Module 4 Assignment - Sentiment Analysis

Sentiment Analysis of Prince Lyrics

Load the required libraries

Read in the data

read-in the following data set in R: prince_text.csv (the data set is available on Canvas).The Data is the result of scraping Billboard Chart information and Prince lyrics from various sites.

Modify the dataset

Tokenize and preprocess text

1. Calculate the Match Ratio between the tidy text data frame and the three lexicons bing, NRC, and Afinn. Which lexicon has more of the distinct words from the lyrics? (10 points)

The NRC lexicon (match ratio of 0.20697790) has more of the distinct words from the reviews than AFINN (match ratio of 0.09585299) or Bing (match ratio of 0.14713186).

```
# Use an inner_join() between tidy_text and new_sentiments and then group by
lexicon.
##note: The NRC lexicon has 10 different categories, and a word may appear in
more than one category: that is, words can be negative and sad. That means
that you'll want to use n_distinct() in summarise() to get the distinct word
count per lexicon.

bing<-get_sentiments("bing")
nrc<-get_sentiments("nrc")
afinn<-get_sentiments("afinn")

#convert the values in the afinn lexicon to positive and negative sentiments
afinn_neg_pos <- afinn %>%
  mutate( sentiment = ifelse( value >= 0, "positive",
                              ifelse( value < 0,
                                      "negative", value)))
afinn_neg_pos <-afinn_neg_pos %>%
  select(word, sentiment)


#Combine the three lexicons
sentiments <-bind_rows(list(bing=bing,nrc=nrc,afinn=afinn_neg_pos),.id =
```

```
"lexicon")


new_sentiments <- sentiments %>%
  group_by(lexicon) %>%
  mutate(words_in_lexicon = n_distinct(word)) %>%
  ungroup()


tidy_prince %>%
  mutate(words_in_reviews = n_distinct(word)) %>%
  inner_join(new_sentiments) %>%
  group_by(lexicon,words_in_reviews, words_in_lexicon) %>%
  summarise(lex_match_words = n_distinct(word)) %>%
  ungroup() %>%
  mutate(total_match_words = sum(lex_match_words), #Not used but good to have
         match_ratio = lex_match_words / words_in_reviews) %>%
  select(lexicon, lex_match_words,  words_in_reviews, match_ratio)
```

```
## Warning in inner_join(., new_sentiments): Detected an unexpected many-to-
many relationship between `x` and `y`.
## ℹ Row 1 of `x` matches multiple rows in `y`.
## ℹ Row 3857 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## # A tibble: 3 × 4
##   lexicon lex_match_words words_in_reviews match_ratio
##   <chr>             <int>            <int>       <dbl>
## 1 afinn               772             7879      0.0980
## 2 bing               1186             7879      0.151
## 3 nrc                1673             7879      0.212
```

```
# Error: Detected an unexpected many-to-many relationship between `x` and
`y`.
```

```
head(tidy_prince)
```

```
## # A tibble: 6 × 9
##   song   year album   peak US.Pop US.R.B decade chart_level word
##   <chr> <dbl> <chr>  <dbl> <chr>  <chr>  <chr>  <chr>       <chr>
## 1 7      1992 Symbol     3 7      61     1990s  Top 10      watch
## 2 7      1992 Symbol     3 7      61     1990s  Top 10      fall
## 3 7      1992 Symbol     3 7      61     1990s  Top 10      stand
## 4 7      1992 Symbol     3 7      61     1990s  Top 10      love
## 5 7      1992 Symbol     3 7      61     1990s  Top 10      smoke
## 6 7      1992 Symbol     3 7      61     1990s  Top 10      intellect
```

