Sentiment Analysis

Harini

02/01/2021

INTRODUCTION

Sentiment analysis is a type of text mining which aims to determine the opinion and subjectivity of its content.When applied to lyrics, the results can be representative of not only the artist's attitudes, but can also reveal pervasive, cultural influences. There are different methods used for sentiment analysis, including training a known dataset, creating own classifiers with rules, and using predefined lexical dictionaries (lexicons). In this project we focus on lexicon-based approach. There are different levels of analysis based on the text.These levels are typically identified as document, sentence, and word. In lyrics, the document could be defined as sentiment per decade, year, chart-level, or song. The sentence level is not usually an option with lyrics as punctuation can detract from rhymes and patterns.

```
## Rows: 824
## Columns: 11
## $ X           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ...
## $ lyrics      <chr> "all 7 and we will watch them fall they stand in the wa...
## $ song        <chr> "7", "319", "1999", "2020", "3121", "7779311", "u", "ed...
## $ year        <int> 1992, NA, 1982, NA, 2006, NA, NA, NA, NA, NA, NA, NA, N...
## $ album       <chr> "Symbol", NA, "1999", "Other Songs", "3121", NA, NA, NA...
## $ peak        <int> 3, NA, 2, NA, 1, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ us_pop      <chr> "7", NA, "12", NA, "1", NA, NA, NA, NA, NA, NA, NA, NA,...
## $ us_rnb      <chr> "61", NA, "4", NA, "1", NA, NA, NA, NA, NA, NA, NA, NA,...
## $ decade      <chr> "1990s", NA, "1980s", NA, "2000s", NA, NA, NA, NA, NA, ...
## $ chart_level <chr> "Top 10", "Uncharted", "Top 10", "Uncharted", "Top 10",...
## $ charted     <chr> "Charted", "Uncharted", "Charted", "Uncharted", "Charte...
```

prince_data is a data frame of 824 songs and 10 columns. This means that a record is a song. ## DATA CLEANING Remove undesirable words (manual list of unnecessary words),stop words (overly common words such as "and", "the", "a", "of", etc.), words with fewer than three characters (often used for phonetic effect in music) and Split the lyrics into individual words. In order to turn raw data into a tidy format, use unnest_tokens() from tidytext to create prince_tidy which breaks out the lyrics into individual words with one word per row.Then use anti_join() and filter() from dplyr for the remaining cleaning steps.

```
undesirable_words <- c("prince", "chorus", "repeat", "lyrics",
                       "theres", "bridge", "fe0f", "yeah", "baby",
                       "alright", "wanna", "gonna", "chorus", "verse",
                       "whoa", "gotta", "make", "miscellaneous", "2",
                       "4", "ooh", "uurh", "pheromone", "poompoom", "3121",
                       "matic", " ai ", " ca ", " la ", "hey", " na ",
                       " da ", " uh ", " tin ", "  ll", "transcription",
                       "repeats", "la", "da", "uh", "ah")

#Create tidy text format: Unnested, Unsummarized, -Undesirables, Stop and Short words
prince_tidy <- prince_data %>%
  unnest_tokens(word, lyrics) %>% #Break the lyrics into individual words
  filter(!word %in% undesirable_words) %>% #Remove undesirables
  filter(!nchar(word) < 3) %>% #Words like "ah" or "oo" used in music
  anti_join(stop_words) #Data provided by the tidytext package
```

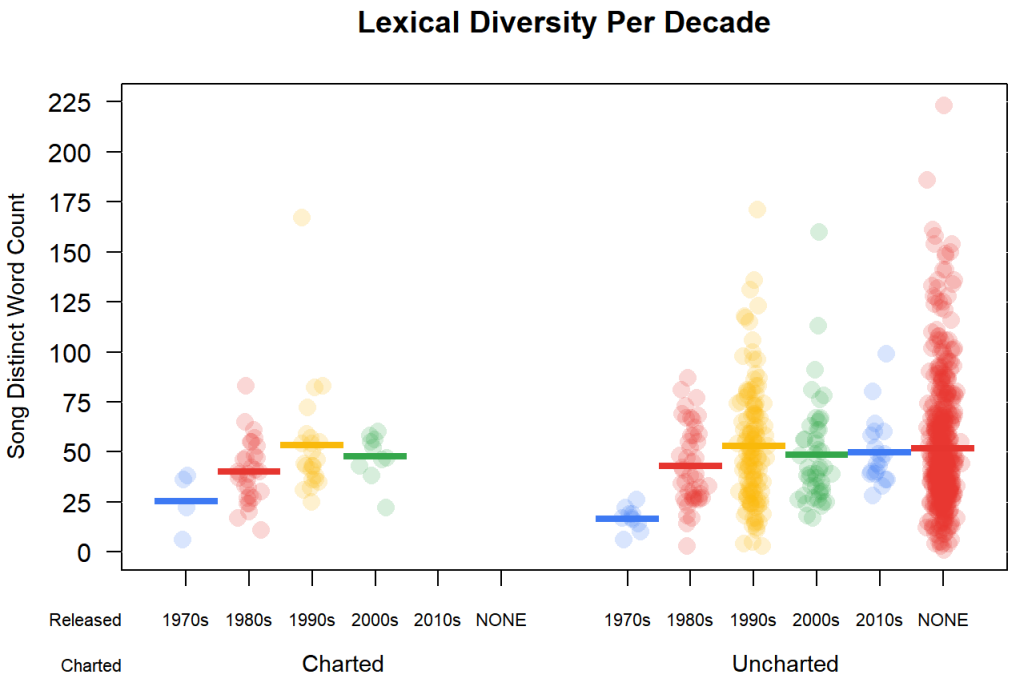
```
## Joining, by = "word"
```

```
glimpse(prince_tidy)
```

```
## Rows: 76,116
## Columns: 11
## $ X           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ song        <chr> "7", "7", "7", "7", "7", "7", "7", "7", "7", "7", "7", ...
## $ year        <int> 1992, 1992, 1992, 1992, 1992, 1992, 1992, 1992, 1992, 1...
## $ album       <chr> "Symbol", "Symbol", "Symbol", "Symbol", "Symbol", "Symb...
## $ peak        <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3...
## $ us_pop      <chr> "7", "7", "7", "7", "7", "7", "7", "7", "7", "7", "7", ...
## $ us_rnb      <chr> "61", "61", "61", "61", "61", "61", "61", "61", "61", "...
## $ decade      <chr> "1990s", "1990s", "1990s", "1990s", "1990s", "1990s", "...
## $ chart_level <chr> "Top 10", "Top 10", "Top 10", "Top 10", "Top 10", "Top ...
## $ charted     <chr> "Charted", "Charted", "Charted", "Charted", "Charted", ...
## $ word        <chr> "watch", "fall", "stand", "love", "smoke", "intellect",...
```

