Reproducible Research: Peer Assessment 1

David Rozene

Feb. 12, 2018

1.0 Load and preprocess the data

1.1 Load data

Set directories

The working directory is part of the R environment and should be controlled by the user, not a script.
* Set your own working directory in *filePath*.
* Name the directories *mainDir* and *subDir* according to personal preference.
* Function **mkdirs** will create a project specific directory and raw data subdirectory in the working directory.

```
filePath <- "/Volumes/Red Seagate 2 Tb -1/Courses/Coursea.org/Data Science Specialization Courses/5. Reproducible Research"
setwd(filePath)
mainDir <- "Project 1"
subDir <- "Raw Data"
mkdirs <- function(fp) {
  if(!file.exists(fp)) {
    mkdirs(dirname(fp))
    dir.create(fp)
  }
}
mkdirs((paste(filePath, mainDir, subDir, "/", sep = "/", collapse = "/")))
setwd(mainDir)
```

Download and read raw data to subdirectory from weblink.

Datetime stamp the event in case the dataset changes in the future.
In this script, the name corresponds to the original data set name. The numeric suffix is a personal preference to indicate where the dataset is deployed within the project.

```
setwd(subDir)
weblink <- "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"
if(!file.exists("dataZIPfile")) {
  download.file(weblink, destfile = "dataZIPfile")
  dataUNZfile <- unzip("dataZIPfile")
  datetimestamp <-  Sys.time()
}
print(paste("Data sets downloaded on ", Sys.time(), " from weblink ", weblink))
```

```
## [1] "Data sets downloaded on  2018-02-12 10:30:10  from weblink  https://d396qusza
40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"
```

```
activity1 <-read.csv("activity.csv", sep = ",")
setwd((paste(filePath, mainDir, "/", sep = "/", collapse = "/")))
```

2.0 What is mean total number of steps taken per day? Omit NA values.

2.1 Make a histogram of the total number of steps taken each day.

Subset the dataset to include only rows without NA.

Rename the original dataset to a working dataset to avoid corrupting the original dataset while preparing code.
The numeric suffix is a personal preference to indicate where in the project the dataset is deployed.
which( ) returns rows which do not have NA values in the column steps.

```
activity2 <- activity1
activity2 <- (activity2[which(!is.na(activity2$steps)), ])
```

Revise data classes for use in date related and classification type operations.

as.Date() parses text input (e.g. “2012–10–02”) to date data type object. **as.factor** Convert the values in the interval column from integers to categorical variables.
Sys.timezone() returns the local time zone.

Convert values in the interval column from digits to hours and minutes for each day. Assign new values to column interval_dt.

sprintf() returns a character vector containing a formatted combination of text and variable values.
%04d“ pads the interval data with preceding zeros and make values a uniform 4 digits.
%H converts hours (00-23) and *%M* converts minutes (00-59) to decimal numbers.
strptime() converts character vectors to class POSIXlt.
as.POSIXct functions convert an object to one of the two classes used to represent date/times.

```
activity2$date <- as.Date(activity2$date, format="%Y-%m-%d")
activity2$interval <- as.factor(activity2$interval)
timezone <- Sys.timezone()
activity2$interval_dt <- as.POSIXct(strptime(sprintf("%04d", activity2$interval), tz
= timezone, format = "%H%M"))
```

The *lubridate* package makes dealing with dates and times easier. see also

[LINK] (https://blog.exploratory.io/5-most-practically-useful-operations-when-working-with-date-and-time-in-r-9f9eb8a17465 (https://blog.exploratory.io/5-most-practically-useful-operations-when-working-with-date-and-time-in-r-9f9eb8a17465)) **lubrdiate::wday ()** adds a column for weekdays corresponding to dates.

The *dplyr* package provides a gramar for data manipulation.

see also [LINK] (http://genomicsclass.github.io/book/pages/dplyr_tutorial.html (http://genomicsclass.github.io/book/pages/dplyr_tutorial.html))

