Data Analytics HW3_Part B

Tanmay Nema

10/27/2020

PART B: Data analysis with the R tidyverse package

Question 4: Tidying the dataset

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ purrr   0.3.4
## ✓ tibble  3.0.4     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.2     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

4.a: Begin again with the “raw” data from the CSV file, and again create a dataframe called raw_data_tidy including only the date (dt or dt_iso), city_id, and seven other variables (temp, humidity, wind_speed, rain_1h, snow_1h, weather_main and weather_description). Note the last two are slightly different than before.

Ans: Tidyverse is used for this portion of the homework assignment. read_csv (instead of read.csv) is used to import csv file into R studio. The data frame had to be set to character as default to avoid errors. Therefore, once it was imported correctly, the column names and the attribute classes were adjusted properly.

```
#4.a
holder <- read_csv("raw_data.csv",col_names=TRUE,col_types = cols(.default = col_character()))
raw_data_tidy <- data.frame(holder$dt_iso,holder$city_id,holder$temp,holder$humidity,holder$wind_speed,holder$rai
n_1h,holder$snow_1h,holder$weather_main,holder$weather_description)

raw_data_tidy <- raw_data_tidy %>% rename(Date=holder.dt_iso,City_ID=holder.city_id,Temperature=holder.temp,Humid
ity=holder.humidity,Wind_Speed=holder.wind_speed,Rain_1hr=holder.rain_1h,Snow_1hr=holder.snow_1h,Weather_Main=hol
der.weather_main,Weather_Description=holder.weather_description)

raw_data_tidy$Date <- as.POSIXct(raw_data_tidy$Date, tz="UTC")
raw_data_tidy$City_ID <- as.factor(raw_data_tidy$City_ID)
raw_data_tidy$Temperature <- as.numeric(raw_data_tidy$Temperature)
raw_data_tidy$Humidity <- as.integer(raw_data_tidy$Humidity)
raw_data_tidy$Wind_Speed <- as.integer(raw_data_tidy$Wind_Speed)
raw_data_tidy$Rain_1hr <- as.integer(raw_data_tidy$Rain_1hr)
raw_data_tidy$Snow_1hr <- as.integer(raw_data_tidy$Snow_1hr)
raw_data_tidy$Weather_Main <- as.factor(raw_data_tidy$Weather_Main)
raw_data_tidy$Weather_Description <- as.factor(raw_data_tidy$Weather_Description)

head(raw_data_tidy)
```

```
##                  Date City_ID Temperature Humidity Wind_Speed Rain_1hr Snow_1hr
## 1 2012-10-10 16:00:00 2643743      286.10       66          4       NA       NA
## 2 2012-10-10 17:00:00 2643743      285.34       71          4       NA       NA
## 3 2012-10-10 18:00:00 2643743      284.48       71          4       NA       NA
## 4 2012-10-10 19:00:00 2643743      284.16       91          0        0       NA
## 5 2012-10-10 20:00:00 2643743      283.78       76          2       NA       NA
## 6 2012-10-10 21:00:00 2643743      283.41       96          0        0       NA
##   Weather_Main Weather_Description
## 1       Clouds       broken clouds
## 2       Clouds       broken clouds
## 3       Clouds       broken clouds
## 4        Clear        sky is clear
## 5       Clouds    scattered clouds
## 6        Clear        sky is clear
```

4.b: Remove duplicate values from raw_data_tidy.

Ans: Similar to unique() in Base R, distinct() function was used to remove the duplicate records. Effectively, 4289 recirds were deleted.

```
#4.b
raw_data_tidy <- distinct(raw_data_tidy)
str(raw_data_tidy)
```

```
## 'data.frame':    249626 obs. of  9 variables:
##  $ Date               : POSIXct, format: "2012-10-10 16:00:00" "2012-10-10 17:00:00" ...
##  $ City_ID            : Factor w/ 6 levels "2643743","2950159",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Temperature        : num  286 285 284 284 284 ...
##  $ Humidity           : int  66 71 71 91 76 96 85 85 85 85 ...
##  $ Wind_Speed         : int  4 4 4 0 2 0 2 2 2 2 ...
##  $ Rain_1hr           : int  NA NA NA 0 NA 0 NA NA NA NA ...
##  $ Snow_1hr           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Weather_Main       : Factor w/ 13 levels "Clear","Clouds",..: 2 2 2 1 2 1 1 1 1 1 ...
##  $ Weather_Description: Factor w/ 49 levels "broken clouds",..: 1 1 1 38 33 38 38 38 38 38 ...
```

4.c: Show all unique text weather_descriptions associated with the weather_main values of Cloudy and Rain in raw_data_tidy. Use assumptions to reduce the number of unique weather_descriptions (e.g., observe similarities in the weather_description text strings, and make assumptions to combine them to slightly reduce the number of weather_descriptions by using regular expressions!)

