Reproducible Research Week2 Project

Ethan Schnelle

October 29, 2019

Introduction

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.

This assignment makes use of data from a personal activity monitoring device. This device collects data at 5 minute intervals through out the day. The data consists of two months of data from an anonymous individual collected during the months of October and November, 2012 and include the number of steps taken in 5 minute intervals each day.

Assignment

This assignment will be described in multiple parts. You will need to write a report that answers the questions detailed below. Ultimately, you will need to complete the entire assignment in a single R markdown document that can be processed by knitr and be transformed into an HTML file.

Fork/clone the GitHub repository created for this assignment. You will submit this assignment by pushing your completed files into your forked repository on GitHub. The assignment submission will consist of the URL to your GitHub repository and the SHA-1 commit ID for your repository state.

NOTE: The GitHub repository also contains the dataset for the assignment so you do not have to download the data separately.

Loading and preprocessing the data

The data for this assignment can be downloaded from the course web site: Activity monitoring data [52K] . The dataset is stored in a comma-separated-value (CSV) file and there are a total of 17,568 observations in this dataset.

Show any code that is needed to:

- Load the data **read.csv()**
- Process/transform the data (if necessary) into a format suitable for your analysis

Answer and do the following for this project

What is mean total number of steps taken per day?

1. Mean/Median and Total Steps Per Day? - For this part of the assignment, you can ignore the missing values in the dataset.
2. Make a histogram of the total number of steps taken each day
3. What is the average daily activity pattern? Calculate and report the mean and median of the total number of steps taken per day

What is the average daily activity pattern?

1. Make a time series plot **type="l"** of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days (y-axis) Which 5-minute interval, on average across all the days in the dataset, contains the maximum number of steps? Imputing missing values Note that there are a number of days/intervals where there are missing values **NA** . The presence of missing days may introduce bias into some calculations or summaries of the data.

Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with **NA** Devise a strategy for filling in all of the missing values in the dataset. The strategy does 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. Create a new dataset that is equal to the original dataset but with the missing data filled in. Make a histogram of the total number of steps taken each day 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 imputing missing data on the estimates of the total daily number of steps? Are there differences in activity patterns between weekdays and weekends? For this part the **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 vs. weekend indicating whether a given date is a weekday or weekend day. Make a panel plot containing a time series plot **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 should look like using simulated data.

Submitting the Assignmentless To submit the assignment:

Commit your completed **PA1_template.Rmd** file to the **master branch** of your git repository (you should already be on the **master branch** unless you created new ones) Commit your PA1_template.md and PA1_template.html files produced by processing your R markdown file with knit2html() function in R (from the knitr package) by running the function from the console. If your document has figures included (it should) then they should have been placed in the figure/ directory by default (unless you overrided the default). Add and commit the figure/ directory to your git repository so that the figures appear in the markdown file when it displays on github. Push your **master branch** to GitHub. Submit the URL to your GitHub repository for this assignment on the course web

site. In addition to submitting the URL for your GitHub repository, you will need to submit the 40 character SHA-1 hash (as string of numbers from 0-9 and letters from a-f) that identifies the repository commit that contains the version of the files you want to submit. You can do this in GitHub by doing the following

Going to your GitHub repository web page for this assignment Click on the **commits** link where ?? is the number of commits you have in the repository. For example, if you made a total of 10 commits to this repository, the link should say **commits**.

You will see a list of commits that you have made to this repository. The most recent commit is at the very top. If this represents the version of the files you want to submit, then just click the **copy to clipboard** button on the right hand side that should appear when you hover over the SHA-1 hash. Paste this SHA-1 hash into the course web site when you submit your assignment. If you don't want to use the most recent commit, then go down and find the commit you want and copy the SHA-1 hash.

Setting global options and loading required libraries

```
# Set Working Directory
setwd("~/Documents/Personal/Coursera/Reproducible Research/Week2 Project")

# required r libraries
library(knitr)

## Warning: package 'knitr' was built under R version 3.4.4

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.4.4

library(data.table)

## Warning: package 'data.table' was built under R version 3.4.4

library(reshape2)

## Warning: package 'reshape2' was built under R version 3.4.4

##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.4.4

## -- Attaching packages ---------------------------------- tidyverse 1.2.1 -
-
```

```
## v tibble  2.1.1       v purrr   0.2.5
## v tidyr   0.8.1       v dplyr   0.8.0.1
## v readr   1.1.1       v stringr 1.3.1
## v tibble  2.1.1       v forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.4.4

## Warning: package 'tidyr' was built under R version 3.4.4

## Warning: package 'readr' was built under R version 3.4.4

## Warning: package 'purrr' was built under R version 3.4.4

## Warning: package 'dplyr' was built under R version 3.4.4

## Warning: package 'stringr' was built under R version 3.4.4

## Warning: package 'forcats' was built under R version 3.4.4

## -- Conflicts ---------------------------------- tidyverse_conflicts() -
-
## x dplyr::between()   masks data.table::between()
## x dplyr::filter()    masks stats::filter()
## x dplyr::first()     masks data.table::first()
## x dplyr::lag()       masks stats::lag()
## x dplyr::last()      masks data.table::last()
## x purrr::transpose() masks data.table::transpose()

library(lattice)

library (readr) # for GitHub CSV read
library(httr)

## Warning: package 'httr' was built under R version 3.4.4

opts_chunk$set(echo = TRUE, results = 'hold')
```