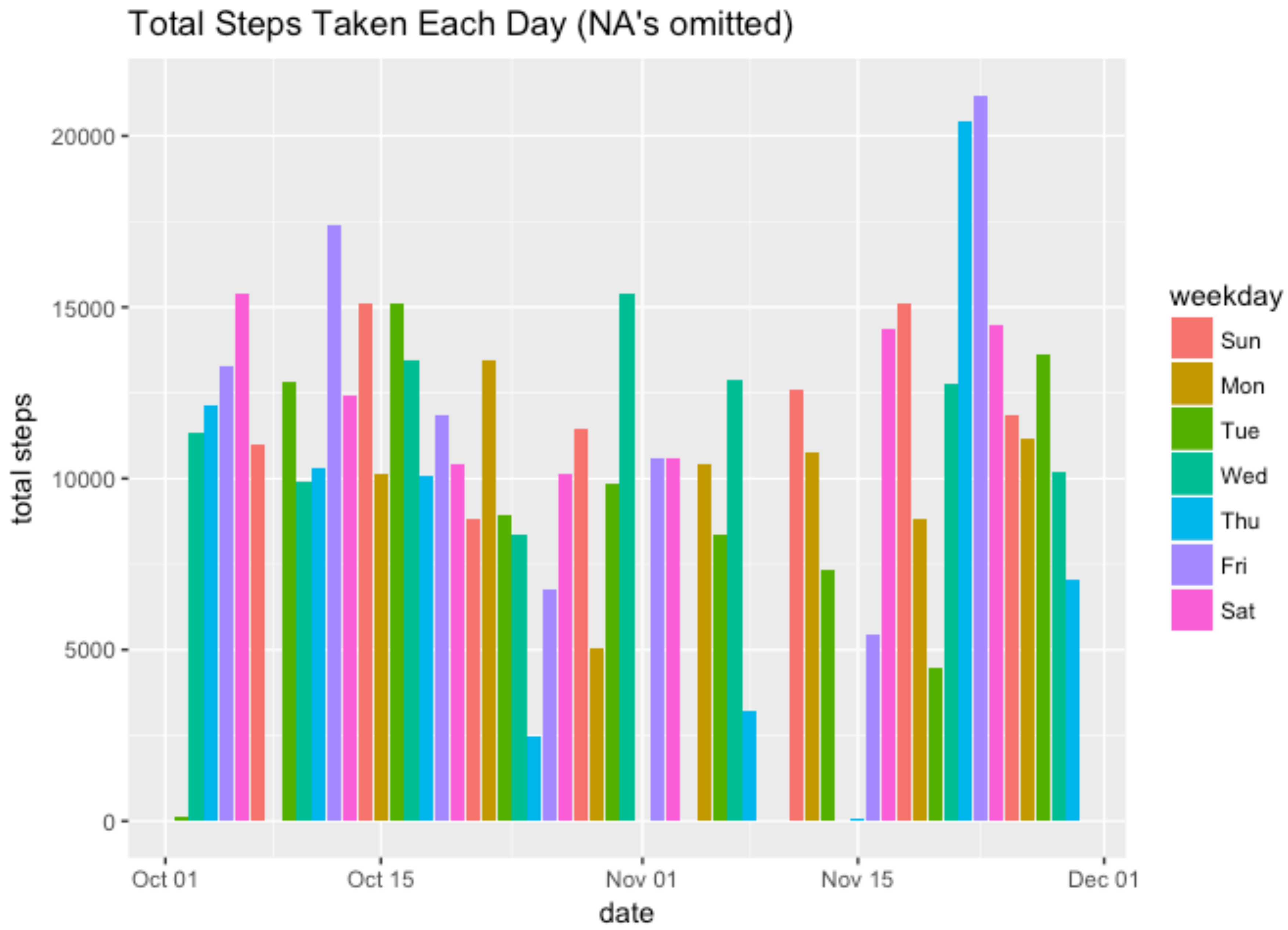
Group dataset by date and sum the steps

```
suppressMessages(library(lubridate))
suppressMessages(library(dplyr))
activity2 <- mutate(activity2, weekday = lubridate::wday(activity2$date, label=TRUE, abbr = TRUE))
activity2_sum_daily <- activity2 %>%
  group_by(date, weekday) %>%
  dplyr::summarise(Sum_Daily_Steps = sum(steps))
```

Make a histogram of the total number of steps taken each day

The **ggplot2** package creates elegant data visualisation using the grammar of graphics. (see also tidyverse.org)
If you want the heights of the bars to represent values in the data, use *stat="identity"* and map a value to the y aesthetic.

```
library(ggplot2)
g2 <- ggplot (data = activity2_sum_daily, aes(x = date, y = Sum_Daily_Steps, fill = weekday)) +
      geom_bar(stat = "identity")
myxlab2 <- "date"
myylab2 <- "total steps"
mytitle2 <- "Total Steps Taken Each Day (NA's omitted)"
g2 <- g2 +
  xlab(myxlab2) +
  ylab(myylab2) +
  ggtitle(mytitle2)
plot(g2)
```



2.3 Calculate and report the mean and median of the total number of steps taken per day

mean

```
activity2_mean <- mean(activity2_sum_daily$Sum_Daily_Steps)
```

median

```
activity2_median <-  median(activity2_sum_daily$Sum_Daily_Steps)
```

```
## [1] "average of steps taken per day is "
## [2] "10766"
```

```
## [1] "median of of total steps taken per day taken is "
## [2] "10765"
```

3.0 What is the average daily activity pattern?

3.1 Make a time series plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days

Process and manipulate dataset that omits NA values for plotting

```
activity3 <- activity2
activity3_interval_avg <- activity3 %>%
                          group_by(interval_dt) %>%
                          summarise(Average_Steps_by_Interval = mean(steps)) %>%
                          arrange(interval_dt)
```

Plot average of all steps by interval

The **scales** package scales functions for visualization.

geom_line() is suitable for time series, linetype is solid

theme_classic() adds the signature ggplot2 theme with a grey background and white gridlines, designed to put the data forward yet make comparisons easy

scale_x_dateime # overrides the default scale for date/time class. Otherwise current date will appear.

