Where do my 1440 minutes go: An analysis of how I spend my time

STATS 231: Calendar Query

Lynca Jones Kaminka

Mar 2th, 2023

Academic Honesty:

a. Attempting to beautify ggplot2 was done thanks to: https://www.cedricscherer.com/2019/08/05/a-ggplot2-tutorial-for-beautiful-plotting-in-r/
b. Chapter 10 of the textbook (10.2. The kableExtra package) helped me with the layout of the table.

Introduction

My Questions of interests were:

1. Does my phone time usage fluctuates throughout the 14 days depending on when I am the most stressed (which is typically Wednesdays and Fridays) and does it increase during the weekends when I don't have much going on. This is relevant because I believe that I have enough time on my hands to be great at most things I do and one of the things that are holding me back is my inability to manage my time effectively.

2. How the time I spent doing specific activities like studying for classes, going to office hours, doing my personal work and sleeping has changed in the last 14 days compared to each other. It is relevant because I want to understand where I should invest more time.

3. To what extent (in percentage) am I following the weekly schedule that I have established for myself. It can help me become more aware of ways I can adjust how I spend my time. A little caveat, this question had to be revised because when I was extracting the data set from my time-tracking app, I realized that the time when I start and end the activities I was doing was not imported despite being recorded. Thus, I will only be comparing the expected duration and actual direction of my weekly activities).

Methods

Data collection

I used 3 raw data sets from: - My Google calendar: it contains how I am plan to spend my days; it is the same from one week to the other. Since the data corresponds to my weekly routine, I manually entered in those activities.I retrieved the data by importing the Google calendar data set into R.

- An app called StayFree: since this is an app that tracks the daily usage time of every app on my smartphone,I didn't have to actively collect any data except for importing the resulting data set in R.

- An app called Clockify (I used it as a timer). I activated and deactivated a task on it before and after each activity I intended to analyze. As was the case with the other tree data sets, I retrieved the data by importing the data sets. I was unable to import a single data set since the app calculates the time spent doing an activity in a period of time by combining the time I spent doing that activity during that whole time. it would have lost the day-to-day specificity if I had chosen to import the data over a long period of time.

Data wrangling

1. For the first visualization, I imported several data sets (14 to be exact) from Clockify. Since Clockify only allows the users to see the duration of their activities and not the start or end of each activity entered, I didn't have to infer the start and end time of each activity. The next step was to retrieve the specific information that were relevant to the data visualization I wanted (name of the activities, the duration, the day when it happened,..), and I merged all the day specific data sets. After that, the next step was to convert times to numeric values since they were originally expressed as characters. And then, I did some more data set manipulating to get the data set in the form that would get me to the display I wanted to go to. The last step in data wrangling was to keep the activities of interest in the data set
2. For the second visualization, I imported a single data set from StayFree. I also had to go through the process of converting the type the recorded times were in (which were characters) into numeric values. Also, since each time an app is opened, a unique record of the time spent on it is created, I had multiple time records that corresponded to each time an app was used, even on the same day. Thus, I had to find the daily time I spent on each application. I also had to filter out the features related to the time observation that I didn't need. Lastly, since I have several apps, I collapsed some of them into categories that are; social media (for apps like Twitter, Instagram, Whatsapp), Gmail_Youtube (which I use more for informative reason), and the rest of other apps.
3. For the third visualization where I sought to show the difference between how I spent each of the 14 days on average versus how I originally planned to spend it, I had to use both my Calendar and my time-tracking app. The biggest challenge was bringing those two data sets together since I hadn't preemptively made sure that some features (like names given to observations) were consistent across the two data sets.