2. Sentiment analysis (10 points)

Implement sentiment analysis using the inner join function and the "nrc" lexicon by performing an inner_join() on the get_sentiments() function.

```
prince_nrc <- tidy_prince %>%
  inner_join(get_sentiments("nrc"))       # valued from -5 to +5
```

```
## Warning in inner_join(., get_sentiments("nrc")): Detected an unexpected
many-to-many relationship between `x` and `y`.
## ℹ Row 1 of `x` matches multiple rows in `y`.
## ℹ Row 7729 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
head(prince_nrc) # tidyprince now has nrc sentiments also
```

```
## # A tibble: 6 × 10
##   song   year album   peak US.Pop US.R.B decade chart_level word
sentiment
##   <chr> <dbl> <chr>  <dbl> <chr>  <chr>  <chr>  <chr>       <chr> <chr>
## 1 7      1992 Symbol     3 7      61     1990s  Top 10      watch
anticipation
## 2 7      1992 Symbol     3 7      61     1990s  Top 10      watch fear
## 3 7      1992 Symbol     3 7      61     1990s  Top 10      fall  negative
## 4 7      1992 Symbol     3 7      61     1990s  Top 10      fall  sadness
## 5 7      1992 Symbol     3 7      61     1990s  Top 10      love  joy
## 6 7      1992 Symbol     3 7      61     1990s  Top 10      love  positive
```

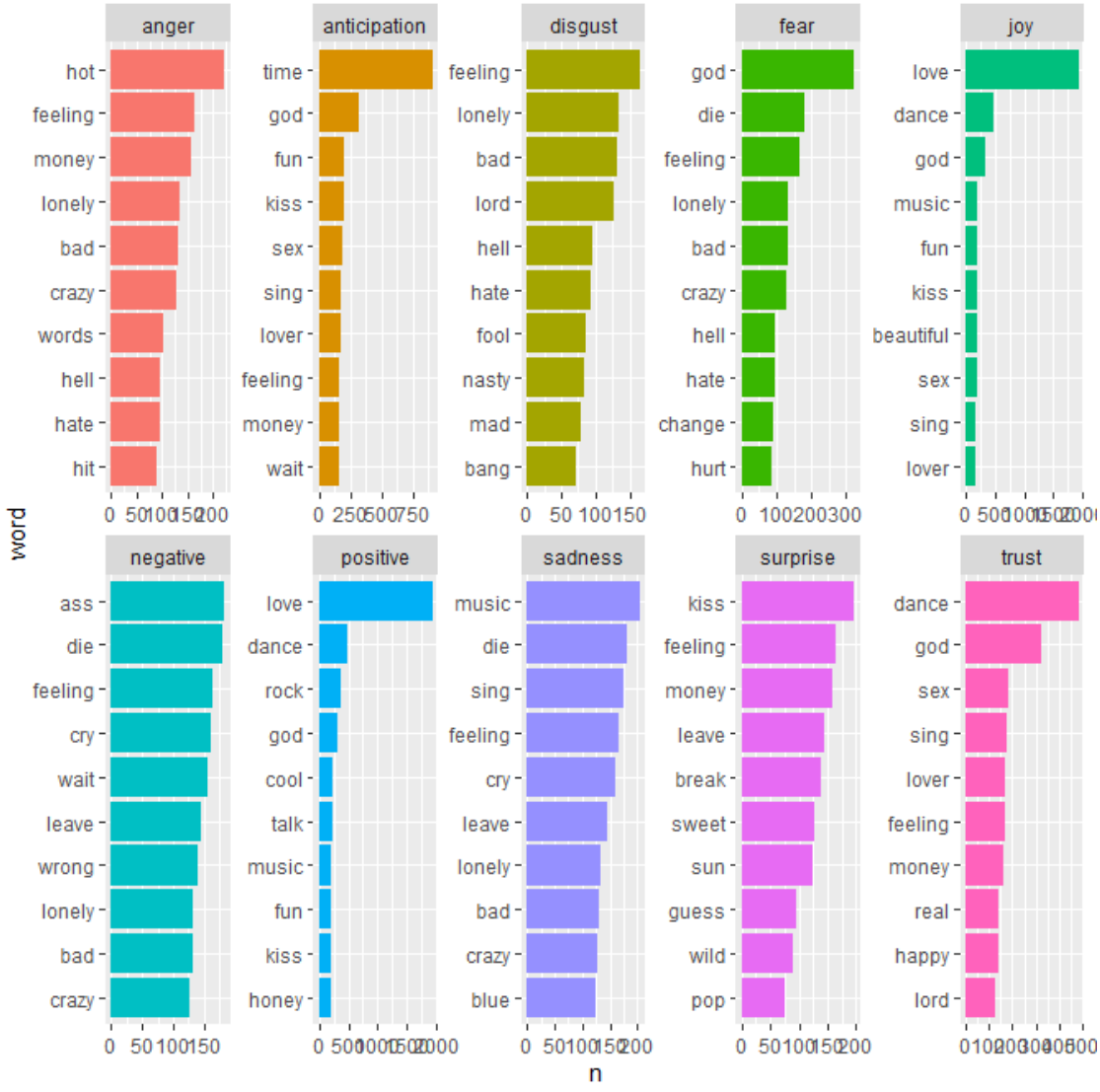
3. Which words contribute to the sentiment scores? (10 points)