Read in Activity Data (Downloaded and Stored on GitHub Repository) and Extract

```
# Data Stored on GitHub
download_urlxx='https://raw.githubusercontent.com/ESchnelle/Reproducible-
Research-Week2-Project/master/activity.csv'
# Load CSV File
activity_data <-as.data.frame(read_csv(url(download_urlxx)))

## Parsed with column specification:
## cols(
##   steps = col_integer(),
##   date = col_date(format = ""),
##   interval = col_integer()
## )
```

```r
#activity_data <- read.csv("activity.csv" , header = TRUE)

# Top of File
head (activity_data)

# Structure of Data
str(activity_data)

#summary of data
summary(activity_data)
```

```
##   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
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: int  0 5 10 15 20 25 30 35 40 45 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 3
##   .. ..$ steps   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ date    :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
##   .. ..$ interval: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
##      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   : 37.38   Mean   :2012-10-31   Mean   :1177.5
##  3rd Qu.: 12.00   3rd Qu.:2012-11-15   3rd Qu.:1766.2
##  Max.   :806.00   Max.   :2012-11-30   Max.   :2355.0
##  NA's   :2304
```

The variables included in this dataset are:

- steps: Number of steps taking in a 5-minute interval (missing values are coded as NA)
- date: The date on which the measurement was taken in YYYY-MM-DD format. May need to convert to date, vs. factor default
- interval: Identifier for the 5-minute interval in which measurement was taken

Convert Variables to More Appropriate Types

```
# convert date to date vs. factor as is now
activity_data$date <- as.Date(as.character(activity_data$date) , "%Y-%m-%d")

# convert interval, now seen as integer, to a factor variable
activity_data$interval <- as.factor(activity_data$interval)

# take a look at the new structure
str(activity_data)

#look at top records again after conversion
head(activity_data)
```

```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: Factor w/ 288 levels "0","5","10","15",..: 1 2 3 4 5 6 7 8 9
10 ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 3
##   .. ..$ steps   : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ date    :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
##   .. ..$ interval: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
##   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
```

What is mean total number of steps taken per day?

NOTE: Ignoring Missings in the dataset at this point.

1. Calculate the total number of steps taken per day

Summarize Total Steps for Each Day

```
# sum up total steps for each day from raw data, regardless of NA
steps_daily <- aggregate(steps ~ date, data=activity_data, FUN=sum, na.rm =
TRUE)
```

```r
#change column names to be descriptive in output
colnames(steps_daily) <- c("date", "steps")

# show top records in number of steps table
head(steps_daily)
```

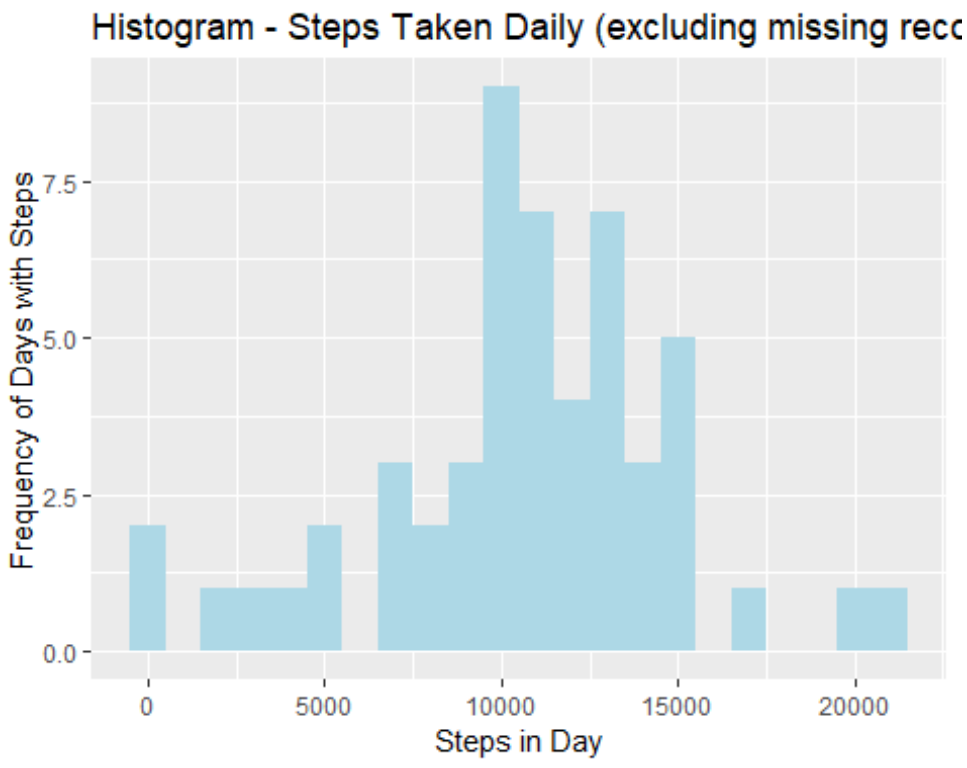
```
##         date steps
## 1 2012-10-02   126
## 2 2012-10-03 11352
## 3 2012-10-04 12116
## 4 2012-10-05 13294
## 5 2012-10-06 15420
## 6 2012-10-07 11015
```

2. Histogram showing total number steps taken per day

NOTE: Ignoring Missings in the dataset

Create a Histogram of the Total Steps Each Day

```r
ggplot(steps_daily, aes(x = steps)) +
  geom_histogram(fill = "light blue", binwidth = 1000) +
  labs(title = "Histogram - Steps Taken Daily (excluding missing records)", x
= "Steps in Day", y = "Frequency of Days with Steps")
```



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

```
#calculate mean and median steps from raw data regardless of NA
#summary of steps daily
summary(steps_daily)
```

```
##       date               steps
##  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
```

Date Range for Activity Observations: 10/02/2012 to 11/29/2012 ##### **ANSWER: Mean and Median Steps Daily** Mean Steps Daily = 10,766 Median Steps Daily = 10,765 ##### **NOTE: Mean and Median are very close indicating a less likelihood of outliers in the data to be concerned with.**

