Data Analysis for TechPointX Xbot Digital Assistant

Dante Razo

10/20/2018

Abstract

I've been tasked with analyzing app store data to assess its current state and predict the success of TechPointX's upcoming **Xbot** digital assistant. Due to the popularity of the company's OSXtern operating system, expectations are high for the app. I observed trends in the app store to help the team make the launch as impactful as possible. I used *RMarkdown*, *RStudio*, and the *DataExplorer* library for R to create this report.

Data Preprocessing

Importing Data & Packages

First, I imported the data. The *.csv* has headers and entries are separated by commas, so I used `read.csv()`'s default settings.

```
require(DataExplorer)  # package that provides additional visualization tools for data analysis
```

```
## Loading required package: DataExplorer
```

```
appStore <- read.csv(file = "AppStoreAssessmentDataScience.csv")
appStore.og <- appStore  # store copy of the original before preprocessing
```

Preliminary Observations

Before any preprocessing is done, we can observe that *appStore* contains 7197 objects of 9-dimensions. There are no missing values, so imputation is not necessary.

```
dim(appStore)
```

```
## [1] 7197    9
```

```
sum(is.na(appStore))  # no missing values in dataset
```

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

The first column of *appStore* is in numerical order, but only the first 18 entries match the column number. It's unknown why numbers are skipped over. This vector has a 99% correlation with column numbers, so I removed it from the dataset. It is stored under a new name in case it can be used later.

```
sum(appStore[1] == seq(1, nrow(appStore)))  # checks if entry equals row number; 18 matches
```

```
## [1] 18
```

```
cor(appStore$X, seq(1, nrow(appStore)))  # computes correlation between two vectors
```

```
## [1] 0.9936812
```

```
appStore.V1 <- appStore[1]  # save first column as new variable
appStore <- appStore[, 2:9]  # remove first column from dataset
```

The *app_content_rating* column contains integers with a "+" character appended to the end. I removed the pluses and converted the resulting strings to integers. This will allow me to take averages and analyze this vector if I need to.

```
appStore$app_content_rating <- as.numeric(gsub("\\+", "", appStore$app_content_rating))
```

It was at this point that I remembered to check for other types of missing values (such as zeroes where they don't make sense). Using the `summary()` function revealed that the last column of the dataset (*app_total_supported_langs*) contained 0's. It doesn't make sense for an app to have 0 supported languages, so these are effectively missing values. Due to the small number of affected entries, I elected to simply remove them.

```
summary(appStore$app_total_supported_langs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   1.000   5.435   8.000  75.000
```

```
sum(appStore$app_total_supported_langs == 0)
```

