Basic Quantitative Text Analytics and Visualization

Tabitha Hagen

2023-03-30

MIS 506 Module 3 - Assignment 1 – Basic Quantitative Text Analytics and Visualization

Word Frequency and TF-IDF

Task 1: Load the required libraries: tidyverse, tidytext, gridExtra

```
library (tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.1     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(tidytext)
#library(tokenizers)
#library(hcandersenr)
library(RColorBrewer) #define palattes
#library(wordcloud)
#library(wordcloud2) #wordclouds as .html
library(gridExtra) #viewing multiple plots together
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

Task 2: Read-in the following data set in R (the data set is available on Canvas - It may help to see Module 3 notes, examples, and the previous script from Module 2).

Dataset: prince_text.csv

#####The Data is the result of scraping Billboard Chart information and Prince Lyrics from various sites.

```
prince_original <-read_csv("prince_raw_data.csv")   # Read raw dataframe and save as prince_original
prince <-read_csv("prince_raw_data.csv")   # Read raw dataframe and save as prince
names(prince) #look at columns in dataframe
```

```
##  [1] "X"            "text"         "artist"       "song"         "year"
##  [6] "album"        "Release.Date" "US.Pop"       "US.R.B"       "CA"
## [11] "UK"           "IR"           "NL"           "DE"           "AT"
## [16] "FR"           "JP"           "AU"           "NZ"           "peak"
```

```
dim(prince)  #look at the dimensions (#observations, # columns)
```

```
## [1] 824  20
```

Task 3: Describe this dataset. How many variables and observations are there? (You can use the names() function to see the columns in the data frame)

There are 824 observations/rows and 20 variables/columns. Most of the variables detail the ranking of each country's music chart.

Task 4: The function select() allows you to select and rename the columns all in one step. So, select the columns lyrics, song, year, album, peak, US.Pop, and US.R.B. Change the name of the column "text" to "lyrics."

```
 prince <-prince %>%   # select and rename the columns all in one step
  select (lyrics = text, song, year, album, peak, US.Pop, US.R.B)
```

Task 5: Create a column name decade: Use dplyr's mutate() function to create the new decade field. One way to create the buckets is by utilizing ifelse() along with the %in% operator to filter by year and bin the songs into decades. Then store the results back into prince. Also, create a column name chart_level.

```
prince <- prince %>%
 mutate(decade =         # filter by year and bin the songs into decades
 ifelse(prince$year %in% 1978:1979, "1970s",
 ifelse(prince$year %in% 1980:1989, "1980s",
 ifelse(prince$year %in% 1990:1999, "1990s",
 ifelse(prince$year %in% 2000:2009, "2000s",
 ifelse(prince$year %in% 2010:2015, "2010s",
 "NA"))))))

prince <- prince %>%
 mutate (chart_level =      # create a column name chart_level
```

```
 ifelse(prince$peak %in% 1:10, "Top 10",
 ifelse(prince$peak %in% 11:100, "Top 100", "Uncharted")))
```

Task 6: Now look at the data set and see how many variables are there in the data set? (You can use the names() function to see the columns in the data frame)

```
names(prince) #look at columns in dataframe

## [1] "lyrics"      "song"        "year"        "album"       "peak"
## [6] "US.Pop"      "US.R.B"      "decade"      "chart_level"

dim(prince)  #look at the dimensions (#observations, # columns)

## [1] 824   9
```

There are 824 observations/rows and 9 variables/columns with the added column chart_level

Task 7: You can remove contractions by creating a function that handles most scenarios using gsub(), and then apply that function across all lyrics

```
#create a local function to fix or expand contractions
fix.contractions <- function(doc) {
 # "won't" is a special case as it does not expand to "wo not"
 doc <- gsub("won't", "will not", doc)
 doc <- gsub("can't", "can not", doc)
 doc <- gsub("n't", " not", doc)
 doc <- gsub("'ll", " will", doc)
 doc <- gsub("'re", " are", doc)
 doc <- gsub("You're", "You are", doc)
 doc <- gsub("'ve", " have", doc)
 doc <- gsub("'m", " am", doc)
 doc <- gsub("'d", " would", doc)
 # 's could be 'is' or could be possessive: it has no expansion
 doc <- gsub("'s", "", doc)
 return(doc)
}
# fix (expand) contractions
prince$lyrics <- sapply(prince$lyrics, fix.contractions)  # apply new
contractions function to df
```

Task 8: Transform the column lyrics into a tidy data structure, remove the stop words, undesirable words, and words with less than three characters. Below is a list of undesirable words in this data set.

The new data set is now tokenised with one word per row format along with the song from which it came, the year, the album, peak (which shows a song's placement on the Billboard charts), US.Pop, and US.R.B (which are peak chart positions for the US Pop and R&B charts).

