Assignment 1

SA01 Team 2

Q1

Suppose that you have the following vector storing the sales value you have with your top 12 customers:

```
sales <- c(1736134, 10034,1003948,209445,98878.76,398454,777734.12,1039489.34,293894,9834,938
4754.65)
```

Now, you want to perfect your record in the following ways:

1.1

Convert the figures in sales vector into currency format and assign each element in the vector a name, which follows the format of customer[id], where id is a two-digit number following the sequence of elements in the vector. For example, 1736134 is named as customer01, 10034 is named as customer02, etc.

```
# currency format
sales_currency <- paste0('$',formatC(sales, digits=2, big.mark=',', format='f'))

# customer names
cust_name <- sprintf("customer%02d",1:11)
names(sales_currency) <- cust_name

# answer
sales_currency
```

```
##      customer01      customer02      customer03      customer04      customer05
## "$1,736,134.00"    "$10,034.00" "$1,003,948.00"   "$209,445.00"    "$98,878.76"
##      customer06      customer07      customer08      customer09      customer10
##   "$398,454.00"   "$777,734.12" "$1,039,489.34"   "$293,894.00"     "$9,834.00"
##      customer11
## "$9,384,754.65"
```

1.2

You want to classify your customers into three categories based on the sales value. Those with sales value above 1 million are VVIC, those with sales value in between 100,000 and 1 million are VIC, and the rest are IC. Create a vector named category to store the categories of your customers corresponding to sales vector.

```
# assigning categories
category <- c()
category[sales < 100000] <- 'IC'
category[sales > 100000 & sales < 1000000] <- 'VIC'
category[sales > 1000000] <- 'VVIC'

# convert to factor
category <- factor(category)
names(category) <- cust_name

# answer
category
```

```
## customer01 customer02 customer03 customer04 customer05 customer06 customer07
##       VVIC         IC       VVIC        VIC         IC        VIC        VIC
## customer08 customer09 customer10 customer11
##       VVIC        VIC         IC       VVIC
## Levels: IC VIC VVIC
```

Q2

Let n=5. Write R codes using n to generate the following matrix:

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    3    4    5    6
## [2,]    7    8    9   10   11
## [3,]   12   13   14   15   16
## [4,]   17   18   19   20   21
## [5,]   22   23   24   25    1
```

```
n <- 5

# generate vector
vec <- append(2:n^2, 1)
m <- matrix(vec, nrow=n, byrow=T)

# answer
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    3    4    5    6
## [2,]    7    8    9   10   11
## [3,]   12   13   14   15   16
## [4,]   17   18   19   20   21
## [5,]   22   23   24   25    1
```

Q3

Assign the following string to a single variable countries.

```
countries <- "Athens (Greece), Paris (France), St. Louis (United States), London (England), S
tockholm (Sweden), Berlin (Germany) [cancelled], Antwerp (Belgium), Paris (France), Amsterdam
(Netherlands), Los Angeles (United States), Berlin (Germany), Tokyo (Japan) [cancelled], Lond
on (England) [cancelled], London (England), Helsinki (Finland), Melbourne (Australia), Rome
 (Italy), Tokyo (Japan), Mexico City (Mexico), Munich (West Germany), Montreal (Canada), Mosc
ow (Soviet Union), Los Angeles (United Statesz), Seoul (South Korea), Barcelona (Spain), Atla
nta (United States), Sydney (Australia), Athens (Greece), Beijing (China), London (England),
 Rio de Janeiro (Brazil), Tokyo (Japan)"
```

The string contains Summer Olympic host cities and countries from 1896 to 2020 in chronological order. Unpack the string into a vector of strings, each containing both the host city (without its corresponding country) and the year of the Olympics. The resultant vector should look like:
“Athens Olympics 1896”, “Paris Olympics 1900”, “St. Louis Olympics 1904”, …..

3.1

The string contains the host countries in parentheses. Remove the countries information by removing all texts inside parentheses.

```
cities <- gsub(" \\([^)]*)", "", countries)

# answer
cities
```

```
## [1] "Athens, Paris, St. Louis, London, Stockholm, Berlin [cancelled], Antwerp, Paris, Amst
erdam, Los Angeles, Berlin, Tokyo [cancelled], London [cancelled], London, Helsinki, Melbourn
e, Rome, Tokyo, Mexico City, Munich, Montreal, Moscow, Los Angeles, Seoul, Barcelona, Atlant
a, Sydney, Athens, Beijing, London, Rio de Janeiro, Tokyo"
```

3.2

In 1916, 1940 and 1944, the Summer Olympics were cancelled due to the world wars. Remove all texts inside square brackets containing this information.

```
cities_names <- gsub(" \\[cancelled\\]", "", cities)

# answer
cities_names
```

```
## [1] "Athens, Paris, St. Louis, London, Stockholm, Berlin, Antwerp, Paris, Amsterdam, Los A
ngeles, Berlin, Tokyo, London, London, Helsinki, Melbourne, Rome, Tokyo, Mexico City, Munich,
Montreal, Moscow, Los Angeles, Seoul, Barcelona, Atlanta, Sydney, Athens, Beijing, London, Ri
o de Janeiro, Tokyo"
```

3.3

Split the current string into a vector of strings, each containing only one city.

```
cities_split <- unlist(strsplit(cities_names,', '))

# answer
cities_split
```

```
##  [1] "Athens"         "Paris"          "St. Louis"      "London"
##  [5] "Stockholm"      "Berlin"         "Antwerp"        "Paris"
##  [9] "Amsterdam"      "Los Angeles"    "Berlin"         "Tokyo"
## [13] "London"         "London"         "Helsinki"       "Melbourne"
## [17] "Rome"           "Tokyo"          "Mexico City"    "Munich"
## [21] "Montreal"       "Moscow"         "Los Angeles"    "Seoul"
## [25] "Barcelona"      "Atlanta"        "Sydney"         "Athens"
## [29] "Beijing"        "London"         "Rio de Janeiro" "Tokyo"
```

3.4

The Olympics is held every 4 years. Create a vector of years from 1896 to 2020 when the Olympic Games were held.

```
years <- seq(1896, 2020, by=4)

# answer
years
```

```
##  [1] 1896 1900 1904 1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952
## [16] 1956 1960 1964 1968 1972 1976 1980 1984 1988 1992 1996 2000 2004 2008 2012
## [31] 2016 2020
```

3.5

Combine the strings from part 3 and 4 to create the desired vector.

```
olympics <- paste(cities_split, "Olympics", years)

# final vector
olympics
```

```
##  [1] "Athens Olympics 1896"         "Paris Olympics 1900"
##  [3] "St. Louis Olympics 1904"      "London Olympics 1908"
##  [5] "Stockholm Olympics 1912"      "Berlin Olympics 1916"
##  [7] "Antwerp Olympics 1920"        "Paris Olympics 1924"
##  [9] "Amsterdam Olympics 1928"      "Los Angeles Olympics 1932"
## [11] "Berlin Olympics 1936"         "Tokyo Olympics 1940"
## [13] "London Olympics 1944"         "London Olympics 1948"
## [15] "Helsinki Olympics 1952"       "Melbourne Olympics 1956"
## [17] "Rome Olympics 1960"           "Tokyo Olympics 1964"
## [19] "Mexico City Olympics 1968"    "Munich Olympics 1972"
## [21] "Montreal Olympics 1976"       "Moscow Olympics 1980"
## [23] "Los Angeles Olympics 1984"    "Seoul Olympics 1988"
## [25] "Barcelona Olympics 1992"      "Atlanta Olympics 1996"
## [27] "Sydney Olympics 2000"         "Athens Olympics 2004"
## [29] "Beijing Olympics 2008"        "London Olympics 2012"
## [31] "Rio de Janeiro Olympics 2016" "Tokyo Olympics 2020"
```

```
# Format output to be the same as the question
final <- paste("\"", olympics,"\"",collapse = ", ", sep="")
cat(final)
```

```
## "Athens Olympics 1896", "Paris Olympics 1900", "St. Louis Olympics 1904", "London Olympics 1908", "Stockholm Olympics 1912", "Berlin Olympics 1916", "Antwerp Olympics 1920", "Paris Olympics 1924", "Amsterdam Olympics 1928", "Los Angeles Olympics 1932", "Berlin Olympics 1936", "Tokyo Olympics 1940", "London Olympics 1944", "London Olympics 1948", "Helsinki Olympics 1952", "Melbourne Olympics 1956", "Rome Olympics 1960", "Tokyo Olympics 1964", "Mexico City Olympics 1968", "Munich Olympics 1972", "Montreal Olympics 1976", "Moscow Olympics 1980", "Los Angeles Olympics 1984", "Seoul Olympics 1988", "Barcelona Olympics 1992", "Atlanta Olympics 1996", "Sydney Olympics 2000", "Athens Olympics 2004", "Beijing Olympics 2008", "London Olympics 2012", "Rio de Janeiro Olympics 2016", "Tokyo Olympics 2020"
```

Assignment 2

DBA3702 SA1 Group 2

8/26/2021

Q1

Load the data into your R environment. Note that the data is not in a neat table format. You need to extract the key information from the data and store it into a data frame.

