CapstoneProject_FINAL

Jennifer Klein
3/31/2018

Introduction

Podcasts have emerged as the trendy new medium in media. But there's still so little we know about the audience listening to this nascent form of entertainment. We know how many times a podcast is downloaded, but we don't know audience demographics, how many people are skipping through ads, etc. That's all about to change with Apple's recent announcement that they are finally going to provide comprehensive analytics about listener behavior to podcast publishers. We can use data to seek out emerging trends in the genre.

Podcasts cultivate engaged listeners, have large audiences, and pose a legitimate challenge to the radio industry and other content mediums. And now, with podcasts slated to hit television and movie screens in the coming months, it's crucial to learn as much about Hollywood's hottest new source for material as possible.

According to Recode, the podcast industry's ad revenue will reach $220 million in 2017, up 85 percent from 2016's $119 million. The blog also discusses several interesting points regarding the medium's huge potential in both advertising revenue and growth rate:

Up until now there has been comically little data about podcast consumption, especially compared to other digital media.
This matters to podcast creators because they are unable to tell how the stuff they make performs — at best, they can usually only tell if someone has downloaded an episode or started to stream it.
This also matters to podcast advertisers, who would like to know if people are listening to the ads they pay for. Right now, many of them are doing a crude end run around this data void by asking listeners to use a show-specific code when they visit a site after hearing an ad.
Some podcast software has already provided some of this data. And the data that Apple is offering now is still fairly crude. But the majority of podcast consumption happens on Apple's software, and up until now it has been a black hole. So this is a big move for the industry, which generates a lot of attention (among media types, at least) but a very modest amount of money so far.

Despite the steady growth of podcast listening and spike in media attention over the past few years, the industry itself has trafficked in a relatively minuscule volume of cash money compared to its digital-media peers.

I would like to prove that certain topics are more appealing to the podcast listener and therefore publishers can tailor their content around these subjects to attract more listeners, garner more downloads and, ultimately, attract more advertisers and increase their revenue.

My Approach

Since I don't want my sample to be too disparate, I will stick to the top 50 or so podcasts listed in the Society & Culture section on Apple Podcasts. I also don't want to get bogged down in too much information, so I will only use the podcast name, episode title, date, running time and, most importantly, episode description and popularity rating in my set. I'll use a self-created dataset, since similar datasets on Data World or Kaggle are either few and far between or otherwise unusable for this project. Apple's API is not the most user friendly, though, so I will scrape a different podcast hosting site, Stitcher, to obtain the data for my dataset and use Apple Podcasts for their popularity rating, the most important information I need for what I'm hoping to accomplish.

Loading My Libraries

First, I'll load all the relevant libraries for this project.

```
library(tidyverse)
## Warning: package 'tidyverse' was built under R version 3.4.2
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──
## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
```

```
## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
## ✔ tibble  1.4.2     ✔ dplyr   0.7.4
## ✔ tidyr   0.8.0     ✔ stringr 1.3.0
## ✔ readr   1.1.1     ✔ forcats 0.3.0
## Warning: package 'tibble' was built under R version 3.4.3
## Warning: package 'tidyr' was built under R version 3.4.3
## Warning: package 'purrr' was built under R version 3.4.2
## Warning: package 'dplyr' was built under R version 3.4.2
## Warning: package 'stringr' was built under R version 3.4.3
## Warning: package 'forcats' was built under R version 3.4.3
## ── Conflicts
```

```
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
library(tm)
## Warning: package 'tm' was built under R version 3.4.3
## Loading required package: NLP
##
## Attaching package: 'NLP'
## The following object is masked from 'package:ggplot2':
##
##     annotate
library(wordcloud)
## Loading required package: RColorBrewer
```

Collecting My Data

When I first decided to investigate whether there was a correlation between podcast episode popularity and descriptions, I hoped to find a recent dataset on Kaggle or Data World that included that information. This proved difficult: the few datasets on podcasts I could find were either out of date or were simply lists of all podcasts listed on iTunes. Knowing that the data I needed didn't exsist, I set out to create the dataset myself from scracth.

This proved to be a fairly tedious task, but I knew that it would be a time saver when it came time for me to wrangle and clean my data. My initial thought was to use the top 100 podcasts overall on Apple podcasts (https://www.apple.com/itunes/podcasts/). On closer inspection, it became clear that the topics are so dispirate – covering history, politics, pop culture, sports, etc. – that it would be hard to prove trends in keywords and popularity. I decided to stick to one subcategory – Society & Culture – to focus on for my dataset. I also decided that since each podcast can have dozens, if not hundreds, of episodes, I would scale my dataset down to the Top 50 podcasts in Society & Culture knowing that this would still yield hundreds of episodes for me to comb through.