What is the 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)

NOTE: REMOVING NA RECORDS

```
# average steps per 5 min interval period - removing records using NA (na.rm
= TRUE)
steps_interval               <- aggregate(steps ~ interval, data =
activity_data, FUN = mean, na.rm = TRUE)
# convert steps to integer
steps_interval$interval      <-
as.integer(levels(steps_interval$interval)[steps_interval$interval])

# summary of intervals
summary(steps_interval)

# define field names in output
colnames(steps_interval) <- c("interval", "steps")

# show top 5 in steps_interval
head(steps_interval)
```

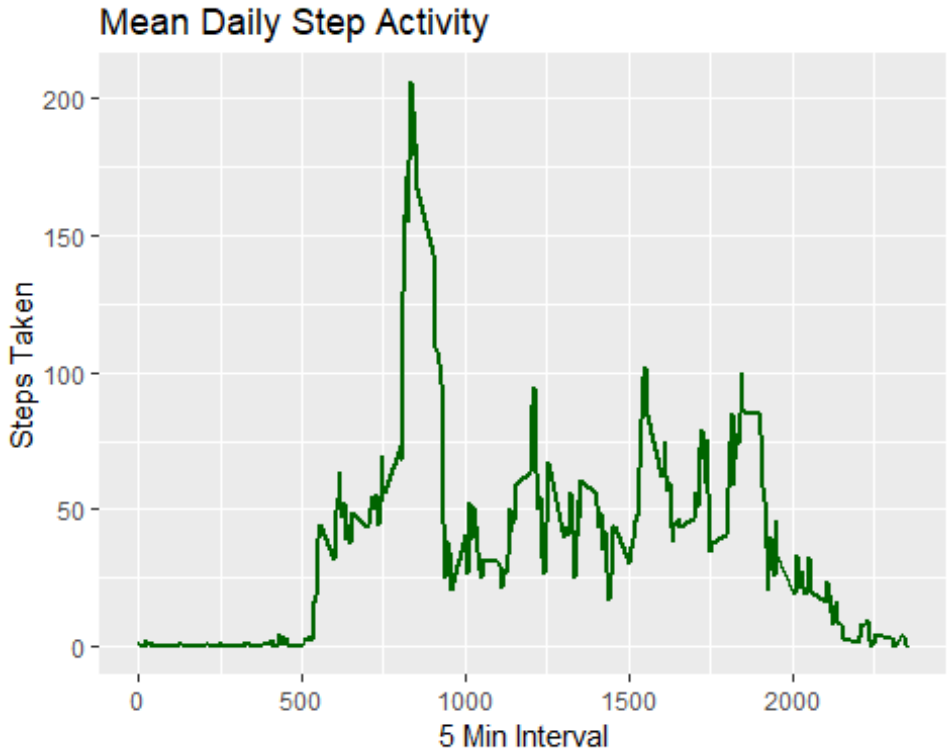
```
##     interval          steps
##  Min.   :   0.0   Min.   :  0.000
##  1st Qu.: 588.8   1st Qu.:  2.486
##  Median :1177.5   Median : 34.113
##  Mean   :1177.5   Mean   : 37.383
##  3rd Qu.:1766.2   3rd Qu.: 52.835
##  Max.   :2355.0   Max.   :206.170
##   interval     steps
```

```
## 1        0 1.7169811
## 2        5 0.3396226
## 3       10 0.1320755
## 4       15 0.1509434
## 5       20 0.0754717
## 6       25 2.0943396
```

Note: There are up 2,355 5 minute intervals in the data

Plot the timeseries graph

```
# using ggplot show time series of interval data graphically
ggplot(steps_interval, aes(x = interval, y = steps)) +
  geom_line(col = "dark green", size = 1) +
  labs(title = "Mean Daily Step Activity", x = "5 Min Interval", y = "Steps
Taken")
```



NOTE: It appears that mornings are popular for step activity

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

```
# using r which.max, identify interval with highest number of steps
highest_step_5min_interval <-
steps_interval[which.max(steps_interval$steps),]

# print out highest step interval
highest_step_5min_interval
```

```
##     interval    steps
## 104      835 206.1698
```

ANSWER: The 835th 5 minute interval showed the highest number of steps with 206.2 steps

Imputing missing values

```
 ***What happens if we replace the missing values in our data (imputation)***
```

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

```
# How many records in the data have missing values
na_records_for_steps <- sum(is.na(activity_data$steps))
# print how many records have missing/na with steps
na_records_for_steps
```

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

ANSWER: Confirming our early summary during data load showed 17,568 obs. in the raw data and 2,304 "NA" records and that it is specific to Steps.

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

AND

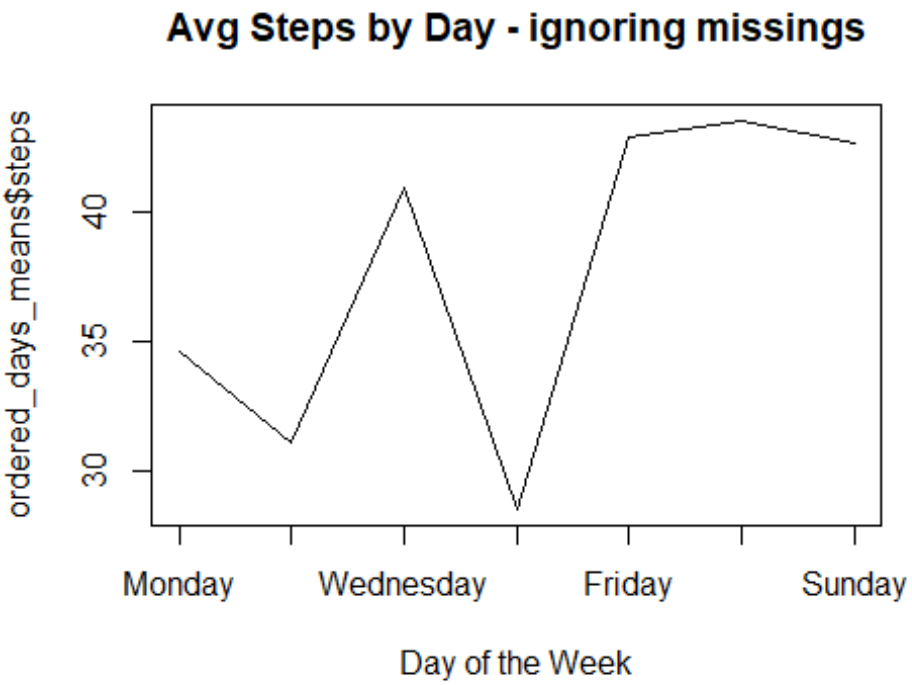
3. Create a new dataset that is equal to the original dataset but with the missing data filled in.

