COMP2501 Assignment 2

## Requirements

**Submission deadline: Oct 31st, 2025 at 23:59 (HKT).**

**Full mark of assignment 2: 50.**

For the following questions, please:

1. Replace all [Input here] places with your information or your answer.
2. Complete the code block by adding your own code to fulfill the requirements in each question. Please use the existing code block and do not add your own code block. Noting that please use head() to show the corresponding results if there are too many rows in them.

Please make sure your Rmd file is a valid Markdown document and can be successfully knitted.

For assignment submission, please knit your final Rmd file into a Word document, and submit both your **Rmd** file and the knitted **Microsoft Word** document file to Moodle. You get 0 score if 1) the Rmd file you submitted cannot be knitted, and 2) you have not submitted a Word document. For each visualization question, please make sure that the generated plot is shown in-place with the question and after the code block.

## Name and UID

Name: Shen Hongshan

UID: 3036290936

### Environmental setup

You need to have the datasets, tidyr, dplyr, rvest, stringr, lubridate, gutenbergr, tidytext, textdata and ggplot2 packages installed. If not yet, please run install.packages(c("datasets", "tidyr", "dplyr", "rvest", "stringr", "lubridate", "gutenbergr", "tidytext", "textdata", "ggplot2")) in your R environment.

# Load the package.  
library(datasets)  
library(tidyr)  
library(readr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(rvest)

##   
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':  
##   
## guess\_encoding

library(stringr)  
library(lubridate)

##   
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':  
##   
## date, intersect, setdiff, union

library(gutenbergr)  
library(tidytext)  
library(textdata)  
library(ggplot2)

### 1. (3 points) Load the built-in co2 dataset.

data("co2")  
head(co2, 24)

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39 314.65 313.68 313.18  
## 1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01 315.74 314.00 313.68  
## Nov Dec  
## 1959 314.66 315.43  
## 1960 314.84 316.03

#### 1) (1 point) Transform it into a long data frame. Note that co2 is a time series, you can use time and cycle to extract year and month. The resulting data frame should contain three columns: "Year", "Month", and "co2". Show the first 5 rows.

co2\_long <- data.frame(  
 Year = floor(time(co2)),  
 Month = cycle(co2),  
 co2 = as.numeric(co2)  
)  
head(co2\_long,5)

## Year Month co2  
## 1 1959 1 315.42  
## 2 1959 2 316.31  
## 3 1959 3 316.50  
## 4 1959 4 317.56  
## 5 1959 5 318.13

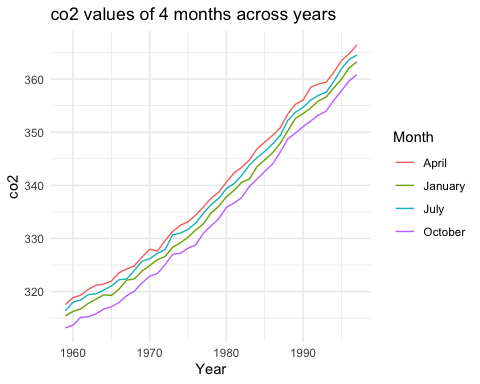
#### 2) (1 point) Reshape the dataset into a wide data frame using pivot functions, with one year as a row, and each month as a column, entries representing co2 levels. Show the co2 values of summer months (June, July, August) from 1970 to 1975.

co2\_long\_named <- co2\_long  
co2\_long\_named$Month <- month.name[co2\_long$Month]  
co2\_wide <- co2\_long\_named |>   
 pivot\_wider(names\_from = Month, values\_from = co2)  
co2\_wide |>   
 select(Year, June, July, August) |>  
 filter(Year >=1970 & Year <=1975)

## # A tibble: 6 × 4  
## Year June July August  
## <dbl> <dbl> <dbl> <dbl>  
## 1 1970 328. 326. 325.  
## 2 1971 328. 327. 325.  
## 3 1972 329. 328. 326.  
## 4 1973 332. 331. 329.  
## 5 1974 332. 331. 329.  
## 6 1975 333. 332. 330.