I started collecting data in a CSV document focusing on collecting the podcast name, episode title, date, running time and, most importantly, episode description and popularity rating. I wanted to focus on the bare minimum knowing that cutting out the clutter would yield cleaner data. This yielded 199 rows in my dataset and since Apple's API is not the most user friendly, I scraped a different podcast hosting site, Stitcher (https://www.stitcher.com/), to obtain the data and used Apple Podcasts for their popularity rating.

```
library(tidyverse)
Podcast_Dataset <- read_csv("~/Documents/Podcast_Dataset.csv")
## Parsed with column specification:
## cols(
##   Podcast = col_character(),
##   `Episode Name` = col_character(),
##   `Running Time` = col_integer(),
##   `Release Date` = col_integer(),
##   `Episode Description` = col_character(),
##   `Popularity Rating` = col_integer()
## )
glimpse(Podcast_Dataset)
## Observations: 199
## Variables: 6
## $ Podcast             <chr> "Oprah's SuperSoul Conversations", "Opra...
## $ `Episode Name`      <chr> "ALANIS MORISSETTE: IS HAPPINESS TEMPORA...
## $ `Running Time`      <int> 33, 39, 35, 30, 28, 33, 30, 34, 31, 30, ...
## $ `Release Date`      <int> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8...
## $ `Episode Description` <chr> "Grammy award-winning singer/songwriter ...
## $ `Popularity Rating`  <int> 5, 5, 5, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3...
```

Once my data was collected, the next step was to create a corpus. Since I was planning to

do some text mining later in my analysis, creating the corpus was a crucial step.

```
library(tm)
podcast_corpus <- Corpus(VectorSource(Podcast_Dataset))
```

Cleaning My Data

Once my data was collected, I set to work on cleaning it. For this, I used some basic data cleaning functions. I used:

the tolower function to make my text uniform
removed stopwords, numbers and punctuation
used the stemDocument function

```
library(tm)
library(readr)
Podcast_Dataset <- read_csv("~/Documents/Podcast_Dataset.csv")
## Parsed with column specification:
## cols(
##   Podcast = col_character(),
##   `Episode Name` = col_character(),
##   `Running Time` = col_integer(),
##   `Release Date` = col_integer(),
##   `Episode Description` = col_character(),
##   `Popularity Rating` = col_integer()
## )
podcast_corpus <- tm_map (podcast_corpus, tolower)
podcast_corpus <- tm_map (podcast_corpus, removeWords, stopwords("english"))
podcast_corpus <- tm_map (podcast_corpus, removeNumbers)
podcast_corpus <- tm_map (podcast_corpus, removePunctuation)
podcast_corpus <- tm_map (podcast_corpus, stemDocument)
podcast_corpus <- tm_map (podcast_corpus, stripWhitespace)
str(podcast_corpus)
## Warning in as.POSIXlt.POSIXct(Sys.time(), tz = "GMT"): unknown timezone
## 'default/America/Los_Angeles'
## List of 6
##  $ 1:List of 2
##   ..$ content: chr "coprah' supersoul convers oprah' supersoul convers oprah' supersoul convers
oprah' supersoul convers oprah' sup"| __truncated__
##   ..$ meta   :List of 7
##   .. ..$ author       : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "1"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  $ 2:List of 2
##   ..$ content: chr "calani morissett happi temporari mayb 's okaynn shauna niequist start live
simpler soul life brene brown part d"| __truncated__
##   ..$ meta   :List of 7
##   .. ..$ author       : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "2"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  $ 3:List of 2
##   ..$ content: chr "c"
##   ..$ meta   :List of 7
##   .. ..$ author       : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "3"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  $ 4:List of 2
##   ..$ content: chr "c"
##   ..$ meta   :List of 7
```

```
##   .. ..$ author      : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "4"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  $ 5:List of 2
##   ..$ content: chr "cgrammi awardwin singersongwrit alani morissett reflect passion art spiritu
lesson help becom ground alani cand"| __truncated__
##   ..$ meta   :List of 7
##   .. ..$ author      : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "5"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  $ 6:List of 2
##   ..$ content: chr "c"
##   ..$ meta   :List of 7
##   .. ..$ author      : chr(0)
##   .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-03-31 23:29:23"
##   .. ..$ description  : chr(0)
##   .. ..$ heading      : chr(0)
##   .. ..$ id           : chr "6"
##   .. ..$ language     : chr "en"
##   .. ..$ origin       : chr(0)
##   .. ..- attr(*, "class")= chr "TextDocumentMeta"
##   ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
##  - attr(*, "class")= chr [1:2] "SimpleCorpus" "Corpus"
```

Once my data was cleaned, I could move forward with analysis.