1. Data Wrangling for the first visualization:

```
# Data import form clockify:
###############
Day_1 <- read_csv("Data/Day1.csv")
Day_2 <- read_csv("Data/Day2.csv")
Day_3 <- read_csv("Data/Day3.csv")
Day_4 <- read_csv("Data/Day4.csv")
Day_5 <- read_csv("Data/Day_5.csv")
Day_6 <- read_csv("Data/Day_6.csv")
Day_7 <- read_csv("Data/Day_7.csv")
Day_8 <- read_csv("Data/Day_8.csv")
Day_9 <- read_csv("Data/Day_9.csv")
Day_10 <- read_csv("Data/Day_10.csv")
Day_11 <- read_csv("Data/Day_11.csv")
Day_12 <- read_csv("Data/Day_12.csv")
Day_13 <- read_csv("Data/Day_13.csv")
Day_14 <- read_csv("Data/Day_14.csv")
```

```r
Day_1_a <- mutate(select(Day_1,Project,  "Time (h)" ), Day = "Day 1")


Day_2_a <- mutate(select(Day_2,Project,  "Time (h)" ), Day = "Day 2")
Day_3_a <- mutate(select(Day_3,Project,  "Time (h)" ), Day = "Day 3")
Day_4_a <- mutate(select(Day_4,Project,  "Time (h)" ), Day = "Day 4")
Day_5_a <- mutate(select(Day_5,Project,  "Time (h)" ), Day = "Day 5")
Day_6_a <- mutate(select(Day_6,Project,  "Time (h)" ), Day = "Day 6")
Day_7_a<- mutate(select(Day_7,Project,  "Time (h)" ), Day = "Day 7")
Day_8_a <- mutate(select(Day_8,Project, "Time (h)" ), Day = "Day 8")
Day_9_a <- mutate(select(Day_9,Project,  "Time (h)" ), Day = "Day 9")
Day_10_a <- mutate(select(Day_10,Project,  "Time (h)" ), Day = "Day 10")
Day_11_a <- mutate(select(Day_11,Project,  "Time (h)" ), Day = "Day 11")
Day_12_a <- mutate(select(Day_12,Project,  "Time (h)" ), Day = "Day 12")
Day_13_a <- mutate(select(Day_13,Project,  "Time (h)" ), Day = "Day 13")
Day_14_a <- mutate(select(Day_14,Project,  "Time (h)" ), Day = "Day 14")




All_Days <- bind_rows(Day_1_a, Day_2_a,Day_3_a, Day_4_a,Day_5_a, Day_6_a, Day_7_a, Day_8_a,Day_9_a, Day


 #how to group these by days and projects so that I can avoid having duplicate values inside my data se


 All_Days_1 <- All_Days %>%
   mutate(
     "Time" = as.numeric(as_datetime(`Time (h)`, "HH:MM:SS"))
   )
```

```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘Time = as.numeric(as_datetime(‘Time (h)‘, "HH:MM:SS"))‘.
## Caused by warning in ‘with_tz.default()‘:
## ! Unrecognized time zone ’HH:MM:SS’
```

```r
All_Days_2 <- All_Days_1 %>%
  filter(Project != "(Without project)")


 All_Days_3 <- All_Days_2 %>%
   select(c("Project", "Day", "Time")) %>%
   mutate("Time" = as.integer(`Time`)) %>%
   group_by(Day, Project) %>%
   summarise(
     Total_time = sum(`Time`)
   )



All_Days_4<- All_Days_3 %>%
  pivot_wider(names_from = "Project", values_from = "Total_time")
```

And then, because on some days there are activities that I didn't do, R tends to fill in the empty observation with "N/A", I am changing it the cells that contain the "N/A" tags into numeric values.

```
 for (i in 1:nrow(All_Days_4)) {
  for (j in 1:ncol(All_Days_4)) {
    if (is.na(All_Days_4[i, j])) {
      All_Days_4[i, j] = 0
    }
  }
}
```

Last wrangling step; filtering out the activities that I do not want to analyze.

```
All_Days_5 <- All_Days_4 %>%
separate(Day, into = c("Noun", "Day"), sep = " ", convert = TRUE)

All_Days_6 <- arrange(select(All_Days_5, !Noun), Day)

All_Days_7 <- All_Days_6 %>%
 select(Day, "Bedtime/Sleep", "Work shift", "Office Hours", "Personal work", "Classes", "School work" )

All_Days_8 <- All_Days_7 %>%
    pivot_longer(cols=-Day, names_to="Activities", values_to="Daily_time_Spent")
```

2.Second Wrangling: for the data that was recorded by the Screen Time Recorder App.