Ans: Pipe were employed to filter the unique weather descriptions associated with ‘clouds’ and ‘rain’. These were stored in separate data frames. There are 14 distinct values of weather descriptions for the relevant weather main categories

```
#4.c
cloudy <- raw_data_tidy %>%
  filter(Weather_Main == "Clouds") %>%
  select(Weather_Main,Weather_Description) %>% distinct_all()
cloudy
```

```
##   Weather_Main Weather_Description
## 1       Clouds       broken clouds
## 2       Clouds    scattered clouds
## 3       Clouds          few clouds
## 4       Clouds     overcast clouds
```

```
rainy <- raw_data_tidy %>%
  filter(Weather_Main == "Rain") %>%
  select(Weather_Main,Weather_Description) %>% distinct_all()
rainy
```

```
##    Weather_Main         Weather_Description
## 1          Rain light intensity shower rain
## 2          Rain                   light rain
## 3          Rain               moderate rain
## 4          Rain                 shower rain
## 5          Rain       proximity shower rain
## 6          Rain heavy intensity shower rain
## 7          Rain        heavy intensity rain
## 8          Rain             very heavy rain
## 9          Rain               freezing rain
## 10         Rain          ragged shower rain
```

In addition, regular expression were used to segregate specific categories and group them as distinct categories.Since the clouds were only 4 and relatively unrelated, they weren’t included. However, rain categories were revised.

```
category_heavy <- rainy%>%filter(str_detect(Weather_Description, "^.*heavy.*$")) %>%
  count(Weather_Description)
colnames(category_heavy) = c("Heavy_Rains")
category_heavy
```

```
##                   Heavy_Rains NA
## 1        heavy intensity rain  1
## 2 heavy intensity shower rain  1
## 3              very heavy rain  1
```

```
category_light <- rainy%>%filter(str_detect(Weather_Description, "^light")) %>%
  count(Weather_Description)
colnames(category_light) = c("Light_Rains")
category_light
```

```
##                   Light_Rains NA
## 1 light intensity shower rain  1
## 2                  light rain  1
```

```
category_freezing <- rainy%>%filter(str_detect(Weather_Description, "^.*free.*$")) %>%
  count(Weather_Description)
colnames(category_freezing) = c("Freezing_Rains")
category_freezing
```

```
##   Freezing_Rains NA
## 1  freezing rain  1
```

4.d: Using your findings from Part A on other problems with the raw data (outliers, missing values, data gaps, etc.), use tidyverse to perform at least three types of data cleaning to fix major data errors in raw_data_tidy. Two of these should be filling missing values and fixing outliers. Of course, be sure to document your cleaning steps and discuss and show how the clean data compares to the raw data. Name the cleaned dataframe clean_data_tidy.

Ans: The following three data cleaning activities were performed on the rat data - (1) Removing of extreme outliers from temperature, wind speed, humidity attributes based on their spread. (2) Second, all missing values, which were “NA”, were filled as“0” into the data so as to make it uniform and coherent in terms of the data types. (3) Lastly, data cleaning in terms of proper column names, data types eg. POSIXct, and factors (for categorical variables) such as City_ID, Weather_Main, Weather_Description etc. were created to avoid any confusion with the data. This step was, however, completed at the very beginning of Past B. We can see the difference after cleaning the data (compared to structure in 4.b).

```
# 4.d
clean_data_tidy <- raw_data_tidy%>%filter(Temperature > 200, Humidity < 110, Wind_Speed < 20)
clean_data_tidy$Rain_1hr <- clean_data_tidy$Rain_1hr%>% replace_na(0)
clean_data_tidy$Snow_1hr <- clean_data_tidy$Snow_1hr%>% replace_na(0)
head(clean_data_tidy)
```

```
##                   Date City_ID Temperature Humidity Wind_Speed Rain_1hr Snow_1hr
## 1 2012-10-10 16:00:00 2643743      286.10       66          4        0        0
## 2 2012-10-10 17:00:00 2643743      285.34       71          4        0        0
## 3 2012-10-10 18:00:00 2643743      284.48       71          4        0        0
## 4 2012-10-10 19:00:00 2643743      284.16       91          0        0        0
## 5 2012-10-10 20:00:00 2643743      283.78       76          2        0        0
## 6 2012-10-10 21:00:00 2643743      283.41       96          0        0        0
##   Weather_Main Weather_Description
## 1       Clouds       broken clouds
## 2       Clouds       broken clouds
## 3       Clouds       broken clouds
## 4        Clear        sky is clear
## 5       Clouds    scattered clouds
## 6        Clear        sky is clear
```

```
str(clean_data_tidy)
```

```
## 'data.frame':    249610 obs. of  9 variables:
##  $ Date               : POSIXct, format: "2012-10-10 16:00:00" "2012-10-10 17:00:00" ...
##  $ City_ID            : Factor w/ 6 levels "2643743","2950159",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Temperature        : num  286 285 284 284 284 ...
##  $ Humidity           : int  66 71 71 91 76 96 85 85 85 85 ...
##  $ Wind_Speed         : int  4 4 4 0 2 0 2 2 2 2 ...
##  $ Rain_1hr           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Snow_1hr           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Weather_Main       : Factor w/ 13 levels "Clear","Clouds",..: 2 2 2 1 2 1 1 1 1 1 ...
##  $ Weather_Description: Factor w/ 49 levels "broken clouds",..: 1 1 1 38 33 38 38 38 38 38 ...
```