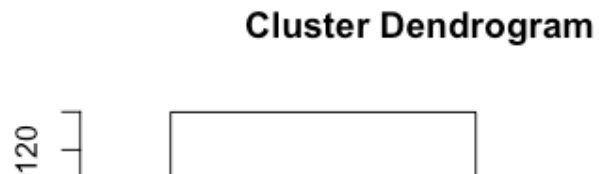
Apply Machine Learning

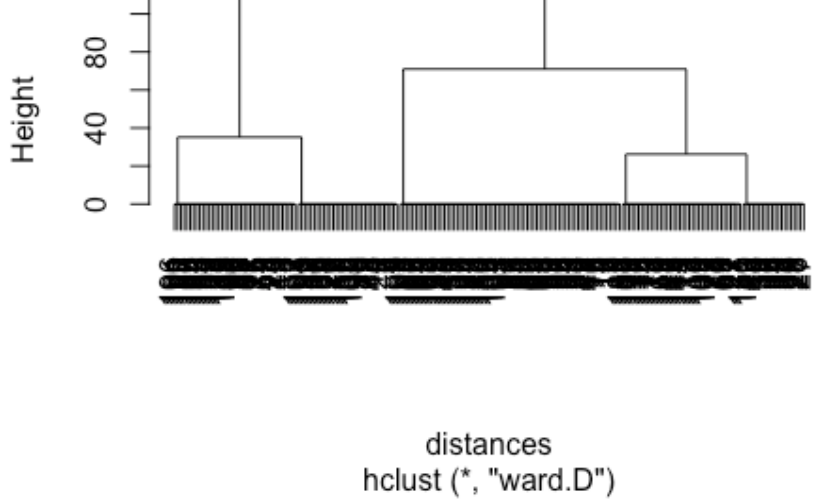
Now that my data has been cleaned, wrangled and explored, it's time to perform some analysis using machine learning.

Using the Popularity Rating system I devised earlier, I decided to use those rankings as a way to make a loose recommendation system for podcast episodes. Figuring that a listener would be eager to find more episodes that received the same rating as the episode they just enjoyed, it looks like both collaborative filtering and clustering could work well here.

Clustering is an unsupervised learning method, where I'll try to segment the data into similar groups, instead of trying to predict an outcome. Clustering can also be used to improve predictive methods. And by using Hierarchical clustering, I created a dendrogram to decide how many clusters to use in my final model.

```
library(readr)
Podcast_Dataset <- read_csv("~/Documents/Podcast_Dataset.csv")
## Parsed with column specification:
## cols(
##   Podcast = col_character(),
##   `Episode Name` = col_character(),
##   `Running Time` = col_integer(),
##   `Release Date` = col_integer(),
##   `Episode Description` = col_character(),
##   `Popularity Rating` = col_integer()
## )
distances <- dist(Podcast_Dataset[ ,6], method = "euclidean")
clusterPodcasts <- hclust(distances, method = "ward.D")
plot(clusterPodcasts)
```





