Diane Sullivan

2023-04-01

## R Markdown

Load the required libraries:

#install.packages("tidyverse")  
#install.packages("tidytext")  
#install.packages("gridExtra")  
  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.8 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.1  
## ✔ readr 2.1.2 ✔ forcats 0.5.2  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(tidytext) #text mining

## Warning: package 'tidytext' was built under R version 4.2.3

library(gridExtra) #viewing multiple plots together

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

prince <-read\_csv("prince\_raw\_data.csv")

## Rows: 824 Columns: 20  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (17): text, artist, song, album, Release.Date, US.Pop, US.R.B, CA, UK, I...  
## dbl (3): X, year, peak  
##   
## 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.

#This is a datasetof Prince Lyrics gathered by scaping Billboard chart information. There are 824 rows (observations) and 20 variables.

prince %>%  
 names()

## [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 <-prince %>%   
 select (lyrics = text, song, year, album, peak, US.Pop, US.R.B)

#Create a new column named decade. Then make buckets with the ifelse command and%in% to filter by year and put songs in the appropriate bins by decade. Store this new result back to original df.

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 %>%  
 names()

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

#This new data frame has 9 variables.

#I will 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)

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")

## Tidy Text Format

Transform the lyrics into a tidy data structure with one word per row. The tidytext’s unnest\_tokens() function requires at least two arguments: the output column name that will be created as the text is unnested into it (“word”, in this case), and the input column that holds the current text, which is “lyrics”.

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

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).

## Pre-processing Text

#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"

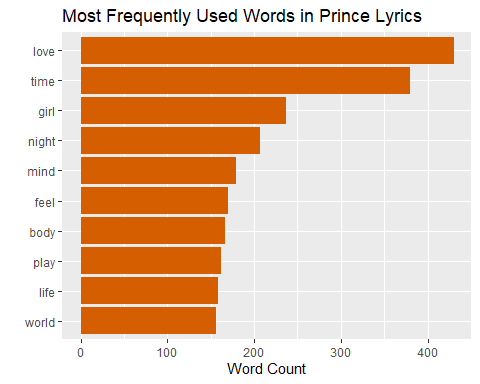
#tidy\_prince

## Word Frequesncy

We can use the tidy format to do a simple evaluation and visualization of the most frequently used words in the full set of lyrics

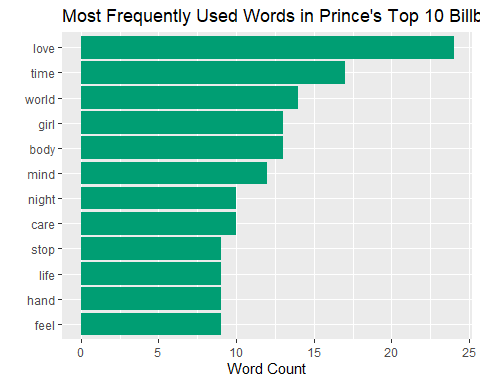
#define some colors to use throughout  
my\_colors <- c("#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#D55E00", "#D65E00")  
  
  
tidy\_prince %>%  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[5]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince Lyrics") +  
 coord\_flip()

## Selecting by n

 ## Word Frequency Break apart the most popular words per chart level; I will use the tidy format to do a simple evaluation and visualization of the most frequently used words by chart level in the full set of lyrics

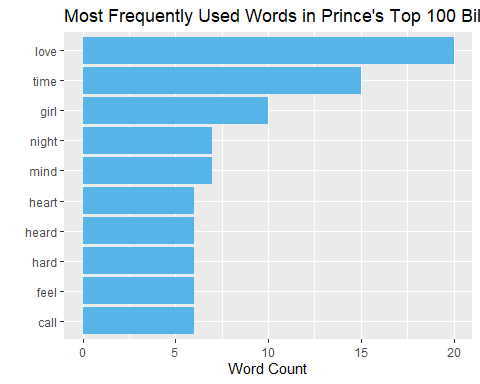
tidy\_prince %>%  
 filter(chart\_level == "Top 10") %>% #filter for Top 10 chart level  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[3]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's Top 10 Billboard Chart's Lyrics") +  
 coord\_flip()