Step1: Importing the data set, getting rid of unneccesary features.

```
#Data set 2
StayFree <- read_csv("Data/StayFreeExport.csv")


StayFree_2 <- select(mutate( StayFree, app = ...1  ), ! ...1 & !Device) %>%
    pivot_longer(cols=-app, names_to="Date", values_to="time_char")
```

Step2: Converting the screen time from a character data type to a numeric data type.

```
StayFree_3 <- StayFree_2 %>%
  separate(time_char, into=c("time1", "time2", "time3"), sep=" ", remove=FALSE) %>%
  mutate(
    hours = case_when(str_detect(time1, "h") ~parse_number(time1)
                    , TRUE ~ 0),
    minutes = case_when(str_detect(time1, "m") ~ parse_number(time1)
                        , str_detect(time2, "m") ~ parse_number(time2)
                             , TRUE ~ 0)
         , seconds = case_when(str_detect(time1, "s") ~ parse_number(time1)
                             , str_detect(time2, "s") ~ parse_number(time2)
                             , str_detect(time3, "s") ~ parse_number(time3)
                             , TRUE ~ 0)
         , time_seconds = minutes*60 + seconds + hours*360
         ,  time_minutes = minutes + seconds/60 + hours*60
         ,  time_hours = hours + minutes/60 + seconds/360)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with 'NA' in 917 rows [1, 2, 3, 4, 5,
## 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

Step3: Deriving the observations of interests by a) renaming columns, ordering the days and arranging the data set in such a way that all apps and their associated usage times can be derived and manipulated separately.

```
#names(StayFree_3)

StayFree_4 <- StayFree_3 %>%
  select(app, Date, "time_seconds") %>%
  filter(!app == "Settings" )%>%
  pivot_wider(names_from = "Date", values_from = "time_seconds")


 names(StayFree_4) <- c ("Apps", "Day 1", "Day 2", "Day 3", "Day 4", "Day 5", "Day 6", "Day 7", "Day 8"


# View(StayFreeModified)

StayFree_5 <- StayFree_4%>%
  pivot_longer(cols=-Apps, names_to="Days", values_to="Time_T")%>%
  filter(Days != "TotalUsage")
  ##Grouping the amount of length of an app by days and apps.
  StayFree_6 <- StayFree_5 %>%
   group_by(Days, Apps) %>%
   summarise(
     Total_time = sum(Time_T)
   )
 #View(StayFree_7)


  StayFree_7 <- StayFree_6 %>%
  separate(Days, into = c("Noun", "Day"), sep = " ", convert = TRUE)

 StayFree_8<- arrange(select(StayFree_7, !Noun), Day)
```

Step4: Since I want to compare the times I spent on different app categories, I will have to first isolate the apps that belong to the same category. The categories are social media apps, Youtube and Gmail and all the other apps. Once apps have been regrouped under the right categories, I can plot the graph that shows the difference of usage between those three different app categories.

```
 ############
Social_Media_Apps <- StayFree_8 %>%
   filter(Apps == "Instagram" |
          Apps == "Twitter" |
           Apps == "Snapchat"|
           Apps == "Zara"|
           Apps =="Whatsapp" |
           Apps == "YouTube" |
           Apps == "Netflix"
           )
   Social_Media_Apps_2 <- Social_Media_Apps %>%
   group_by(Day) %>%
   summarise(
     Total_time_2 = sum(Total_time)
   )


#############

OtherApps <- StayFree_8 %>%
  filter(Apps != "Instagram" &
          Apps != "Twitter" &
           Apps != "Snapchat"&
           Apps != "Zara"&
           Apps !="Whatsapp" &
           Apps != "YouTube" &
           Apps != "Gmail" &
           Apps != "Netflix"
           )

  OtherApps_2 <- OtherApps %>%
      group_by(Day) %>%
      summarise(
      Total_time_2 = sum(Total_time)
 )

 ######################
 Youtube_Gmail <- StayFree_8 %>%
     filter( Apps == "YouTube" |
             Apps ==  "Gmail"
         )
      Youtube_Gmail_2  <- Youtube_Gmail %>%
      group_by(Day) %>%
      summarise(
      Total_time_2 = sum(Total_time)
 )