Look for a reasonable value to use to impute for missing step values?

First see if there it is reasonable to impute missing/NA date values with day of week average steps. This seems like a reasonable approach given that logically people follow a day of week pattern for activities

```
# Find Day of Week step Pattern

# Add day of the week as a column based on date of interval
activity_data$day_of_week <- weekdays(activity_data$date)

# Average Steps by Day of Week - leaving our NA records with na.omit
mean_steps_by_day <- aggregate(steps ~ day_of_week, data=activity_data, mean,
na.action = na.omit)


# Print Average Steps by Day of Week
mean_steps_by_day
# Plot Average Steps by Day of Week
ordered_days_means <- mean_steps_by_day [ c(2,6,7,5,1,3,4),]
plot.ts (ordered_days_means$steps,xaxt = "n", xlab = "Day of the Week",
main='Avg Steps by Day - ignoring missings')
```

```
# Rename the labels for x-axis
axis(1, at=1:7, labels=c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))
```



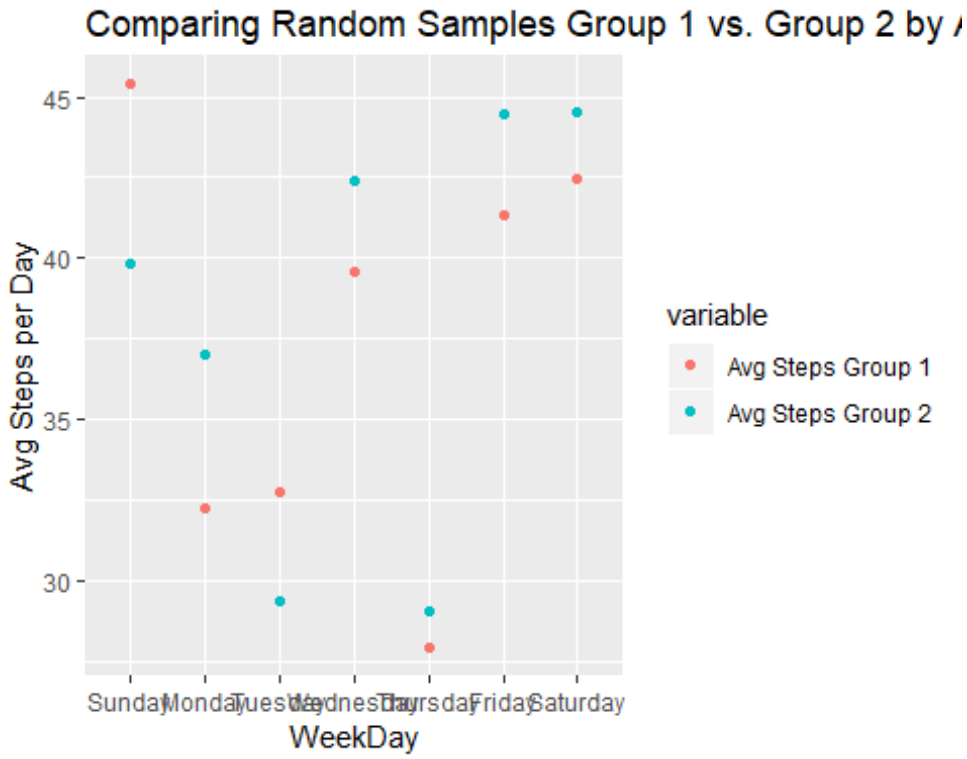

```
##   day_of_week    steps
## 1      Friday 42.91567
## 2      Monday 34.63492
## 3    Saturday 43.52579
## 4      Sunday 42.63095
## 5    Thursday 28.51649
## 6     Tuesday 31.07485
## 7   Wednesday 40.94010
```

** NOTE: It again seems reasonable to use average steps per weekday to impute for days with missing steps, presuming that people follow a typical day of week pattern for activities**