```
# Transform the lyrics into a tidy data structure with one word per row.
tidy_prince <-prince %>%
  unnest_tokens("word",lyrics)
```

```
## Warning: Outer names are only allowed for unnamed scalar atomic inputs

# Warning - "Outer names are only allowed for unnamed scalar atomic inputs"

#Below is a list of undesirable words in this data set. Finding a list of
undesirable words is an iterative process of identifying and excluding common
words that don't add meaning to the analysis and undesirable words that are
not included in the stop words list. (placed in local variable)

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")

#First glance at words
tidy_prince %>%
  count(word, sort = TRUE)

## # A tibble: 8,742 × 2
##    word      n
##    <chr> <int>
##  1 i     10696
##  2 the    9744
##  3 you    6601
##  4 u      5133
##  5 a      4921
##  6 it     4591
##  7 not    4396
##  8 and    3992
##  9 me     3871
## 10 do     3712
## # … with 8,732 more rows

#Start of Cleaning
tidy_prince <- tidy_prince %>%
  #remove the stop words using the lexicon called stop_words from the
tidytext package
  anti_join(stop_words)%>%
  filter (!word %in% undesirable_words) %>%
  #remove the undesirable words using dplyr's filter() function with the %in%
operator
  distinct() %>% #get rid of any duplicate records using distinct()  verb
  filter(nchar(word) > 3) %>%  #remove words with less than 3 characters
using dplyr's filter() verb
  filter(
        !str_detect(word, "^\\b\\d+\\b"),          #remove numbers
        !str_detect(word, "\\s+"),                 #remove white spaces
```

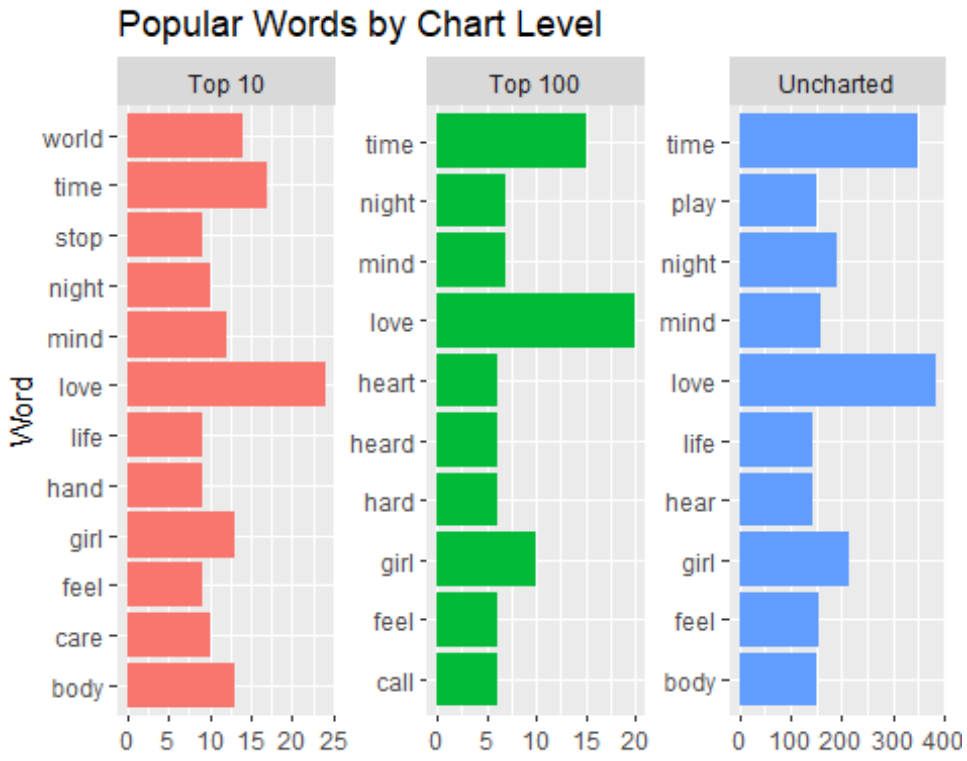
```
        !str_detect(word, "[^a-zA-Z]"))            #remove special characters
tidy_prince
```

```
## # A tibble: 36,657 × 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
##  7 7      1992 Symbol     3 7      61     1990s  Top 10      savior
##  8 7      1992 Symbol     3 7      61     1990s  Top 10      faire
##  9 7      1992 Symbol     3 7      61     1990s  Top 10      universe
## 10 7      1992 Symbol     3 7      61     1990s  Top 10      compare
## # … with 36,647 more rows
```

Task 9: Break up the most popular words per chart level. What are the most frequently used words by chart level? Are some words more popular in songs that reached the charts versus uncharted songs? Visualize the results.