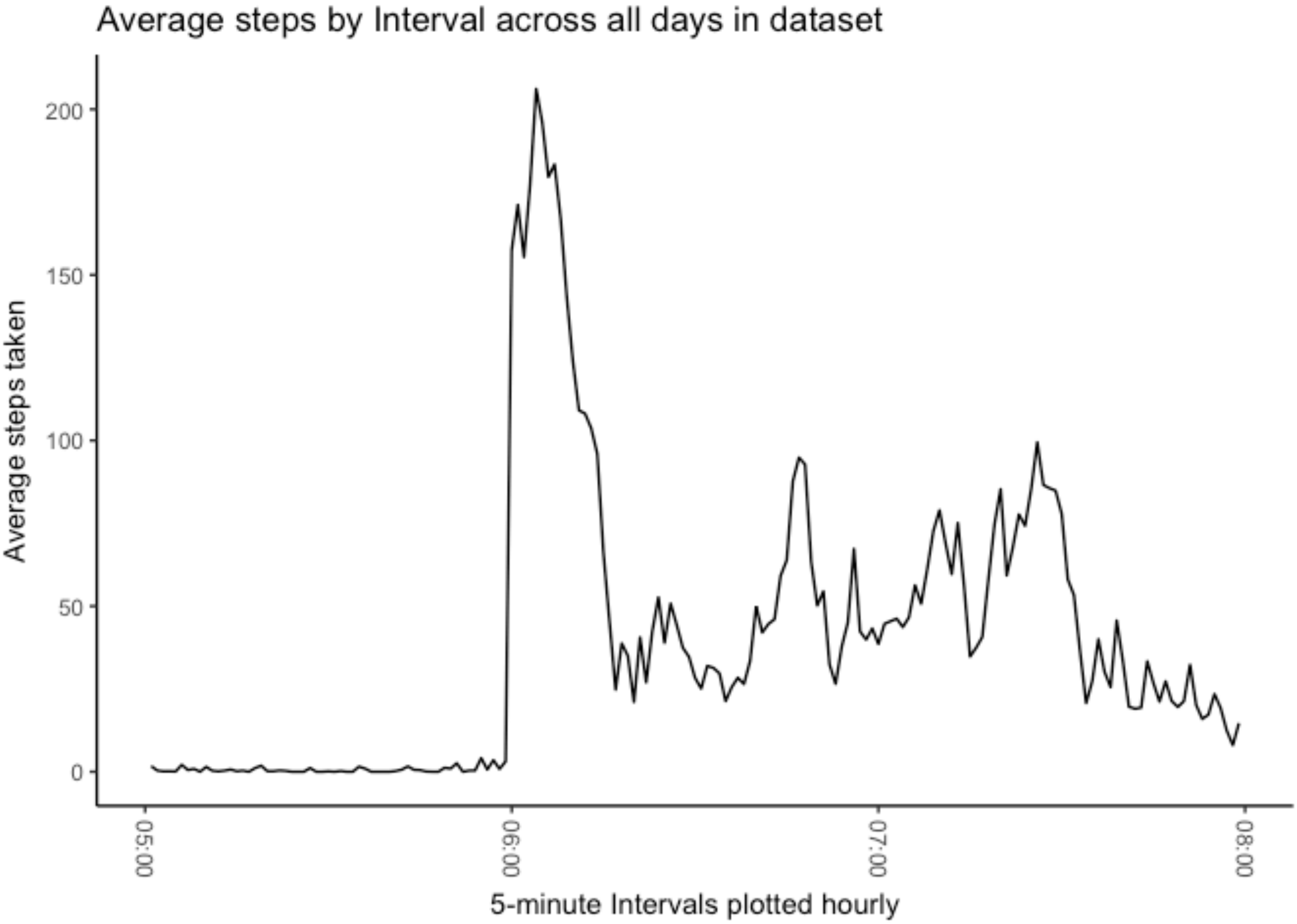
In conjunction with **theme**, *element_functions* specify for the display of how non-data components of the plot are drawn.

hjust controls horizontal justification and vjust controls vertical justification.

