Lyric Mining

David

May 30, 2019

```
#
# Source:
#   Kaggle; GyanendraMishra
#   Various artists and genres, 380,000+ lyrics from MetroLyrics
#
# Sentiment Analysis
```

Library and packages

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 3.5.3
```

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.5.2
```

```
# Would data.table and fread() help here?
```

Importing the data

```
song_lyrics <- read_csv("lyrics.csv")
```

```
## Parsed with column specification:
## cols(
##   index = col_double(),
##   song = col_character(),
##   year = col_double(),
##   artist = col_character(),
##   genre = col_character(),
```

```
##   lyrics = col_character()
## )
```

```
# fread() would read it faster but some serious memory issues came up at tidy_lyrics.
# Though using data.tables should be more memory-efficient, I'm unsure how it reacts
# with the dplyr package.

# First look to see what we have
glimpse(song_lyrics)
```

```
## Observations: 362,237
## Variables: 6
## $ index  <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
## $ song   <chr> "ego-remix", "then-tell-me", "honesty", "you-are-my-roc...
## $ year   <dbl> 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2...
## $ artist <chr> "beyonce-knowles", "beyonce-knowles", "beyonce-knowles"...
## $ genre  <chr> "Pop", "Pop", "Pop", "Pop", "Pop", "Pop", "Pop", "Pop",...
## $ lyrics <chr> "Oh baby, how you doing?\nYou know I'm gonna cut right ...
```

```
# To do:
# Unnest lyrics to words
# Total word counts
# Sentiment counts
# Plots, fit model
#
# Questions:
# Has sentiments changed over time ?
#   -> is it significant ?
#
# Which genre uses the fewest words ?
```

Cleaning up lyrics

```
# Unnest the lyrics
tidy_lyrics <- song_lyrics %>%
  unnest_tokens( word, lyrics)

rm(song_lyrics)

# Each word for a song is now a record instead of the full lyrics as one record
glimpse(tidy_lyrics)
```

```
## Observations: 61,077,385
## Variables: 6
## $ index  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ song   <chr> "ego-remix", "ego-remix", "ego-remix", "ego-remix", "eg...
## $ year   <dbl> 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2009, 2...
## $ artist <chr> "beyonce-knowles", "beyonce-knowles", "beyonce-knowles"...
## $ genre  <chr> "Pop", "Pop", "Pop", "Pop", "Pop", "Pop", "Pop", "Pop",...
## $ word   <chr> "oh", "baby", "how", "you", "doing", "you", "know", "i'...
```

```
sentiments %>%
  group_by(lexicon) %>%
  summarize(n_distinct(lexicon))
```

```
## # A tibble: 4 x 2
##   lexicon  `n_distinct(lexicon)`
##   <chr>                    <int>
## 1 AFINN                        1
## 2 bing                         1
## 3 loughran                     1
## 4 nrc                          1
```

```
# Bing sorts words into positive or negative positions.
# AFINN grades words between -5 and 5 (scaled negative/positive sentiments)
# NRC lexicon categorizes sentiment words into positive, negative, anger,
#   anticipation, disgust, fear, joy, sadness, surprise and trust.
#
#
# Lets choose NRC for now before we scale with AFINN, an overall view of
# what kind of songs we have.
```

Word totals by song

```
totals <- tidy_lyrics %>%
  select(song,genre) %>%
  # Count by song to find the word totals for each song
  count(song) %>%
  # Rename the new column
  rename(total_words = n)


# Print totals
glimpse(totals)
```

```
## Observations: 249,455
## Variables: 2
## $ song        <chr> "0-0", "0-0-0", "0-0-where-evil-dwells", "0-71", "...
## $ total_words <int> 204, 93, 193, 14, 879, 979, 879, 1, 153, 73, 250, ...
```

```
lyric_counts <- tidy_lyrics %>%
  # Combine totals with tidy_lyrics using the "song" column
  left_join(totals, by = "song")



# Removed
rm(totals, tidy_lyrics)
```

Here<<

Words by Genre [song_total used for queries]

```
#memory issues

#What genre tends to use a low amount of total words?

song_total <- distinct(lyric_counts[, -6])
```

