# Sentiment del

Simon

30/11/2021

```
library(tidyverse)
library(lubridate)
library(magrittr)
library(FactoMineR)
library(factoextra)
library(uwot)
library(GGally)
library(rsample)
library(ggridges)
library(xgboost)
library(recipes)
library(parsnip)
library(glmnet)
library(tidymodels)
library(skimr)
library(VIM)
library(visdat)
library(ggmap)
library(ranger)
library(vip)
library(SnowballC)
library(tokenizers)
library(formatR)
```

## Data

```
library(readr)

data_start <- read_csv("C:/Users/Simon ik mig/Downloads/lyrics-data.csv.zip") #Simon
```

```
## Rows: 209522 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): ALink, SName, SLink, Lyric, Idiom

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
artists_data <- read_csv("C:/Users/Simon ik mig/Downloads/artists-data (1).csv")# Simon
```

```
## Rows: 3242 Columns: 6

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Artist, Link, Genre, Genres
## dbl (2): Songs, Popularity

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#data_start <- read_csv("C:/Users/Mikkel/Desktop/UNI/SDS/M3/lyrics-data.csv") Mikkel
#artists_data <- read_csv("C:/Users/Mikkel/Desktop/UNI/SDS/M3/artists-data.csv") Mikkel
```

### Artist data

```r
artists = artists_data %>%
  group_by(Artist) %>%
  count(Genre) %>%
  pivot_wider(names_from = Genre, values_from = n) %>%
  replace_na(list(Pop = 0, "Hip Hop" = 0, Rock = 0, "Funk Carioca" = 0,
                  "Sertanejo" = 0, Samba = 0 )) %>%
  ungroup() %>%
  left_join(artists_data, by = c("Artist")) %>%
  select(-c(Genre, Genres, Popularity, Songs)) %>%
  distinct()
```

### Data Rock or Pop

```r
data_genre = data_start %>%
  filter(Idiom == "ENGLISH") %>%
  rename("Link" = "ALink") %>%
  inner_join(artists, by = c("Link")) %>%
  distinct() %>%
  mutate(name = paste(Artist, SName))%>%
  rename(text=Lyric) %>%
  filter(Pop==1 | Rock==1) %>%
  select(name, text, Pop, Rock) %>%
  distinct(name, .keep_all = T)



data_pop_rock=data_genre %>%
  mutate(genre = ifelse(Pop==1 & Rock == 1, "pop/rock",
                        ifelse(Rock==1 & Pop==0, "Rock",
                               ifelse(Rock == 0 & Pop == 1, "Pop", 0)))) %>%
  select(-c(Pop, Rock))

data_pop_rock_labels= data_pop_rock %>%
  select(name, genre)
```

### Data Rock and Pop

```
data = data_start %>%
  filter(Idiom == "ENGLISH") %>%
  rename("Link" = "ALink") %>%
  inner_join(artists, by = c("Link")) %>%
  distinct() %>%
  mutate(name = paste(Artist, SName))%>%
  rename(text=Lyric) %>%
  filter(Rock==1 & Pop==1) %>%
  select(name, text)%>%
  distinct(name, .keep_all = T)
```

# Preprocessing / EDA

First we tokenize the data.

```
library(tidytext)
text_genre_tidy = data_pop_rock %>% unnest_tokens(word, text, token = "words")

head(text_genre_tidy)
```

```
## # A tibble: 6 x 3
##   name                         genre    word
##   <chr>                        <chr>    <chr>
## 1 10000 Maniacs More Than This pop/rock i
## 2 10000 Maniacs More Than This pop/rock could
## 3 10000 Maniacs More Than This pop/rock feel
## 4 10000 Maniacs More Than This pop/rock at
## 5 10000 Maniacs More Than This pop/rock the
## 6 10000 Maniacs More Than This pop/rock time
```

We remove short words and stopwords.

```
text_genre_tidy %<>%
  filter(str_length(word) > 2 ) %>%
  group_by(word) %>%
  ungroup() %>%
  anti_join(stop_words, by = 'word')
```

We use the hunspell package, which seems to produce the best stemming for our data. Reducing a word to its "root" word.

```
library(hunspell)
text_genre_tidy %<>%
  mutate(stem = hunspell_stem(word)) %>%
  unnest(stem) %>%
  select(-word) %>%
  rename(word = stem)
```

We weight the data using tf-idf (Term-frequency Inverse document frequency).

```r
# TFIDF weights
text_tf_idf= text_genre_tidy %>%
group_by(name) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, name, n) %>%
  arrange(desc(tf_idf))


text_genre_tf_idf = text_tf_idf %>%
  left_join(data_pop_rock_labels)
```

```
## Joining, by = "name"
```

We show the 25 most common words within the 3 genres.

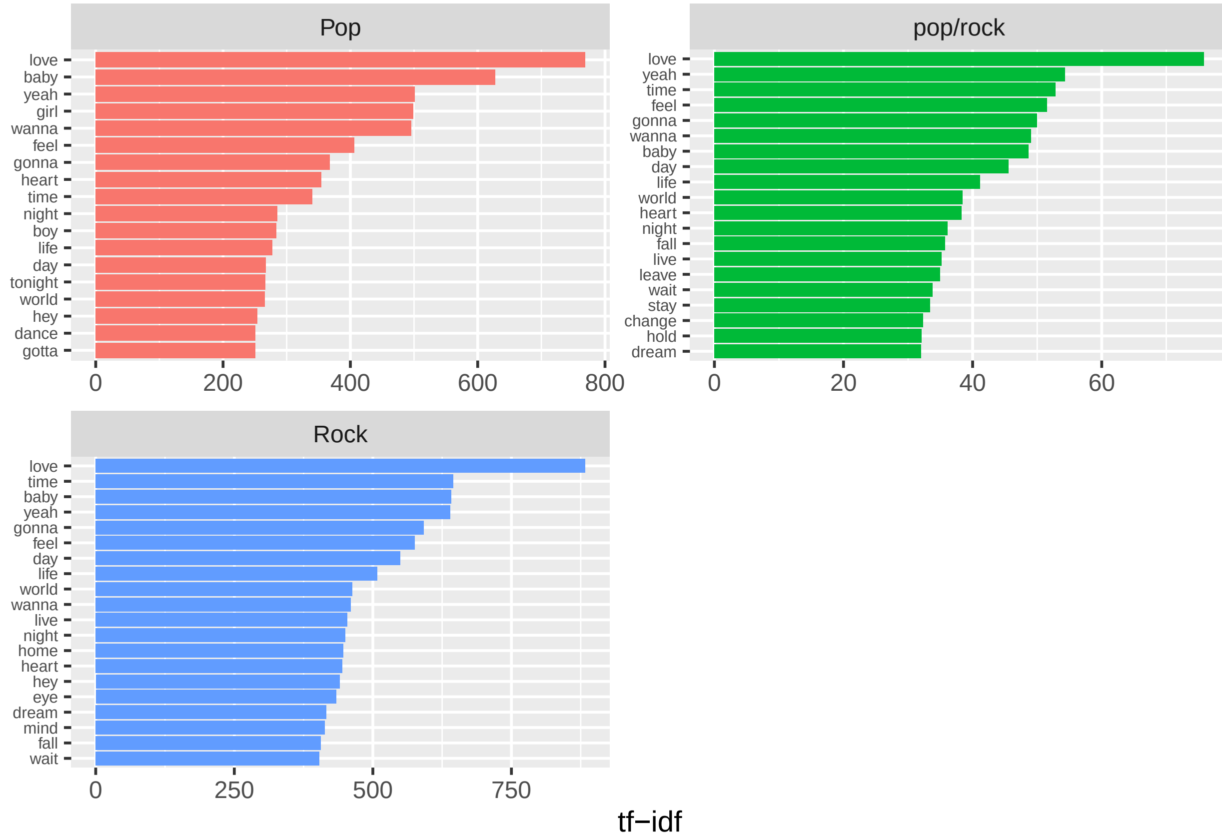
```r
# TFIDF topwords
text_genre_tidy_rock= text_genre_tf_idf %>%
  filter(genre == "Rock")%>%
count(word, wt = tf_idf, sort = TRUE)%>%
  filter(!word == "chorus") %>% #remove
head(25)

text_genre_tidy_rock_pop= text_genre_tf_idf %>%
  filter(genre == "Pop/Rock")%>%
count(word, wt = tf_idf, sort = TRUE) %>% #remove
head(25)

text_genre_tidy_pop= text_genre_tf_idf %>%
  filter(genre == "Pop")%>%
count(word, wt = tf_idf, sort = TRUE)%>%
  filter(!word == "chorus")%>%
  filter(!word == "ooh")%>% #remove
head(25)
```

