WH DataSciR Course Peer Project 1

Wanda Hernandez

2024-01-24

```
#-------------------- LOAD PACKAGES  --------------------

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.3     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.4     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(dplyr)

library(ggplot2)
```

```
# Set the directory:
setwd("~/Wanda/Coursera/Data Science Foundations Using R Specialization/5 Reproducible Research/
WK2 Coding Standards in R")
```

```
#From the Files window, located CSV file, clicked on it and selected IMPORT DATASET.#Copied and
pasted code provided to import dataset.
#Run code and dataset will appear in global environment


library(readr)
activity <- read_csv("activity.csv")
```

```
## Rows: 17568 Columns: 3
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## dbl  (2): steps, interval
## date (1): date
##
## 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.
```

```
View(activity)
```

```
#------------------- EXPLORE DATA  --------------------

dim(activity)
```

```
## [1] 17568     3
```

```
names(activity)
```

```
## [1] "steps"    "date"     "interval"
```

```
#The dim or dimension function  tells us that there are over 17000 observations(rows) and 3 vari
ables(columns).
#The names function tells us that the variables are  steps, date and intervals.
```

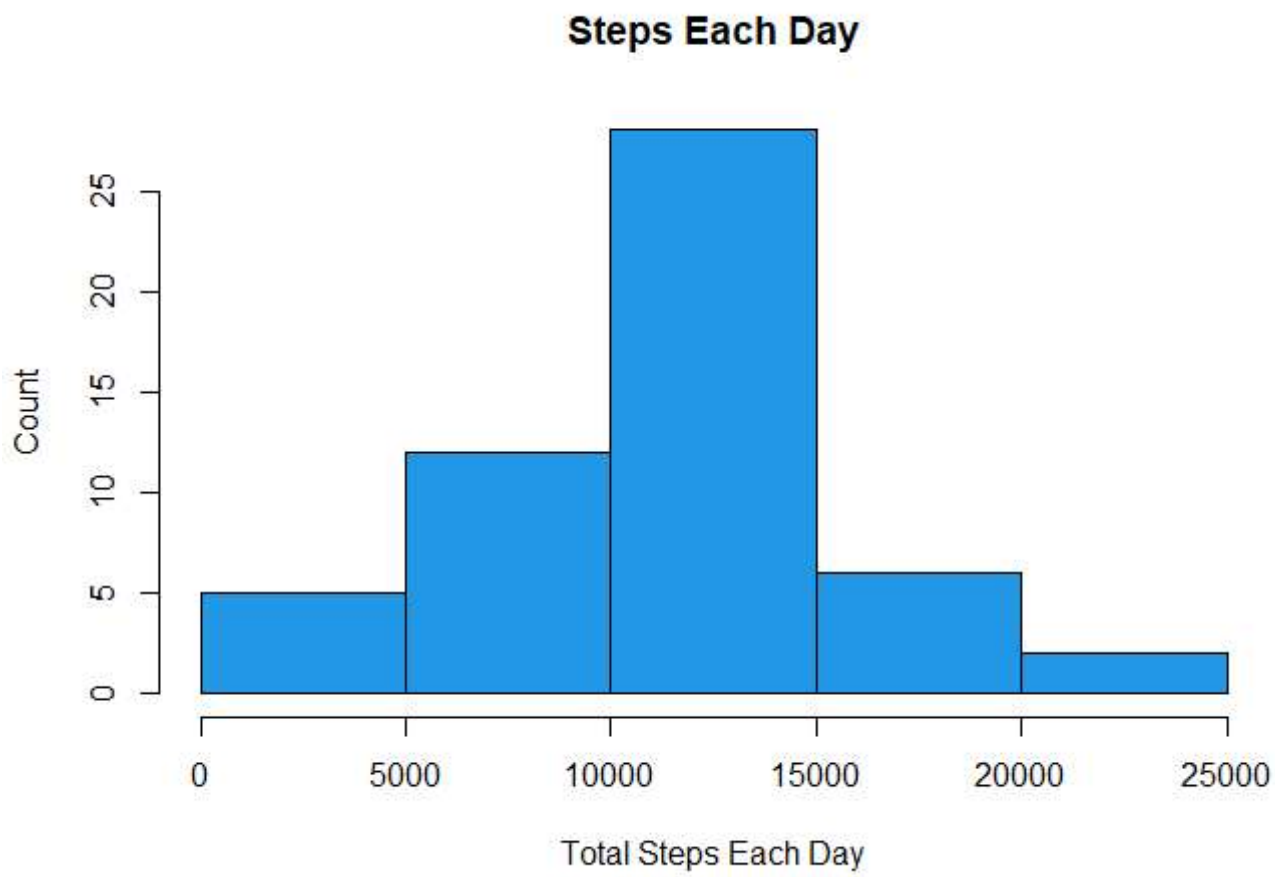
```
#Glimpse function gives mea summary of data. I can see the number rows(observations) and columns
(variables)
#I see that the steps variable has many missing values noted as NA.
