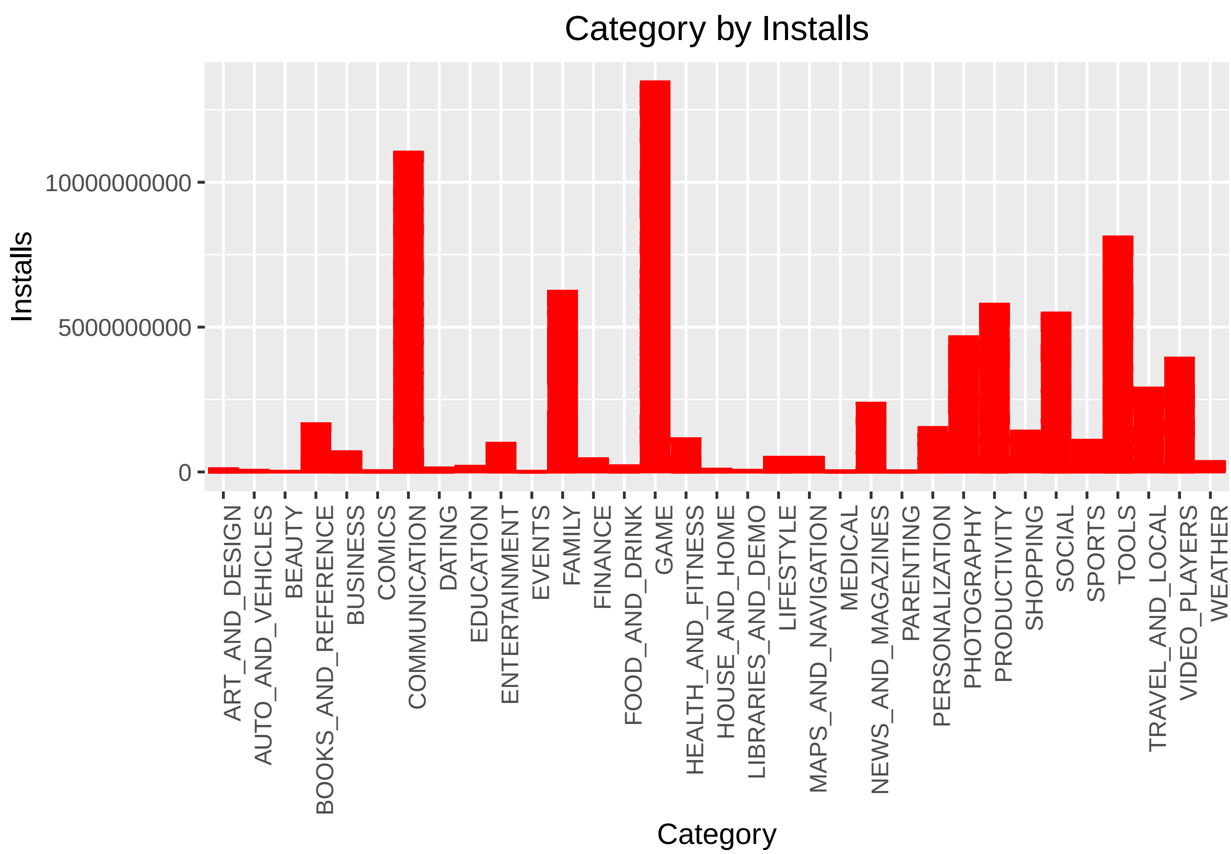
Genre by Installs

```r
ggplot(appData, aes(x=Genre, y = Installs)) +
  geom_bar(stat="identity", color = "red", fill = "red") +
  ggtitle("Genre by Installs") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))
```



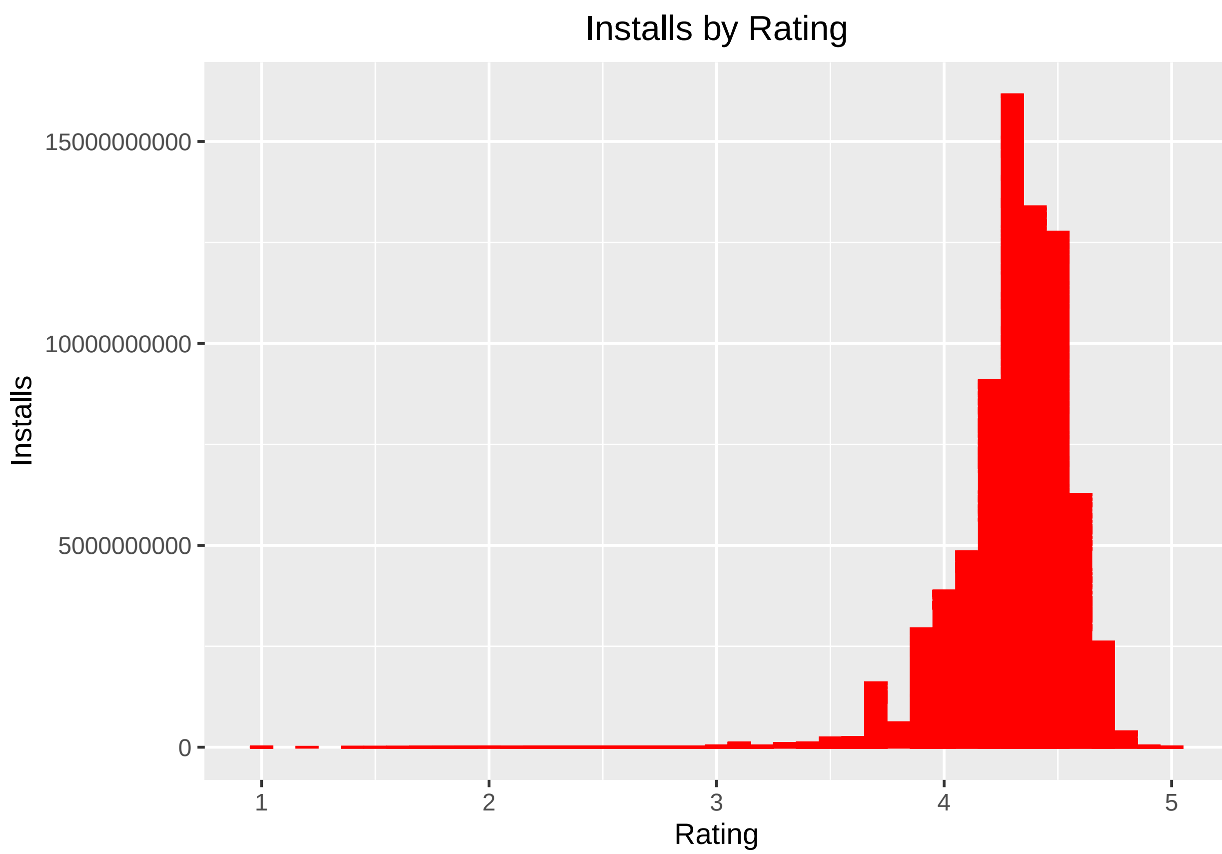
Category by Installs

```r
ggplot(appData, aes(x=Category, y = Installs)) +
  geom_bar(stat="identity", color = "red", fill = "red") +
  ggtitle("Category by Installs") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))
```