## Selecting by n



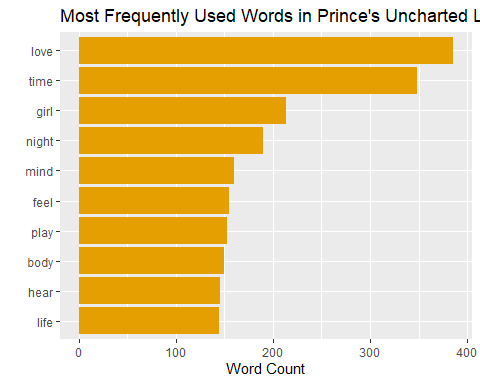
tidy\_prince %>%  
 filter(chart\_level == "Top 100") %>% #filter for Top 100 chart level  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[2]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's Top 100 Billboard Chart's Lyrics") +  
 coord\_flip()

## Selecting by n



tidy\_prince %>%  
 filter(chart\_level == "uncharted") %>% #filter for Top 10 chart level  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[1]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's Uncharted Lyrics") +  
 coord\_flip()

## Selecting by n



#Q9 Some frequently used words in uncharted songs that don’t appear in the top 10 lists of Top 10 or Top 100 songs include: play and hear. Popular songs in the charts contain these words that do not appear often in uncharted songs: world, care, stop, hand, heart, hard, call.

Here is the original Prince data frame and unnesting tokens to words, removing undesirable words, words with less than 3 characters, and the stop words. Then using bind\_tf\_idf() to create new columns of tf, idf, and tf\*idf.

popular\_tfidf\_words <- 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 chart level  
 bind\_tf\_idf(word,chart\_level, n) #examine the most important words per chart level with the bind\_tf\_idf() function.

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

## Joining, by = "word"

#Q10:

head(popular\_tfidf\_words)

## # A tibble: 6 × 6  
## # Groups: chart\_level [1]  
## 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

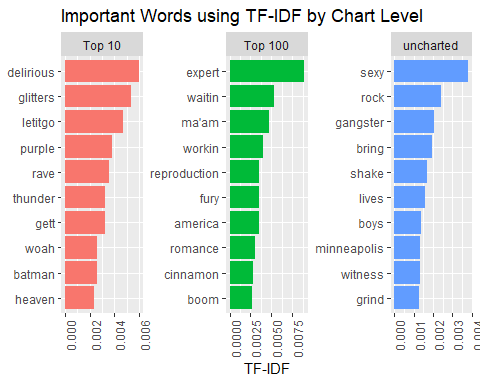
Let’s look Prince’s songs by chart level. Let’s look at terms with high tf-idf. Pipe the results from above into arrange() and descend by tf-idf and visualize the results using ggplot() and geom\_col() functions. #Q11

list <- c("uncharted", "Top 10", "Top 100")  
  
top\_popular\_tfidf\_words <- popular\_tfidf\_words %>%  
 filter(chart\_level %in% list) %>% #We will only look at chartlevels.  
 arrange(desc(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) +   
 theme(axis.text.x = element\_text(angle = 90)) +  
 xlab("TF-IDF") +  
 ggtitle("Important Words using TF-IDF by Chart Level") +  
 scale\_y\_reordered() +  
 facet\_wrap(~chart\_level, ncol = 3, scales = "free")

## Selecting by tf\_idf

top\_popular\_tfidf\_words

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



The formula for tf-idf can be summarized below:

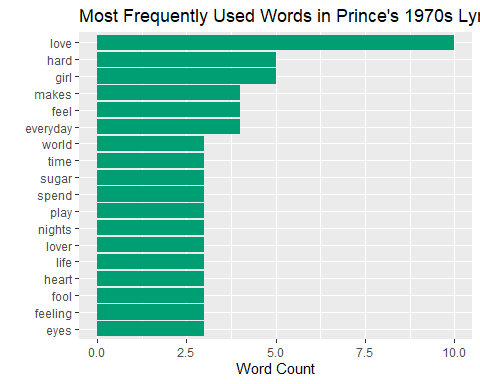
* Term Frequency (tf): Number of times a term occurs in a document
* Document Frequency (df): Number of documents that contain each word
* Inverse Document Frequency (idf) = 1/df [The idf of any term is therefore a higher number for words that occur in fewer of the documents in the collection.]

## Word Frequency

Break up the most popular words per decade; I will use the tidy format to do a simple evaluation and visualization of the most frequently used words by decade in the full set of lyrics

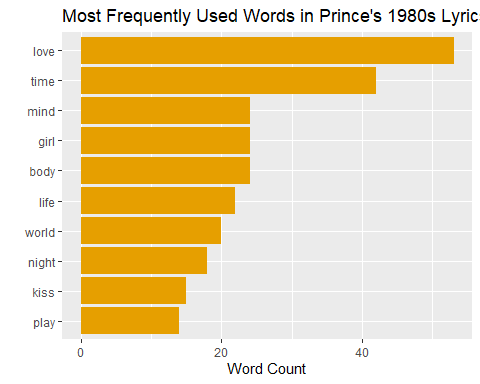
tidy\_prince %>%  
 filter(decade == "1970s") %>% #filter for 1970s lyrics  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[3]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's 1970s Lyrics") +  
 coord\_flip()

## Selecting by n



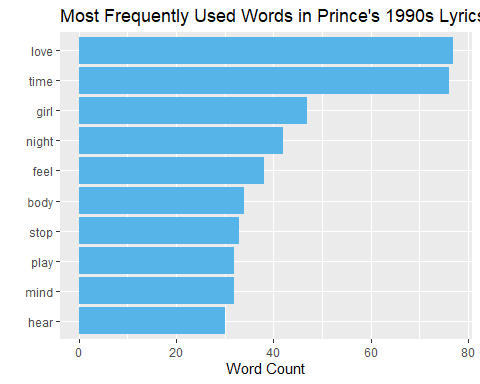
tidy\_prince %>%  
 filter(decade == "1980s") %>% #filter for 1980s lyrics  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[1]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's 1980s Lyrics") +  
 coord\_flip()

## Selecting by n



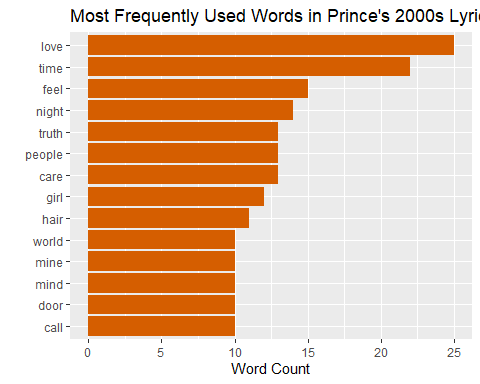
tidy\_prince %>%  
 filter(decade == "1990s") %>% #filter for 1990s lyrics  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[2]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's 1990s Lyrics") +  
 coord\_flip()

## Selecting by n



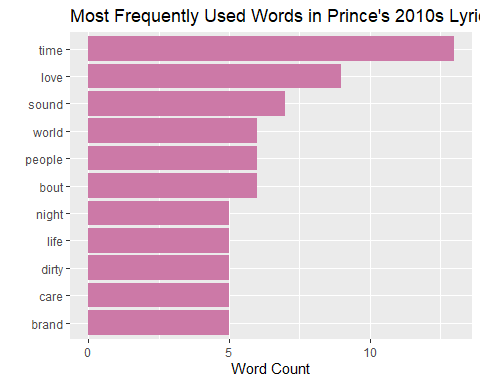
tidy\_prince %>%  
 filter(decade == "2000s") %>% #filter for 2000s lyrics  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[5]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's 2000s Lyrics") +  
 coord\_flip()