```
#Look at most popular words per chart level using common dplyr's functions:

popular_words <- tidy_prince %>%
  #filter(peak == "1") %>% #look at only #1 songs
  select(year, song, word, chart_level) %>% #show year, song, chart_level
  arrange(year) #arrange in ascending order BY YEAR

# define some colors to use throughout
#my_colors <- c("#FFE1FF", "#FFE4E1", "#FFA07A", "#CDC8B1", "#C1CDCD",
"#EED5B7")
               #thistle1  #mistyrose #litesalmon #cornsilk3 #azure3
#bisque2

popular_words %>%
  group_by(chart_level) %>%
  count(chart_level, word, sort = TRUE)  %>%   # Counting the words for each
song
  top_n(10) %>% # look at top 10 words
  ggplot(aes(n, word, fill = chart_level)) +
    geom_col(show.legend = NULL) + # draw histogram
    xlab(NULL) +  # no label for x-axis
    ylab("Word") +  # y-axis label
    ggtitle("Popular Words by Chart Level") +
    scale_y_reordered() +  # makes sure the values are reordered
    facet_wrap(~chart_level, ncol = 3, scales = "free")  # put all the plots
next to each other
```

```
## Warning: `show.legend` must be a logical vector.
```



Task 10: Break up the most important/unique words per chart level using the bind_tf_idf()function. Take the original Prince Dataset, tokenize the text into individual words, remove the stop words, undesirable words, and words with less than three characters. Then use bind_tf_idf() to create new columns of tf, idf, and tf*idf.

```
#head(prince) # view a few instances from the data frame with chart_level
column

popular_tfidf_words <- prince %>%   # go back to data frame with chart_level
column
  unnest_tokens("word", lyrics) %>% # tokenize again
  anti_join(stop_words)%>% # leave out stop words
  filter(!word %in% undesirable_words) %>%
    # leave out undesirable words using dplyr's filter() function with the
%in% operator
  filter(nchar(word) > 3) %>% # remove words with less than 3 characters
using dplyr's filter() verb
  count(chart_level,song, word, sort = TRUE) %>%     # Counting the words for
each song
     # examine the most important words per song with the bind_tf_idf()
function.
  group_by(chart_level) %>%
  bind_tf_idf(word,song, n)  # also, create new columns of tf, idf, and
tf_idf

## Warning: Outer names are only allowed for unnamed scalar atomic inputs
```

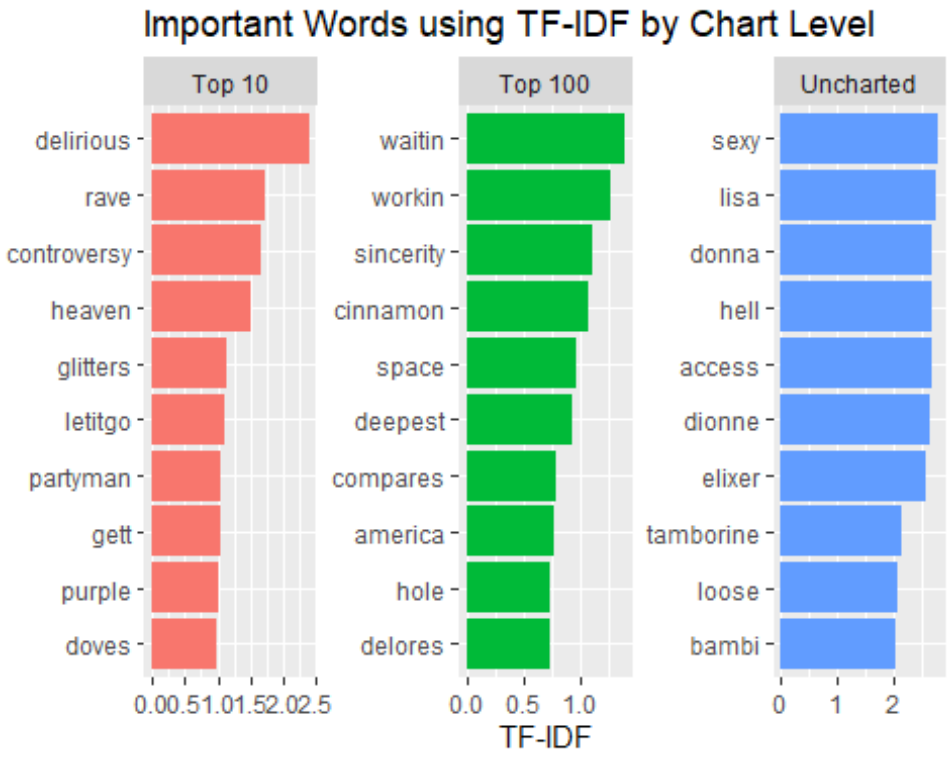
```
head(popular_tfidf_words) # view just a few observations
```