It's important to understand which words specifically are driving sentiment scores, and since we are using tidy data principles, it's not too difficult to check.

Count by word and sentiment to find which words are contributing most overall to the sentiment scores. Group by sentiment. Take the top 10 words for each sentiment using top_n(). Set up the plot using aes(), with the words on the x-axis, the number of uses n on the y-axis, and fill corresponding to sentiment. Explain the results.

```
review_nrc <- prince_nrc %>%
  # Count by word and sentiment
      inner_join(get_sentiments("nrc")) %>%         # valued from -5 to +5
    count(word,sentiment,sort = TRUE) #%>%

review_nrc  %>%
  # Group by sentiment
    group_by(sentiment)%>%

  # Take the top 10 words for each sentiment
    top_n(10)%>%
```

```
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%

 # Set up the plot with aes() using the ggplot() and geom_col(). set the
graph aes so x is word and y is n and the columns are filled with sentiment.
  ggplot(aes(word,n, fill=sentiment)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~ sentiment, ncol =5, scales = "free")+
    coord_flip()
```




```
head(review_nrc)
```

```
## # A tibble: 6 × 3
##   word  sentiment        n
##   <chr> <chr>        <int>
## 1 love  joy           1937
```

```
## 2 love  positive      1937
## 3 time  anticipation   907
## 4 dance joy            484
## 5 dance positive       484
## 6 dance trust          484
```

4. Which song uses the most positive words? (15 points)

Make a new column called song_total in the dataframe that tallies the total number of words from each song; the mutate() verb will make a new column and the function n() counts the number of observations in the current group:mutate(song_total=n()).

Define a new column percent using mutate() that is n divided by song_total, the proportion of words that belong to that sentiment. Filter only for the positive sentiment rows. Arrange by percent so you can see the results sorted by proportion of positive words.Explain the results.

The results have 805 songs listed but it is actually looking at specific positive words in the songs. So, "rock hard in a funky place" is listed twice in the top 10 but it is referring to 2 different versions because the words per song are different.

```
prince_sentiment_song <- prince_nrc%>%
  group_by(song)%>%                # Group by song
  mutate(song_total=n()) %>%       # Define a new column song_total
  ungroup()                        # Ungroup

prince_sentiment_song %>%
   count(song, sentiment, song_total) %>%

 # Define a new column percent that is n divided by song_total
   mutate(percent =(n/song_total)) %>%

 # Filter only for positive words
   filter(sentiment == "positive") %>%

 # Arrange by percent
   arrange(desc(percent))
```

```
## # A tibble: 803 × 5
##    song                          sentiment song_total     n percent
##    <chr>                         <chr>          <int> <int>   <dbl>
##  1 jam of the year               positive          56    33   0.589
##  2 rock hard in a funky place    positive          96    56   0.583
##  3 the glamorous life            positive          51    29   0.569
##  4 rockhard in a funky place     positive          88    47   0.534
##  5 walk dont walk                positive          29    15   0.517
##  6 young and beautiful           positive          31    16   0.516
##  7 about prince  the black album positive           4     2   0.5
##  8 flutestramental               positive           2     1   0.5
##  9 in love                       positive          36    18   0.5
```