glimpse(activity)
```

```
## Rows: 17,568
## Columns: 3
## $ steps    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ date     <date> 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01, …
## $ interval <dbl> 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 100, 105, 110, …
```

```r
# -------------------- EXPLORATORY DATA ANALYSIS: COURSE QUESTIONS --------------------

# -------------------- WHAT IS MEAN TOTAL NUMBER OF STEPS TAKEN PER DAY? --------------------
# 1. Calculate the total number of steps taken per day

# Group data by date, and summarize the sum of steps. Can leave out Nas.

StepsEachDay <- activity %>%
    drop_na(steps) %>%
    group_by(date) %>%
    summarise("TotalSteps" = sum(steps), .groups = "keep")

print(StepsEachDay)
```

```
## # A tibble: 53 × 2
## # Groups:   date [53]
##    date       TotalSteps
##    <date>          <dbl>
##  1 2012-10-02        126
##  2 2012-10-03      11352
##  3 2012-10-04      12116
##  4 2012-10-05      13294
##  5 2012-10-06      15420
##  6 2012-10-07      11015
##  7 2012-10-09      12811
##  8 2012-10-10       9900
##  9 2012-10-11      10304
## 10 2012-10-12      17382
## # ℹ 43 more rows
```

```r
# 2.  Make a histogram of the total number of steps taken each day

hist(StepsEachDay$TotalSteps,
     xlab="Total Steps Each Day",
     ylab="Count",
     main="Steps Each Day", col = 4)
```



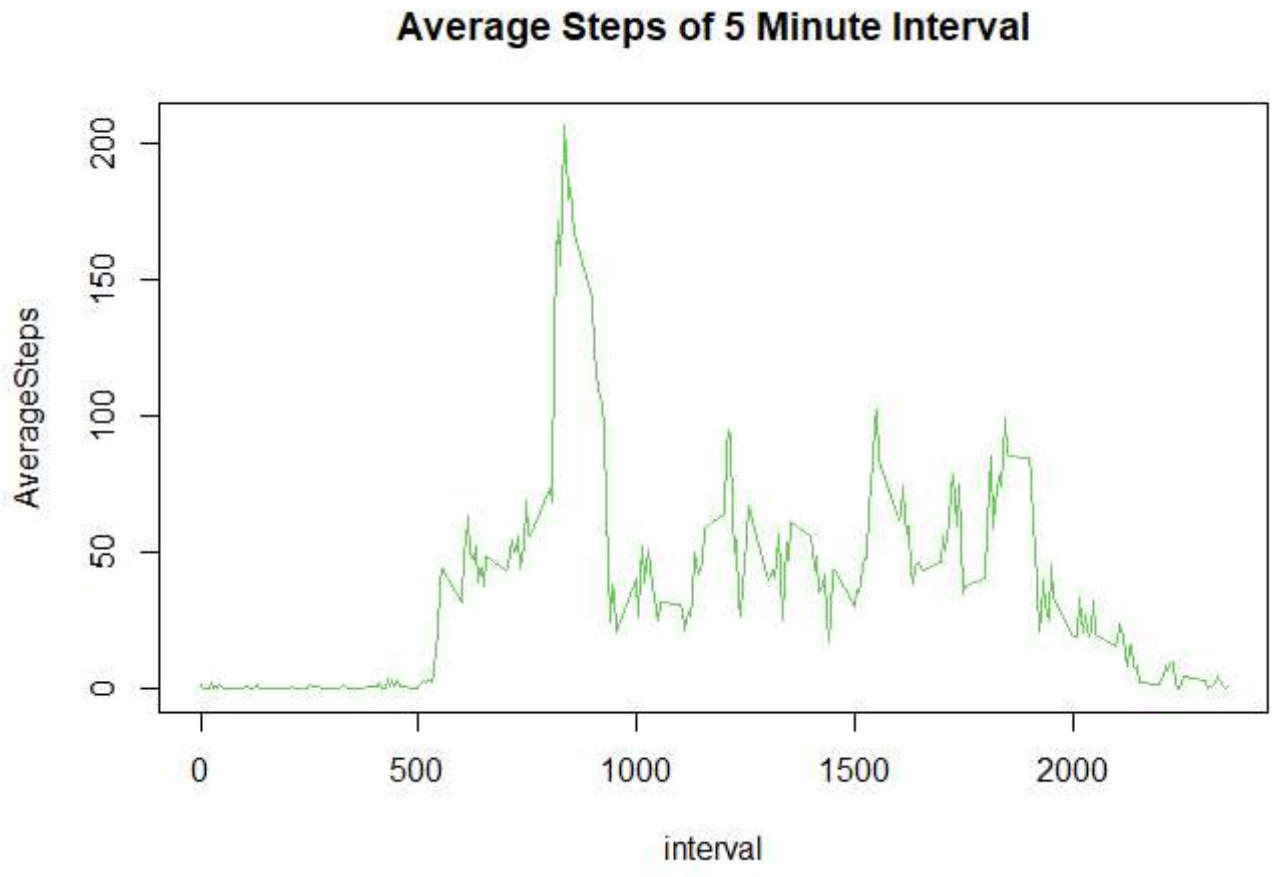

```
# 3.  Calculate and report the mean and median of the total number of steps taken per day

AverageSteps <- mean(StepsEachDay$TotalSteps)
medianSteps <- median(StepsEachDay$TotalSteps)
print(AverageSteps)
```

```
## [1] 10766.19
```

```
print(medianSteps)
```

```
## [1] 10765
```

```