Thoughts about Using Day of Week Mean Replacement as Imputation Method: Given that using Day of Week mean as imputation for missing values seems reasonable, let's test further by randomly samping the dataset into two and seeing if random samples have reasonably same mean values for Steps by Day of Week.

```
# remove NA rows from activity_data raw data
activity_data_no_na   <- na.omit(activity_data)

## 50% Split of list for intra-comparison of steps by day of week
smp_size <- floor(0.5 * nrow(activity_data_no_na))
```

```r
# set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(activity_data_no_na)), size = smp_size)
group1 <- activity_data_no_na[train_ind, ]
group2 <- activity_data_no_na[-train_ind, ]

mean_steps_by_day_grp1 <- aggregate(steps ~ day_of_week, data=group1, mean)
mean_steps_by_day_grp2 <- aggregate(steps ~ day_of_week, data=group2, mean)
      #, na.action = na.omit removed since done in first step above...

mean_steps_by_day_by_grp <-
as.data.frame(cbind(mean_steps_by_day_grp1$day_of_week,
as.character(mean_steps_by_day_grp1$steps),
as.character(mean_steps_by_day_grp2$steps)))

colnames(mean_steps_by_day_by_grp) <- c("day_of_week", "steps_grp1",
"steps_grp2")

mean_steps_by_day_by_grp$steps_grp1 <-
as.numeric(as.character(mean_steps_by_day_by_grp$steps_grp1))

mean_steps_by_day_by_grp$steps_grp2 <-
as.numeric(as.character(mean_steps_by_day_by_grp$steps_grp2))

#
mean_steps_by_day_by_grp$day_of_week <-
factor(mean_steps_by_day_by_grp$day_of_week, levels = c("Sunday", "Monday",
"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))

mean_steps_by_day_by_grp[order(mean_steps_by_day_by_grp$day_of_week),]

ggplot(mean_steps_by_day_by_grp, aes(day_of_week, y = steps_grp1, color =
variable))  +
       geom_point(aes(y = steps_grp1, col = "Avg Steps Group 1"))   +
       geom_point(aes(y = steps_grp2, col = "Avg Steps Group 2")) +
       labs(title = "Comparing Random Samples Group 1 vs. Group 2 by Avg
Steps by Weekday - excluding missings") +
       ylab("Avg Steps per Day") +
       xlab("WeekDay")
```




```
##   day_of_week steps_grp1 steps_grp2
## 4      Sunday   45.40337   39.85303
## 2      Monday   32.27056   37.00397
## 6     Tuesday   32.75194   29.42343
## 7   Wednesday   39.55942   42.38455
## 5    Thursday   27.95296   29.07612
## 1      Friday   41.35394   44.46199
## 3    Saturday   42.49449   44.53884
```

Choice on Imputation Method for Missing Step Values: It appears that average steps by weekday is consistent across two randomly drawn samples from the population, so using weekday average to impute for missing steps by day of week is appropriate.

Now to Impute for Missing Step Values using Day of Week Mean to Replace

```
## Impute with Day of Week Average Steps for Days with Missing Step Values

# Show number of rows missing and top 10, Before Imputation
activity_data_original <- activity_data

# tabulate number of rows not missing data
nrow(na.omit(activity_data_original))
head(activity_data_original, 10)

# Impute/Replace NA Step Value with Day of Week Average Steps
activity_data_imputed <- activity_data
activity_data_imputed$steps[is.na(activity_data_imputed$steps)] <-
```

```
ave(activity_data_imputed$steps, activity_data_imputed$day_of_week,
FUN=function(x)mean(x, na.rm = TRUE))[is.na(activity_data_imputed$steps)]

# add imputed steps as steps_imputed, so both original steps and imputed
steps exist on the original data
activity_data_original$steps_imputed <- activity_data_imputed$steps

#summary of data with and without imputations
summary(activity_data_original$steps)
summary(activity_data_original$steps_imputed)

# Show number of rows missing and top 10, After Imputation
nrow(na.omit(activity_data_imputed))
head(activity_data_imputed, 10)

## [1] 15264
##    steps       date interval day_of_week
## 1     NA 2012-10-01        0      Monday
## 2     NA 2012-10-01        5      Monday
## 3     NA 2012-10-01       10      Monday
## 4     NA 2012-10-01       15      Monday
## 5     NA 2012-10-01       20      Monday
## 6     NA 2012-10-01       25      Monday
## 7     NA 2012-10-01       30      Monday
## 8     NA 2012-10-01       35      Monday
## 9     NA 2012-10-01       40      Monday
## 10    NA 2012-10-01       45      Monday
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00    0.00    0.00   37.38   12.00  806.00    2304
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00   37.57   34.63  806.00
## [1] 17568
##       steps       date interval day_of_week
## 1  34.63492 2012-10-01        0      Monday
## 2  34.63492 2012-10-01        5      Monday
## 3  34.63492 2012-10-01       10      Monday
## 4  34.63492 2012-10-01       15      Monday
## 5  34.63492 2012-10-01       20      Monday
## 6  34.63492 2012-10-01       25      Monday
## 7  34.63492 2012-10-01       30      Monday
## 8  34.63492 2012-10-01       35      Monday
## 9  34.63492 2012-10-01       40      Monday
## 10 34.63492 2012-10-01       45      Monday
```

Summarize Total Steps for Each Day without imputations for Missing Values in Place

```
# sum up total steps for each day from raw data, regardless of NA
steps_daily_before_imputes <- aggregate(steps ~ date,
data=activity_data_original, FUN=sum, na.action = na.pass)
```

```r
#change column names to be descriptive in output
colnames(steps_daily_before_imputes) <- c("date", "steps_original")

#show number of rows of daily steps before imputation
nrow(na.omit(steps_daily_before_imputes))

# show top records in number of steps table
head(steps_daily_before_imputes)
```

```
## [1] 53
##         date steps_original
## 1 2012-10-01             NA
## 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
```

Summarize Total Steps for Each Day with imputations for Missing Values in Place

```r
# sum up total steps for each day with table having imputed values for
missing steps values
steps_daily_after_imputes <- aggregate(steps_imputed ~ date,
data=activity_data_original, FUN=sum, na.action = na.pass)
steps_daily_after_imputes <- aggregate(steps ~ date,
data=activity_data_imputed, FUN=sum)

#change column names to be descriptive in output
colnames(steps_daily_after_imputes) <- c("date", "steps_imputed")

#show number of rows of daily steps after imputation
nrow(na.omit(steps_daily_after_imputes))

# show top records in number of steps table
head(steps_daily_after_imputes)
```

