COMP2501 Assignment 2

Sibo Ding

Spring 2023

Requirements

Submission deadline: Mar 28th, 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: Sibo Ding

UID: 3035637204

Environmental setup

You need to have the `datasets`, `dplyr`, `tidyr`, `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(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
```

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)

head(airquality) # First 6 rows

##   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(Ozone, Solar.R, Wind, Temp),
  names_to = "Measurement", values_to = "Value")
head(airquality_long)

## # 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 = "-")
head(airquality_unite)
```

```
## # 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)
)

# If there are multiple matches between Left and Right, all
combinations of the matches are returned.
left_join_df <- left_join(customers, orders, by = "customer_id")
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")
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")
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")
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
```

```
# Return all rows from Left where there are matching values in Right,
keeping just columns from Left.
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
```

```
# Return all rows from Left where there are not matching values in
Right, keeping just columns from Left.
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
```

```
##   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
```

```
##   id value
## 1  3     c
```

```
setdiff_df_1_2 <- dplyr::setdiff(df1, df2)
setdiff_df_1_2
```

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

```
setdiff_df_2_1 <- dplyr::setdiff(df2, df1)
setdiff_df_2_1
```

```
##   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)

```r
library(rvest)
library(stringr)

url <- "https://www.imdb.com/chart/top/"
h <- read_html(url)
tab <- h |> html_nodes("table")
movies <- tab[[1]] |> html_table() # Get the first table

movies <- movies |> select("Rank & Title", "IMDb Rating") |>
  rename(title = "Rank & Title", rating = "IMDb Rating")
# Extract release year within ( )
movies <- movies |> mutate(release_year =
  title |> str_extract("\\d{4}\\)") |> str_replace("\\)", ""))
# Remove everything before "\n" and after "\n"
movies <- movies |> mutate(title = title |>
  str_replace(".*\\n\\s{6}", "") |> str_replace("\\n.*", ""))
head(movies, 10)
```

```
## # A tibble: 10 × 3
##    title                                             rating
release_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 or 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.

```r
library(stringr)

# \\d: digit, {}: numbers of occurrence
```

```
"Please call us at 123-456-7890 or 555-555-5555." |>
  str_extract_all("\\d{3}-\\d{3}-\\d{4}")

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

# \\w: word character; +: pne or more occurrences
"Contact us at info@example.com or support@example.com." |>
  str_extract_all("\\w+@\\w+.\\w+")

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

"Check out our website at https://www.example.com and our blog at
https://blog.example.com." |>
  str_replace_all("\\w+://\\w+.\\w+.\\w+", "URL")

## [1] "Check out our website at URL and our blog at URL."
```

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)

## Loading required package: timechange

##
## Attaching package: 'lubridate'

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

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_data |> mutate(date = date(date_time))
date_data <- date_data |> mutate(time = as_hms(ymd_hms(date_time)))
date_data

##              date_time       date     time
## 1  2023-02-22 7:30:15 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) # First 6 rows

## # 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 ¹gutenberg_author_id, ²language,
## #   ³gutenberg_bookshelf, ⁴has_text

nrow(books) # Number of observations

## [1] 69199

ncol(books) # Number of variables

## [1] 8

names(books) # Names of variables

## [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 |> filter(!is.na(author)) |> # Keep not N/A
  group_by(author) |> count() |>
  arrange(desc(n)) |> head()

## # A tibble: 6 × 2
## # Groups:   author [6]
##   author                                n
##   <chr>                             <int>
## 1 Various                            3798
## 2 Anonymous                           867
## 3 Shakespeare, William                326
## 4 Twain, Mark                         235
## 5 Lytton, Edward Bulwer Lytton, Baron   223
## 6 Ebers, Georg                        175

# Shakespeare, William has the most publications (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")
head(shakespeare_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          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 ¹gutenberg_author_id, ²language,
## #   ³gutenberg_bookshelf, ⁴has_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.

