Module 3 Homework

# Problem 1

## Load the required libraries:

Load the required libraries:

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidytext) #text mining  
library(gridExtra) #viewing multiple plots together

##   
## Attaching package: 'gridExtra'

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

# Problem 2

## Read-in the following data set in R (the data set is available on Canvas

Read-in the following data set in R: prince\_text.csv

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

# Problem 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 rows and 20 columns in the dataset. columns include text - Lyrics in song song - name of the song year - The year the song was released album - The album the song was on 13 columns representing the level it reached on different countries charts

prince <- read\_csv("~/MIS 506/Module 3/Homework/prince\_text.csv")

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## .default = col\_character(),  
## X = col\_double(),  
## year = col\_double(),  
## peak = col\_double()  
## )  
## i Use `spec()` for the full column specifications.

names(prince)

## [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"

prince

## # A tibble: 824 x 20  
## X text artist song year album Release.Date US.Pop US.R.B CA UK   
## <dbl> <chr> <chr> <chr> <dbl> <chr> <chr> <chr> <chr> <chr> <chr>  
## 1 49 "All 7~ prince 7 1992 Symb~ 17 Nov. 1992 7 61 3 27   
## 2 669 "319, ~ prince 319 NA <NA> <NA> <NA> <NA> <NA> <NA>   
## 3 78 "Don't~ prince 1999 1982 1999 24 Sep. 1982 12 4 6 25   
## 4 475 "Princ~ prince 2020 NA Othe~ <NA> <NA> <NA> <NA> <NA>   
## 5 811 "One m~ prince 3121 2006 3121 21 Mar. 2006 1 1 5 9   
## 6 478 "Princ~ prince 7779~ NA <NA> <NA> <NA> <NA> <NA> <NA>   
## 7 208 "u, yo~ prince u NA <NA> <NA> <NA> <NA> <NA> <NA>   
## 8 397 "ed if~ prince ed i~ NA <NA> <NA> <NA> <NA> <NA> <NA>   
## 9 714 "Nuest~ prince cont~ NA <NA> <NA> <NA> <NA> <NA> <NA>   
## 10 898 "Ohh, ~ prince 1000~ NA <NA> <NA> <NA> <NA> <NA> <NA>   
## # ... with 814 more rows, and 9 more variables: IR <chr>, NL <chr>, DE <chr>,  
## # AT <chr>, FR <chr>, JP <chr>, AU <chr>, NZ <chr>, peak <dbl>

# Problem 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 (lyrics = text, song, year, album, peak, US.Pop, US.R.B)

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

## # A tibble: 824 x 9  
## lyrics song year album peak US.Pop US.R.B decade chart\_level  
## <chr> <chr> <dbl> <chr> <dbl> <chr> <chr> <chr> <chr>   
## 1 "All 7 and we'll ~ 7 1992 Symbol 3 7 61 1990s Top 10   
## 2 "319, 'bout time,~ 319 NA <NA> NA <NA> <NA> NA Uncharted   
## 3 "Don't worry, I w~ 1999 1982 1999 2 12 4 1980s Top 10   
## 4 "Prince\nMiscella~ 2020 NA Other~ NA <NA> <NA> NA Uncharted   
## 5 "One mix, one mix~ 3121 2006 3121 1 1 1 2000s Top 10   
## 6 "Prince\nMiscella~ 77793~ NA <NA> NA <NA> <NA> NA Uncharted   
## 7 "u, you're so fin~ u NA <NA> NA <NA> <NA> NA Uncharted   
## 8 "ed if I do\nTell~ ed if~ NA <NA> NA <NA> <NA> NA Uncharted   
## 9 "Nuestra presenta~ contr~ NA <NA> NA <NA> <NA> NA Uncharted   
## 10 "Ohh, ah, yeah\nU~ 1000 ~ NA <NA> NA <NA> <NA> NA Uncharted   
## # ... with 814 more rows

# Problem 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)

There are 9 variables (columns) in the dataset after these operations.

