CptS 575_Assignment 03

Sajjad Uddin Mahmud | WSU ID: 011789534

2022-09-21

Solution of Problem 01 : Women's National Basketball Association

Initialization

```
# Loading the packages
suppressMessages(library(dplyr))

# Reading the data set
WNBA <- read.csv("WNBA_Stats_21.csv")
```

```
# Printing the first few values of the columns with a header contains the string "FG"
Column_FG <- head(select(WNBA,contains("FG")))
Column_FG
```

```
##   FGM FGA
## 1 207 466
## 2   6  26
## 3  56 174
## 4  61 143
## 5   8  34
## 6 121 270
```

1(a)

```
# Filtering the #players with Free Throws Made>50 and Assists>75
FTM_50_AST_75 <- count((filter(WNBA, FTM>50 & AST>75 )))
```

The number of players with Free Throws Made greater than 50 and Assists greater than 75 is **18**

1(b)

```
# Initiating Pipe
WNBA %>%
  select(PLAYER, TEAM, FGM, TO, PTS) %>% # Selecting columns
  arrange(desc(PTS)) %>% # Arranging players in descending order of points
```

```
  head(n=10) -> # Printing the players with 10 highest points
  Highest_Points # Assigning the data in variable

Highest_Points
```

```
##                  PLAYER TEAM FGM  TO PTS
## 1          Tina Charles  WAS 238  59 631
## 2       Brittney Griner  PHO 248  66 615
## 3      Arike Ogunbowale  DAL 199  68 599
## 4           A'ja Wilson  LVA 207  46 584
## 5       Breanna Stewart  SEA 194  47 569
## 6       Kelsey Mitchell  IND 212  65 569
## 7  Skylar Diggins-Smith  PHO 177  82 566
## 8           Jewell Loyd  SEA 193  71 555
## 9         Betnijah Laney  NYL 203 119 536
## 10   Courtney Williams  ATL 228  58 529
```

```
# Finding the player with the second highest point

Highest_Points %>%
  slice(2) %>% # Slicing the second row only (as it was already in descending order)
  select(PLAYER) -> # Selecting the name of the player
  Player_Name
```

The player who has the second highest point is **Brittney Griner**

1(c)

```
# Adding new columns
WNBA %>%
  mutate(FGP=round(c(.$FGM/.$FGA*100),2)) -> # Adding %FGP = FGM/FGA *100
  WNBA

WNBA %>%
  mutate(FTP=round(c(.$FTM/.$FTA*100),2)) -> # Adding %FTP = FTM/FTA *100
  WNBA

# FGP and FTP for Tina Charles
Profile_TinaCharles <- WNBA[WNBA$PLAYER=="Tina Charles",] # Selecting rows of Tina Charles
FGP_TinaCharles <- Profile_TinaCharles$FGP
FTP_TinaCharles <- Profile_TinaCharles$FTP
```

For Tina Charles, **FGP = 44.91** and **FTP = 82.03**

1(d)

Average (Mean), Max and Min REB for each teams:

```
WNBA_Team <- group_by(WNBA,TEAM) # Grouping by teams
REB_Summary <- summarise_at(WNBA_Team, vars(REB), funs(mean,max,min)) # Calculating mean, max and min o
REB_Summary_Sort_Mean <- arrange(REB_Summary, desc(mean)) # Arranging players in descending order of me
REB_Summary_Sort_Mean
```

```
## # A tibble: 12 x 4
##    TEAM   mean   max   min
##    <chr> <dbl> <int> <int>
##  1 LVA   115.    298     0
##  2 CON   105     303    10
##  3 PHO   104.    302     4
##  4 CHI    98.9   193    11
##  5 DAL    95.8   173     3
##  6 SEA    93.9   267    19
##  7 NYL    93.5   171    21
##  8 MIN    93.3   312     4
##  9 ATL    89.5   219    14
## 10 WAS    86.6   258    13
## 11 IND    84.4   308     6
## 12 LAS    78     154     2
```

```
# Finding the team with the highest REB
REB_Summary_Sort_Max <- arrange(REB_Summary,desc(max)) # Arranging players in descending order of mean
REB_Max_Team_Summary <- slice(REB_Summary_Sort_Max,1) # Slicing the 1st row
REB_Max_Team_Name <- REB_Max_Team_Summary$TEAM # Getting team's name
```

Team **MIN** has the max REB.

