Quantified Self Movement Monitor Data Analysis

Anandu R

8/3/2020

It is now possible to collect a large amount of data about personal movement using activity monitoring devices such as a Fitbit, Nike Fuelband, or Jawbone Up. These type of devices are part of the "quantified self" movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. But these data remain under-utilized both because the raw data are hard to obtain and there is a lack of statistical methods and software for processing and interpreting the data.

Loading the data Downloading the data

```
fileUrl = "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"
if(!file.exists("./data/activity.csv")){
  download.file(fileUrl, destfile = "./data/data.zip")
  unzip("./data/data.zip", exdir = "./data")
}
unlink("./data/data.zip", recursive = T)
```

Reading the raw data

```
data = read.csv("./data/activity.csv")
```

Preliminary analysis on data

```
dim(data)
```

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

```
##   steps       date interval
## 1    NA 2012-10-01        0
## 2    NA 2012-10-01        5
## 3    NA 2012-10-01       10
## 4    NA 2012-10-01       15
## 5    NA 2012-10-01       20
## 6    NA 2012-10-01       25
```

There seems to be a bit of missing values in the data.

Taking a look at how much of the data is missing

```
temp = mean(is.na(data))
```

which is just 0.0437158 of the data, we might as well get rid of the specific records which have missing values.

calculating the % of records that have valid activity measurement, we find that 0.8688525 of data is good, so we get rid of rest of the records that have missing values

```
## good_data
## FALSE  TRUE
##  2304 15264
```

```
data_clean = data[good_data,]
## Corrected datatype of date field
data_clean = transform(data_clean, date= as.Date(date))
```

Taking a look at the valid activity tracked, for all 'steps'>0

```
##     steps       date interval
## 555   117 2012-10-02     2210
## 556     9 2012-10-02     2215
## 627     4 2012-10-03      410
## 631    36 2012-10-03      430
## 644    25 2012-10-03      535
## 647    90 2012-10-03      550
```

Histogram of the total number of steps taken each day

```
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
```

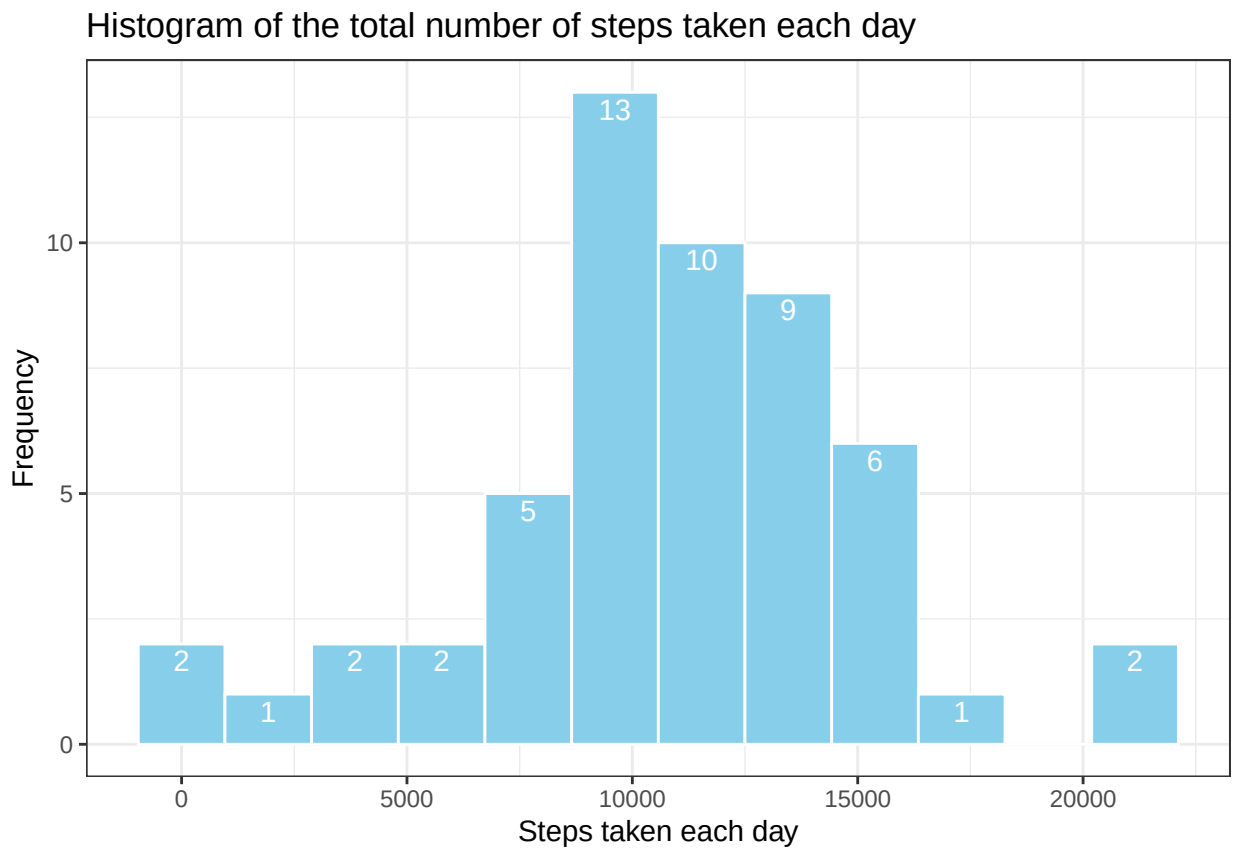
```
data_grpd = data_clean %>%
  group_by(date) %>%
  summarise(steps_per_day = sum(steps))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
data_grpd
```

```
## # A tibble: 53 x 2
##    date       steps_per_day
##    <date>             <int>
##  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
## # ... with 43 more rows
```

```
library(ggplot2)
ggplot(
  data_grpd,
  aes(
    x = steps_per_day,
  ),
) +
  geom_histogram(bins = 12, fill = "skyblue", col = "white") +
  stat_bin(bins = 12,geom="text", aes(label=..count..), col ="white", vjust = 1.3) +
  theme_bw() + guides(fill = F) +
  labs(
    title = "Histogram of the total number of steps taken each day",
    x = "Steps taken each day",
    y = "Frequency")
```



