Reproducible Research: Course Project 1

Chris Shattock

6 January 2019

Objectives

Some sample activity monitoring data in CSV format is supplied as a zipped file. With this data, we have the following tasks:

1. Code for reading in the dataset and/or processing the data
2. Histogram of the total number of steps taken each day
3. Mean and median number of steps taken each day
4. Time series plot of the average number of steps taken
5. The 5-minute interval that, on average, contains the maximum number of steps
6. Code to describe and show a strategy for imputing missing data
7. Histogram of the total number of steps taken each day after missing values are imputed
8. Panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends

We make use of the following libraries:

```
library(data.table)  # Encapsulate data in table format
library(dplyr)       # Use table related mainoulation functions
library(ggplot2)     # Plot generation
library(xtable)      # Formatting tabular R output as HTML.
library(tidyverse)   # Extra functionality for data tables
library(purrr)       # Functional extensions for R
library(missForest)  # For imputing missing values
```

Loading and preprocessing the data

To load the data (and cache it) we:

1. Create a temporary file in the user's system.
2. Download the given URL into the temporary file.
3. Formulate a shell/system command to unzip the target download file for piping into the `fread` function.
4. Read the file into the `data` variable.
5. Close and release the temporary file.

The following code performs this process:

```
tempFile <- tempfile()
download.file(
  'https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip', tempFile)
unzipFile <- sprintf('unzip -q -p "%s"', tempFile)
data <- fread(cmd = unzipFile)
unlink(tempFile)
```

The loaded data table has 17568 rows and 3 columns. The structure of the data is as follows:

```
str(data)
```

```
## Classes 'data.table' and 'data.frame':   17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : chr  "2012-10-01" "2012-10-01" "2012-10-01" "2012-10-01" ...
```

```
##  $ interval: int  0 5 10 15 20 25 30 35 40 45 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The column `date` is of character format, so we use `dplyr` to mutate the data into the R `Date` format:

```
data <- mutate(data, date=as.Date(date))
```

The class of the column `date` is now reported as Date.

This completes the load/preprocessing requirement of objective 1.

What is mean total number of steps taken per day?

From the foregoing there are a number of missing values in the data for the column `steps`, indeed, we have the table with these counts for missing values (*IsNA*) generated via:

```
xt <- xtable(count(data, IsNA = is.na(data$steps)))
print(xt, type='html',
      html.table.attributes=
        'style="text-align: center; margin-left: auto; margin-right: auto;"')
```

IsNA

n

1

FALSE

15264

2

TRUE

2304

For this part of the assignment, you can ignore the missing values in the dataset.

The total number of steps taken in each day is evaluable by grouping the data via the day part of the `date` column. However, the data contains the distinct months of October and November, so we should group by the entire date so as to not then summarize steps per day *across* months. In order to do this we:

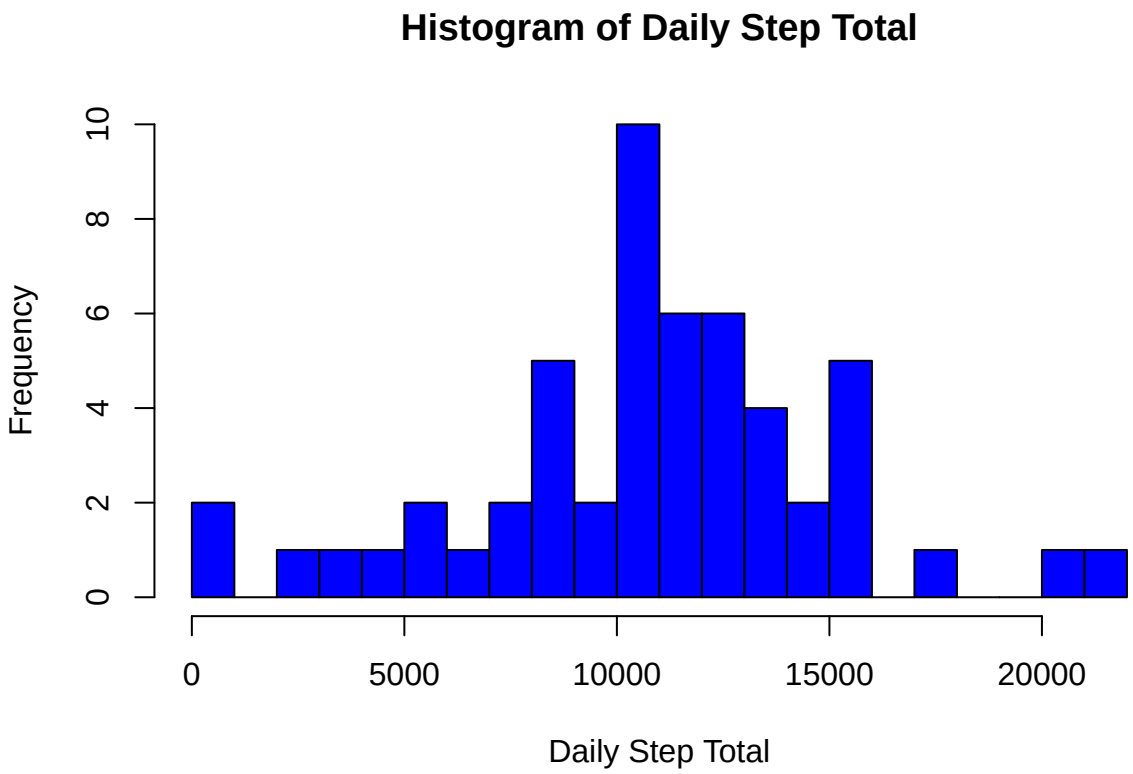
1. Filter our `data` to remove `NA`s and pipe the output to...
2. Select only the columns of `date` and `steps` and pipe the output to...
3. Undertake a grouping of the subset via the `date` column, then pipe the detail grouped rows to...
4. Summarise the detail data whilst evaluating the total number of steps - a new column named `TotalSteps` using the `sum` function.

This code is as follows:

```
dailyStepTotals <-
  filter(data, !is.na(steps)) %>%
  select(date, steps) %>%
  group_by(date) %>%
  summarise(TotalSteps = sum(steps))
```

We can plot a histogram of the total number of steps recorded per day via:

```
lab <- 'Daily Step Total'
chart <- hist(dailyStepTotals$TotalSteps, breaks=20,
              col = 'blue', xlab=lab,
              main = paste("Histogram of", lab))
```



The histogram has 23 breaks of width 1000 between 0 and 22000. The maximum frequency of 10 occurs for the bar datum from 10000.

The median and mean of the total number of daily steps taken is yielded and reported via:

```
summary(dailyStepTotals$TotalSteps)[3:4]
```

```
##   Median     Mean
## 10765.00 10766.19
```

This completes the requirements of objectives 2 and 3.

What is the average daily activity pattern?

In this case we wish to group data using the `interval` column and to summarise using the mean of the `steps` column. However, if we are just using five minute interval data, from midnight, then we can transform the `interval` into the time of the day from midnight at which each interval starts. To do this, we'll create the vector:

```
intervalTimes <- as.POSIXct(head(
  format(seq.POSIXt(as.POSIXct(Sys.Date()),
         as.POSIXct(Sys.Date()+1),
         by = "5 min"), "%H:%M", tz="GMT"), -1), format="%H:%M")
```

Now, for the data set required:

1. Filter our `data` to remove `NA`s and pipe the output to...
2. Select only the columns of `interval` and `steps` and pipe the output to...
3. Undertake a grouping of the subset via the `interval` column, then pipe the detail grouped rows to...

4. Summarise the detail data whilst evaluating the mean of steps - a new column named `MeanSteps` using the `mean` function. Pipe the output to...
5. Now append the column which has the time of day from which each five minute interval starts, then pipe the output to...
6. Selection only the columns required of `TimeOfDay` and `MeanSteps`.

This code is as follows:

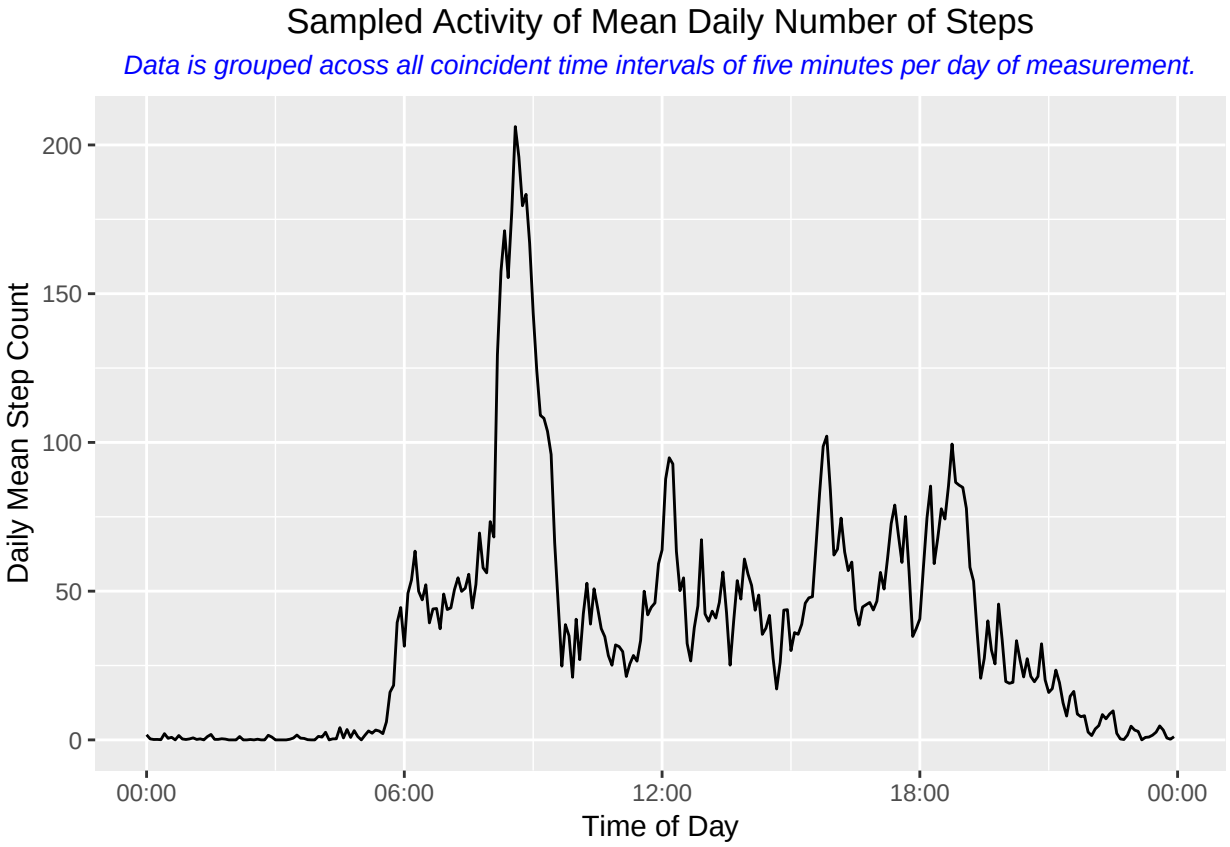
```
stepsByInterval <-
  filter(data, !is.na(steps)) %>%
  select(interval, steps) %>%
  group_by(interval) %>%
  summarise(MeanSteps = mean(steps)) %>%
  mutate(TimeOfDay = intervalTimes) %>%
  select(TimeOfDay, MeanSteps)
```

We can now plot this data for which, using `ggplot`, the pertinent points are:

- Re-specification of axis labels and titles - which will be centre-aligned.
- Modify the x-axis scale to show just the time of day for its labels.