1(e)

At first lets see where the missing FTPs are:

```
WNBA_Team %>% # Already grouped by teams in 1(d)
  select(PLAYER, TEAM, FGP, FTP) %>%
  arrange(desc(FTP))->
  WNBA_Filter

tail(WNBA_Filter,10) # Showing last 10 rows which has NaN values
```

```
## # A tibble: 10 x 4
## # Groups:   TEAM [8]
##    PLAYER              TEAM    FGP   FTP
##    <chr>               <chr> <dbl> <dbl>
##  1 Beatrice Mompremier CON    49.1  41.7
##  2 Shekinna Stricklen  ATL    25.9  40
##  3 Alanna Smith        PHO    23.5  25
##  4 Jasmine Walker      LAS     0     0
##  5 Angel McCoughtry    LVA   NaN   NaN
##  6 Blake Dietrick      ATL    29.6 NaN
##  7 Bria Hartley        PHO    56.2 NaN
##  8 Chelsea Dungee      DAL    17.6 NaN
```

```
##  9 Jillian Alleyne     MIN    NaN    NaN
## 10 Lexie Brown         CHI     26.3 NaN
```

From the above table we can see that, Angel McCoughtry, Blake Dietrick, Bria Hartley, Chelsea Dungee, Jillian Alleyne and Lexie Brown have NaN in their FTP. Also note that Angel McCoughtry and Jillian Alleyne have NaN in FGP.

Method 1 - Replacing NaN of FTP by FGP times mean of Team FTP:

```
WNBA_Filter %>%
  mutate(FTP=ifelse(is.na(FTP),(mean(FTP,na.rm=TRUE)*FGP/100),FTP)) ->
  WNBA_FTP_Method1

tail(WNBA_FTP_Method1,10) # Showing last 10 rows which had NaN values
```

```
## # A tibble: 10 x 4
## # Groups:   TEAM [8]
##    PLAYER              TEAM    FGP   FTP
##    <chr>               <chr> <dbl> <dbl>
##  1 Beatrice Mompremier CON    49.1  41.7
##  2 Shekinna Stricklen  ATL    25.9  40
##  3 Alanna Smith        PHO    23.5  25
##  4 Jasmine Walker      LAS     0     0
##  5 Angel McCoughtry    LVA    NaN   NaN
##  6 Blake Dietrick      ATL    29.6  20.9
##  7 Bria Hartley        PHO    56.2  40.2
##  8 Chelsea Dungee      DAL    17.6  14.1
##  9 Jillian Alleyne     MIN    NaN   NaN
## 10 Lexie Brown         CHI    26.3  22.7
```

Method 2 - Replacing NaN of FTP by mean of Team FTP:

```
WNBA_Filter %>%
  mutate(FTP=ifelse(is.na(FTP),(mean(FTP,na.rm=TRUE)),FTP)) ->
  WNBA_FTP_Method2

tail(WNBA_FTP_Method2,10) # Showing last 10 rows which had NaN values
```

```
## # A tibble: 10 x 4
## # Groups:   TEAM [8]
##    PLAYER              TEAM    FGP   FTP
##    <chr>               <chr> <dbl> <dbl>
##  1 Beatrice Mompremier CON    49.1  41.7
##  2 Shekinna Stricklen  ATL    25.9  40
##  3 Alanna Smith        PHO    23.5  25
##  4 Jasmine Walker      LAS     0     0
##  5 Angel McCoughtry    LVA    NaN   79.3
##  6 Blake Dietrick      ATL    29.6  70.5
##  7 Bria Hartley        PHO    56.2  71.4
##  8 Chelsea Dungee      DAL    17.6  79.8
##  9 Jillian Alleyne     MIN    NaN   80.9
## 10 Lexie Brown         CHI    26.3  86.3
```

As the FGP of Angel McCoughtry and Jillian Alleyne are NaN and it is multiplied by the mean FTP in method 1, so the replacement value is also NaN.

Note that, in method 1, the multiplication is also divided by hundred as FGP and FTP are both in percentage.

Assumptions behind two methods:

Here, FGP essentially indicates a player's field goal success percentage, while FTP indicates a player's free throw success rate. The theory behind approach 1 may be that if we don't know a player's FTP but do know their team's mean FTP and their own personal FGP, we may anticipate their FTP by multiplying their FGP by the team's mean FTP. The second approach, on the other hand, simply adds the average FTP of the squad to the FTP of the absent player.

Which method is better?

The first of these two approaches makes more sense. This is so that it correlates with both the player's personal ability and the team's FTP by multiplying with their FGP. However, the second approach may be appropriate in the cases where the player's FGP is missing, as was described before.