We now plot the 20 most used words within each genre.

```r
labels_words <- text_genre_tf_idf %>%
group_by(genre) %>%
count(word, wt = tf_idf, sort = TRUE, name = "tf_idf") %>%
dplyr::slice(1:20)%>%
  filter(!word == "chorus")%>%
  filter(!word == "ooh") %>% #slice
ungroup()
```

```r
labels_words %>%
mutate(word = reorder_within(word, by = tf_idf, within = genre)) %>% #Pop & Rock
ggplot(aes(x = word, y = tf_idf, fill = genre)) +
geom_col(show.legend = FALSE) +
labs(x = NULL, y = "tf-idf") +
facet_wrap(~genre, ncol = 2, scales = "free") +
coord_flip() +
```

```
scale_x_reordered() +
theme(axis.text.y = element_text(size = 6))
```




## Rock wordcloud

EDA within the Rock genres.

```
text_tidy_rock = text_genre_tidy %>%
  filter(genre == "Rock")
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
text_tidy_rock %>%
count(word) %>%
with(wordcloud(word, n,
max.words = 50,
color = "blue"))
```


### Pop wordcloud

EDA within the Pop genres.

```
text_tidy_Pop = text_genre_tidy %>%
filter(genre == "Pop")
```

```
text_tidy_Pop %>%
count(word) %>%
with(wordcloud(word, n,
max.words = 50,
color = "blue"))
```


### Pop/Rock wordcloud

EDA within the Pop/Rock genres.

```
text_tidy_Pop_Rock = text_genre_tidy %>%
filter(genre == "pop/rock")
```

```
text_tidy_Pop_Rock %>%
count(word) %>%
with(wordcloud(word, n,
max.words = 50,
color = "blue"))
```


## Sentiment Analysis

### Rock_Pop

We do a sentiment analysis based on the Pop genre.

```
library(textdata)

text_tidy_Pop_Rock_index= text_tidy_Pop_Rock %>%
mutate(index= 1:n())
```

We use the lexicons “bing” and “afinn” to get a measure for positivity and negativity for each word. We use inner_join to only get the words we use from the lexicon.

```
#Bing
sentiment_bing <- text_tidy_Pop_Rock_index %>%
inner_join(get_sentiments("bing")) %>%
count(word, index = index %/% 100, sentiment) %>%
mutate(lexicon = 'Bing')
```

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

```r
# Afinn
sentiment_afinn <- text_tidy_Pop_Rock_index %>%
inner_join(get_sentiments("afinn")) %>%
group_by(index = index %/% 100) %>%
summarise(sentiment = sum(value, na.rm = TRUE)) %>%
mutate(lexicon = 'AFINN')
```

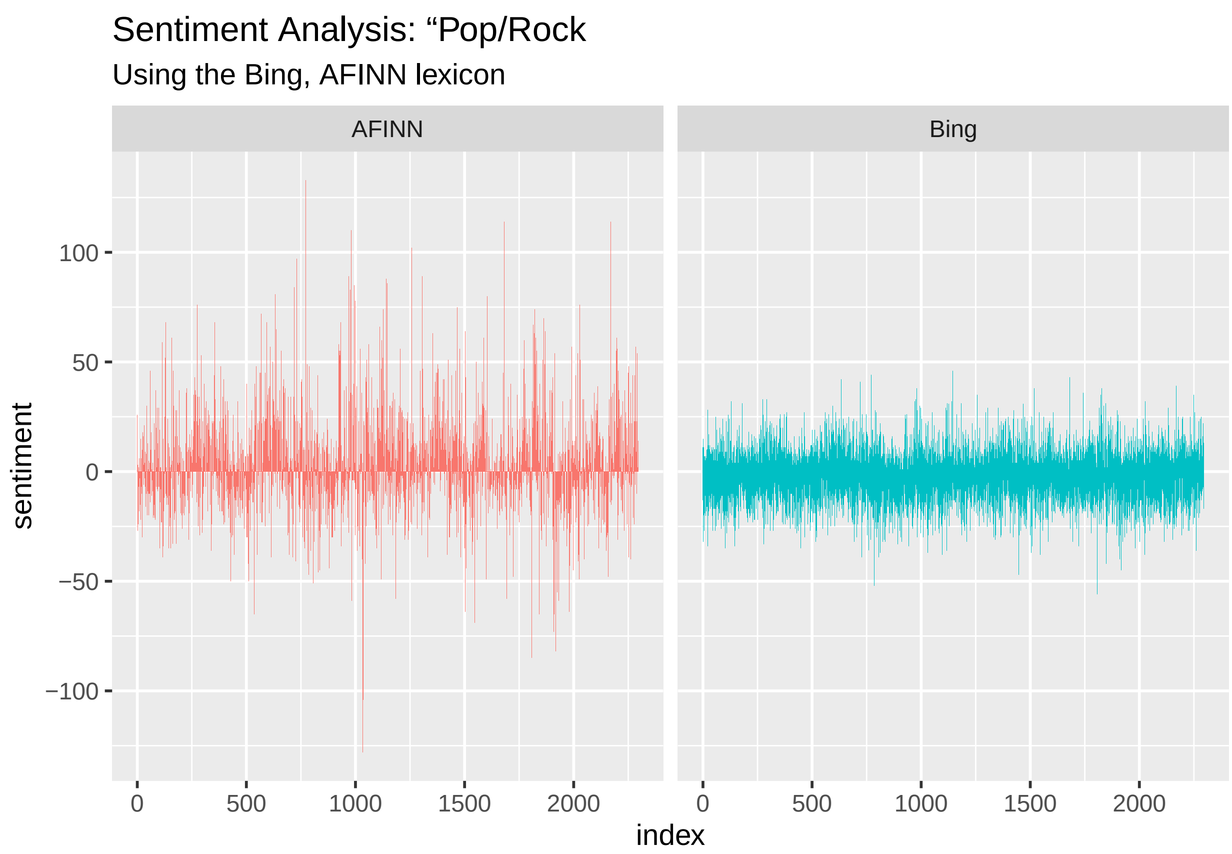
```
## Joining, by = "word"
```

We join the measures from both lexicons.

```r
# Lets join them all together for plotting
sentiment_all <- sentiment_afinn %>%
bind_rows(sentiment_bing %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
mutate(sentiment = positive - negative) %>%
select(index, sentiment, lexicon))
```

We create a plot for the distribution between negative and positive words within the Pop/Rock genre.

```r
sentiment_all %>%
ggplot(aes(x = index, y = sentiment, fill = lexicon)) +
geom_col(show.legend = FALSE) +
facet_wrap(~ lexicon) +
labs(title = "Sentiment Analysis: "Pop/Rock",
subtitle = 'Using the Bing, AFINN lexicon')
```

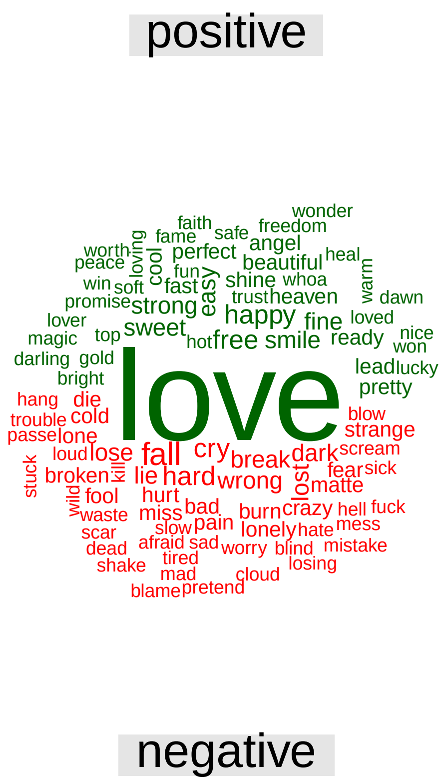


### Senteminet wordcloud

We can now create a wordcloud looking at the positive and negative words in the Pop/Rock genre.

```
text_tidy_Pop_Rock %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
filter(sentiment %in% c("positive", "negative")) %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
as.data.frame() %>%
remove_rownames() %>%
column_to_rownames("word") %>%
comparison.cloud(colors = c("darkgreen", "red"),
max.words = 100,
title.size = 1.5)
```

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




## Pop

We do a sentiment analysis based on the Pop genre.

```r
library(textdata)

text_tidy_Pop_index= text_tidy_Pop %>%
mutate(index= 1:n())
```