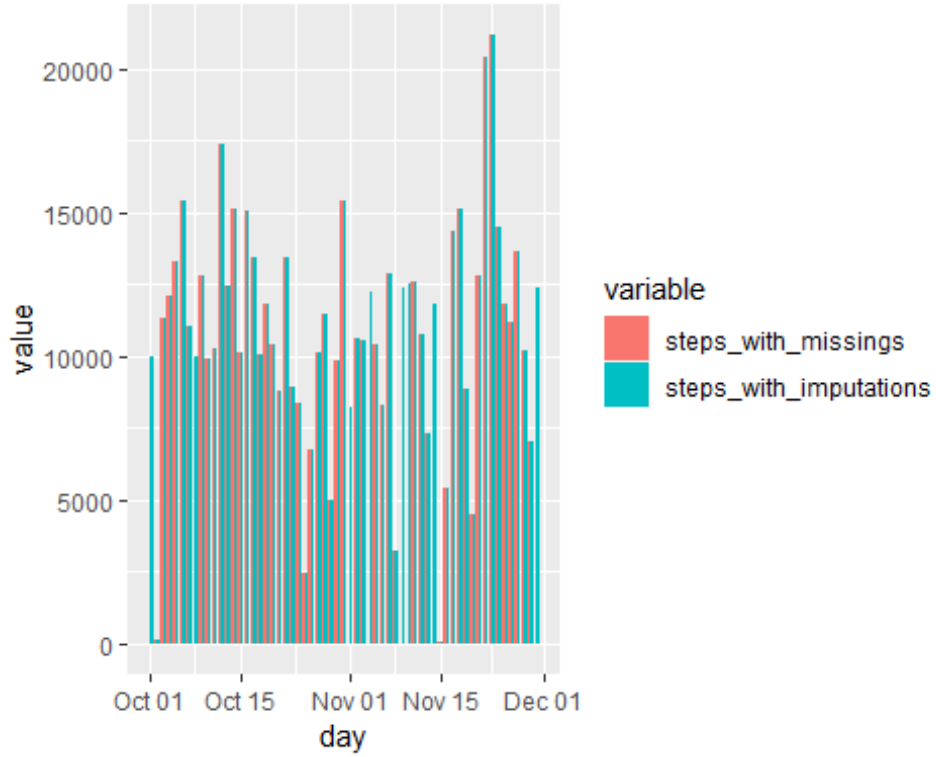
```
## [1] 61
##         date steps_imputed
## 1 2012-10-01      9974.857
## 2 2012-10-02       126.000
## 3 2012-10-03     11352.000
## 4 2012-10-04     12116.000
## 5 2012-10-05     13294.000
## 6 2012-10-06     15420.000
```

FINDING : That 61 days (8 additional days) are now seen in the daily steps data vs. 53 before imputation. So imputation for missing step values had an impact on the number of days with data.

Create a Histogram of the Total Steps Each Day with imputations for Missing Values in Place

```
# Original Daily Summary without Imputations: steps_daily_before_imputes #
column names ("date", "steps_original")
# Original Daily Summary with Imputations   : steps_daily_after_imputes  #
column names ("date", "steps_imputed")

# join with cbind the two daily step totals from original data and imputed
data
steps_daily_compare <-
as.data.frame(cbind(as.character(steps_daily_before_imputes$date),
as.character(steps_daily_before_imputes$steps_original),
as.character(steps_daily_after_imputes$steps_imputed)))
# update names to be appropriate
colnames(steps_daily_compare) <- c("date", "steps_original", "steps_imputed")
# format date character to be a date format
steps_daily_compare$date <- as.Date(as.character(steps_daily_compare$date) ,
"%Y-%m-%d")
# format steps to be numeric values in original daily data
steps_daily_compare$steps_with_missings <-
as.numeric(as.character(steps_daily_compare$steps_original))
# format steps to be numeric values in imputed daily data
steps_daily_compare$steps_with_imputations <-
as.numeric(as.character(steps_daily_compare$steps_imputed))

# add day of week (weekday) to prepare for summary by weekday comparison
steps_daily_compare$day_of_week <-
factor(as.factor(weekdays(steps_daily_compare$date)), levels = c("Sunday",
"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))

# aggregate mean steps and total steps by day of week from original and
imputed series
mean_steps_by_day_missings <- aggregate(steps_with_missings     ~
day_of_week, data=steps_daily_compare, FUN=mean, na.action = na.omit)
mean_steps_by_day_imputed  <- aggregate(steps_with_imputations  ~
day_of_week, data=steps_daily_compare, FUN=mean, na.action = na.omit)
total_steps_by_day_missings <- aggregate(steps_with_missings    ~
day_of_week, data=steps_daily_compare, FUN=sum)
total_steps_by_day_imputed  <- aggregate(steps_with_imputations ~
day_of_week, data=steps_daily_compare, FUN=sum)

# put together into one table
missing_and_imputed_totteps_by_weekday <- as.data.frame(cbind(c("Sunday",
"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"),
```

```
round(as.numeric(as.character(total_steps_by_day_missings$steps_with_missings
)), digits = 0),
round(as.numeric(as.character(total_steps_by_day_imputed$steps_with_imputatio
ns)), digits = 0)) )
colnames(missing_and_imputed_totteps_by_weekday) <- c("day_of_week",
"total_steps_original", "total_steps_imputed")
missing_and_imputed_totteps_by_weekday$Difference <-
round((total_steps_by_day_imputed$steps_with_imputations -
total_steps_by_day_missings$steps_with_missings)  , digits = 0)

# show totals by day of week with missings and after imputing for missings
missing_and_imputed_totteps_by_weekday
# show mean differenc in totals by day of week with missings and after
imputing for missings
round(mean(missing_and_imputed_totteps_by_weekday$Difference) , digits = 0)

df1 <- data.frame(steps_daily_compare$date,
as.numeric(as.character(steps_daily_compare$steps_original)),
as.numeric(as.character(steps_daily_compare$steps_imputed)))
colnames(df1) <- c("day", "steps_with_missings", "steps_with_imputations")
df2 <- melt(df1, id.vars='day')
ggplot(df2, aes(x=day, y=value, fill=variable)) + geom_bar(stat='identity',
position='dodge')

## Warning: Removed 8 rows containing missing values (geom_bar).
```



```
##   day_of_week total_steps_original total_steps_imputed Difference
## 1      Sunday                85944               98222      12278
## 2      Monday                69824               89774      19950
## 3     Tuesday                80546               80546          0
## 4   Wednesday                94326              106117      11791
## 5    Thursday                65702               73915       8213
## 6      Friday                86518              111237      24719
## 7    Saturday                87748              100283      12535
## [1] 12784
```