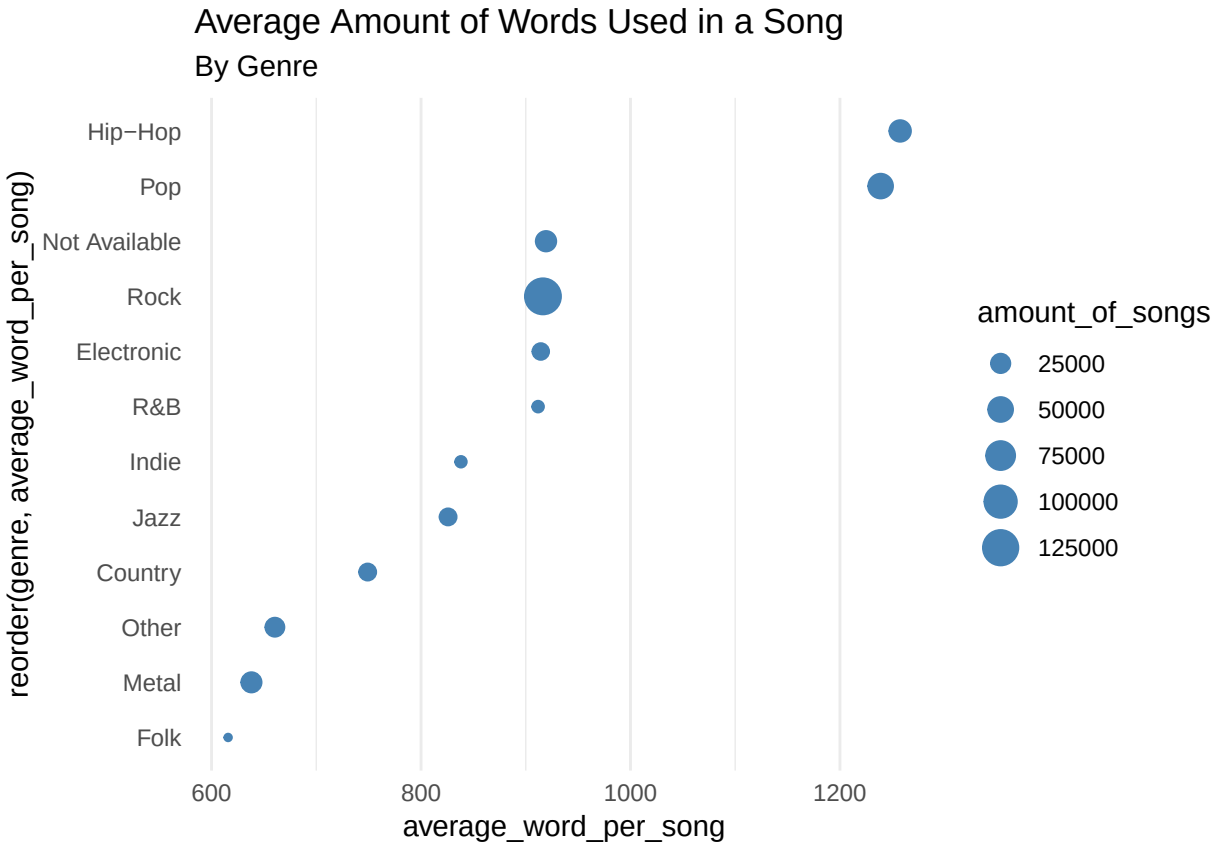
```
song_total <- distinct(lyric_counts[, -6])

q1 <- song_total %>%
  group_by(genre) %>%
  summarise(
    word_total_per_genre = sum(total_words),
    amount_of_songs = n(),
    average_word_per_song = word_total_per_genre/amount_of_songs) %>%
    arrange(desc(average_word_per_song))
```

```
## Warning: package 'bindrcpp' was built under R version 3.5.2
```

```
# show_query(q1) how to use function?

# Check to see if query working as should
song_total %>%
  filter(genre == "Rock") %>%
  count()
```

```
## # A tibble: 1 x 1
##        n
##    <int>
## 1 130789
```

```
q1 %>%
  ggplot( aes(x = reorder(genre,average_word_per_song), y = average_word_per_song, size = amount_of_song
  geom_point( color = "steelblue" ) +
  coord_flip() +
  theme_minimal() +
  theme( panel.grid.major.y = element_blank()   ) +
  labs(title = "Average Amount of Words Used in a Song" , subtitle = "By Genre")
```

