COMP2501 Assignment 2

## Requirements

**Submission deadline: Mar 21th, 2023 at 23:59.**

**Full mark of assignment 2: 33.**

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: Li Hanwen

UID: 3035719519

### 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(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)  
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 airquality dataset and view its first 6 rows. 1) Reshape the dataset (named airquality\_long) using the pivot\_longer function to convert the variables Ozone, Solar.R, Wind, and Temp into a new column named Measurement, with corresponding values in a new column named Value. 2) Reshape the airquality\_long dataset (named airquality\_unite) using the unite function to combine the Month and Day columns (with - as a separator) into a new column named Date. Use head() to show the results of each sub-question. (hint: you may refer to this link for information: <https://www.statology.org/pivot_longer-in-r/>)

library(tidyr)  
data("airquality")  
airquality |> head()

## Ozone Solar.R Wind Temp Month Day  
## 1 41 190 7.4 67 5 1  
## 2 36 118 8.0 72 5 2  
## 3 12 149 12.6 74 5 3  
## 4 18 313 11.5 62 5 4  
## 5 NA NA 14.3 56 5 5  
## 6 28 NA 14.9 66 5 6

# 1)  
airquality\_long <- airquality |>   
 pivot\_longer(cols = -c(Month, Day), names\_to = "Measurement", values\_to = "Value")  
airquality\_long |> head()

## # A tibble: 6 × 4  
## Month Day Measurement Value  
## <int> <int> <chr> <dbl>  
## 1 5 1 Ozone 41   
## 2 5 1 Solar.R 190   
## 3 5 1 Wind 7.4  
## 4 5 1 Temp 67   
## 5 5 2 Ozone 36   
## 6 5 2 Solar.R 118

# 2)  
airquality\_unite <- airquality\_long |>  
 unite(col = "Date", c(Month, Day), sep = "-")  
airquality\_unite |> head()

## # A tibble: 6 × 3  
## Date Measurement Value  
## <chr> <chr> <dbl>  
## 1 5-1 Ozone 41   
## 2 5-1 Solar.R 190   
## 3 5-1 Wind 7.4  
## 4 5-1 Temp 67   
## 5 5-2 Ozone 36   
## 6 5-2 Solar.R 118

### 2. (3 points) Join the following customers and orders data frames by customer\_id, with different join function, including: left\_join, right\_join, inner\_join, full\_join, semi\_join, anti\_join (separately), and print the corresponding results (named left\_join\_df, right\_join\_df, inner\_join\_df, full\_join\_df, semi\_join\_df and anti\_join\_df respectively). (hint: <https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/join>, <https://dplyr.tidyverse.org/reference/mutate-joins.html>)

customers <- data.frame(  
 customer\_id = c(1, 2, 3, 4, 5),  
 customer\_name = c("Alice", "Bob", "Charlie", "Dave", "Eve"),  
 city = c("New York", "San Francisco", "Boston", "Seattle", "Chicago")  
)  
orders <- data.frame(  
 customer\_id = c(1, 1, 2, 2, 2, 3, 3, 4, 5),  
 order\_id = c(101, 102, 201, 202, 203, 301, 302, 401, 501),  
 order\_amount = c(100, 200, 150, 75, 225, 300, 225, 175, 250)  
)  
  
left\_join\_df <- left\_join(customers, orders, by = "customer\_id")

## Warning in left\_join(customers, orders, by = "customer\_id"): Each row in `x` is expected to match at most 1 row in `y`.  
## ℹ Row 1 of `x` matches multiple rows.  
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this  
## warning.

left\_join\_df

## customer\_id customer\_name city order\_id order\_amount  
## 1 1 Alice New York 101 100  
## 2 1 Alice New York 102 200  
## 3 2 Bob San Francisco 201 150  
## 4 2 Bob San Francisco 202 75  
## 5 2 Bob San Francisco 203 225  
## 6 3 Charlie Boston 301 300  
## 7 3 Charlie Boston 302 225  
## 8 4 Dave Seattle 401 175  
## 9 5 Eve Chicago 501 250

right\_join\_df <- right\_join(customers, orders, by = "customer\_id")

## Warning in right\_join(customers, orders, by = "customer\_id"): Each row in `x` is expected to match at most 1 row in `y`.  
## ℹ Row 1 of `x` matches multiple rows.  
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this  
## warning.