```
univ <- read.csv("Univ Education.csv")
univ_sliced <- univ[4:36,] #slice to get the dataset in the middle
names(univ_sliced) <- univ_sliced[1,] #change the headers to be the first row
univ_sliced <- univ_sliced[-1,]
head(univ_sliced)
```

```
##                                  Variables   1993  1994  1995  1996  1997
## 5                                    Males  3,565 3,889 4,003 4,022 4,276
## 6                          Males: Education    na    na     12    46    45
## 7                        Males: Applied Arts    na    na    na    na    na
## 8       Males: Humanities & Social Sciences    481   512   497   497   554
## 9                   Males: Mass Communication    na    na    na    na    28
## 10                         Males: Accountancy    295   271   289   262   227
##     1998  1999  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009
## 5  4,455 4,573 4,536 4,735 4,858 5,197 5,246 4,949 5,207 5,823 5,736 6,004
## 6     56    44    35    63    63    97    76    86    89    73    53    67
## 7     na    na    na    na    na    11    11    11    11    28    31    47
## 8    547   576   574   534   591   456   383   351   412   441   478   547
## 9     27    19    34    39    30    33    34    36    33    28    32    36
## 10   264   297   250   257   226   222   234   211   176   260   295   346
##     2010  2011  2012  2013  2014  2015  2016  2017
## 5  6,496 6,428 6,778 7,724 7,756 7,872 7,703 8,963
## 6     67   103   116   167   124    99    79    65
## 7     84    84   108   138   165   187   173   175
## 8    703   708   737   818   803   743   803 1,078
## 9     30    41    37    62    44    42    35    78
## 10   380   332   447   399   473   480   439   574
```

Q2

Convert the data frame into a new data frame with 4 columns: “Year”, “Gender”, “Major”, and “Count”.

```
#Q2
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
```

```
#Remove the total males and total females
univ_sliced_edit <- univ_sliced[-c(1,17),]

#wide to long
univ_sliced_long <-univ_sliced_edit %>% gather(Year,Count,2:26)

df <- data.frame(Year=integer(),
                 Gender=character(),
                 Major=character(),
                 Count=integer())

#Get the Gender
univ_edit <- data.frame(univ_sliced_long)
univ_edit$Gender<-ifelse(grepl("Males",univ_edit$X.Variables.),"Male",ifelse(grepl("Females",
univ_edit$X.Variables.),"Female",NA))

#Get the Major
univ_edit$Major <- gsub("^.+:", "", univ_edit$X.Variables.) %>% trimws()

#Need remove the , in Count
univ_edit$Count <- gsub(",", "", univ_edit$Count)

#Change the count column into integer
univ_edit$Count <- gsub("na",NA,univ_edit$Count)
univ_edit$Count <- as.numeric(univ_edit$Count)

#Drop na
df <- univ_edit[complete.cases(univ_edit),c("Year","Gender","Major","Count")]

#Answer
head(df,20)
```

```
##    Year Gender                                    Major Count
## 3  1993   Male              Humanities & Social Sciences   481
## 5  1993   Male                               Accountancy   295
## 6  1993   Male                  Business & Administration   282
## 7  1993   Male                                       Law    92
## 8  1993   Male Natural, Physical & Mathematical Sciences   404
## 9  1993   Male                                  Medicine    95
## 10 1993   Male                                Dentistry    14
## 11 1993   Male                           Health Sciences    10
## 12 1993   Male                    Information Technology   264
## 13 1993   Male                   Architecture & Building   132
## 14 1993   Male                      Engineering Sciences  1496
## 18 1993 Female              Humanities & Social Sciences  1173
## 20 1993 Female                               Accountancy   396
## 21 1993 Female                  Business & Administration   708
## 22 1993 Female                                       Law    93
## 23 1993 Female Natural, Physical & Mathematical Sciences   588
## 24 1993 Female                                  Medicine    61
## 25 1993 Female                                Dentistry    11
## 26 1993 Female                           Health Sciences    40
## 27 1993 Female                    Information Technology   215
```

Q3

Compare the total number of graduates by gender. What is your conclusion?

```
#Female
df %>% filter(Gender == "Female") %>% summarise(sum(Count))
```

```
##   sum(Count)
## 1     146115
```

```
#Male
df %>% filter(Gender == "Male") %>% summarise(sum(Count))
```

```
##   sum(Count)
## 1     140783
```

Q3 Answer:

There are more female graduates than male graduates across all faculties from the year 1993 to 2017.

Female graduates: 146,115

Male graduates: 140,783

Q4

Which year has the largest number of dentistry graduates?

```
dentistry <- df[df$Major=="Dentistry",]
dentistry[order(-dentistry$Count),]$Year[1]
```

```
## [1] "2017"
```

```
#Alternative method
#df %>% filter(Major =="Dentistry") %>% group_by(Year) %>% summarise(Count = sum(Count)) %>%
 filter(Count==max(Count))
```

Q4 Answer:

Year 2017 has the largest number of Dentistry graduates.

Q5

We define a major as the most favorable major if it has the largest number of graduates. Please perform an analysis on the data and describe how the most favorable major for females changed over years.

```
df %>% filter(Gender =="Female") %>% group_by(Year) %>% filter(Count == max(Count))
```

```
## # A tibble: 25 x 4
## # Groups:   Year [25]
##    Year  Gender Major                        Count
##    <chr> <chr>  <chr>                        <dbl>
##  1 1993  Female Humanities & Social Sciences  1173
##  2 1994  Female Humanities & Social Sciences  1133
##  3 1995  Female Humanities & Social Sciences  1240
##  4 1996  Female Humanities & Social Sciences  1364
##  5 1997  Female Humanities & Social Sciences  1367
##  6 1998  Female Humanities & Social Sciences  1547
##  7 1999  Female Humanities & Social Sciences  1492
##  8 2000  Female Humanities & Social Sciences  1452
##  9 2001  Female Humanities & Social Sciences  1520
## 10 2002  Female Humanities & Social Sciences  1609
## # ... with 15 more rows
```

Q5 Answer:

In general, the most favorable major by females has been Humanities & Social Sciences.

Most favorable major by females over the years:

Year 1993 to 2003: Humanities & Social Sciences

Year 2004 to 2008: Engineering Sciences

Year 2009 to 2017: Humanities & Social Sciences

Assignment 3

DBA3702 SA1 Group 2

```
library(rvest)
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(XML)
```

```
## Warning: package 'XML' was built under R version 4.1.1
```

```
library(curl)
```

```
## Using libcurl 7.64.1 with Schannel
```

```
library(countrycode)
```

```
## Warning: package 'countrycode' was built under R version 4.1.1
```

Q1

Crawl the data about COVID-19 mortality from the following website: https://coronavirus.jhu.edu/data/mortality (https://coronavirus.jhu.edu/data/mortality)

```
theurl <- "https://coronavirus.jhu.edu/data/mortality"
url <- curl(theurl)
urldata <- readLines(url)
```

```
## Warning in readLines(url): incomplete final line found on 'https://
## coronavirus.jhu.edu/data/mortality'
```

```
table <- readHTMLTable(urldata, stringsasFactors = FALSE)
covid <- as.data.frame(table)
```

1. merge the data with the population data in “Countries.csv” file

```
countries <- read.csv('Countries.csv')
combined <- left_join(covid, countries, by=c("NULL.Country"="Country"))
head(combined)
```

