Module 5 - MATH 241

Contents

Posted: April 20, 2022

Due: April 29, 2022 to May 4, 2022

Instructions:

- **Please knit your R Markdown file as PDF file.** Don't put your name in any part of the document. Your document upload will correspond to your name automatically in Gradescope.
- **Please provide complete solutions for each problem.** If it involves mathematical computations, explanations, or analysis, please provide your reasoning or detailed solutions.
- **It is recommended that you work on the module incrementally.** The modules are designed for you to work on it at anytime until the designated deadline. This method will let you transfer what you have learned from lectures and mini-assignments in small parts rather than stressing on completing the entire assignment at the last minute.
- **Note that some problems have multiple solutions or ways to solve it.** Make sure that your solutions are clear enough to showcase your work and understanding of the material.
- **Creativity and collaborations are encouraged.** Use all of the resources you have and what you need to complete the module. Each student must take personal responsibility and submit their work individually. Please abide by the Reed College Honor Principle.
- **Code Blocks.** You can add and change R code blocks anywhere in this document but for code blocks with comments `### [BEGIN]` and `### [END]` are indicators that you are expected to change the R code specifically in between those lines.
- If you can't figure out why a code chunk is preventing you from knitting the document, replace "r" at the top of the code chunk with "r eval = FALSE, echo = TRUE." The code will not be executed, but it will be printed in your pdf, earning you some partial credit.
- Please follow the general module guidelines written in the course website. If you have any questions, please send them to the instructor as a direct message on Slack or through email.

R Packages:

- Below are pre-loaded general packages required for this module assignment. You can load more packages here or throughout the module if necessary.
- Note that you need to install R packages before you can use them. You can use the `install.packages()` in the R console, or go to the "Tools" tab and click "Install Packages..." in R Studio.
- Be careful on loading R packages because sometimes any two packages can have conflicting functions when calling them.

```
# pre-load packages here
library(tidyverse)
library(tidytext)
library(textdata)
```

I. Understanding Movies Data Utilizing Visualizations (Homework)

Materials

- The `wiki_movie_plots_deduped.csv` data set, which contains 34,886 movie descriptions scraped from Wikipedia entries of movies. This data set - which is NOT provided with this Module - was downloaded from Kaggle Wikipedia Movie Plots. Below lists the variables in the dataset. Please go to the link provided and download the csv file before you work on the exercises.
  - *Release Year*: The release year of the movie
  - *Title*: The title of the movie
  - *Origin/Ethnicity*: Movie origin (e.g. American, etc.)
  - *Director* - The name(s) of the movie director
  - *Cast* - A list of main actors/cast of the movie separated by a comma
  - *Genre* - The genre(s) of the movie separated by a comma
  - *Wiki Page* - Wikipedia URL from where the movie description was scraped
  - *Plot* - The summarized plot of the movie in the English language

```
# load data
movie_plots <- read_csv("wiki_movie_plots_deduped.csv") %>%
  mutate(year = as.numeric(`Release Year`), # convert year variable to numeric
         origin = `Origin/Ethnicity`, # change origin column name
         Title = str_to_lower(Title), # lowercase the titles
         Plot = str_to_lower(Plot)) %>% # lowercase the plots
  select(-`Release Year`,-`Origin/Ethnicity`, -`Wiki Page`) # remove some columns
```

- The `stop_words` list from the `tidytext` package, which is the stopword list retrieved from the SMART lexicon for text categorization benchmarking as explained by Lewis et al. (2004).

```
# load the stopwords lexicon
swd_list <- tidytext::stop_words %>%
  filter(lexicon == "SMART") %>%
  select(word)
swd_list <- swd_list$word
```

- The `lexicon_nrc_eil()` lexicon from the `textdata` package, which is the sentiment and emotions lexicon from the National Research Council (NRC) of Canada. The NRC lexicon is a data set of categorized words according to there associated human emotions. The `lexicon_nrc_eil()` function loads a set of words with associated emotions in 4 categories with a numerical variable measuring its intensity (Mohammad, 2018). See the Sentiment and Emotion Lexicon for more information.

```
# load NRC emotion lexicons
emotions_4_with_intensity <- textdata::lexicon_nrc_eil()
```