right\_join\_df

## customer\_id customer\_name city order\_id order\_amount  
## 1 1 Alice New York 101 100  
## 2 1 Alice New York 102 200  
## 3 2 Bob San Francisco 201 150  
## 4 2 Bob San Francisco 202 75  
## 5 2 Bob San Francisco 203 225  
## 6 3 Charlie Boston 301 300  
## 7 3 Charlie Boston 302 225  
## 8 4 Dave Seattle 401 175  
## 9 5 Eve Chicago 501 250

inner\_join\_df <- inner\_join(customers, orders, by = "customer\_id")

## Warning in inner\_join(customers, orders, by = "customer\_id"): Each row in `x` is expected to match at most 1 row in `y`.  
## ℹ Row 1 of `x` matches multiple rows.  
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this  
## warning.

inner\_join\_df

## customer\_id customer\_name city order\_id order\_amount  
## 1 1 Alice New York 101 100  
## 2 1 Alice New York 102 200  
## 3 2 Bob San Francisco 201 150  
## 4 2 Bob San Francisco 202 75  
## 5 2 Bob San Francisco 203 225  
## 6 3 Charlie Boston 301 300  
## 7 3 Charlie Boston 302 225  
## 8 4 Dave Seattle 401 175  
## 9 5 Eve Chicago 501 250

full\_join\_df <- full\_join(customers, orders, by = "customer\_id")

## Warning in full\_join(customers, orders, by = "customer\_id"): Each row in `x` is expected to match at most 1 row in `y`.  
## ℹ Row 1 of `x` matches multiple rows.  
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this  
## warning.

full\_join\_df

## customer\_id customer\_name city order\_id order\_amount  
## 1 1 Alice New York 101 100  
## 2 1 Alice New York 102 200  
## 3 2 Bob San Francisco 201 150  
## 4 2 Bob San Francisco 202 75  
## 5 2 Bob San Francisco 203 225  
## 6 3 Charlie Boston 301 300  
## 7 3 Charlie Boston 302 225  
## 8 4 Dave Seattle 401 175  
## 9 5 Eve Chicago 501 250

semi\_join\_df <- semi\_join(customers, orders, by = "customer\_id")  
semi\_join\_df

## customer\_id customer\_name city  
## 1 1 Alice New York  
## 2 2 Bob San Francisco  
## 3 3 Charlie Boston  
## 4 4 Dave Seattle  
## 5 5 Eve Chicago

anti\_join\_df <- anti\_join(customers, orders, by = "customer\_id")  
anti\_join\_df

## [1] customer\_id customer\_name city   
## <0 rows> (or 0-length row.names)

### 3. (2 points) Find the union, intersection and difference of the following df1 and df2 data frames, and print the corresponding results (named union\_df, intersect\_df, setdiff\_df\_1\_2 and setdiff\_df\_2\_1 respectively).

df1 <- data.frame(id = c(1, 2, 3), value = c("a", "b", "c"))  
df2 <- data.frame(id = c(3, 4, 5), value = c("c", "d", "e"))  
  
union\_df <- dplyr::union(df1, df2)  
union\_df |> head()

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

intersect\_df <- dplyr::intersect(df1, df2)  
intersect\_df |> head()

## id value  
## 1 3 c

setdiff\_df\_1\_2 <- dplyr::setdiff(df1, df2)  
setdiff\_df\_1\_2 |> head()

## id value  
## 1 1 a  
## 2 2 b

setdiff\_df\_2\_1 <- dplyr::setdiff(df2, df1)  
setdiff\_df\_2\_1 |> head()

## id value  
## 1 4 d  
## 2 5 e

### 4. (3 points) Scrape the 1) movie titles, 2) their ratings, and 3) release years from the IMDb Top Rated Movies webpage (<https://www.imdb.com/chart/top/>) with the rvest package. Store the data in a data frame (named movies) and print the top 10 observations in movies. (hint: <https://jtr13.github.io/cc19/web-scraping-using-rvest.html>)

library(rvest)  
url <- "https://www.imdb.com/chart/top/"  
movies <- read\_html(url) |>  
 html\_node("table") |>  
 html\_table()  
movies <- movies[, 2:3] |>  
 setNames(c("Rank and Title", "Ratings"))  
movies <- movies |>  
 separate("Rank and Title", c("Rank", "Title", "Year"), "\n") |>  
 mutate\_if(is.character, str\_trim)  
movies <- movies[, 2:4]  
movies$Year <- substr(movies$Year, 2, 5)  
movies <- movies[, c(1, 3, 2)]  
movies |> head(10)