```
##             NULL.Country NULL.Confirmed NULL.Deaths NULL.Case.Fatality
## 1                   Peru      2,154,132     198,447               9.2%
## 2                Hungary        813,040      30,061               3.7%
## 3 Bosnia and Herzegovina        216,124       9,862               4.6%
## 4        North Macedonia        179,587       6,045               3.4%
## 5                Czechia      1,680,354      30,406               1.8%
## 6             Montenegro        117,632       1,753               1.5%
##   NULL.Deaths.100K.pop. Population GDP_Per_Capita Life_Expectancy
## 1                610.41   32933835           7002            75.5
## 2                307.69    9655361          15924            75.9
## 3                298.76         NA             NA              NA
## 4                290.14         NA             NA              NA
## 5                284.97   10630589          22850            78.8
## 6                281.77     629355           8652            76.1
```

2. calculate the number confirmed cases per 100k and store the result as a new column

```
combined$NULL.Confirmed <- gsub("," ,"", combined$NULL.Confirmed)
combined$NULL.Confirmed <- as.integer(combined$NULL.Confirmed)

combined <- transform(combined, confirmed_per_100k = (NULL.Confirmed/Population)*100000)
head(combined)
```

```
##             NULL.Country NULL.Confirmed NULL.Deaths NULL.Case.Fatality
## 1                   Peru        2154132     198,447               9.2%
## 2                Hungary         813040      30,061               3.7%
## 3 Bosnia and Herzegovina         216124       9,862               4.6%
## 4        North Macedonia         179587       6,045               3.4%
## 5                Czechia        1680354      30,406               1.8%
## 6             Montenegro         117632       1,753               1.5%
##   NULL.Deaths.100K.pop. Population GDP_Per_Capita Life_Expectancy
## 1                610.41   32933835           7002            75.5
## 2                307.69    9655361          15924            75.9
## 3                298.76         NA             NA              NA
## 4                290.14         NA             NA              NA
## 5                284.97   10630589          22850            78.8
## 6                281.77     629355           8652            76.1
##   confirmed_per_100k
## 1           6540.787
## 2           8420.607
## 3                 NA
## 4                 NA
## 5          15806.782
## 6          18690.882
```

3. which continent has the highest average confirmed cases per 100k? (answer the question by coding)

Answer: Europe has the highest average confirmed cases per 100k.

```
countries_v <- c(combined[[1]])

# matching countries with their continents
combined$continent <- countrycode(sourcevar = countries_v, origin="country.name", destination
="continent")
```

```
## Warning in countrycode_convert(sourcevar = sourcevar, origin = origin, destination = dest,
: Some values were not matched unambiguously: Kosovo
```

```
# converting data type of vector
combined.clean <- na.omit(combined)
combined.clean$confirmed_per_100k <- as.numeric(combined.clean$confirmed_per_100k)

continent_highest <- combined.clean %>% group_by(continent) %>% summarise(Mean=mean(confirmed
_per_100k))
continent_highest %>% top_n(n=1)
```

```
## Selecting by Mean
```

```
## # A tibble: 1 x 2
##   continent  Mean
##   <chr>     <dbl>
## 1 Europe    8333.
```

Q2

Clean the country vaccination data that we went through in class as much as you can. Note that some cleaning has to be done by using for loop or while loop, you may not need to do it as I will only teach it next week.

```
vaccinations <- read.csv("country_vaccinations.csv")

# fill in missing iso codes for England, northern ireland, scotland, wales
vacc_na <- vaccinations[vaccinations$iso_code=="",]
vaccinations[vaccinations$iso_code=="",]$iso_code <- "GBR"

# separating vaccine types
vacc_types = unique(unlist(strsplit(vaccinations$vaccines, ", ")))
for (vacc in vacc_types)
  {
  vaccinations[vacc] = grepl(vacc, vaccinations$vaccines, fixed=T)
}

# resolving missing data from total_vaccinations
rows = which(is.na(vaccinations$total_vaccinations), arr.ind=T)
vaccinations[rows, "total_vaccinations"] = vaccinations[rows, "people_vaccinated"] +
vaccinations[rows, "people_fully_vaccinated"]

rows = which(is.na(vaccinations$people_vaccinated), arr.ind=T)
vaccinations[rows, "people_vaccinated"] = vaccinations[rows, "total_vaccinations"] -
vaccinations[rows, "people_fully_vaccinated"]

rows = which(is.na(vaccinations$people_fully_vaccinated), arr.ind=T)
vaccinations[rows, "people_fully_vaccinated"] = vaccinations[rows, "total_vaccinations"] -
vaccinations[rows, "people_vaccinated"]

head(vaccinations)
```

```
##   country iso_code       date total_vaccinations people_vaccinated
## 1 Albania      ALB 2021-01-10                  0                 0
## 2 Albania      ALB 2021-01-11                 NA                NA
## 3 Albania      ALB 2021-01-12                128               128
## 4 Albania      ALB 2021-01-13                188               188
## 5 Albania      ALB 2021-01-14                266               266
## 6 Albania      ALB 2021-01-15                308               308
##   people_fully_vaccinated daily_vaccinations_raw daily_vaccinations
## 1                       0                    NA                 NA
## 2                      NA                    NA                 64
## 3                       0                    NA                 64
## 4                       0                    60                 63
## 5                       0                    78                 66
## 6                       0                    42                 62
##   total_vaccinations_per_hundred people_vaccinated_per_hundred
## 1                           0.00                          0.00
## 2                             NA                            NA
## 3                           0.00                          0.00
## 4                           0.01                          0.01
## 5                           0.01                          0.01
## 6                           0.01                          0.01
##   people_fully_vaccinated_per_hundred daily_vaccinations_per_million
## 1                                  NA                              NA
## 2                                  NA                              22
## 3                                  NA                              22
## 4                                  NA                              22
## 5                                  NA                              23
## 6                                  NA                              22
##          vaccines        source_name
## 1 Pfizer/BioNTech Ministry of Health
## 2 Pfizer/BioNTech Ministry of Health
## 3 Pfizer/BioNTech Ministry of Health
## 4 Pfizer/BioNTech Ministry of Health
## 5 Pfizer/BioNTech Ministry of Health
## 6 Pfizer/BioNTech Ministry of Health
##
source_website
## 1 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
## 2 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
## 3 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
## 4 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
## 5 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
## 6 https://shendetesia.gov.al/covid19-ministria-e-shendetesise-1031-te-vaksinuar-3691-testi
me-849-te-sheruar-986-raste-te-reja-dhe-21-humbje-jete-ne-24-oret-e-fundit/
##   Pfizer/BioNTech Sputnik V Oxford/AstraZeneca Moderna Sinopharm/Beijing
## 1            TRUE     FALSE              FALSE   FALSE             FALSE
## 2            TRUE     FALSE              FALSE   FALSE             FALSE
## 3            TRUE     FALSE              FALSE   FALSE             FALSE
## 4            TRUE     FALSE              FALSE   FALSE             FALSE
## 5            TRUE     FALSE              FALSE   FALSE             FALSE
## 6            TRUE     FALSE              FALSE   FALSE             FALSE
##   Sinovac Sinopharm/Wuhan Covaxin EpiVacCorona Johnson&Johnson
```

```
## 1   FALSE           FALSE   FALSE        FALSE           FALSE
## 2   FALSE           FALSE   FALSE        FALSE           FALSE
## 3   FALSE           FALSE   FALSE        FALSE           FALSE
## 4   FALSE           FALSE   FALSE        FALSE           FALSE
## 5   FALSE           FALSE   FALSE        FALSE           FALSE
## 6   FALSE           FALSE   FALSE        FALSE           FALSE
```

Q3

Crawl data from one page of any of the following websites: sgcarmark, srx, Lazada, or a commercial website alike.

