Mid-term 1920 Sem I with Solutions

Question 1

Question 1(a)

Suppose that we wish to generate random variables from the following joint pdf:

$$f_{X,Y}(x,y) = \frac{1}{2\pi\sqrt{1-\rho^2}} \times \exp\left(-\frac{1}{2(1-\rho^2)}\left[x^2+y^2-2\rho xy\right]\right) \tag{1}$$

ρ is a value between -1 and 1; it denotes the covariance between X and Y. The marginal distributions of X and Y are both $N(0,1)$ in this case.

Here is an algorithm that will return 20000 pairs of (X,Y) random variables from the above pdf:

1. Initialise your seed to be 2101, set $\rho = 0.76$ and set $X_0 = Y_0 = 0$.
2. For i from 1 to 25000:
    1. Generate X_i from $N(\rho Y_{i-1},\ 1-\rho^2)$.
    2. Generate Y_i from $N(\rho X_i,\ 1-\rho^2)$.
3. Remove the first 5000 pairs of (X,Y); this is the burn-in period.
4. Store the remaining XY pairs in a matrix, named `XY`, with 20000 rows and 2 columns.

```
nsims <- 25000
XY <- matrix(0, ncol=nsims, nrow=2)
rho <- 0.76
set.seed(2101)

XY[1, 1] <- rnorm(1, 0, 1 - rho^2)
XY[2, 1] <- rnorm(1, XY[1,1], 1 - rho^2)
for(i in 2:nsims) {
  XY[1, i] <- rnorm(1, XY[2, i-1]*rho, sqrt(1 - rho^2))
  XY[2, i] <- rnorm(1, XY[1, i]*rho, sqrt(1 - rho^2))
}
XY <- t(XY)
XY <- XY[-(1:5000), ]
```

Question 1(b)

The theoretical properties of the algorithm above can be found in textbooks. But how would you verify that your implementation is correct? Describe two checks that you made.

No R code required

Question 2

The file `ben_davis.zip` contains essays on the following topic:

Ben Davis should be allowed to defer his National Service.

The name of each file contains information on the gender and nationality of the author. The file `sent_scores.txt` contains sentiment scores (positive or negative emotions) for adjectives. The format of the file is:

```
word mean_sentiment_score sd_sentiment_score
```

You can ignore the standard deviation of the score. A positive value for a word means that that word is associated with a positive sentiment. Match the words in the files to the sentiment scores to obtain a summary sentiment score for each essay. Also compute the number of sentences in each file and the average number words per sentence in each file. Your final tibble, called `bd_df`, should have the following columns:

```
library(tidyverse)
library(stringr)
library(readxl)

fnames <- list.files("../data/ben_davis", pattern="*.txt$", full.names = TRUE)

ids <- str_extract(fnames, "[0-9]+")
gender <- str_extract(fnames, "[MF]+")
nationality <- if_else(str_detect(fnames, "non"), "non_local", "local")

derive_sent <- function(fname) {
  #all_lines <- readLines("../data/ben_davis/10_F_non_local.txt")
  all_lines <- readLines(fname)
  num_sent <- length(str_split(paste0(all_lines, collapse=" "),
                               boundary("sentence"))[[1]])
  all_words <- str_split(all_lines, boundary("word")) %>% unlist
  sent_len <- length(all_words)/num_sent

  word_counts <- table(all_words)
  word_counts <- as.data.frame(word_counts, stringsAsFactors = FALSE) %>%
    mutate(all_words = str_to_lower(all_words)) %>%
    group_by(all_words) %>% summarise(Freq = sum(Freq))

  sent_table <- read.delim("../data/sent_scores.txt",
                           sep = "\t", header=FALSE, stringsAsFactors = FALSE) %>%
    as_tibble() %>% rename(word = V1, sent_score = V2, sent_sd = V3)
  out <- left_join(word_counts, sent_table, by=c("all_words" = "word")) %>%
    filter(!is.na(sent_score)) %>%
    summarise(mm = sum(Freq * sent_score)) %>% pull
  return(c(out, num_sent, sent_len))
}
txt_features <- t(sapply(fnames, derive_sent, USE.NAMES = FALSE))
colnames(txt_features) <- c("sentiment", "num_sent", "sent_len")

bd_df <- tibble(ids, gender, nationality)
bd_df <- cbind(bd_df, as_tibble(txt_features)) %>% as_tibble()
bd_df
```

```
## # A tibble: 11 x 6
##    ids   gender nationality sentiment num_sent sent_len
##    <chr> <chr>  <chr>           <dbl>    <dbl>    <dbl>
##  1 01    M      non_local       5.41        11     22.4
##  2 02    F      local           1.05         9     26.3
##  3 03    M      local           0.570        7     18
##  4 04    M      local           2.78        19     12.2
##  5 05    F      non_local       8.68        11     24.8
##  6 06    M      local          -5.88        16     16.1
##  7 07    M      local           0.510        7     37.1
##  8 08    F      non_local      -0.66        13     18.9
##  9 09    F      non_local       1.59        12     20.7
```

```
## 10 10    F      non_local       1.85        13     19.4
## 11 11    F      non_local       9.55        19     23.3
```