We use the lexicons "bing" and "afinn" to get a measure for positivity and negativity for each word. We use inner_join to only get the words we use from the lexicon.

```r
#Bing
sentiment_bing_pop <- text_tidy_Pop_index %>%
inner_join(get_sentiments("bing")) %>%
count(word, index = index %/% 100, sentiment) %>%
mutate(lexicon = 'Bing')
```

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

```r
# Afinn
sentiment_afinn_pop <- text_tidy_Pop_index %>%
inner_join(get_sentiments("afinn")) %>%
group_by(index = index %/% 100) %>%
summarise(sentiment = sum(value, na.rm = TRUE)) %>%
mutate(lexicon = 'AFINN')
```

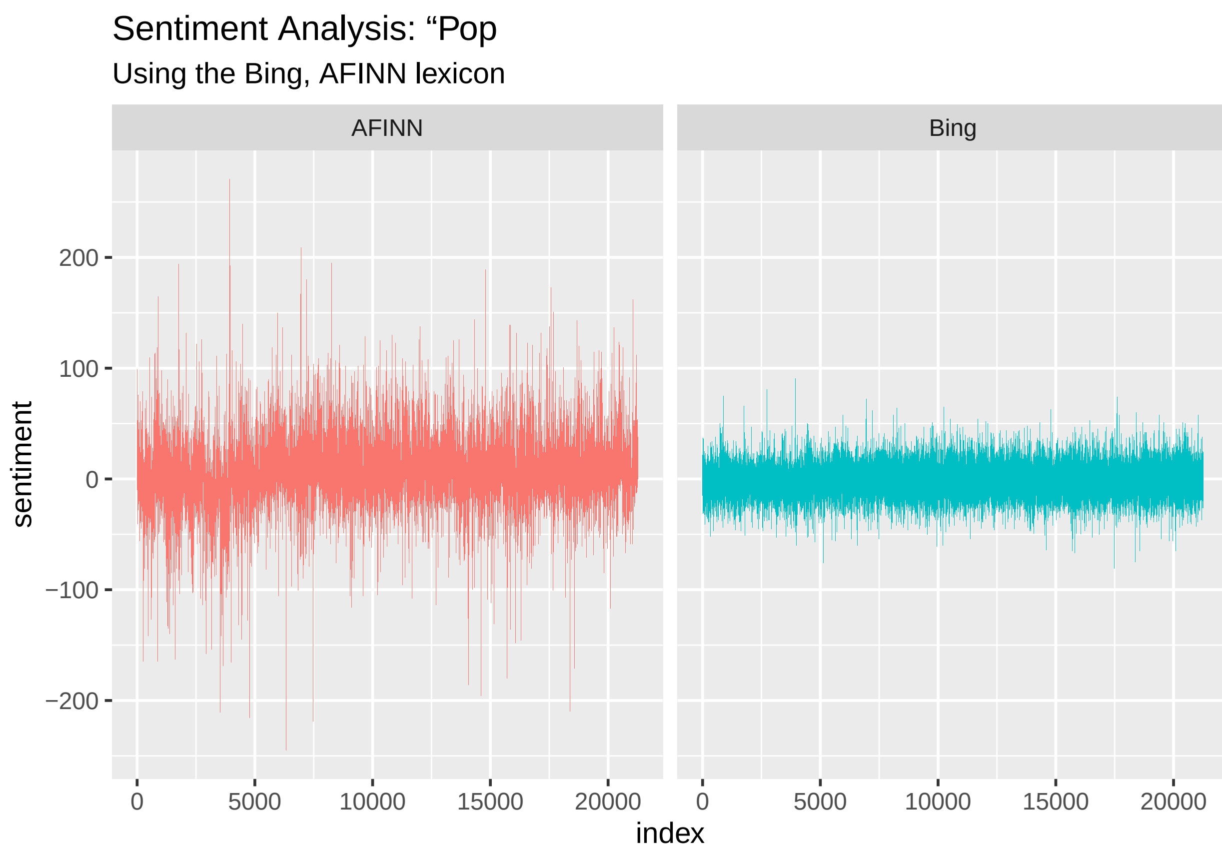
```
## Joining, by = "word"
```

We join the measures from both lexicons.

```r
# Lets join them all together for plotting
sentiment_all_pop <- sentiment_afinn_pop %>%
bind_rows(sentiment_bing_pop %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
mutate(sentiment = positive - negative) %>%
select(index, sentiment, lexicon))
```

We create a plot for the distribution between negative and positive words within the Pop genre.

```r
sentiment_all_pop %>%
ggplot(aes(x = index, y = sentiment, fill = lexicon)) +
geom_col(show.legend = FALSE) +
facet_wrap(~ lexicon) +
labs(title = "Sentiment Analysis: "Pop",
subtitle = 'Using the Bing, AFINN lexicon')
```

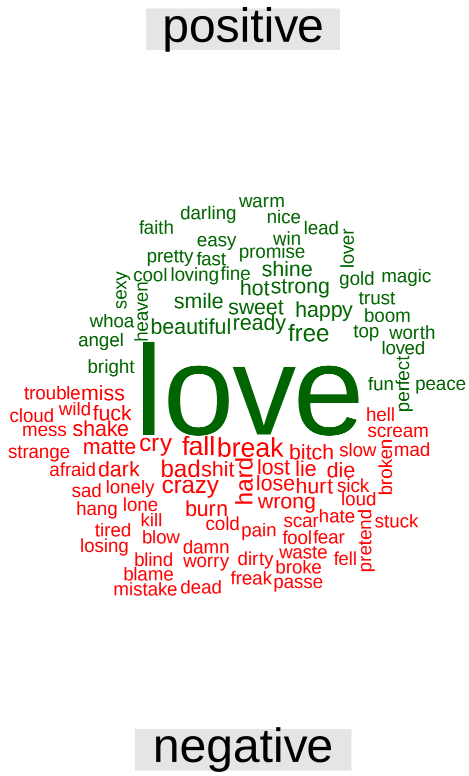



**Senteminet wordcloud**

We can now create a wordcloud looking at the positive and negative words in the Pop genre.

```
text_tidy_Pop %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
filter(sentiment %in% c("positive", "negative")) %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
as.data.frame() %>%
remove_rownames() %>%
column_to_rownames("word") %>%
comparison.cloud(colors = c("darkgreen", "red"),
max.words = 100,
title.size = 1.5)
```

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




### Rock

We do a sentiment analysis based on the Rock genre.

```
library(textdata)

text_tidy_Rock_index= text_tidy_rock %>%
mutate(index= 1:n())
```

We use the lexicons "bing" and "afinn" to get a measure for positivity and negativity for each word. We use inner_join to only get the words we use from the lexicon.

```
#Bing
sentiment_bing_rock <- text_tidy_Rock_index %>%
inner_join(get_sentiments("bing")) %>%
count(word, index = index %/% 100, sentiment) %>%
mutate(lexicon = 'Bing')
```

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

```
# Afinn
sentiment_afinn_rock <- text_tidy_Rock_index %>%
inner_join(get_sentiments("afinn")) %>%
group_by(index = index %/% 100) %>%
```

```
summarise(sentiment = sum(value, na.rm = TRUE)) %>%
mutate(lexicon = 'AFINN')
```