Question 5: Custom functions, iterations, conditions, vectorized methods

a. Define a custom R function to compute apparent temperature. We use wind-chill temperature for apparent temperature when temperatures are at or below 50 °F and wind speeds above 3 mph. The wind chill temperature (units of degrees F) can be calculated as follows1: Twc= 35.74 + 0.6215T - 35.75$V^{0.16}$ + 0.4275T(V0.16) where T is air temperature in degrees Fahrenheit and V is wind speed in miles per hour

Ans: A user defined function was created to estimated the apparent temperature given the minimum temperature and wind speed conditions are satisfied. The function accepts the temperature and wind speed values in K and mps and converts these to Fahrenheit and mph respectively. Condition was applied that if temperature in F is > 50 & wind speed < 3 then the function return original value. Otherwise, the apparent value was calculated using the given formula.

```
# 5.a
apparent_temperature <- function(temp,wind){
  temp <- (temp-273.15)*(9/5) + 32
  wind <- 2.23694*wind
  if (temp>50|wind<3){
    return (temp)
  } else{
    twc <- 35.47+0.6215*temp - 35.75*(wind^0.16) + 0.4275*temp*(wind^0.16)
    return (twc)
  }
}
```

5.b: b) Using your apparent temperature function above, add a column to clean_data_tidy for apparent temperatures of each hourly temperature value. Do this with three different methods: (1) by using a for loop, (2) with map or apply, and (3) with a dplyr function. For each way, record the elapsed run time (by using the R Sys.time function). Compare the elapsed times needed to run each of the three ways and discuss which seems to be the most efficient.

Ans: The three different methods were used to update the apparent temperature value in the clean_data_tdy data frame.

First, a for loop was created which ran from 1st row to the last row updating the apparent temperature using the formula/function used above. Timing analysis was performed using the Sys.time() function before and after the loop to get the difference in time. Second, lapply was used to inherently loop over the records and update the apparent temperature. Similarly timing analysis was performed for this.

Finally, mutate() was used through pipe to update the apparent temperature. The timing analysis was performed for this as well.

It was found that loop has the highest required time followed by dplyr function and lapply was to be the most computationally inexpensive alternative.

```
# 5.b.1
clean_data_tidy_temp <- clean_data_tidy
start <- Sys.time()
iter = c(1:nrow(clean_data_tidy))
for (i in iter){
  clean_data_tidy_temp$Apparent_Temperature[i] <- apparent_temperature((clean_data_tidy$Temperature[i]),(clean_da
ta_tidy$Wind_Speed[i]))
}
end <- Sys.time()
duration <- end-start
head(clean_data_tidy_temp)
```

```
##                  Date City_ID Temperature Humidity Wind_Speed Rain_1hr Snow_1hr
## 1 2012-10-10 16:00:00 2643743      286.10       66          4        0        0
## 2 2012-10-10 17:00:00 2643743      285.34       71          4        0        0
## 3 2012-10-10 18:00:00 2643743      284.48       71          4        0        0
## 4 2012-10-10 19:00:00 2643743      284.16       91          0        0        0
## 5 2012-10-10 20:00:00 2643743      283.78       76          2        0        0
## 6 2012-10-10 21:00:00 2643743      283.41       96          0        0        0
##   Weather_Main Weather_Description Apparent_Temperature
## 1       Clouds       broken clouds               55.310
## 2       Clouds       broken clouds               53.942
## 3       Clouds       broken clouds               52.394
## 4        Clear        sky is clear               51.818
## 5       Clouds    scattered clouds               51.134
## 6        Clear        sky is clear               50.468
```

```
duration
```

```
## Time difference of 3.295377 mins
```

```
# 5.b.2
wind_chill <- function(temp,wind){
    twc <- 35.47+0.6215*temp - 35.75*(wind^0.16) + 0.4275*temp*(wind^0.16)
    return (twc)
  }
start2 <- Sys.time()
clean_data_tidy_temp1 <- clean_data_tidy%>% mutate(Temp_F=((Temperature-273.15)*(9/5) + 32),Wind_MPH = 2.23694*Wi
nd_Speed) %>% filter(Temp_F<50 & Wind_MPH>3)
l <- list(t=clean_data_tidy_temp1$Temp_F,w=clean_data_tidy_temp1$Wind_MPH)
clean_data_tidy_appTemp <- lapply(l, wind_chill,wind=l$w)
end2 <- Sys.time()
duration2 <- end2-start2
clean_data_tidy_temp1$Apparent_Temperature2 <- unlist(clean_data_tidy_appTemp[1])
head(clean_data_tidy_temp1)
```

```
##                  Date City_ID Temperature Humidity Wind_Speed Rain_1hr Snow_1hr
## 1 2012-10-10 23:00:00 2643743       282.2       85          2        0        0
## 2 2012-10-11 00:00:00 2643743       282.2       85          2        0        0
## 3 2012-10-11 01:00:00 2643743       282.2       85          2        0        0
## 4 2012-10-11 02:00:00 2643743       282.2       85          2        0        0
## 5 2012-10-11 03:00:00 2643743       282.2       85          2        0        0
## 6 2012-10-11 04:00:00 2643743       282.2       85          2        0        0
##   Weather_Main Weather_Description Temp_F Wind_MPH Apparent_Temperature2
## 1        Clear        sky is clear  48.29  4.47388              46.28407
## 2        Clear        sky is clear  48.29  4.47388              46.28407
## 3        Clear        sky is clear  48.29  4.47388              46.28407
## 4        Clear        sky is clear  48.29  4.47388              46.28407
## 5        Clear        sky is clear  48.29  4.47388              46.28407
## 6        Clear        sky is clear  48.29  4.47388              46.28407
```

```
duration2
```

```
## Time difference of 0.1928039 secs
```

```
# 5.b.3
start3 <- Sys.time()
clean_data_tidy_temp2 <- clean_data_tidy%>% mutate(Temp_F=((Temperature-273.15)*(9/5) + 32),Wind_MPH = 2.23694*Wi
nd_Speed) %>% filter(Temp_F<50 & Wind_MPH>3) %>% mutate(Apparent_Temperature3 = (35.47+(0.6215*Temp_F) - 35.75*(W
ind_MPH^0.16) + 0.4275*Temp_F*(Wind_MPH^0.16)))
end3 <- Sys.time()
duration3 <- end3-start3
head(clean_data_tidy_temp2)
```

```
##                  Date City_ID Temperature Humidity Wind_Speed Rain_1hr Snow_1hr
## 1 2012-10-10 23:00:00 2643743       282.2       85          2        0        0
## 2 2012-10-11 00:00:00 2643743       282.2       85          2        0        0
## 3 2012-10-11 01:00:00 2643743       282.2       85          2        0        0
## 4 2012-10-11 02:00:00 2643743       282.2       85          2        0        0
## 5 2012-10-11 03:00:00 2643743       282.2       85          2        0        0
## 6 2012-10-11 04:00:00 2643743       282.2       85          2        0        0
##   Weather_Main Weather_Description Temp_F Wind_MPH Apparent_Temperature3
## 1        Clear        sky is clear  48.29  4.47388              46.28407
## 2        Clear        sky is clear  48.29  4.47388              46.28407
## 3        Clear        sky is clear  48.29  4.47388              46.28407
## 4        Clear        sky is clear  48.29  4.47388              46.28407
## 5        Clear        sky is clear  48.29  4.47388              46.28407
## 6        Clear        sky is clear  48.29  4.47388              46.28407
```

```
duration3
```

```
## Time difference of 0.03351307 secs
```