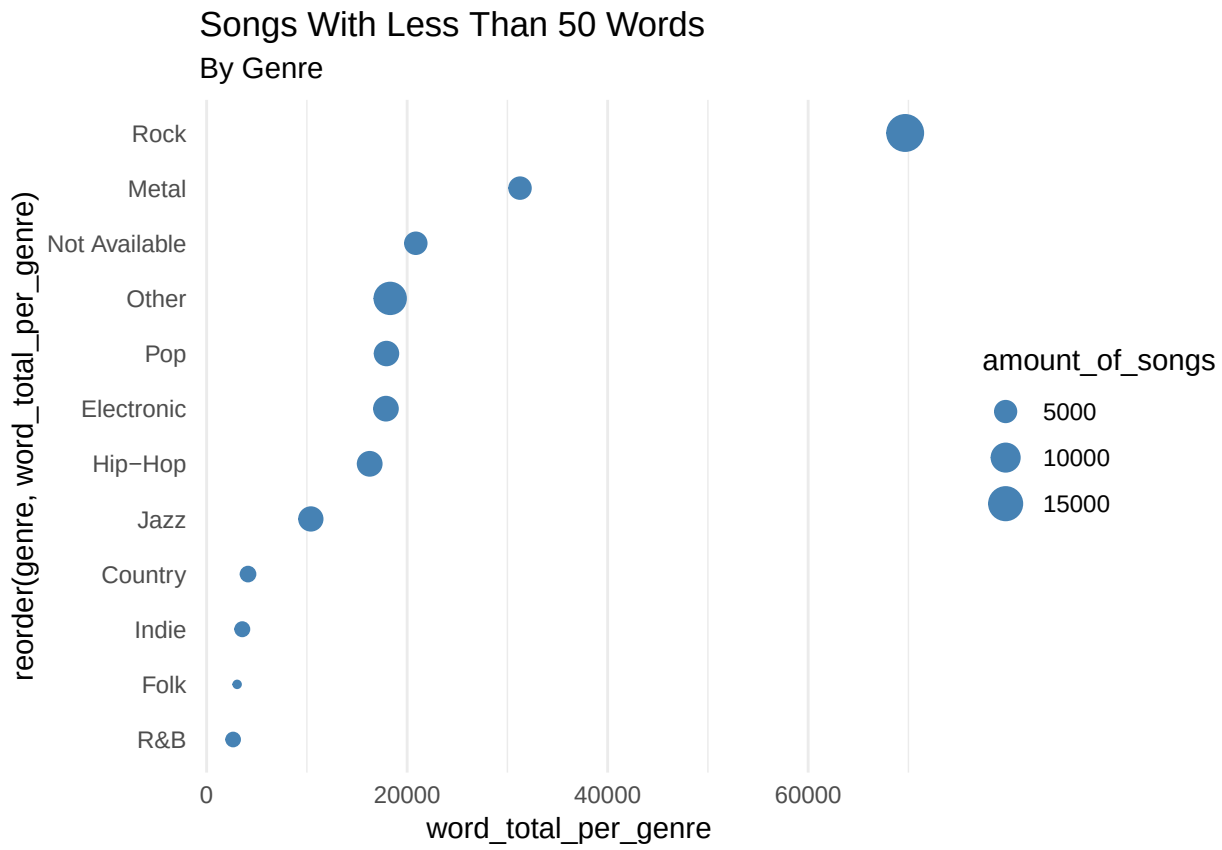


Here<<

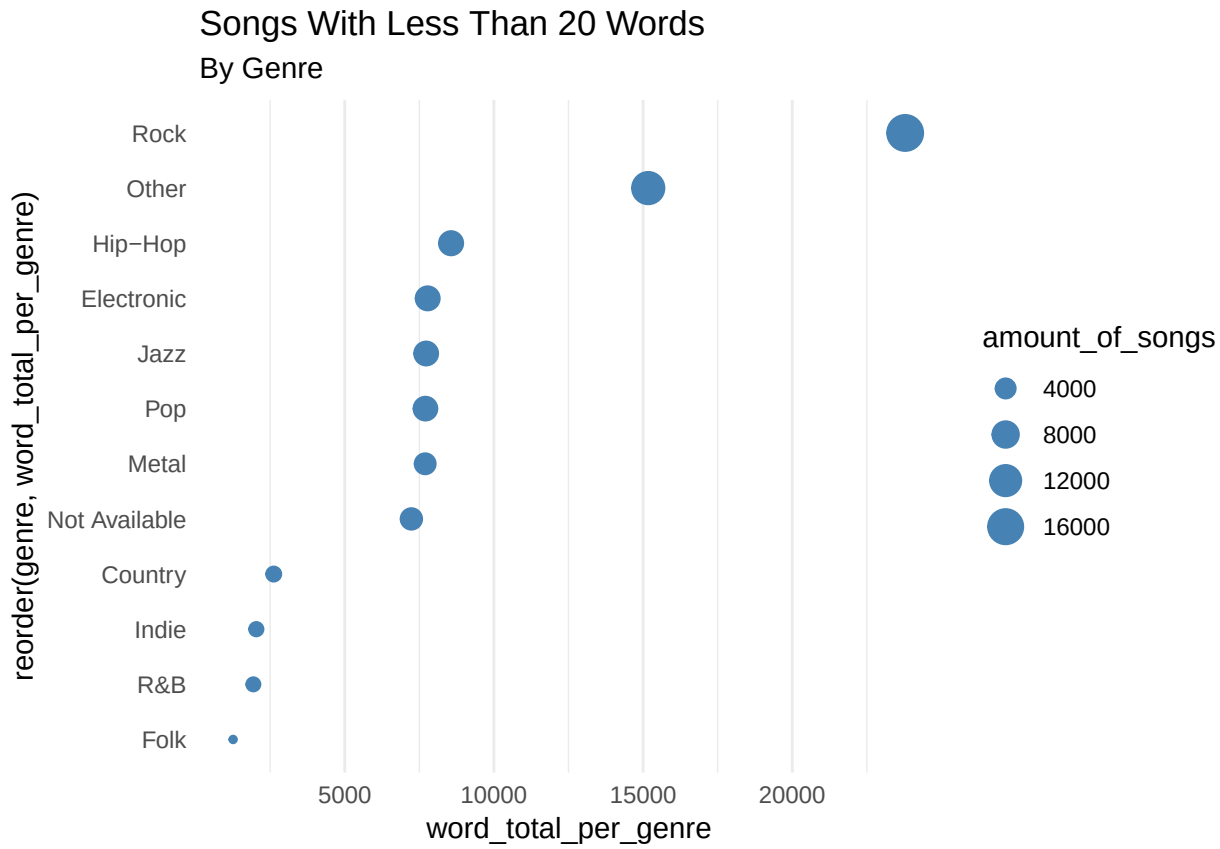
Fewest words

```
# What we want:
# Count of songs by genre
# When lyric total < 50, 20, 10
# Visualize this with plots

# Less than 50

q2 <- song_total %>%
  filter(total_words < 50) %>%
  group_by(genre) %>%
  summarise(
    word_total_per_genre = sum(total_words),
    amount_of_songs = n())

# Less than 20
q3 <- song_total %>%
  filter(total_words < 20) %>%
  group_by(genre) %>%
  summarise(
    word_total_per_genre = sum(total_words),
    amount_of_songs = n())
```