# --------------------what is average daily activity pattern? ---------------------

# 1. Make a time series plot (i.e. type = "l") of the 5-minute interval (x-axis) and the average
number of steps taken, averaged across all days (y-axis)
#type = "l" for line graph

#Must group data and calculate average in summarize
FiveMinAvg <- activity %>%
    drop_na(steps) %>%
    group_by(interval) %>%
    summarize(AverageSteps=mean(steps), .groups="keep")

plot(FiveMinAvg,  type="l", main = "Average Steps of 5 Minute Interval", col = 3)
```




```
# 2.  Which 5-minute interval, on average across all the days in the dataset, contains the maxim
um number of steps?
#used which max function that finds the maximum value and its index.

MaxSteps <- FiveMinAvg$interval[which.max(FiveMinAvg$AverageSteps)]
print(paste("The maximum number of steps on average across all the days is", MaxSteps))
```

```
## [1] "The maximum number of steps on average across all the days is 835"
```

```
# -------------------- Imputing missing values --------------------

#Note that there are a number of days/intervals where there are missing values (coded as NA). Th
e presence of missing days may introduce bias into some calculations or summaries of the data.

# 1. Calculate and report the total number of missing values in the dataset (i.e. the total numb
er of rows with NAs)

# count total missing values
print("Count of total missing values  ")
```

```
## [1] "Count of total missing values  "
```

```
sum(is.na(activity$steps))
```

```
## [1] 2304
```

```
#13% data is missing
```

```
#2.  Devise a strategy for filling in all of the missing values in the dataset. The strategy doe
s not need to be sophisticated. For example, you could use the mean/median for that day, or the
mean for that 5-minute interval, etc.

#3. Create a new dataset that is equal to the original dataset but with the missing data filled
in.