Question 6: Assessing seasonal differences in weather

6.a: Make a data structure of the total raining hours for each city during each season in every year. Show the values in the data structure for 2013 for each city and each season.

6.b: Make a data structure of the total hours where the temperature is below 0 degrees Celsius for each city during each season in every year. Show the values in the data structure for 2013 for each city and each season.

Ans: Month, Year values were extracted from the date information and added as factor to the clean_data_tidy data frame. Seasons were arrived at through mutate() function and if-else conditions. The Rain_1hr and the Temperature values were then grouped by City_ID,Year and Season using the group_by() function. Summarise() was used to calculate the final hours where the rain and the temperature satisfied the conditions mentioned in the question.

To get 2013 values, filter was used by (year == 2013).

```
# 6.a & 6.b
t <- strptime(clean_data_tidy$Date, format= "%Y-%m-%d %H:%M:%S")
m <- as.numeric(format(t, "%m"))
y <- as.numeric(format(t, "%Y"))
clean_data_tidy$Month <- as.factor(m)
clean_data_tidy$Year <- as.factor(y)
clean_data_tidy <- clean_data_tidy %>% mutate(Season = ifelse(Month == 12 | Month == 1 | Month == 2, "Winter", if
else(Month == 3 | Month == 4 | Month == 5, "Spring", ifelse(Month == 6 | Month == 7 | Month == 8, "Summer", "Fal
l")))
rain_data <- clean_data_tidy %>% mutate(rain_binary = ifelse(Rain_1hr!= 0, 1, 0))
temp_data <- clean_data_tidy %>% mutate(temp_binary = ifelse(Temperature<273.15,1,0))

hours_of_rain <- rain_data %>% group_by(Year,City_ID,Season)%>%summarise(sum(rain_binary))
```

```
## `summarise()` regrouping output by 'Year', 'City_ID' (override with `.groups` argument)
```

```
display1 <- hours_of_rain%>%filter(Year==2013)
display1
```

```
## # A tibble: 24 x 4
## # Groups:   Year, City_ID [6]
##    Year  City_ID Season `sum(rain_binary)`
##    <fct> <fct>   <chr>                <dbl>
##  1 2013  2643743 Fall                    72
##  2 2013  2643743 Spring                   0
##  3 2013  2643743 Summer                  93
##  4 2013  2643743 Winter                  79
##  5 2013  2950159 Fall                    88
##  6 2013  2950159 Spring                   0
##  7 2013  2950159 Summer                  75
##  8 2013  2950159 Winter                  15
##  9 2013  2988507 Fall                    36
## 10 2013  2988507 Spring                   0
## # … with 14 more rows
```

```
hours_of_cold <- temp_data %>% group_by(Year,City_ID,Season)%>%summarise(sum(temp_binary))
```

```
## `summarise()` regrouping output by 'Year', 'City_ID' (override with `.groups` argument)
```