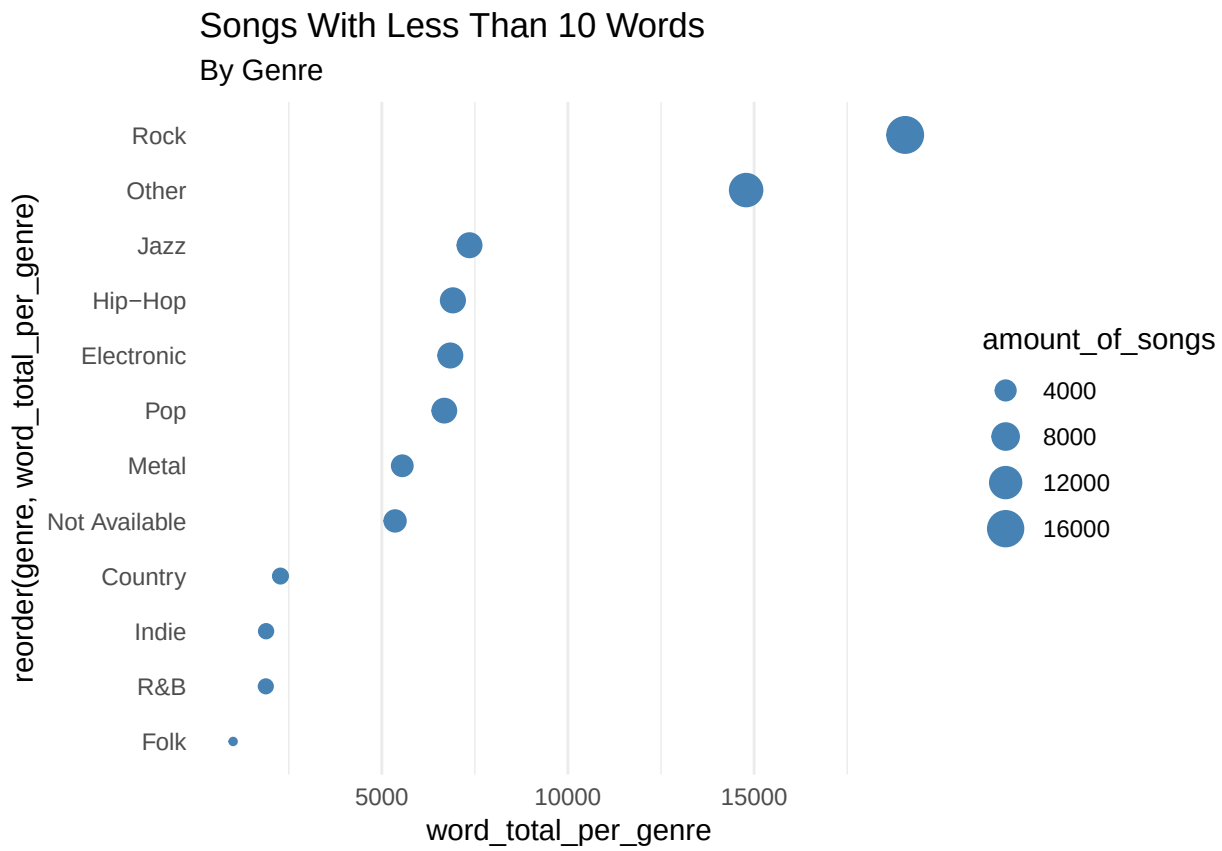
```r
# Less than 10
q4 <- song_total %>%
  filter(total_words < 10) %>%
  group_by(genre) %>%
  summarise(
    word_total_per_genre = sum(total_words),
    amount_of_songs = n())

par(mfrow = c(3,1), mar = c(4,5,3,3))

q2 %>%
  ggplot( aes(x = reorder(genre,word_total_per_genre), y = word_total_per_genre, size = amount_of_songs)
  geom_point( color = "steelblue" ) +
  coord_flip() +
  theme_minimal() +
  theme( panel.grid.major.y = element_blank()   ) +
  labs(title = "Songs With Less Than 50 Words" , subtitle = "By Genre")
```




```r
q3 %>%
  ggplot( aes(x = reorder(genre,word_total_per_genre), y = word_total_per_genre, size = amount_of_songs)
  geom_point( color = "steelblue" ) +
  coord_flip() +
  theme_minimal() +
  theme( panel.grid.major.y = element_blank()   ) +
  labs(title = "Songs With Less Than 20 Words" , subtitle = "By Genre")
```




```r
q4 %>%
  ggplot( aes(x = reorder(genre,word_total_per_genre), y = word_total_per_genre, size = amount_of_songs)
  geom_point( color = "steelblue" ) +
  coord_flip() +
  theme_minimal() +
  theme( panel.grid.major.y = element_blank()   ) +
  labs(title = "Songs With Less Than 10 Words" , subtitle = "By Genre")
```




```
# Removed
rm(totals, tidy_lyrics, q1,q2,q3,q4)
```

```
## Warning in rm(totals, tidy_lyrics, q1, q2, q3, q4): object 'totals' not
## found
```

```
## Warning in rm(totals, tidy_lyrics, q1, q2, q3, q4): object 'tidy_lyrics'
## not found
```

Sentiment Count by song (NRC)

```
lyric_sentiment <- lyric_counts %>%
    # Sentiment analysis with the "nrc" lexicon
    inner_join( get_sentiments("nrc"), by = "word" )

# This should help with some memory, neutral words will be taken out and reducing the
# records(observations) from 61 million to 15.2 million.


# Removed
rm(lyric_counts)



lyric_sentiment %>%
    # How many sentiment words each song has
```

```
    count(song, sentiment, sort = TRUE) %>%
    head()
```

```
## # A tibble: 6 x 3
##   song         sentiment     n
##   <chr>        <chr>     <int>
## 1 silent-night positive   2072
## 2 beautiful    positive   1945
## 3 angel        positive   1640
## 4 shine        positive   1568
## 5 beautiful    joy        1534
## 6 i-love-you   positive   1533
```