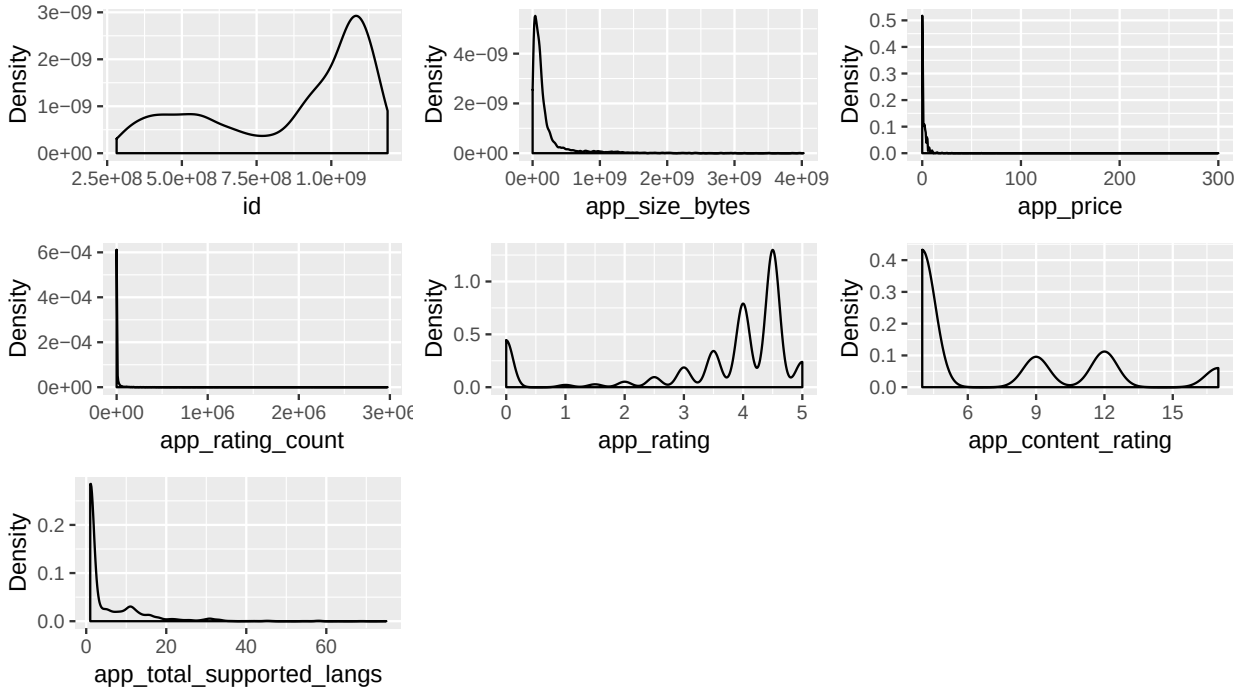
```
## [1] 41
```

```
appStore$app_total_supported_langs[appStore$app_total_supported_langs == 0] <- NA  # replace 0's
appStore <- na.omit(appStore)
```

Data Analysis

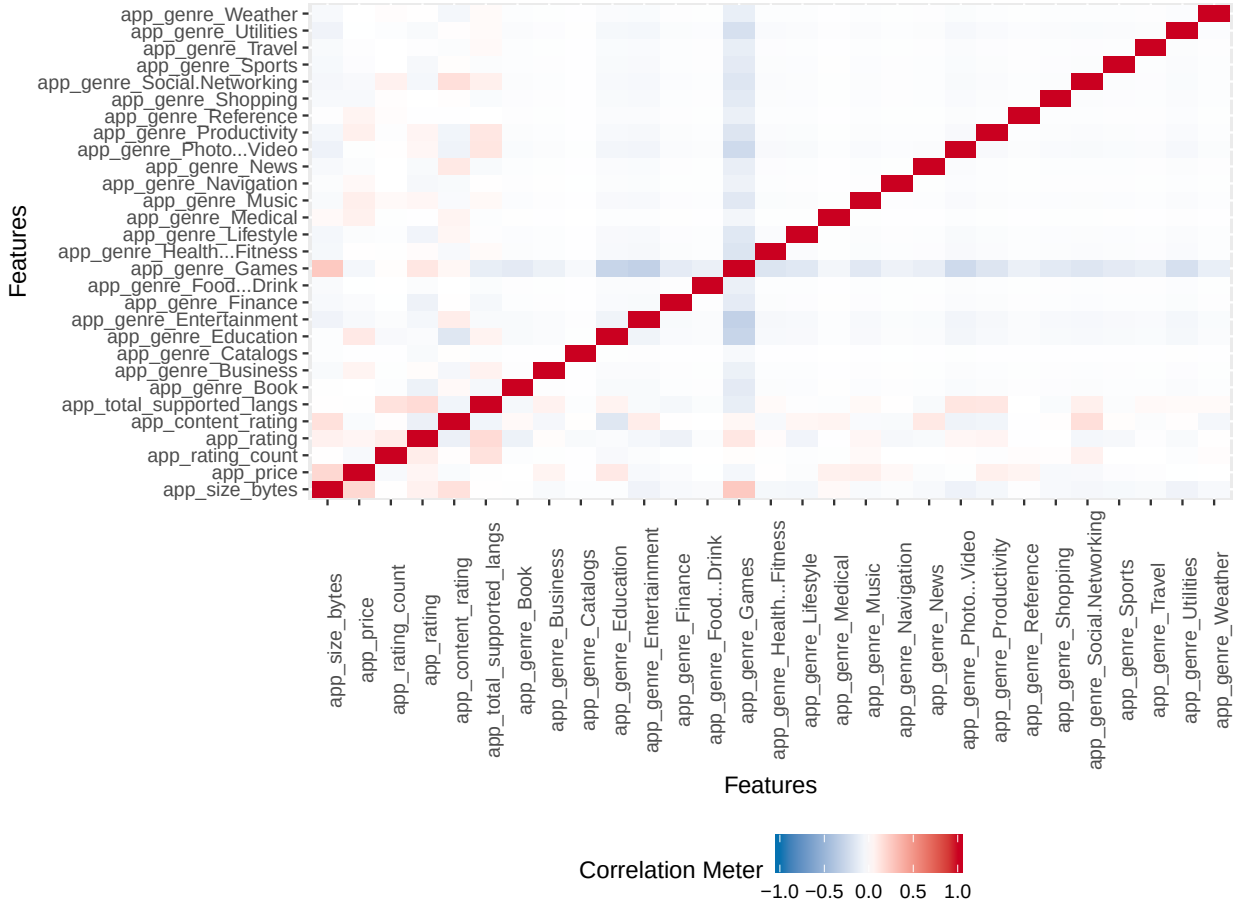
Now that the data has been processed, I can begin to make sense of it. I began by making density plots of every column in the dataset. The *DataExplorer* library makes it easy to view all plots at once.

```
plot_density(appStore)  # function from DataExplorer library
```