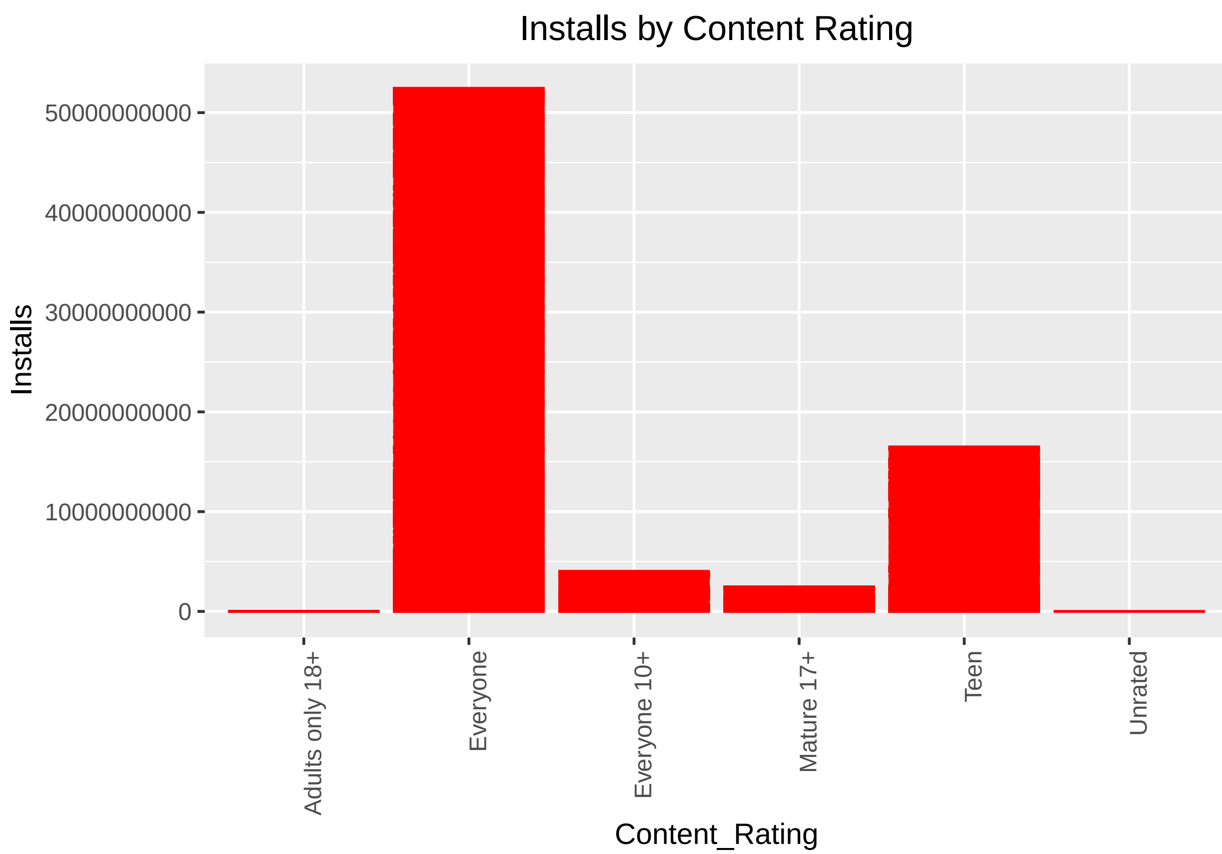
Installs by Ratings

```
ggplot(appData, aes(x=Rating, y = Installs)) +
  geom_bar(stat="identity", color = "red", fill = "red") +
  ggtitle("Installs by Rating") +
  theme(plot.title = element_text(hjust = 0.5))
```



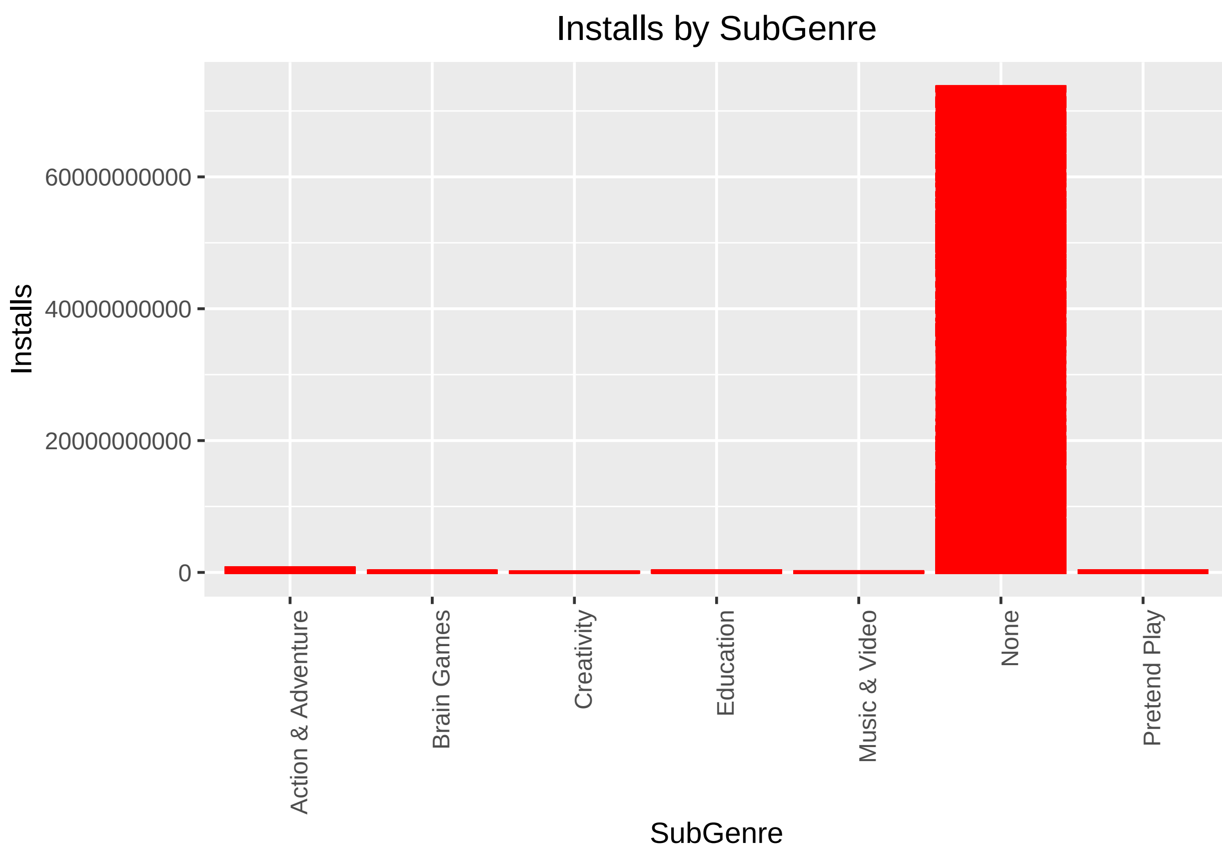
Installs by Content_Rating

```
ggplot(appData, aes(x=Content_Rating, y = Installs)) +
  geom_bar(stat="identity", color = "red", fill = "red") +
  ggtitle("Installs by Content Rating") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))
```