According to the dendrogram, it looks like 4 clusters would be a good choice when making episode recommendations.

```
clusterGroups <- cutree(clusterPodcasts, k = 4)
tapply(Podcast_Dataset$`Popularity Rating`, clusterGroups)
##   [1] 1 1 1 2 2 2 2 2 3 3 3 3 3 3 3 3 4 1 1 2 4 4 1 1 1 4 4 4 4 4 4 4 4 4 4
##  [36] 4 4 4 4 4 4 4 4 1 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 4 4 4 4
##  [71] 4 4 4 4 1 1 1 1 2 2 3 1 3 3 3 3 3 1 1 2 2 3 1 3 3 3 1 1 1 2 2 3 2 2 3 3
## [106] 3 4 4 4 4 4 4 4 4 4 1 2 2 2 3 3 3 3 3 3 1 2 2 2 3 3 3 1 1 2 2 2 3 1 3 3
## [141] 3 3 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4
## [176] 4 4 4 4 4 4 4 4 1 1 1 1 1 1 2 1 1 2 2 2 3 1 2 3 3
cluster1 <- subset(Podcast_Dataset, clusterGroups == 1)
cluster1[1:6]
## # A tibble: 39 x 6
##    Podcast  `Episode Name` `Running Time` `Release Date` `Episode Descrip…
##    <chr>    <chr>                  <int>          <int> <chr>
##  1 Oprah's… "ALANIS MORIS…            33              8 "Grammy award-wi…
##  2 Oprah's… Shauna Niequi…            39              8 Does the life yo…
##  3 Oprah's… Brene Brown P…            35              8 What is the real…
##  4 Oprah's… Mitch Albom: …            38              8 It's been twenty…
##  5 This Am… DIY                       60              8 After four lawye…
##  6 Stuff Y… How Personali…            60              8 For millennia, w…
##  7 Stuff Y… SYSK Selects:…            37              8 In this week's S…
##  8 Stuff Y… A Lip-Smackin…            54              8 Barbecue, or for…
##  9 Revisio… The  Basement…            30              8 What is a son's …
## 10 Dan. Ca… The Celtic Ho…           359              8 Julius Caesar is…
## # ... with 29 more rows, and 1 more variable: `Popularity Rating` <int>
```

We can see that cluster 1 has the episodes rated 5, cluster 2 has the episodes rated 4, cluster 3 has the episodes rated 3 and cluster 4 has the episodes rated 2 or 1. We can then take a subset of episodes that are only rated 5, make a new dataset with only those episodes and pull recommendations just from that set.

Recommendation systems can definitely be an advantage to companies trying to reach consumers in an increasingly fractured entertainment marketplace. An excellent recommendation system can provide a company with the leg up they need. Clustering algorithms can certainly go a long way to help reach your intended audience.

Statistical Analysis

Some background on my dataset: it features 20 podcasts and 199 episodes. The shortest episode clocked in at a mere 15 minutes, while the longest was a whopping 6 hours. The most recent podcast episode was published August 31, 2017 (the day I pulled my data) and an episode of Dan. Carlin's Hardcore History from October 30, 2013 was still hanging in there with a 3 ranking.

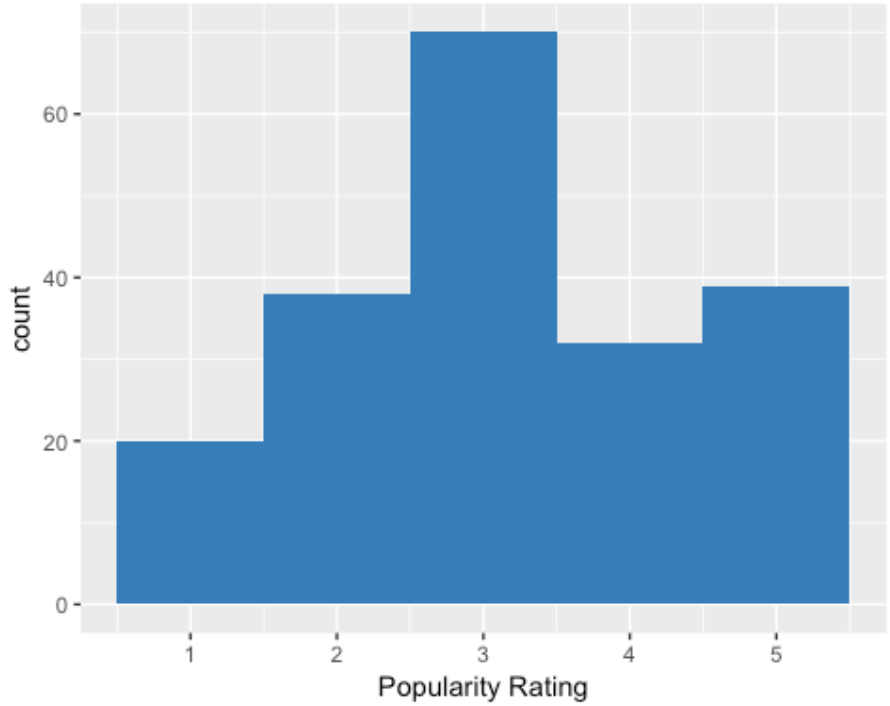
I decided to create a histogram, a scatterplot and a time-series plot to try to gain insight into my data.

Creating The Histogram

For my histogram, I decided to look into the "Episode Popularity" column from my dataset.