Solution of Problem 02

Reading and tidying the WHO data set

```
# Initialization
library(tidyverse)

# Reading WHO data
WHO_Data <- tidyr::who

# Tidying the data set
WHO_Data_Tidy <- WHO_Data %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  ) %>%
  mutate(
    key = stringr::str_replace(key, "newrel", "new_rel")
  ) %>%
  separate(key, c("new", "var", "sexage")) %>%
  select(-new, -iso2, -iso3) %>%
  separate(sexage, c("sex", "age"), sep = 1)
```

2(a)

> mutate(key = stringr::str_replace(key, “newrel”, “new_rel”))

This is done to make all variable names consistent.

The “key” variable includes a number of parts, as can be seen if we examine the variable’s structure.For instance, the input “new_sp_m1524” has a few pieces of information.The first three characters “new” show whether this is a new or old case; the next two letters “sp” explain the kind of TB; the sixth letter “m” provides the sex of TB patients; and the remaining numbers “1524” identify the age group (age 15-24).

Here, the “_” separates the first three characters from the following two letters in line. However, it was absent in the case of “newrel.” Therefore, if we don’t make this the same as the other cases, we can run into trouble or lose information in the next analysis utilizing this variable.

2(b)

```
# Removing missing values while tidying data set
WHO_Data_Tidy2 <- WHO_Data %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  )
```

```
# Keeping missing values while tidying data set
WHO_Data_Tidy3 <- WHO_Data %>%
  pivot_longer(
    cols = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
  )

# Counting the entries that are removed
Removed_Data <- count(WHO_Data_Tidy3) - count(WHO_Data_Tidy2)
```

329394 entries are removed from the dataset when we set values_drop_na to true in the pivot_longer command (in this dataset).

2(c)

Explicit missing values are the values that are defined already as missing in the dataset. On the contrary, the implicit missing values are the values that are just not present in the dataset. As per Zen-like koan's statement: "An explicit missing value is the presence of an absence; an implicit missing value is the absence of a presence."

```
# Counting implicit data

WHO_Implicit <- complete(WHO_Data, fill=list("implicit_miss"),explicit=FALSE)
Implicit_Data_Count <- length(str_count(WHO_Implicit, "implicit_miss"))
```

Out of 329394 missing data, there are only **60** implicit missing data in the WHO data set.

2(d)

```
head(WHO_Data_Tidy, n=10)
```

```
## # A tibble: 10 x 6
##    country      year var   sex   age   cases
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1997 sp    m     1524     10
##  3 Afghanistan  1997 sp    m     2534      6
##  4 Afghanistan  1997 sp    m     3544      3
##  5 Afghanistan  1997 sp    m     4554      5
##  6 Afghanistan  1997 sp    m     5564      2
##  7 Afghanistan  1997 sp    m     65        0
##  8 Afghanistan  1997 sp    f     014       5
##  9 Afghanistan  1997 sp    f     1524     38
## 10 Afghanistan  1997 sp    f     2534     36
```

The tidied data are appropriately typed. It shows all the information in a structured way. However, if the age can be shown as 35-44 not as 3544, that would be better.

2(e)