## # A tibble: 10 × 3  
## Title Ratings Year   
## <chr> <dbl> <chr>  
## 1 The Shawshank Redemption 9.2 1994   
## 2 The Godfather 9.2 1972   
## 3 The Dark Knight 9 2008   
## 4 The Godfather Part II 9 1974   
## 5 12 Angry Men 9 1957   
## 6 Schindler's List 8.9 1993   
## 7 The Lord of the Rings: The Return of the King 8.9 2003   
## 8 Pulp Fiction 8.8 1994   
## 9 The Lord of the Rings: The Fellowship of the Ring 8.8 2001   
## 10 Il buono, il brutto, il cattivo 8.8 1966

### 5. (3 points) Using the stringr package in R, perform the following tasks: 1) Extract all the phone numbers from the following text: “Please call us at 123-456-7890 or 555-555-5555.” 2) Extract all the email addresses from the following text: “Contact us at [info@example.com](mailto:info@example.com) or [support@example.com](mailto:support@example.com).” 3) Replace all the URLs (<https://www.xxx.com>) in the following text with the string “URL”: “Check out our website at <https://www.example.com> and our blog at <https://blog.example.com>.”. Print the corresponding results.

library(stringr)  
  
# 1)  
text1 <- "Please call us at 123-456-7890 or 555-555-5555."  
# p1 <- "[0-9-]+"  
p1 <- "\\d{3}-\\d{3}-\\d{4}"  
text1 |> str\_extract\_all(p1)

## [[1]]  
## [1] "123-456-7890" "555-555-5555"

# 2)  
text2 <- "Contact us at info@example.com or support@example.com."  
p2 <- "[a-z0-9]+@([a-z]+\\.)\*[a-z]{2,3}"  
text2 |> str\_extract\_all(p2)

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

# 3)  
text3 <- "Check out our website at https://www.example.com and our blog at https://blog.example.com."  
p3 <- "https://www\\.[\\w-]+\\.com"  
text3 |> str\_replace\_all(p3, "URL")

## [1] "Check out our website at URL and our blog at https://blog.example.com."

### 6. (2 points) Using the lubridate package in R, parse the date\_time column in the date\_data and create new columns for standard date and time components, and print the final results.

library(lubridate)  
library(hms)

##   
## Attaching package: 'hms'

## The following object is masked from 'package:lubridate':  
##   
## hms

date\_data <- data.frame(date\_time = c("2023-02-22 7:30:15", "2023-02-23 12:15:30", "2023-02-24 23:59:59"))  
date\_data$date\_time <- ymd\_hms(date\_data$date\_time)  
date\_data <- date\_data |>   
 mutate(date = as\_date(date\_data$date\_time)) |>   
 mutate(time = as\_hms(date\_data$date\_time))  
date\_data

## date\_time date time  
## 1 2023-02-22 07:30:15 2023-02-22 07:30:15  
## 2 2023-02-23 12:15:30 2023-02-23 12:15:30  
## 3 2023-02-24 23:59:59 2023-02-24 23:59:59

### 7. (17 points) Explore the advanced data wrangling with the gutenbergr package and its corresponding datasets, and answer the following questions.

#### a. (1 points) Install the gutenbergr package and load the gutenberg\_metadata as books. Print the first 6 rows, the number of observations (rows) and variables (columns), and the names of all variables in books.

library(gutenbergr)  
books <- gutenberg\_metadata  
head(books)

## # A tibble: 6 × 8  
## gutenberg\_id title author guten…¹ langu…² guten…³ rights has\_t…⁴  
## <int> <chr> <chr> <int> <chr> <chr> <chr> <lgl>   
## 1 1 "The Declaration o… Jeffe… 1638 en Politi… Publi… TRUE   
## 2 2 "The United States… Unite… 1 en Politi… Publi… TRUE   
## 3 3 "John F. Kennedy's… Kenne… 1666 en <NA> Publi… TRUE   
## 4 4 "Lincoln's Gettysb… Linco… 3 en US Civ… Publi… TRUE   
## 5 5 "The United States… Unite… 1 en United… Publi… TRUE   
## 6 6 "Give Me Liberty o… Henry… 4 en Americ… Publi… TRUE   
## # … with abbreviated variable names ¹​gutenberg\_author\_id, ²​language,  
## # ³​gutenberg\_bookshelf, ⁴​has\_text

dim(books)