As I outlined in the Data Wrangling section, I used Apple Podcasts' popularity rankings to assign a numerical rating (1-5, with 5 being the highest) to each episode from the top 50 most listened to podcasts in the "Society & Culture" section on Apple's site.

```
library(ggplot2)
library(readr)
Podcast_Dataset <- read_csv("~/Documents/Podcast_Dataset.csv")
## Parsed with column specification:
## cols(
##   Podcast = col_character(),
##   `Episode Name` = col_character(),
##   `Running Time` = col_integer(),
##   `Release Date` = col_integer(),
##   `Episode Description` = col_character(),
##   `Popularity Rating` = col_integer()
## )
ggplot(Podcast_Dataset, aes(`Popularity Rating`)) +
 geom_histogram(binwidth = 1, fill = "#377EB8")
```



Here we can see that the majority of podcast episodes are only somewhat popular, with most being ranked at around 3. Interestingly, we can also see that popular podcasts ranked at a 5 pop up only slightly more frequently than those that are slightly unpopular, ranked at a 2.

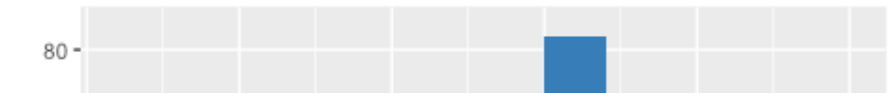
As for the Release Date data, we can see that podcast episodes from the summer months appear with the highest density. This makes sense, as I pulled my data in the summer

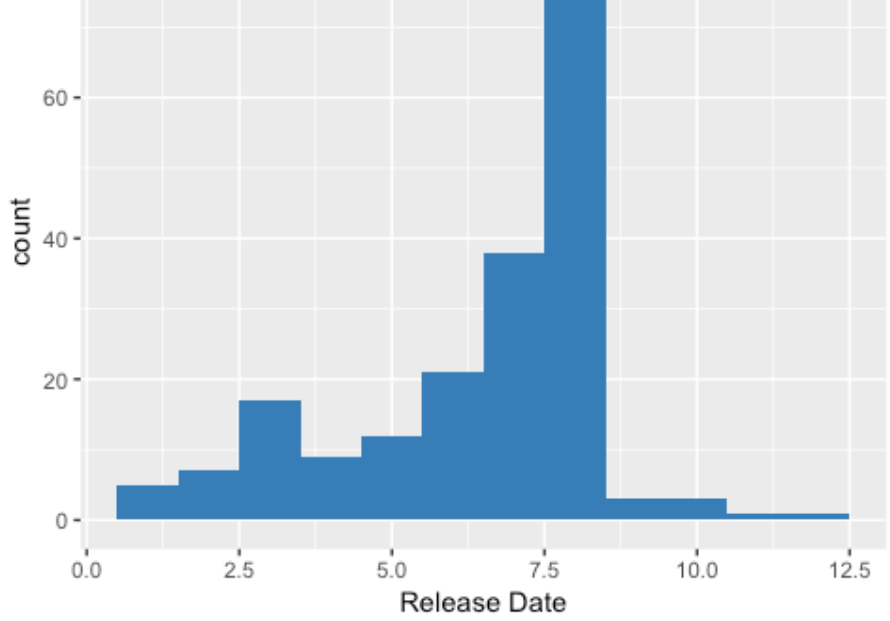
Creating The Scatterplot

Since I collected my data over the summer, I wondered if perhaps the popularity ratings were showing a recency bias. As you can see in the histogram below, podcast episodes released in July and August appear with by far the highest density.

```
library(ggplot2)
library(readr)

Podcast_Dataset <- read_csv("~/Documents/Podcast_Dataset.csv")
## Parsed with column specification:
## cols(
##   Podcast = col_character(),
##   `Episode Name` = col_character(),
##   `Running Time` = col_integer(),
##   `Release Date` = col_integer(),
##   `Episode Description` = col_character(),
##   `Popularity Rating` = col_integer()
## )
ggplot(Podcast_Dataset, aes(`Release Date`)) +
geom_histogram(binwidth = 1, fill = "#377EB8")
```