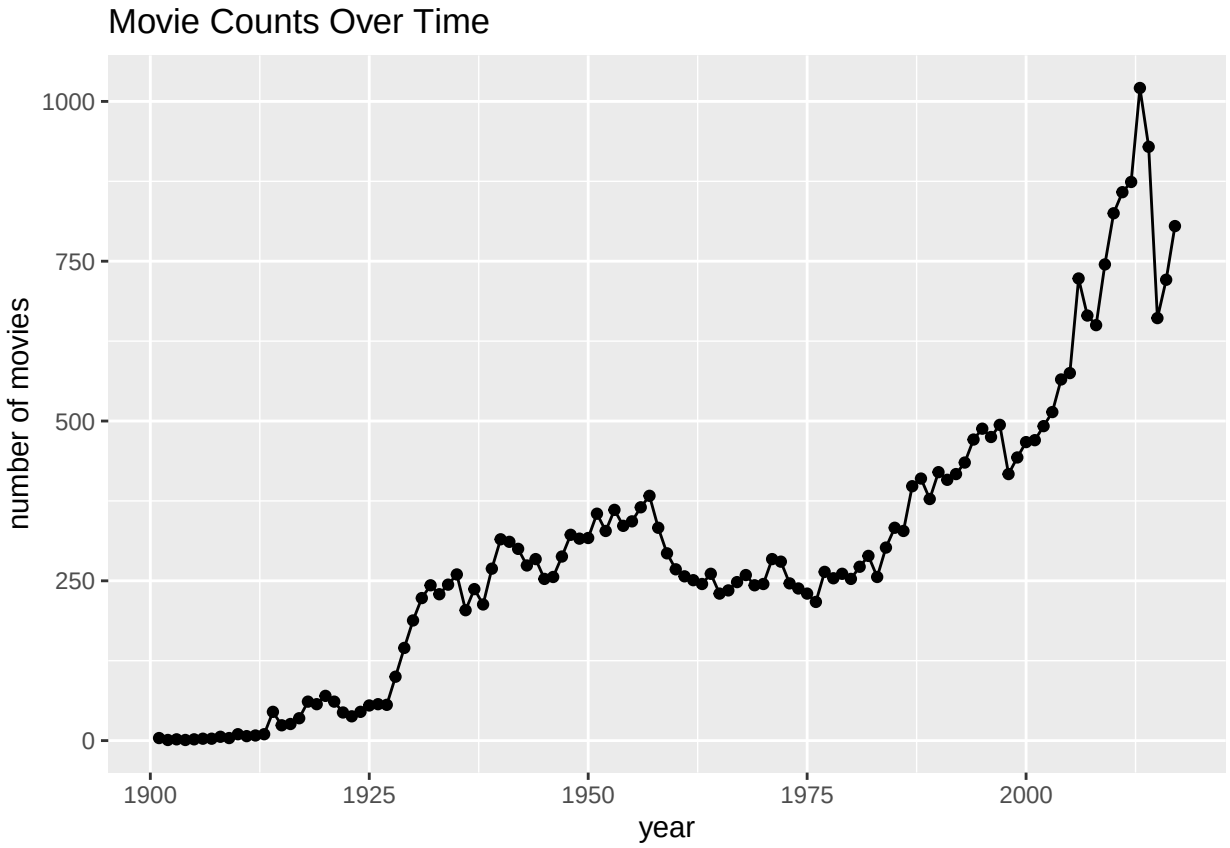
Exercises

The purpose of these exercises is for you to continue practicing on data wrangling and visualization focusing on categorized data and time-series data.