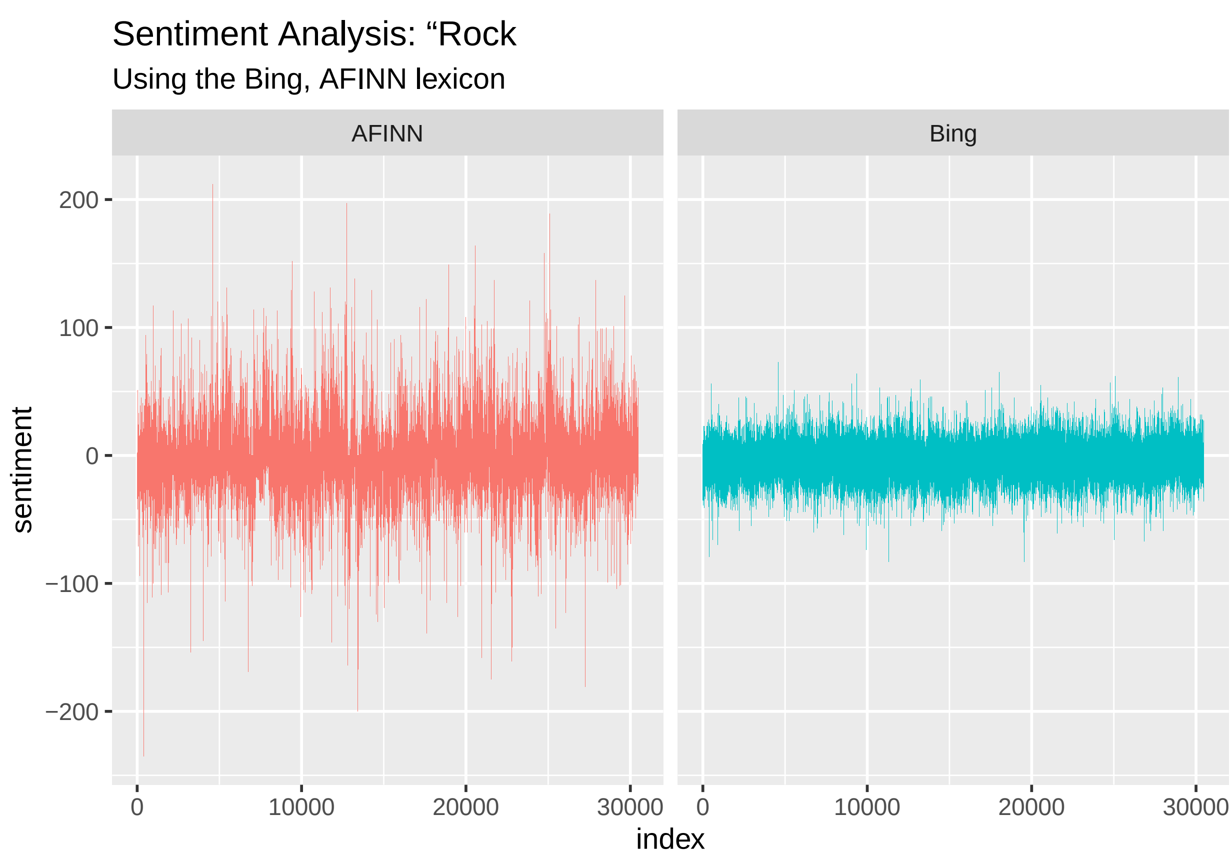
```
## Joining, by = "word"
```

We join the measures from both lexicons.

```
# Lets join them all together for plotting
sentiment_all_rock <- sentiment_afinn_rock %>%
bind_rows(sentiment_bing_rock %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
mutate(sentiment = positive - negative) %>%
select(index, sentiment, lexicon))
```

We create a plot for the distribution between negative and positive words within the Rock genre.

```
sentiment_all_rock %>%
ggplot(aes(x = index, y = sentiment, fill = lexicon)) +
geom_col(show.legend = FALSE) +
facet_wrap(~ lexicon) +
labs(title = "Sentiment Analysis: "Rock",
subtitle = 'Using the Bing, AFINN lexicon')
```



### Senteminet wordcloud

We can now create a wordcloud looking at the positive and negative words in the Rock genre.

```
text_tidy_rock %>%
inner_join(get_sentiments("bing")) %>%
count(word, sentiment, sort = TRUE) %>%
filter(sentiment %in% c("positive", "negative")) %>%
pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
as.data.frame() %>%
remove_rownames() %>%
column_to_rownames("word") %>%
comparison.cloud(colors = c("darkgreen", "red"),
max.words = 100,
title.size = 1.5)
```

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


## Bands analysis

We dont need the genre any more so we remove it.

```
data_band = data_start %>%
  filter(Idiom == "ENGLISH") %>%
  rename("Link" = "ALink") %>%
  inner_join(artists, by = c("Link")) %>%
  distinct() %>%
  rename(text=Lyric) %>%
  filter(Pop==1 | Rock==1) %>%
  select(Artist, text)
```

We want to see the most active artists.

```
data_band %>%
  count(Artist, sort = T)
```

```
## # A tibble: 974 x 2
##    Artist                 n
##    <chr>              <int>
##  1 Elvis Presley        747
##  2 Glee                 687
##  3 Chris Brown          562
##  4 Bee Gees             549
##  5 Bob Dylan            534
##  6 Neil Young           488
##  7 Van Morrison         485
##  8 Bruce Springsteen    477
##  9 Elvis Costello       474
## 10 Rod Stewart          435
## # ... with 964 more rows
```

We pick the top 3 artists (in our opinion) Green day, Bon Jovi and Red Hot Chili Peppers!

```
best_artists= data_band %>%
  filter(Artist %in% c("Green Day",  "Bon Jovi" , "Red Hot Chili Peppers" ))
```

First we tokenize the data.

```
text_band_tidy = best_artists %>% unnest_tokens(word, text, token = "words")

head(text_band_tidy)
```

```
## # A tibble: 6 x 2
##   Artist   word
##   <chr>    <chr>
## 1 Bon Jovi this
## 2 Bon Jovi romeo
## 3 Bon Jovi is
## 4 Bon Jovi bleeding
## 5 Bon Jovi but
## 6 Bon Jovi you
```

We remove short words and stopwords.

```
text_band_tidy %<>%
  filter(str_length(word) > 2 ) %>%
  group_by(word) %>%
  ungroup() %>%
  anti_join(stop_words, by = 'word')
```

We use the hunspell package, which seems to produce the best stemming for our data. Reducing a word to its “root” word.

```
text_band_tidy %<>%
  mutate(stem = hunspell_stem(word)) %>%
  unnest(stem) %>%
  select(-word) %>%
  rename(word = stem)
```

We weight the data using tf-idf (Term-frequency Inverse document frequency).

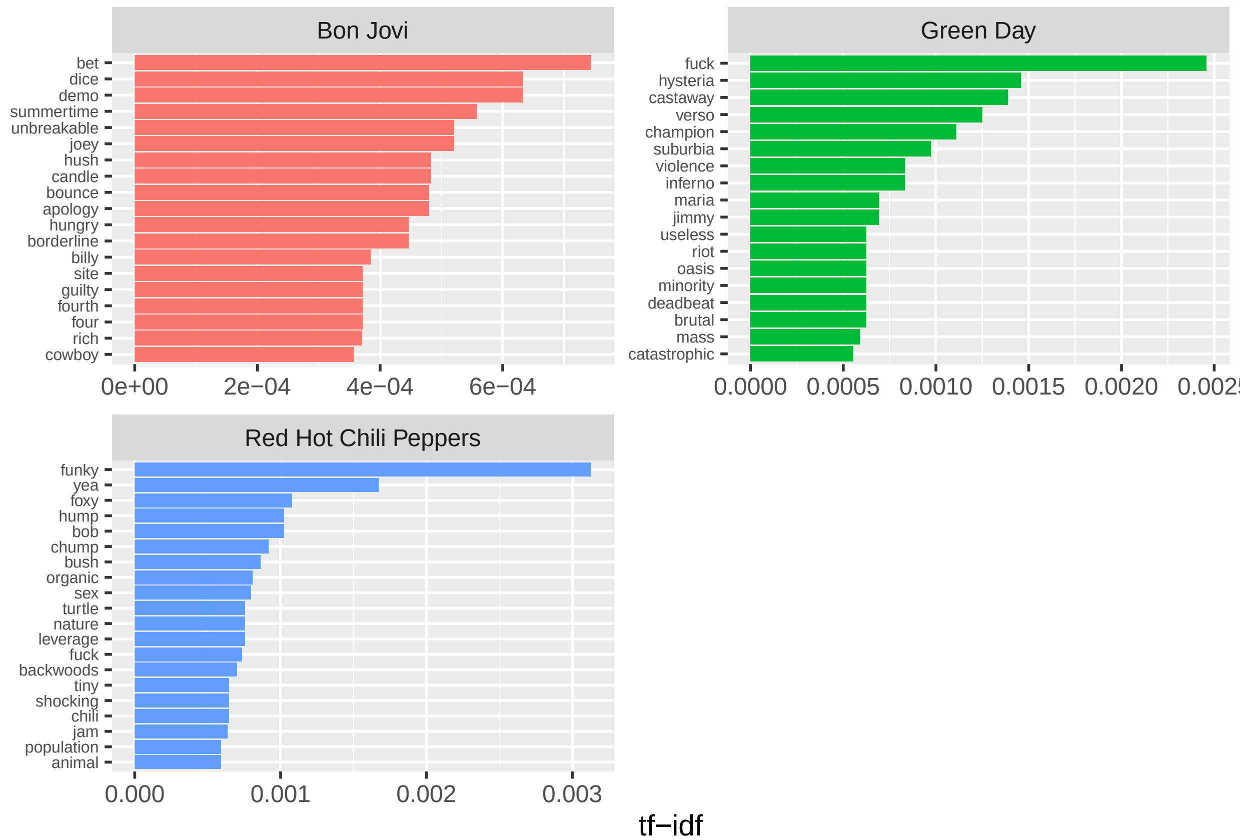
```
# TFIDF weights
text_band_tf_idf= text_band_tidy %>%
group_by(Artist) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, Artist, n) %>%
  arrange(desc(tf_idf))
```