prince_tidy, is now in a tokenized format with one word per row along with the song from which it came.Its a data frame of 76116 words and 10 columns. ## DESCRIPTIVE STATISTICS ## Shipshape: Word Count Per Song A pirate would say shipshape when everything is in good order, tidy and clean. A pirate plot is an advanced method of plotting a continuous dependent variable, such as the word count, as a function of a categorical independent variable, like decade. This combines raw data points, descriptive and inferential statistics into a single effective plot.It uses pirateplot() from the yarrr package. Create the word_summary data frame that calculates the distinct word count per song. The more diverse the lyrics, the larger the vocabulary. Reset the decade field to contain the value “NONE” for songs without a release date and relabel those fields with cleaner labels using select().

```
word_summary <- prince_tidy %>%
  mutate(decade = ifelse(is.na(decade),"NONE", decade)) %>%
  group_by(decade, song) %>%
  mutate(word_count = n_distinct(word)) %>%
  select(song, Released = decade, Charted = charted, word_count) %>%
  distinct() %>% #To obtain one record per song
  ungroup()

pirateplot(formula =  word_count ~ Released + Charted, #Formula
   data = word_summary, #Data frame
   xlab = NULL, ylab = "Song Distinct Word Count", #Axis labels
   main = "Lexical Diversity Per Decade", #Plot title
   pal = "google", #Color scheme
   point.o = .2, #Points
   avg.line.o = 1, #Turn on the Average/Mean line
   theme = 0, #Theme
   point.pch = 16, #Point `pch` type
   point.cex = 1.5, #Point size
   jitter.val = .1, #Turn on jitter to see the songs better
   cex.lab = .9, cex.names = .7) #Axis label size
```



Every colored circle in this pirate plot represents a song. The dense red area with the “NONE” value shows that a large number of songs in the dataset do not have a

release date. There is a slight upward trend in the unique number of words per song in the early decades: the solid horizontal line shows the mean word count for that decade. The words become more illuminating throughout Prince's career. ## All Year Round: Song Count Per Year Circular graphs are a unique way to visualize complicated relationships among several categories. The graph below is simply a circular bar chart using coord_polar() from ggplot2 that shows the relative number of songs per year.

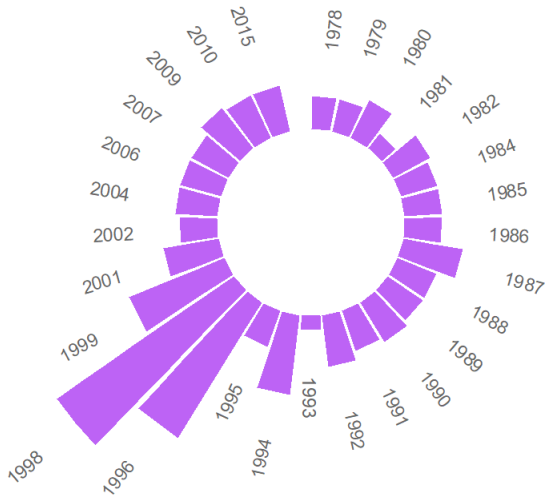
```
songs_year <- prince_data %>%
  select(song, year) %>%
  group_by(year) %>%
  summarise(song_count = n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
id <- seq_len(nrow(songs_year))
songs_year <- cbind(songs_year, id)
label_data = songs_year
number_of_bar = nrow(label_data) #Calculate the ANGLE of the labels
angle = 90 - 360 * (label_data$id - 0.5) / number_of_bar #Center things
label_data$hjust <- ifelse(angle < -90, 1, 0) #Align label
label_data$angle <- ifelse(angle < -90, angle + 180, angle) #Flip angle
ggplot(songs_year, aes(x = as.factor(id), y = song_count)) +
  geom_bar(stat = "identity", fill = alpha("purple", 0.7)) +
  geom_text(data = label_data, aes(x = id, y = song_count + 10, label = year, hjust = hjust), color = "blac
k", alpha = 0.6, size = 3, angle =  label_data$angle, inherit.aes = FALSE ) +
  coord_polar(start = 0) +
  ylim(-20, 150) + #Size of the circle
  theme_minimal() +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank(),
        plot.margin = unit(rep(-4,4), "in"),
        plot.title = element_text(margin = margin(t = 10, b = -10)))
```