I immediately noticed that `appStore$id` is a a left-skewed bimodal distribution. The ID is simply a number and won't be useful in identifying trends in the App Store, so I moved on to other columns. The majority of apps on the market are less than 1000MB ($10^9 = 1,000,000,000$ bytes). Depending on how OSXtern and other supported platforms defines a gigabyte, you could say that most apps are less than 1GB as well.

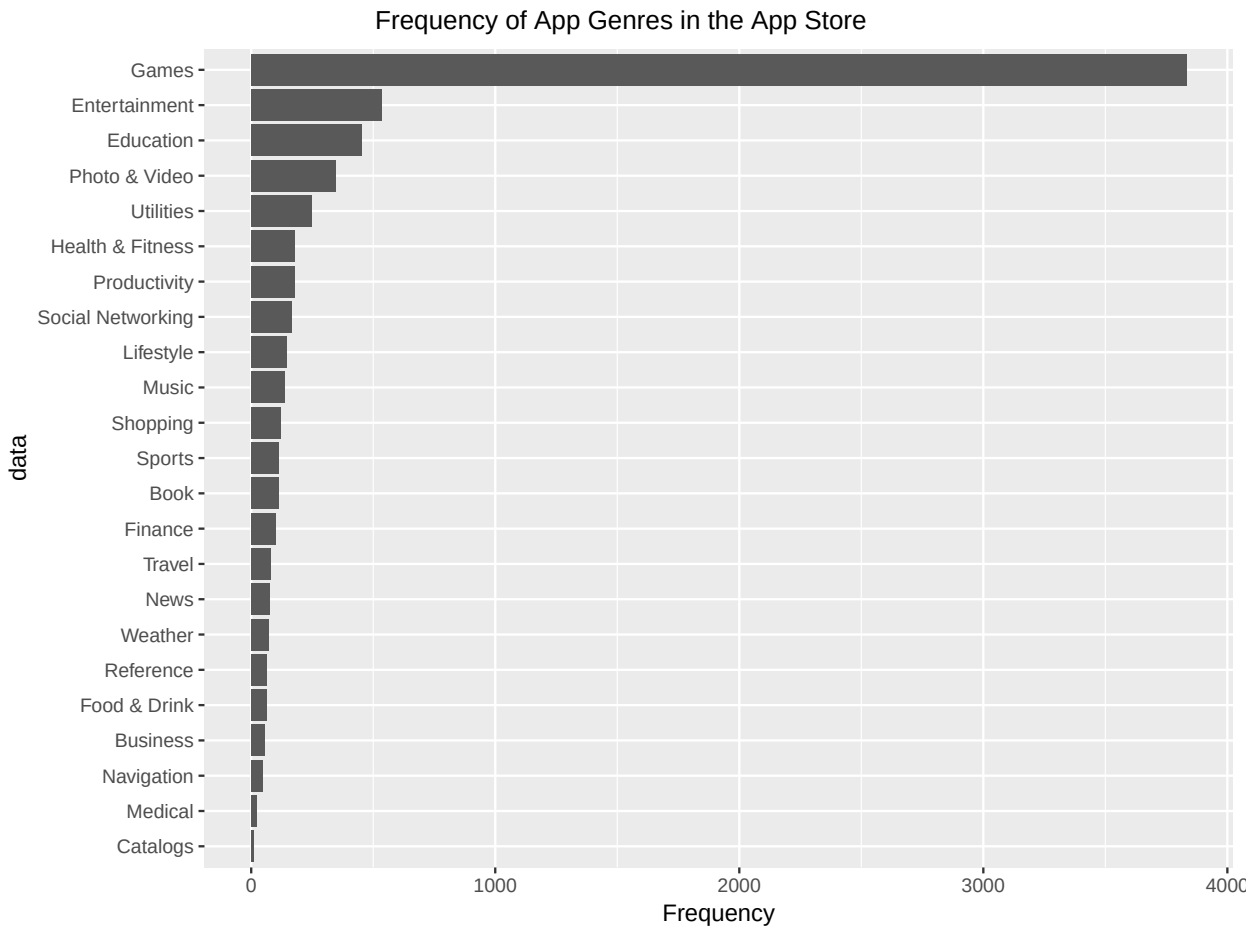
```
plot_correlation(appStore[, 2:ncol(appStore)], maxcat = 24)  # function from DataExplorer library
```