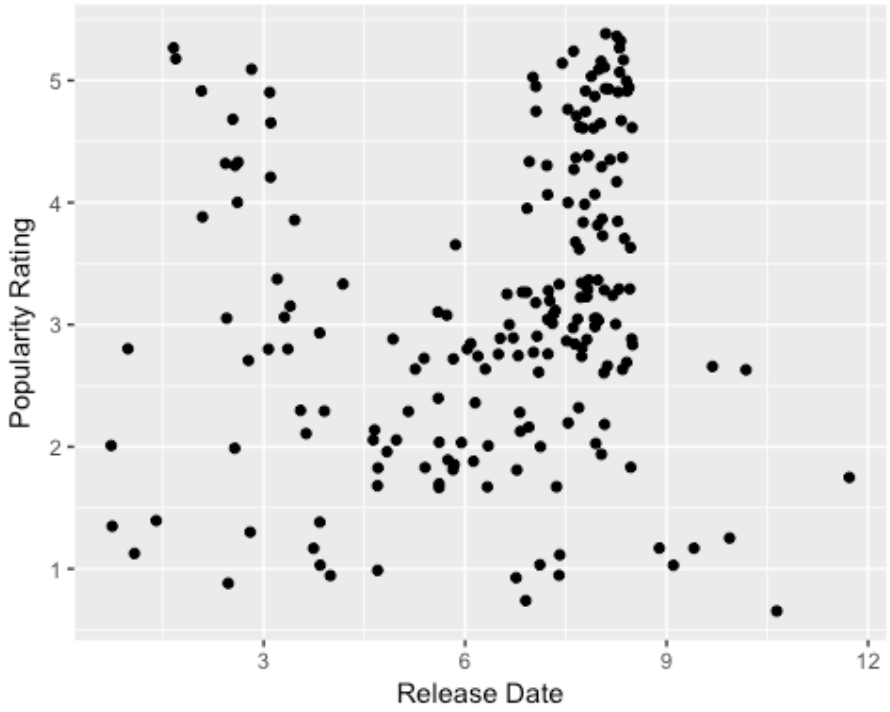


My thought was that perhaps the most popular podcasts were the ones most recently listened to by subscribers who downloaded new episodes every week. These summer episodes would be the most popular, receiving ratings of 4 and 5, while podcasts with earlier release dates would have ratings of 3 or lower.

I decided to make a scatterplot of the "Release Date" and "Popularity Rating" variables. I assigned each month their numeric value (1 for January, 2 for February, etc.) to make my data easier to plot.

```
library(ggplot2)

ggplot(Podcast_Dataset, aes(x = `Release Date`, y = `Popularity Rating`)) +
 geom_point(position = position_jitter(0.5))
```



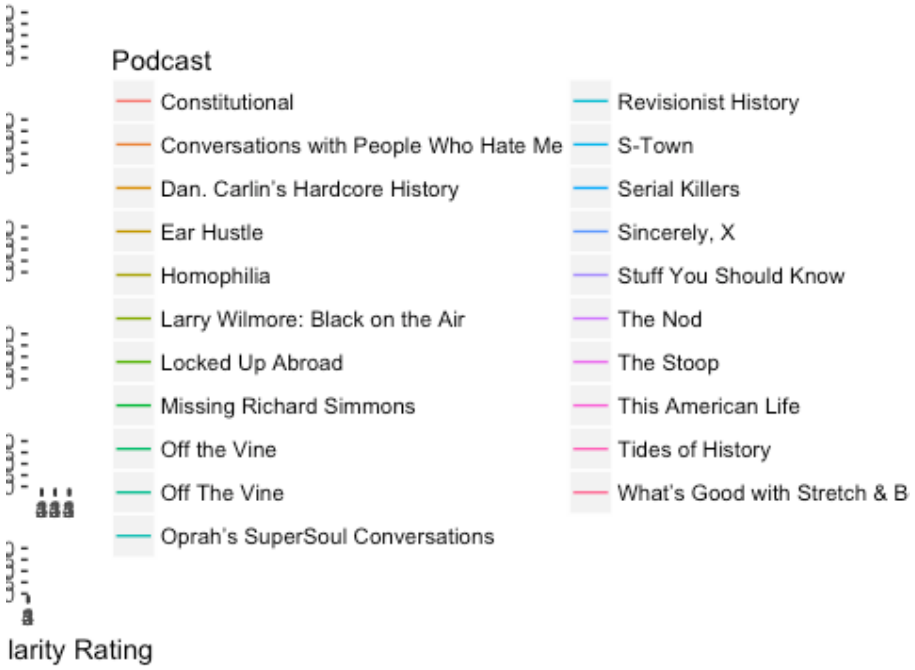
It turns out, there is somewhat of a correlation there. There are definitely more podcast episodes pulled from the summer months of July and August, and their popularity ratings are pretty high - we can see a lot of 3s, 4s and 5s. Of course, there are some outliers there: what is it about those episodes from January and February that remained so popular months later? I wanted to see what else I could analyze in relation to popularity rating.