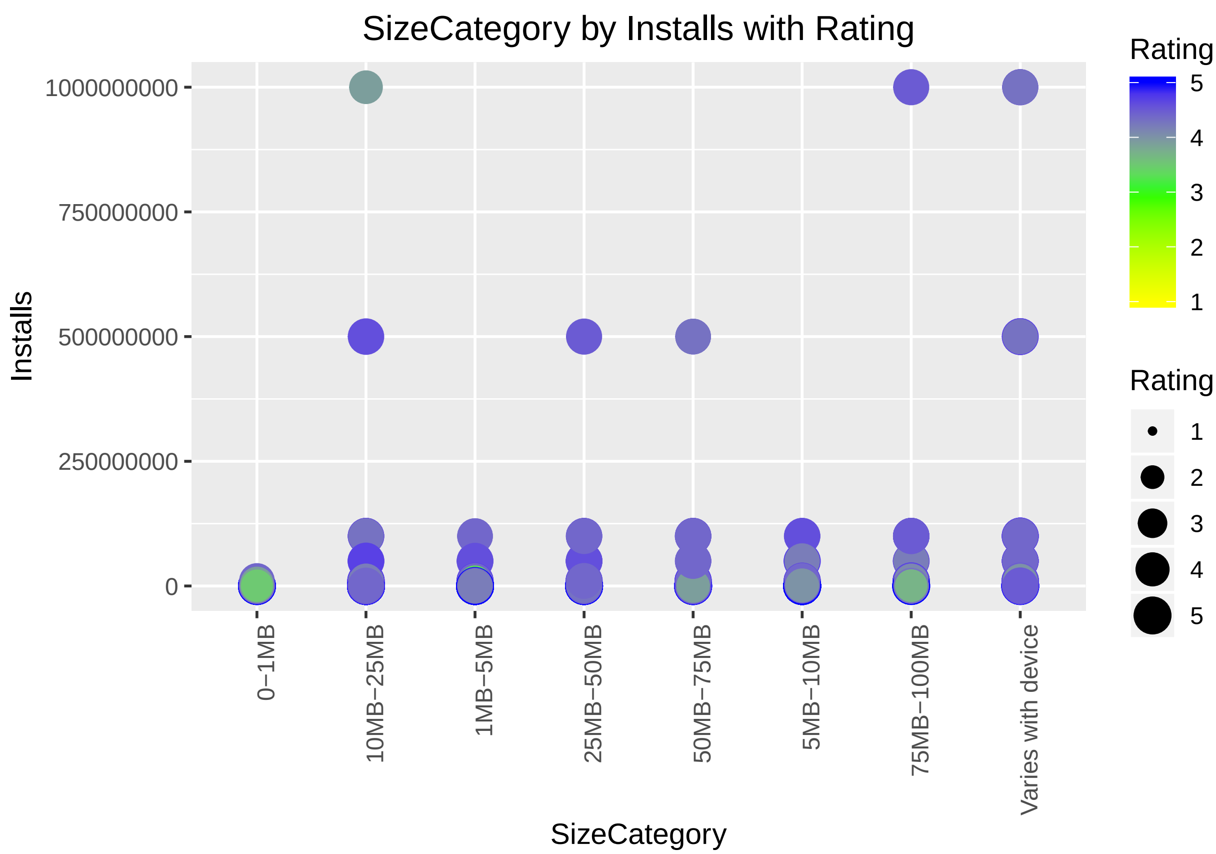
SubGenre impact on Installs

```
ggplot(appData, aes(x=SubGenre, y = Installs)) +
  geom_bar(stat="identity", color = "red", fill = "red") +
  ggtitle("Installs by SubGenre") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))
```



