Reproducible Research

thenewdeal

Friday, August 14, 2015

Reproducible Research (Peer assessment 1)

Loading and preprocessing the data

```
## Our first task is data input. The read.csv() function helps us to read a file in table
format and create a data frame from it.

cls = c("integer", "character", "integer")
df <- read.csv("activity.csv", head=TRUE, colClasses=cls, na.strings="NA")

## Returns the first parts of a loaded data frame with the R command of head(). It servic
es us to take the first look of those data, knowing any modification is required.
head(df)
```

```
##   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
```

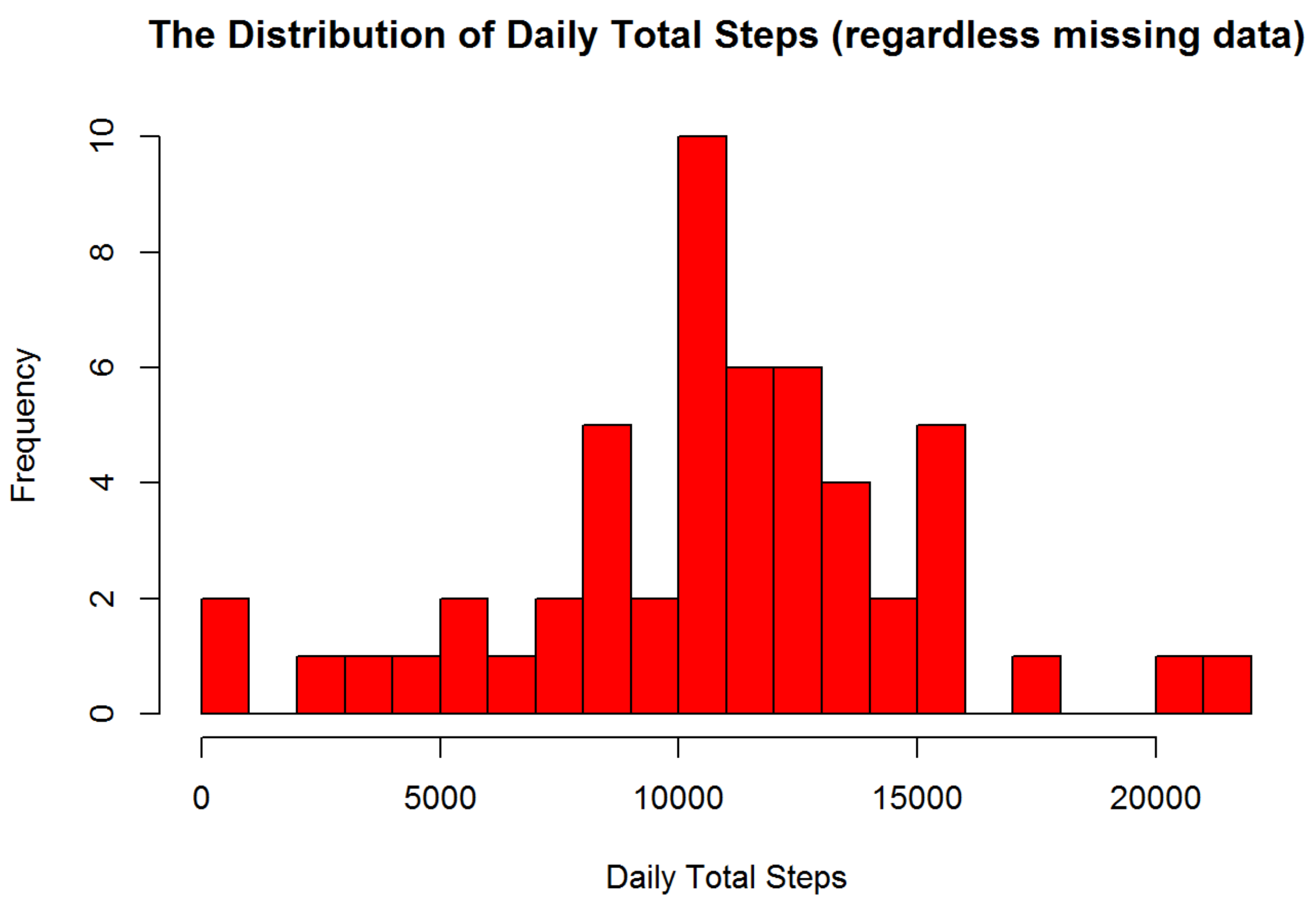
```
## We shall use the command of as.date() to convert between character representations and
"Date" object representing calendar dates. There are also some missing values which we pl
an to ignore, with the assistance of is.na() function.
df$date <- as.Date(df$date)
df_ign <- subset(df, !is.na(df$steps))
```