We show the 25 most common words within the 3 artists.

```
# TFIDF topwords
text_band_tidy_Bon_Jovi= text_band_tf_idf %>%
  filter(Artist == "Bon Jovi")%>%
count(word, wt = tf_idf, sort = TRUE)%>%
  filter(!word == "mo") %>% #remove
head(25)

text_band_tidy_Green_Day= text_band_tf_idf %>%
  filter(Artist == "Green Day")%>%
count(word, wt = tf_idf, sort = TRUE)%>%
  filter(!word == "intro")%>%
  filter(!word == "riff") %>% #remove
head(25)

text_band_tidy_RHCP= text_band_tf_idf %>%
  filter(Artist == "Red Hot Chili Peppers")%>%
count(word, wt = tf_idf, sort = TRUE)%>%
  filter(!word == "co")%>%
  filter(!word == "cos") %>% #remove
head(25)
```

We now plot the 20 most used words within each genre.

```
labels_words_band <- text_band_tf_idf %>%
  group_by(Artist) %>%
  count(word, wt = tf_idf, sort = TRUE, name = "tf_idf") %>%
  dplyr::slice(1:20)%>%
  filter(!word == "mo")%>%
  filter(!word == "intro")%>%
  filter(!word == "riff")%>%
  filter(!word == "co")%>%
  filter(!word == "cos") %>%
  ungroup()

labels_words_band %>%
  mutate(word = reorder_within(word, by = tf_idf, within = Artist)) %>%
  ggplot(aes(x = word, y = tf_idf, fill = Artist)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~Artist, ncol = 2, scales = "free") +
  coord_flip() +
  scale_x_reordered() +
  theme(axis.text.y = element_text(size = 6))
```



### Songs of the top 3 artist

```r
# Greenday

sentiment_green_day= text_band_tidy %>%
  filter(Artist == "Green Day") %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative owrds
```

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

```r
sentiment_green_day
```

```
## # A tibble: 1 x 3
##   negative positive sentiment
##      <dbl>    <dbl>     <dbl>
## 1     3309      923     -2386
```

```r
# Bon Jovi

sentiment_Bon_Jovi= text_band_tidy %>%
  filter(Artist == "Bon Jovi") %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative owrds
```

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

```r
sentiment_Bon_Jovi
```

```
## # A tibble: 1 x 3
##   negative positive sentiment
##      <dbl>    <dbl>     <dbl>
## 1     3819     2445     -1374
```

```r
# Red Hot Chilie Pepper

sentiment_RHCP= text_band_tidy %>%
  filter(Artist == "Red Hot Chili Peppers") %>%
  inner_join(get_sentiments("bing")) %>% # pull out only sentiment words
  count(sentiment) %>% # count the # of positive & negative words
  spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
  mutate(sentiment = positive - negative) # # of positive words - # of negative owrds
```

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

```
sentiment_RHCP
```

```
## # A tibble: 1 x 3
##   negative positive sentiment
##      <dbl>    <dbl>     <dbl>
## 1     2686     1996      -690
```

We can see all the artist use more negative laden words than positive

To see sentiment for each song we can join the song names again.

```
data_song_name = data_start %>%
  filter(Idiom == "ENGLISH") %>%
  rename("Link" = "ALink") %>%
  inner_join(artists, by = c("Link")) %>%
  distinct() %>%
  filter(Artist %in% c("Bon Jovi", "Green Day", "Red Hot Chili Peppers")) %>%
  rename(text=Lyric) %>%
  filter(Pop==1 | Rock==1) %>%
  select(SName, text)
```

We tokenize

```
text_song_tidy = data_song_name %>% unnest_tokens(word, text, token = "words")
```

We remove short words and stopwords.

```
text_song_tidy %<>%
  filter(str_length(word) > 2 ) %>%
  group_by(word) %>%
  ungroup() %>%
  anti_join(stop_words, by = 'word')
```

We use the hunspell package, which seems to produce the best stemming for our data. Reducing a word to its “root” word.

```
text_song_tidy %<>%
  mutate(stem = hunspell_stem(word)) %>%
  unnest(stem) %>%
  select(-word) %>%
  rename(word = stem)
```

We will now wheight by tf-idf

```
# TFIDF weights
text_song_tf_idf= text_song_tidy %>%
group_by(SName) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, SName, n) %>%
  arrange(desc(tf_idf))
```

We can now add the band name for our chossen artists

```
data_song_artist = data_start %>%
  filter(Idiom == "ENGLISH") %>%
  rename("Link" = "ALink") %>%
  inner_join(artists, by = c("Link")) %>%
  distinct() %>%
  filter(Artist %in% c("Bon Jovi", "Green Day", "Red Hot Chili Peppers")) %>%
  rename(text=Lyric) %>%
  filter(Pop==1 | Rock==1) %>%
  select(Artist,SName)


text_song_tf_idf %<>%
  inner_join(data_song_artist, by= c("SName"))

# For Green Day

green_day_songs=text_song_tf_idf %>%
  filter(Artist == "Green Day") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative")))%>%
  count(sentiment_song)
```

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

```
# For RHCP

RHCP_songs=text_song_tf_idf %>%
  filter(Artist == "Red Hot Chili Peppers") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative")))%>%
  count(sentiment_song)
```

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

```
# Bon Jovi

Bon_Jovi_songs=text_song_tf_idf %>%
  filter(Artist == "Bon Jovi") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
```

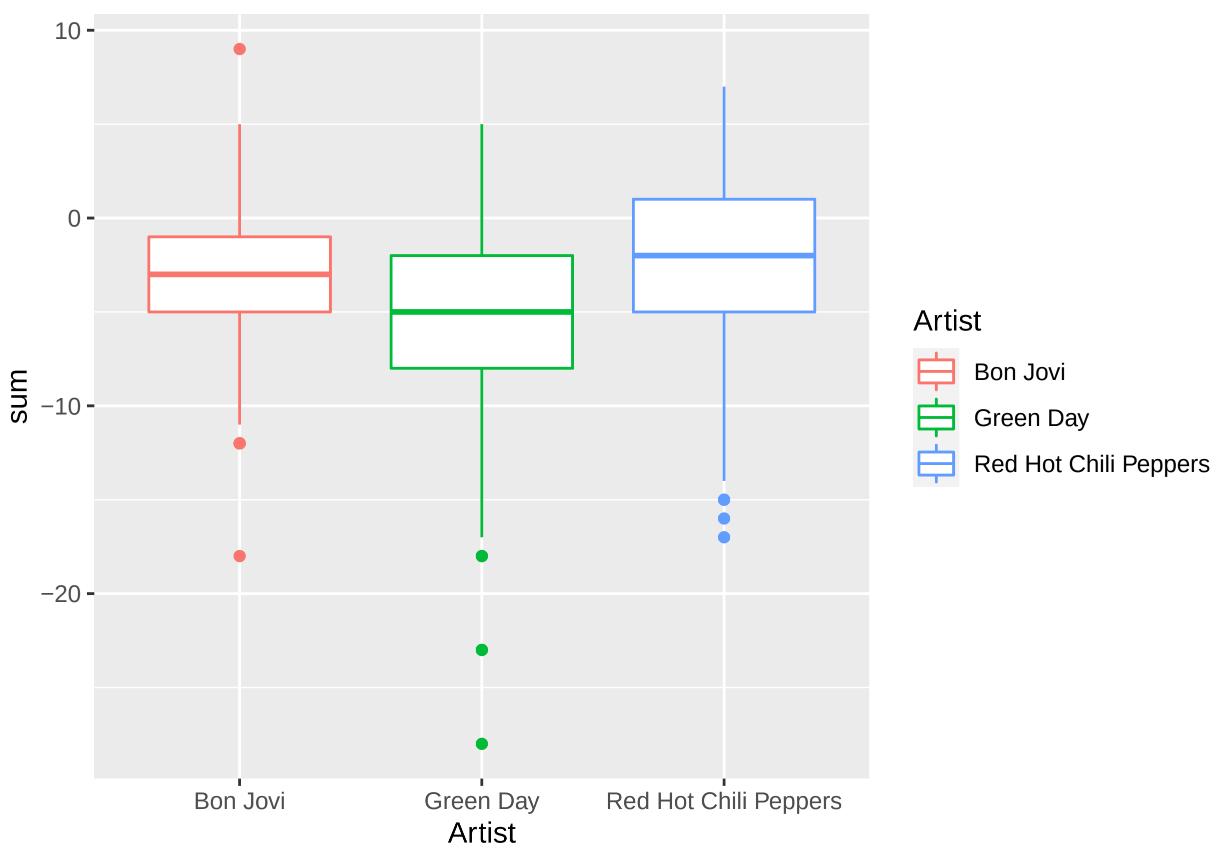
```r
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative")))%>%
  count(sentiment_song)
```

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

```r
# For all

all_songs=text_song_tf_idf %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative")))%>%
  inner_join(data_song_artist, by= c("SName"))
```

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

```r
ggplot(all_songs, aes(x = Artist, y = sum, color = Artist)) +
  geom_boxplot()
```




We can here that the overall score of the songs seems to be negative for all three artists. We can tho see that RHCP on the averrage has ths most positive songs looking at the three artists.