#create new dataset called activityZ
activityZ <- activity

#create new variable from new dataset for steps data that has NA
NA_steps <- is.na(activityZ$steps)

#using tapply function, lets get the average of interval
avg_interval <- tapply(activityZ$steps, activityZ$interval, mean, na.rm=TRUE, simplify = TRUE)

activityZ$steps[NA_steps] <- avg_interval[as.character(activityZ$interval[NA_steps])]

glimpse(activityZ)
```

```
## Rows: 17,568
## Columns: 3
## $ steps    <dbl> 1.7169811, 0.3396226, 0.1320755, 0.1509434, 0.0754717, 2.0943…
## $ date     <date> 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01, …
## $ interval <dbl> 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 100, 105, 110, …
```

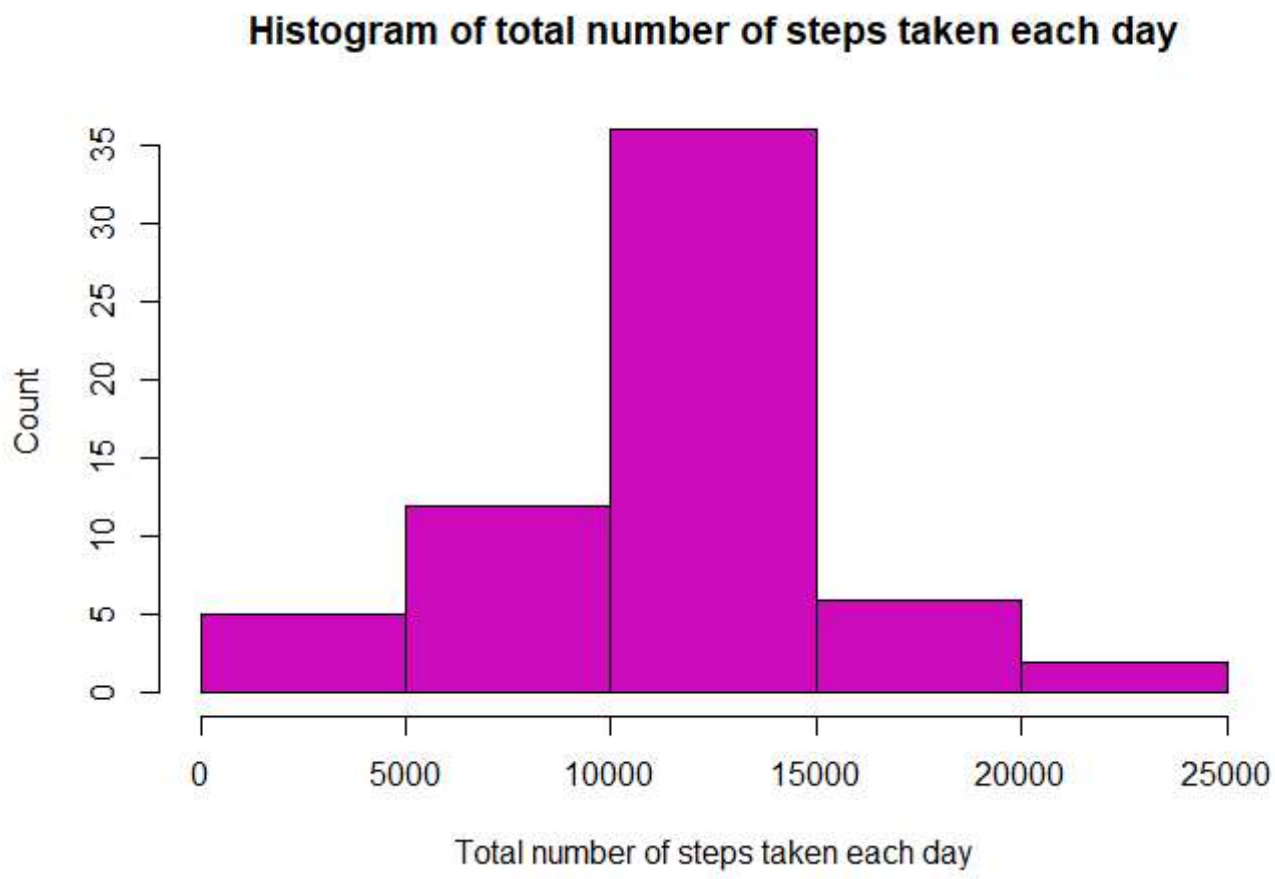
```
## Check for no-NA in new dataset activityZ
sum(is.na(activityZ))
```

```
## [1] 0
```

```
# 4a. Make a histogram of the total number of steps taken each day


#first - calculate number of steps each day with NAs replaced with the median

StepsEachDayZ <- activityZ %>%
    group_by(date) %>%
    summarise("TotalSteps" = sum(steps), .groups = "keep")

print(StepsEachDayZ)
```