Creating The Time-series Plot

I decided to make a time-series plot for each Podcast's Popularity Rating and Running Time. Maybe shorter podcasts are more popular?

```
ggplot(Podcast_Dataset, aes(x = `Popularity Rating`, y = `Running Time`, color = `Podcast`)) +
```

```
geom_line(position = position_jit(0.5)) +facet_wrap(~Podcast, ncol=4)
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
```



According to this time-series plot, there isn't really a correlation between episode length and popularity. The most popular podcasts come in all different lengths: short, long and in between.

So outside of recency, what makes a podcast popular? What do listeners want and how can that translate for advertisers and publishers? My guess is that content is king - it's time to get into some text analysis.

Creating a DTM

I then created a Document Term Matrix so I can see the frequency of terms that occur in my data set. We can see that the DTM has 2,567 terms.

```
podcast_corpus <- tm_map(podcast_corpus, PlainTextDocument)
podcast_corpus <- Corpus(VectorSource(podcast_corpus))
podcast_DTM <- DocumentTermMatrix(podcast_corpus)

dim(podcast_DTM)
## [1]    3 2549
```

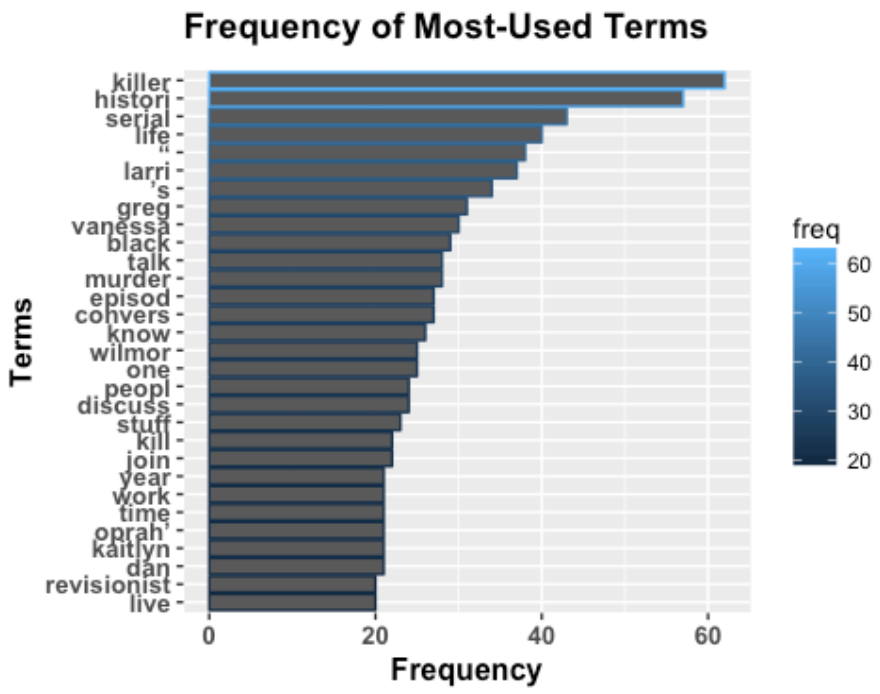
Plotting Frequency

Now for the fun stuff: we can start to look for stories in our data.

First, we'll look at the most frequent words - words occurring more than 20 times, in our case. Since the data set we're working with is fairly small, it made sense to keep this number relatively low. If we used a number much higher, we'd run the risk of not returning any results at all.

```
min_freq <- 20
term_freq <- colSums(as.matrix(podcast_DTM))
term_freq <- subset(term_freq, term_freq >= min_freq)
freq_words_df <- data.frame(term = names(term_freq), freq = term_freq)
findFreqTerms(podcast_DTM, lowfreq = min_freq)
##  [1] "black"      "convers"    "dan"        "discuss"    "episod"
##  [6] "greg"       "histori"    "join"       "kaitlyn"    "kill"
## [11] "killer"     "know"       "larri"      "life"       "live"
## [16] "murder"     "one"        "oprah'"     "peopl"      "revisionist"
## [21] "serial"     "stuff"      "talk"       "time"       "vanessa"
## [26] "wilmor"     "work"       "year"       "'s"         "“"
ggplot(data = freq_words_df, aes(x = reorder(term, freq), y = freq, colour = freq)) +
 geom_bar(stat="identity") +
 coord_flip() +
 ggtitle("Frequency of Most-Used Terms") +
 xlab("Terms") +
 ylab("Frequency") +
```