#### 3) (1 point) Use an appropriate graph to plot co2 values of January, April, July, and October across years. Which month has the highest co2 levels?

month4\_long <- co2\_long\_named |>  
 filter(Month %in% c("January", "April", "July", "October"))  
month4\_long |>  
 ggplot(aes(Year, co2, color = Month)) +  
 geom\_line() +  
 labs(title = "co2 values of 4 months across years") +  
 theme\_minimal()



month4\_long |>   
 group\_by(Month) |>   
 summarise(mean\_co2 = mean(co2)) |>   
 arrange(desc(mean\_co2)) |>  
 head(1) |>   
 pull(Month)

## [1] "April"

### 2. (2 points) Perform different types of joins between the following employees and payroll data frames using the "emp\_id" column. Use left\_join, right\_join, inner\_join, and full\_join to create four resulting data frames, name them appropriately. Display the data frame that contains the most rows.

employees <- data.frame(  
 emp\_id = c(101,102,103,104,105),  
 name = c("Ana","Ben","Chen","Divya","Evan"),  
 dept = c("Data","Data","HR","Finance","Ops")  
)  
payroll <- data.frame(  
 emp\_id = c(101,101,102,103,103,103,106),  
 month = c(1,2,1,1,2,3,1),  
 salary = c(95,100,88,76,80,82,70)  
)  
  
left\_join\_df <- left\_join(employees, payroll, by = "emp\_id")  
right\_join\_df <- right\_join(employees, payroll, by = "emp\_id")  
inner\_join\_df <- inner\_join(employees, payroll, by = "emp\_id")  
full\_join\_df <- full\_join(employees, payroll, by = "emp\_id")  
  
list\_df = list(left\_join\_df, right\_join\_df, inner\_join\_df, full\_join\_df)  
max\_row <- max(sapply(list\_df, nrow))  
for(df in list\_df){  
 if(nrow(df) == max\_row){  
 print(df)  
 }  
}

## emp\_id name dept month salary  
## 1 101 Ana Data 1 95  
## 2 101 Ana Data 2 100  
## 3 102 Ben Data 1 88  
## 4 103 Chen HR 1 76  
## 5 103 Chen HR 2 80  
## 6 103 Chen HR 3 82  
## 7 104 Divya Finance NA NA  
## 8 105 Evan Ops NA NA  
## 9 106 <NA> <NA> 1 70

### 3. (2 points) Find the union, intersection, and set difference of the following df1 and df2 data frames, and display all resulting data frames. Additionally, calculate the union, intersection, and set difference using only the "value" column from both data frames, and show these results as well.

df1 <- data.frame(id = c(1, 2, 3, 3, 5), value = c("a", "b", "c", "d", "e"))  
df2 <- data.frame(id = c(3, 4, 5), value = c("a", "f", "e"))  
  
union(df1,df2)

## id value  
## 1 1 a  
## 2 2 b  
## 3 3 c  
## 4 3 d  
## 5 5 e  
## 6 3 a  
## 7 4 f

intersect(df1,df2)

## id value  
## 1 5 e

setdiff(df1,df2)

## id value  
## 1 1 a  
## 2 2 b  
## 3 3 c  
## 4 3 d

setdiff(df2,df1)

## id value  
## 1 3 a  
## 2 4 f

union(df1$value,df2$value)

## [1] "a" "b" "c" "d" "e" "f"

intersect(df1$value,df2$value)

## [1] "a" "e"

setdiff(df1$value,df2$value)

## [1] "b" "c" "d"

setdiff(df1$value,df2$value)

## [1] "b" "c" "d"

### 4. (4 points) Use the rvest package to scrape the titles, ratings, and prices of books from <https://books.toscrape.com/> (the 20 entries on the first page). Store the scraped data in a data frame named books and display the top 5 entries ranked by price (highest to lowest). Refer to this tutorial to learn more about HTML scraping: <https://r4ds.hadley.nz/webscraping.html>.

url <- "https://books.toscrape.com/"  
html <- read\_html(url)  
  
titles <- html |>  
 html\_elements("article.product\_pod h3 a")|>  
 html\_attr("title")  
  
ratings <- html%>%  
 html\_elements("article.product\_pod p.star-rating") %>%  
 html\_attr("class")%>%  
 gsub("star-rating ", "", .)   
  
prices <- html %>%  
 html\_elements("article.product\_pod p.price\_color") %>%  
 html\_text() %>%  
 gsub("£", "", .) %>%  
 as.numeric()  
  
books <- data.frame(  
 Title = titles,  
 Rating\_stars = ratings,  
 Price = prices  
)  
  
book <- books |> arrange(desc(Price))  
head(books, 5)