     #View(OtherApps_2)
     #View(Social_Media_Apps_2)
     #View(Youtube_Gmail_2)
```

Step5: Creation of the last dataset which will gather the duration of use of different application categories during the last 14 days

```
 Final_Result <- OtherApps_2 %>%
   full_join(Social_Media_Apps_2, by="Day")

 Final_Result_1 <- Final_Result %>%
   full_join(Youtube_Gmail_2, by="Day")

  names(Final_Result_1) <- c ("Day", "Other Apps", "Social Media Apps", "Youtube_Gmail")



Final_Result_2 <- Final_Result_1 %>%
    pivot_longer(cols=-Day, names_to="Type_of_app", values_to="Daily_time_spent")
#View(Final_Result_2)
```

Last step: Plotting the graph of interest ==> in the other section.

3. Data wrangling for the third data set, that was used to plot the table.

```r
# Data import (requires **ical** package)
cal_import <- ical_parse_df("Data/lkaminka25@amherst.edu.ics")

# Data wrangling
mycal <-
  cal_import %>%
  # Google Calendar event names are in a variable called "summary";
  # "activity" is a more relevant/informative variable name.
  rename(activity = summary) %>%
  mutate(
    # Specify time zone (defaults to UTC otherwise)
    across(c(start, end),
           .fns = with_tz,
           tzone = "America/New_York"),
    # Compute duration of each activity in hours
    duration_hours = interval(start, end) / hours(1),
    # Examples of getting components of dates/times
    # Note:
    # i. these could be based on either start datetime or end datetime
    # ii. you do NOT need all of these!! so only use what you need
    date = date(start),
    year = year(start),
    month_number = month(start),
    month_label = month(start,
                        label = TRUE,
                        abbr = FALSE),
    weekday_number = wday(start),
    weekday_label = wday(start,
                         label = TRUE,
                         abbr = FALSE),
    hour = hour(start),
    time = hour(start) + minute(start)/60,
    # Convert text to lowercase and remove repeated or leading/trailing
    # spaces to help clean up inconsistent formatting.
    across(c(activity, description),
           .fns = str_to_lower),
    across(c(activity, description),
           .fns = str_squish)
  ) %>%
  # The first Google Calendar entry is always an empty 1969 event
  filter(year == 2023)
```

```
## Warning: There was 1 warning in ‘mutate()‘.
## i In argument: ‘across(c(start, end), .fns = with_tz, tzone =
##   "America/New_York")‘.
## Caused by warning:
## ! The ‘...‘ argument of ‘across()‘ is deprecated as of dplyr 1.1.0.
## Supply arguments directly to ‘.fns‘ through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
```

```
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
My_Calendar<- mycal %>%
  select("activity","date", "month_number", "duration_hours", "start", "end", "hour", "time" ) %>%
  filter(month_number == 2) %>%
  separate(date, into = c("Year", "Month", "Day"), sep = "-", convert = TRUE) %>%
  arrange(Day) %>%
  filter(!Day < 14 & !Day > 20) %>%
  select("activity", "Day", "duration_hours")


My_Calendar_2 <- My_Calendar %>%
     group_by(Day, activity) %>%
     summarise(
     Total_time_hours = sum(`duration_hours`)
   )




#View(My_Calendar_2)
```

```
#duplicating the values of the days since my schedule isn't really day specific.
#And then, because on some days there are activities that I didn't do, R tends to fill in the empty obs



My_Calendar_3<- My_Calendar_2 %>%
  pivot_wider(names_from = Day, values_from = Total_time_hours)

 for (i in 1:nrow(My_Calendar_3)) {
  for (j in 1:ncol(My_Calendar_3)) {
    if (is.na(My_Calendar_3[i, j])) {
      My_Calendar_3[i, j] = 0
    }
  }
}


 names(My_Calendar_3) <- c ("Activities", "Day 1", "Day 2", "Day 3", "Day 4", "Day 5", "Day 6", "Day 7")
 #View(My_Calendar_3)
 My_Calendar_4 <- My_Calendar_3
 names(My_Calendar_4) <- c ("Activities", "Day 8", "Day 9", "Day 10", "Day 11", "Day 12", "Day 13", "Da