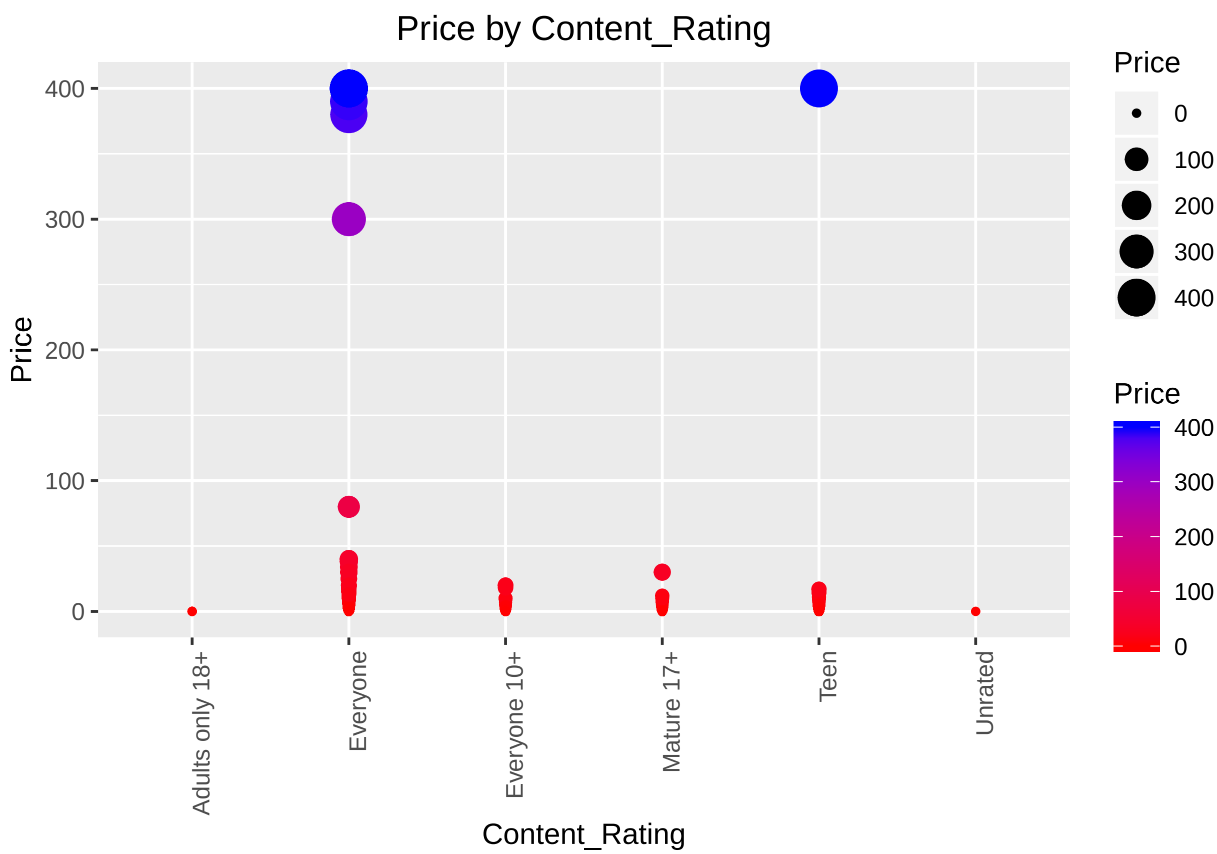
Size Category by Installs with Rating

```
ggplot(rfappData, aes(x=SizeCategory, y = Installs)) +
  geom_point(aes(color = Rating, size = Rating)) +
  scale_color_gradient2(midpoint=3, low="yellow", mid="green", high="blue") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5)) +
  ggtitle("SizeCategory by Installs with Rating")
```



Price by Content_Rating

```
ggplot(appData, aes(x=Content_Rating, y = Price, size = Price, color = Price)) +
  geom_point(stat="identity") +
  scale_color_gradient(low="red", high="blue") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5)) +
  ggtitle("Price by Content_Rating")
```