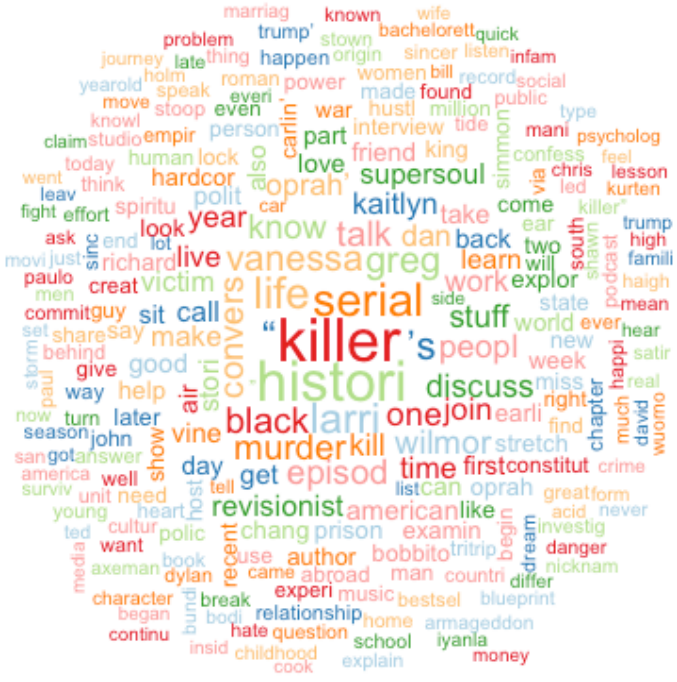
```
theme(plot.title = element_text(size=14, face="bold", margin = margin(10, 0, 10, 0)),
axis.title.x = element_text(face="bold", size = 12),
axis.title.y = element_text(face="bold", size = 12),
axis.text.x = element_text(face="bold", size=10),
axis.text.y = element_text(face="bold", size=10))
```



It looks like true crime is a very popular Podcast topic: “killer”, “murder”, and “victim” are all made it into our most used terms. This makes sense for a medium that exploded in popularity after the series “Serial” premiered in late 2014 (in fact, “Serial” is the 4th most used term in our matrix.) “Conversation,” “Discuss” and “Call” are all in our chart. This, to me, suggests that podcasts that feature two hosts interacting with each other, or with listeners calling in, might be more engaging to listeners than one person telling a story or monologue. We also see that the term “historical,” “learn” and “political” pop up a lot - perhaps listeners are interested in historical narratives or in learning something about our past to inform our present political climate. And of course, perennial favorites “love” and “life” make an appearance, too.

To better visualize term frequency, we can also look at a wordcloud of the terms.

```
wordcloud (podcast_corpus, scale = c(2, 0.5), colors = brewer.pal(8, "Paired"), random.color = TRUE, random.order = FALSE, max.words = 250)
```



Going Forward: Next Steps and Recommendations

Going Forward: Next Steps and Recommendations

Recommendation systems can definitely be an advantage to companies trying to reach consumers in an increasingly fractured entertainment marketplace. An excellent recommendation system can provide a company with the leg up they need.

Clustering algorithms can certainly go a long way to help reach your intended audience. It's crazy that Apple hasn't tried this approach yet; recommendation algorithms can make their interface less intimidating and more fun and addictive for the listener. If publishers can successfully implement the "If You Liked X, Try Y" models popular on Netflix and Amazon, it would really help podcasts become more mainstream and, ultimately more lucrative.

When I started this project, I initially wanted to use more text mining in my analysis and see if I could use term associations to see if there was a correlation between episode description and popularity. Unfortunately, this is where some of the limitations of my dataset became apparent. Due to its relatively small size, very few terms came back once I split the set into sections divided by popularity rating. Try as I might, I couldn't draw any conclusions from the sole results of "language" and "list."

But, clearly, my data still tells a story. Going forward, I would recommend that publishers focus on producing content with the themes that popped up again and again in the Top 50 Society & Culture podcasts on Apple: true crime, historical and political content and lifestyle/wellness. Celebrity hosts don't hurt, either.