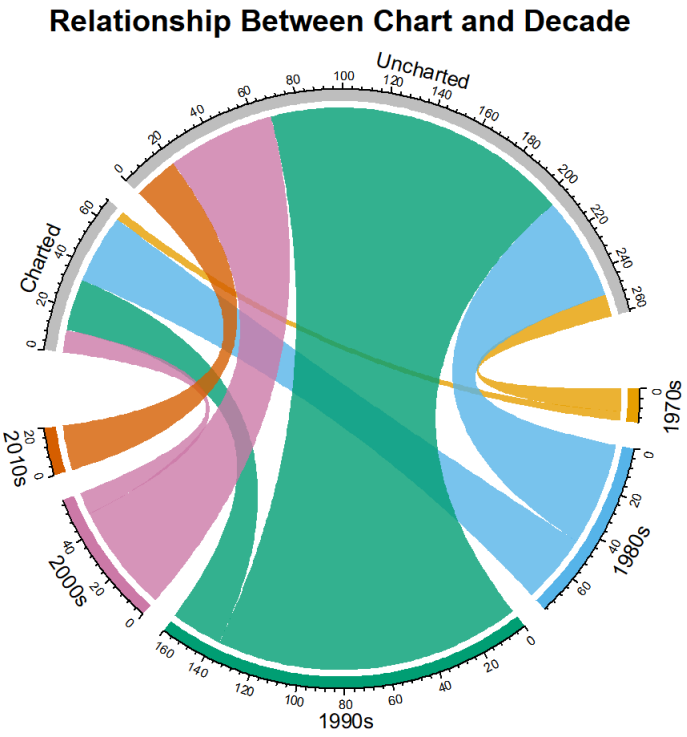
```
## Warning: Removed 1 rows containing missing values (position_stack).
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```



Gap is an indicator of the hundreds of songs without release dates.The missing years indicate those where no songs were released. The most prolific years were 1996 and 1998. ## Chords: Charted Songs By Decade The following graph shows the relationship between the decade a song was released and whether or not it hit the Billboard charts.Using a chordDiagram() for musical analysis just seemed appropriate! This graphical tool is from the beautiful circlize package by Zuguang Gu. The graph is split into two categories: charted (top), and decade (bottom). The two categories are separated by wide gaps, with smaller gaps between the values.

```
decade_chart <-  prince_data %>%
  filter(decade != "NA") %>% #Remove songs without release dates
  count(decade, charted)  #Get SONG count per chart level per decade. Order determines top or bottom.

circos.clear() #Very important - Reset the circular layout parameters!
grid.col = c("1970s" = my_colors[1], "1980s" = my_colors[2], "1990s" = my_colors[3], "2000s" = my_colors[4]
, "2010s" = my_colors[5], "Charted" = "grey", "Uncharted" = "grey") #assign chord colors
# Set the global parameters for the circular layout. Specifically the gap size
circos.par(gap.after = c(rep(5, length(unique(decade_chart[[1]])) - 1), 15,
                         rep(5, length(unique(decade_chart[[2]])) - 1), 15))

chordDiagram(decade_chart, grid.col = grid.col, transparency = .2)
title("Relationship Between Chart and Decade")
```



Chord chart nicely illustrates

the counts of songs per decade, per chart level. Prince began his career in the 1970s with only a few releases, some of which charted. There were only a few commercially successful songs in the 2000s and in the 2010s there were no hit songs. ## Lexicons and Lyrics The tidytext package includes a dataset called sentiments which provides several distinct lexicons. These lexicons are dictionaries of words with an assigned sentiment category or value. tidytext provides three general purpose lexicons:

AFINN: assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment Bing: assigns words into positive and negative categories NRC: assigns words into one or more of the following ten categories: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. In order to examine the lexicons, create a data frame called new_sentiments. Filter out a financial lexicon, create a binary (also described as polar) sentiment field for the AFINN lexicon by converting the numerical score to positive or negative, and add a field that holds the distinct word count for each lexicon. new_sentiments has one column with the different sentiment categories, so for a better view of the word counts per lexicon, per category, use spread() from tidyr to pivot those categories into separate fields.The table above gives an idea of the size and structure of each lexicon.

Lyrics Found In Lexicons

Use an inner_join() between prince_tidy and new_sentiments and then group by lexicon. 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. spread() from tidyr to pivot those categories into separate fields.

The NRC lexicon has more of the distinct words from the lyrics than AFINN or Bing. Notice the sum of the match ratios is low. No lexicon could have all words, nor should they. Many words are considered neutral and would not have an associated sentiment. ## DATA PREPARATION Here are three techniques to consider before performing sentiment analysis:

1)Stemming: generally refers to removing suffixes from words to get the common origin 2)Lemmatization: reducing inflected (or sometimes derived) words to their word stem, base or root form 3)Word replacement: replace words with more frequently used synonyms The detail analysis is executed in follwing steps 1)Create lexicon-specific datasets 2)Look at polar sentiment across all songs 3)Examine sentiment change over time 4)Validate your results against specific events in Prince's life 5)Study song level sentiment 6)Review how pairs of words affect sentiment ## Create Sentiment Datasets Creating Prince sentiment datasets for each of the lexicons by performing an inner_join() on the get_sentiments() function. Pass the name of the lexicon for each call.Using Bing for binary and NRC for categorical sentiments. Since

words can appear in multiple categories in NRC, such as Negative/Fear or Positive/Joy.

```
prince_bing <- prince_tidy %>%
  inner_join(get_sentiments("bing"))
```

```
## Joining, by = "word"
```

```
prince_nrc <- prince_tidy %>%
  inner_join(get_sentiments("nrc"))
```

```
## Joining, by = "word"
```

```
prince_nrc_sub <- prince_tidy %>%
  inner_join(get_sentiments("nrc")) %>%
  filter(!sentiment %in% c("positive", "negative"))
```