```
## # A tibble: 61 × 2
## # Groups:   date [61]
##    date       TotalSteps
##    <date>          <dbl>
##  1 2012-10-01     10766.
##  2 2012-10-02       126
##  3 2012-10-03     11352
##  4 2012-10-04     12116
##  5 2012-10-05     13294
##  6 2012-10-06     15420
##  7 2012-10-07     11015
##  8 2012-10-08     10766.
##  9 2012-10-09     12811
## 10 2012-10-10      9900
## # ℹ 51 more rows
```

```
NoNAstepsEachDay <- activityZ %>%
    group_by(date) %>%
    summarize(TotalSteps=sum(steps))

# Show histogram of steps per day
hist(NoNAstepsEachDay$TotalSteps,
     xlab="Total number of steps taken each day",
     ylab="Count",
     main="Histogram of total number of steps taken each day",
     col=6)
```




```
# 4b. and Calculate and report the mean and median total number of steps taken per day. Do these values differ from the estimates from the first part of the assignment? What is the impact of im puting missing data on the estimates of the total daily number of steps?

AverageStepsZ <- mean(StepsEachDayZ$TotalSteps)
medianStepsZ <- median(StepsEachDayZ$TotalSteps)

print(AverageStepsZ)
```

```
## [1] 10766.19
```

```
print(medianStepsZ)
```

```
## [1] 10766.19
```

```
print("StepsEachDay: ")
```

```
## [1] "StepsEachDay: "
```

```
summary(StepsEachDay)
```

```
##       date              TotalSteps
##  Min.   :2012-10-02   Min.   :   41
##  1st Qu.:2012-10-16   1st Qu.: 8841
##  Median :2012-10-29   Median :10765
##  Mean   :2012-10-30   Mean   :10766
##  3rd Qu.:2012-11-16   3rd Qu.:13294
##  Max.   :2012-11-29   Max.   :21194
```

```
print("StepsEachDayZ: ")
```

```
## [1] "StepsEachDayZ: "
```

```
summary(StepsEachDayZ)
```

```
##       date              TotalSteps
##  Min.   :2012-10-01   Min.   :   41
##  1st Qu.:2012-10-16   1st Qu.: 9819
##  Median :2012-10-31   Median :10766
##  Mean   :2012-10-31   Mean   :10766
##  3rd Qu.:2012-11-15   3rd Qu.:12811
##  Max.   :2012-11-30   Max.   :21194
```

```
#4c. Do these values differ from the estimates from the first part of the assignment? What is th
e impact of imputing missing data on the estimates of the total daily number of steps?

#There is very little differnce between the activity values with the NAs and when they are repla