## Title Rating\_stars Price  
## 1 A Light in the Attic Three 51.77  
## 2 Tipping the Velvet One 53.74  
## 3 Soumission One 50.10  
## 4 Sharp Objects Four 47.82  
## 5 Sapiens: A Brief History of Humankind Five 54.23

### 5. (3 points) Extract entries from the following texts using regular expression:

#### 1) (1 point) Extract all the phone numbers from the following text: “You can reach us through +1(123)456-7890, (+852) 1234 5678, +86-10-12345678.” Don’t remove any non-numeric characters (plus symbol, parentheses, spaces in the phone number strings, etc.).

text <- "You can reach us through +1(123)456-7890, (+852) 1234 5678, +86-10-12345678."  
pattern <- "\\(?\\+[0-9()\\s-]+\\)?"  
phone\_number <- str\_extract\_all(text, pattern)[[1]]  
phone\_number

## [1] "+1(123)456-7890" "(+852) 1234 5678" "+86-10-12345678"

#### 2) (1 point) Extract all the email addresses from the following text: “Contact us at [info@example.com](mailto:info@example.com), [support@example.com](mailto:support@example.com).” Note that email addresses may contain letters, numbers, dots, and underscores. Do not include the trailing period (.) in this text example.

text <- "Contact us at info@example.com or support@example.com."  
pattern <- "[a-z]+@example.com"  
email <- str\_extract\_all(text, pattern)[[1]]  
email

## [1] "info@example.com" "support@example.com"

#### 3) (1 point) Extract all valid IPv4 addresses from the following texts: “Servers: 192.168.0.1, 10.0.0.5, and 251.0.0.1”.

text <- "Servers: 192.168.0.1, 10.0.0.5, and 251.0.0.1"  
pattern <- "[0-9]+.[0-9]+.[0-9]+.[0-9]+"  
address <- str\_extract\_all(text, pattern)[[1]]  
address

## [1] "192.168.0.1" "10.0.0.5" "251.0.0.1"

### 6. (2 points) Use the lubridate package in R to parse the "date\_time" column in the date\_data. Create new columns for year, month, day, hour in Hong Kong time ("Asia/Hong\_Kong"). You might need to use mapply or sapply to vectorize the time parse step.

date\_data <- data.frame(date\_time = c("2025-09-22 07:30:15 America/New\_York",  
 "2025-09-23 12:15:30 America/Los\_Angeles",  
 "2025-09-24 23:56:50 Europe/London",  
 "2025-09-25 13:50:05 Asia/Shanghai"))  
  
date\_data <- date\_data %>%  
 mutate(  
 parsed\_datetime = as.POSIXct(date\_time, tz = "UTC") %>% with\_tz("Asia/Hong\_Kong"),  
 year = year(parsed\_datetime),  
 month = month(parsed\_datetime),  
 day = day(parsed\_datetime),  
 hour = hour(parsed\_datetime)  
 )  
  
date\_data

## date\_time parsed\_datetime year month day  
## 1 2025-09-22 07:30:15 America/New\_York 2025-09-22 15:30:15 2025 9 22  
## 2 2025-09-23 12:15:30 America/Los\_Angeles 2025-09-23 20:15:30 2025 9 23  
## 3 2025-09-24 23:56:50 Europe/London 2025-09-25 07:56:50 2025 9 25  
## 4 2025-09-25 13:50:05 Asia/Shanghai 2025-09-25 21:50:05 2025 9 25  
## hour  
## 1 15  
## 2 20  
## 3 7  
## 4 21

### 7. (7 points) Scrape data from Wikipedia with the rvest package, and answer the following questions.

#### a. (2 points) Choose a Wikipedia page of your interest (or just use today’s featured article). Scrape the first paragraph from the Introduction section of that page. Store the scraped text in a variable and print it.

url <- "https://en.wikipedia.org/wiki/Quantum\_entanglement"  
page <- read\_html(url)  
  
paragraphs <- page %>%  
 html\_elements("p") %>%  
 html\_text()   
  
first\_paragraph <-   
 paragraphs[grepl("[a-z]", paragraphs)] %>%   
 ##dismiss the paragraphs without words e.g. "\n\n"  
 .[1]  
first\_paragraph

## [1] "Quantum entanglement is the phenomenon where the quantum state of each particle in a group cannot be described independently of the state of the others, even when the particles are separated by a large distance. The topic of quantum entanglement is at the heart of the disparity between classical physics and quantum physics: entanglement is a primary feature of quantum mechanics not present in classical mechanics.[1]: 867 "