Negative vs Positive sentiment (NRC)

```
# What songs have the highest proportion of negative words?
lyric_sentiment %>%
    # Count by song, sentiment, & total_words
    count(song, sentiment, total_words) %>%
    ungroup() %>%
    # New percent column
    mutate(percent = n / total_words) %>%
    # Filter for only negative words
    filter(sentiment == "negative") %>%
    # Arranged by descending percent
    arrange(desc(percent)) %>%
    head()
```

```
## # A tibble: 6 x 5
##   song                         sentiment total_words     n percent
##   <chr>                        <chr>           <int> <int>   <dbl>
## 1 installation-no1             negative            1     1   1
## 2 reprezione                   negative            1     1   1
## 3 stonesphere                  negative            1     1   1
## 4 just-bang                    negative           44    34   0.773
## 5 the-decapitaion-of-a-cattle negative            3     2   0.667
## 6 the-decapitation-of-cattle  negative            3     2   0.667
```

```
# What songs have the highest proportion of positive words?
lyric_sentiment %>%
    count(song, sentiment, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words) %>%
    filter(sentiment == "positive") %>%
    arrange(desc(percent)) %>%
    head()
```

```
## # A tibble: 6 x 5
##   song              sentiment total_words     n percent
##   <chr>             <chr>           <int> <int>   <dbl>
## 1 000111            positive            1     1       1
## 2 0100101110        positive            1     1       1
## 3 10-ft-5-ft-bag-em positive            1     1       1
## 4 1014-a-d          positive            1     1       1
```

```
## 5 11-97-shape-cd    positive            1     1       1
## 6 12                positive            1     1       1
```

```
# Misleading with total_words = 1, lets use 20 as a base of how many total words in a song.
# This should exlude a lot of instrumental music.

lyric_sentiment %>%
    count(song, sentiment, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words) %>%
    filter(sentiment == "negative", total_words > 20) %>%
    arrange(desc(percent)) %>%
    head()
```

```
## # A tibble: 6 x 5
##   song                          sentiment total_words     n percent
##   <chr>                         <chr>           <int> <int>   <dbl>
## 1 just-bang                     negative           44    34   0.773
## 2 tripping-the-light-fantastic negative           44    28   0.636
## 3 lie-lie-lie-lie               negative          168    91   0.542
## 4 fight-war-not-wars            negative           32    16   0.5
## 5 nag-nag-nag                   negative          106    53   0.5
## 6 split-coconut                 negative           24    12   0.5
```

```
lyric_sentiment %>%
    count(song, sentiment, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words) %>%
    filter(sentiment == "positive", total_words > 20) %>%
    arrange(desc(percent)) %>%
    head()
```

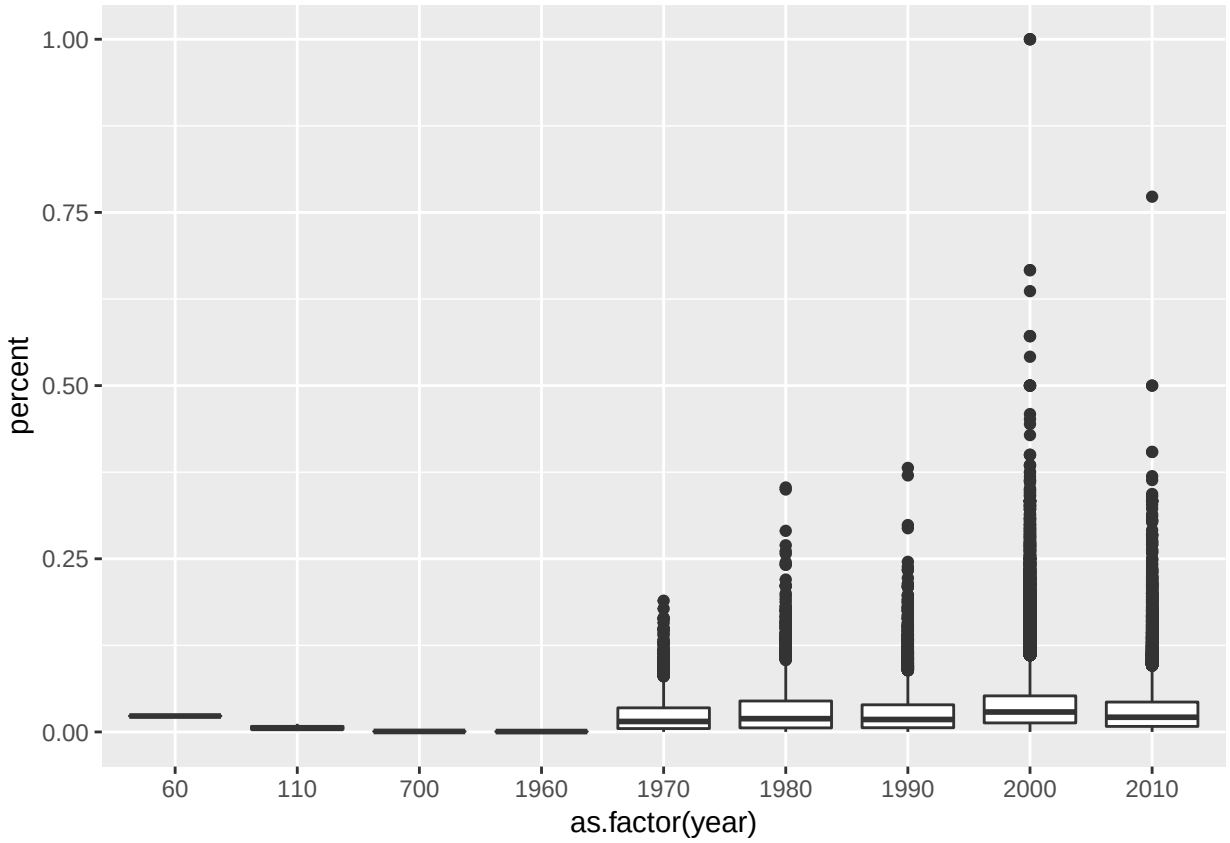
```
## # A tibble: 6 x 5
##   song                       sentiment total_words     n percent
##   <chr>                      <chr>           <int> <int>   <dbl>
## 1 rock-n-rave                positive           29    20   0.690
## 2 robot-rock                 positive          120    80   0.667
## 3 robot-rock-oh-yeah         positive          156   104   0.667
## 4 holy-god                   positive          153    94   0.614
## 5 hold-me-hug-me-rock-me     positive          209   117   0.560
## 6 oh-sing-sweet-nightingale positive           59    32   0.542
```

Q:Total words is only 1, are these short audio clips in albums or just instrumental music? A:instrumental