```
## Joining, by = "word"
```

```
nrc_plot <- prince_nrc %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  #Use `fill = -word_count` to make the larger bars darker
  ggplot(aes(sentiment, word_count, fill = -word_count)) +
  geom_col() +
  guides(fill = FALSE) + #Turn off the legend
  theme_lyrics() +
  labs(x = NULL, y = "Word Count") +
  scale_y_continuous(limits = c(0, 15000)) + #Hard code the axis limit
  ggtitle("Prince NRC Sentiment") +
  coord_flip()
```

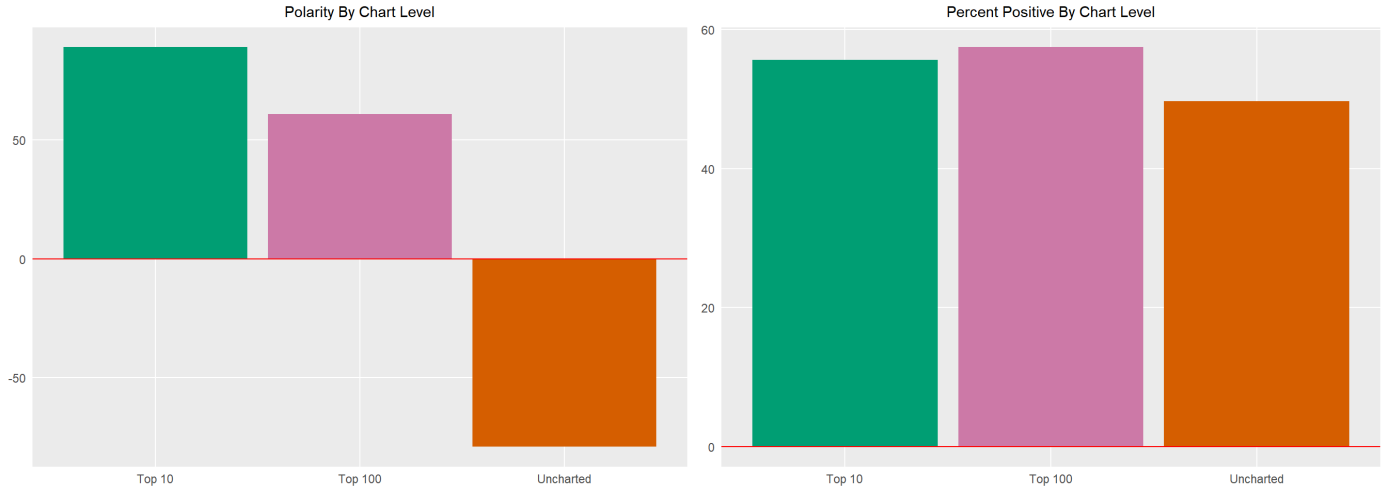
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

It appears that for Prince's lyrics, NRC strongly favors the positive. But are all words with a sentiment of disgust or anger also in the negative category as well.Now take a look at Bing overall sentiment. Of the 1185 distinct words from Prince's lyrics that appear in the Bing lexicon.

```
bing_plot <- prince_bing %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  ggplot(aes(sentiment, word_count, fill = sentiment)) +
  geom_col() +
  guides(fill = FALSE) +
  theme_lyrics() +
  labs(x = NULL, y = "Word Count") +
  scale_y_continuous(limits = c(0, 8000)) +
  ggtitle("Prince Bing Sentiment") +
  coord_flip()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

In acoustics, there is something called phase cancellation where the frequency of two instances of the same wave are exactly out of phase and cancel each other out, for example, when you're recording a drum with two mics and it takes the sound longer to get to one mic than the other. This results in total silence at that frequency! ## Chart Level 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. For the polarity graph, add a yintercept with geom_hline().



charted songs are typically more positive than negative. Looking at the positive sentiment relative to total sentiment, it seems like the charted songs are just slightly more positive than the negative. This is interesting given that the Bing lexicon itself has more negative than positive words. Use use geom_smooth() with the loess method for a smoother curve and another geom_smooth() with method = lm for a linear smooth curve for a polar perspective.

```
prince_polarity_year <- prince_bing %>%
  count(sentiment, year) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(polarity = positive - negative,
    percent_positive = positive / (positive + negative) * 100)

polarity_over_time <- prince_polarity_year %>%
  ggplot(aes(year, polarity, color = ifelse(polarity >= 0,my_colors[5],my_colors[4]))) +
  geom_col() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_smooth(method = "lm", se = FALSE, aes(color = my_colors[1])) +
  theme_lyrics() + theme(plot.title = element_text(size = 11)) +
  xlab(NULL) + ylab(NULL) +
  ggtitle("Polarity Over Time")

relative_polarity_over_time <- prince_polarity_year %>%
  ggplot(aes(year, percent_positive , color = ifelse(polarity >= 0,my_colors[5],my_colors[4]))) +
  geom_col() +
  geom_smooth(method = "loess", se = FALSE) +
  geom_smooth(method = "lm", se = FALSE, aes(color = my_colors[1])) +
  theme_lyrics() + theme(plot.title = element_text(size = 11)) +
  xlab(NULL) + ylab(NULL) +
  ggtitle("Percent Positive Over Time")

grid.arrange(polarity_over_time, relative_polarity_over_time, ncol = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

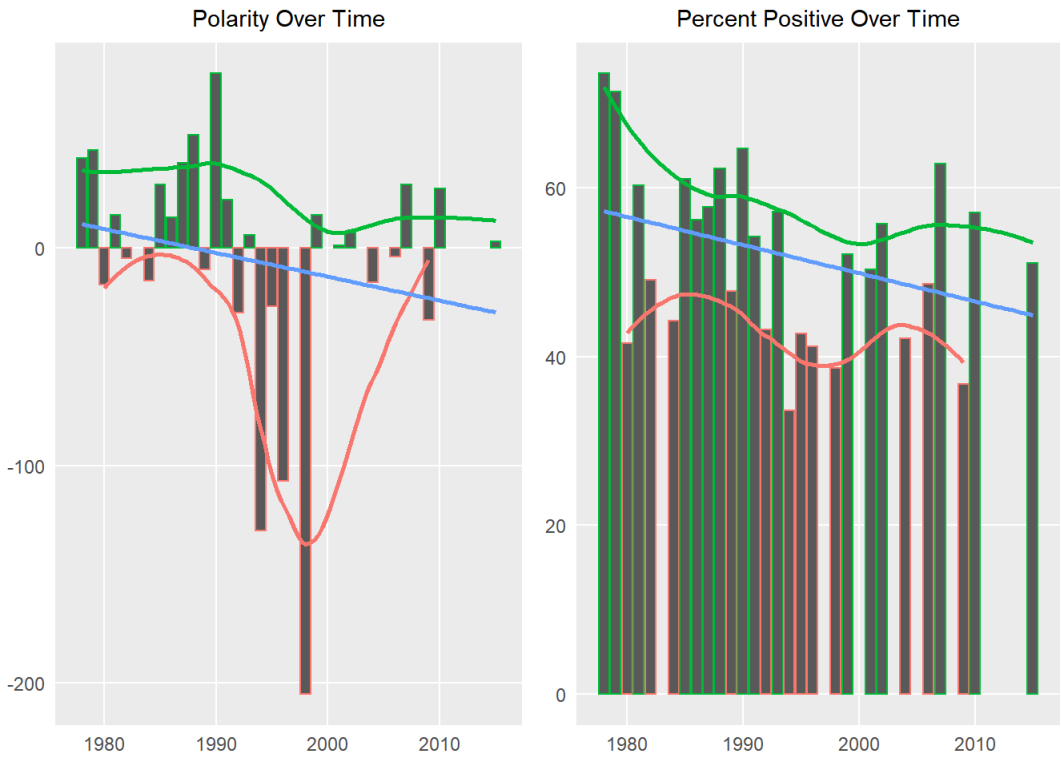
```
## Warning: Removed 1 rows containing missing values (position_stack).
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (position_stack).
```

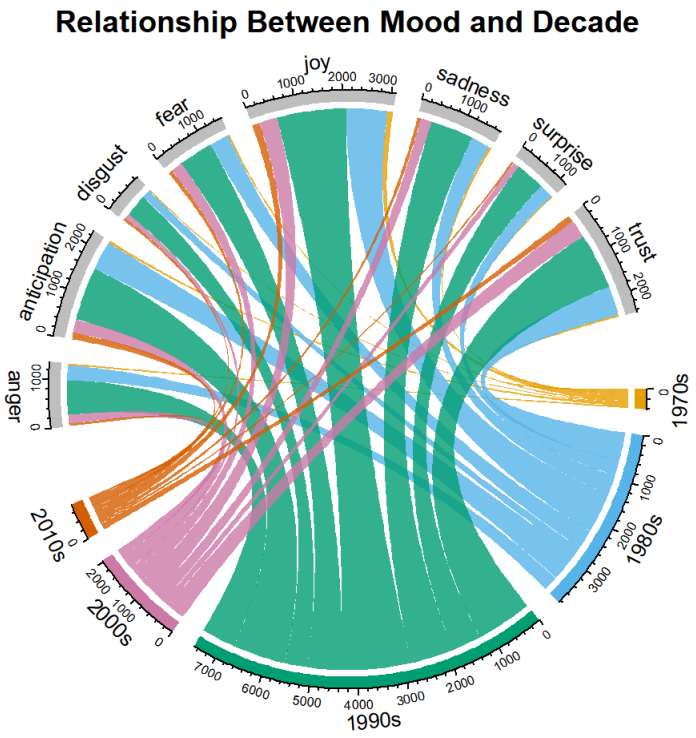


Mood Ring Using the power of the chordDiagram() to examine the relationships between NRC sentiments and decades. Note that sentiment categories appear on the top part of the ring and decades on the bottom.