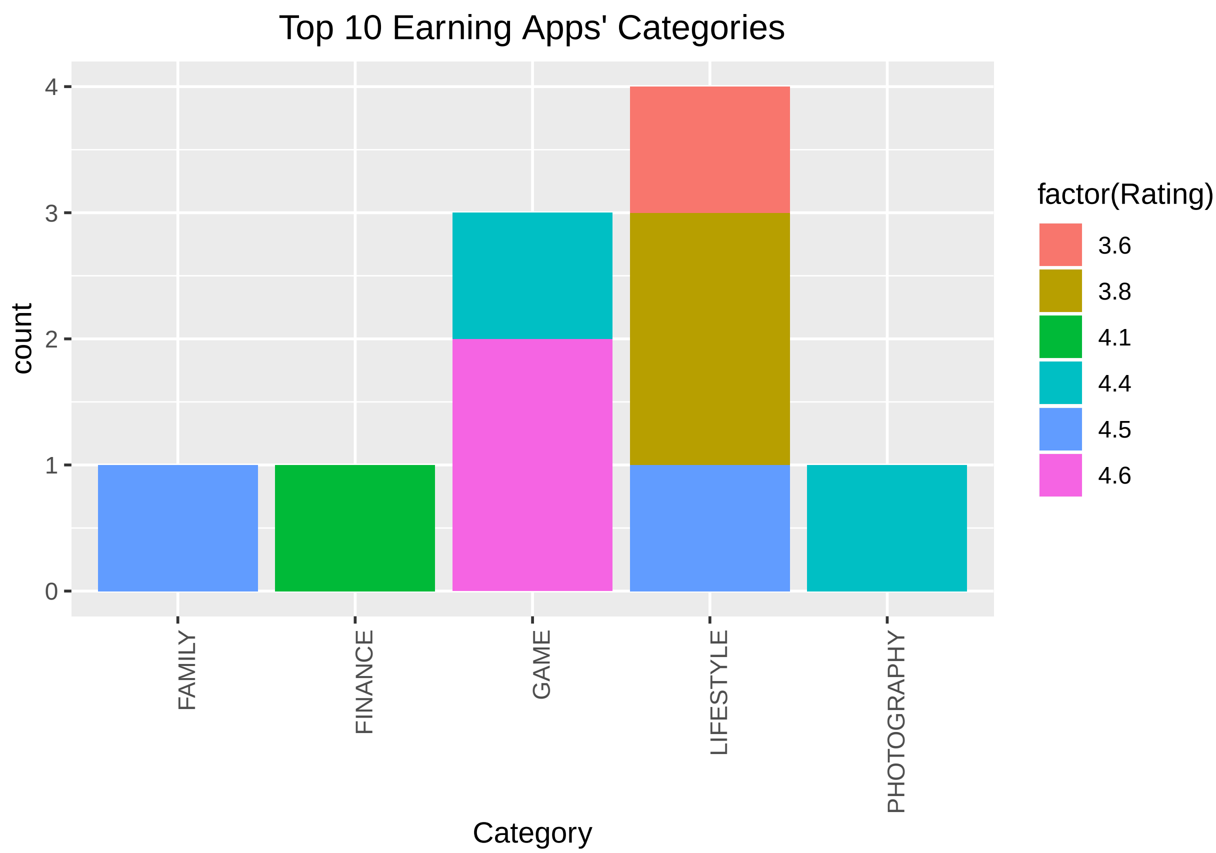


Top 10 Apps' Earning Categories with Price (Excluding in game Purchases)

```
# Calculate the minimum amount earned by game purchases alone:
appData$MinEarned <- appData$Price * appData$Installs

# Identify Top 10 Earners from game purchases/minimum installs alone
topEarners <- appData[order(appData$MinEarned, decreasing=TRUE), ]
head(topEarners, 10)
```

```
## # A tibble: 10 x 12
##    App   Category Rating Reviews Size  Installs Type   Price Content_Rating
##    <fct> <fct>     <dbl>   <dbl> <chr>    <dbl> <fct>  <dbl> <fct>
##  1 Mine~ FAMILY      4.5 2376564 Vari~ 10000000 Paid    6.99 Everyone 10+
##  2 I am~ LIFESTY~    3.8    3547 1843~   100000 Paid  400.   Everyone
##  3 I Am~ FINANCE     4.1    1867 4812~    50000 Paid  400.   Everyone
##  4 Hitm~ GAME        4.6  408292 29696 10000000 Paid    0.99 Mature 17+
##  5 Gran~ GAME        4.4  348962 26624  1000000 Paid    6.99 Mature 17+
##  6 Face~ PHOTOGR~    4.4   49553 49152  1000000 Paid    5.99 Everyone
##  7 Slee~ LIFESTY~    4.5   23966 872    1000000 Paid    5.99 Everyone
##  8 DraS~ GAME        4.6   87766 12288  1000000 Paid    4.99 Everyone
##  9 I'm ~ LIFESTY~    3.6     275 7475~    10000 Paid  400    Everyone
## 10 ðŸ'Ž~ LIFESTY~    3.8     718 26624    10000 Paid  400.   Everyone
## # ... with 3 more variables: Genre <chr>, SubGenre <chr>, MinEarned <dbl>
```

```r
#Visualization
ggplot(head(topEarners, 10), aes(x=Category, fill=factor(Rating))) +
  geom_bar() +
  ggtitle("Top 10 Earning Apps' Categories") +
  theme(axis.text.x = element_text(angle=90, hjust=1), plot.title = element_text(hjust = 0.5))
```



Correlation checks:

```r
#install.packages("devtools")
#install.packages("ggpubr")
library(devtools)
library(ggpubr)

# Correlation between Ratings and Installs
riCorr <- data.frame(appData$Rating, appData$Installs)
riCorrTest <- cor.test(riCorr$appData.Rating, riCorr$appData.Installs, method="pearson")
riCorrTest
```

```
##
##  Pearson's product-moment correlation
##
## data:  riCorr$appData.Rating and riCorr$appData.Installs
## t = 3.6, df = 8209, p-value = 0.0003
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.01860 0.06179
## sample estimates:
##     cor
## 0.04022
```

Installs and Rating are not correlative

Average Size of an application (where size does not vary by device):

```
ap2 <- appData
ap2$Size <- as.numeric(ap2$Size)
ap2 <- na.omit(ap2)
avgSize <- mean(ap2$Size)
paste0("Average Size of an application: ", avgSize, " KB ")
```

```
## [1] "Average Size of an application: 22264.9281392045 KB "
```

Modeling for Random Forest and Support Vector Machine

Prep for randomForest and Run

```
#Remove Size
rfappData <- rfappData[,-4]

# Convert varibles into factor for rfappData
cols <- c("Category","Type","Genre","SubGenre","SizeCategory")
for (i in cols){
  rfappData[,i]<-as.factor(rfappData[,i])
}

# Convert varibles into numeric for rfappData
cols <- c("Rating","Reviews","Installs","Price")
for (i in cols){
  rfappData[,i]<-as.numeric(rfappData[,i])
}

str(rfappData)
```

```
## 'data.frame':    8211 obs. of  10 variables:
##  $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 6 30 7 12 29 30 30 24 24 24 ...
##  $ Rating        : num  3.5 4.5 4.7 3.6 3.2 3.9 4.2 4 4.5 4.4 ...
##  $ Reviews       : num  115 259 573 21433 4 ...
##  $ Installs      : num  10000 10000 10000 1000000 100 500000 1000000 100 1000 5000 ...
##  $ Type          : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 2 1 1 ...
##  $ Price         : num  0 0 0 0 0 0 0 0.99 0 0 ...
##  $ Content_Rating: Factor w/ 6 levels "Adults only 18+",..: 4 2 4 4 2 2 2 2 2 2 ...
```

```
##  $ Genre         : Factor w/ 48 levels "Action","Adventure",..: 13 43 14 18 41 43 43 32 32 32 ...
##  $ SubGenre      : Factor w/ 7 levels "Action & Adventure",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ SizeCategory  : Factor w/ 8 levels "0-1MB","10MB-25MB",..: 6 1 5 2 6 1 1 2 2 6 ...
```

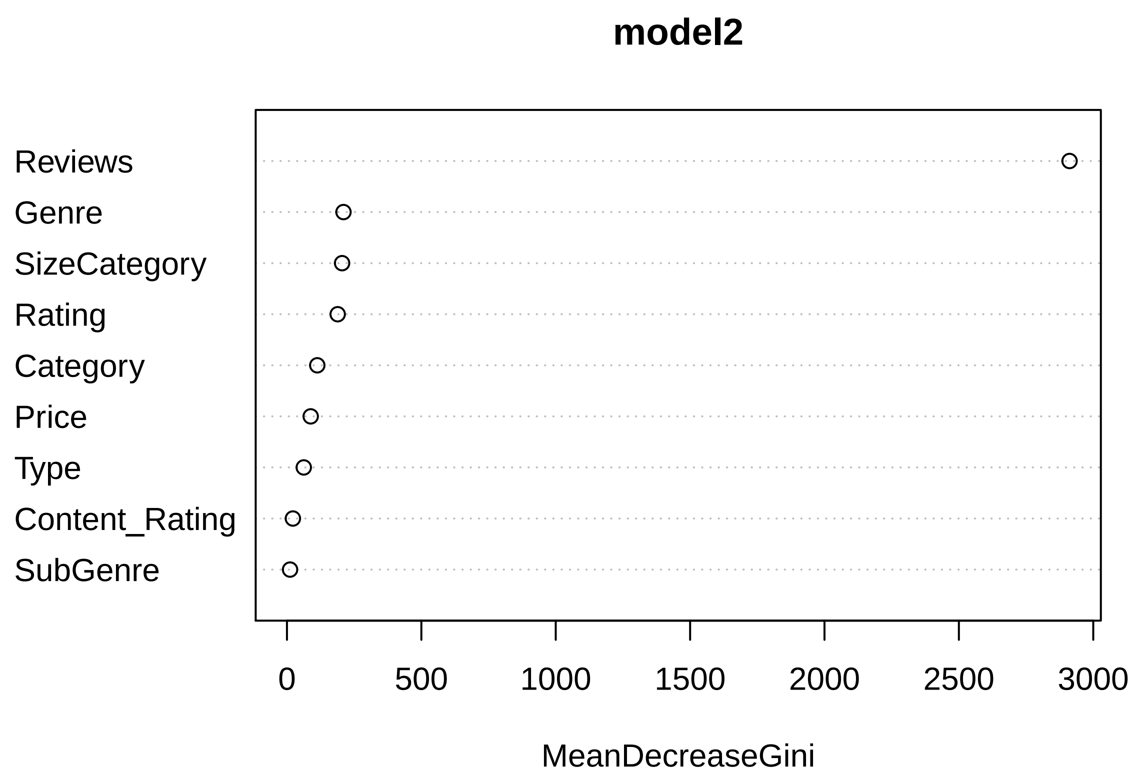
```r
#Create Random Forest model splitting categories into >= 100000 installs and below
library(randomForest)
model2 <- randomForest(rfappData[,-4], as.factor(rfappData[,4] >= 100000))
model2
```

```
##
## Call:
##  randomForest(x = rfappData[, -4], y = as.factor(rfappData[, 4] >=      100000))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.3%
## Confusion matrix:
##       FALSE TRUE class.error
## FALSE  2942  262     0.08177
## TRUE    173 4834     0.03455
```

```r
importance(model2)
```

```
##                  MeanDecreaseGini
## Category                   112.73
## Rating                     188.68
## Reviews                   2911.82
## Type                        62.42
## Price                       88.23
## Content_Rating              21.97
## Genre                      210.24
## SubGenre                    11.48
## SizeCategory               205.06
```

```r
varImpPlot(model2)
```