 My_Calendar_5 <- My_Calendar_4 %>%
 full_join(My_Calendar_3, by="Activities")

 #View(My_Calendar_5)

 My_Calendar_6 <- My_Calendar_5%>%
    pivot_longer(cols=-Activities, names_to="Days", values_to="Daily_time_Spent_h")%>%
    separate(Days, into = c("Noun", "Day"), sep = " ", convert = TRUE) %>%
    select(Activities, Day, Daily_time_Spent_h )
```

```
#View(My_Calendar_6)

My_Calendar_7 <- My_Calendar_6 %>%
   group_by( Activities) %>%
   summarise(
    Average_Daily_Time_Spent_h = mean(`Daily_time_Spent_h`)
  ) %>%
  filter(Activities != "fashion show meeting" &
         Activities ==  "bedtime/sleep" |
         Activities == "work shift" |
         Activities == "office hours" |
         Activities == "personal work"|
         Activities == "classes" |
         Activities == "school work" |
         Activities == "clubs"  |
         Activities == "socializing")


  All_Days_7_b <- All_Days_6 %>%
  select(Day, "Bedtime/Sleep", "Work shift", "Office Hours", "Personal work", "Classes", "School work"

  View(All_Days_7_b)

  names(All_Days_7_b) <- c ("Day", "bedtime/sleep", "work shift", "office hours", "personal work", "cla
  All_Days_8_b <- All_Days_7_b %>%
     pivot_longer(cols=-Day, names_to="Activities", values_to="Daily_time_Spent")%>%
     group_by(Activities) %>%
     summarise(
     Average_Daily_Time_Spent_s  = mean(Daily_time_Spent)
    )

 All_Days_9_b <- All_Days_8_b %>%
   mutate(Average_Daily_Time_Spent_h = Average_Daily_Time_Spent_s/3600)%>%
   select(Activities, Average_Daily_Time_Spent_h)
```

Last wrangling step; bringing the data sets together

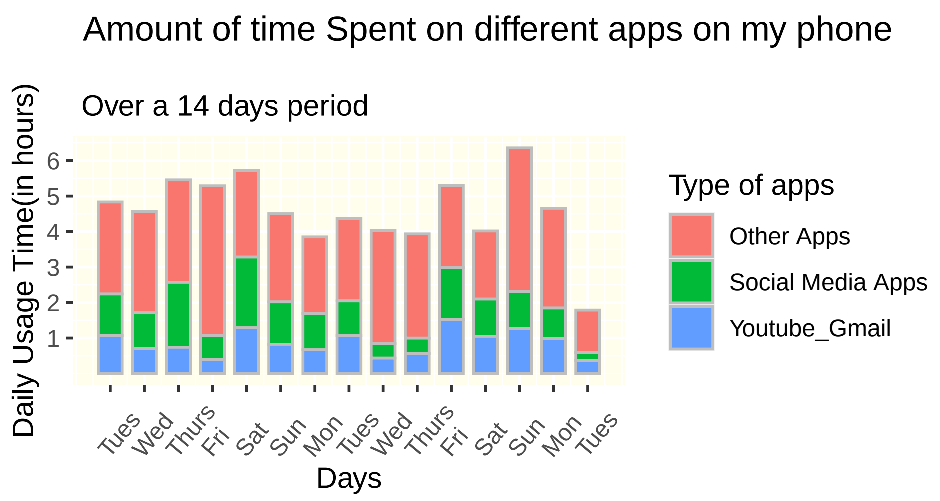
```
Table <- My_Calendar_7 %>%
  full_join(All_Days_9_b, by = "Activities")

 names(Table) <- c ("Activities", "Duration_Planned", "Actual_Duration")

Table_1 <- Table %>%
  mutate(Consistency = (Actual_Duration/Duration_Planned)*100)
View(Table_1)
```

Visualizations and Tables

Visualization 1: Self-reported time spent on selected activities (represented via a bar chart)