```
## # A tibble: 6 × 7
## # Groups:   chart_level [1]
##   chart_level song          word          n    tf   idf tf_idf
##   <chr>       <chr>         <chr>     <int> <dbl> <dbl>  <dbl>
## 1 Uncharted   push it up    push        112 0.389  3.76  1.46
## 2 Uncharted   shake         shake       112 0.56   3.02  1.69
## 3 Uncharted   party up      party        80 0.645  2.30  1.48
## 4 Uncharted   style         style        79 0.403  3.45  1.39
## 5 Uncharted   do your dance dance        71 0.384  1.88  0.720
## 6 Uncharted   beautiful     beautiful    62 0.459  3.04  1.40
```

Task 11: Visualize terms with high tf-idf by chart level. Pipe the results from question 10 into arrange() and descend by tf-idf. Visualize the results using ggplot() and geom_col() functions.

```
top_popular_tfidf_words <- popular_tfidf_words %>%
  arrange(desc(tf_idf)) %>%       # list in descending order
  mutate(word=reorder_within(word,tf_idf,song))%>%     # sort the counts in
all the plots
  top_n(10) %>%     # look at top 10 words

  ggplot(aes(tf_idf, word, fill = chart_level) )+
    geom_col(show.legend = NULL) + # draw histogram
    ylab(NULL) + # no y-axis label
    xlab("TF-IDF") + #name of x-axis label
    ggtitle("Important Words using TF-IDF by Chart Level") +
    scale_y_reordered() + # makes sure the values are reordered
    facet_wrap(~chart_level, ncol = 3, scales = "free")  # put all the plots
next to each other

top_popular_tfidf_words
```

```
## Warning: `show.legend` must be a logical vector.
```

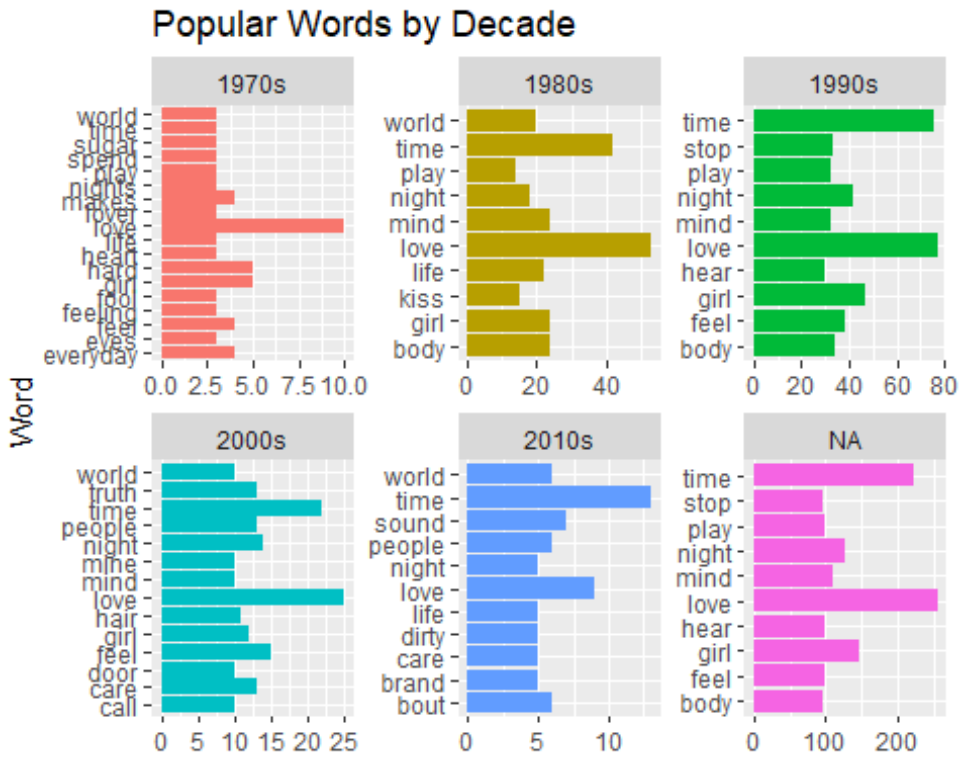


Task 12: Redo questions 9, 10, and 11 to identify and visualize the most popular words and the most unique/important words per decade.

A) Break up the most popular words per decade. What are the most frequently used words by chart level? Are some words more popular in songs that reached the charts versus uncharted songs? Visualize the results.