```
grid.col = c("1970s" = my_colors[1], "1980s" = my_colors[2], "1990s" = my_colors[3], "2000s" = my_colors[4]
, "2010s" = my_colors[5], "anger" = "grey", "anticipation" = "grey", "disgust" = "grey", "fear" = "grey", "
joy" = "grey", "sadness" = "grey", "surprise" = "grey", "trust" = "grey")

decade_mood <-  prince_nrc %>%
  filter(decade != "NA" & !sentiment %in% c("positive", "negative")) %>%
  count(sentiment, decade) %>%
  group_by(decade, sentiment) %>%
  summarise(sentiment_sum = sum(n)) %>%
  ungroup()
```

```
## `summarise()` regrouping output by 'decade' (override with `.groups` argument)
```

```
circos.clear()
#Set the gap size
circos.par(gap.after = c(rep(5, length(unique(decade_mood[[1]])) - 1), 15,
                         rep(5, length(unique(decade_mood[[2]])) - 1), 15))
chordDiagram(decade_mood, grid.col = grid.col, transparency = .2)
title("Relationship Between Mood and Decade")
```

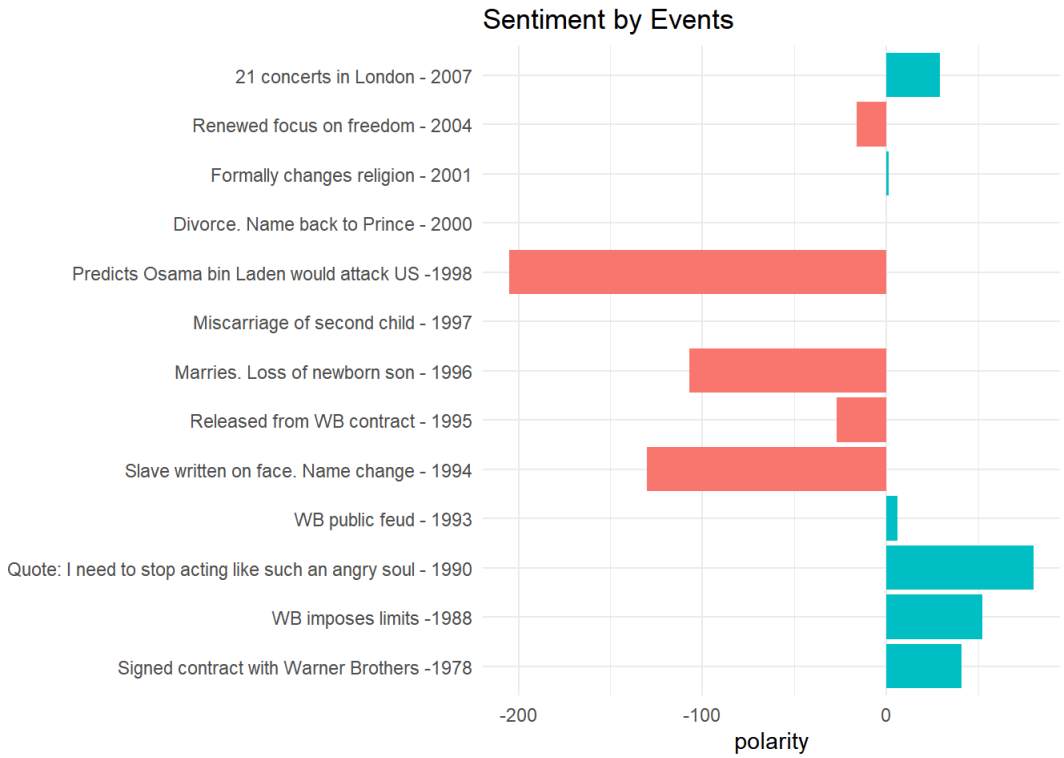


This shows the counts of words per NRC category per decade. It's a lot to take in on a small graph, but it provides tons of information on relationships between categories and time. These diagrams are incredibly customizable and can be as simple or informative as desired. ## Real time analysis