## Sentiment over time

We found a dataset including release date for songs on spotify

```
data_releaseyear <- read_csv("data.csv") # ligger på Github
```

```
## Rows: 169909 Columns: 19
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): artists, id, name, release_date
## dbl (15): acousticness, danceability, duration_ms, energy, explicit, instrum...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
release_year_bon_jovi= data_releaseyear %>%
  filter(artists == "['Bon Jovi']") %>%
  select(name, year)

release_year_RHCP= data_releaseyear %>%
  filter(artists == "['Red Hot Chili Peppers']") %>%
  select(name, year)

release_year_Green_Day= data_releaseyear %>%
  filter(artists == "['Green Day']") %>%
  select(name, year)
```

We will innerjoin with the datasets above

```
#Bon Jovi

Bon_Jovi_songs=text_song_tf_idf %>%
  filter(Artist == "Bon Jovi") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative"))) %>%
  inner_join(release_year_bon_jovi, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

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

```
#RHCP
RHCP_songs=text_song_tf_idf %>%
  filter(Artist == "Red Hot Chili Peppers") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
```

```
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative"))) %>%
  inner_join(release_year_RHCP, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

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

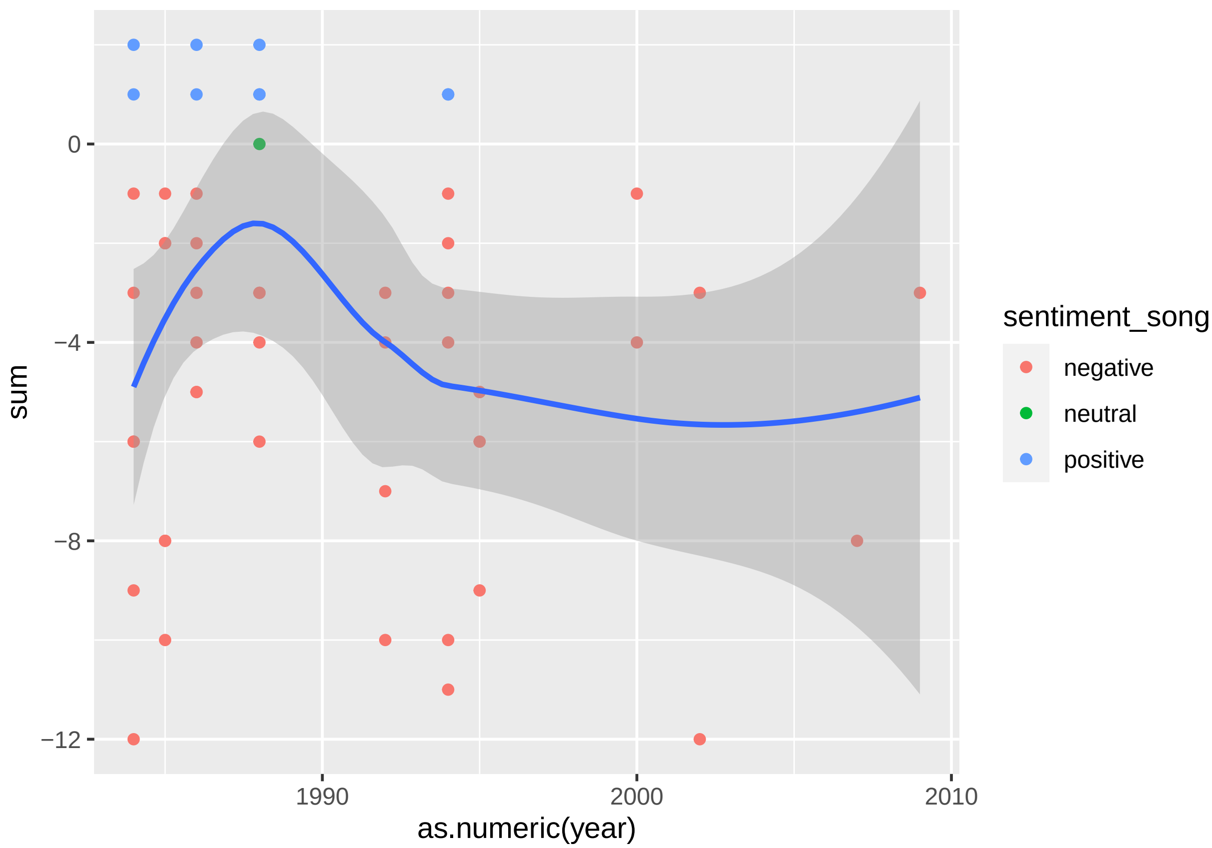
```
## Green Day

Green_day_songs=text_song_tf_idf %>%
  filter(Artist == "Green Day") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("bing"))%>%
  mutate(sentiment= ifelse(sentiment == "negative", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "positive", ifelse(sum == 0, "neutral", "negative"))) %>%
  inner_join(release_year_Green_Day, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

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

Development over time

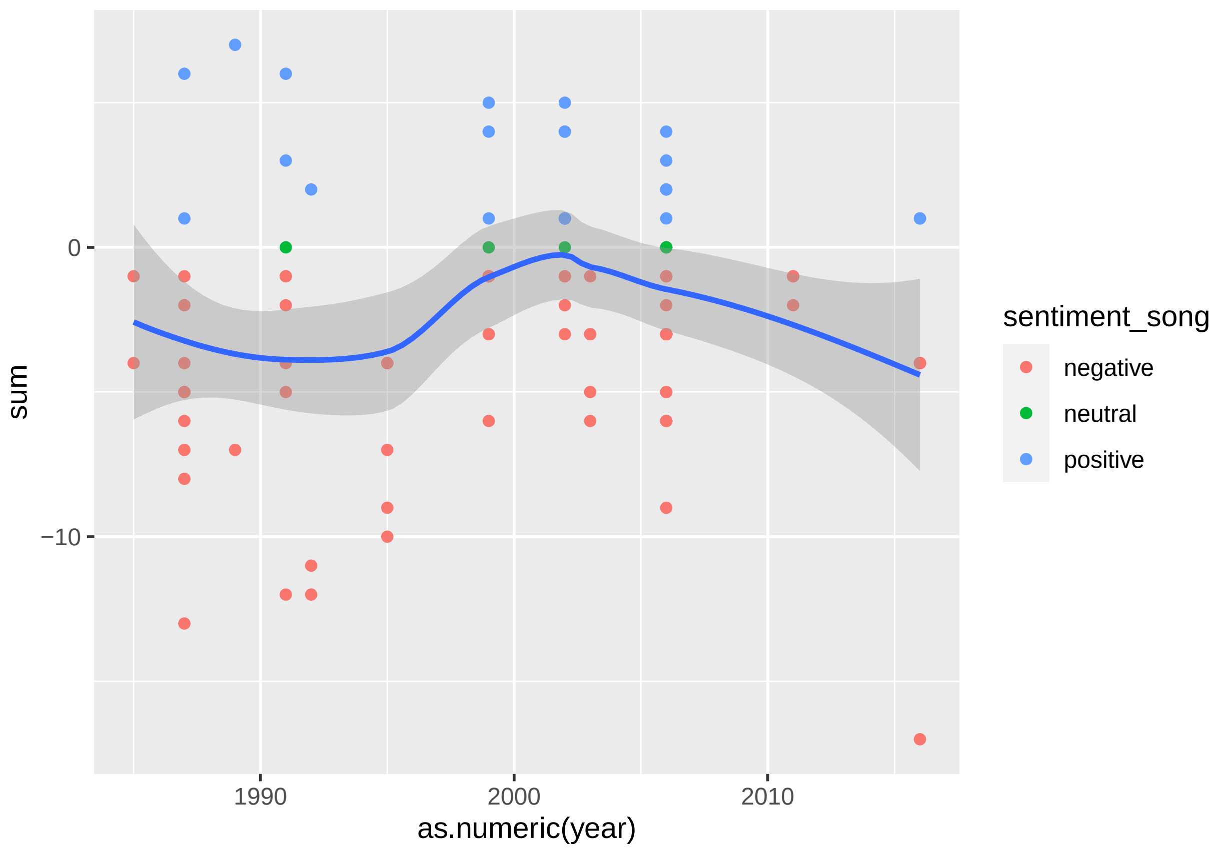
```
## Bon Jovi
ggplot(Bon_Jovi_songs, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