#### b. (2 points) If there is an image in the page you picked, try to get the image URL of the first image. Please output the full image URL, prepend “https:” if necessary. Download the image and display it using knitr::include\_graphics.

image\_url <- page |>  
 html\_node("img") |>  
 html\_attr("src")  
  
if (grepl("^//", image\_url)) {   
 ##prepend "https:" if relative path  
 image\_url <- paste0("https:", image\_url)  
} else if (grepl("^/", image\_url)) {  
 ## prepend base URL if absolute path  
 base\_url <- "https://en.wikipedia.org"  
 image\_url <- paste0(base\_url, image\_url)  
}  
  
print(paste("Image URL:", image\_url))

## [1] "Image URL: https://en.wikipedia.org/static/images/icons/wikipedia.png"

knitr::include\_graphics(image\_url)

![](data:text/plain;base64,UGxlYXNlIHNldCBhIHVzZXItYWdlbnQgYW5kIHJlc3BlY3Qgb3VyIHJvYm90IHBvbGljeSBodHRwczovL3cud2lraS80d0pTLiBTZWUgYWxzbyBUNDAwMTE5Lgo=)

#### c. (3 points) Visit the Wikipedia page of “Hong Kong” (<https://en.wikipedia.org/wiki/Hong_Kong>) and identify a table that lists monthly climate data (such as average temperatures, precipitation, etc.). Scrape the data and transform it into a data frame by. The first column’s header should be named as “Measurement”, entries should be “Record high °C (°F)”, etc. Following columns’ headers should be January, February, etc. (hints: you can use the first row of data as your column headers.)

url <- "https://en.wikipedia.org/wiki/Hong\_Kong"  
page <- read\_html(url)   
tables <- page |>  
 html\_nodes("table")  
for (table in tables){  
 table\_text <- table |> html\_text()  
 if (grepl("Climate data|Averag.\*temperature|Precipitation", table\_text, ignore.case = TRUE)){  
 climate\_table = table  
 break  
 }  
}  
table\_df <- climate\_table |> html\_table()  
new\_col\_name = c("Measurement", month.name[], "Year")  
table\_df <- table\_df[-c(1, nrow(table\_df)), ]  
colnames(table\_df) <- new\_col\_name  
  
table\_df

## # A tibble: 13 × 14  
## Measurement January February March April May June July August September  
## <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>   
## 1 Record high … 26.9(8… 28.3(82… 31.5… 33.4… 36.1… 35.6… 36.1… 36.6(… 35.9(96.…  
## 2 Mean maximum… 24.0(7… 25.1(77… 27.5… 30.2… 32.3… 33.6… 34.1… 34.2(… 33.4(92.…  
## 3 Mean daily m… 18.7(6… 19.4(66… 21.9… 25.6… 28.8… 30.7… 31.6… 31.3(… 30.5(86.…  
## 4 Daily mean °… 16.5(6… 17.1(62… 19.5… 23.0… 26.3… 28.3… 28.9… 28.7(… 27.9(82.…  
## 5 Mean daily m… 14.6(5… 15.3(59… 17.6… 21.1… 24.5… 26.5… 26.9… 26.7(… 26.1(79.…  
## 6 Mean minimum… 9.1(48… 10.2(50… 12.2… 16.3… 20.7… 23.6… 24.2… 24.3(… 23.5(74.…  
## 7 Record low °… 0.0(32… 2.4(36.… 4.8(… 9.9(… 15.4… 19.2… 21.7… 21.6(… 18.4(65.…  
## 8 Average rain… 33.2(1… 38.9(1.… 75.3… 153.… 290.… 491.… 385.… 453.2… 321.4(12…  
## 9 Average rain… 5.70 7.97 10.50 11.37 15.37 19.33 18.43 17.50 14.90   
## 10 Average rela… 74 79 82 83 83 82 81 81 78   
## 11 Average dew … 11.7(5… 13.2(55… 16.1… 19.7… 23.0… 24.9… 25.2… 25.1(… 23.6(74.…  
## 12 Mean monthly… 145.8 101.7 100.0 113.2 138.8 144.3 197.3 182.1 174.4   
## 13 Percentage p… 43 32 27 30 34 36 48 46 47   
## # ℹ 4 more variables: October <chr>, November <chr>, December <chr>, Year <chr>

### 8. (10 points) Explore the Twitter US Airline Sentiment dataset (<https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>). The main data file is provided as A2\_airline\_tweets.csv.