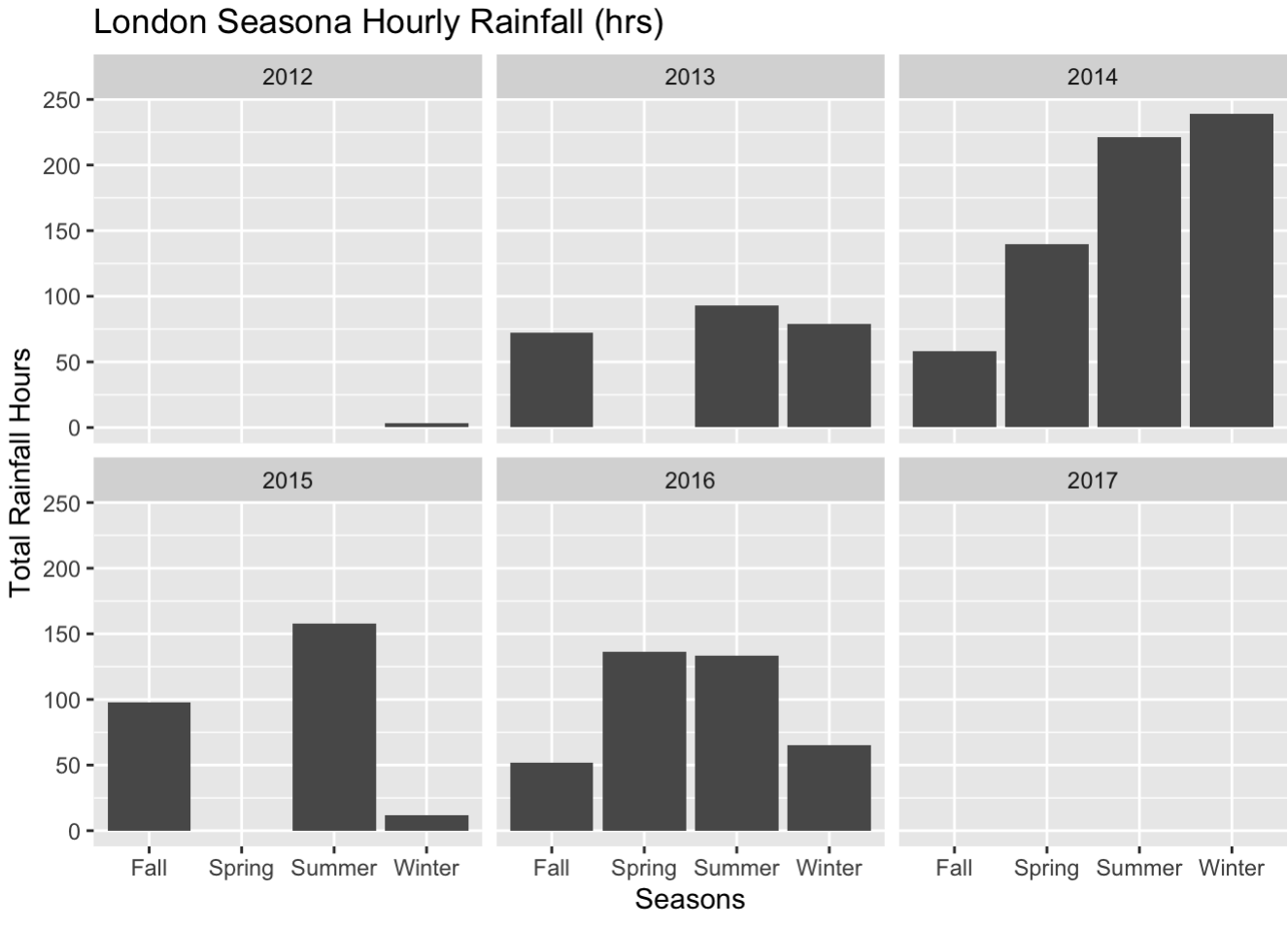
```
display2 <- hours_of_cold%>%filter(Year==2013)
display2
```

```
## # A tibble: 24 x 4
## # Groups:   Year, City_ID [6]
##    Year  City_ID Season `sum(temp_binary)`
##    <fct> <fct>   <chr>                <dbl>
##  1 2013  2643743 Fall                     3
##  2 2013  2643743 Spring                 104
##  3 2013  2643743 Summer                   0
##  4 2013  2643743 Winter                 177
##  5 2013  2950159 Fall                    38
##  6 2013  2950159 Spring                 409
##  7 2013  2950159 Summer                   0
##  8 2013  2950159 Winter                 653
##  9 2013  2988507 Fall                     0
## 10 2013  2988507 Spring                  68
## # … with 14 more rows
```

6.c: Use ggplot2 to create a bar graph of the number of raining hours for each season in the dataset for London (over all of the years). Be sure to label the x and y-axes and to create a title/caption for the graph. You do not need to use color for this plot.