axis.text.x are x axis tick labels (element_text; inherits from axis.text)

```
suppressMessages(library(scales))
g3 <-  ggplot(data = activity3_interval_avg,
              aes (x=interval_dt, y = Average_Steps_by_Interval)) +
  geom_line(linetype = 1)
myxlab3 <- "5-minute Intervals plotted hourly"
myylab3 <- "Average steps taken"
mytitle3 <- "Average steps by Interval across all days in dataset"
g3 <- g3 +  xlab(myxlab3) +
            ylab(myylab3) +
            ggtitle(mytitle3)
g3 <- g3 + theme_classic()
g3 <- g3 + scale_x_datetime(breaks = date_breaks(width = "1 hours"),
                                      labels = date_format("%H:%M"))
g3 <- g3 + theme(axis.text.x = element_text(angle = 270, hjust = 1,vjust = 0.5, size
= 9))
plot(g3)
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



3.2 Which 5-minute interval, on average across all the days in the dataset, contains the maximum number of steps?

```
activity3_max <- activity3_interval_avg [which.max(activity3_interval_avg$Average_Ste
ps_by_Interval), ]
activity3_max_interval <- (activity3_max[1])
activity3_max_steps <- round(activity3_max[, 2], digits = 0)
```

```
## [[1]]
## [1] "Maximum number of steps are "
##
## $Average_Steps_by_Interval
## [1] 206
##
## [[3]]
## [1] "during interval"
##
## $interval_dt
## [1] "2018-02-12 01:04:00 EST"
```

4.0 Imputing missing values

“impute”: assign (a value) to something by inference from the value of the products or processes to which it contributes.

4.1 Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with `NA`s)

Process data

```
activity4 <- activity1
activity4 <- mutate(activity4, weekday = lubridate::wday(activity4$date, label=TRUE,
abbr = TRUE))
activity4$date <- as.Date(activity4$date, format="%Y-%m-%d")
activity4$interval_dt <- as.POSIXct(strptime(sprintf("%04d", activity4$interval), tz
= timezone, format = "%H%M"))
```

mice package has a function known as md.pattern().

It returns a tabular form of missing value present in each variable in a data set.
see also LINK (https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/)

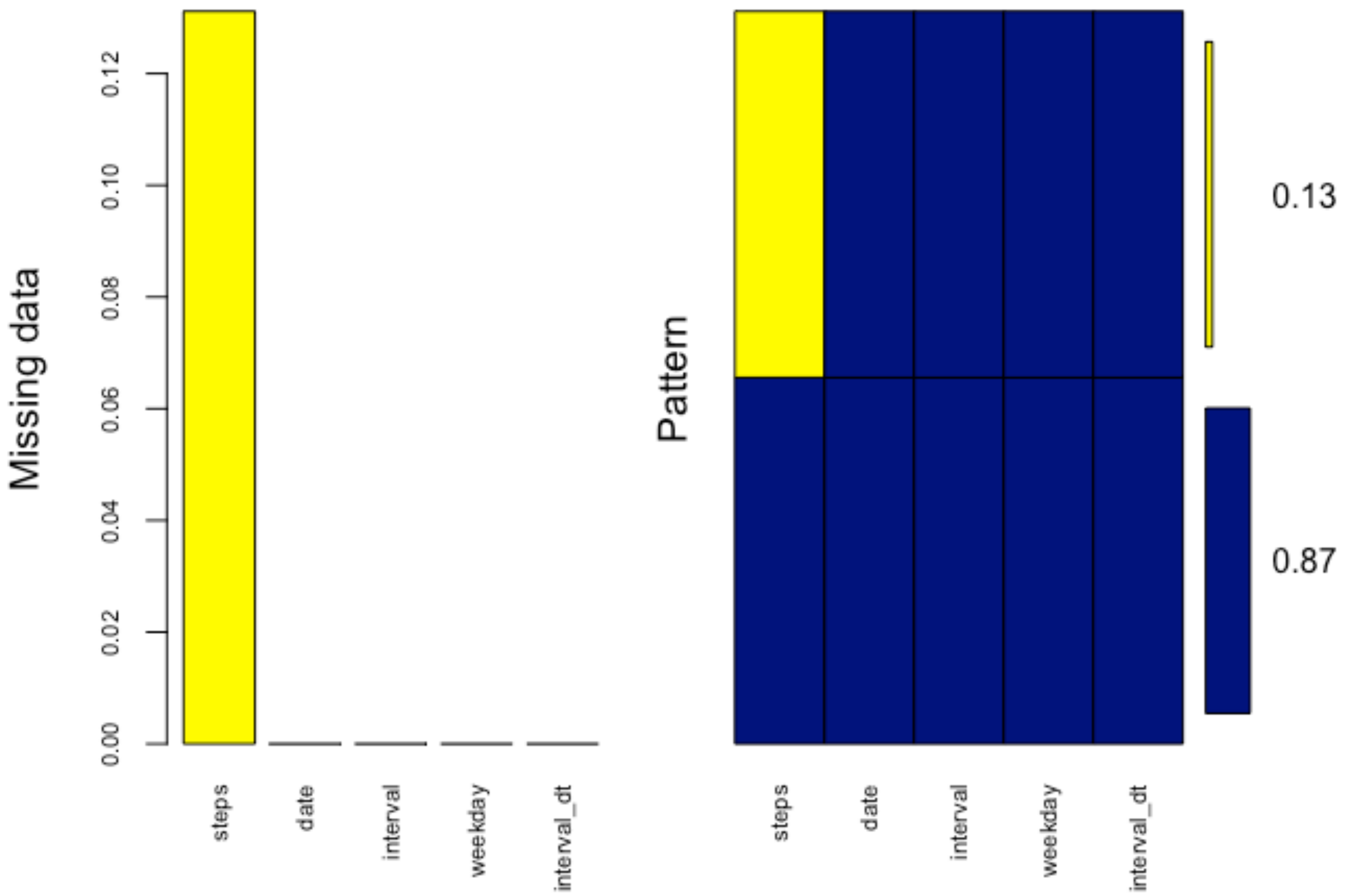
```
suppressMessages(library(mice))
md.pattern(activity4)
```