Question 3

The table in `archae.html` contains information on counts of archaeological artefacts found at various distances to water. Read the table into R, and create a tidy version of this dataset. Store it as a tibble `q3_tidy`.

```
library(rvest)
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
page1 <- read_html("../data/archae.html")
table2 <- html_nodes(page1, "table")[[2]]
data_table <- html_table(table2) %>% as_tibble()
q3_tidy <- data_table %>% gather(2:7, key='distance', value='count')
```

If the types of artefacts were indeed independent of distance to water, we could use the following formula to compute the expected number of artefacts:

$$\text{expected number in cell } ij = \frac{(\text{row } i \text{ sum}) \times (\text{column } j \text{ sum})}{\text{grand total}} \tag{2}$$

Create a tibble `q3_tbl` with the same dimensions as the original, and the same ordering of rows and columns, but with expected cell counts instead of the actual counts.

```
total_count <- select(data_table, 2:7) %>% rowSums() %>% sum
q3_tbl <- group_by(q3_tidy, Artefact) %>%
  mutate(row_sum = sum(count)) %>%
  group_by(distance, add=FALSE) %>%
  mutate(col_sum = sum(count),
         exp_count = row_sum*col_sum/total_count) %>%
  select(1, 2, 6) %>%
  spread(key=distance, value=exp_count) %>%
  left_join(select(data_table, 1), ., by="Artefact") %>%
  select(1, 5, 7, 3, 2, 4, 6)
```

Question 4

The data in `online_retail.xlsx` contains information on transactions made with an UK-based online retailer. The company mainly sells unique all-occasion gifts. Here is more information about the columns in the dataset:

- *InvoiceNo*: Invoice number. A number uniquely assigned to each transaction. If this code starts with letter 'C', it indicates a cancellation. Each invoice consists of a set of items bought by a particular customer.
- *StockCode*: Product (item) code.
- *Description*: Product (item) name.
- *Quantity*: The quantities of each product (item) per transaction.
- *InvoiceDate*: Invoice Date and time.
- *UnitPrice*: Unit price in Sterling Pounds.
- *CustomerID*: Customer identifier.
- *Country*: The name of the country where each customer resides.

```
retail_org <- read_excel("../data/online_retail.xlsx")
```

Question 4(a)

In this sub-part, we shall perform some simple data cleaning. Some StockCodes consist only of non-integer characters. Remove these, along with missing values in the CustomerID and Description columns.

```
retail_org <- read_excel("../data/online_retail.xlsx")
retail_org <- filter(retail_org, !str_detect(StockCode, "^[A-Za-z ]+$"),
                     !is.na(CustomerID), !is.na(Description))
```

Question 4(b)

Let us focus on the cancellation orders now. Write a function, that takes in the InvoiceNo for a cancellation order, and returns the most recent purchases for the items in that invoice by that customer. For instance, suppose my function is named `match_cancelled`, and my dataset is named `retail_org`.

The UnitPrice and Country can be ignored for now.

```
match_cancelled("C536383", retail_org)
```

```
## # A tibble: 1 x 9
##   CustomerID StockCode Description InvoiceNo.x Quantity.x
##        <dbl> <chr>     <chr>       <chr>            <dbl>
## 1      15311 35004C    SET OF 3 C~ <NA>                NA
## # ... with 4 more variables: InvoiceDate.x <dttm>, InvoiceNo.y <chr>,
## #   Quantity.y <dbl>, InvoiceDate.y <dttm>
```

```
match_cancelled("C578355", retail_org)
```

```
## # A tibble: 4 x 9
##   CustomerID StockCode Description InvoiceNo.x Quantity.x
##        <dbl> <chr>     <chr>       <chr>            <dbl>
## 1      14397 20956     PORCELAIN ~ 577060              36
## 2      14397 22178     VICTORIAN ~ 577060              24
## 3      14397 22596     CHRISTMAS ~ 575684              60
## 4      14397 84946     ANTIQUE SI~ 577060              12
## # ... with 4 more variables: InvoiceDate.x <dttm>, InvoiceNo.y <chr>,
## #   Quantity.y <dbl>, InvoiceDate.y <dttm>
```

```
match_cancelled("C580764", retail_org)
```

```
## # A tibble: 1 x 9
##   CustomerID StockCode Description InvoiceNo.x Quantity.x
##        <dbl> <chr>     <chr>       <chr>            <dbl>
```

```
## 1      14562 22667     RECIPE BOX~ 575517              24
## # ... with 4 more variables: InvoiceDate.x <dttm>, InvoiceNo.y <chr>,
## #   Quantity.y <dbl>, InvoiceDate.y <dttm>
```

Make sure that your function works correctly on the following Invoice Numbers, at least: C536383, C578355, C580764.