```
url = "https://www.srx.com.sg/singapore-property-listings/hdb-for-sale"
page = read_html(url)
nodes = html_nodes(page, ".listingDetailTitle")

listings = html_attr(nodes,"href")
url_header = "https://www.srx.com.sg"
urls = paste0(url_header, listings)

houses = html_text(nodes) %>% gsub("\n", "", .) %>%
  gsub("\t", "", .) %>% trimws()

hdb_data = unique(data.frame(Housing=houses, URL=urls))

for (i in 1:nrow(hdb_data)) {
  page = read_html(hdb_data[i, "URL"])
  nodes = html_nodes(page, ".row.listing-about")
  hdb_about = html_children(nodes) %>% html_text() %>%
    gsub("\n", "", .) %>% gsub("\t", "", .) %>% trimws()
  headers = hdb_about[seq(1, length(hdb_about), by=2)]
  values = hdb_about[seq(2, length(hdb_about), by=2)]
  hdb_data[i, c(headers)] = c(values)
}

head(hdb_data)
```

```
##                     Housing
## 1  Blk 935 Yishun Central 1
## 3       Blk 61 Marine Drive
## 5      Blk 28 Kelantan Road
## 7      Blk 141 Bishan Green
## 9        Blk 608 Senja Road
## 11    Blk 423 Casa Clementi
##
URL
## 1                          https://www.srx.com.sg/listings/93183871/for-sale-yishun-centra
l-1-call-now-to-enquire
## 3      https://www.srx.com.sg/listings/93209971/for-sale-marine-drive-cheap-4-room-hdb-nea
r-taonan-renovated-unit
## 5                                    https://www.srx.com.sg/listings/93299301/for-sale-kela
ntan-road-kelantan-road
## 7  https://www.srx.com.sg/listings/93222891/for-sale-bishan-green-tastefully-renovated-squ
arish-layout-high-floor
## 9  https://www.srx.com.sg/listings/92575751/for-sale-senja-road-renovated-5-room-corner-fo
r-sale-in-bukit-panjang
## 11                              https://www.srx.com.sg/listings/93284731/for-sale-casa-cle
menti-clementi-avenue-1
##                          Address    Property Name Property Type          Model
## 1   935 Yishun Central 1 (760935) Yishun Central 1   HDB 4 Rooms        Model A
## 3        61 Marine Drive (440061)     Marine Drive   HDB 4 Rooms       Improved
## 5       28 Kelantan Road (200028)    Kelantan Road   HDB 3 Rooms           <NA>
## 7   141 Bishan Street 12 (570141)     Bishan Green   HDB 4 Rooms New Generation
## 9         608 Senja Road (670608)       Senja Road   HDB 5 Rooms       Standard
## 11 423 Clementi Avenue 1 (120423)    Casa Clementi   HDB 3 Rooms        Model A
##    Bedrooms Bathrooms             Furnish Floor Level             Tenure
## 1         3         2     Fully Furnished         MID LEASEHOLD/99 years
## 3         3         1       Not Furnished         MID LEASEHOLD/99 years
## 5         2         1                <NA>        <NA> LEASEHOLD/99 years
## 7         2         2     Fully Furnished        HIGH LEASEHOLD/99 years
## 9         3         2 Partially Furnished        HIGH LEASEHOLD/99 years
## 11        2         2                <NA>         MID LEASEHOLD/99 years
##    Built Year      HDB Town                      Asking               Size
## 1        1993        Yishun $550,000 (Offer in Excess) 104 sqm (Built-up)
## 3        1976 Marine Parade      $650,000 (Negotiable)  87 sqm (Built-up)
## 5        1977  Central Area      $400,000 (Negotiable)  65 sqm (Built-up)
## 7        1988        Bishan      $550,000 (Negotiable)  84 sqm (Built-up)
## 9        2001 Bukit Panjang      $628,000 (Negotiable) 110 sqm (Built-up)
## 11       2013      Clementi      $595,000 (Negotiable)  74 sqm (Built-up)
##                    PSF Tenancy Status Date Listed Developer
## 1  $492 psf (Built-up)   Not tenanted  1629521095      <NA>
## 3  $694 psf (Built-up)   Not tenanted  1629739038      <NA>
## 5  $572 psf (Built-up)   Not tenanted  1630338005       HDB
## 7  $608 psf (Built-up)   Not tenanted  1630852004      <NA>
## 9  $530 psf (Built-up)   Not tenanted  1630851921      <NA>
## 11 $747 psf (Built-up)   Not tenanted  1630848611       HDB
```

Assignment 4

SA01 Team 2

9/12/2021

```
library(curl)
library(rvest)
library(tidyverse)
library(XML)
```

Q1

Write a programme to crawl as much information as possible about NBA teams and NBA players from ESPN NBA website: http://www.espn.com/nba/players (http://www.espn.com/nba/players) .

```
url = "http://www.espn.com/nba/players"
page = read_html(url)
nodes = html_nodes(page, ".small-logos div a")

# gets all the url of the different teams
rosters = html_attr(nodes,"href")
partial = "https://espn.com"
urls = paste0(partial,rosters)

# gets team names
teams = html_text(nodes)

# no. of teams
nteams = length(teams)

# loops through urls and crawls
df = data.frame("Team" = teams)

for (i in 1:nteams)
{
  url = curl(urls[[i]])
  urldata = readLines(url)
  data = readHTMLTable(urldata)
  data$Team = teams[[i]]
  df = merge(df, data, on="Team", all=T)
  close(url)
}

head(df)
```

```
##            Team NULL.           NULL.Name NULL.POS NULL.Age NULL.HT NULL.WT
## 1 Atlanta Hawks                 AJ Lawson        G       21   6' 5" 215 lbs
## 2 Atlanta Hawks       Bogdan Bogdanovic13       SG       29   6' 6" 220 lbs
## 3 Atlanta Hawks          Brandon Goodwin0       PG       25   6' 0" 180 lbs
## 4 Atlanta Hawks             Cam Reddish22       SF       22   6' 8" 218 lbs
## 5 Atlanta Hawks            Clint Capela15        C       27  6' 10" 240 lbs
## 6 Atlanta Hawks         Danilo Gallinari8       PF       33  6' 10" 233 lbs
##         NULL.College NULL.Salary
## 1     South Carolina          --
## 2                 -- $18,000,000
## 3 Florida Gulf Coast          --
## 4               Duke  $4,670,160
## 5                 -- $18,603,448
## 6                 -- $20,475,000
```

Cleaning df:

```
# handle NA
nba_data = df[!is.na(df$NULL.Name), ]
nba_data[nba_data == "--"] = NA

# column names
nba_data = nba_data[-2]
header = colnames(nba_data) %>%
  lapply(gsub, pattern="NULL.", replacement="")
names(nba_data) = header

# team number
nba_data = nba_data %>% separate(Name,
           into = c("Name", "Jersey"),
           sep = "(?<=[A-Za-z])(?=[0-9])",
           )

# data type
nba_data$Team = factor(nba_data$Team)
nba_data$POS = factor(nba_data$POS)
nba_data$Jersey = as.integer(nba_data$Jersey)
nba_data$Age = as.integer(nba_data$Age)

nba_data$WT = nba_data$WT %>%
  gsub(" lbs", "", .) %>%
  as.integer()
names(nba_data)[names(nba_data) == 'WT'] <- 'WT(lbs)'

nba_data$Salary = nba_data$Salary %>%
  gsub("[$,]", "", .) %>%
  as.numeric()
names(nba_data)[names(nba_data) == 'Salary'] <- 'Salary($)'

# Answer
head(nba_data)
```

```
##            Team              Name Jersey POS Age     HT WT(lbs)
## 1 Atlanta Hawks         AJ Lawson     NA   G  21  6' 5"     215
## 2 Atlanta Hawks Bogdan Bogdanovic     13  SG  29  6' 6"     220
## 3 Atlanta Hawks   Brandon Goodwin      0  PG  25  6' 0"     180
## 4 Atlanta Hawks       Cam Reddish     22  SF  22  6' 8"     218
## 5 Atlanta Hawks      Clint Capela     15   C  27 6' 10"     240
## 6 Atlanta Hawks  Danilo Gallinari      8  PF  33 6' 10"     233
##              College Salary($)
## 1     South Carolina        NA
## 2               <NA>  18000000
## 3 Florida Gulf Coast        NA
## 4               Duke   4670160
## 5               <NA>  18603448
## 6               <NA>  20475000
```

Q2

Based on the discussion today, write a procedure to clean family data (only the gene sequence column). It should include - A function to validate if a particular input complies with the format requirement; - A sequence of functions (handlers) to handle all error patterns you could identify in the column. Run the procedure written above on the given data and make sure you could obtain clean data at the end.