On average people take 10000 steps daily.

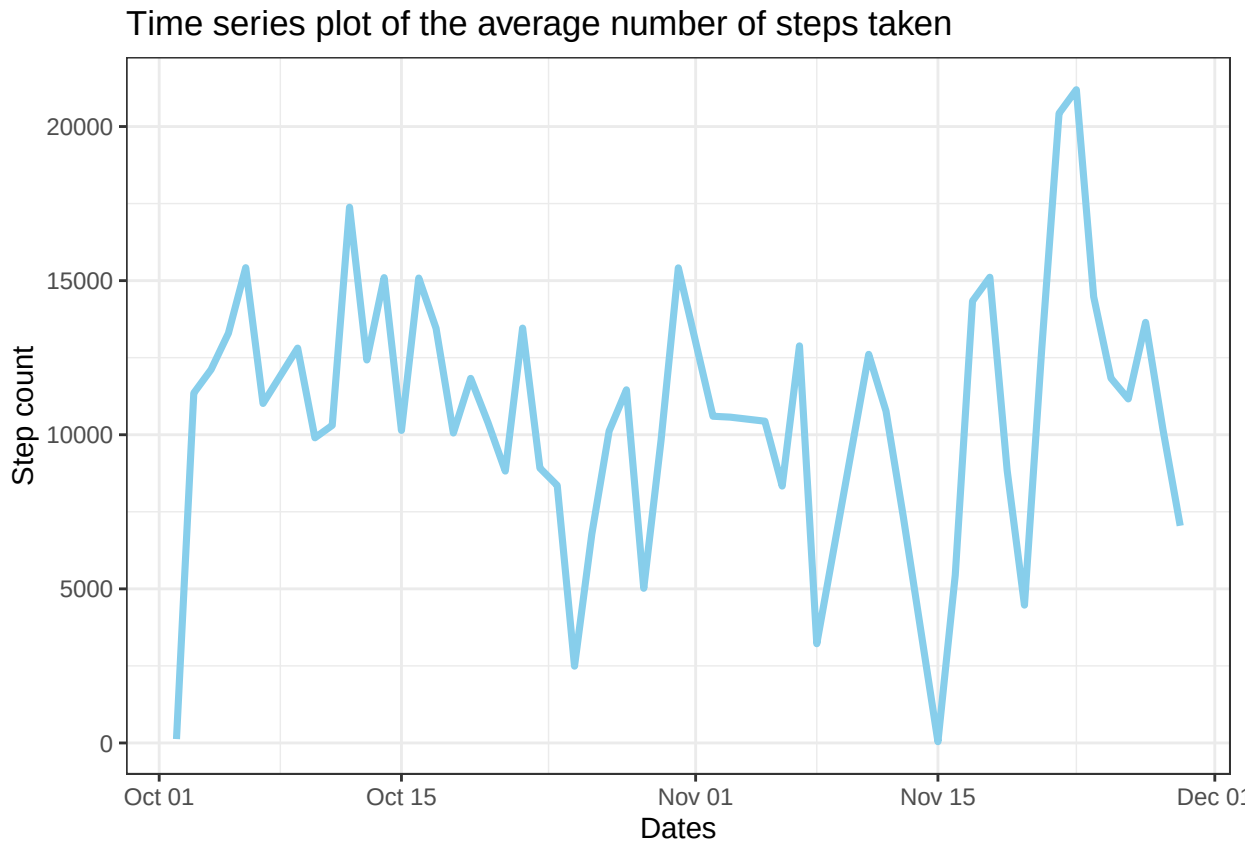
Mean and median number of steps taken each day

```
data_mm = suppressMessages(
  {
    data_grpd %>%
      group_by(month = format(date, format= "%m")) %>%
      summarise(
        mean_steps = mean(steps_per_day),
        median_steps = median(steps_per_day)
      )
    }
  )
data_mm = transform(data_mm, month = month.abb[as.numeric(month)])
data_mm
```

```
##   month mean_steps median_steps
## 1   Oct   10787.97      11015.0
## 2   Nov   10739.88      10682.5
```