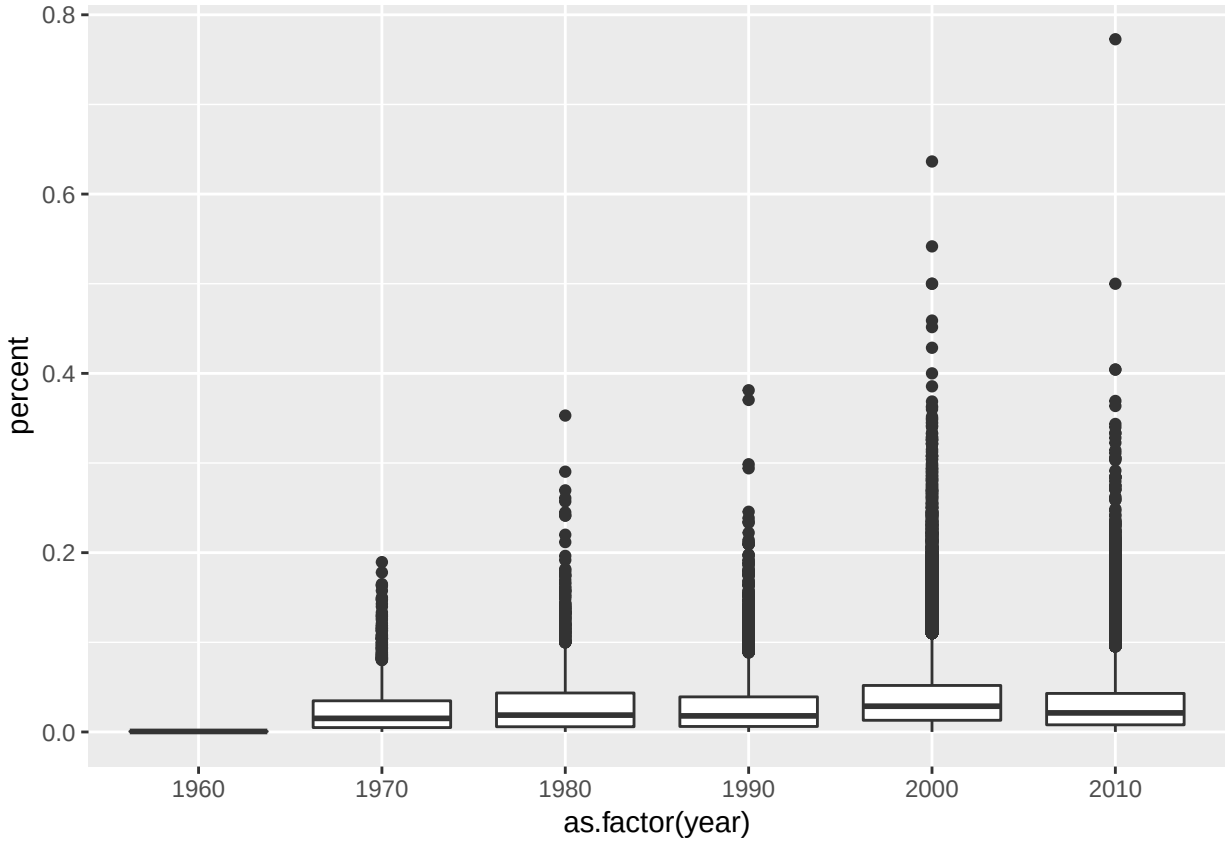
Negative vs positive sentiments over time [*Add plot titles, remove outliers from boxplots keep IQR*]

```
# geom_violin > geom_boxplot

# How is negative sentiment changing over time?
lyric_sentiment %>%
    # Filter for only negative words
    filter(sentiment == "negative") %>%
    count(song, year, total_words) %>%
    ungroup() %>%
```

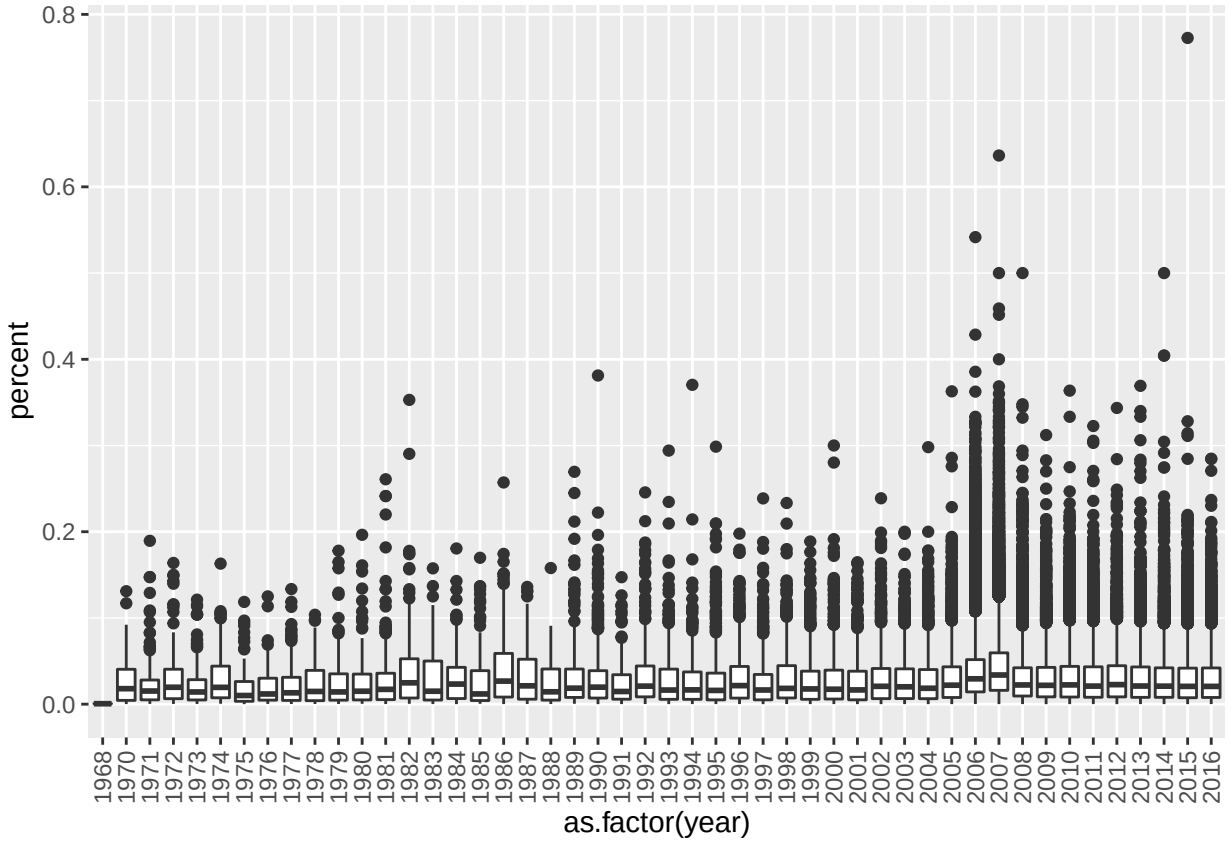
```
    mutate(percent = n / total_words,
           year = 10 * floor(year / 10)) %>%
    ggplot( aes(x = as.factor(year), y = percent) ) +
    geom_boxplot()
```