```
##       date interval weekday interval_dt steps
## 15264    1        1       1           1     1    0
##  2304    1        1       1           1     0    1
##          0        0       0           0  2304 2304
```

VIM package (Visualization and Imputation of Missing Values)

Introduces new tools for the visualization of missing or imputed values in, # which can be used for exploring the data and the structure of the missing or imputed values.
see also LINK

```
suppressMessages(library(VIM))
activity4a_ice_plot <- aggr(activity4, col=c('navyblue','yellow'),
                            numbers=TRUE, sortVars=TRUE,
                            labels=names(activity4), cex.axis=.7,
                            gap=3, ylab=c("Missing data","Pattern"))
```



```
##
##  Variables sorted by number of missings:
##     Variable     Count
##        steps 0.1311475
##         date 0.0000000
##     interval 0.0000000
##      weekday 0.0000000
##  interval_dt 0.0000000
```

Devise a strategy for filling in all of the missing values in the dataset.

The strategy is to create a function impute.mean() that replaces NA's in each interval with avearage of available data in the interval

```
impute.mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))
```

4.3 Ceate a new dataset that is equal to the original dataset but with the missing data filled in.

Group_by() by interval. **impute.mean()** to replace NA's with the average steps of corresponding interval groups.

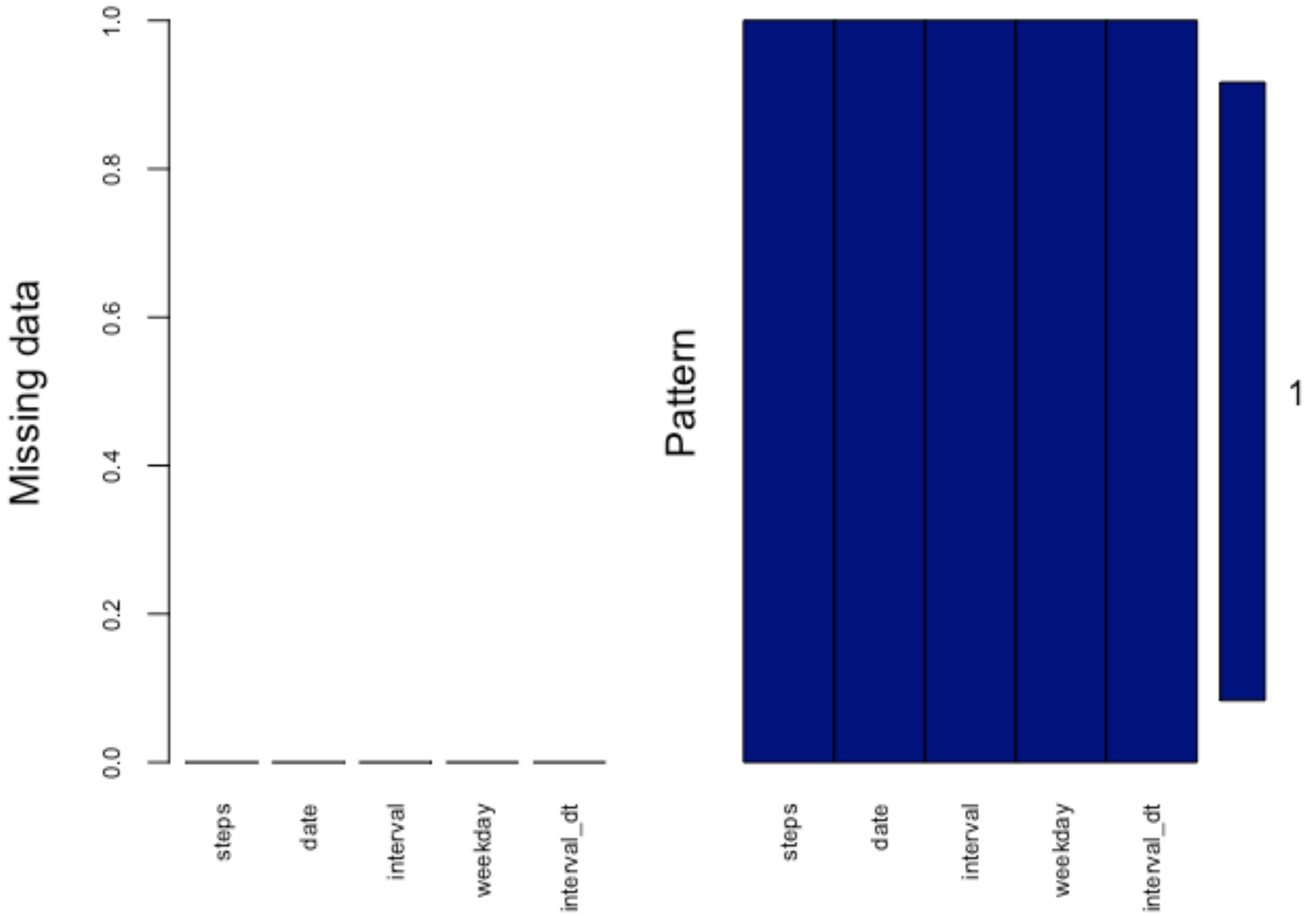
```
activity4_imputed <-  activity4 %>%
  group_by(interval_dt) %>%
  mutate(steps = impute.mean(steps))
```

Confirm that NA values are replaced.