```
#Look at most popular words per decade using common dplyr's functions:

popular_words <- tidy_prince %>%
  #filter(peak == "1") %>% #Look at only #1 songs
  select(year, song, word, decade) %>% #show year, song, decade
  arrange(year) #arrange in ascending order BY YEAR

#define some colors to use throughout
#my_colors <- c("#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#D55E00",
"#D65E00")

popular_words %>%
  group_by(decade) %>%
  count(decade, word, sort = TRUE)  %>%   # Counting the words for each song
  top_n(10) %>% # look at top 10 words
  ggplot(aes(n, word, fill = decade)) +
    geom_col(show.legend = NULL) + # draw histogram
    xlab(NULL) +  # no label for x-axis
    ylab("Word") +  # y-axis label
    ggtitle("Popular Words by Decade") +
```

```
    scale_y_reordered() +  # makes sure the values are reordered
    facet_wrap(~decade, ncol = 3, scales = "free")  # put all the plots next
to each other

## Warning: `show.legend` must be a logical vector.
```



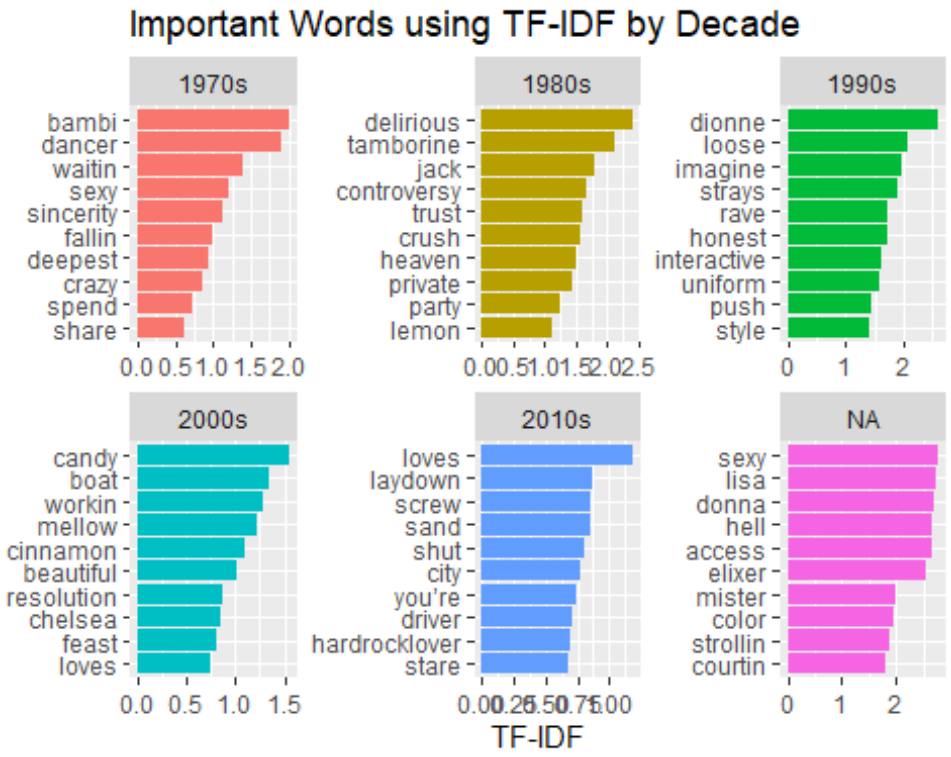
*B) Break up the most important/unique words per decade using the bind_tf_idf()function. Take the original Prince Dataset, tokenize the text into individual words, remove the stop words, undesirable words, and words with less than three characters. Then use bind_tf_idf() to create new columns of tf, idf, and tf*idf.*