## [1] 69199 8

names(books)

## [1] "gutenberg\_id" "title" "author"   
## [4] "gutenberg\_author\_id" "language" "gutenberg\_bookshelf"  
## [7] "rights" "has\_text"

#### b. (2 points) Remove any rows in books that have missing values in the author column, and then count the number of books for each author in a descending order. Who has the most publications and what’s the exact numer (ignoring Various and Anonymous as an author name)?

books <- books |> drop\_na(author)  
# books[complete.cases(books$author), ]  
# books[!is.na(books$author), ]  
books\_summary <- books |>   
 filter(!(author %in% c("Various", "Anonymous"))) |>   
 group\_by(author) |>   
 summarize(count = n()) |>  
 arrange(desc(count))  
books\_summary |> head()

## # A tibble: 6 × 2  
## author count  
## <chr> <int>  
## 1 Shakespeare, William 326  
## 2 Twain, Mark 235  
## 3 Lytton, Edward Bulwer Lytton, Baron 223  
## 4 Ebers, Georg 175  
## 5 Dickens, Charles 172  
## 6 Verne, Jules 169

books\_summary |> top\_n(1)

## Selecting by count

## # A tibble: 1 × 2  
## author count  
## <chr> <int>  
## 1 Shakespeare, William 326

#### c. (2 points) Create a subset of books with only Shakespeare, William’s English publications, named shakespeare\_books. Print the first 6 rows in shakespeare\_books.

shakespeare\_books <- books |> filter(author == "Shakespeare, William" & language == "en")  
shakespeare\_books |> head()

## # A tibble: 6 × 8  
## gutenberg\_id title author guten…¹ langu…² guten…³ rights has\_t…⁴  
## <int> <chr> <chr> <int> <chr> <chr> <chr> <lgl>   
## 1 100 The Complete Works… Shake… 65 en Plays Publi… TRUE   
## 2 1041 Shakespeare's Sonn… Shake… 65 en <NA> Publi… TRUE   
## 3 1045 Venus and Adonis Shake… 65 en <NA> Publi… TRUE   
## 4 1100 The First Part of … Shake… 65 en <NA> Copyr… TRUE   
## 5 1101 The Second Part of… Shake… 65 en <NA> Copyr… TRUE   
## 6 1102 The Third Part of … Shake… 65 en <NA> Copyr… TRUE   
## # … with abbreviated variable names ¹​gutenberg\_author\_id, ²​language,  
## # ³​gutenberg\_bookshelf, ⁴​has\_text

#### d. (4 points) Filter the dataset shakespeare\_books to only include specifically the book Hamlet as shakespeare\_hamlet, and extract only gutenberg\_id, title and author columns to save, and if there are more that one observation in shakespeare\_hamlet, just preserve the first observation with slice(). Then use gutenberg\_download() to download the corresponding texts according to shakespeare\_hamlet$gutenberg\_id as hamlet\_text. Lastly join shakespeare\_hamlet and hamlet\_text with left\_join() as hamlet\_data, and remove any missing values in the text column as well as convert the text column to lowercase.

empty\_is\_na <- TRUE  
shakespeare\_hamlet <- shakespeare\_books |>   
 filter(title == "Hamlet") |>  
 select(gutenberg\_id, title, author) |>  
 slice\_head(n = 1)  
hamlet\_text <- shakespeare\_hamlet$gutenberg\_id |> gutenberg\_download()

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

## Using mirror http://aleph.gutenberg.org

hamlet\_data <- left\_join(shakespeare\_hamlet, hamlet\_text)

## Joining with `by = join\_by(gutenberg\_id)`

## Warning in left\_join(shakespeare\_hamlet, hamlet\_text): Each row in `x` is expected to match at most 1 row in `y`.  
## ℹ Row 1 of `x` matches multiple rows.  
## ℹ If multiple matches are expected, set `multiple = "all"` to silence this  
## warning.

if(empty\_is\_na){  
 hamlet\_data$text <- str\_trim(hamlet\_data$text)  
 hamlet\_data <- hamlet\_data |> filter(hamlet\_data$text != "")  
}  
hamlet\_data$text <- str\_to\_lower(hamlet\_data$text)  
hamlet\_data