```
family= read_csv("./Data/The family with the largest number of children.csv")
```

```
## Rows: 165 Columns: 3
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Gender Sequence, Country, Race
```

```
##
## 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.
```

```
head(family)
```

```
## # A tibble: 6 x 3
##   `Gender Sequence` Country   Race
##   <chr>             <chr>     <chr>
## 1 female            China     Chinese
## 2 F, M, M           Singapore Chinese
## 3 M,M,F,M           China     Chinese
## 4 M,F,F             China     Chinese
## 5 M,F,F,M,M         USA       American
## 6 F,F,F,F,F         China     Asian
```

Check compliance:

```
complies = function(vec) {
  return(grepl("^([FM],)+[FM]$", vec))
}
```

Using validation function to process the data:

```
valid = complies(family$`Gender Sequence`)
family_unclean = family[!valid, ]
family_unclean$original = family_unclean$`Gender Sequence`
family = family[valid, ]
```

Sequence of handlers to handle the error patterns

1. Convert all to uppercase

```
family_unclean$`Gender Sequence` = toupper(family_unclean$`Gender Sequence`)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

2. Convert female/male into m or f

```
family_unclean$`Gender Sequence` =
  gsub("FEMALE", "F", family_unclean$`Gender Sequence`)
# FMALE vague -> F, Male or Female, handle manually
family_unclean$`Gender Sequence` =
  gsub("(?<!F)MALE", "M", family_unclean$`Gender Sequence`, perl=T)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

3. Removal of whitespaces

```
family_unclean$`Gender Sequence` =
  gsub(" ", "", family_unclean$`Gender Sequence`, fixed=T)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

4. Drop blank rows

```
family_unclean = family_unclean[!is.na(family_unclean$`Gender Sequence`), ]
```

5. Remove additional symbols

```
family_unclean$`Gender Sequence` =
  gsub("[^A-Z, ]", "", family_unclean$`Gender Sequence`)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

6. Remove additional commas

```
# remove consecutive commas
family_unclean$`Gender Sequence` =
  gsub("[,]+", ",", family_unclean$`Gender Sequence`)
# remove from the ends
family_unclean$`Gender Sequence` = family_unclean$`Gender Sequence` %>%
  gsub("^,", "", .) %>% gsub(",$", "", .)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

7. Adding in the commas

```
# find FM and breaks it with ,
select = grepl("^[FM,]+$", family_unclean$`Gender Sequence`)
family_unclean$`Gender Sequence`[select] =
  family_unclean$`Gender Sequence`[select] %>%
  gsub("(?<=[FM])(?=[FM])", ",", ., perl=T)

# Update validation
valid = complies(family_unclean$`Gender Sequence`)
family = rbind(family, subset(family_unclean[valid, ], select=-c(original)))
family_unclean = family_unclean[!valid, ]
```

8. Remove Single Child

```
only_child = grepl("^[FM]$", family_unclean$`Gender Sequence`)
family_unclean = family_unclean[!only_child, ]
```

9. Wrong words (Manual intervention)

```
family_unclean # is an only child in this case, no need to add
```

```
## # A tibble: 1 x 4
##   `Gender Sequence` Country Race    original
##   <chr>             <chr>   <chr>   <chr>
## 1 FAM               China   Chinese famale
```

Answer:

```
head(family)
```

```
## # A tibble: 6 x 3
##   `Gender Sequence` Country Race
##   <chr>             <chr>   <chr>
## 1 M,M,F,M           China   Chinese
## 2 M,F,F             China   Chinese
## 3 M,F,F,M,M         USA     American
## 4 F,F,F,F,F         China   Asian
## 5 F,F,M,F,F         China   Chinese
## 6 F,M,F,F,F         China   Chinese
```

Assignment 5 Group 2

Universal Studio Singapore announced a promotion event: each day they will give a free ticket to the first person in line whose birthday is the same as someone who has already bought a ticket. You have the option of getting in line at any time. Assuming that you don't know anyone else's birthday, that birthdays are randomly distributed throughout the year, etc.

(1) On Sunday, you wanted to try you luck. When you arrived, there are already 30 people in the queue. What is your chance of winning the free ticket if you join the queue?

Take my birthday as shown below:

```
bday <- sample(1:366, 1)
```

```
# iteration function
same_birthday <- function(iteration,birthday,queue) {

  # generate birthdays of visitors
  days <- c(rep(1:365,4),366) # where day 366 is 29 Feb
  birthdays <- sample(days,queue,TRUE)

  # check that before you, no one has had duplicated birthdays
  duplicates <- duplicated(birthdays)
  if (sum(duplicates) > 0) {

    # does not win free ticket
    ticket <- FALSE

  } else {

    # check if your birthday coincides with someone else
    all_bdays <- c(birthdays, birthday)
    if (duplicated(all_bdays)[queue+1] == TRUE){
      # wins free ticket
      ticket <- TRUE
    } else {ticket<-FALSE}
  }
  return(ticket)
}

# simulation function
uss_ticket <- function(q,no.iter){

  simulation.outcome <- data.frame(iteration = 1:no.iter,
                                   result= sapply(1:no.iter,same_birthday,birthday=bday,queue
=q))

  # probability of winning ticket
  prob <- mean(simulation.outcome$result)

  return(prob)
}
```

```
n <- 10000 # no. of iterations
q <- 30 # no. of people in queue before you

paste("The probability of winning the free ticket is",uss_ticket(q,n))
```

```
## [1] "The probability of winning the free ticket is 0.0252"
```

(2) If you have a chance to trade your position with someone in the queue, what position would you like to trade with?

```
scenarios <- data.frame(position = 2:31, winning.probability = sapply(2:31,uss_ticket,no.iter
=10000))

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
```

```
# find position with highest possibility of winning
scenarios %>% slice(which.max(winning.probability)) %>% select(position)
```

```
##   position
## 1       22
```

Assignment 6

SA01 Team 2

10/16/2021

Assignment 6

Packages:

```
library("dplyr")
library("ggmap")
library("ggplot2")
library("htmltools")
library("leaflet")
library("leaflet.extras")
library("lubridate")
library("rworldmap")
library("tidyverse")
```

Data:

```
flights = read.csv("./Data/flights.csv")
airports = read.csv("./Data/airports.csv")
airlines = read.csv("./Data/airlines.csv")
```