```
md.pattern(activity4_imputed)
```

```
##      steps date interval weekday interval_dt
## [1,]     1    1        1       1           1 0
## [2,]     0    0        0       0           0 0
```

```
activity4b_mice_plot <- aggr(activity4_imputed, col=c('navyblue','yellow'),
                             numbers=TRUE, sortVars=TRUE,
                             labels=names(activity4_imputed), cex.axis=.7,
                             gap=3, ylab=c("Missing data","Pattern"))
```



```
##
##  Variables sorted by number of missings:
##     Variable Count
##        steps     0
##         date     0
##     interval     0
##      weekday     0
##  interval_dt     0
```

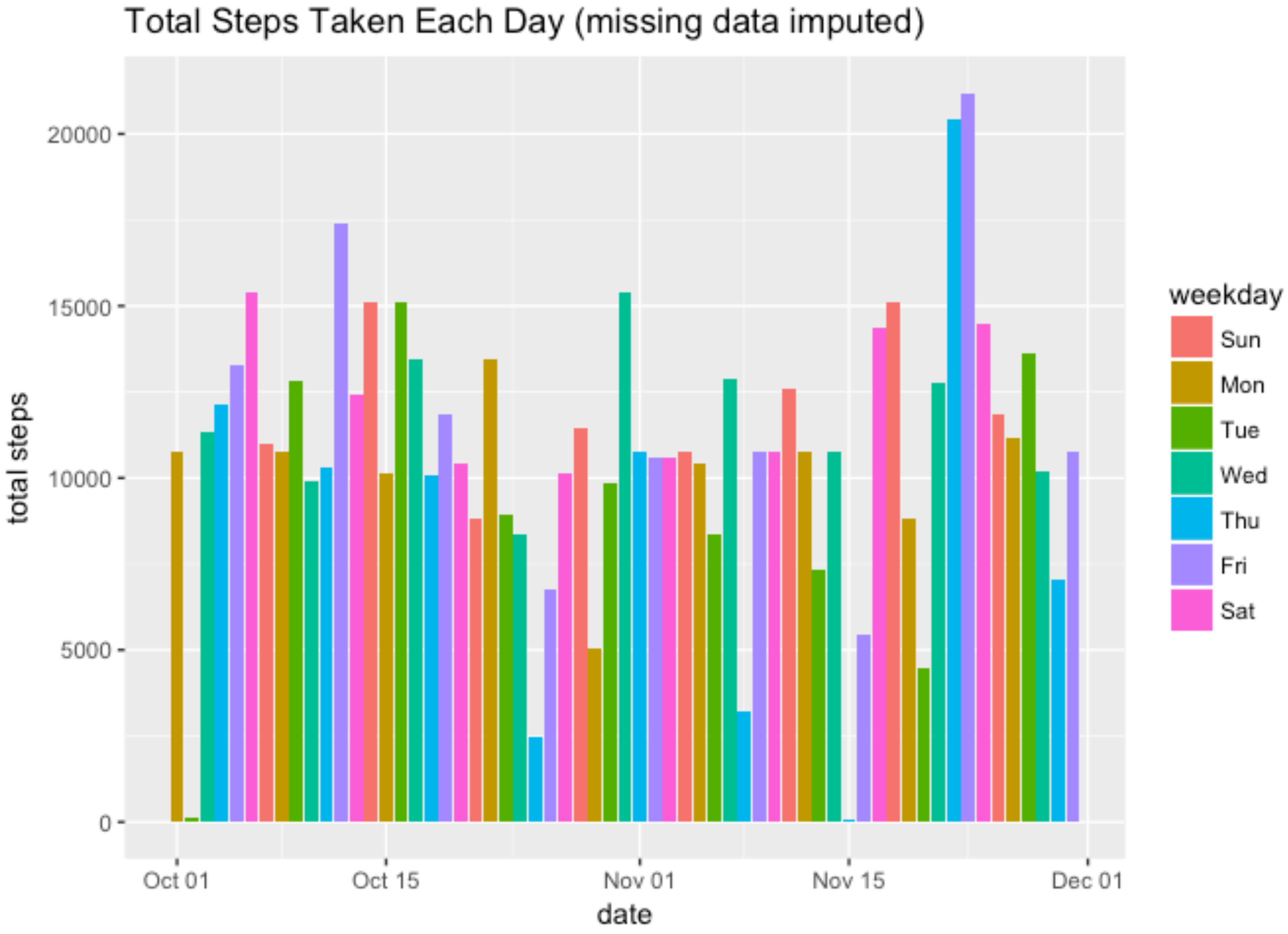
4.4 Make a histogram of the total number of steps taken each day. Calculate and report the mean and median total number of steps taken per day.

Calculate the total number of steps taken per day with imputed NA values (ref. 2.1.4). Group dataset by date and sum the steps.