Time series plot of the average number of steps taken

```
ggplot(
  data_grpd,
  aes(
    x = date,
    y = steps_per_day,
  )
) +
  geom_line(col = 'skyblue', lwd = 1.25) +
  theme_bw() + guides(fill = F) +
  labs(
    title = "Time series plot of the average number of steps taken",
    x = "Dates",
    y = "Step count"
  )
```



Step count average in day across the time period Making a summarised data with step count average in day across the time period

```
data_consdt = data_clean %>%
  group_by(interval) %>%
  summarise(steps_by_interval = mean(steps))
```

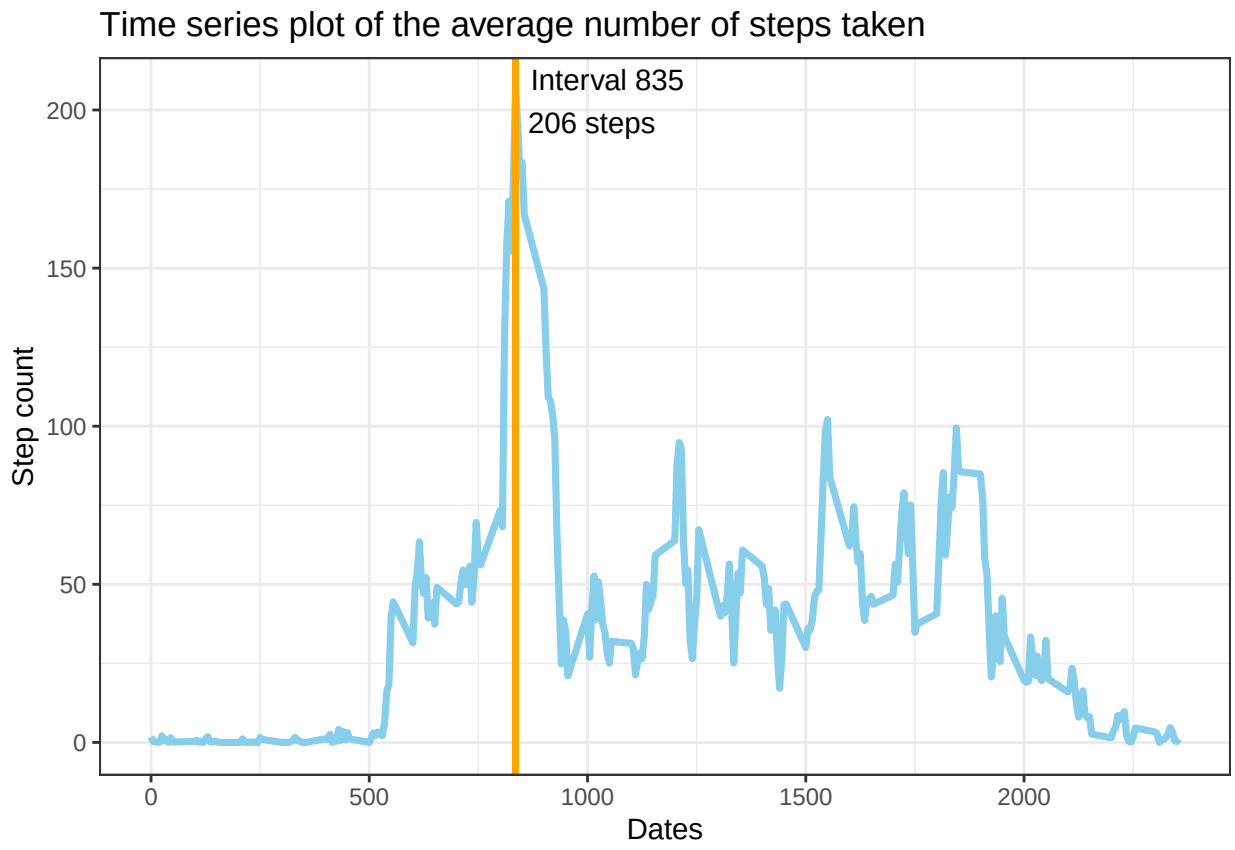
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
data_consdt
```