DataExplorer's `plot_correlation()` function produced a nice correlation graph of every feature. I didn't learn anything new from it, but it confirmed that no high-correlation vector pairs remained. The only feature not pictured is `app_genre` because each entry is a string. It contains nominal data; usually, I'd assign numbers to each value before working with them, but *DataExplorer* has a visualization solution that negates the need to first quantify the genres.

Surprisingly, app price and size have a somewhat strong correlation with a correlation coefficient (r) of 0.18. $r^2 = 0.0324$, which means that only 3.24% of the variation is explained by r. Another strange observation is that $r = 0.14$ for `app_content_rating` and `app_size_bytes`. Less surprising and somewhat strong correlations include: `app_rating_count` vs. `app_total_supported_langs` (0.14) and `app_rating` vs. `app_total_supported_langs` (0.17). Finally, games tend to be bigger in size (bytes) and have higher ratings than other apps.

```
plot_bar(appStore$app_genre, title = "Frequency of App Genres in the App Store")
```



The 'Games' category stands out as an outlier. The number of games (3832) is over 7 times greater than the number of 'Entertainment' apps (534). I created a separate dataset that contains everything but apps labeled 'Games' in case the outlier affects future observations.

```
numGames <- sum(grepl("Games", appStore$app_genre))  # most common genre
numEntertainment <- sum(grepl("Entertainment", appStore$app_genre))  # second most common genre
numGames/numEntertainment  # ratio (7x increase)
```

```
## [1] 7.17603
```

```
appStore.noGames <- appStore  # create copy of dataset
appStore.noGames$app_genre[grepl("Games", appStore.noGames$app_genre)] <- NA  # replace games with NA
appStore.noGames <- na.omit(appStore.noGames)  # remove games (now NA)
```

Xbot is an assistant, so it'd best fit in the 'Utilities' category. I compared this category to its nearest competitors below:

```
numPhoto <- sum(grepl("Photo & Video", appStore$app_genre))
numUtil <- sum(grepl("Utilities", appStore$app_genre))
numHealth <- sum(grepl("Health & Fitness", appStore$app_genre))

numUtil  # number of utility apps
```

```
## [1] 248
```

```
numPhoto - numUtil  # distance to upper neighbor
```

```
## [1] 100
```

```
numUtil - numHealth  # distance to lower neighbor
```

```
## [1] 68
```