## Selecting by n



tidy\_prince %>%  
 filter(decade == "2010s") %>% #filter for 2010s lyrics  
 count(word, sort = TRUE) %>% #get the n top words from the tidied, clean, filtered dataset using count() and top\_n()   
 top\_n(10) %>%  
 ungroup() %>%   
 mutate(word = reorder(word, n)) %>% #sort words according to the count using reorder()and reassign the ordered value to word using mutate()  
 ggplot() +  
 geom\_col(aes(word, n), fill = my\_colors[4]) +  
 xlab("") +   
 ylab("Word Count") +  
 ggtitle("Most Frequently Used Words in Prince's 2010s Lyrics") +  
 coord\_flip()

## Selecting by n



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) %>%   
 bind\_tf\_idf(word,decade, n)

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

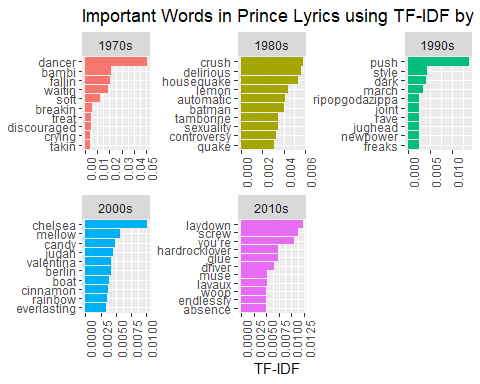
## Joining, by = "word"

list <- c("1970s", "1980s", "1990s", "2000s", "2010s")  
  
top\_popular\_tfidf\_words\_decade <- popular\_tfidf\_words\_decade %>%  
 filter(decade %in% list) %>% #We will only look at chartlevels.  
 arrange(desc(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") +  
 theme(axis.text.x = element\_text(angle = 90)) +  
 ggtitle("Important Words in Prince Lyrics using TF-IDF by Decades") +  
 scale\_y\_reordered() +  
 facet\_wrap(~decade, ncol = 3, scales = "free")

## Selecting by tf\_idf

top\_popular\_tfidf\_words\_decade

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



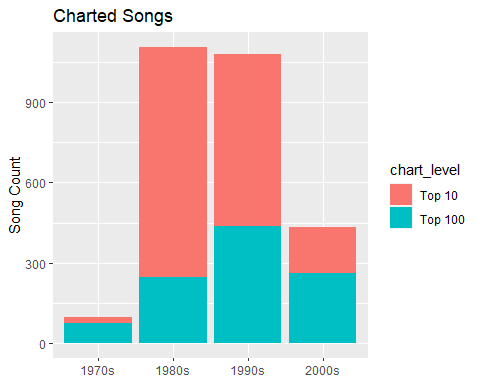
#Q13 Frequent words, while interesting, doesn’t translate to important words. Even after processing out the frequent stop words, still, frequent words may not have importance. Using tf-idf gives me insigts into words that occur in fewer of the documents in the collections of Prince lyrics and are frequent enough to have some weight or to matter. I noticed a softer feeling in the 1970s tf-idf words: dancer, bambi, soft, waitin’ versus the 2010 lyrics: screw, hardrock lover, driver.This was not conveyed by simply comparing frequent words sorted by the decades.I noted distinctive words in the decades as well, for example, “lemon” in the 80s and “candy” “cinnamon” “rainbow” in the 2000s.

#Bonus

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")

 #Q14: The top 100 Billboard chart songs are shown in teal and the top 10 billboard chart songs are shown in red, grouped by decades. These stacked bars make it easy to compare within a decade; in the 1970s and 2000s, Prince had more Top 100 songs vs Top 10. In the 1980s and 1990s, the stacked bars indicate that he had more Top 10s vs Top 100s. The y-axis gives the song count for each bar, which represents a decade.