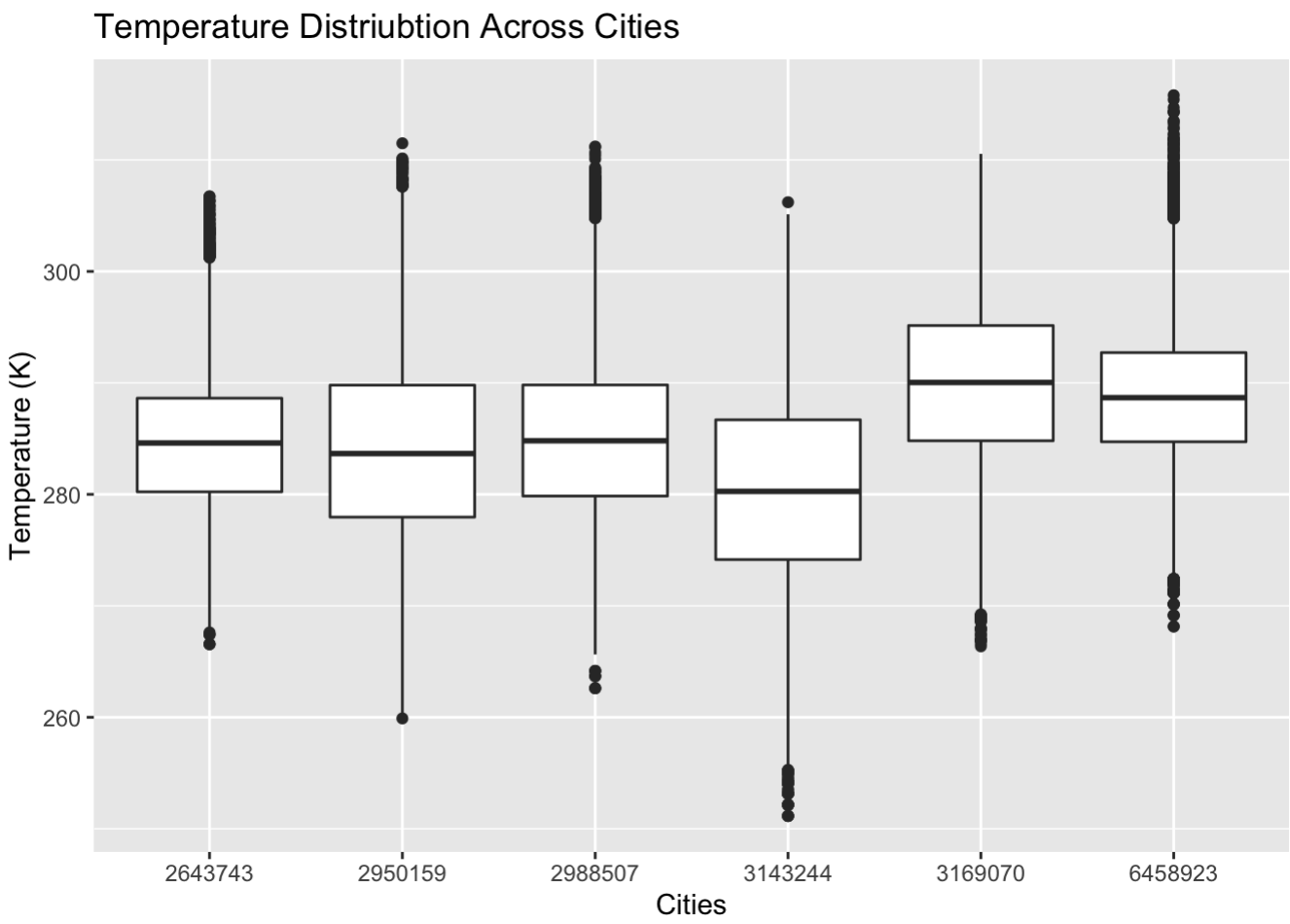
```
# 6.c
london_Rain_1hr <- hours_of_rain %>%filter(City_ID=="2643743")
colnames(london_Rain_1hr) <- c("Year","City_ID","Season","Hours_of_Rain")

season_rain <- ggplot(data = london_Rain_1hr, aes(x = london_Rain_1hr$Season, y = london_Rain_1hr$Hours_of_Rain
 ))+
  geom_col()+facet_wrap(~ Year) + ggtitle("London Seasona Hourly Rainfall (hrs)") + xlab("Seasons")+ylab("Total R
ainfall Hours")
season_rain
```