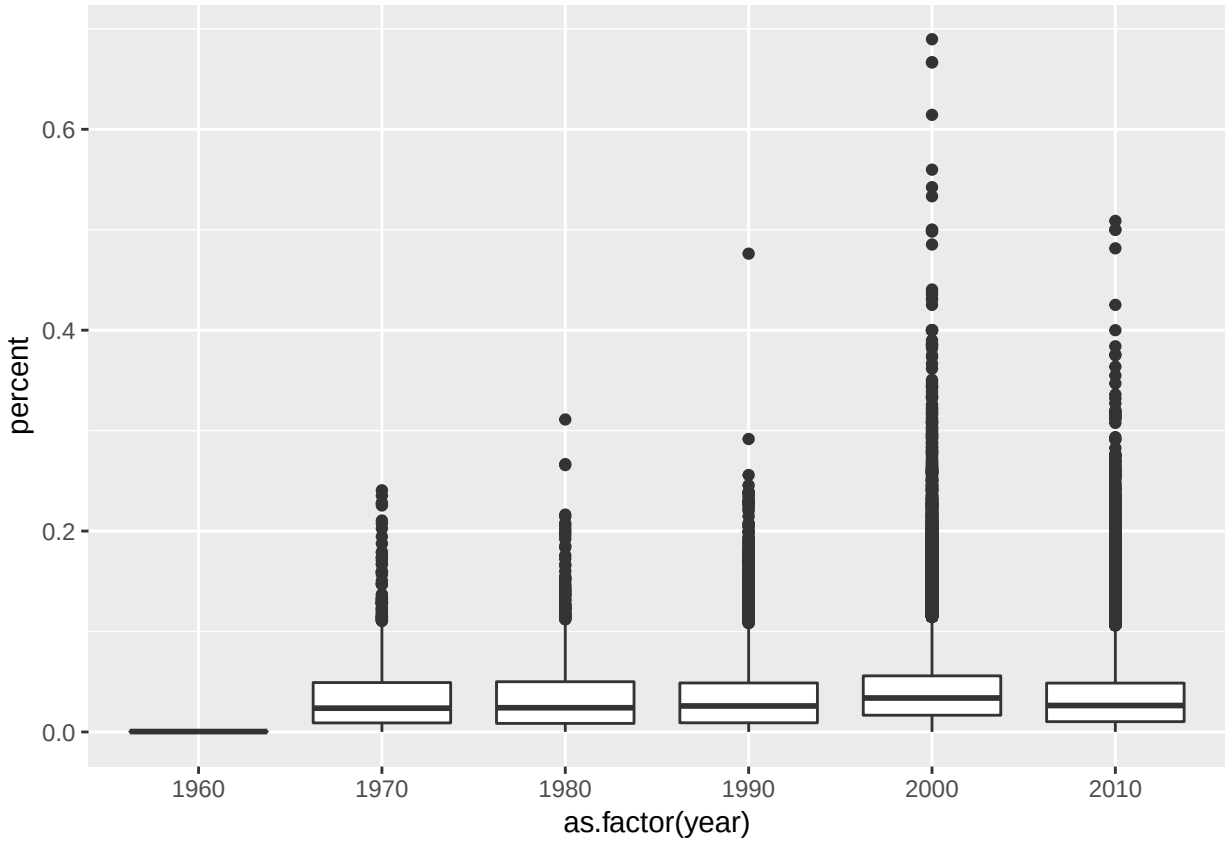

```
# Odd years, okay. Not sure what exactly that might be, early musical period would
# include 702 but not 67 and 112.


# Blocks of 10-years
lyric_sentiment %>%
    filter(sentiment == "negative", total_words > 20, year > 1000) %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words,
           year = 10 * floor(year / 10)) %>%
    ggplot( aes(x = as.factor(year), y = percent) ) +
    geom_boxplot()
```