```
## # A tibble: 288 x 2
##    interval steps_by_interval
##       <int>             <dbl>
##  1        0            1.72
##  2        5            0.340
##  3       10            0.132
##  4       15            0.151
##  5       20            0.0755
##  6       25            2.09
##  7       30            0.528
##  8       35            0.868
##  9       40            0
## 10       45            1.47
## # ... with 278 more rows
```

```
idx_max_meanSteps = which.max(data_consdt$steps_by_interval)
max_meanSteps_interval = data_consdt$interval[idx_max_meanSteps]
ggplot(
  data_consdt,
  aes(
    x = interval,
    y = steps_by_interval,
  )
) +
  geom_line(col = 'skyblue', lwd = 1.25) +
  theme_bw() + guides(fill = F) +
  labs(
    title = "Time series plot of the average number of steps taken",
    x = "Dates",
    y = "Step count"
  ) +
  geom_vline(
    xintercept = max_meanSteps_interval,
    col = "orange",
    lwd = 1.25
  ) +
  geom_text(
    aes(
      label = ifelse(
        interval == max_meanSteps_interval,
        paste("Interval",as.character(interval)),
        ''
        )
      ),
    hjust=-0.1,
    vjust=0
  ) +
  geom_text(
    aes(
      label = ifelse(
        interval == max_meanSteps_interval,
```

```
        paste(as.character(round(steps_by_interval)),"steps"),
        ''
        )
      ),
    hjust=-0.1,
    vjust=2
  )
```



The 5-minute interval that, on average, contains the maximum number of steps

The previous plot depicts the the interval at which the most number of steps are tracked on an average ie. at 835, the user takes 206 steps.

Code to describe and show a strategy for imputing missing data

To impute missing values we'll use the mean step count data we've calculated previously and insert the mean value in the records with missing value with the corresponding mean value

```
data_imputed = data
idx = which(is.na(data_imputed$steps))
for(i in idx){
  data_imputed$steps[i] =
    round(
        data_consdt$steps_by_interval[which(
```

```
          data_imputed$interval[i] == data_consdt$interval
          )
        ]
    )
}
```

Imputed data

```
head(data_imputed)
```

```
##   steps       date interval
## 1     2 2012-10-01        0
## 2     0 2012-10-01        5
## 3     0 2012-10-01       10
## 4     0 2012-10-01       15
## 5     0 2012-10-01       20
## 6     2 2012-10-01       25
```

There are 0 missing values in the imputed dataset 'data_imputed'

```
sum(is.na(data_imputed))
```

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

Histogram of the total number of steps taken each day after missing values are imputed

Summarising the data