#### a. (2 points) Load the dataset. Count the number of positive, negative, and neutral tweets (using the "airline\_sentiment" column) for each airline. Then calculate an average\_score for each airline using tweets, where a positive tweet is scored as +1, a negative tweet as -1, and a neutral tweet as 0. Rank all airlines by their average scores.

A2 <- read\_csv("A2\_airline\_tweets.csv")

## Rows: 14640 Columns: 15  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (11): airline\_sentiment, negativereason, airline, airline\_sentiment\_gold...  
## dbl (4): tweet\_id, airline\_sentiment\_confidence, negativereason\_confidence,...  
##   
## 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.

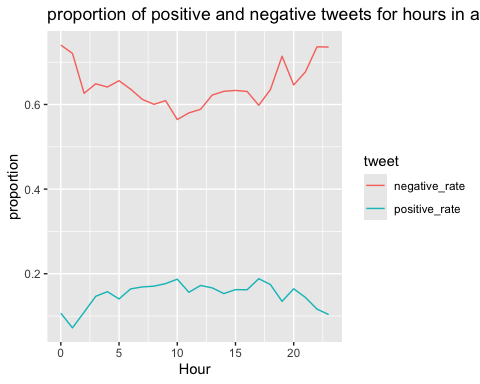
tweets <- A2 |> group\_by(airline) |> count(airline\_sentiment) |>  
 pivot\_wider(names\_from = airline\_sentiment, values\_from = n)  
  
rank\_A2 <- tweets |>   
 mutate(average\_score = (positive - negative)/(positive+negative+neutral))|>  
 arrange(desc(average\_score))   
  
rank\_A2

## # A tibble: 6 × 5  
## # Groups: airline [6]  
## airline negative neutral positive average\_score  
## <chr> <int> <int> <int> <dbl>  
## 1 Virgin America 181 171 152 -0.0575  
## 2 Delta 955 723 544 -0.185   
## 3 Southwest 1186 664 570 -0.255   
## 4 United 2633 697 492 -0.560   
## 5 American 1960 463 336 -0.589   
## 6 US Airways 2263 381 269 -0.685

#### b. (3 points) Extract the hour of day using the column tweet\_created (the timestamps were Pacific Standard Time, use "America/Los\_Angeles" for time zone), and create a line plot showing the proportion of positive and negative tweets across the hours of day (0–23). Do not exclude neutral tweets. When do you see highest proportions of negative tweets?

A2\_timed <- A2 |>  
 mutate(  
 #time zone  
 tweet\_time = as.POSIXct(tweet\_created, tz = "America/Los\_Angeles"),  
 hour\_of\_day = hour(tweet\_time)  
 ) |>   
 select(airline\_sentiment, hour\_of\_day)|>  
 group\_by(hour\_of\_day)|>  
 count(airline\_sentiment)  
  
A2\_timed\_wide <- A2\_timed |>  
 pivot\_wider(names\_from = airline\_sentiment, values\_from = n) |>  
 mutate(  
 positive\_rate = positive/(positive+negative+neutral),  
 negative\_rate = negative/(positive+negative+neutral)  
 )  
  
A2\_timed\_long <- A2\_timed\_wide |>  
 pivot\_longer(c(positive\_rate, negative\_rate),names\_to = "tweet", values\_to = "rate")  
  
A2\_timed\_long |> ggplot(aes(hour\_of\_day, rate, color = tweet)) +  
 geom\_line() +  
 labs(  
 title = "proportion of positive and negative tweets for hours in a day",  
 x = "Hour",  
 y = "proportion",   
 label = "Type of tweets"  
 )

## Ignoring unknown labels:  
## • label : "Type of tweets"



A2\_timed\_wide |>   
 filter(negative\_rate == max(A2\_timed\_wide$negative\_rate)) |>  
 select(hour\_of\_day, negative\_rate)

## # A tibble: 1 × 2  
## # Groups: hour\_of\_day [1]  
## hour\_of\_day negative\_rate  
## <int> <dbl>  
## 1 0 0.740

#### c. (2 points) Extract all words from negative tweets using the text column. Exclude stop words, pure numbers, and compute the top 20 most frequent words.