```
#Downloading the file and saving it to dl
dl1 <- tempfile()
download.file("https://s3.amazonaws.com/assets.datacamp.com/blog_assets/sentiment_analysis/princeEvents.csv"
, dl1)
#Read the csv file
events <- read.csv(dl1, stringsAsFactors = FALSE)
year_polarity_bing <- prince_bing %>%
  group_by(year, sentiment) %>%
  count(year, sentiment) %>%
  spread(sentiment, n) %>%
  mutate(polarity = positive - negative,
         ratio = polarity / (positive + negative)) #use polarity ratio in next graph

 events %>%
   #Left join gets event years with no releases
   left_join(year_polarity_bing) %>%
   filter(event != " ") %>% #Account for bad data
   mutate(event = reorder(event, year), #Sort chart by desc year
          sentiment = ifelse(positive > negative,
                             "positive", "negative")) %>%
   ggplot(aes(event, polarity, fill = sentiment)) +
   geom_bar(stat = "identity") +
   theme_minimal() + theme(legend.position = "none") +
   xlab(NULL) +
   ggtitle("Sentiment by Events") +
   coord_flip()
```

```
## Joining, by = "year"
```

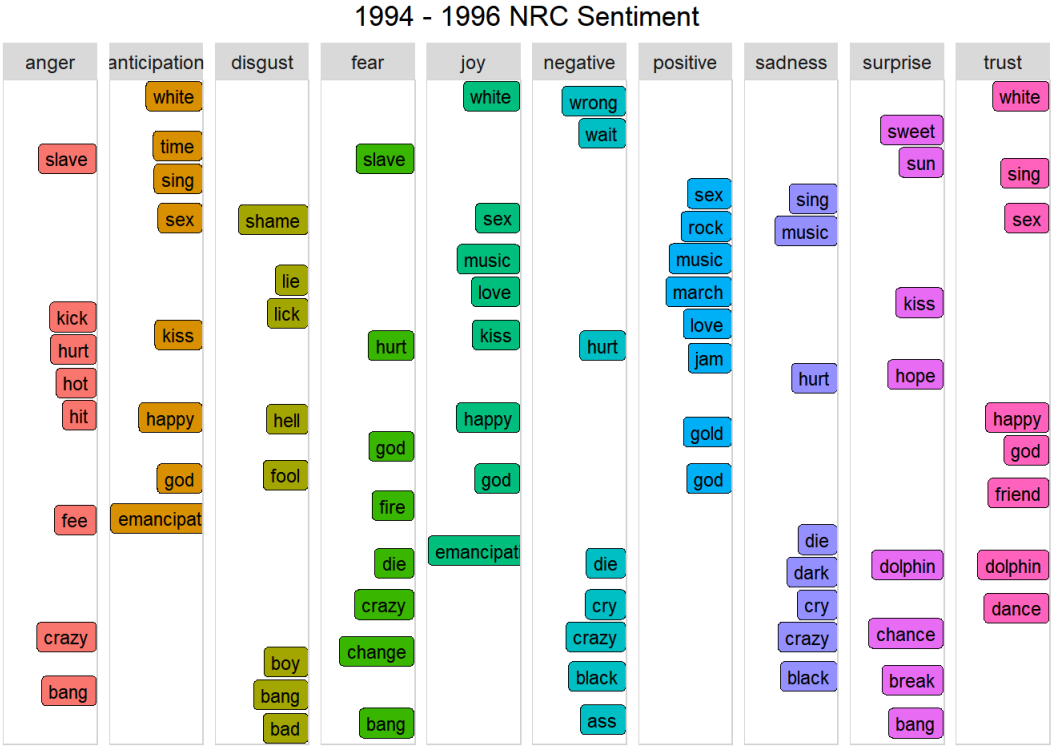
```
## Warning: Removed 2 rows containing missing values (position_stack).
```



The Black Album: 1994 -

1996 In 1994, Prince released The Black Album as an attempt to regain his African-American audience.

```
plot_words_94_96 <- prince_nrc %>%
  filter(year %in% c("1994", "1995", "1996")) %>%
  group_by(sentiment) %>%
  count(word, sort = TRUE) %>%
  arrange(desc(n)) %>%
  slice(seq_len(8)) %>% #consider top_n() from dplyr also
  ungroup()