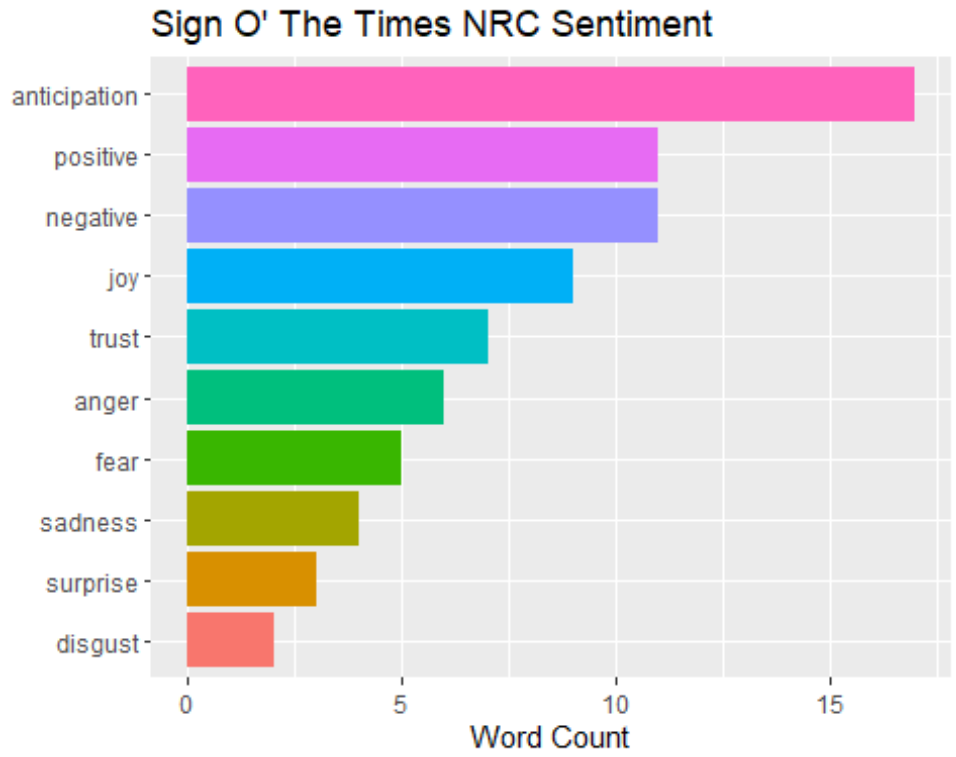
```
## 10 orgasm                          positive           2     1   0.5
## # … with 793 more rows
```

5.Sign O' the Times (15 points)

In 1987, Prince wrote a song called "Sign O' the Times". What is the mood of this song using the "nrc" lexicon? Visualize and explain the results. Try this for the song "so blue". what is the mood of that song?