ced. The mean and median and mean are the same.
```

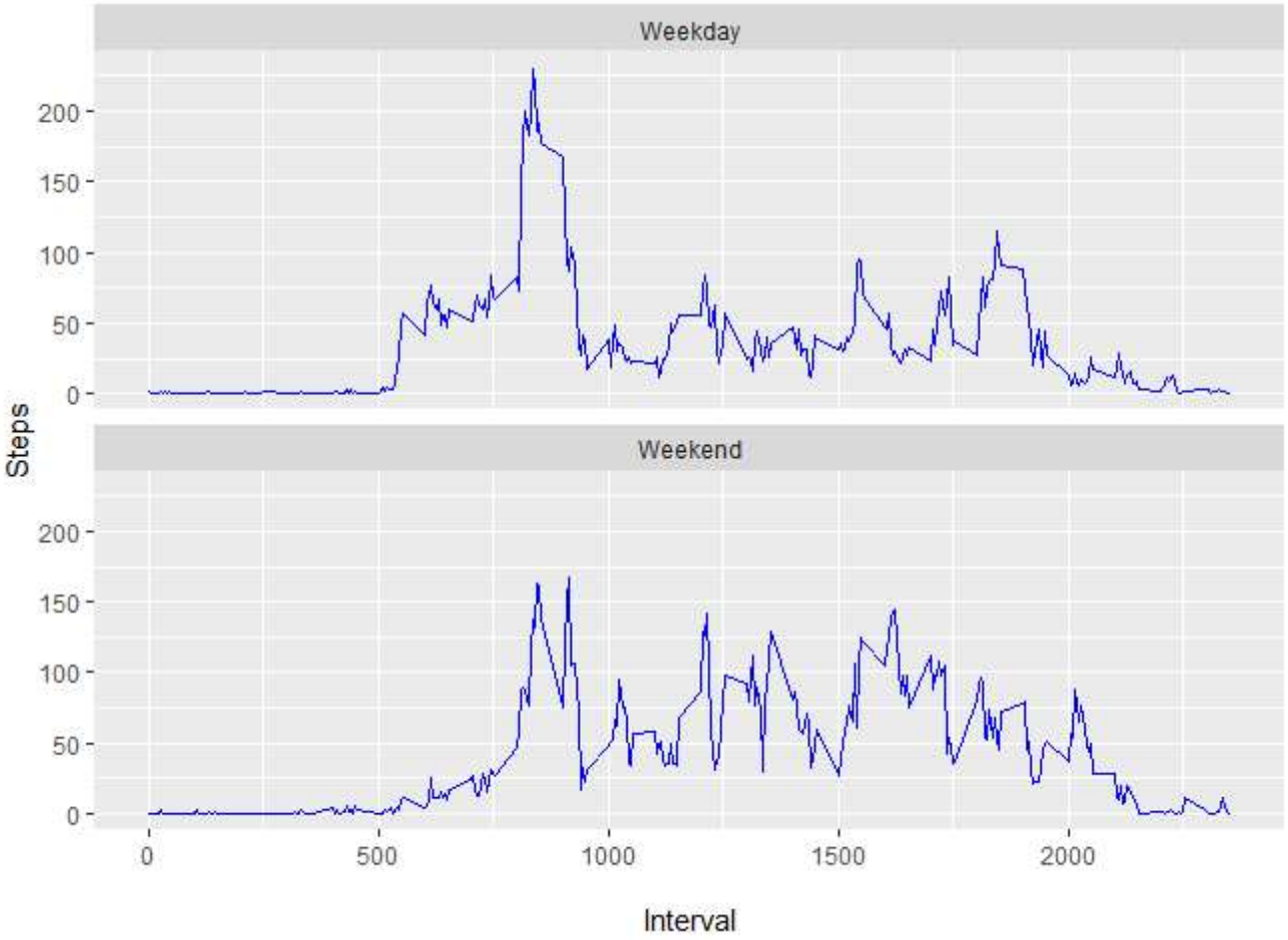
```
# ----- DIFFERENCES IN ACTIVITY PATTERNS BETWEEN WEEKDAYS AND WEEKENDS -----


#For this part the weekdays() weekdays() function may be of some help here. Use the dataset with
the filled-in missing values for this part.

#Create a new factor variable in the dataset with two levels – "weekday" and "weekend" indicatin
g whether a given date is a weekday or weekend day.

activityZ<- activityZ%>%
        mutate(typeofday= ifelse(weekdays(activityZ$date)=="Saturday" | weekdays(activityZ$date)
=="Sunday", "Weekend", "Weekday"))
glimpse(activityZ)
```

```
## Rows: 17,568
## Columns: 4
## $ steps     <dbl> 1.7169811, 0.3396226, 0.1320755, 0.1509434, 0.0754717, 2.094…
## $ date      <date> 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01, 2012-10-01,…
## $ interval  <dbl> 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 100, 105, 110,…
## $ typeofday <chr> "Weekday", "Weekday", "Weekday", "Weekday", "Weekday", "Week…
```

```
# 2. Make a panel plot containing a time series plot (i.e. type = "l" ) of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all weekday days or weekend days (y-axis). See the README file in the GitHub repository to see an example of what this plot shoul d look like using simulated data.

# Group data by interval and summarize the average
# number of steps in that interval

AvgEachDay <- activityZ %>%
    group_by(typeofday, interval) %>%
    summarize(AverageSteps=mean(steps), .groups = "keep")

head(AvgEachDay)
```

```
## # A tibble: 6 × 3
## # Groups:   typeofday, interval [6]
##   typeofday interval AverageSteps
##   <chr>        <dbl>        <dbl>
## 1 Weekday          0       2.25
## 2 Weekday          5       0.445
## 3 Weekday         10       0.173
## 4 Weekday         15       0.198
## 5 Weekday         20       0.0990
## 6 Weekday         25       1.59
```

```
ggplot(AvgEachDay, aes(x=interval, y=AverageSteps)) +
  geom_line(color="blue") +
  facet_wrap(~typeofday, nrow=2) +
  labs(x="\nInterval", y="\nSteps")
```


Weekday
Weekend
Steps
200
150
100
50
0
0
500
1000
1500
2000
Interval