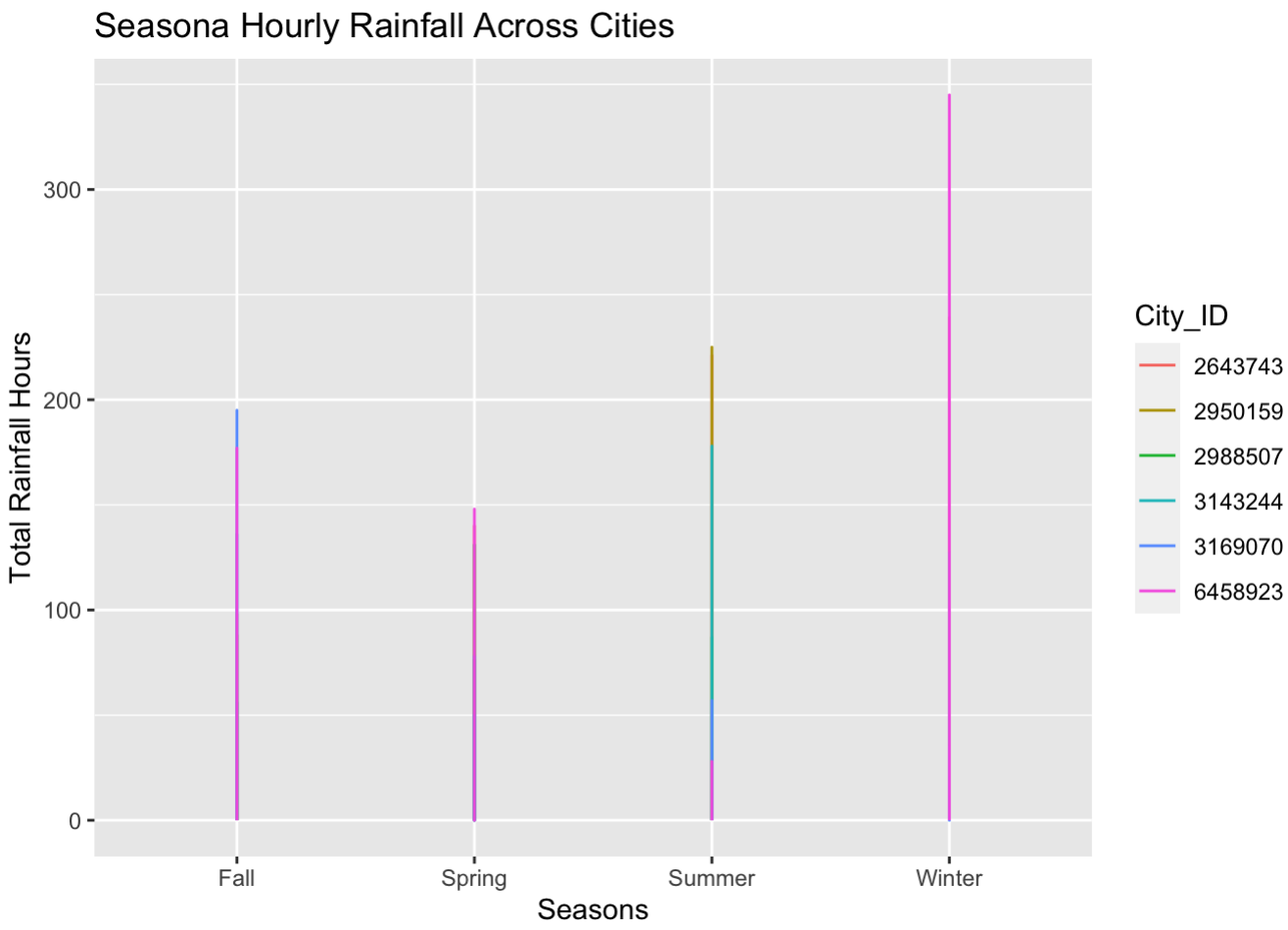


6.d: Use ggplot2 to make a single box plot of the temperatures for each of the six cities. Label each axis and provide a title. You do not need to use color. [To be clear, this is a single chart that contains 6 box plots.]

```
# 6.d
city_temp <- ggplot(data = clean_data_tidy, aes(x = City_ID, y = Temperature ))+
  geom_boxplot() + ggtitle("Temperature Distriubtion Across Cities") + xlab("Cities")+ylab("Temperature (K)")
city_temp
```