```r
ggplot(Final_Result_2, aes(fill = Type_of_app, y=(Daily_time_spent)/3600, x=Day)) +
    geom_bar(position="stack", color = "grey", stat="identity", width=0.7) +
    scale_x_continuous(breaks=c(1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15),
                       labels=c("Tues","Wed","Thurs","Fri","Sat", "Sun",
                           "Mon","Tues","Wed","Thurs","Fri","Sat", "Sun",
                           "Mon", "Tues")) +
    scale_y_continuous(breaks=c(1,2, 3, 4, 5, 6)) +
    labs(x="Days", y="Daily Usage Time(in hours)",
         title="\n Amount of time Spent on different apps on my phone"
        ,subtitle="\n Over a 14 days period"
        ,fill= "Type of apps")  +
    theme(panel.background = element_rect(
         fill = "#FFFEEC", color = "#FFFEEC", size = 2)
          ) +
    theme(axis.text.x = element_text(angle = 50, vjust = 0.05, hjust=0.05))+
    theme(aspect.ratio=18/40)
```

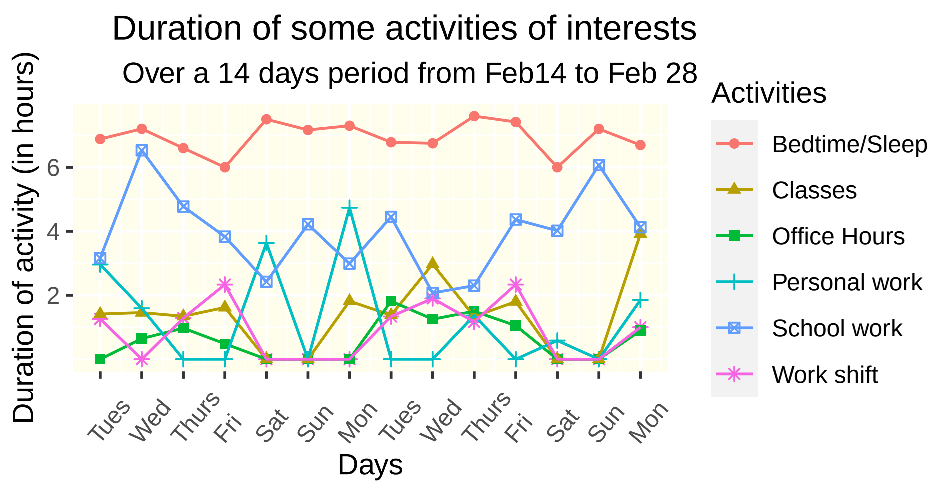
```
## Warning: The ‘size‘ argument of ‘element_rect()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



Visualization 2: Self-reported time spent on selected activities (represented via a line graph).

```r
ggplot(All_Days_8, aes( x=Day, y= Daily_time_Spent/3600, color = Activities)) +
    geom_point(aes(shape = Activities))+
    geom_line() +
    scale_x_continuous(breaks=c(1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14),
                       labels=c("Tues","Wed","Thurs","Fri","Sat", "Sun",
                            "Mon","Tues","Wed","Thurs","Fri","Sat", "Sun",
                            "Mon")) +
    scale_y_continuous(breaks=c(2,4, 6, 8 ,10, 12, 14, 16),
                       ) +
    labs(x="Days", y="Duration of activity (in hours)",
         title="\t \t Duration of some activities of interests"
        ,subtitle="\t \t \t Over a 14 days period from Feb14 to Feb 28")+

    theme(panel.background = element_rect(
         fill = "#FFFEEC", color = "#FFFEEC", size = 2),
          ) +
    theme(axis.text.x = element_text(angle = 50, vjust = 0.05, hjust=0.05))+
    theme(aspect.ratio=18/40)
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font width unknown for character 0x9
```



Visualization 3: Table that compares with the amount of the length of some of my activities over the last 14 days, and the amount I planned to spend doing those activities.

```r
kable(head(Table_1, 8), align = 'c', booktabs = TRUE) %>%
  row_spec(1,italic = TRUE, color = '#2E3840', background = '#F7C8E0' ) %>%
  row_spec(2,italic = TRUE, color = '#2E3840', background = '#DFFFD8' ) %>%
  row_spec(3,italic = TRUE, color = '#2E3840', background = '#B4E4FF' ) %>%
```