# Problem 7

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

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("'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)

# Problem 8

## Transform the column lyrics into a tidy data structure, remove the stop words, undesirable words, and words with less than 3 characters.

tidy\_prince <-prince %>%  
 unnest\_tokens("word",lyrics)

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

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")  
  
#Start of Cleaning  
tidy\_prince <- tidy\_prince %>%   
 anti\_join(stop\_words)%>% #remove the stop words using the lexicon called stop\_words from the tidytext package  
 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

## Joining, by = "word"

tidy\_prince

## # A tibble: 36,657 x 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

# Problem 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 verses uncharted songs? Visualize the results

The words “love” and “time” are 1 and 2 in all charts, and “girl”, “night”, and “mind” are in the top 7 in all of them. “world” comes in at #3 on the Top 10 chart and doesn’t appear in the others.

Unique words in the Top 10 chart are: care, stop, life, and hand. Unique words in the Top 100 chart are: heart, heard, heart, feel. Unique words in the Uncharted list are: feel, play, and hear

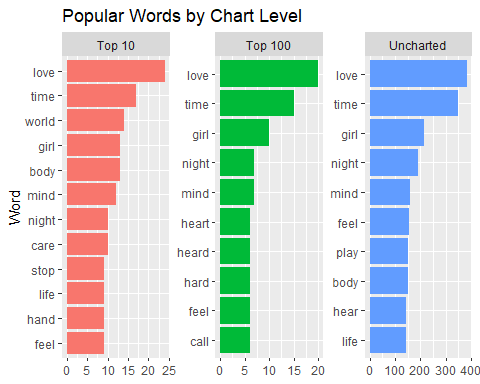
popular\_words\_by\_level <- tidy\_prince %>%  
 group\_by(chart\_level) %>%  
 count(chart\_level, word, sort = TRUE)   
   
popular\_words\_by\_level

## # A tibble: 8,825 x 3  
## # Groups: chart\_level [3]  
## chart\_level word n  
## <chr> <chr> <int>  
## 1 Uncharted love 386  
## 2 Uncharted time 348  
## 3 Uncharted girl 214  
## 4 Uncharted night 190  
## 5 Uncharted mind 160  
## 6 Uncharted feel 155  
## 7 Uncharted play 153  
## 8 Uncharted body 150  
## 9 Uncharted hear 145  
## 10 Uncharted life 144  
## # ... with 8,815 more rows

popular\_words\_by\_level %>%   
 arrange(desc(chart\_level,n)) %>%   
 mutate(word=reorder\_within(word,n,chart\_level)) %>% #sort the counts in all the plots  
 top\_n(10) %>%  
 ggplot(aes(n, word, fill = chart\_level)) +  
 geom\_col(show.legend = NULL) +  
 xlab(NULL) +   
 ylab("Word") +  
 ggtitle("Popular Words by Chart Level") +  
 scale\_y\_reordered() +  
 facet\_wrap(~chart\_level, ncol = 3, scales = "free")

## Selecting by n

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



# Problem 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 3 characters. Then use bind\_tf\_idf() to create new columns of tf, idf, and tf\*idf.

popular\_tfidf\_words\_level <- prince %>%  
 unnest\_tokens("word", lyrics) %>%  
 anti\_join(stop\_words)%>%   
 filter(!word %in% undesirable\_words) %>%  
 filter(nchar(word) > 3) %>%  
 group\_by(chart\_level) %>%  
 count(chart\_level, word, sort = TRUE) %>% #Counting the words for each song  
 bind\_tf\_idf(word,chart\_level, n) #examine the most important words per song with the bind\_tf\_idf() function.