```
activity4_sum_daily <- activity4_imputed %>%
  group_by(date, weekday) %>%
  dplyr::summarise(Sum_Daily_Steps = sum(steps))
```

Plot with ggplot

```
g4 <- ggplot(data = activity4_sum_daily,
             aes(x = date, y = Sum_Daily_Steps, fill = weekday)
) +
  geom_bar(stat = "identity")
myxlab4 <- "date"
myylab4 <- "total steps"
mytitle4 <- "Total Steps Taken Each Day (missing data imputed)"
g4 <- g4 +
  xlab(myxlab4) +
  ylab(myylab4) +
  ggtitle(mytitle4)
plot(g4)
```



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

```
activity4_mean <- mean(activity4_sum_daily$Sum_Daily_Steps)
activity4_median <-  median(activity4_sum_daily$Sum_Daily_Steps)
```

```
## [1] "average of steps taken per day is "
## [2] "10766"
```

```
## [1] "median of of total steps taken per day taken is"
## [2] "10766"
```

Do these values differ from the estimates from the first part of the assignment?

The difference is less than 1/2 step, possibly because of the imputation method (simple mean/median).

What is the impact of imputing missing data on the estimates of the total daily number of steps?

The dates with missing values are added to the plots. Whether the subject took any steps on those days is uknown.

5.0 Are there differences in activity patterns between weekdays and weekends?

5.1 Create a new factor variable in the dataset with two levels – “weekday” and “weekend” indicating whether a given date is a weekday or weekend day.

1. Process data from the imputed dataset in part 4 of this project.
2. Assign the string “workday” to all values in a new column workday_or_weekend
3. Use the match operator %in% on the weekday column to return the locations of the strings Sat or Sun. weekend returns the logical values (TRUE or FALSE) depending on whether the string is Sat or Sun.
4. Convert the string values to factors. Factors are usually easier for data manipulation.