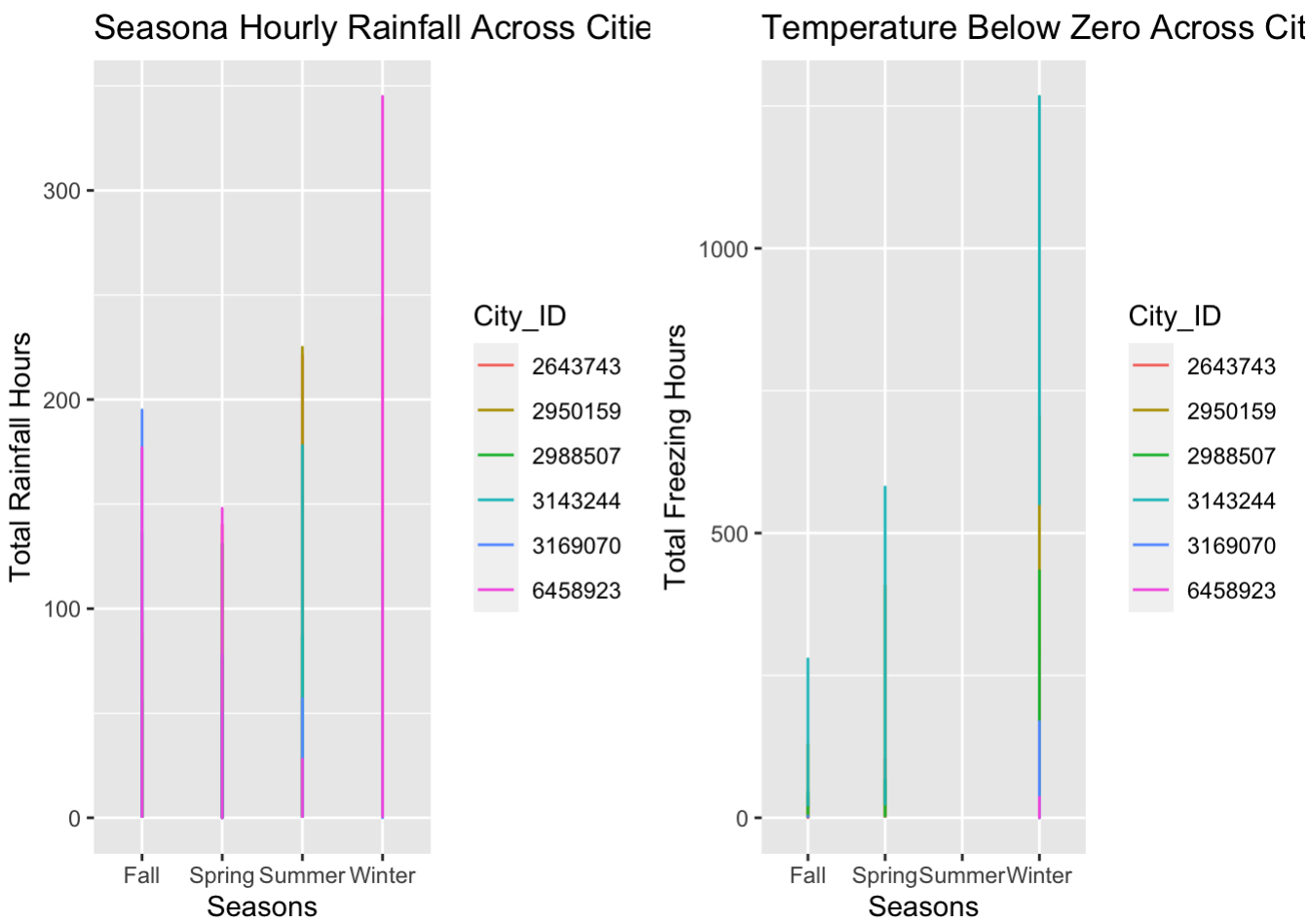



```
# 6.e
colnames(hours_of_rain) <- c("Year","City_ID","Season","Hours_of_Rain")
hours_of_rain <- hours_of_rain%>%group_by(City_ID,Season,)
season_rain <- ggplot(data = hours_of_rain, aes(x = Season, y = Hours_of_Rain ,color=City_ID))+
  geom_line()  + ggtitle("Seasona Hourly Rainfall Across Cities") + xlab("Seasons")+ylab("Total Rainfall Hours")
season_rain
```




```
# 6.f
colnames(hours_of_cold) <- c("Year","City_ID","Season","Freezing_Hours")
hours_of_cold <- hours_of_cold%>%group_by(City_ID,Season,)
season_cold <- ggplot(data = hours_of_cold, aes(x = Season, y = Freezing_Hours ,color=City_ID))+
  geom_line()  + ggtitle("Temperature Below Zero Across Cities (Hrs.)") + xlab("Seasons")+ylab("Total Freezing Ho
urs")

final_plot <- grid.arrange(season_rain, season_cold, nrow = 1, ncol = 2)
```




```
final_plot
```

```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

Question 7: Homework Assessment

Write a summary (about one page in length) that discusses what aspects of this assignment were easy and hard to do in R (and think about how long it would have taken to do similar things in Excel or Tableau). From your experience, in which situations would you want to keep using R and in which would you seek different tools?

Ans: R is very efficient with large datasets and cause no issue even with an average laptop, unlike Tableau and Excel which used to lag and hang constantly. RStudio is a very flexible platform and you have the ability to tailor your data analysis whichever way you want. Unlike in commercial packages, there any many ways, and increasingly efficient ways of doing the same task. The help is also very effective. The online community of data analysts and developers is also a great resource.

Although R is a very good tool, it is however slightly difficult to master in the beginning. The GUI of the other platforms makes it more user friendly in the beginning. But, the flexibility that R offers overcomes that disadvantage very quickly. R is an open source, free resource which is diametrically opposite to the expensive toolpacks.

The difficult part of the homework was getting acquainted with Base R and tidyverse syntax. Being a new user, it was difficult and very time consuming over all. There was also a stark transition from Base R to tidyverse, making it all the more difficult to cope up.