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

## Joining, by = "word"

popular\_tfidf\_words\_level

## # A tibble: 9,021 x 6  
## # Groups: chart\_level [3]  
## chart\_level word n tf idf tf\_idf  
## <chr> <chr> <int> <dbl> <dbl> <dbl>  
## 1 Uncharted love 1776 0.0282 0 0   
## 2 Uncharted time 827 0.0131 0 0   
## 3 Uncharted girl 635 0.0101 0 0   
## 4 Uncharted stop 504 0.00800 0 0   
## 5 Uncharted dance 472 0.00749 0 0   
## 6 Uncharted feel 454 0.00721 0 0   
## 7 Uncharted body 404 0.00641 0 0   
## 8 Uncharted night 396 0.00629 0 0   
## 9 Uncharted life 391 0.00621 0 0   
## 10 Uncharted rock 372 0.00590 0.405 0.00239  
## # ... with 9,011 more rows

# Problem 11

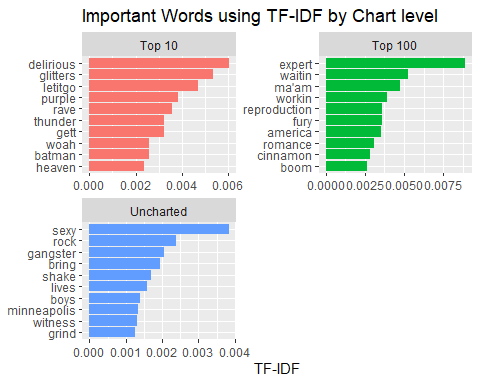
## Visualize terms with high tf-idf by chart level. Pipe the results from question 10 into arrange() and descend by tf-idf. Visualize the results using ggplot() and geom\_col() functions

top\_popular\_tfidf\_words\_level <- popular\_tfidf\_words\_level %>%  
 arrange(desc(chart\_level,tf\_idf)) %>%   
 mutate(word=reorder\_within(word,tf\_idf,chart\_level))%>% #sort the counts in all the plots  
 top\_n(10) %>%   
 ggplot(aes(tf\_idf, word, fill = chart\_level) )+  
 geom\_col(show.legend = NULL) +  
 ylab(NULL) +   
 xlab("TF-IDF") +  
 ggtitle("Important Words using TF-IDF by Chart level") +  
 scale\_y\_reordered() +  
 facet\_wrap(~chart\_level, ncol = 2, scales = "free")

## Selecting by tf\_idf

top\_popular\_tfidf\_words\_level

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



# Problem 12

## Redo questions 9, 10, and 11 to identify and visualize the most popular words and the most unique/important words per decade

Words popular across all Decades: love, time, girl …the 70s: hard, makes, everyday …the 80s: mind, life, world, kiss …the 90s: night, feel, body, stop …the 00s: night, truth, people, care …the 10s: sound, world, people, dirty …NA: night, mind, play, feel

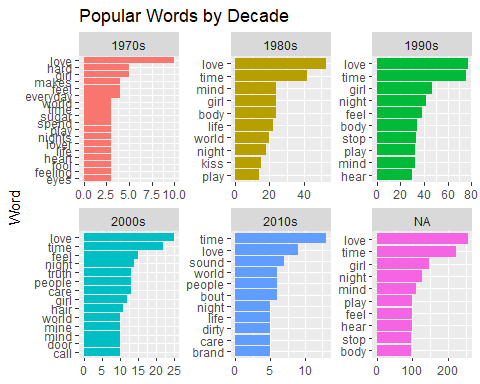
popular\_words\_by\_decade <- tidy\_prince %>%  
 group\_by(decade) %>%  
 count(decade, word, sort = TRUE)   
   
popular\_words\_by\_decade

## # A tibble: 12,481 x 3  
## # Groups: decade [6]  
## decade word n  
## <chr> <chr> <int>  
## 1 NA love 256  
## 2 NA time 224  
## 3 NA girl 148  
## 4 NA night 126  
## 5 NA mind 109  
## 6 NA feel 100  
## 7 NA play 100  
## 8 NA hear 99  
## 9 NA body 97  
## 10 NA stop 97  
## # ... with 12,471 more rows

popular\_words\_by\_decade %>%   
 arrange(desc(decade,n)) %>%   
 mutate(word=reorder\_within(word,n,decade)) %>% #sort the counts in all the plots  
 top\_n(10) %>%  
 ggplot(aes(n, word, fill = decade)) +  
 geom\_col(show.legend = NULL) +  
 xlab(NULL) +   
 ylab("Word") +  
 ggtitle("Popular Words by Decade") +  
 scale\_y\_reordered() +  
 facet\_wrap(~decade, ncol = 3, scales = "free", shrink = FALSE)