```
#head(prince) # view a few instances from the data frame with decade column

popular_tfidf_words <- prince %>%   # go back to data frame with decade
column
  unnest_tokens("word", lyrics) %>% # tokenize again
  anti_join(stop_words)%>% # leave out stop words
  filter(!word %in% undesirable_words) %>%
    # leave out undesirable words using dplyr's filter() function with the
%in% operator
  filter(nchar(word) > 3) %>% # remove words with less than 3 characters
using dplyr's filter() verb
  count(decade,song, word, sort = TRUE) %>%     # Counting the words for each
song
     # examine the most important words per song with the bind_tf_idf()
function.
  group_by(decade) %>%
```

```
  bind_tf_idf(word,song, n)  # also, create new columns of tf, idf, and
tf_idf

## Warning: Outer names are only allowed for unnamed scalar atomic inputs

head(popular_tfidf_words) # view just a few observations

## # A tibble: 6 × 7
## # Groups:   decade [2]
##   decade song          word          n    tf   idf tf_idf
##   <chr>  <chr>         <chr>     <int> <dbl> <dbl>  <dbl>
## 1 1990s  push it up    push        112 0.389  3.76  1.46
## 2 NA     shake         shake       112 0.56   3.02  1.69
## 3 NA     party up      party        80 0.645  2.30  1.48
## 4 1990s  style         style        79 0.403  3.49  1.40
## 5 NA     do your dance dance        71 0.384  1.88  0.720
## 6 NA     beautiful     beautiful    62 0.459  3.04  1.40
```

C) Visualize terms with high tf-idf by decade. Pipe the results from question 10 into arrange() and descend by tf-idf. Visualize the results using ggplot() and geom_col() functions.

```
top_popular_tfidf_words <- popular_tfidf_words %>%
  arrange(desc(tf_idf)) %>%       # list in descending order
  mutate(word=reorder_within(word,tf_idf,song))%>%     # sort the counts in
all the plots
  top_n(10) %>%     # look at top 10 words

  ggplot(aes(tf_idf, word, fill = decade) )+
    geom_col(show.legend = NULL) + # draw histogram
    ylab(NULL) + # no y-axis label
    xlab("TF-IDF") + #name of x-axis label
    ggtitle("Important Words using TF-IDF by Decade") +
    scale_y_reordered() + # makes sure the values are reordered
    facet_wrap(~decade, ncol = 3, scales = "free")  # put all the plots next
to each other

top_popular_tfidf_words

## Warning: `show.legend` must be a logical vector.
```



Task 13: What are the insights that you now gather from the above charts? Does using tf-idf give you a different perspective on potentially important words? Can you tell that there are important and distinctive words per decades or chart levels? Explain.

I believe that popular words will show you what the media has published. It's like what they want you to hear on the radio. However, the TF-IDF most important or unique words gives you a better glimpse into the culture behind the lyrics similar to what you would find in the smaller, live music venues around a college.

Task 14: (Bonus) Look for song trends across time using dplyr's filter(),group_by() and summarise() functions:

- filter out everything else other than charted songs using peak > 0.
- group the songs by decade and chart_level.
- pipe the group_by object into summarise() using n() to count the number of songs.
- using ggplot() and geom_bar(), create a bar chart where the x-axis represents decades, the y axis represents the number of songs, and fill the bars with the chart level category.

Most of the coding for this question is provided. Please fill out the blanks and describe the graph.

Most of the chart shows that the Top 10 songs overlap the Top 100 songs because the Top 100 songs include the Top 10.

```r
charted_songs_over_time <- tidy_prince %>%
 filter (peak>0) %>% # filter out songs that aren't on a chart
 group_by (decade, chart_level) %>% # group by decade and chart level
 summarise (number_of_songs = n()) # count number of songs

 charted_songs_over_time %>%
 ggplot() +
    geom_bar(aes(x = decade, y = chart_level, # create a bar chart
    fill = chart_level), stat = "identity") + # plot aesthetics
    labs(x = NULL, y = "Song Count") + # label axis
    ggtitle("Charted Songs")  # plot title
```