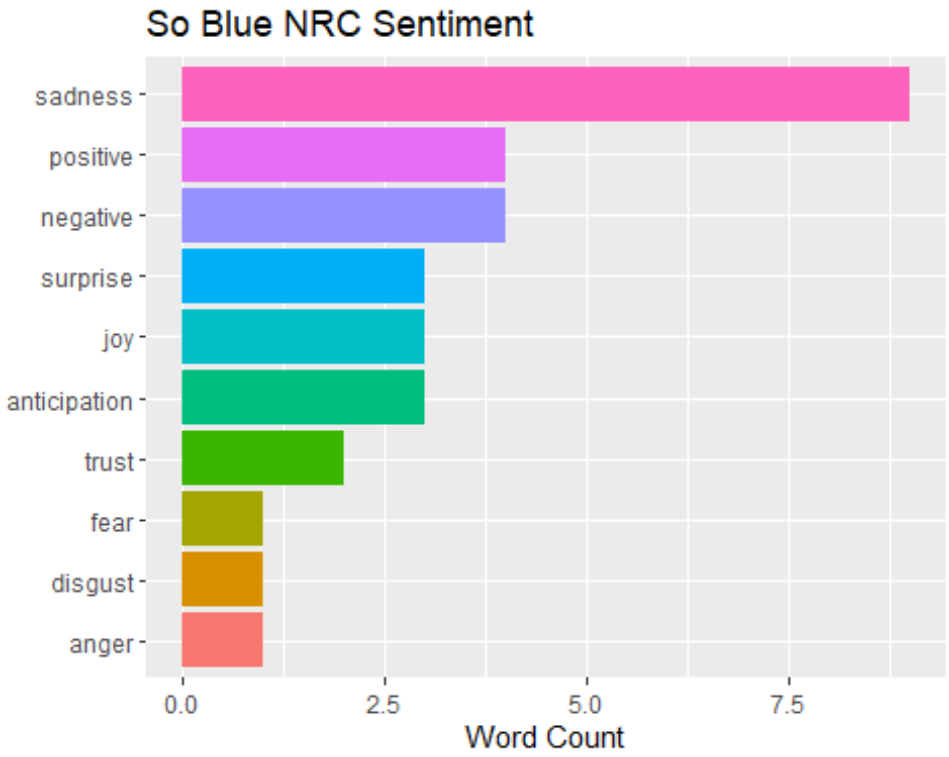
```
prince_nrc %>%
# Filter songs in "sign o the times"
filter(song %in% "sign o the times") %>%
# Group by sentiment
  group_by(sentiment)%>%
  summarise(word_count = n()) %>%
  ungroup() %>%
# Define a new column sentiment and reorder the sentiment based on the
word_count
  mutate(sentiment = reorder(sentiment, word_count)) %>%
# Visualize the results using ggplot() and geom_col(). set the graph aes so x
is sentiment and y is word_count and the columns are filled with word_count
  ggplot(aes(sentiment, word_count,fill=sentiment))  +
     geom_col(show.legend = FALSE) +
     guides(fill = FALSE) +
     labs(x = NULL, y = "Word Count") +
     ggtitle("Sign O' The Times NRC Sentiment") +
     coord_flip()
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use
"none" instead as
## of ggplot2 3.3.4.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



"Sign o' The Times" is a song that presents itself as contrasting the life and death moments in life. Anticipation is a strong element of the song as the lyrics question what will happen as time goes on. The lyrics refrain mention a negative thing ("a rocket blows"), with contrasting a positive thing ("everybody still wants to fly"). The setniment is pretty much a 50/50 split between positive sentiment and negative sentiment.

```
prince_nrc %>%
# Filter songs in "so blue"
filter(song %in% "so blue") %>%
# Group by sentiment
  group_by(sentiment)%>%
  summarise(word_count = n()) %>%
  ungroup() %>%
# Define a new column sentiment and reorder the sentiment based on the
word_count
  mutate(sentiment = reorder(sentiment, word_count)) %>%
# Visualize the results using ggplot() and geom_col(). set the graph aes so x
is sentiment and y is word_count and the columns are filled with word_count
  ggplot(aes(sentiment, word_count,fill=sentiment))  +
     geom_col(show.legend = FALSE) +