***FINDING On Imputation Impact:*

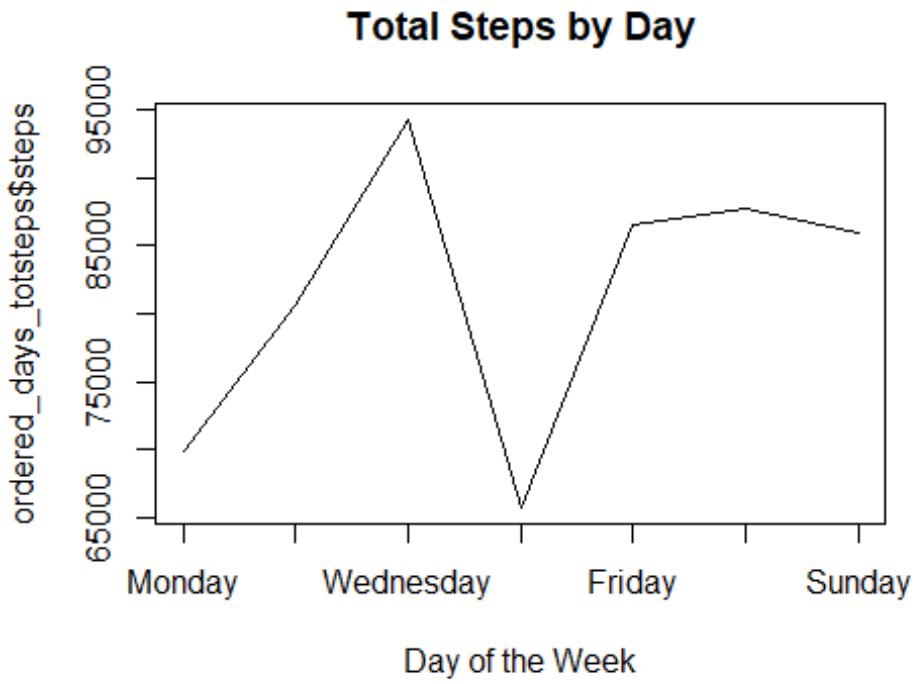
- There are 8 additional days of data replacing missing data with imputations, 61 vs. 53 before imputation.

- There is an impact on total steps but not a visible impact on days already with data.
- New data are imputed for Oct 1st intermittently between Nov 1-15 and at the end of Nov. The addition of data on these days are days in turquoise, days in orignal seriies in red. Looking at impact of imputations by day of week there is a difference in all day's step totals except Tuesday.
- Average day of week increase with imputations was +12,784 steps with highest impact on Friday (+24,719) and lowest Tuesday (+0).**

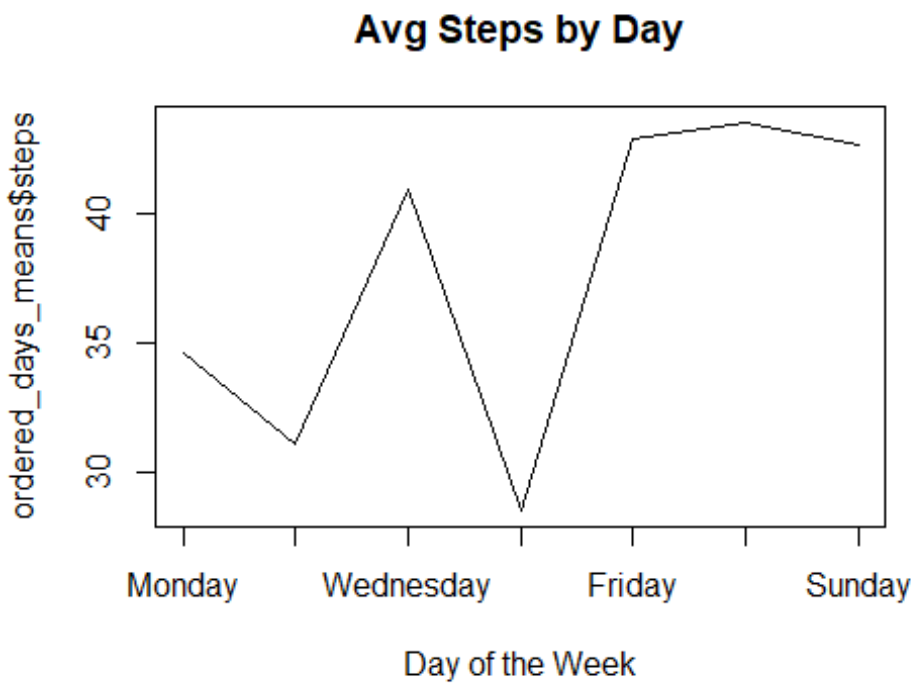
And now rerun statistics with Imputed Values where Step Values are Missing