1. **Number of movies per year.** Below is an example R code that creates a visualization that shows a time-series plot of the total number of movies each year. Your task is to modify the R code below so that you can plot the time-series movie counts showing the top 9 most popular genre (excluding the "unknown" genre) of all time. Note that the data set contains single genre movies as well as multiple genre movies - we are only considering single genre movies. The final result should be a visualization with 9 time-series plots. Adjust the scales and colors accordingly and provide a paragraph of your observations. Hint: Count the number of movies in each genre, sort it and then get the top 9 genres and then - using that vector of genres, filter your data using `filter(...)` and then summarize it by using `group_by(...) %>% summarise(...)`.

```
### [BEGIN] - WORK ON YOUR DATA WRANGLING HERE
# count the number of movies each year
movie_counts_peryear <- movie_plots %>%
                          group_by(year) %>%
                          summarise(counts = n())
### [END] - WORK ON YOUR DATA WRANGLING HERE
```

```
### [BEGIN] - MODIFY GGPLOT PIPELINE HERE
# plot the time-series
p1 <- ggplot(data = movie_counts_peryear, aes(x = year, y = counts)) +
  geom_line() +
  geom_point() +
  labs(x = "year",
       y = "number of movies",
       title = "Movie Counts Over Time")
### [END] - MODIFY GGPLOT PIPELINE HERE
p1
```


Movie Counts Over Time
number of movies
1000
750
500
250
0
1900
1925
1950
1975
2000
year

2. **Tracking emotions of movie genres over time.** Below is a series of R code blocks that uses the `movie_plots` data frame and creates a new variable that tokenizes the movie plots into unigrams and counts them by genre and by year. The resulting data frame is called `words_by_genre` and it has 4 variables, which are `Genre`, `year`, `word`, and `count`.

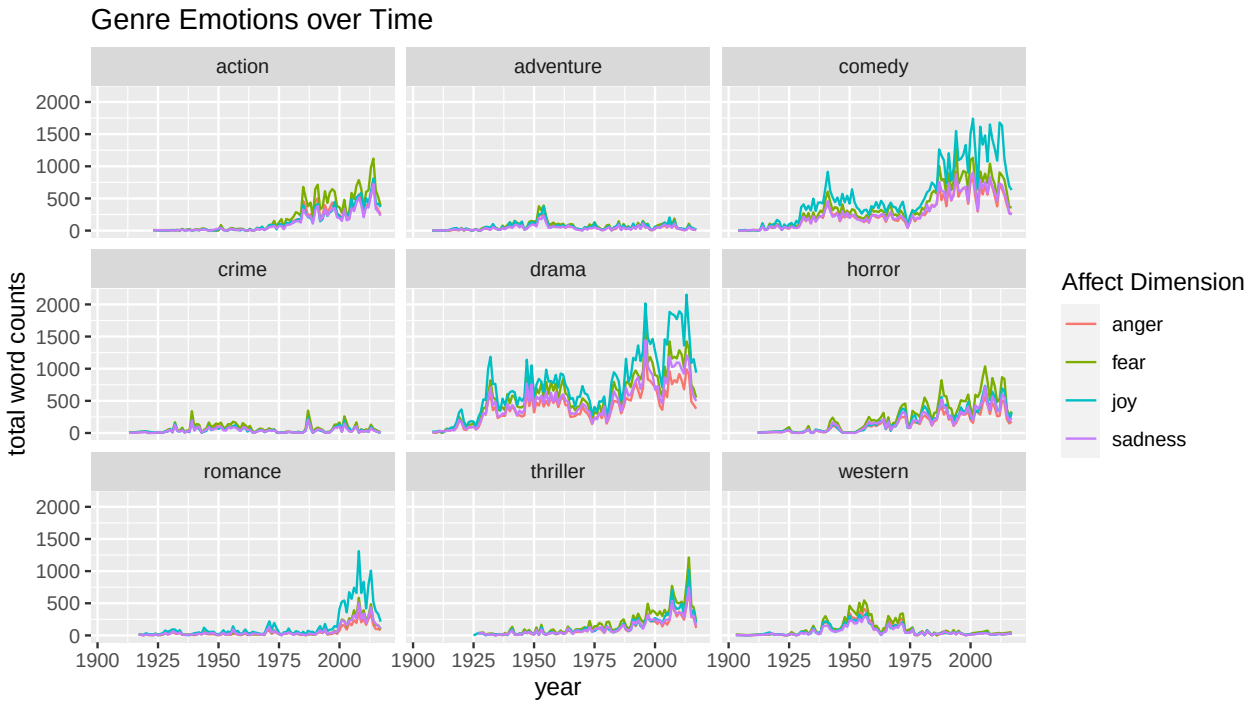
```
# word tokenize the movie plots and count the words in each decade
chosen_genres <- c("drama","comedy","horror",
                   "action","thriller","romance",
                   "western","crime","adventure")
words_by_genre <- movie_plots %>%
  # filter chosen genres
  filter(Genre %in% chosen_genres) %>%
  group_by(Genre,year) %>%
  # tokenize by unigrams (words)
  unnest_ngrams(word, Plot, n = 1) %>%
  group_by(Genre,year,word) %>%
  # count the words
  summarise(count = n(), .groups = "drop") %>%
  # filter out the stopwords
  filter(!word %in% swd_list)
```