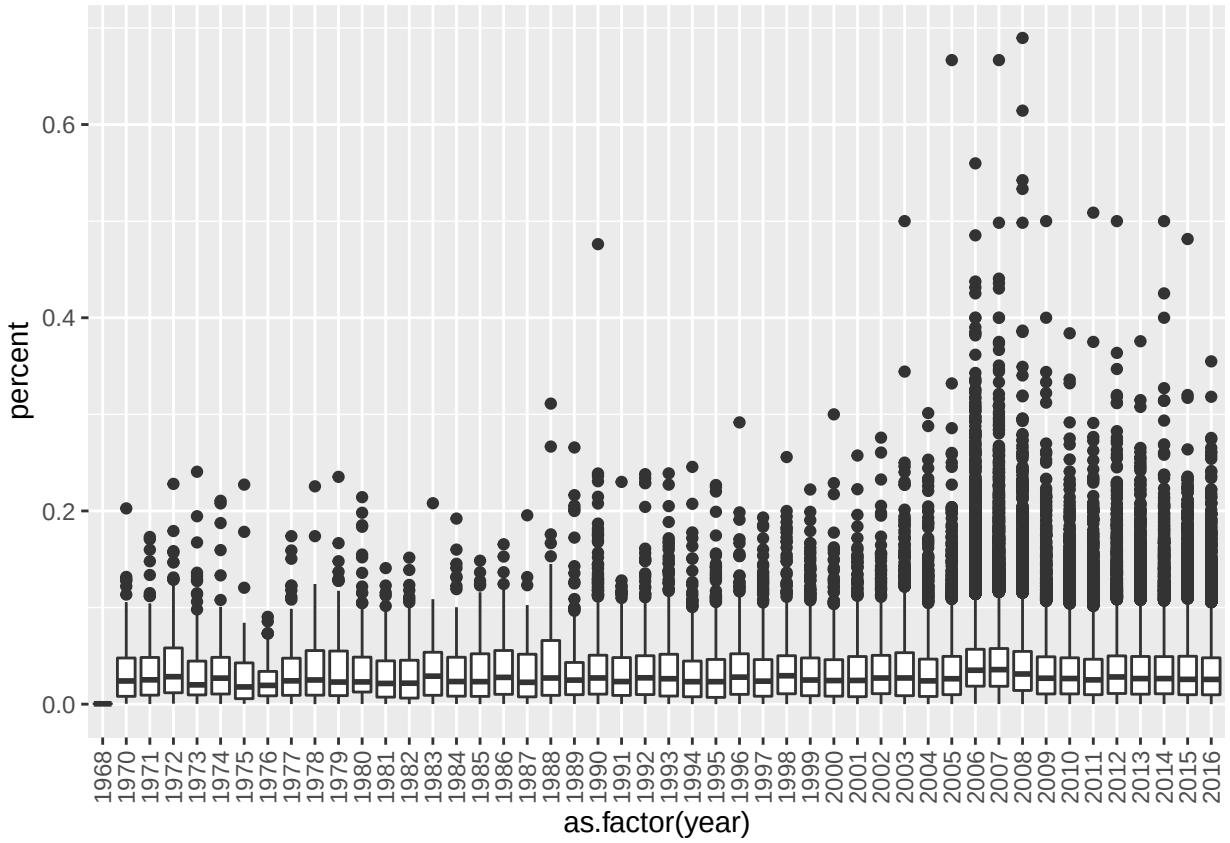

```
lyric_sentiment %>%
    filter(sentiment == "negative", total_words > 20, year > 1000) %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words) %>%
    ggplot( aes(x = as.factor(year), y = percent) ) +
    geom_boxplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```




```
# How is positive sentiment changing over time?

# Blocks of 10-years
lyric_sentiment %>%
    filter(sentiment == "positive", total_words > 20, year > 1000) %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words,
           year = 10 * floor(year / 10)) %>%
    ggplot( aes( x = as.factor(year) , y = percent ) ) +
    geom_boxplot()
```




```
lyric_sentiment %>%
    filter(sentiment == "positive", total_words > 20, year > 1000) %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    mutate(percent = n / total_words) %>%
    ggplot( aes( x = as.factor(year) , y = percent ) ) +
    geom_boxplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```



Model Fitting

```
# Do negative sentiments change over years?

negative_by_year <- lyric_sentiment %>%
    # Filter for negative words
    filter(sentiment == "negative") %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    # Percent
    mutate( percent = n / total_words)

model_negative <- lm(percent ~ year, data = negative_by_year)

# Results of the model fitting
summary(model_negative)
```

```
##
## Call:
## lm(formula = percent ~ year, data = negative_by_year)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06441 -0.02429 -0.00914  0.01377  0.96473
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.000e-02  1.332e-02   5.254 1.49e-07 ***
## year        -1.730e-05  6.635e-06  -2.608  0.00911 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03398 on 212427 degrees of freedom
## Multiple R-squared:  3.201e-05,  Adjusted R-squared:  2.731e-05
## F-statistic: 6.801 on 1 and 212427 DF,  p-value: 0.009112
```

```
# Do positive sentiments change over years?

positive_by_year <- lyric_sentiment %>%
    # Filter for positive words
    filter(sentiment == "positive") %>%
    count(song, year, total_words) %>%
    ungroup() %>%
    # Percent
    mutate( percent = n / total_words)


model_positive <- lm(percent ~ year, data = positive_by_year)

# Results of the model fitting
summary(model_positive)
```

```
##
## Call:
## lm(formula = percent ~ year, data = positive_by_year)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19669 -0.03730 -0.01972  0.00354  0.94898
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.135e-01  4.409e-02   4.843 1.28e-06 ***
## year        -8.061e-05  2.196e-05  -3.671 0.000242 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1131 on 217335 degrees of freedom
## Multiple R-squared:  6.199e-05,  Adjusted R-squared:  5.739e-05
## F-statistic: 13.47 on 1 and 217335 DF,  p-value: 0.0002419
```

Both negative and positive sentiments have significantly changed over the years. Though how should we examine outliers in this kind of a analysis?

```
# library(data.table)
#
#
# system.time(read_csv("lyrics.csv"))
# system.time(fread("lyrics.csv"))

# Wow, that's faster
```