(1) Clean Data

flights:

```
# Format scheduled departure time
flights$SCHEDULED_DEPARTURE_DATE = paste(flights$YEAR, flights$MONTH, flights$DAY, sep="-")
flights$SCHEDULED_DEPARTURE = sprintf("%04.0f", flights$SCHEDULED_DEPARTURE)
flights$SCHEDULED_DEPARTURE = strptime(paste(flights$SCHEDULED_DEPARTURE_DATE, flights$SCHEDU
LED_DEPARTURE), format='%Y-%m-%d %H%M')

# Format departure time
flights$DEPARTURE_TIME = sprintf("%04.0f", flights$DEPARTURE_TIME)
flights$DEPARTURE_TIME = strptime(paste(flights$SCHEDULED_DEPARTURE_DATE, flights$DEPARTURE_T
IME), format='%Y-%m-%d %H%M')
wrong_date = (flights$SCHEDULED_DEPARTURE > flights$DEPARTURE_TIME) &
  (flights$DEPARTURE_DELAY > 0) &
  !is.na(flights$DEPARTURE_TIME)
flights[wrong_date, ]$DEPARTURE_TIME = flights[wrong_date, ]$DEPARTURE_TIME + days(1)

# Format wheels off time
flights$WHEELS_OFF = flights$DEPARTURE_TIME + minutes(flights$TAXI_OUT)

# Format wheels on time
flights$WHEELS_ON = flights$WHEELS_OFF + minutes(flights$AIR_TIME)

# Format arrival time
flights$ARRIVAL_TIME = flights$WHEELS_ON + minutes(flights$TAXI_IN)

# replace NA delays with 0
not_cancelled = flights$CANCELLED == 0
flights[not_cancelled, ] = flights[not_cancelled, ] %>% mutate_at(c(27:31), ~replace(., is.na
(.), 0))

# Remove unnecessary column
flights = flights[-32]

head(flights)
```

```
##   YEAR MONTH DAY DAY_OF_WEEK AIRLINE FLIGHT_NUMBER TAIL_NUMBER ORIGIN_AIRPORT
## 1 2015     1   1           4      AS            98      N407AS            ANC
## 2 2015     1   1           4      DL          2336      N958DN            DEN
## 3 2015     1   1           4      DL          2440      N651DL            SEA
## 4 2015     1   1           4      AS           108      N309AS            ANC
## 5 2015     1   1           4      AA          2392      N3HRAA            DEN
## 6 2015     1   1           4      OO          6358      N926SW            IDA
##   DESTINATION_AIRPORT SCHEDULED_DEPARTURE      DEPARTURE_TIME DEPARTURE_DELAY
## 1                 SEA 2015-01-01 00:05:00 2015-01-01 23:54:00             -11
## 2                 ATL 2015-01-01 00:30:00 2015-01-01 00:24:00              -6
## 3                 MSP 2015-01-01 00:40:00 2015-01-01 00:39:00              -1
## 4                 SEA 2015-01-01 00:45:00 2015-01-01 00:41:00              -4
## 5                 MIA 2015-01-01 01:20:00 2015-01-01 01:41:00              21
## 6                 DEN 2015-01-01 05:41:00 2015-01-01 05:39:00              -2
##   TAXI_OUT          WHEELS_OFF SCHEDULED_TIME ELAPSED_TIME AIR_TIME DISTANCE
## 1       21 2015-01-02 00:15:00            205          194      169     1448
## 2       12 2015-01-01 00:36:00            173          149      133     1199
## 3       28 2015-01-01 01:07:00            189          198      166     1399
## 4       17 2015-01-01 00:58:00            204          194      173     1448
## 5       12 2015-01-01 01:53:00            227          208      188     1709
## 6       30 2015-01-01 06:09:00            104          113       72      458
##             WHEELS_ON TAXI_IN SCHEDULED_ARRIVAL        ARRIVAL_TIME
## 1 2015-01-02 03:04:00       4               430 2015-01-02 03:08:00
## 2 2015-01-01 02:49:00       4               523 2015-01-01 02:53:00
## 3 2015-01-01 03:53:00       4               549 2015-01-01 03:57:00
## 4 2015-01-01 03:51:00       4               509 2015-01-01 03:55:00
## 5 2015-01-01 05:01:00       8               707 2015-01-01 05:09:00
## 6 2015-01-01 07:21:00      11               725 2015-01-01 07:32:00
##   ARRIVAL_DELAY DIVERTED CANCELLED CANCELLATION_REASON AIR_SYSTEM_DELAY
## 1           -22        0         0                                   0
## 2           -30        0         0                                   0
## 3             8        0         0                                   0
## 4           -14        0         0                                   0
## 5             2        0         0                                   0
## 6             7        0         0                                   0
##   SECURITY_DELAY AIRLINE_DELAY LATE_AIRCRAFT_DELAY WEATHER_DELAY
## 1              0             0                  0             0
## 2              0             0                  0             0
## 3              0             0                  0             0
## 4              0             0                  0             0
## 5              0             0                  0             0
## 6              0             0                  0             0
```

airports:

```
missing = is.na(airports$LATITUDE)

loc = paste(airports$AIRPORT, airports$CITY, ", ")
loc = loc[missing]
loc = geocode(loc, output="latlon")
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Northwest+Florida+Beach
es+International+Airport+Panama+City+,&key=xxx-tmhyhkx2Suxw3HNa6P0c0fjHc0
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Plattsburgh+International+Airport+Plattsburgh+,&key=xxx-tmhyhkx2Suxw3HNa6P0c0fjHc0
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Northeast+Florida+Regional+Airport%C3%82%C2%A0(St.+Augustine+Airport)+St.+Augustine+,&key=xxx-tmhyhkx2Suxw3HNa6P0c0fjHc0
```

```
airports[missing, c("LATITUDE", "LONGITUDE")] = loc[c("lat", "lon")]
```

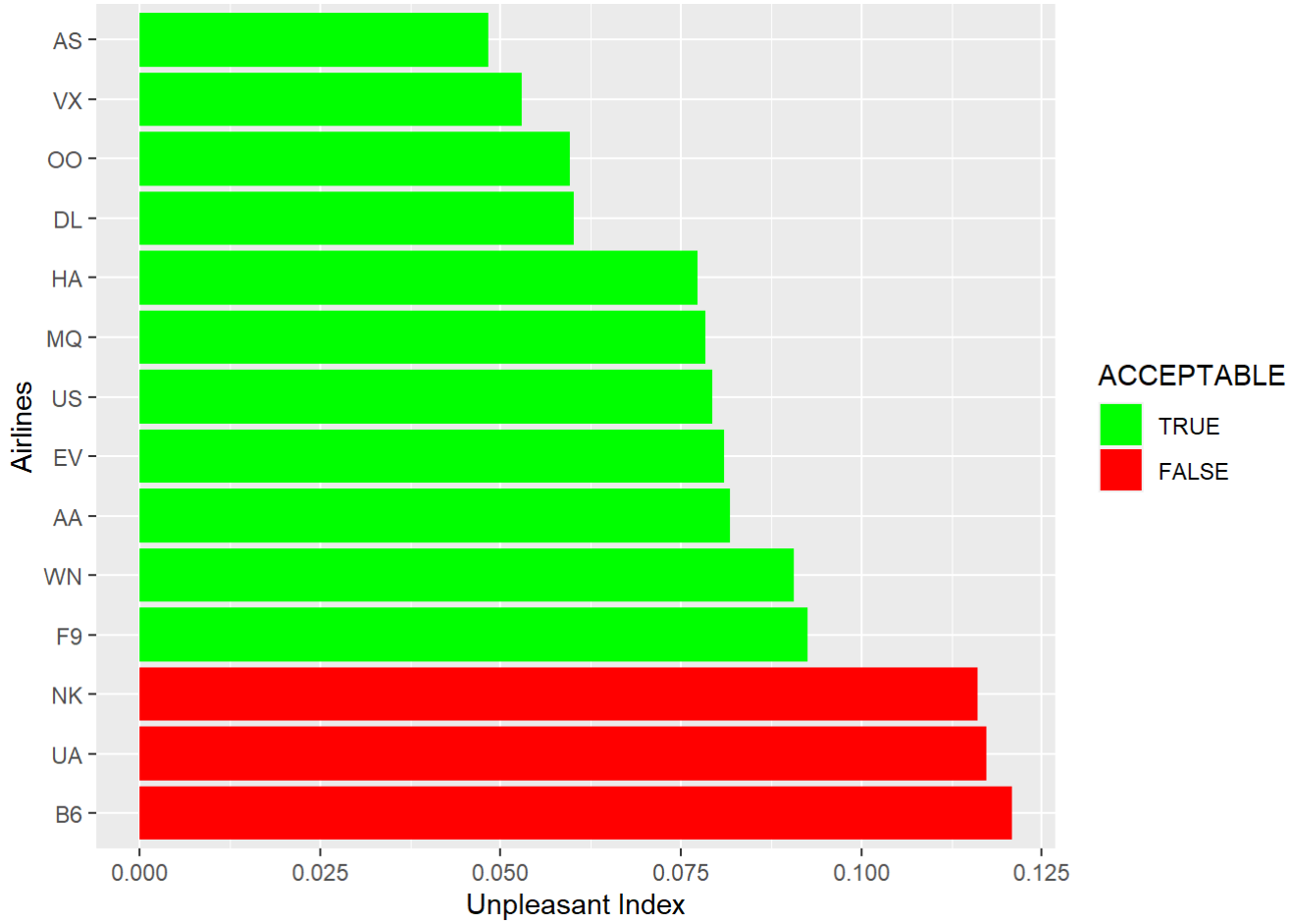
(2) Data Visualisation

Which airline to choose or avoid when travelling?