negative\_comment <- A2 |>  
 filter(airline\_sentiment =="negative")|>  
 select(text)  
negative\_words <- negative\_comment |>  
 unnest\_tokens(word, text) |>  
 anti\_join(stop\_words, by = "word") |>  
 filter(!str\_detect(word, "^[0-9]+$")) |>  
 count(word, sort = TRUE)  
head(negative\_words,20)

## # A tibble: 20 × 2  
## word n  
## <chr> <int>  
## 1 flight 2931  
## 2 united 2891  
## 3 usairways 2374  
## 4 americanair 2109  
## 5 southwestair 1214  
## 6 jetblue 1050  
## 7 cancelled 925  
## 8 service 748  
## 9 hours 648  
## 10 hold 614  
## 11 customer 609  
## 12 time 596  
## 13 plane 526  
## 14 delayed 508  
## 15 amp 503  
## 16 call 461  
## 17 t.co 452  
## 18 hour 448  
## 19 flightled 447  
## 20 http 437

#### d. (3 points) Use the nrc lexicon to assign an emotion category to each word. Calculate the total counts of different emotion categories and sort the categories by their occurrences. Do you find the results counter-intuitive? Inspect the top-occurring words and their related emotion categories, are there any words that you think should be excluded?

nrc <- get\_sentiments("nrc")  
words\_with\_emotions <- negative\_words |>  
 inner\_join(nrc, by = "word")  
category <- words\_with\_emotions |>  
 group\_by(sentiment) |>  
 summarise(occurrence = sum(n))|>  
 arrange(desc(occurrence))  
category

## # A tibble: 10 × 2  
## sentiment occurrence  
## <chr> <int>  
## 1 positive 8151  
## 2 trust 6481  
## 3 negative 6105  
## 4 sadness 3425  
## 5 anticipation 3321  
## 6 fear 2528  
## 7 anger 2392  
## 8 disgust 1897  
## 9 joy 1660  
## 10 surprise 1236

### 9. (17 points) Explore the advanced data wrangling with the gutenbergr package and its corresponding datasets, and answer the following questions.

#### a. (1 points) Load gutenberg\_metadata as books. Print the number of entries (rows) and variables (columns) of the metadata, as well as the names of all variables in books. Print the first 5 rows.

data("gutenberg\_metadata")  
books <- gutenberg\_metadata  
dim(books)

## [1] 79491 8

names(books) |> head(5)

## [1] "gutenberg\_id" "title" "author"   
## [4] "gutenberg\_author\_id" "language"

#### b. (2 points) Create a subset of books containing only English language books with known authors (excluding “NA”, Various” and “Anonymous”), named english\_books. Then, identify the top 5 authors with the most publications, show their names and number of publications. Store their names in a vector called top\_authors.

english\_books <- books |>  
 filter(language =="en",   
 !is.na(author),  
 author != "Various",  
 author != "Anonymous"  
 )  
top\_authors <- (english\_books |> count(author, sort = TRUE) |> head(5) )$author

#### c. (3 points) For each author in top\_authors, find the work in English with the shortest title. If there are ties, use the entry with a smaller “gutenberg\_id”. Create a subset of books containing these five works by the five authors. (Hint: use slice, slice\_min, or slice\_max.)

shortest\_title\_df <- english\_books|>  
 filter(author %in% top\_authors) |>  
 mutate(title\_length = str\_length(title)) |>  
 group\_by(author) |>  
 slice\_min(order\_by = title\_length, n = 1, with\_ties = TRUE) |>  
 slice\_min(order\_by = gutenberg\_id, n = 1)   
short\_title\_books <- shortest\_title\_df$title

#### d. (2 points) Download your favourite book out of the five entries above. Pick Hamlet if you don’t have a preference (wait, you don’t see Hamlet in the output? Emmm, try part c again). Use gutenberg\_download to access the texts of the book, named the resulting data frame as texts. Join texts with the metadata table to append metadata. Remove empty lines that contain no characters, and show line 150-200 of texts.