```
activity5 <- activity4_imputed
activity5$workday_or_weekend <- "workday"
weekend <- activity5$weekday %in% c("Sat", "Sun")
activity5$workday_or_weekend[weekend] <- "weekend"
activity5$workday_or_weekend <- as.factor(activity5$workday_or_weekend)
```

5.2 Make a panel plot containing a time series plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all weekday days or weekend days (y-axis).

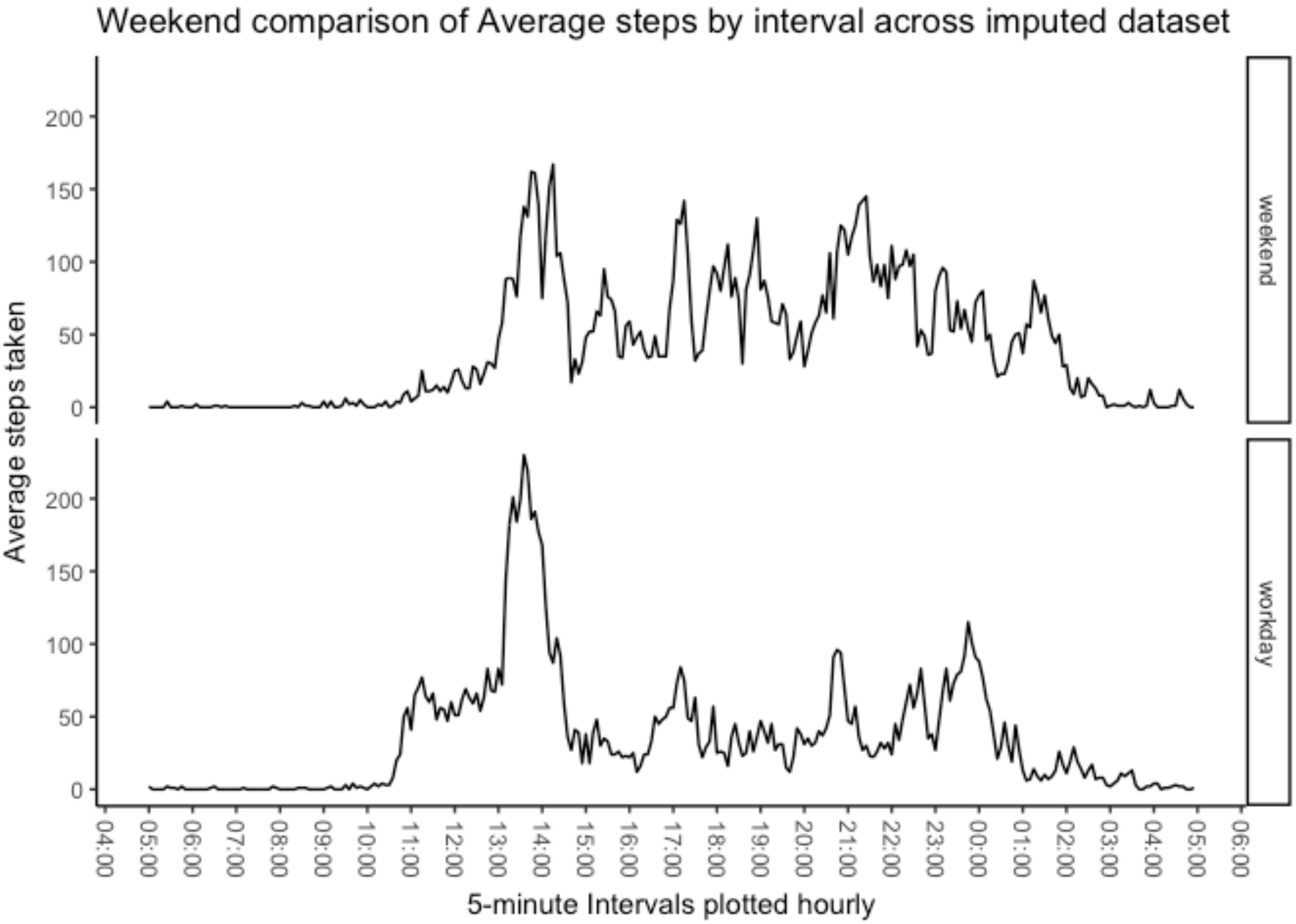
Group dataset with imputed NA values by date and interval.
Calculate the total average of steps taken per day with imputed NA values.

```
activity5_interval_avg <- activity5 %>%
  group_by(workday_or_weekend, interval_dt) %>%
  summarise(Average_Steps_by_Interval = round(mean(steps)))
```

Plot

facet_grid forms a matrix of panels defined by row and column facetting variables. It is most useful when you have two discrete variables, and all combinations of the variables exist in the data. Variables are workday or weekend in the workday_or_weekend column created in the preceeding step.
geom_line() is suitable for time series, linetype is solid

```
g5 <- ggplot(data = activity5_interval_avg, aes (x=interval_dt, y = Average_Steps_by_
Interval)) +
  geom_line(linetype = 1) +
  facet_grid(workday_or_weekend~.)
myxlab5 <- "5-minute Intervals plotted hourly"
myylab5 <- "Average steps taken"
mytitle5 <- "Weekend comparison of Average steps by interval across imputed dataset"
g5 <- g5 +  xlab(myxlab5) +
  ylab(myylab5) +
  ggtitle(mytitle5)
g5 <- g5 + theme_classic()
g5 <- g5 +  scale_x_datetime(breaks = date_breaks("1 hours"),
                             labels = date_format("%H:%M"))
g5 <- g5 + theme(axis.text.x = element_text(angle = 270, hjust = 1, vjust = 0.5, size
= 9))
plot(g5)
```


Weekend comparison of Average steps by interval across imputed dataset
Average steps taken
weekend
workday
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
5-minute Intervals plotted hourly