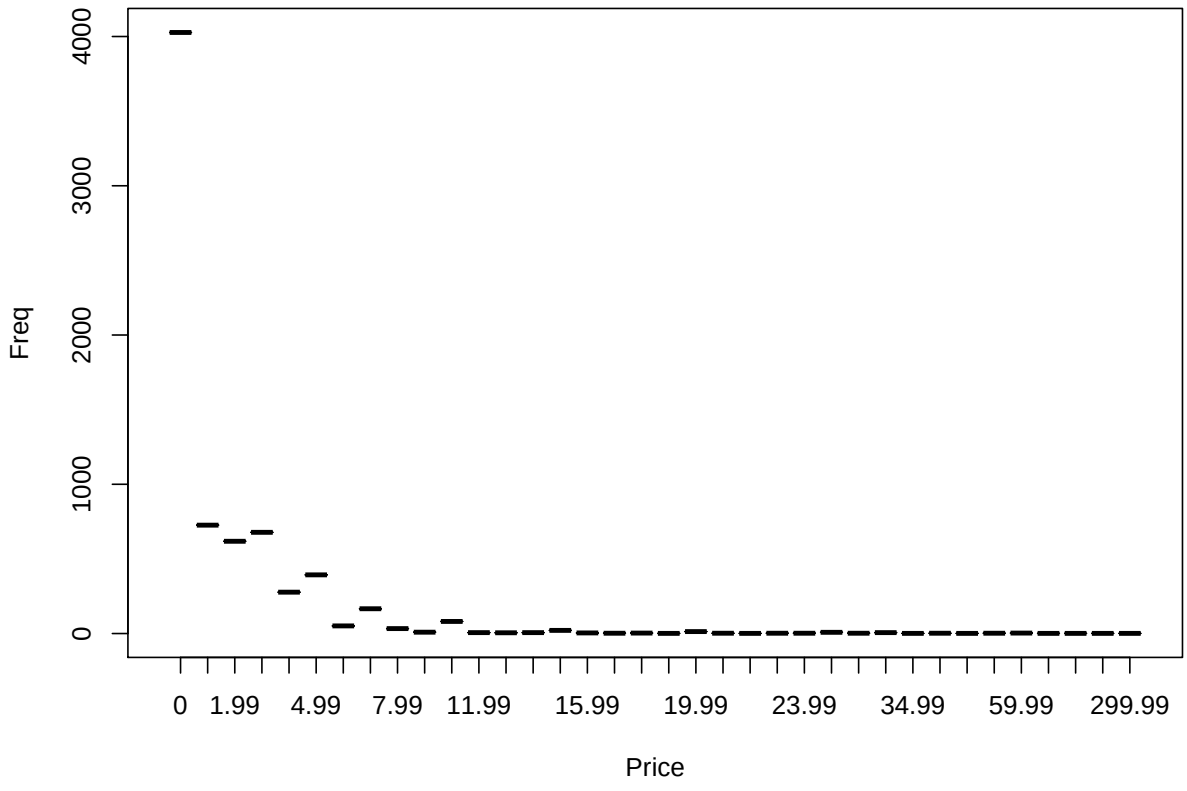
Unlike 'Games', 'Utilities' is reasonably close to its neighbors. It's a popular yet less-saturated genre with only 248 apps. **Xbot** has a higher chance of success than any new game that comes to the app store because it has less competition. If advertised properly, it could very easily top the charts.