plot_words_94_96 %>%
  #Set `y = 1` to just plot one variable and use word as the label
  ggplot(aes(word, 1, label = word, fill = sentiment )) +
  #You want the words, not the points
  geom_point(color = "transparent") +
  #Make sure the labels don't overlap
  geom_label_repel(force = 1,nudge_y = .5,
                   direction = "y",
                   box.padding = 0.04,
                   segment.color = "transparent",
                   size = 3) +
  facet_grid(~sentiment) +
  theme_lyrics() +
  theme(axis.text.y = element_blank(), axis.text.x = element_blank(),
        axis.title.x = element_text(size = 6),
        panel.grid = element_blank(), panel.background = element_blank(),
        panel.border = element_rect("lightgray", fill = NA),
        strip.text.x = element_text(size = 9)) +
  xlab(NULL) + ylab(NULL) +
  ggtitle("1994 - 1996 NRC Sentiment") +
  coord_flip()
```



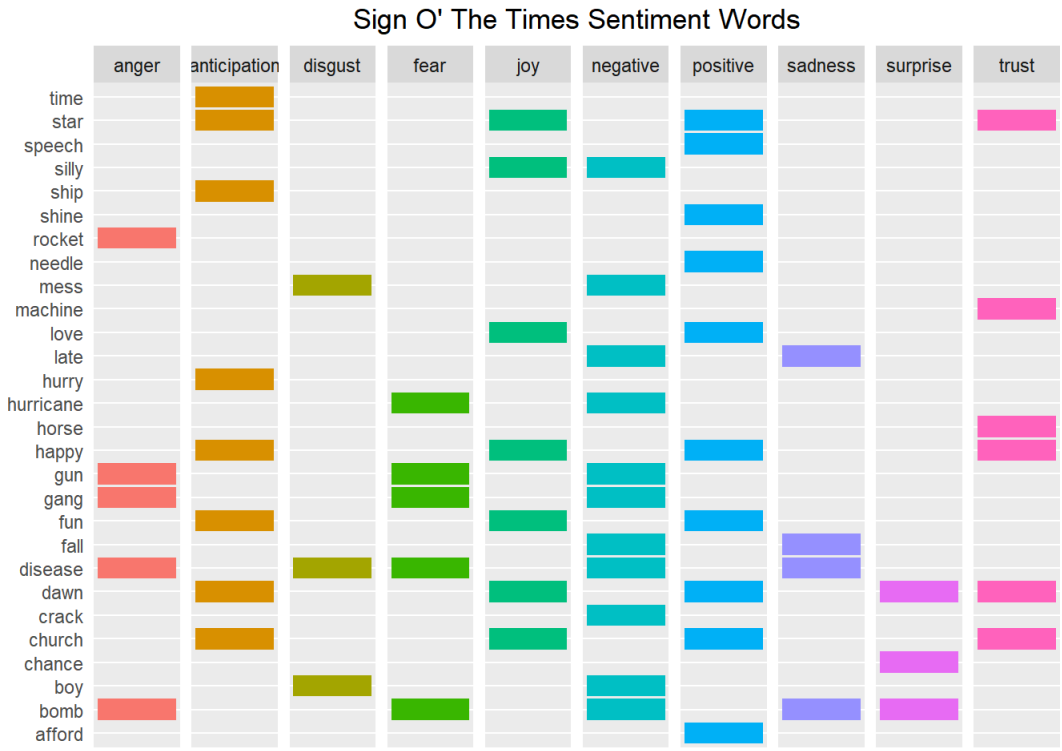
So he appeared in public with the word “slave” prominently penned on his face outwardly declaring his anti-corporate sentiment with his current music label. ## Radar Charts Another great way to compare sentiment across categories is to use a radar chart, which is also known as a spider chart. You can make this type of charts with the radarchart package. These are useful for seeing which variables have similar values or if there are any outliers for each variable. Calculate the total count of words by sentiment per year, as well as the total sentiment for the entire year and obtain a percentage (countofsentimentwordsperyear/totalperyear*100).Filter for the specific years 1978, 1994, 1995, and remove the unneeded fields with select().spread() the year and percent values (key/value pairs) into multiple columns so that you have one row for each sentiment and a column for each year. Then using chartJSRadar() to generate an interactive HTML widget.

```
#Get the count of words per sentiment per year
year_sentiment_nrc <- prince_nrc_sub %>%
  group_by(year, sentiment) %>%
  count(year, sentiment) %>%
  select(year, sentiment, sentiment_year_count = n)

#Get the total count of sentiment words per year (not distinct)
total_sentiment_year <- prince_nrc_sub %>%
  count(year) %>%
  select(year, year_total = n)

#Join the two and create a percent field
year_radar_chart <- year_sentiment_nrc %>%
  inner_join(total_sentiment_year, by = "year") %>%
  mutate(percent = sentiment_year_count / year_total * 100 ) %>%
  filter(year %in% c("1978","1994","1995")) %>%
  select(-sentiment_year_count, -year_total) %>%
  spread(year, percent) %>%
  chartJSRadar(showToolTipLabel = TRUE,
               main = "NRC Years Radar")
```

It’s interesting to note that with a smaller dataset like year, we were able to see the variance in each sentiment more distinctly using this type of visualization.“joy” has the highest score on all charts. Using ggplot2 to create a slightly different chart, look at the words for each category.

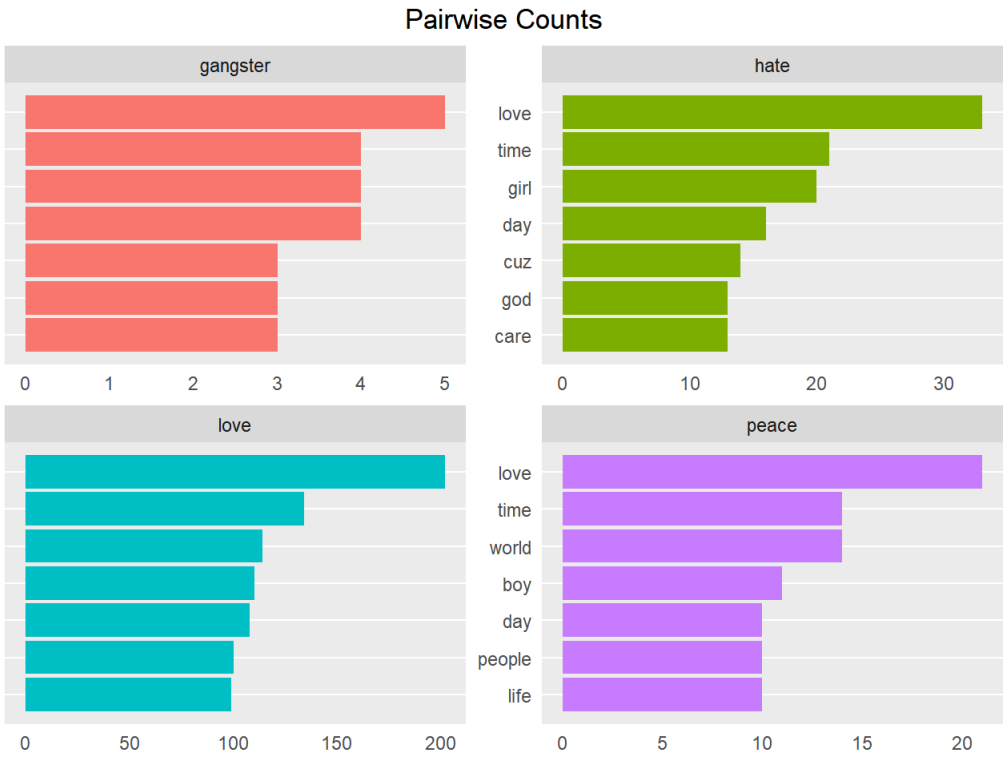
```
prince_tidy %>%
  filter(song %in% 'sign o the times') %>%
  distinct(word) %>%
  inner_join(get_sentiments("nrc")) %>%
  ggplot(aes(x = word, fill = sentiment)) +
  facet_grid(~sentiment) +
  geom_bar() + #Create a bar for each word per sentiment
  theme_lyrics() +
  theme(panel.grid.major.x = element_blank(),
        axis.text.x = element_blank()) + #Place the words on the y-axis
  xlab(NULL) + ylab(NULL) +
  ggtitle("Sign O' The Times Sentiment Words") +
  coord_flip()
```