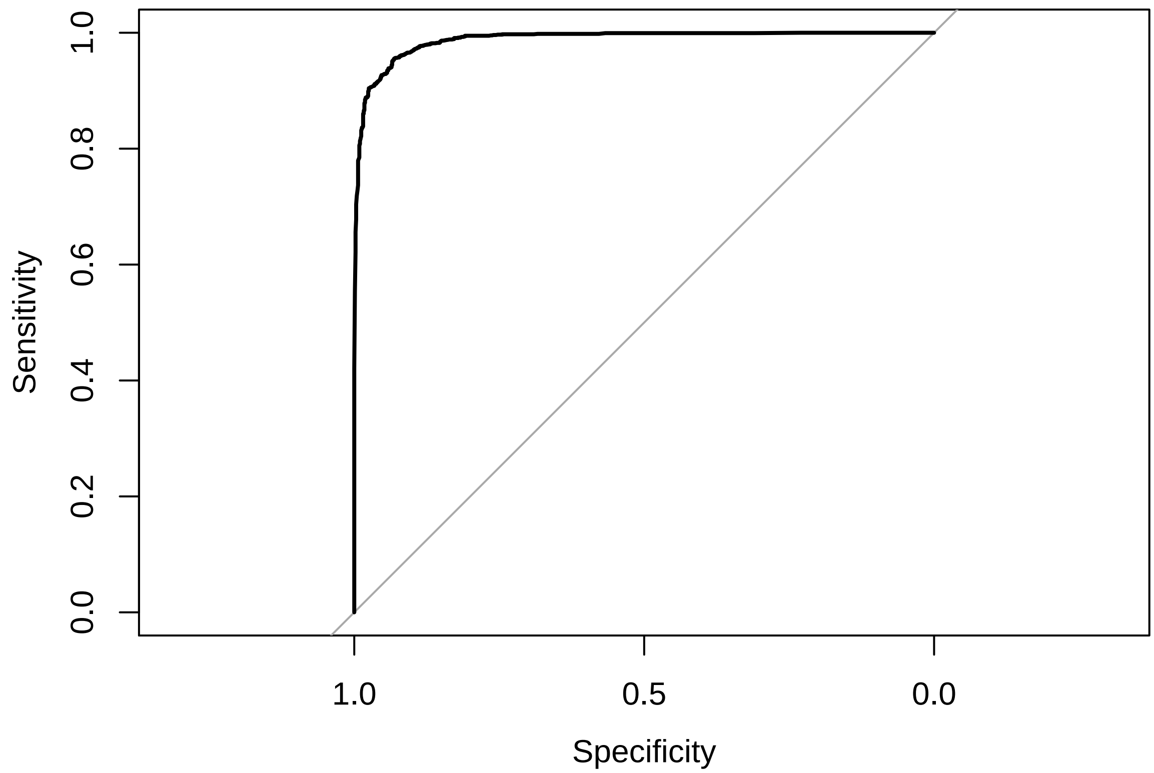
We see the highest ranking independent variables are Reviews, Genre, SizeCategory and then Rating

```
# Set varibles to reflect lessons learned above
num_exmps = nrow(rfappData)
L = replace(integer(num_exmps), rfappData[,4]>=100000, 1)
M <- rfappData[,-4]

# Use Cross validation to build model
train_idx <- sample(c(1:num_exmps), size = num_exmps * 0.7, replace = FALSE)
model2 <- randomForest(M[train_idx,],as.factor(L[train_idx]))
model2
```

```
##
## Call:
##  randomForest(x = M[train_idx, ], y = as.factor(L[train_idx]))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 5.34%
## Confusion matrix:
##      0    1 class.error
## 0 2112  175     0.07652
```

```
## 1  132 3328     0.03815
```

```r
# Generate propsoed answers using Cross validation
pred <- predict(model2, M[-train_idx,],type="prob")

# Plot ROC metric
library(pROC)
plot(roc(L[-train_idx], as.numeric(pred[,1])))
```



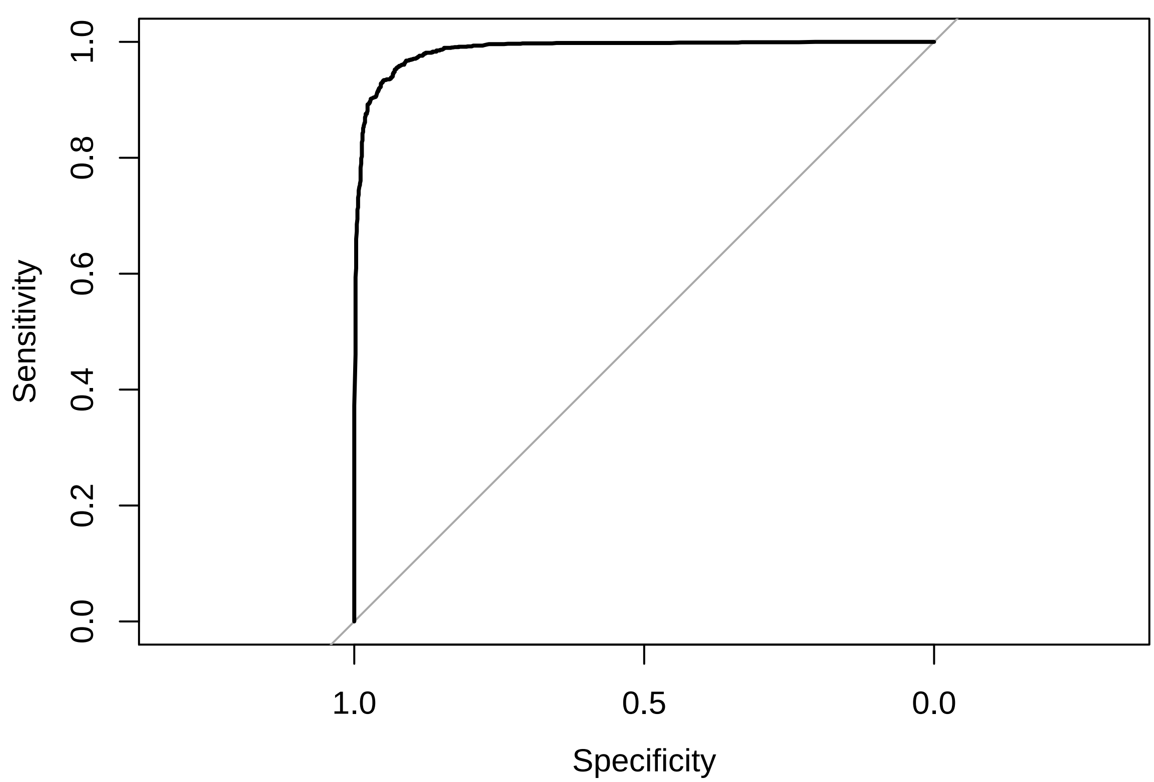

```r
# ROC info https://en.wikipedia.org/wiki/Receiver_operating_characteristic
```