Difference between how I plan vs how I actually spend my days			
Activities	Duration_Planned	Actual_Duration	Consistency
bedtime/sleep	*7.6666667*	*6.9353571*	*90.46118*
classes	*1.6904762*	*1.3604762*	*80.47887*
clubs	*0.5000000*	*0.2940476*	*58.80952*
office hours	*1.2142857*	*0.6149206*	*50.64052*
personal work	*1.2142857*	*1.1946429*	*98.38235*
school work	*3.5476190*	*3.9514484*	*111.38311*
socializing	*0.8928571*	*0.6226190*	*69.73333*
work shift	*0.8571429*	*0.9009127*	*105.10648*

```
  row_spec(4,italic = TRUE, color = '#2E3840', background = '#95BDFF') %>%
  row_spec(5,italic = TRUE, color = '#2E3840', background = '#FFADBC')  %>%
  row_spec(6,italic = TRUE, color = '#2E3840', background = '#863A6F')  %>%
  row_spec(7,italic = TRUE, color = '#2E3840', background = '#975C8D')  %>%
    row_spec(8,italic = TRUE, color = '#2E3840', background = '#D989B5')  %>%
  add_header_above(c("Difference between how I plan vs how I actually spend my days" = 4))%>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

Results

The first visualization shows that I spent almost five hours using various apps on my phone. There doesn't seem to be much difference between weekdays and weekends in general, except for a certain Sunday when I used my phone for almost 7 hours.

On the second visualization, it is clear that there are dramatic increases and decreases in duration for the majority of the activities throughout the 14 days period. I truly believe that if I were consistent in how I spend my days, I would avoid sleeping for less than 6 hours on some days.

The table shows that I don't really follow my schedule to perfection but the difference doesn't seem that huge. There are areas like my involvement in clubs, hanging out with friends, going to office hours that are still works in progress for me.

Conclusions

Besides the fact that this is a class project, I specifically chose these questions because I know I need to work on my time management so that I don't feel overwhelmed for most weeks; which will definitely improve my quality of life. The biggest insights (and especially the hardest to process) were how much I am on my phone and how I can be consistent from a day to another when it comes to things like going to office hours, spending time with my friends, getting enough sleep. I hope this new awareness will help me plan and feel more motivated to stick to my schedule.

Reflection

Two of the biggest difficulties I encountered in the data collection and analysis process were forgetfulness and the inability to analyze what I set out to study because the data could not be derived as I wished. I realized that forgetfulness is not something I can overcome all the time, and sometimes bigger priorities can get in the way of collecting accurate data. Second, I ended up not answering my original main questions of interest since when I tried exporting the data from the time-tracking app, I realized that details like when I had started an activity, when I had finished, were not recorded. As a result, my analysis had to be exclusively about the duration of the activities I was doing.

As a result of seeing how self-reporting while analyzing the reports is error prone, for any future projects, I plan on not endeavoring to analyze something that needs to be constantly checked/updated by me. Besides that, I learned a great deal about pre-testing the tools one uses to save the data, as you may not be able to retrieve the specific information you wanted to export. Also, I believe that since most of my questions had as goal analyzing how I spend my days, I don't believe that collecting more data could have been helpful in any way.

To the question of what expectations I have when I give my data, I believe that we are all coerced into giving our data to tech giants. Therefore, I don't think I have a choice of giving my data or any expectations when I do it because I do not choose to do it. However,I definitely wish data was analyzed en masse in order to discover trends in populations instead of using it to build algorithms that can predict what I as user will do next.

As someone who may one day analyze other people's data, I believe I will strive to refrain from using data in ways that the person who gave their data did not consent to. Besides, let people who "give up" their data know that it's happening, and give me them alternatives of using the technologies they are using without giving up their personal data (as opposed to what normally happens where if you don't want Google to track you then you can't use any of its services or affiliated services).