What is mean total number of steps taken per day?

```
##########################################################################
dailysum <- tapply(df_ign$steps, df_ign$date, sum, na.rm=TRUE, simplify=T)
dailysum <- dailysum[!is.na(dailysum)]
```

```
## We then use hist() function to compute a histogram of the above given data values. The
red color is to be used to fill the bars and the breaking point is 20. We also input the
description to the arguments of main, xlab, ylab, making this diagram more readable. (Ins
ert description is a repeating work for all of the plot/chart below)

hist(x=dailysum,
     col="red",
     breaks=20,
     xlab="Daily Total Steps",
     ylab="Frequency",
     main="The Distribution of Daily Total Steps (regardless missing data)")
```



```
## To compute Arithmetic Mean, we use the function of mean() and retrieve the result of 1
0766.19 steps taken per day.
mean(dailysum)
```

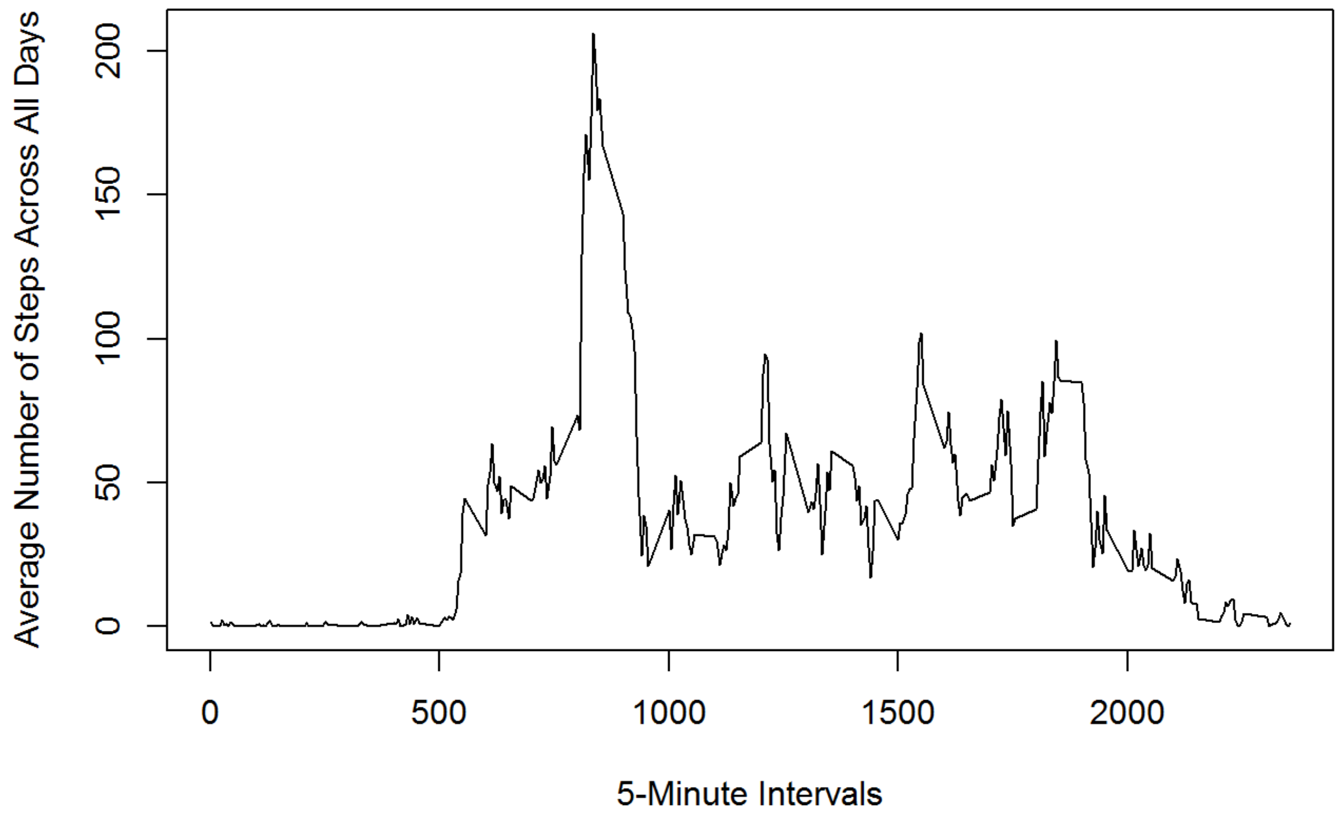
```
## [1] 10766.19
## To obtain the sample median, the median() function responded with the result of 10765
steps taken per day.
median(dailysum)
```