** NOTE: Imputations for Missing Values Done by Day of Week Mean Replacement **

```
#summary of new data with imputations for NA's
summary(activity_data)

# Sum up Steps by Day of Week - leaving our NA records with na.omit
sum_steps_by_day <- aggregate(steps ~ day_of_week, data=activity_data, sum,
na.action = na.omit)
# Print Total Steps by Day of Week
sum_steps_by_day
# Plot Total Steps by Day of Week
ordered_days_totsteps <- sum_steps_by_day [ c(2,6,7,5,1,3,4),]
plot.ts (ordered_days_totsteps$steps,xaxt = "n", xlab = "Day of the Week",
main='Total Steps by Day')
# Rename the labels for x-axis
axis(1, at=1:7, labels=c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))
```




```
# Average Steps by Day of Week - NA records should now be replaced
mean_steps_by_day <- aggregate(steps ~ day_of_week, data=activity_data, mean,
na.action = na.omit)
# Print Average Steps by Day of Week
mean_steps_by_day
# Plot Average Steps by Day of Week
ordered_days_means <- mean_steps_by_day [ c(2,6,7,5,1,3,4),]
plot.ts (ordered_days_means$steps,xaxt = "n", xlab = "Day of the Week",
main='Avg Steps by Day')
# Rename the labels for x-axis
axis(1, at=1:7, labels=c("Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"))
```




```
##      steps             date               interval     day_of_week
##  Min.   :  0.00   Min.   :2012-10-01   0      :   61   Length:17568
##  1st Qu.:  0.00   1st Qu.:2012-10-16   5      :   61   Class :character
##  Median :  0.00   Median :2012-10-31   10     :   61   Mode  :character
##  Mean   : 37.38   Mean   :2012-10-31   15     :   61
##  3rd Qu.: 12.00   3rd Qu.:2012-11-15   20     :   61
##  Max.   :806.00   Max.   :2012-11-30   25     :   61
##  NA's   :2304                          (Other):17202
##   day_of_week steps
## 1      Friday 86518
## 2      Monday 69824
## 3    Saturday 87748
## 4      Sunday 85944
## 5    Thursday 65702
## 6     Tuesday 80546
## 7   Wednesday 94326
##   day_of_week    steps
## 1      Friday 42.91567
## 2      Monday 34.63492
## 3    Saturday 43.52579
## 4      Sunday 42.63095
## 5    Thursday 28.51649
## 6     Tuesday 31.07485
## 7   Wednesday 40.94010
```

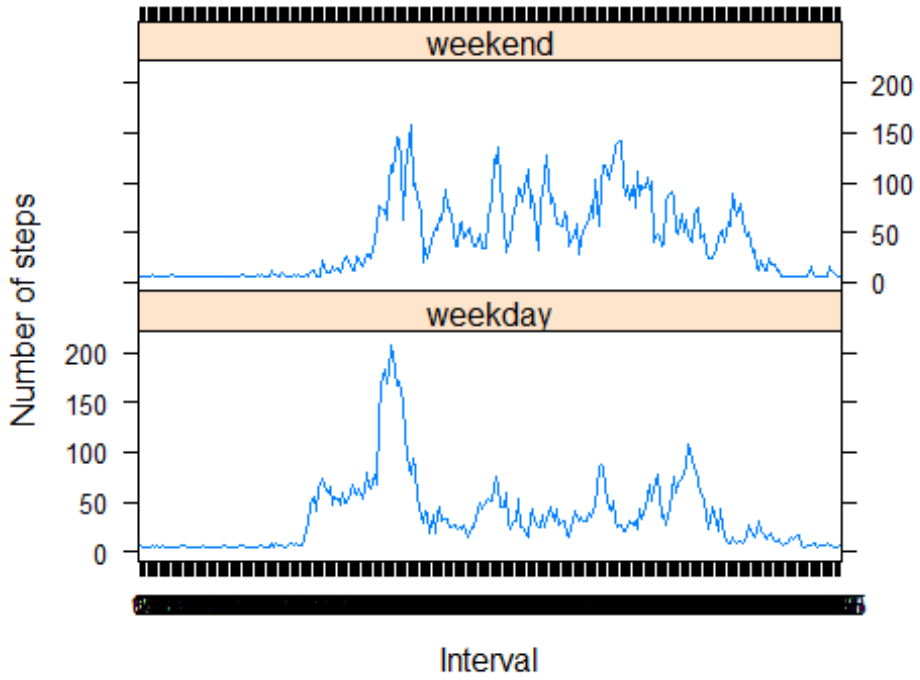
*** FINDING - IMPACT OF IMPUTING FOR MISSING STEP VALUES: Imputations removed NA's and as expected impuations changed total steps by day, but imputation by day did not impact by day of week pattern. ***

Are there differences in activity patterns between weekdays and weekends?

For this part the 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" indicating whether a given date is a weekday or weekend day. 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 should look like using simulated data.

```
#create a weekday vs. weekend field in imputed data table
activity_data_imputed_x <- activity_data_imputed %>%
  mutate(
    weekday = ifelse(weekdays(date) %in% c("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday"), "weekday", "weekend")
  )

# set weekday as factor
activity_data_imputed_x$weekday = as.factor(activity_data_imputed_x$weekday)

#chart weekend and weekday 5 minute time series
smry_data <- aggregate(steps ~ interval + weekday,
data=activity_data_imputed_x, mean)
with(smry_data, xyplot(steps ~ interval | weekday, type="l", xlab =
"Interval", ylab = "Number of steps", layout = c(1, 2)))
```



*** FINDING - There appears to be a difference in steps by interval weekdays and weekends. Weekday has more pronounced step activity early and weekend has activity spread throughout the day. ***