User input:

```
# User unpleasant index threshold
unpleasant_index_cutoff = 0.1

# Options
unpleasant_delay = 5
consider_diverted = TRUE
consider_cancelled = FALSE
```

```
flights$UNPLEASANT = flights$CANCELLED == 0 &
  flights$AIRLINE_DELAY > 0 & flights$AIRLINE_DELAY > unpleasant_delay

if (consider_cancelled) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$CANCELLED == 1
}

if (consider_diverted) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$DIVERTED == 1
}

data = flights %>% group_by(AIRLINE) %>%
  summarise(UNPLEASANT_INDEX = mean(UNPLEASANT)) %>%
  mutate(ACCEPTABLE = UNPLEASANT_INDEX < unpleasant_index_cutoff)

ggplot(data, aes(x=reorder(AIRLINE, -UNPLEASANT_INDEX), y=UNPLEASANT_INDEX, fill=ACCEPTABLE))
+
  geom_bar(stat="identity") + ylab("Unpleasant Index") + xlab("Airlines") +
  scale_fill_manual(values=c("green", "red"), limits=c(TRUE, FALSE)) + coord_flip()
```

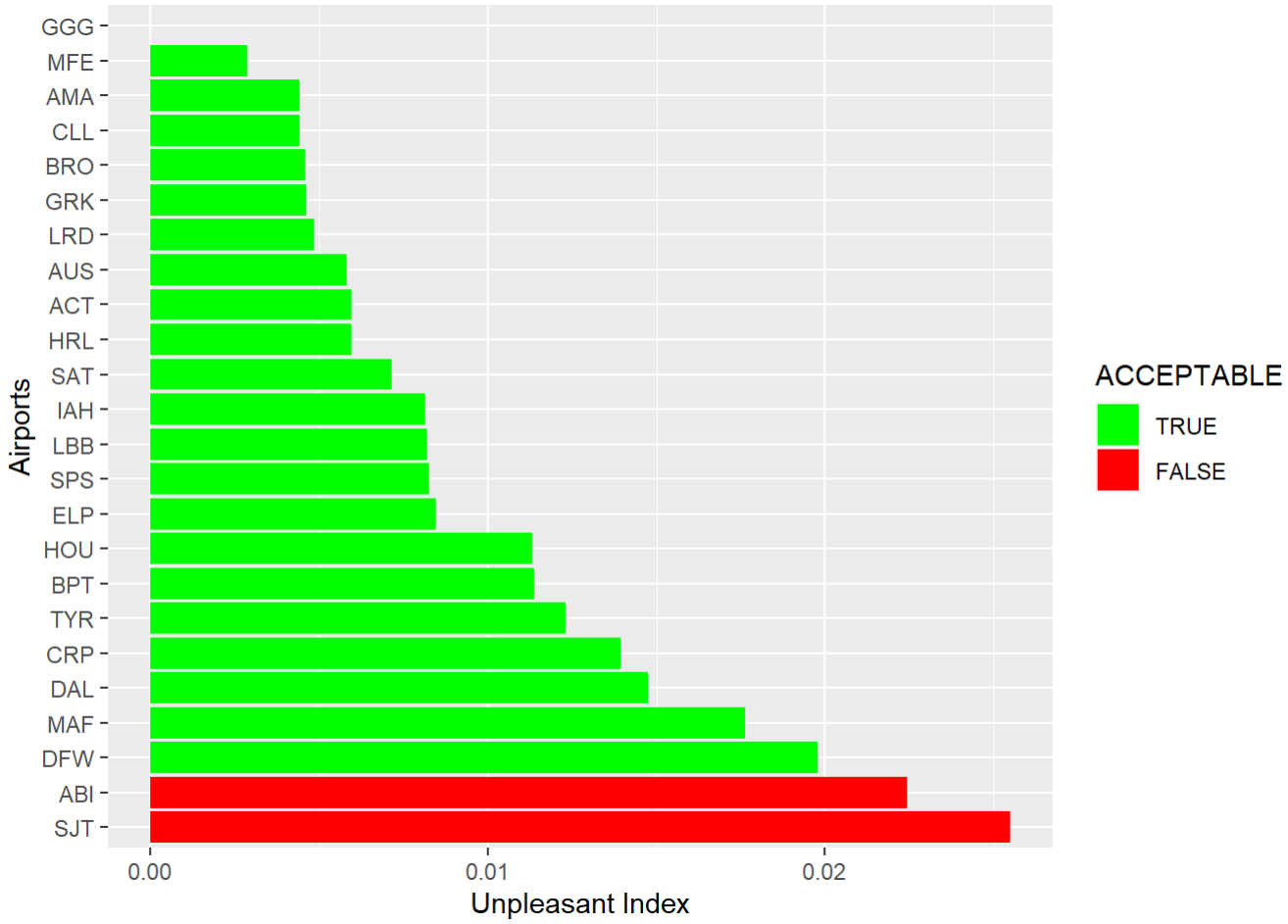


Which airport to choose or avoid when travelling (Departure from USA)?

User input:

```
# User unpleasant index threshold
unpleasant_index_cutoff = 0.02
origin_state = "TX"

# Options
unpleasant_delay = 2
consider_weather = TRUE
consider_cancelled = FALSE
unpleasant_weather_delay = 5
```

```
flights$UNPLEASANT = flights$CANCELLED == 0 &
  flights$SECURITY_DELAY > 0 & flights$SECURITY_DELAY > unpleasant_delay

if (consider_weather) {
  flights$UNPLEASANT = flights$UNPLEASANT |
    (flights$CANCELLED == 0 & flights$WEATHER_DELAY >= 10)
}

if (consider_cancelled) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$CANCELLED == 1
}

data = flights %>% merge(airports, by.x="ORIGIN_AIRPORT", by.y="IATA_CODE") %>%
  filter(tolower(origin_state) == "all" | STATE == origin_state) %>%
  group_by(ORIGIN_AIRPORT) %>%
  summarise(UNPLEASANT_INDEX = mean(UNPLEASANT)) %>%
  mutate(ACCEPTABLE = UNPLEASANT_INDEX < unpleasant_index_cutoff)

ggplot(data,
       aes(x=reorder(ORIGIN_AIRPORT, -UNPLEASANT_INDEX),
           y=UNPLEASANT_INDEX, fill=ACCEPTABLE)) +
  geom_bar(stat="identity") + ylab("Unpleasant Index") + xlab("Airports") +
  scale_fill_manual(values=c("green", "red"), limits=c(TRUE, FALSE)) + coord_flip()
```