Question 4(c)

Now, remove the cancelled Invoices. For each StockCode, compute the maximum and minimum price at which the item was sold over the study period. Store your output in a tibble named `q4_prices`.

```
retail_clean <- filter(retail_org, !str_detect(InvoiceNo, "^C"))

retail_clean %>% group_by(StockCode, Description) %>%
  summarise(max_price=max(UnitPrice), min_price=min(UnitPrice))
```

```
## # A tibble: 3,889 x 4
## # Groups:   StockCode [3,660]
##    StockCode Description                 max_price min_price
##    <chr>     <chr>                           <dbl>     <dbl>
##  1 10002     INFLATABLE POLITICAL GLOBE       0.85      0.85
##  2 10080     GROOVY CACTUS INFLATABLE         0.85      0.39
##  3 10120     DOGGY RUBBER                     0.21      0.21
##  4 10123C    HEARTS WRAPPING TAPE             0.65      0.65
##  5 10124A    SPOTS ON RED BOOKCOVER TAPE      0.42      0.42
##  6 10124G    ARMY CAMO BOOKCOVER TAPE         0.42      0.42
##  7 10125     MINI FUNKY DESIGN TAPES          0.85      0.42
##  8 10133     COLOURING PENCILS BROWN TUBE     0.85      0.42
##  9 10135     COLOURING PENCILS BROWN TUBE     2.46      0.25
## 10 11001     ASSTD DESIGN RACING CAR PEN      3.29      1.27
## # ... with 3,879 more rows
```

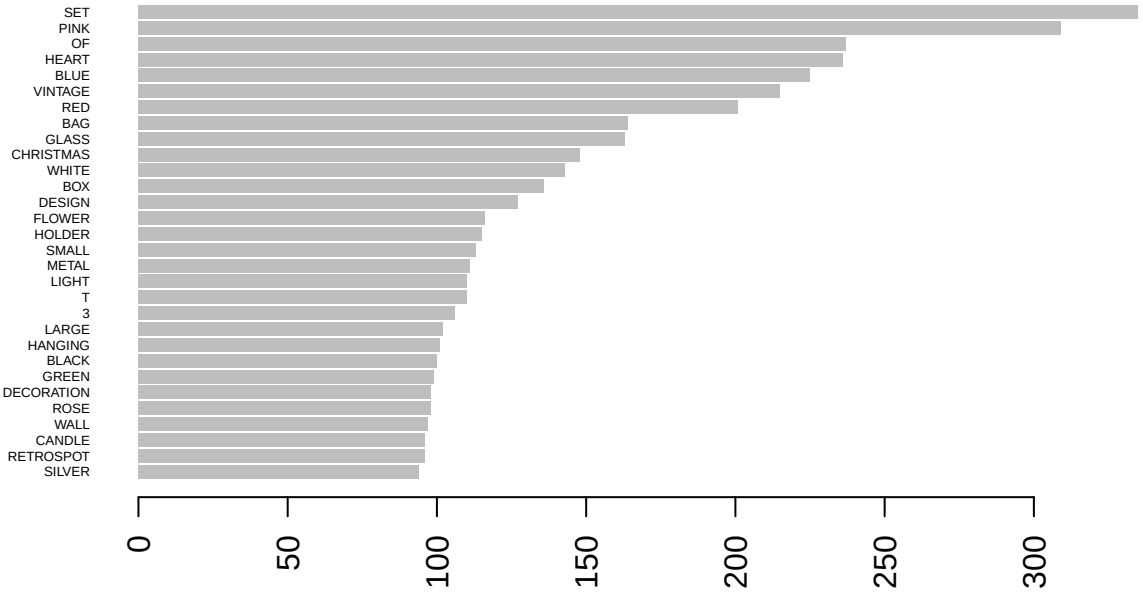
Question 4(d)

So it seems the retailer would modify the price over the period. Suggest one hypothesis that you could follow-up and test regarding the price changes.

Question 4(e)

From the unique Descriptions of the items sold by the retailer, extract the words and tabulate them. Display the counts of the top 30 most popular words in a bar-plot.

```
unique_descriptions <- unique(retail_clean$Description)
word_list <- unlist(str_split(unique_descriptions, boundary("word")))
word_count <- table(word_list)
top30_words <- sort(word_count, decreasing = TRUE)[1:30]
#barplot(top50_words)
barplot(rev(top30_words), cex.names = 0.4, las=2, horiz = TRUE, border=NA,
        main="Most Common Words in Descriptions")
```


Most Common Words in Descriptions
SET
PINK
OF
HEART
BLUE
VINTAGE
RED
BAG
GLASS
CHRISTMAS
WHITE
BOX
DESIGN
FLOWER
HOLDER
SMALL
METAL
LIGHT
T
3
LARGE
HANGING
BLACK
GREEN
DECORATION
ROSE
WALL
CANDLE
RETROSPOT
SILVER
0
50
100
150
200
250
300