Your task is to modify the code below. The modification should include the following.

a. Create a new column that divides the `year` variable into decades. Hint: Use the `mutate()` and `case_when()` function.
b. Create a new column that computes the proportions of the `sum_count` variable by decade - that is the frequency relative to the sum of the counts across the `AffectDimension` variable for each decade.
c. In your plot, use colorblind friendly colors, relabel labels, and adjust scaling as appropriate. Provide a paragraph describing your observations.

```
### [BEGIN] - WORK ON DATA WRANGLING HERE
words_by_genre_emo <- words_by_genre %>%
  left_join(emotions_4_with_intensity, by = c("word" = "term")) %>%
  drop_na() %>%
  group_by(Genre,AffectDimension,year) %>%
  summarise(sum_count = sum(count), .groups = "drop")
### [END] - WORK ON DATA WRANGLING HERE
```

```
### [BEGIN] - MODIFY GGPLOT PIPELINE HERE
p2 <- ggplot(data = words_by_genre_emo, aes(x =year,
                                            y = sum_count,
                                            color = AffectDimension)) +
  geom_line() +
  facet_wrap(~Genre) +
  labs(x = "year",
       y = "total word counts",
       color = "Affect Dimension",
       title = "Genre Emotions over Time")
### [END] - MODIFY GGPLOT PIPELINE HERE
p2
```


Genre Emotions over Time
action
adventure
comedy
crime
drama
horror
romance
thriller
western
total word counts
0
500
1000
1500
2000
1900
1925
1950
1975
2000
year
Affect Dimension
anger
fear
joy
sadness

3. **Popular Casts by Genre.** Modify the code below using the following tasks.
    a. Use the `Cast` variable to separate the actor names that was listed. Hint: You can use the `unnest_token()` function with parameters `token = 'regex'` and `pattern=",|\n"`.
    b. Remove leading and trailing white space of the resulting column of names. Hint: Use the `mutate()` and `trimws()` functions.
    c. Create a visualization that showcases the top 10 most frequent actors in each genre for a given sequence of four decades. Hint: Use `case_when()` and `filter()` functions. Provide a paragraph describing your obervations.

```
### [BEGIN] - MODIFY DATA WRANGLING HERE
cast_df <- movie_plots %>%
  # select chosen variables
  select(Genre, Cast, year) %>%
  # filter chosen genres
  filter(Genre %in% chosen_genres)
### [END] - MODIFY DATA WRANGLING HERE
```

```
### [BEGIN] - CREATE GGPLOT PIPLINE HERE

### [END] - CREATE GGPLOT PIPLINE HERE
```

References

Lewis, D. D., Yang, Y., Russell-Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*(Apr), 361–397.

Mohammad, S. M. (2018). Word affect intensities. *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*.