```
data_grpd_imp = data_imputed %>%
  group_by(date) %>%
  summarise(steps_per_day = sum(steps))
```

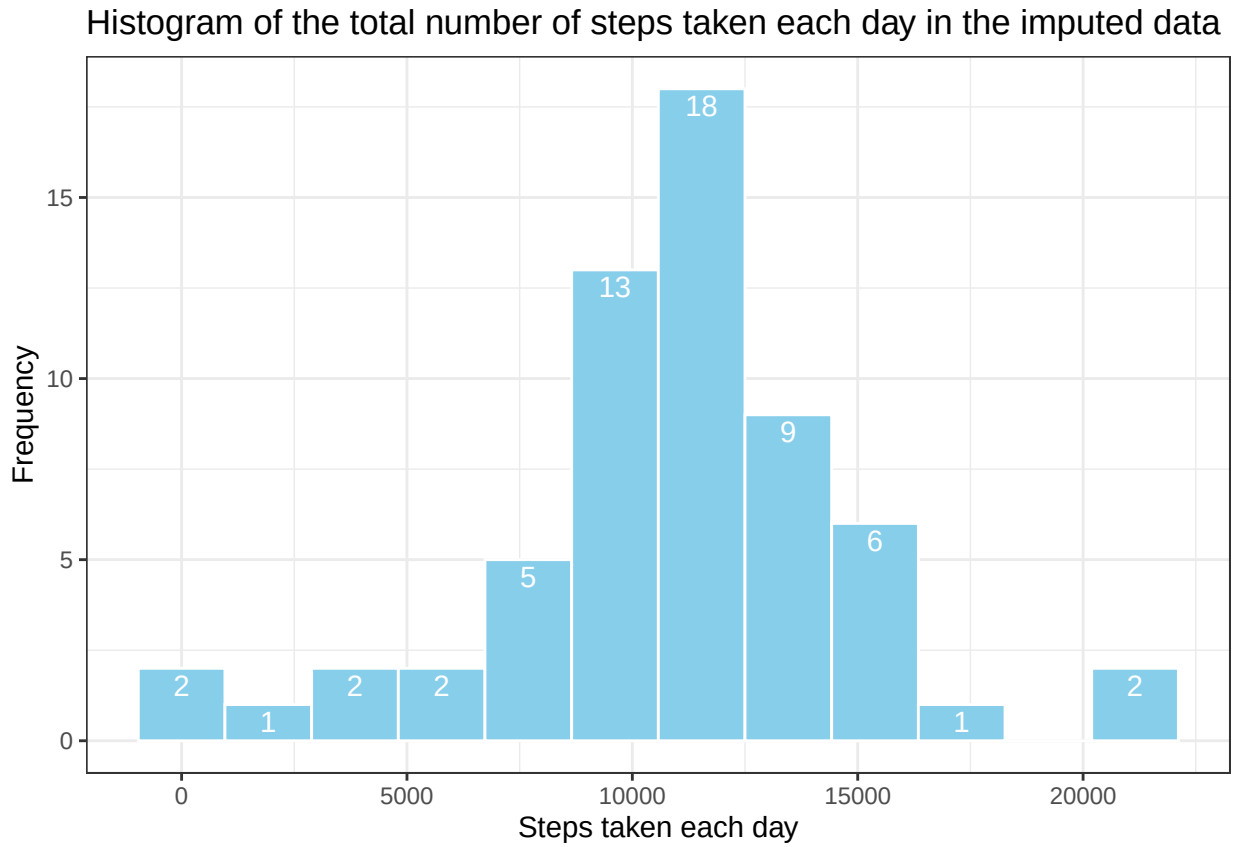
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
data_grpd_imp
```

```
## # A tibble: 61 x 2
##    date       steps_per_day
##    <chr>              <dbl>
##  1 2012-10-01         10762
##  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         10762
##  9 2012-10-09         12811
## 10 2012-10-10          9900
## # ... with 51 more rows
```

Visualization

```
ggplot(
  data_grpd_imp,
  aes(
    x = steps_per_day,
  ),
) +
  geom_histogram(bins = 12, fill = "skyblue", col = "white") +
  stat_bin(bins = 12,geom="text", aes(label=..count..), col ="white", vjust = 1.3) +
  theme_bw() + guides(fill = F) +
  labs(
    title = "Histogram of the total number of steps taken each day in the imputed data",
    x = "Steps taken each day",
    y = "Frequency")
```



Comparing the average number of steps taken per interval across weekdays and weekends

Panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends

Adding a column which indicates whether the date represents weekend or not

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
data_imputed$weekend = factor(ifelse(wday(data_imputed$date) %in% c(2:6),0,1))
```

Considating the data by grouping by weekend first then interval