Hamlet\_id <- 1787  
texts <- gutenberg\_download(Hamlet\_id)

## Determining mirror for Project Gutenberg from  
## https://www.gutenberg.org/robot/harvest.  
## Using mirror http://aleph.gutenberg.org.

texts\_cleaned <- texts |>  
 left\_join(gutenberg\_metadata, by = "gutenberg\_id")|>  
 mutate(  
 text = str\_trim(text),  
 char\_count = str\_length(text)  
 ) |>  
 filter(  
 char\_count > 0  
 )  
  
slice(texts\_cleaned,150:200)

## # A tibble: 51 × 10  
## gutenberg\_id text title author gutenberg\_author\_id language  
## <int> <chr> <chr> <chr> <int> <fct>   
## 1 1787 "public domain etexts… Haml… Shake… 65 en   
## 2 1787 "whatever else you ca… Haml… Shake… 65 en   
## 3 1787 "ject Gutenberg Assoc… Haml… Shake… 65 en   
## 4 1787 "WRITE TO US! We can … Haml… Shake… 65 en   
## 5 1787 "Internet: hart@pobox… Haml… Shake… 65 en   
## 6 1787 "Mail: Prof. Michael… Haml… Shake… 65 en   
## 7 1787 "P.O. Box 2782" Haml… Shake… 65 en   
## 8 1787 "Champaign, IL 61825" Haml… Shake… 65 en   
## 9 1787 "This \"Small Print!\… Haml… Shake… 65 en   
## 10 1787 "Internet (72600.2026… Haml… Shake… 65 en   
## # ℹ 41 more rows  
## # ℹ 4 more variables: gutenberg\_bookshelf <chr>, rights <fct>, has\_text <lgl>,  
## # char\_count <int>

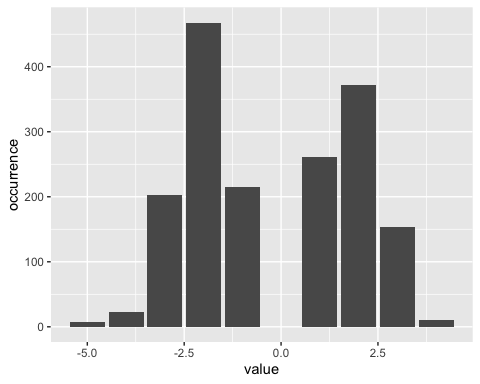
#### e. (2 points) Perform sentiment analysis on your book of interest. Use the sentiment lexicon afinn through get\_sentiments() using the textdata package, save it as a data frame named word\_sentiments. Tokenize texts of your book of interest into words, remove stop words and numbers, and then map the words to word\_sentiments using an appropriate join function. You only need to keep words that have corresponding sentiments in afinn. Count the occurrences of word and their corresponding sentiment value, sort the results in a descending order according to the occurrence count (n) column. What are the five most frequent words and what are their sentiment values?

word\_sentiments <- get\_sentiments("afinn")  
  
tokenized\_words <- texts |>  
 unnest\_tokens(word, text) |>  
 anti\_join(stop\_words, by = "word") |>  
 filter(!str\_detect(word, "^[0-9]+$"))   
  
words\_with\_sentiment <- tokenized\_words |>  
 inner\_join(word\_sentiments, by = "word") |>  
 count(word, value, sort = TRUE)  
  
head(words\_with\_sentiment, 5)

## # A tibble: 5 × 3  
## word value n  
## <chr> <dbl> <int>  
## 1 love 3 68  
## 2 heaven 2 46  
## 3 death -2 37  
## 4 ghost -1 34  
## 5 god 1 33

#### f. (2 points) Plot a bar plot showing the occurrence counts of different sentiment values. (Hint: use geom\_col)

words\_with\_sentiment |>   
 group\_by(value) |>  
 summarise(occurrence = sum(n))|>  
 ggplot(aes(value, occurrence)) +  
 geom\_col()



#### g. (3 points) Analyze the sentiment progression throughout your book of interest. Group lines in texts into 100-line segments, then calculate the average sentiment score for each segment using word\_sentiments. In this case, do not discard words that have no corresponding sentiment, assign a neutral score of 0. Plot the progression of sentiment across the text using an appropriate graph. You should use line number (instead of segment number) wherever applicable.

segments <- seq(1, nrow(texts\_cleaned), by = 100)  
  
line\_sentiments <- texts\_cleaned |>  
 unnest\_tokens(word, text)|>  
 left\_join(word\_sentiments, by = "word") |>  
 mutate(value = ifelse(is.na(value), 0, value)) |>  
 group\_by(value) |>  
 summarise(  
 line\_sentiment = mean(value, na.rm = TRUE),  
 word\_count = n()  
 ) |>  
 ungroup()

#### h. (2 points) Find the 5 most emotionally intense lines, defined as lines with the most amounts of emotional words (absolute of afinn score greater or equal than 2). For draws, pick lines that appear towards the end of the text. Show the lines, along with their line numbers and counts of emotional words.