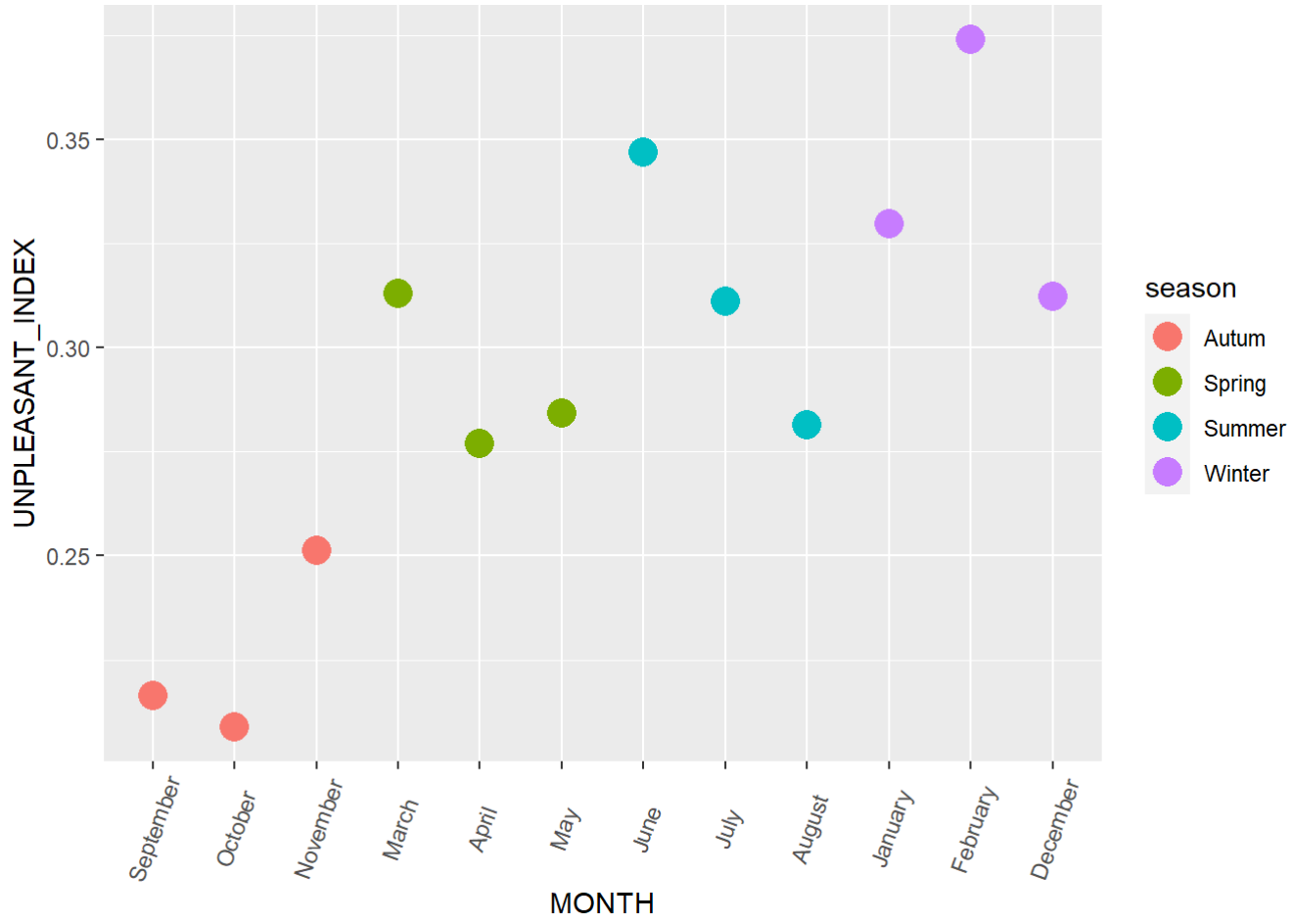


What is the seasonal effect on flight performance?

User input:

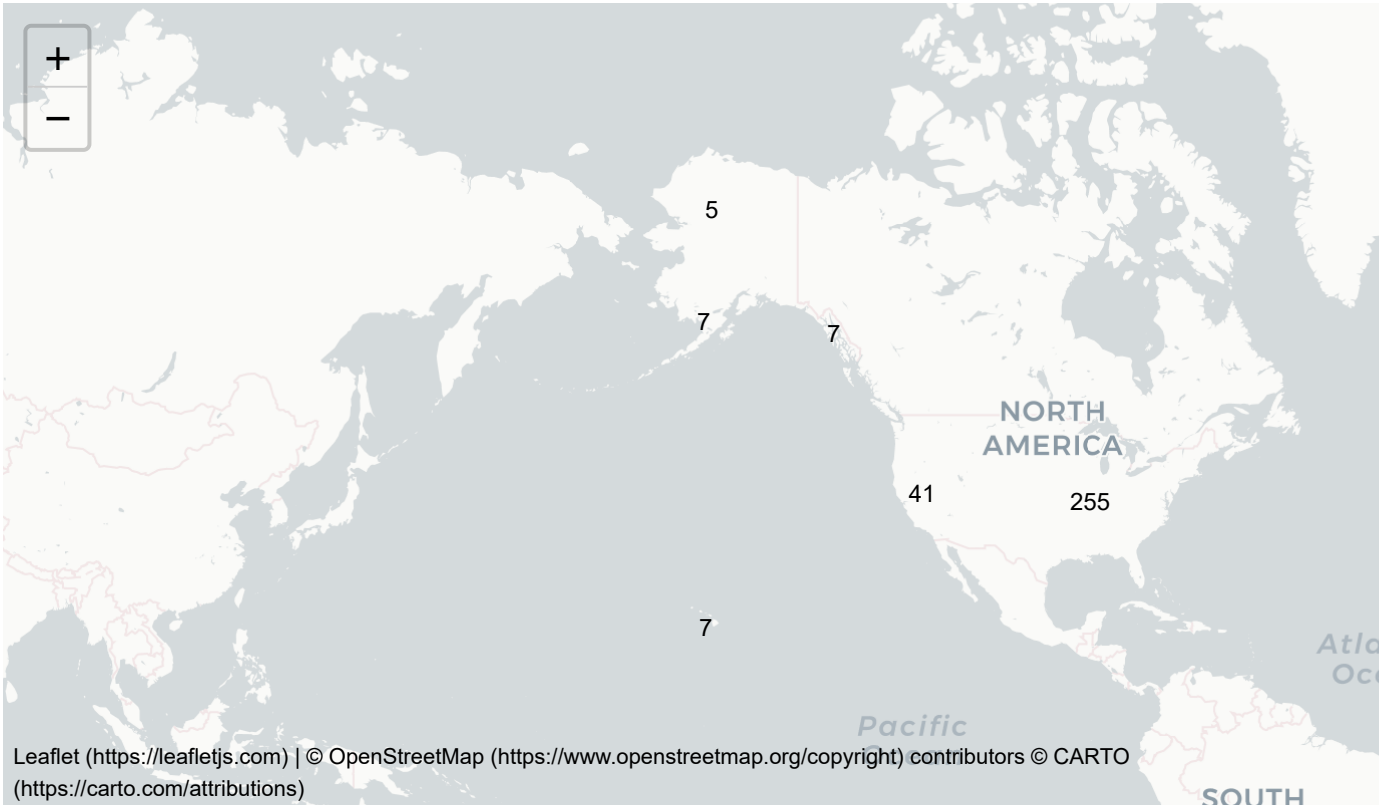
```
# Options
unpleasant_delay = 5
consider_diverted = TRUE
consider_cancelled = TRUE
```

```
flights$UNPLEASANT = flights$CANCELLED == 0 &
  flights$ARRIVAL_DELAY > unpleasant_delay

if (consider_cancelled) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$CANCELLED == 1
}

if (consider_diverted) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$DIVERTED == 1
}

season_ui = flights %>% group_by(MONTH) %>%
  summarise(UNPLEASANT_INDEX = mean(UNPLEASANT)) %>%
  mutate(MONTH_NUM = MONTH, MONTH=month.name) %>%
  mutate(season = ifelse(MONTH_NUM %in% c(3:5), "Spring",
        ifelse(MONTH_NUM %in% c(6:8), "Summer",
        ifelse(MONTH_NUM %in% c(9:11),"Autum","Winter")))) %>%
  arrange(season)

season_ui$MONTH = factor(season_ui$MONTH, levels=season_ui$MONTH)
ggplot(season_ui, aes(x=MONTH, y=UNPLEASANT_INDEX)) +
  geom_point(aes(colour=season), stat="identity", size=5) +
  theme(axis.text.x = element_text(angle=70, vjust=0.5) )
```



How to visualise the data on map?

```
# Plot the airports on world map
interactivemap <- leaflet(airports) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addCircleMarkers(~as.numeric(airports$LONGITUDE), as.numeric(airports$LATITUDE), stroke=FAL
SE,
                   fillOpacity = 0.8, popup=airports$AIRPORT,
                   clusterOptions=markerClusterOptions(), radius=5)

tagList(interactivemap)
```



User Input:

```
# Options
unpleasant_delay = 2
consider_weather = TRUE
consider_cancelled = FALSE
unpleasant_weather_delay = 5
```

```
flights$UNPLEASANT = flights$CANCELLED == 0 &
  flights$SECURITY_DELAY > 0 & flights$SECURITY_DELAY > unpleasant_delay

if (consider_weather) {
  flights$UNPLEASANT = flights$UNPLEASANT |
    (flights$CANCELLED == 0 & flights$WEATHER_DELAY >= 10)
}

if (consider_cancelled) {
  flights$UNPLEASANT = flights$UNPLEASANT | flights$CANCELLED == 1
}

data = flights %>% group_by(ORIGIN_AIRPORT) %>%
  summarise(COUNT=n(), UNPLEASANT_INDEX = mean(UNPLEASANT)) %>%
  merge(airports, by.x="ORIGIN_AIRPORT", by.y="IATA_CODE")

us = getMap(resolution = "low")
ggplot(data, aes(x=LONGITUDE, y=LATITUDE)) +
  borders("world", colour=NA, fill="wheat1")  +
  geom_point(aes(color=UNPLEASANT_INDEX, size=COUNT), alpha=0.4) +
  scale_color_viridis_c(limits=c(0, 0.05)) +
  scale_size_continuous(range=c(0,15)) +
  scale_x_continuous(name="Longitude", limits=c(-130, -60)) +
  scale_y_continuous(name="Latitude", limits=c(15, 50)) +
  labs(title="Magnitude of flights")
```

```
## Warning: Removed 26 rows containing missing values (geom_point).
```