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

What is the average daily activity pattern?

```
##########################################################################
int_avg <- tapply(df_ign$steps, df_ign$interval, mean, na.rm=TRUE, simplify=T)
df_ia <- data.frame(interval=as.integer(names(int_avg)), avg=int_avg)
```

```
# We are going to use tapply() function to make a time series plot of the 5-minute interv
al (x-axis) and the average number of septs taken, average across all days (y-axis).
with(df_ia,
     plot(interval,
          avg,
          type="l",
          xlab="5-Minute Intervals",
          ylab="Average Number of Steps Across All Days"))
```



```
# The next item is to identify the maximum number of steps, with 5-minute interval, on av
erage all of the days in this datasets. The function of which.max() returns the index of
maximum element of a vector. The output shows interval 835 has the maximum number of aver
age steps 206.
max_steps <- max(df_ia$avg)
df_ia[df_ia$avg == max_steps, ]
```

```
##     interval      avg
## 835      835 206.1698
```

Imputing missing values

```
# In the original dataset, there are a number of days/intervals presented as missing valu
es. We shall calculate and report the total number of missing values in the dataset. The
function of is.na() yields with the outcome of 2304, number of rows with missing data.

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

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

```
df_impute <- df
ndx <- is.na(df_impute$steps)
int_avg <- tapply(df_ign$steps, df_ign$interval, mean, na.rm=TRUE, simplify=T)
df_impute$steps[ndx] <- int_avg[as.character(df_impute$interval[ndx])]

new_dailysum <- tapply(df_impute$steps, df_impute$date, sum, na.rm=TRUE, simplify=T)
```

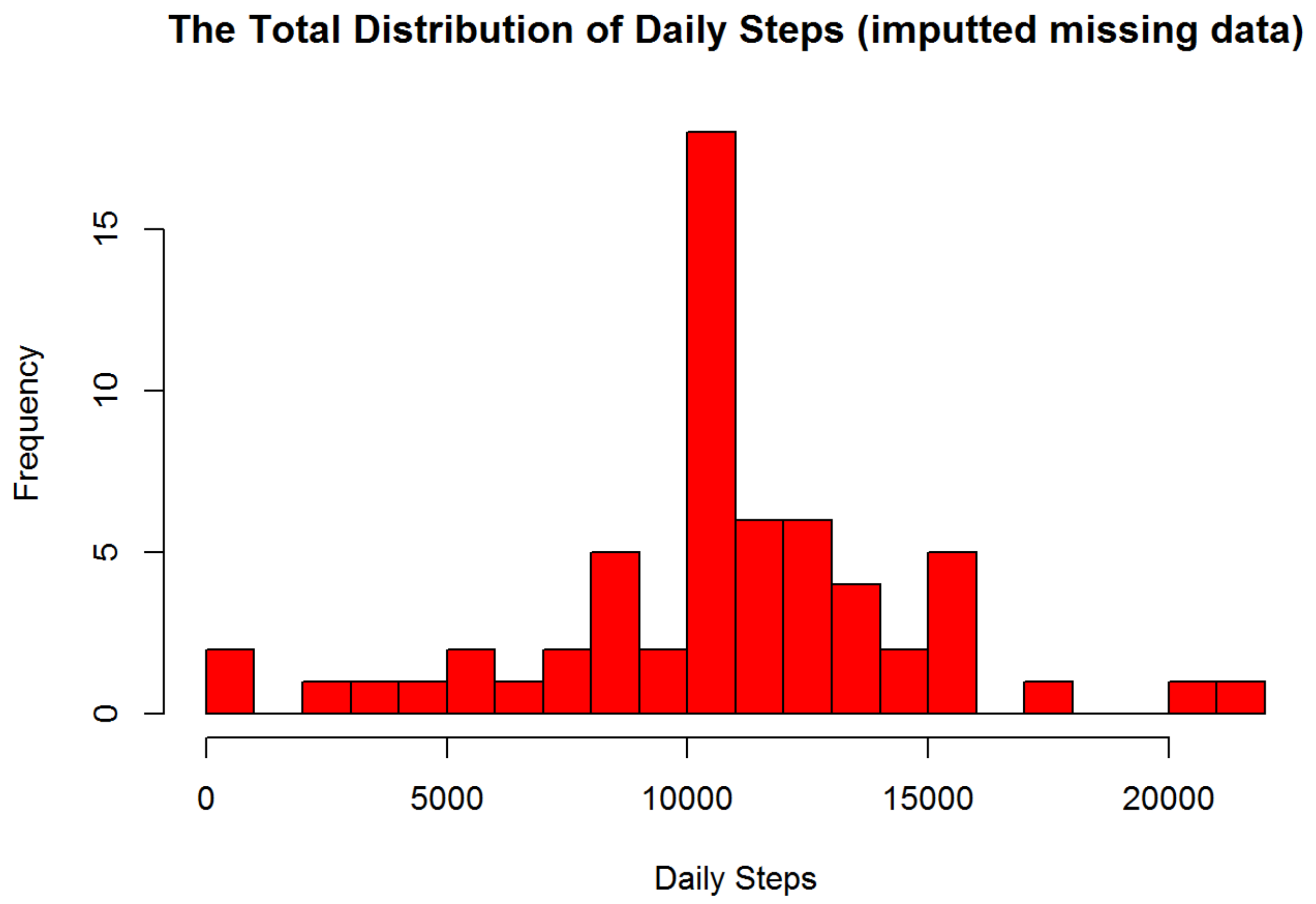
```
# There are two ways to manage missing values in data: leave the data or do data imputati
on to replace them. The first decision can be taken when missing values is around 5% of t
he sample. However, in our situation, we are seeing 13 percent of rows have missing value
s. (Total observations are 17,568, out of which 2304 rows have missing value. 2304/17568
= 13.11%) As such, we shall take another approach: do data imputation
```