     guides(fill = FALSE) +
     labs(x = NULL, y = "Word Count") +
     ggtitle("So Blue NRC Sentiment") +
     coord_flip()
```



The song "So Blue" is a song about sadness that one person left another. Again, we see a contrast between what was positive ("You meant the world to me" and now its negative ("now you're gone and I'm so blue") because that good thing is gone. Overall, the sentiment is negative and shows feelings that are negative.

6. Polarity by chart level (15 points)

Break down your analysis to the chart level using the Bing lexicon. Create a graph of the polar sentiment per chart level. Use spread() to separate the sentiments into columns and mutate() to create a polarity (positive - negative) field and a percent_positive field (positive/totalsentiment∗100), for a different perspective.

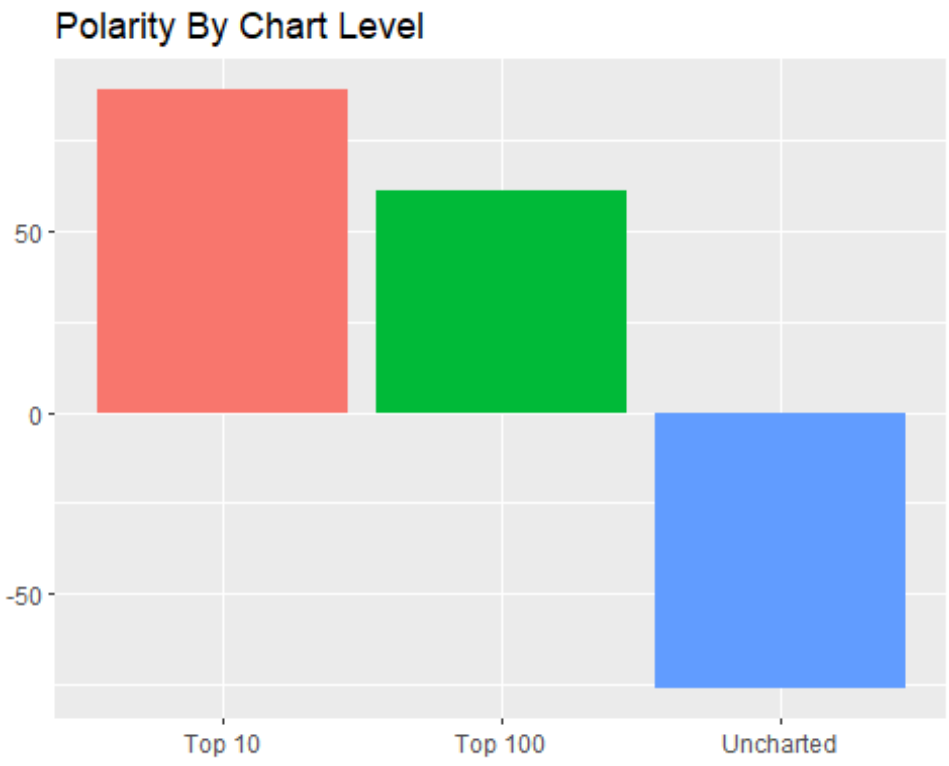
```
#Implement sentiment analysis using the inner join function and the "bing"
lexicon
prince_bing <- tidy_prince %>%
  inner_join(get_sentiments("bing"))

prince_polarity_chart <- prince_bing %>%

   count(sentiment, chart_level) %>% #Count sentiment by chart_level

   #Use spread() to separate the sentiments into columns
   spread(sentiment, n, fill = 0) %>%

   #Use mutate() to create a polarity (positive - negative) field and a
percent_positive field (positive / (positive + negative) * 100)
```

```
   mutate(polarity = positive - negative) %>%
   mutate(percent_positive = (positive / (positive + negative) * 100))

prince_polarity_chart
```

```
## # A tibble: 3 × 5
##   chart_level negative positive polarity percent_positive
##   <chr>          <dbl>    <dbl>    <dbl>            <dbl>
## 1 Top 10           351      440       89             55.6
## 2 Top 100          174      235       61             57.5
## 3 Uncharted       6995     6919      -76             49.7
```

Visualize the results.