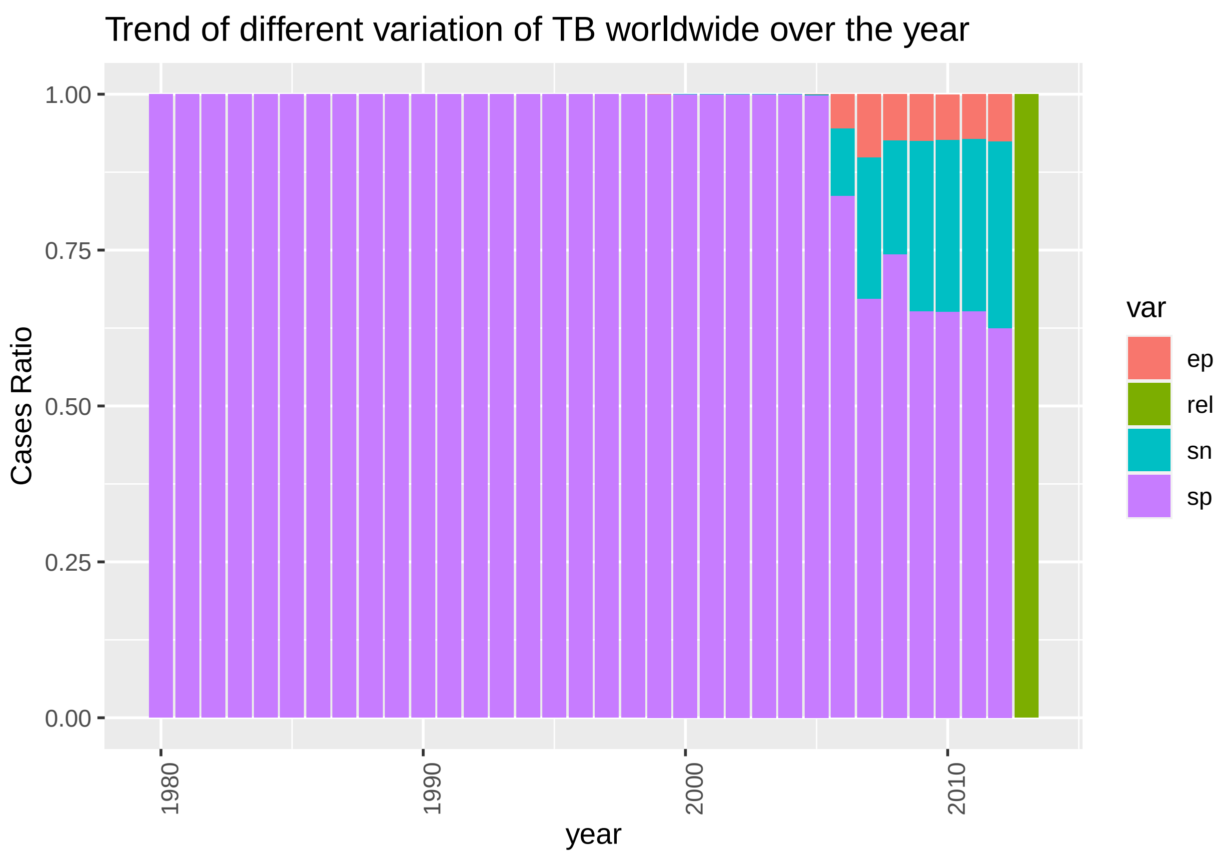
Here, I wanted to show how different TB types was emerged over the year in the world.

```
library(plotly)

WHO_Data_Tidy %>%
  group_by(var,year) %>%
  summarise(cases = sum(cases)) ->
  WHO_Visualization_Data

WHO_Visualization <- ggplot(WHO_Visualization_Data,aes(fill=var, x=year,y=cases))+geom_bar(position="fil

WHO_Visualization
```



From the above graph we can see that the the TB variation data stored since 1980 and until 2005 there is only one variant which is SP = pulmonary TB that could be diagnosed by a pulmonary smear (smear positive). Then other variable except rel was come. However, in 2013, all the recorded cases are from type rel = TB relapse. This chart shows us the different types of TB condition over the year.

2(f)

```
head(WHO_Data_Tidy, n=10)
```

```
## # A tibble: 10 x 6
##    country      year var   sex   age   cases
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1997 sp    m     014       0
##  2 Afghanistan  1997 sp    m     1524     10
##  3 Afghanistan  1997 sp    m     2534      6
##  4 Afghanistan  1997 sp    m     3544      3
##  5 Afghanistan  1997 sp    m     4554      5
##  6 Afghanistan  1997 sp    m     5564      2
##  7 Afghanistan  1997 sp    m     65        0
##  8 Afghanistan  1997 sp    f     014       5
##  9 Afghanistan  1997 sp    f     1524     38
## 10 Afghanistan  1997 sp    f     2534     36
```

```r
# Reading the data set
SchQrt <- read.csv("SchQtr.csv")

# Restructuring the data set
SchQrt %>%
  pivot_longer(
    cols = starts_with("Qtr"),
    names_to = "Interval",
    values_to = "Student_Count",
    values_drop_na = TRUE) %>%
  separate(Interval,c("Interval_Type","Interval_ID")) ->
  SchQrt_Restructured

SchQrt_Restructured
```

```
## # A tibble: 48 x 5
##    School  Year Interval_Type Interval_ID Student_Count
##    <chr>  <int> <chr>         <chr>               <int>
##  1 UNI     2018 Qtr           1                      27
##  2 UNI     2018 Qtr           2                      90
##  3 UNI     2018 Qtr           3                      12
##  4 UNI     2018 Qtr           4                      84
##  5 COL     2018 Qtr           1                      42
##  6 COL     2018 Qtr           2                      27
##  7 COL     2018 Qtr           3                      62
##  8 COL     2018 Qtr           4                       1
##  9 ACA     2018 Qtr           1                       6
## 10 ACA     2018 Qtr           2                      51
## # ... with 38 more rows
```

```r
# Counting rwo number in the new dataset
Row_Number <- nrow(SchQrt_Restructured)
```

There are **48** rows in the new dataset.