Run K-Fold to for cross validation to try to prevent over fitting

```r
#install.packages("caret")
#install.packages("e1071")
library(caret)
library(e1071)

# Set up cross-validation for k=10 folds
train_Control <- trainControl(method="cv", number=10)
# Train the model with K-Fold cross validation training set
model <- train(M[train_idx,],as.factor(L[train_idx]), trControl=train_Control, method="rf")
print(model)
```

```
## Random Forest
##
## 5747 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5172, 5173, 5173, 5172, 5173, 5172, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##   2     0.9455    0.8859
##   5     0.9434    0.8814
##   9     0.9415    0.8774
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```r
# Generate propsoed answers using Cross validation
pred <- predict(model, M[-train_idx,],type="prob")

# Plot ROC metric
library(pROC)
plot(roc(L[-train_idx], as.numeric(pred[,1])))
```



```r
# ROC info https://en.wikipedia.org/wiki/Receiver_operating_characteristic
# Caret Package Reference: https://topepo.github.io/caret/available-models.html
```

Support Vector Machine

```r
library(kernlab)
library(pROC)

svmappData <- data.frame(rfappData)
svmappData$Installs <- ifelse(test = svmappData$Installs>100000, yes = 1, no = 0)

str(svmappData)
```

```
## 'data.frame':    8211 obs. of  10 variables:
##  $ Category      : Factor w/ 33 levels "ART_AND_DESIGN",..: 6 30 7 12 29 30 30 24 24 24 ...
##  $ Rating        : num  3.5 4.5 4.7 3.6 3.2 3.9 4.2 4 4.5 4.4 ...
##  $ Reviews       : num  115 259 573 21433 4 ...
##  $ Installs      : num  0 0 0 1 0 1 1 0 0 0 ...
##  $ Type          : Factor w/ 2 levels "Free","Paid": 1 1 1 1 1 1 1 2 1 1 ...
##  $ Price         : num  0 0 0 0 0 0 0 0.99 0 0 ...
##  $ Content_Rating: Factor w/ 6 levels "Adults only 18+",..: 4 2 4 4 2 2 2 2 2 2 ...
##  $ Genre         : Factor w/ 48 levels "Action","Adventure",..: 13 43 14 18 41 43 43 32 32 32 ...
##  $ SubGenre      : Factor w/ 7 levels "Action & Adventure",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ SizeCategory  : Factor w/ 8 levels "0-1MB","10MB-25MB",..: 6 1 5 2 6 1 1 2 2 6 ...
```

```r
names(svmappData)
```

```
##  [1] "Category"       "Rating"         "Reviews"        "Installs"
##  [5] "Type"           "Price"          "Content_Rating" "Genre"
##  [9] "SubGenre"       "SizeCategory"
```

```r
# Feature analysis from random forest:
# rating > size > genre > review > androvid > Category > price > type > content > rating >"subgenre"

svmappData <- svmappData[,-8] #remove subgenre

## create function
cPercent <- function(predicted, actual){
 confMatrix<- table(predicted, actual, dnn=c("Prediction","Actual"))
 Result <- (confMatrix[1,1]+confMatrix[2,2])/sum(colSums(confMatrix))*100
 print(confMatrix)
 return(sprintf("Correct Percentage: %1.2f%% ", Result))
}

# create a randomized index
randIndex <- sample(1:nrow(svmappData))

# Calculate the cut point and divide the data set into training set & test set:
cutPoint2_3 <- floor(2*nrow(svmappData)/3)
cutPoint2_3
```

```
## [1] 5474
```

```r
# generate test set and training data sets:
trainData <- svmappData[randIndex[1:cutPoint2_3],]
testData <- svmappData[randIndex[(cutPoint2_3+1):nrow(svmappData)],]

2737+5474
```

```
## [1] 8211
```

```r
# Generate a model based on the training data set:
# model 1 --- Radial Basis kernel "Gaussian"
svmOutput <- ksvm(Installs~., data = trainData, kernel = "rbfdot", kpar="automatic", C=5, cross=3, prob
svmOutput
```