```
prince_polarity_chart %>%

# Visualize the results using ggplot() and geom_col(). set the graph aes so x
is chart_level and y is polarity and the columns are filled with chart_level
  ggplot(aes(chart_level, polarity, fill=chart_level))  +
     geom_col(show.legend = FALSE) +
     xlab(NULL) +
     ylab(NULL) +
     ggtitle("Polarity By Chart Level")
```



Does this say that charted songs are typically more positive than negative? If so, what does this tell you about what society wants to hear? Can you even make these assumptions? Note that the Bing lexicon itself has more negative than positive words.

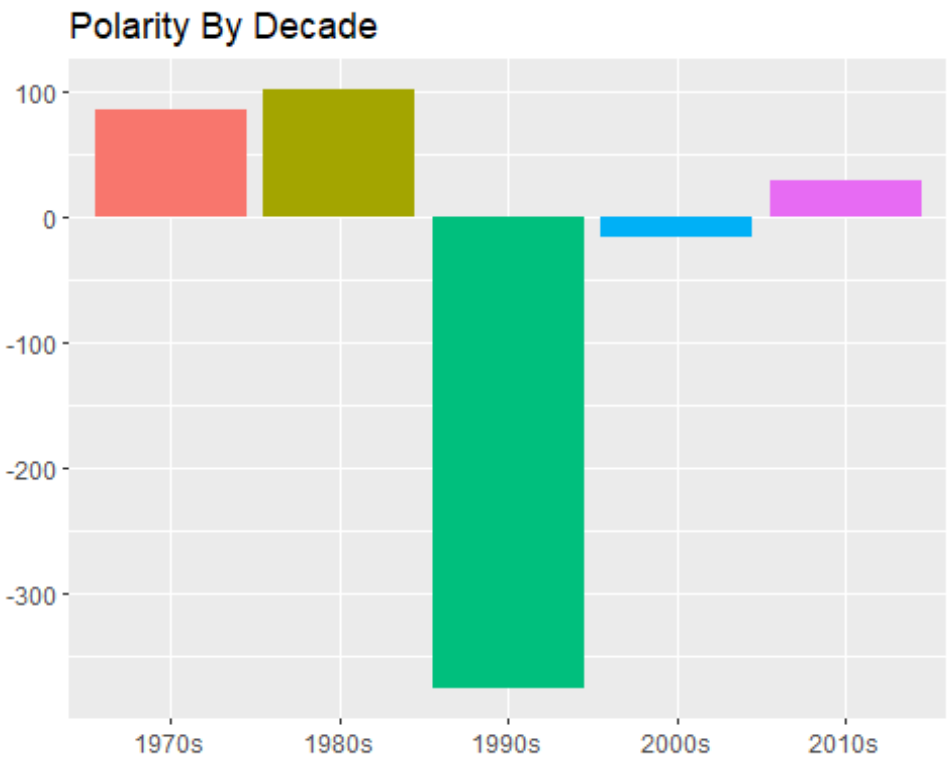
It says that listeners want to hear more positive words. The top 100 is simply more positive. The uncharted are more negative.

7. Polarity by decades (15 points)

Break down your analysis to the decades using the Bing lexicon. Create a graph of the polar sentiment per decade. Use spread() to separate the sentiments into columns and mutate() to create a polarity (positive - negative) field and a percent_positive field (positive/totalsentiment∗100), for a different perspective.

Visualize and explain the results.

```
prince_polarity_decade %>%
# Visualize the results using ggplot() and geom_col(). set the graph aes so x
is decade and y is polarity and the columns are filled with decade
  ggplot(aes(decade, polarity, fill=decade))  +
     geom_col(show.legend = FALSE) +
     xlab(NULL) +
     ylab(NULL) +
     ggtitle("Polarity By Decade")
```



The results of the “Polarity by Decade” chart shows that Prince’s primetime was in the 1970’s and 1980’s. The number of songs released in his name also decreased after the 1990’s. However, during teh 1990’s there were many song collaborations and film soundtracks that Prince worked on such as a live-action film by Tim Burton called “Batman” (all taken from Wikipedia).

A list of Prince's life events is attached, collected from popular sources such as Rolling Stone Magazine, Biography.com, etc. Compare Prince's life events with the sentiment.

In the 1990's, he fought with his record label over which songs to release when as well as confusion over what to call him. His very negative and "evil" album (according to him) was finally released in the 1990's as well. Most of his albums saw poor sales during this time, compared to the previous decades. In the late 1990's he also began singing cover songs rather than using just his own lyrics.

8. Reflect on this assignment (10 points)

a) What have you learned from this assignment?

This assignment has taught me that I need to break my given goals into smaller chunks. I am so used to working with object-oriented programming that it is hard to see the individual instructions.

I also learned how to compare the 3 given sentiment dictionaries. Each was created for a given purpose so they do not all apply well to each corpus.

b) What else do you want to know using this dataset?

I wanted to know how to color positive sentiments in a happier shade and the negative sentiments into a sadder shade. This would require individually coloring the sentiments but I am unsure where this would occur.