```
data_consdt_imp = data_imputed %>%
  group_by(weekend,interval) %>%
  summarise(steps_by_interval = mean(steps))
```

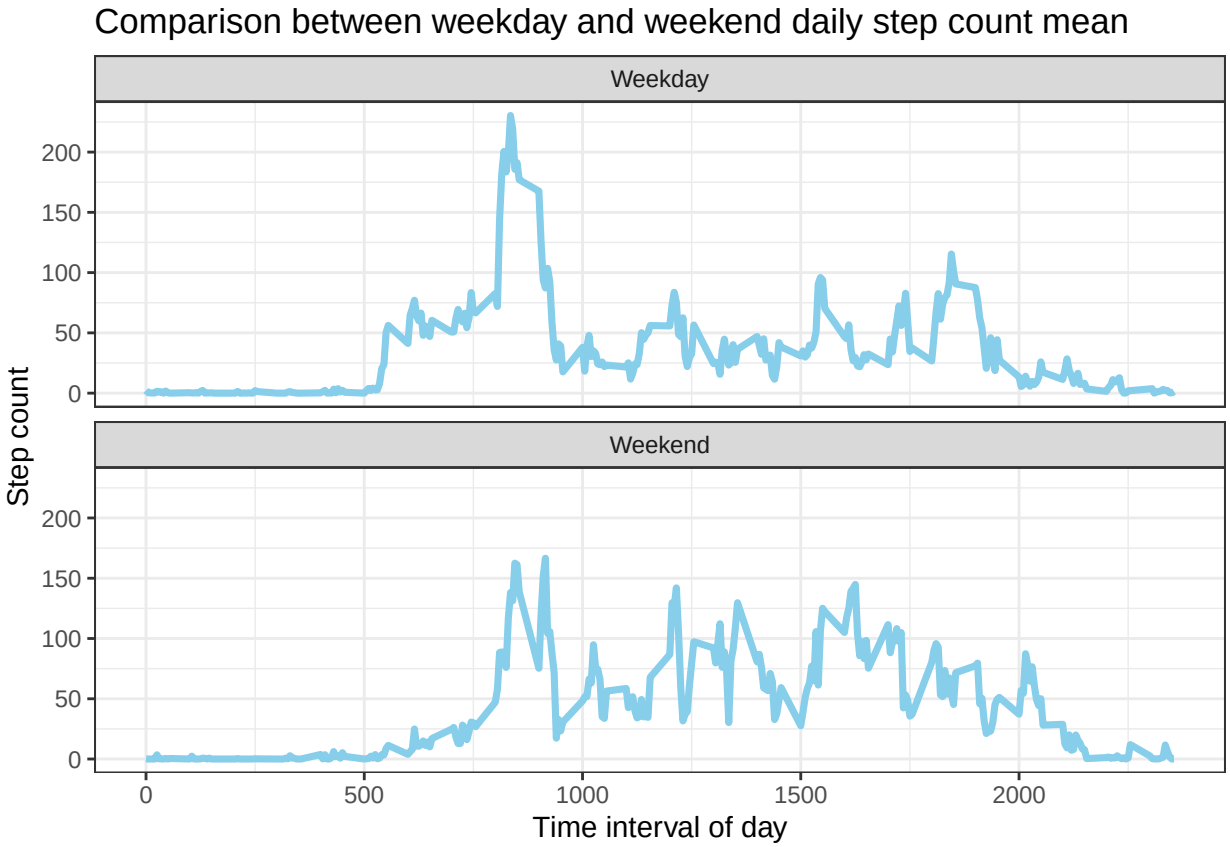
```
## `summarise()` regrouping output by 'weekend' (override with `.groups` argument)
```

```
data_consdt_imp
```

```
## # A tibble: 576 x 3
## # Groups:   weekend [2]
##    weekend interval steps_by_interval
##    <fct>      <int>             <dbl>
##  1 0              0            2.29
##  2 0              5            0.4
##  3 0             10            0.156
##  4 0             15            0.178
##  5 0             20            0.0889
##  6 0             25            1.58
##  7 0             30            0.756
##  8 0             35            1.16
##  9 0             40            0
## 10 0             45            1.73
## # ... with 566 more rows
```

Visualizing the weekday vs weekend step-count through the day

```
levels(data_consdt_imp$weekend) = c("Weekday","Weekend")
ggplot(
  data_consdt_imp,
  aes(
    x = interval,
    y = steps_by_interval,
  )
) +
  geom_line(col = 'skyblue', lwd = 1.25) +
  theme_bw() + guides(fill = F) +
  labs(
    title = "Comparison between weekday and weekend daily step count mean",
    x = "Time interval of day",
    y = "Step count"
  ) +
  facet_wrap(.~weekend, ncol = 1)
```


Comparison between weekday and weekend daily step count mean
Weekday
Weekend
Step count
0
50
100
150
200
0
500
1000
1500
2000
Time interval of day