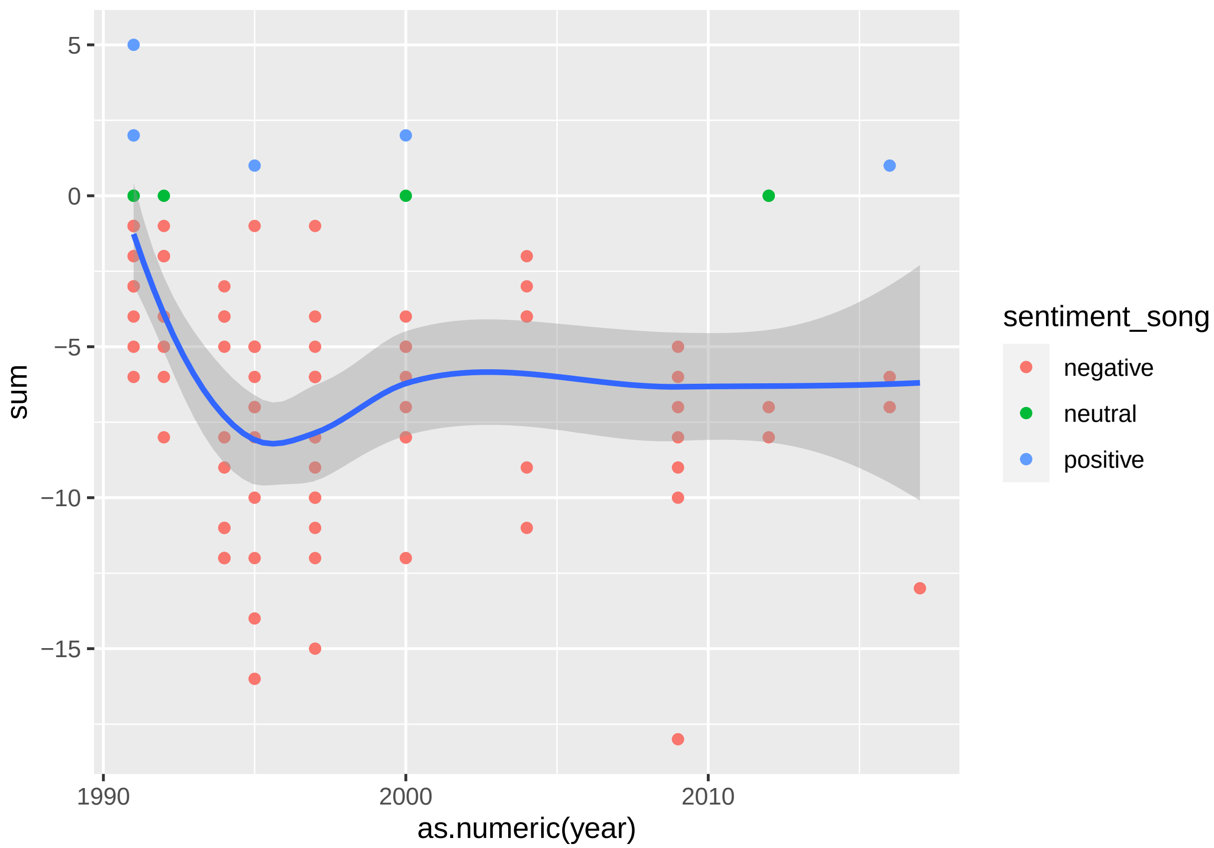

```
## RHCP

ggplot(RHCP_songs, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
## Green Day

ggplot(Green_day_songs, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




We can now see the development of the sentiment of the songs from the three artists. it looks like the artists at some point go for more positive songs, but return to more negative again.

## Saddness og joy

We will now look at the development of Saddness or joy in the songs of the three artists

```
#NRC
sentiment_bing <- text_tidy_Pop_Rock_index %>%
inner_join(get_sentiments("nrc")) %>%
count(word, index = index %/% 100, sentiment) %>%
mutate(lexicon = 'Bing')
```

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

```
#Bon Jovi

Bon_Jovi_songs_joy_sadness=text_song_tf_idf %>%
  filter(Artist == "Bon Jovi") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("nrc"))%>%
  filter(sentiment %in% c("sadness", "joy")) %>%
  mutate(sentiment= ifelse(sentiment == "sadness", -1, 1)) %>%
  group_by(SName) %>%
```

```
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "joy", ifelse(sum == 0, "neutral", "sadness"))) %>%
  inner_join(release_year_bon_jovi, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

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

```
## Green Day

Green_Day_songs_joy_sadness=text_song_tf_idf %>%
  filter(Artist == "Green Day") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("nrc"))%>%
  filter(sentiment %in% c("sadness", "joy")) %>%
  mutate(sentiment= ifelse(sentiment == "sadness", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "joy", ifelse(sum == 0, "neutral", "sadness"))) %>%
  inner_join(release_year_Green_Day, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

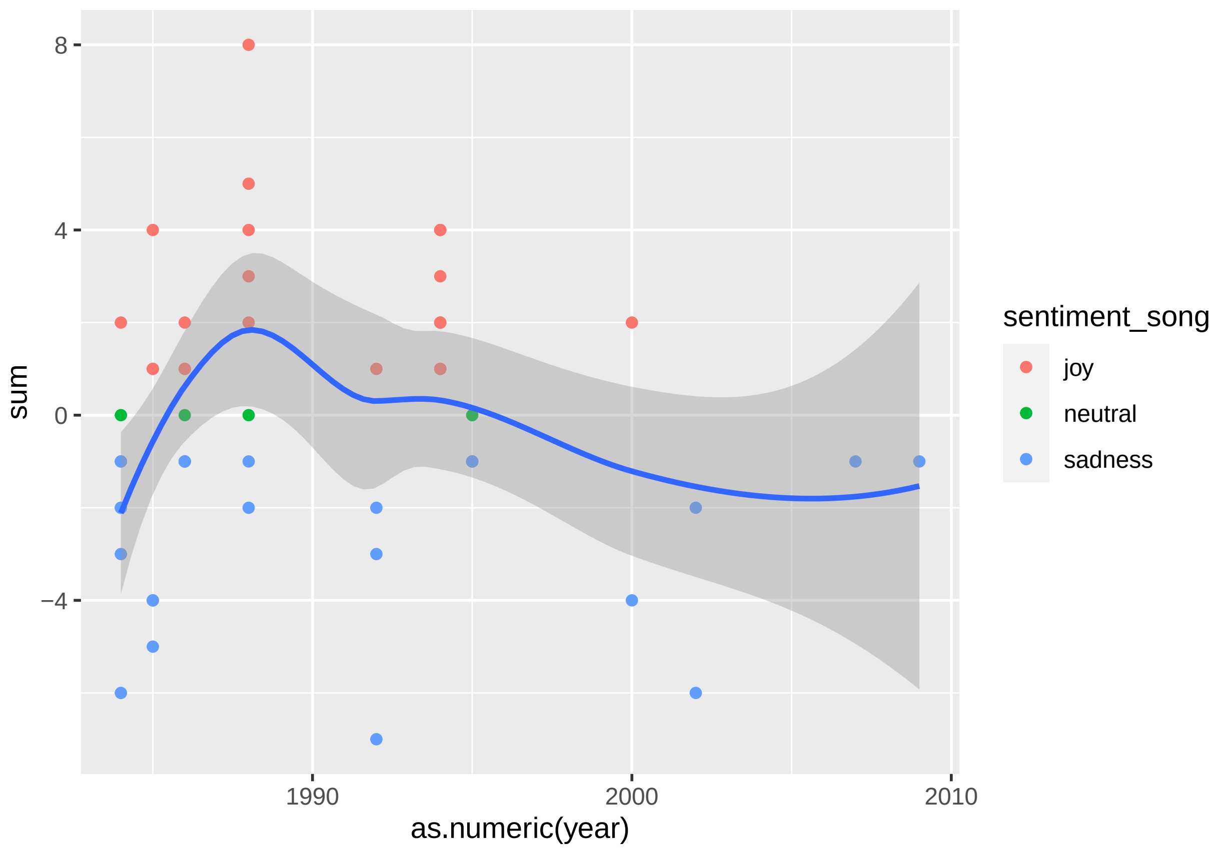
```
## Joining, by = "word"
```