```
g <- ggplot(stepsByInterval, aes(TimeOfDay,MeanSteps)) +
  geom_line()  +
  labs(x="Time of Day", y="Daily Mean Step Count",
       title="Sampled Activity of Mean Daily Number of Steps",
       subtitle=paste("Data is grouped acoss all coincident time",
                      "intervals of five minutes per day of",
                      "measurement.")) +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5,color="blue",
        size=10, face="italic"),
        legend.position="none") +
  scale_x_datetime(date_labels = "%H:%M")
```

This yields the plot:



Finally, we can determine from the persisted `g$data` variable, where the maximum number of mean daily steps occurs - this yields an element number into the variable `maxIntervalNumber`. Using this element number, we may then review the data for that specific plotted point:

```
maxIntervalNumber <-
  which(g$data$MeanSteps == max(g$data$MeanSteps))
g$data[maxIntervalNumber,]$TimeOfDay
```

Using the foregoing, it transpires that the mean daily step maximum occurs in the five minute interval commencing at 08:35 in the morning.

This completes the requirements of objectives 4 and 5.

Imputing missing values

As reported previously, the row counts for missing data (*IsNA = TRUE*) and valid data for the column `steps` may be tabulated as:

IsNA

n

1

FALSE

15264

2

TRUE

2304

For this task we will use the `missForest` package which uses *Nonparametric Missing Value Imputation using Random Forest...*

> `missForest` is used to impute missing values particularly in the case of mixed-type data. It can be used to impute continuous and/or categorical data including complex interactions and nonlinear relations. It yields an out-of-bag (OOB) imputation error estimate. Moreover, it can be run parallel to save computation time

In order to permute missing values in the activity data set we must use categorical variables for our non-numeric data, moreover, `missForest` has a limit of handling of 52 categorical variables. Consequently, we need to partition the data by day to limit factor levels to 31 at maximum. This partitioning, imputation and re-glueing of data is a three step process, as depicted in the following code for which the details are:

1. With the data which includes missing values, we group it using the month of the `date` column then, using the functional extension function `nest`, that grouping is partitioned by the grouping key. In this case, we'll have a list of two elements corresponding to each of the distinct month's data in the source.
2. For each list element in the nested (partitioned) data, we mutate the `date` column into a factor thereby limiting the maximum number of factor levels to 31. `missForest` cannot handle such a list nest element, so we then convert the element to a data frame. We can then invoke `missForest` to analyse the data and evaluate missing values. This is done over a number of iterative passes through the data (maximum defaulting to ten) and the output is a two-element list, the first of which `ximp` is the analysed data with missing values replaced.
3. Given the `ximp` list elements for each of the nested input month datasets, we then extract *just* the complete, imputed data from each `missForest` analysis and combine them, row-wise, into the single data set, `dataNoNA` whose structure is thus identical to our original `data` variable. We re-classify, for plotting purposes, the `data` from a factor into a `Date` class.

```
t1 <- data %>% group_by(month(data$date)) %>% nest()
for(i in 1:length(t1$data)){
  t1$data[[i]] <-
    missForest(
      as.data.frame(
        mutate(t1$data[[i]],
               date=as.factor(date))))}
dataNoNA <-
  do.call("rbind",
          t1$data %>%
            map(function(x){x[["ximp"]]})) %>%
  mutate(date=as.Date(date))
rm(t1)
```

Summarising the data allows us to verify that there are no `NA` values in our data set:

```
summary(dataNoNA)
```

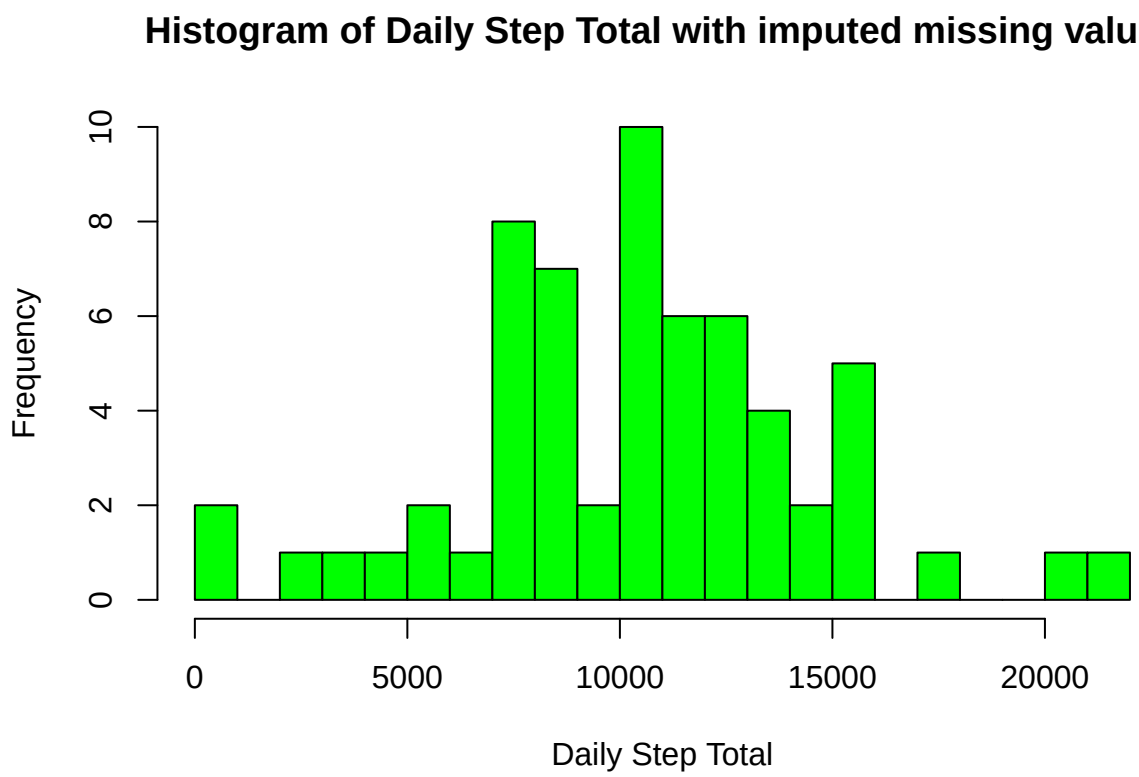
```
##      steps             date               interval
##  Min.   :  0.00   Min.   :2012-10-01   Min.   :   0.0
##  1st Qu.:  0.00   1st Qu.:2012-10-16   1st Qu.: 588.8
##  Median :  0.00   Median :2012-10-31   Median :1177.5
##  Mean   : 35.99   Mean   :2012-10-31   Mean   :1177.5
##  3rd Qu.: 28.00   3rd Qu.:2012-11-15   3rd Qu.:1766.2
##  Max.   :806.00   Max.   :2012-11-30   Max.   :2355.0
```

To produce the required histogram, we simply group by the `date` column and summarise/`sum` the `steps` column:

```
dailyStepTotalsNoNA <- dataNoNA %>%
  group_by(date) %>%
  summarise(TotalSteps = sum(steps))
```

Thence, we plot a histogram of the total number of steps recorded per day for a date set containing no missing values via:

```
hist(dailyStepTotalsNoNA$TotalSteps, breaks=20,
     col = 'green', xlab=lab,
     main = paste("Histogram of", lab,
                  "with imputed missing values"))
```



In terms of comparing the difference in median and men between our original data where `NA` values are ignored, and our imputed data set wherein `NA` values are estimated, we may formulate the following table:

Median

Mean

With Missing Values Ignored

10765

10766.19

With Missing Values Imputed

10395

10366.17

Percentage Change

-6.73%

-9.72%

In the table we see that, in the last row, the effect upon the imputation and replacement of missing values results in a decrease in both the median (-6.73%) and mean (-9.72%) values relative to our data wherein `NA` values are simply ignored in evaluating the media and mean of the total number of steps taken each day.

This completes the requirements of objectives 6 and 7.

Are there differences in activity patterns between weekdays and weekends?

The primary data set for this plot, `allSteps`, is constructed as follows by mutation to form the column `IsWeekday` with subsequent grouping and summarising by `interval` within `IsWeekday` - once again, we convert the intervals to a time of day:

```
allSteps <- dataNoNA %>%
  mutate(IsWeekday = factor(weekdays(date)
                            %in% c("Saturday","Sunday"),
                            labels=c("Weekday","Weekend"))) %>%
  select(IsWeekday, interval, steps) %>%
  group_by(IsWeekday, interval) %>%
  summarise(MeanSteps = mean(steps)) %>%
  mutate(TimeOfDay = intervalTimes) %>%
  select(IsWeekday, TimeOfDay, MeanSteps)
```

As we wish to overlay the plot with the maximum, mean and median values for each facet of `Weeday` and `Weekend`, we create two data frames, `maxima` and `stats` to maintain the distinct, per panel, values of these statistics to use...

```
maxima <-
  allSteps %>%
  filter(TimeOfDay ==
           TimeOfDay[which(MeanSteps == max(MeanSteps))])
stats <- allSteps %>%
  group_by(IsWeekday) %>%
  mutate(Mean=mean(MeanSteps), Median=median(MeanSteps)) %>%
  distinct(IsWeekday,Mean,Median)
```

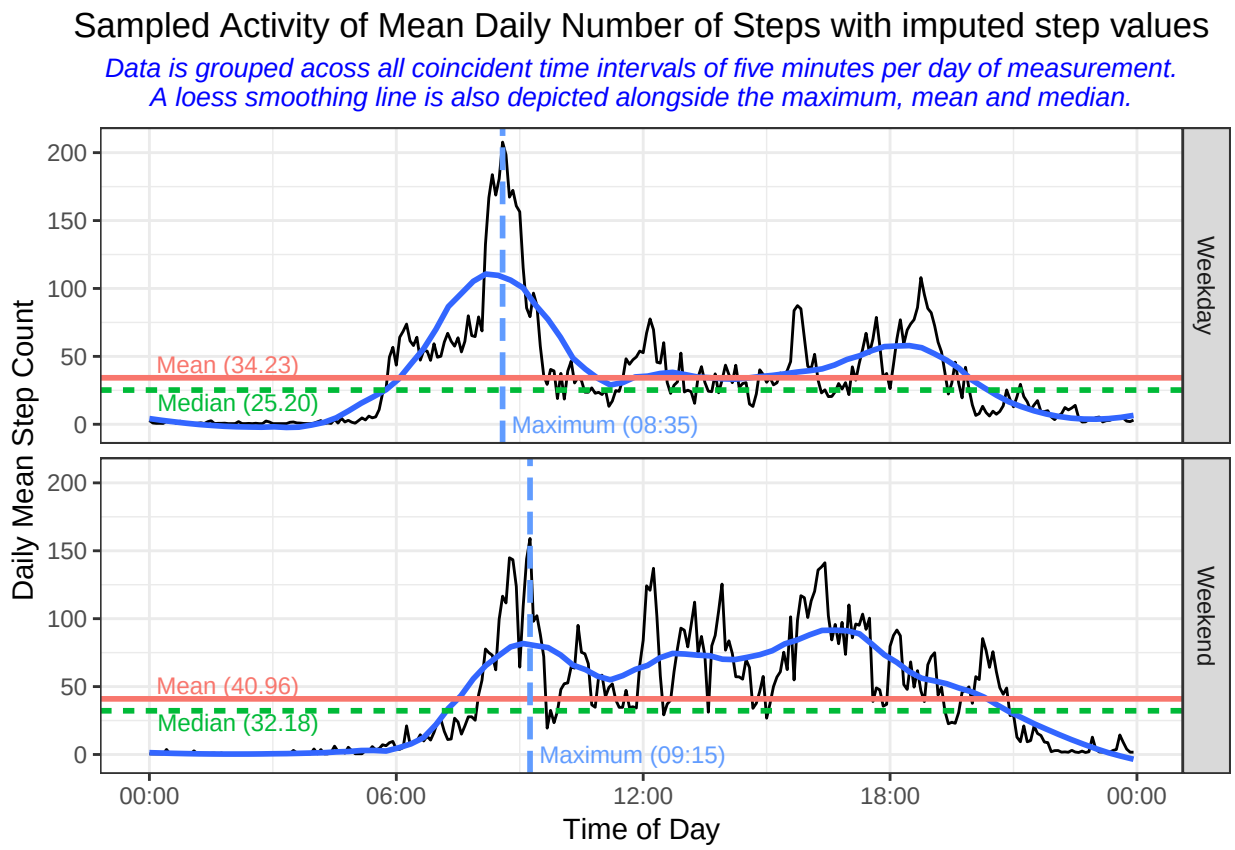
We now construct the plot using the following code in which:

- We set, as before, the basic line plot aesthetic with labelling for the titles and theme variants.
- The data is faceted by the `IsWeekday` column, the time-formatted x-axis labels are set and a *loess* smoothing line is added to the plot.
- We then overlay horizontal/vertical lines to highlight each facet's statistics...
  - For the mean and median horizontal lines we use the prior declared `stats` data to create a horizontal line, then...
    - For the labelling of the line, we create data frame from which each line's source data and labels are derived along the `x` and `y` axis with a label in the `lab` field and the relevant facet in the column `IsWeekday`. Given such a data frame, we may then trivially set the aesthetics for each line to include the required label to which is suffixed a bracketed numeric value for the statistic.
  - For the maximum, vertical line, we use the same principles as for the mean and median lines but the data frame associated with the line is differently populated as is the concatenation of the time of day at which the maximum occurred.