Next, I focused on the price of the apps in the dataset (`appStore$app_price`). Unsurprisingly, the majority (56%) were free. I must admit that I'm unsure what to make of this; the naive answer would be to make **Xbot** a free app too to achieve the same accessibility and popularity as the others. People are more likely to download and try a free app than pay for an app they may not enjoy using.

```
sum(appStore$app_price == 0)/nrow(appStore)  # 56% of apps are free
```

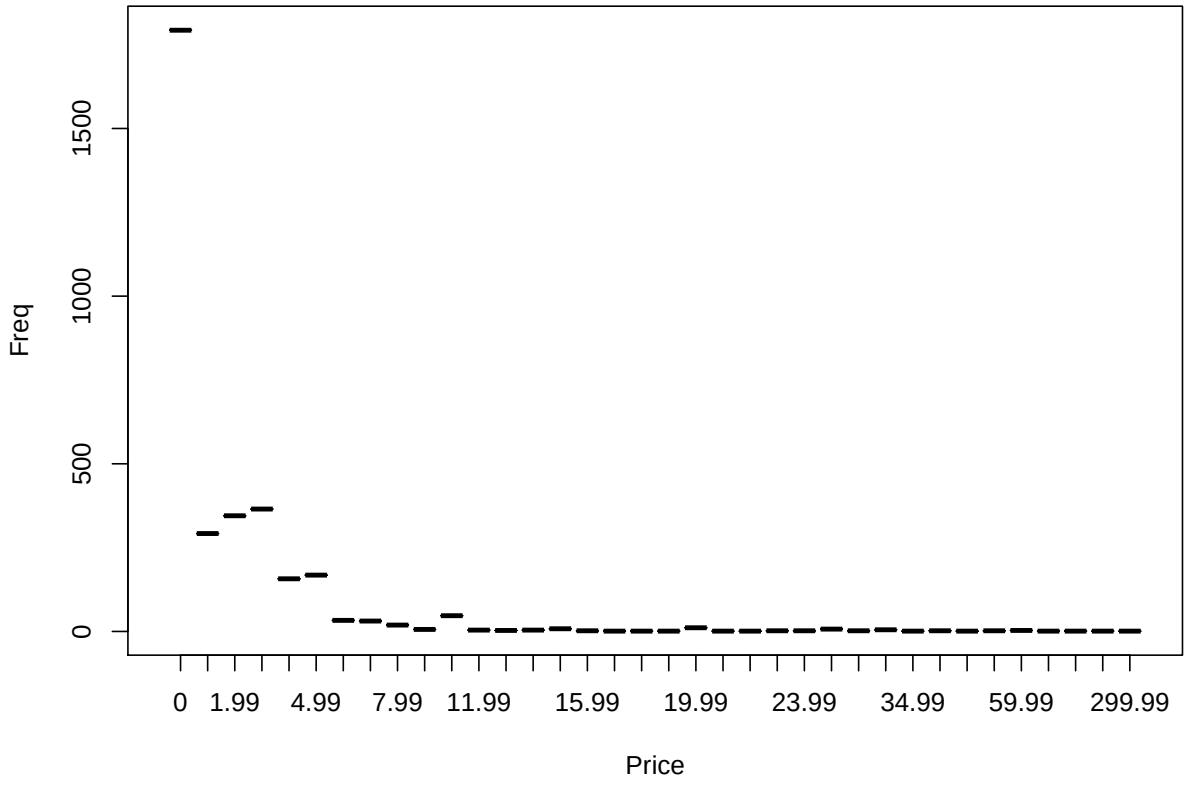
```
## [1] 0.5627446
```

```
appStore.prices <- as.data.frame(table(appStore$app_price))  # store prices in new dataframe
plot(appStore.prices, xlab = "Price")  # plot distribution of prices
```



I removed games from the data and plotted it to see what kind of difference it made (if any). Nothing changed drastically because the price data is still right-skewed.

```
plot(as.data.frame(table(appStore.noGames$app_price)), xlab = "Price")
```



Conclusion

To make **Xbot** a success, TechPointX needs to list the app as a 'Utility' and consider making the app free to incite downloads. App size and rating are positively correlated, but making an app less than 1GB is common and undoutebly expected by consumers. **Xbot** needs to have the lowest app content rating possible to increase the number of potential users. It would be beneficial to hire a localization team to ensure everyone can use the app no matter the locale. The more languages an app supports, the higher the rating according to the dataset.