```
## RHCP

RHCP_songs_joy_sadness=text_song_tf_idf %>%
  filter(Artist == "Red Hot Chili Peppers") %>%
  arrange(desc(tf_idf))%>%
  inner_join(get_sentiments("nrc"))%>%
  filter(sentiment %in% c("sadness", "joy")) %>%
  mutate(sentiment= ifelse(sentiment == "sadness", -1, 1)) %>%
  group_by(SName) %>%
  summarise(sum= sum(sentiment)) %>%
  mutate(sentiment_song= ifelse(sum > 0, "joy", ifelse(sum == 0, "neutral", "sadness"))) %>%
  inner_join(release_year_RHCP, by= c("SName" = "name")) %>%
  distinct(SName, .keep_all = T)
```

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

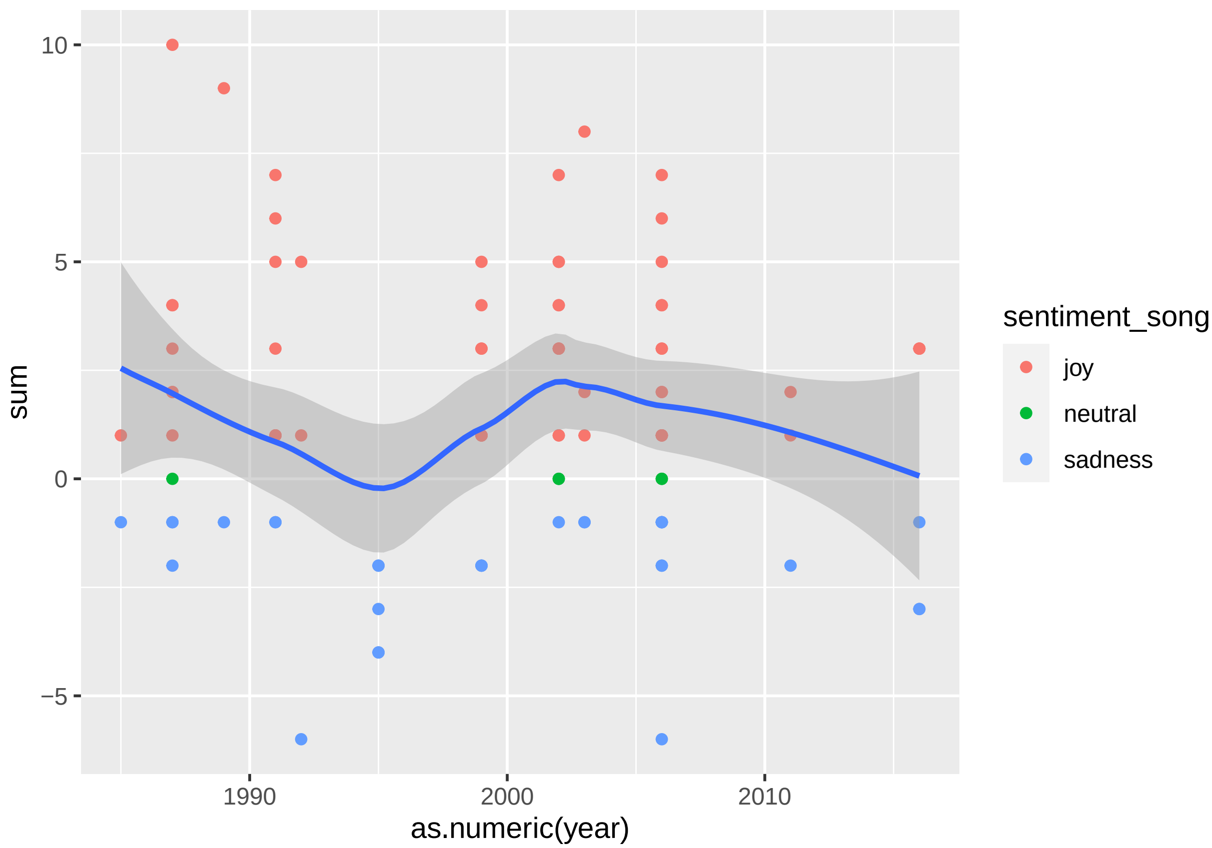
```
## Bon Jovi
ggplot(Bon_Jovi_songs_joy_sadness, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



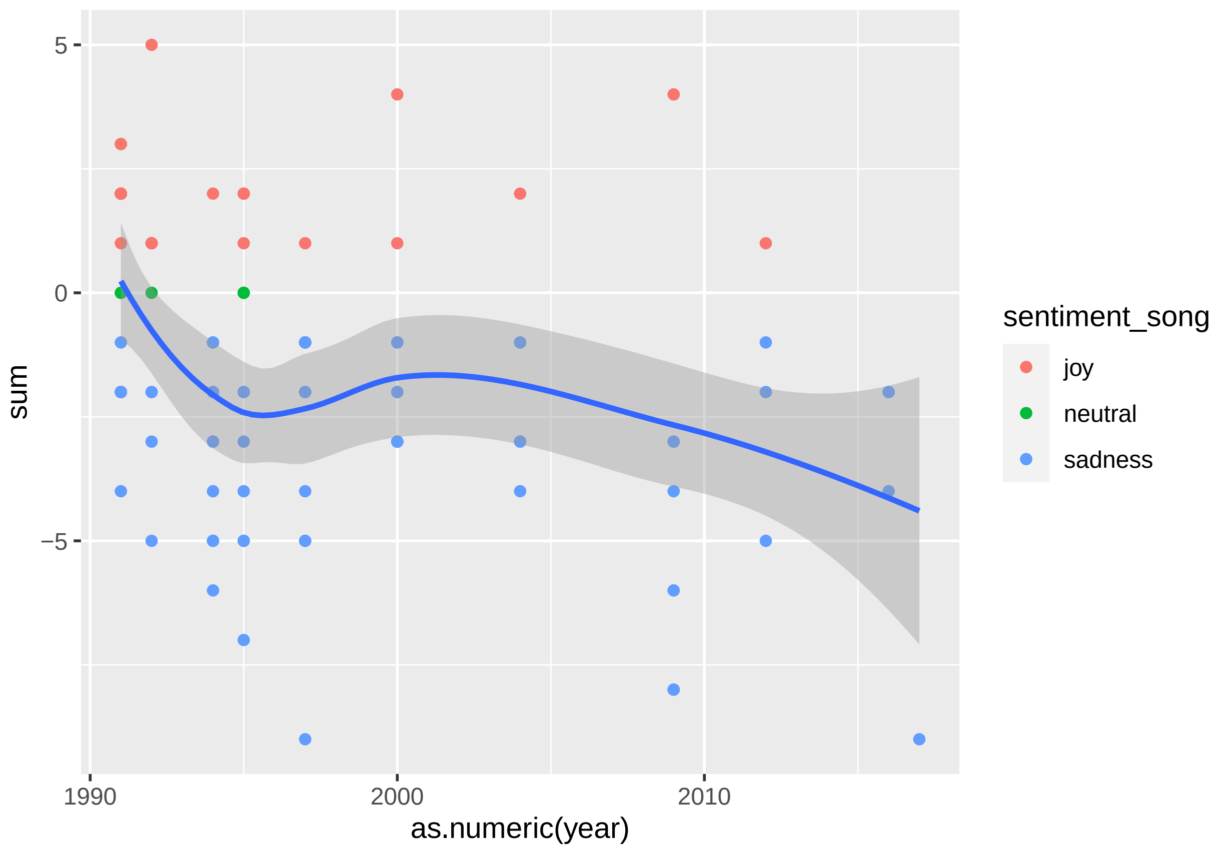

```
## RHCP

ggplot(RHCP_songs_joy_sadness, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
## Green Day

ggplot(Green_Day_songs_joy_sadness, aes(x = as.numeric(year), y = sum)) +
  geom_point(aes(color = sentiment_song))+ # add points to our plot, color-coded by president
  geom_smooth(method = "auto") # pick a method & fit a model
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




It looks like the artists song tend to be more sadd over time, it’s actually kinda of sad to see that. . . ..