```
## Joining, by = "word"
```

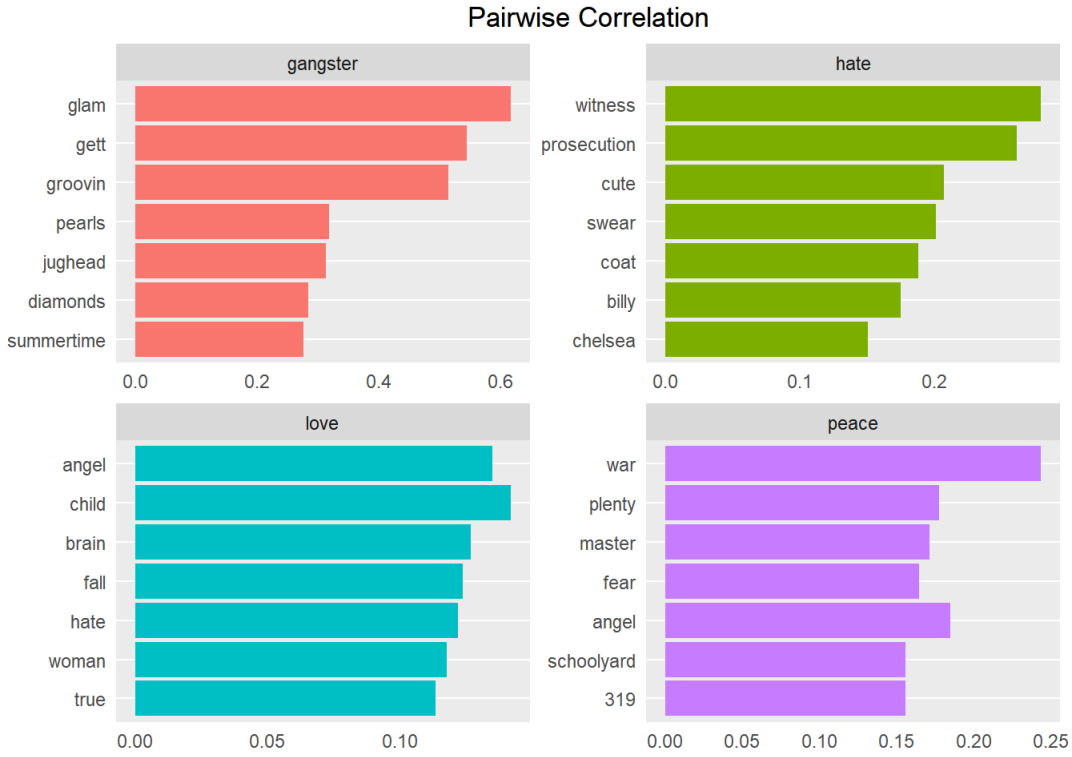


It's pretty easy to see the connection: Challenger disaster -> "rocket", AIDS -> "disease", Drugs -> "crack", Natural Disasters -> "hurricane", Gangs -> "gang".

The word pairs associated with negation words.Some words cross over to multiple nodes which can be seen easily in a visual like this one: for example, "never hurt" and "not hurt". ## Pairwise Comparisons Use the pairwise_count() function from the widyr package to identify co-occurrence counts. The widyr package takes a tidy dataset, and temporarily widens it before returning it to a tidy structure for visualization



and further analysis. Compare that to pairwise correlation. This refers to how often words appear together relative to how often they appear separately. Use pairwise_cor() to determine the correlation between words based on how often they appear in the same song.



These are fascinating insights into the lyrics, just based on these four words alone.

CONCLUSION

In this project we created a tidy dataset and analyzed basic information such as the lexical diversity and song counts per release year, and examined the relationship between release decade and whether a song hit the charts and then explored some sentiment lexicons and how well they matched the lyrics. Performed sentiment analysis on all songs in the dataset, sentiment over time, song level sentiment, and the impact of bigrams. Comparing real life events, both personal and societal, can illuminate the mood of any lyric. Prince's polar sentiment seemed to slightly decline over time, yet overall, joy does seem to stand out. Charted songs seem to be more positive than uncharted songs.But lyrics are complex and too many assumptions can cause problems.