Exploratory Analysis of Prince Lyrics

Ryan Jerving

April 29, 2018

Intro

This is a project based on a DataCamp tutorial by Debbie Liske.[1]

The project aims to demonstrate three things:

- Text mining with R tidyverse tools
- Sentiment analysis, topic modeling, and natural language processing (NLP)
- Predictive analytics using machine learning tools

```
# ![Prince](https://media.boingboing.net/wp-content/uploads
```

[1] Debbie Liske, "Lyric Analysis with NLP & Machine Learning with R," *DataCamp*, @ https://www.datacamp.com/community/tutorials/R-nlp-machine-learning

Loading Libraries and Data Sets

First, we'll import any libraries needed to do these tasks, then pull in the csv that Liske created with Prince's lyrics, release year, and Billboard position for each song.

```
# most of the libraries needed

library(dplyr) #data manipulation
library(ggplot2) #visualizations
library(gridExtra) #viewing multiple plots together
library(tidytext) #text mining
library(wordcloud2) #creative visualizations

# now the data, preventing R from converting strings to fa

prince_orig =
  read.csv("https://s3.amazonaws.com/assets.datacamp.com/bl
           stringsAsFactors = FALSE)
```

Intial Exploration

Let's see what the column headings are:

```
names(prince_orig)
```

```
##  [1] "X"            "text"         "artist"       "song"
##  [5] "year"         "album"        "Release.Date" "US.Po
##  [9] "US.R.B"       "CA"           "UK"           "IR"
## [13] "NL"           "DE"           "AT"           "FR"
## [17] "JP"           "AU"           "NZ"           "peak"
```

From these, we'll only need row number `X`, the lyrics (`text`), song title, `album` title, release `year`, and `peak` Billboard, `US.Pop`, and `US.R.B` variables.

```
# Select desired columns, renaming when it would help to d

prince <- prince_orig %>%
  select(lyrics = text, song, year, album, peak,
         us_pop = US.Pop, us_rnb = US.R.B)
```

Cleaning and Transforming Data

Now, let's work with this dataset to condition it for analysis.

We'll create a function using `gsub()` to replace contractions across the corpus, along with another function to preserve only alphanumeric characters, and then converting all text to lowercase.

```
# Function to expand contractions in an English-language so

fix.contractions <- function(doc) {
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("can't", "can not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'d", " would", doc)
  # we'll leave "'s" alone since it could also be possessi
  doc <- gsub("'s", "", doc)
```

Descriptive Statistics

To start visualizing and analyzing what we've got, we'll work with Debbie Liske's preferred color scheme for creating plots.

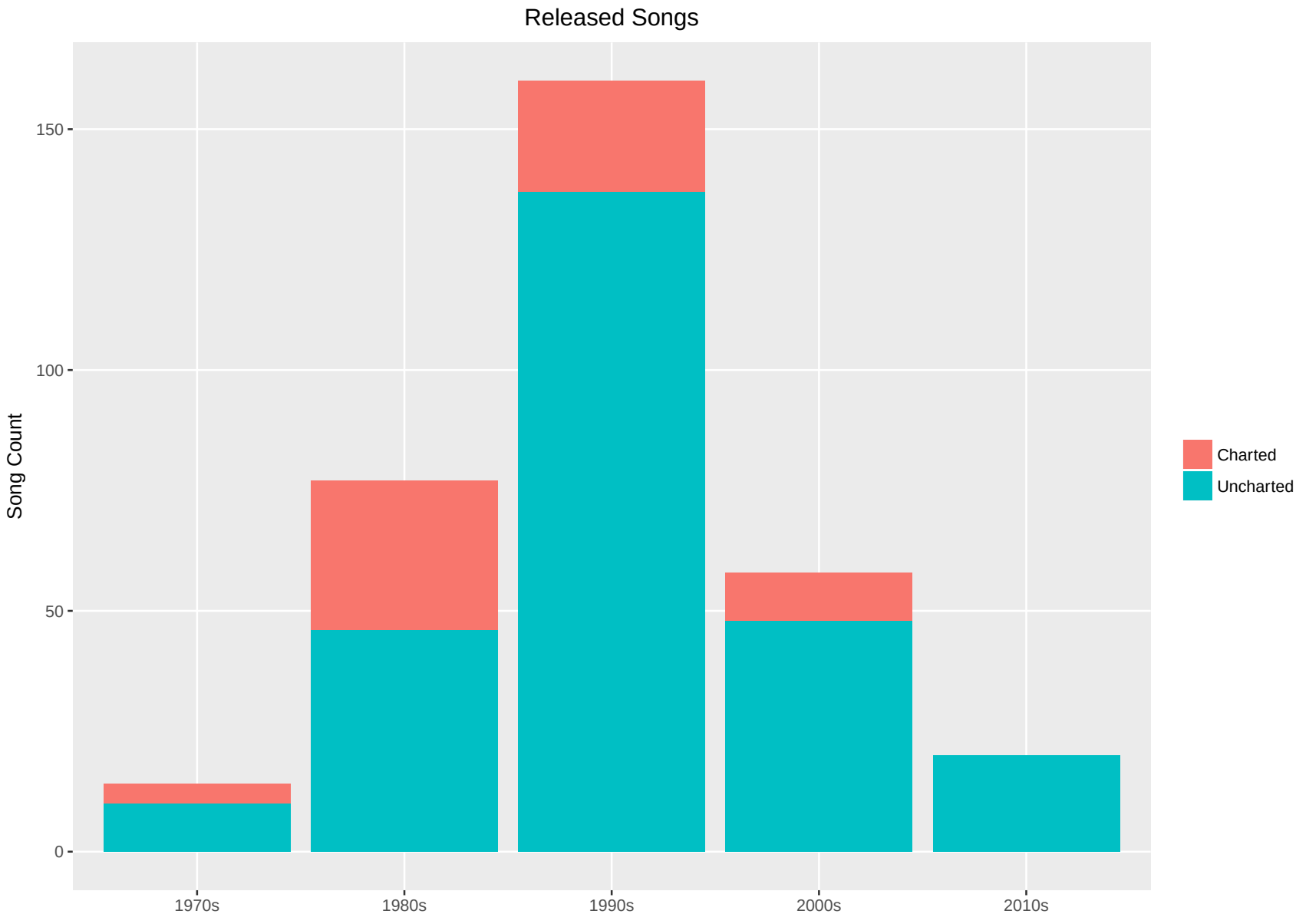
```
# define color scheme

dl_colors <- c("#E69F00", "#56B4E9", "#009E73", "#CC79A7",

# ...and a function for how we'll handle the display

theme_lyrics <- function()
{
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        legend.position = "none")
}
```

Chart


Released Songs
Song Count
150
100
50
0
1970s
1980s
1990s
2000s
2010s
Charted
Uncharted