```
# We continue to develop a strategy by creating a new dataset that is equal to the origin
al dataset, but with the missing data filled in.

# Our goals are achieved with the function of is.na() and na.rm().

# We also want to make a histogram of the total number of steps taken each day and calcul
ate the mean and median total number being taken per day.
```

```
hist(x=new_dailysum,
     col="red",
     breaks=20,
     xlab="Daily Steps",
     ylab="Frequency",
     main="The Total Distribution of Daily Steps (imputted missing data)")
```



```
mean(new_dailysum)
```

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

```
median(new_dailysum)
```

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

```
# The output of mean() and median() provides the mean of 10766.19 and the median 10766.1
9. We do not see any significant difference comparing to the numbers being produced earli
er, although the median is slightly different. (10766.19-10765=1.19)
```

```
# But in the plot associated with imputed data task, we are able to see the frequency bec
omes larger as 12.5 (with imputing missing data) is greater than 9 (without imputed dat
a). It should explain that the advantages of imputing missing data on the estimates of th
e total daily number yields a higher frequency counts. This identification can be seen fr
om those charts.
```

Are there differences in activity patterns between weekdays and weekends?

```
# The function of weekdays() is used to extract the weekday, indicating whether a given d
ate is a weekday or weekend day. Then, we will make a panel plot containing a time series
plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged ac
ross all weekday days or weekend days (y-axis).
```

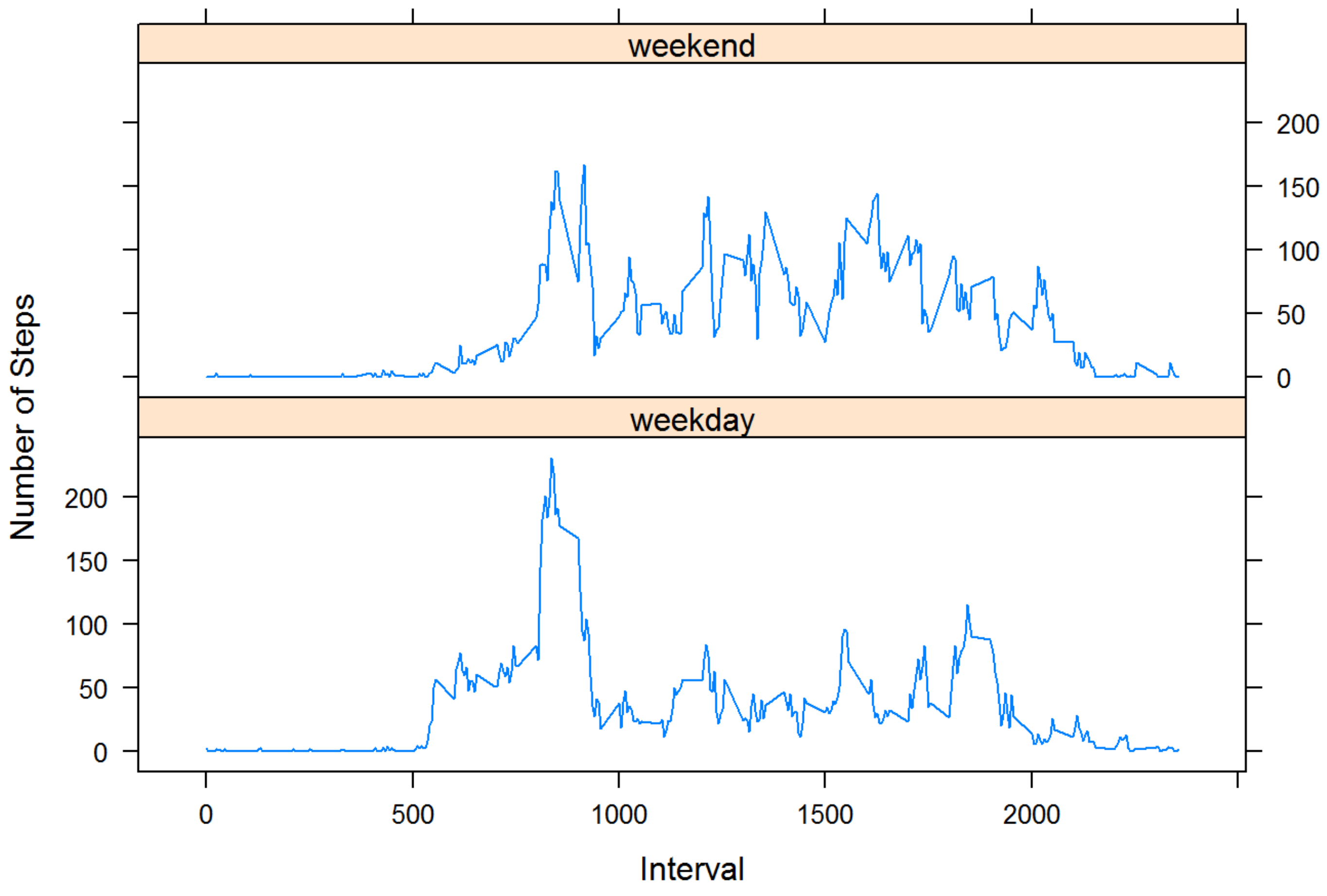
```
is_weekday <- function(d) {
  wd <- weekdays(d)
  ifelse (wd == "Saturday" | wd == "Sunday", "weekend", "weekday")
}

wx <- sapply(df_impute$date, is_weekday)
df_impute$wk <- as.factor(wx)
head(df_impute)
```

```
##       steps       date interval      wk
## 1 1.7169811 2012-10-01        0 weekday
## 2 0.3396226 2012-10-01        5 weekday
## 3 0.1320755 2012-10-01       10 weekday
## 4 0.1509434 2012-10-01       15 weekday
## 5 0.0754717 2012-10-01       20 weekday
## 6 2.0943396 2012-10-01       25 weekday
```

```
wk_df <- aggregate(steps ~ wk+interval, data=df_impute, FUN=mean)

library(lattice)
xyplot(steps ~ interval | factor(wk), layout = c(1, 2), xlab="Interval", ylab="Number of
Steps", type="l", lty=1, data=wk_df)
```



```
#From the output of these two plots, we can see that the highest steps usually occur in t
he morning session, where weekdays activities might start from 5 or 6 AM, but 8 ~ 9 AM on
weekends. That should be a common human-being activity as we tend to wake up later on wee
kends. On average, we also see more activities occur on weekdays, in the range of 50 ~ 15
0, where weekends, 25 ~ 100.
```