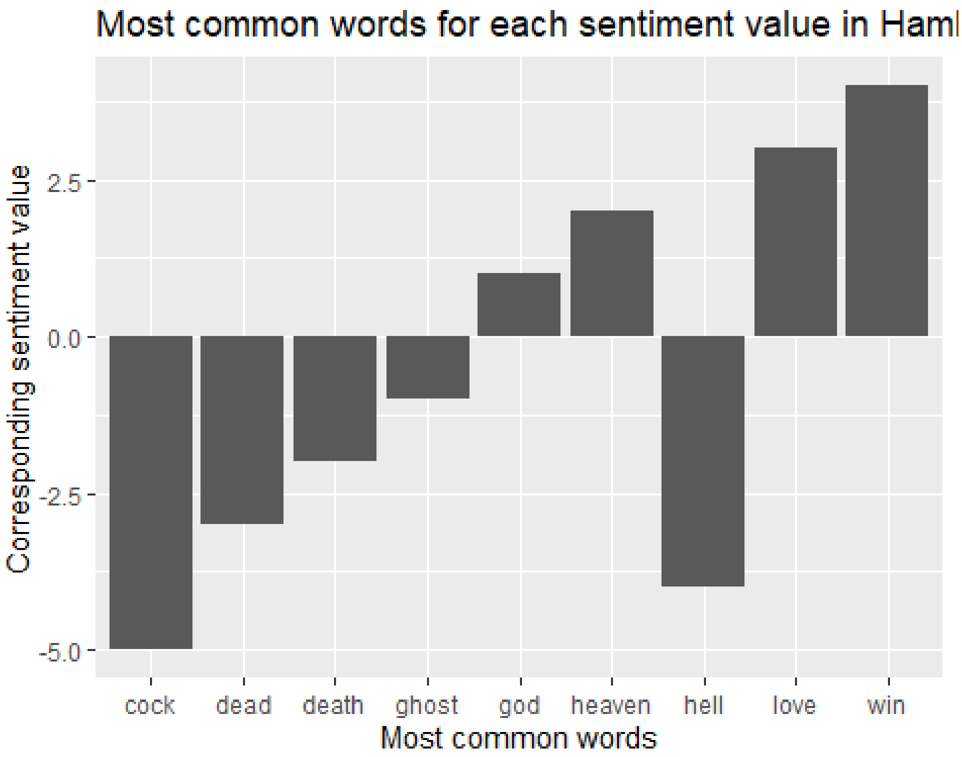
```
shakespeare_hamlet <- shakespeare_books |> filter(title == "Hamlet") |>
  select(gutenberg_id, title, author) |> slice(1)

# Then
hamlet_text <- gutenberg_download(shakespeare_hamlet$gutenberg_id)

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

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

# Lastly
hamlet_data <- left_join(shakespeare_hamlet, hamlet_text, by =
"gutenberg_id") |>
  filter(!is.na(text)) |>
  mutate(text = tolower(text))

head(hamlet_data)

## # A tibble: 6 × 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 gutenb…
## 3         1787 Hamlet Shakespeare, William time when proofing
methods and tools…
## 4         1787 Hamlet Shakespeare, William is an improved edition of
this title…
## 5         1787 Hamlet Shakespeare, William (#100) at
https://www.gutenberg.org/…
## 6         1787 Hamlet Shakespeare, William
"************************************…
```

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. Second, extract each token in text column of hamlet_data with unnest_tokens(). Third, remove the stop words with anti_join(). Fourth, join it with hamlet_sentiments by inner_join. Fifth, count the number of the combination of word and its sentiment value in a descending order by using count(your_data, word, value, sort=TRUE/FALSE), saved as hamlet_words. (hint: http://rafalab.dfci.harvard.edu/dsbook/text-mining.html#sentiment-analysis)

```
library(tidytext)
library(textdata)

hamlet_sentiments <- get_sentiments(lexicon = "afinn") # First

# unnest_tokens(output column name, input column name)
hamlet_words <- hamlet_data |> unnest_tokens(word, text) |> # Second
  anti_join(stop_words, by = "word") |> # Third
  inner_join(hamlet_sentiments, by = "word") |> # Fourth
  count(word, value, sort = TRUE) # Fifth

head(hamlet_words)

## # A tibble: 6 × 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
## 6 dead       -3    31
```

f. (4 points) Following question e, please do operations on a dataset copy of hamlet_words as hamlet_top_words to obtain the results of the top 1 most common word for each value group with group_by(value) and top_n(1, 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 word for each value group in hamlet_words. Set an appropriate plot title and axis titles.

```
library(ggplot2)

hamlet_top_words <- hamlet_words |>
  group_by(value) |> top_n(1, n) |>
  arrange(desc(n))

hamlet_top_words |> ggplot(aes(x = word, y = value)) +
  geom_col() +
  ggtitle("Most common words for each sentiment value in Hamlet") +
  xlab("Most common words") +
  ylab("Corresponding sentiment value")
```


Most common words for each sentiment value in Haml
Corresponding sentiment value
cock
dead
death
ghost
god
heaven
hell
love
win
Most common words