## Selecting by n

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



popular\_tfidf\_words\_decade <- prince %>%  
 unnest\_tokens("word", lyrics) %>%  
 anti\_join(stop\_words)%>%   
 filter(!word %in% undesirable\_words) %>%  
 filter(nchar(word) > 3) %>%  
 group\_by(decade) %>%  
 count(decade, word, sort = TRUE) %>% #Counting the words for each song  
 bind\_tf\_idf(word,decade, n) #examine the most important words per song with the bind\_tf\_idf() function.

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

## Joining, by = "word"

popular\_tfidf\_words\_decade

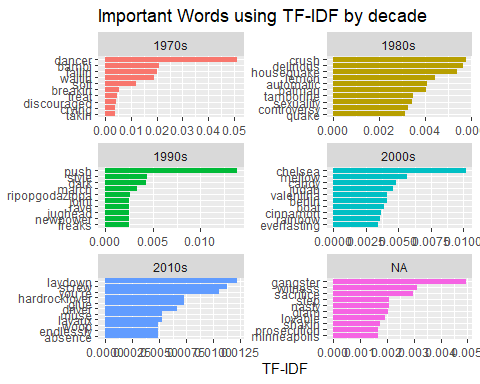
## # A tibble: 12,703 x 6  
## # Groups: decade [6]  
## decade word n tf idf tf\_idf  
## <chr> <chr> <int> <dbl> <dbl> <dbl>  
## 1 NA love 1190 0.0278 0 0   
## 2 NA time 546 0.0127 0 0   
## 3 NA girl 513 0.0120 0 0   
## 4 NA stop 406 0.00948 0 0   
## 5 1990s love 356 0.0260 0 0   
## 6 NA dance 337 0.00787 0 0   
## 7 NA feel 335 0.00782 0 0   
## 8 NA life 296 0.00691 0 0   
## 9 NA rock 277 0.00647 0.182 0.00118  
## 10 NA body 269 0.00628 0 0   
## # ... with 12,693 more rows

top\_popular\_tfidf\_words\_decade <- popular\_tfidf\_words\_decade %>%  
 arrange(desc(decade,tf\_idf)) %>%   
 mutate(word=reorder\_within(word,tf\_idf,decade))%>% #sort the counts in all the plots  
 top\_n(10) %>%   
 ggplot(aes(tf\_idf, word, fill = decade) )+  
 geom\_col(show.legend = NULL) +  
 ylab(NULL) +   
 xlab("TF-IDF") +  
 ggtitle("Important Words using TF-IDF by decade") +  
 scale\_y\_reordered() +  
 facet\_wrap(~decade, ncol = 2, scales = "free")

## Selecting by tf\_idf

top\_popular\_tfidf\_words\_decade

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



# Problem 13

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

The tf-idf approach does offer a different type of approach and analysis. By separating out the “unimportant” words, it allows us to look at those words that make the data sample special. Unique words like “crush”, “delirious”, and “controversy” really show a flavor to the songs that have charted or not or from teh decade they were in.

While looking across decades, the same themes of girl and love are there, but later in his career it looks like there is a more mature vibe, using words like “peace” and “world”.

# Problem 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

Isn’t “number\_of\_songs” really the number of words in the songs? That’s why the numbers are in the hundreds. If we were just interested in the number of songs we could just filter it out and do the analysis, but then we wouldn’t be doing any text analysis.

charted\_songs\_over\_time <- tidy\_prince %>%  
 filter (peak > 0) %>%  
 group\_by (decade, chart\_level) %>%  
 summarise (number\_of\_songs = n())

## `summarise()` has grouped output by 'decade'. You can override using the `.groups` argument.

charted\_songs\_over\_time %>%   
 ggplot() +   
 geom\_bar(aes(x = decade, y = number\_of\_songs,   
 fill = chart\_level), stat = "identity") +  
 labs(x = NULL, y = "Song Count") +  
 ggtitle("Charted Songs")