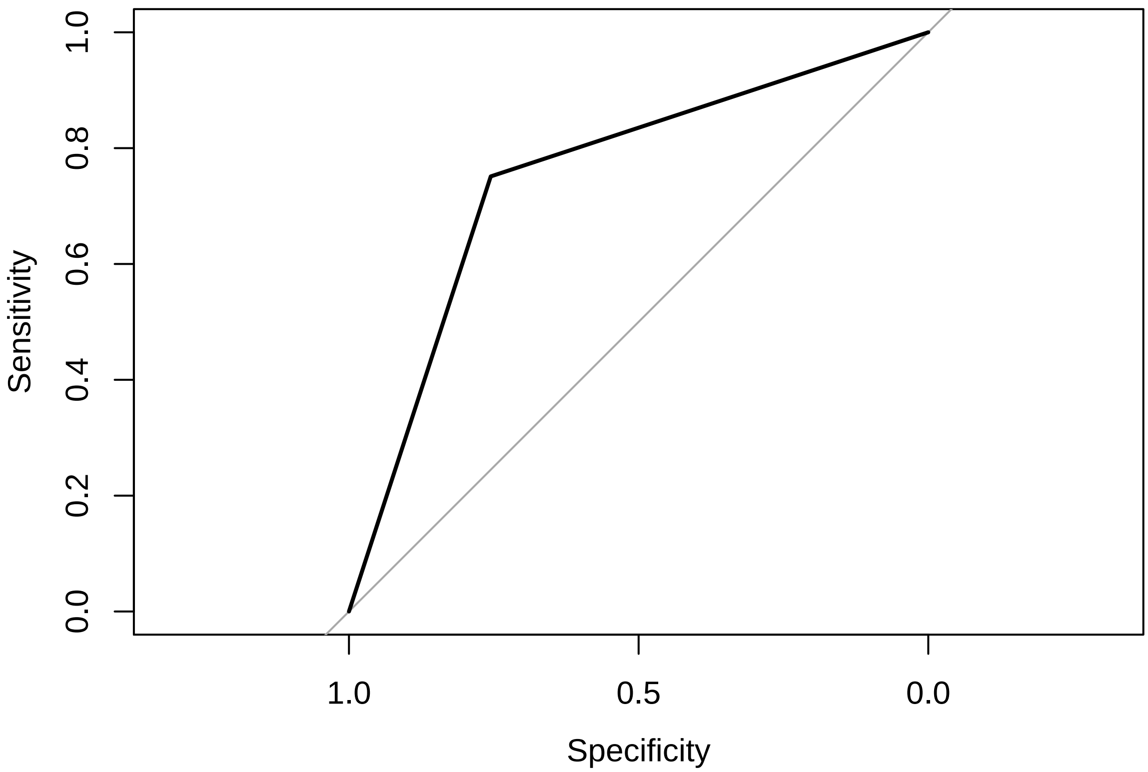
```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 5
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.187216971001782
##
## Number of Support Vectors : 3779
##
## Objective Function Value : -11155
## Training error : 0.585291
## Cross validation error : 0.1885
## Laplace distr. width : 0.4791
```

```r
predSVM <- round(predict(svmOutput,testData))
cPercent(predSVM, testData$Installs)
```

```
##           Actual
## Prediction    0    1
##          0 1123  364
##          1  311  939
```

```
## [1] "Correct Percentage: 75.34% "
```

```r
plot(roc(predSVM, testData$Installs))
```




```r
# model 2 --- Linear kernel
svmOutput2 <- ksvm(Installs~., data = trainData, kernel = "vanilladot", kpar="automatic", C=5, cross=3,
```

```
##  Setting default kernel parameters
```

```r
svmOutput2
```

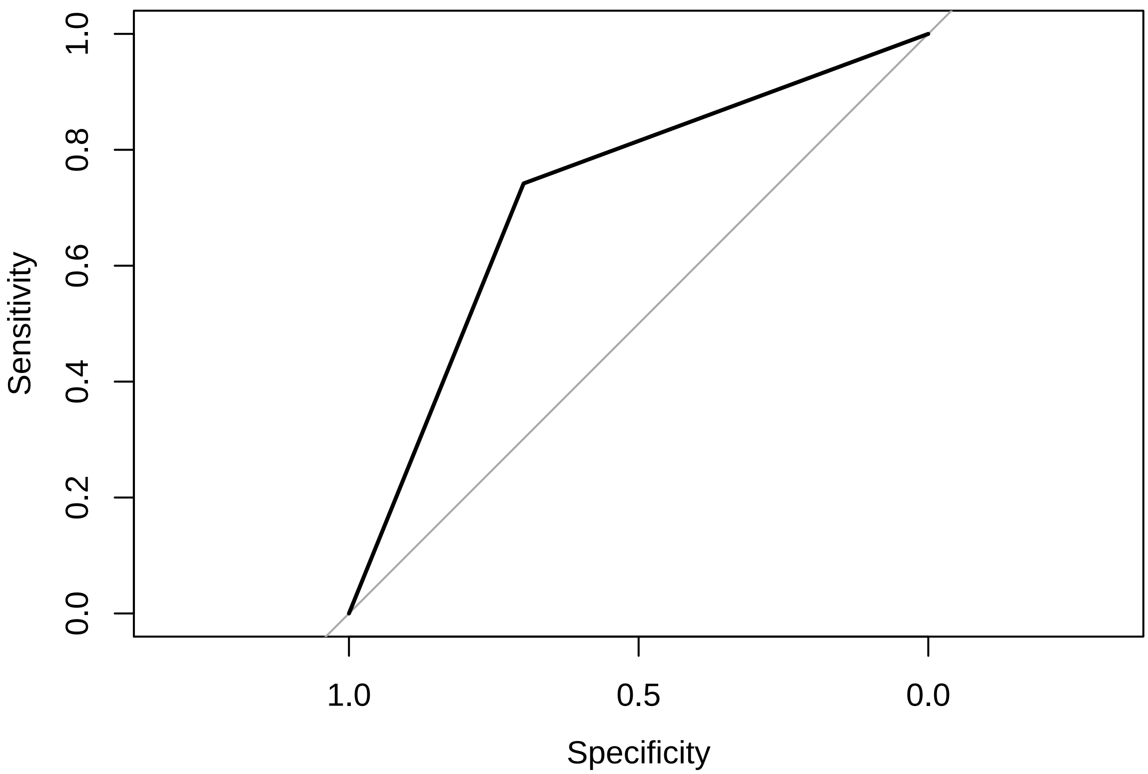
```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 5
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 4593
##
## Objective Function Value : -16582
## Training error : 0.816468
## Cross validation error : 0.2095
## Laplace distr. width : 0.4538
```

```r
predSVM2 <- ifelse(round(predict(svmOutput2,testData))>=1, 1,0)
cPercent(predSVM2, testData$Installs)
```

```
##            Actual
## Prediction    0    1
##          0 1154  498
##          1  280  805
```

```
## [1] "Correct Percentage: 71.57% "
```

```r
plot(roc(predSVM2, testData$Installs))
```




```r
# model 3 --- Polynomial kernel
svmOutput3 <- ksvm(Installs~., data = trainData, kernel = "polydot", kpar="automatic", C=5, cross=3, pr
```

```
##  Setting default kernel parameters
```

```r
svmOutput3
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 5
##
## Polynomial kernel function.
##  Hyperparameters : degree =  1  scale =  1  offset =  1
##
```

```
## Number of Support Vectors : 4593
##
## Objective Function Value : -16582
## Training error : 0.816489
## Cross validation error : 0.2189
## Laplace distr. width : 0.4575
```

```
predSVM3 <- ifelse(round(predict(svmOutput3,testData))>=1, 1,0)
cPercent(predSVM3, testData$Installs)
```

```
##           Actual
## Prediction    0    1
##          0 1154  498
##          1  280  805
```

```
## [1] "Correct Percentage: 71.57% "
```

```
plot(roc(predSVM3, testData$Installs))
```

Random Forest Model is the best model according to ROC plot and Accuracy Rate