## # A tibble: 4,762 × 4  
## gutenberg\_id title author text   
## <int> <chr> <chr> <chr>   
## 1 1787 Hamlet Shakespeare, William \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*…  
## 2 1787 Hamlet Shakespeare, William this ebook was one of project guten…  
## 3 1787 Hamlet Shakespeare, William time when proofing methods and tool…  
## 4 1787 Hamlet Shakespeare, William is an improved edition of this titl…  
## 5 1787 Hamlet Shakespeare, William (#100) at https://www.gutenberg.org…  
## 6 1787 Hamlet Shakespeare, William \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*…  
## 7 1787 Hamlet Shakespeare, William this etext file is presented by pro…  
## 8 1787 Hamlet Shakespeare, William cooperation with world library, inc…  
## 9 1787 Hamlet Shakespeare, William future and shakespeare cdroms. pro…  
## 10 1787 Hamlet Shakespeare, William etexts that are not placed in the p…  
## # … with 4,752 more rows

#### e. (4 points) Perform sentiment analysis on hamlet\_data using the tidytext package. First get the sentiment lexicon afinn through get\_sentiments() using the textdata package and store it in hamlet\_sentiments. Then extract each token in text column of hamlet\_data with unnest\_tokens() and remove the stop words with anti\_join(), and then join it with hamlet\_sentiments by inner\_join, and count the number of word and its sentiment value in a descending order, saved as hamlet\_words. (hint: <http://rafalab.dfci.harvard.edu/dsbook/text-mining.html#sentiment-analysis>)

library(tidytext)  
library(textdata)  
  
hamlet\_sentiments <- get\_sentiments("afinn")  
hamlet\_words <- hamlet\_data |>   
 unnest\_tokens(word, text) |>  
 anti\_join(stop\_words, by = "word") |>  
 inner\_join(hamlet\_sentiments, by = "word") |>  
 group\_by(word, value) |>  
 summarize(n = n()) |>  
 arrange(desc(value))

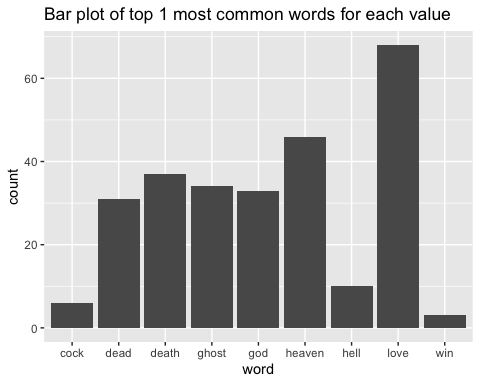
## `summarise()` has grouped output by 'word'. You can override using the  
## `.groups` argument.

hamlet\_words <- hamlet\_words[, c(1, 3, 2)]  
hamlet\_words

## # A tibble: 469 × 3  
## # Groups: word [469]  
## word n value  
## <chr> <int> <dbl>  
## 1 fantastic 1 4  
## 2 heavenly 2 4  
## 3 triumph 1 4  
## 4 win 3 4  
## 5 winner 1 4  
## 6 wonderful 2 4  
## 7 affection 4 3  
## 8 beauties 1 3  
## 9 blessing 5 3  
## 10 celebrated 1 3  
## # … with 459 more rows

#### f. (4 points) Folloing question e, please do operations on a dataset copy of hamlet\_words as hamlet\_top\_words to obtain the results with group\_by(value) and top\_n(), and reorder the results in a descending order of n, then create a bar plot with geom\_col() of the top 1 most common positive and negative words in hamlet\_words. Set an appropriate plot title and axis titles.

library(ggplot2)  
  
hamlet\_top\_words <-  
 hamlet\_words |>  
 group\_by(value) |>  
 # filter(row\_number() == 1) |>  
 top\_n(n = 1, wt = n) |>  
 arrange(desc(n))  
hamlet\_top\_words |>  
 ggplot(aes(word, n)) +  
 geom\_col() +  
 xlab("word") +  
 ylab("count") +  
 ggtitle("Bar plot of top 1 most common words for each value")



# hamlet\_top\_words\_pos <- hamlet\_top\_words |> filter(value > 0) |> arrange(desc(n)) |> head(1)  
# hamlet\_top\_words\_neg <- hamlet\_top\_words |> filter(value < 0) |> arrange(desc(n)) |> head(1)  
# bind\_rows(hamlet\_top\_words\_pos, hamlet\_top\_words\_neg) |>   
# ggplot(aes(word, n)) +  
# geom\_col() +  
# xlab("word") +  
# ggtitle("Bar plot of top 1 most common positive and negative words")