```
g <- ggplot(allSteps, aes(TimeOfDay,MeanSteps)) +
  geom_line(show.legend = FALSE)  +
  labs(x="Time of Day", y="Daily Mean Step Count",
       title=paste("Sampled Activity of Mean Daily Number",
                   "of Steps with imputed step values"),
       subtitle=paste("Data is grouped acoss all coincident time",
                      "intervals of five minutes per day of",
                      "measurement.\nA loess smoothing line is",
                      "also depicted alongside the maximum,",
                      "mean and median.")) +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5,color="blue",
        size=10, face="italic"),
        legend.position="none") +
  facet_grid(IsWeekday~.) +
  scale_x_datetime(date_labels = "%H:%M") +
  geom_smooth(span = 0.3, method="loess", se=FALSE) +
  geom_hline(data=stats,
             mapping=aes(yintercept = Mean,
                         linetype = "Mean", color = "Mean"),
             size=1, show.legend = FALSE) +
  geom_text(data=
              data.frame(x=as.POSIXct("00:00:00",format="%H:%M:%S"),
                         y=stats$Mean,
                         lab=rep("Mean",nrow(stats)),
                         IsWeekday=stats$IsWeekday) %>%
              mutate(lab = sprintf("%s (%.2f)",lab,y)),
            aes(x,y,label=lab,color="Mean"),
            vjust=-.3, hjust=-.05,size=3) +
  geom_hline(data=stats,
             mapping=aes(yintercept = Median,
                         linetype = "Median", color = "Median"),
             size=1, show.legend = FALSE) +
  geom_text(data=
              data.frame(x=as.POSIXct("00:00:00",format="%H:%M:%S"),
                         y=stats$Median,
                         lab=rep("Median",nrow(stats)),
                         IsWeekday=stats$IsWeekday) %>%
              mutate(lab = sprintf("%s (%.2f)",lab,y)),
            aes(x,y,label=lab,color="Median"),
            vjust=1.2, hjust=-.05,size=3) +
  geom_vline(data=maxima,
             mapping=aes(xintercept = TimeOfDay,
                         linetype = "TimeOfDay",
                         color = "TimeOfDay"),
             size=1, show.legend = FALSE) +
  geom_text(data=
              data.frame(x=maxima$TimeOfDay,
                         y=0,
                         lab=rep("Maximum",nrow(maxima)),
                         IsWeekday=maxima$IsWeekday) %>%
              mutate(lab = sprintf(
```

```
            "%s (%s)",
            lab,
            format(maxima$TimeOfDay, format="%H:%M"))),
        aes(x,y,label=lab,color="TimeOfDay"),
        vjust=.5, hjust=-.05,size=3)
```

The foregoing yields the following plot:



This completes the requirements of objective 8.