Peer Assessment 1

Jacques Cherbuin

16/12/2021

Loading and preprocessing the data

1. Load the data (i.e. read.csv())

We read the data and store it in the variable activity using the read.csv command:

```
activity <- read.csv("activity.csv")
str(activity)
```

```
## '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 ...
```

2. Process/transform the data (if necessary) into a format suitable for your analysis

We process the data by changing the date column from character to date class:

```
activity$date <- as.Date(activity$date)
str(activity)
```

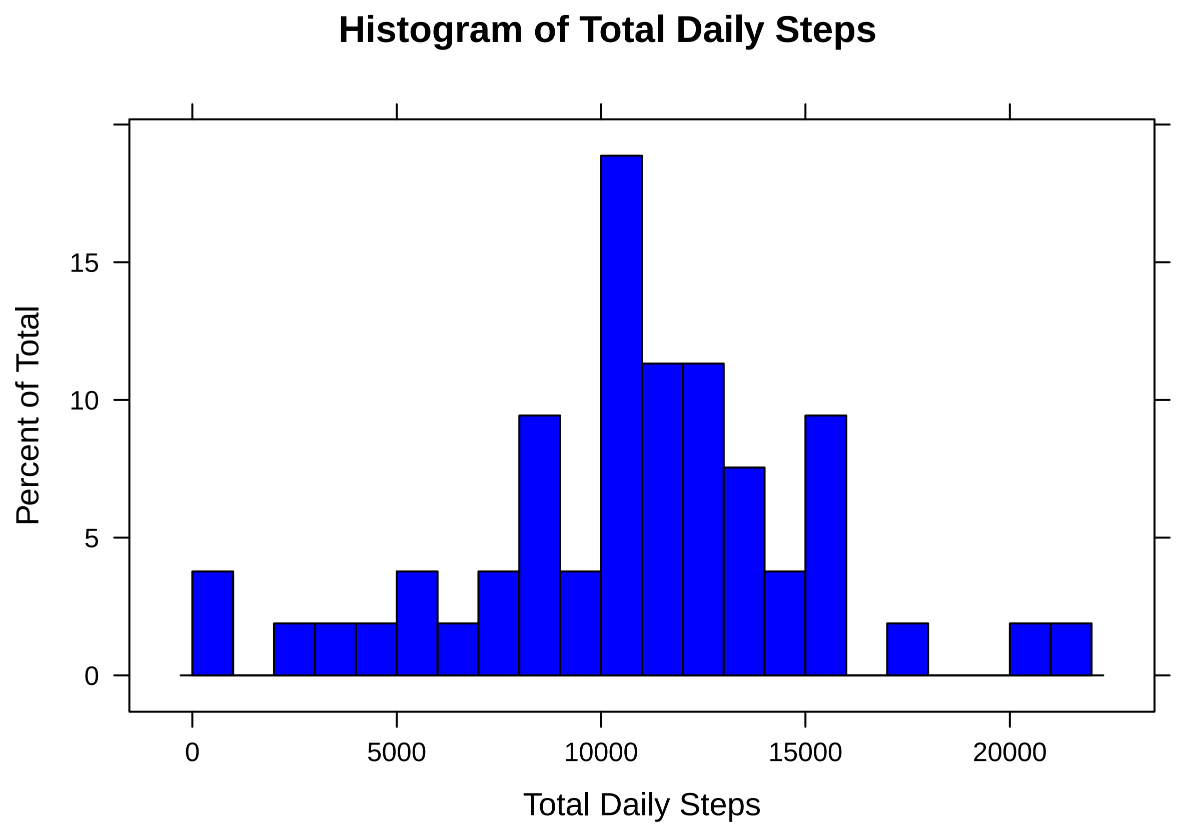
```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: int  0 5 10 15 20 25 30 35 40 45 ...
```

What is mean total number of steps taken per day?

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

We omit NA values and use tapply to sum steps for each day and create a histogram of the total daily steps using the xyplot command from the lattice package:

```
library(lattice)
non_missing = complete.cases(activity)
daily_steps <- tapply(activity$steps[non_missing],activity$date[non_missing],sum)
histogram(daily_steps,breaks=20,main="Histogram of Total Daily Steps",xlab = "Total Daily Steps", col='
```



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

We calculate the mean number of steps using the mean command:

```
mean_steps <- mean(daily_steps)
mean_steps
```

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

We calculate the median number of steps using the median command:

```
median_steps <- median(daily_steps)
median_steps
```

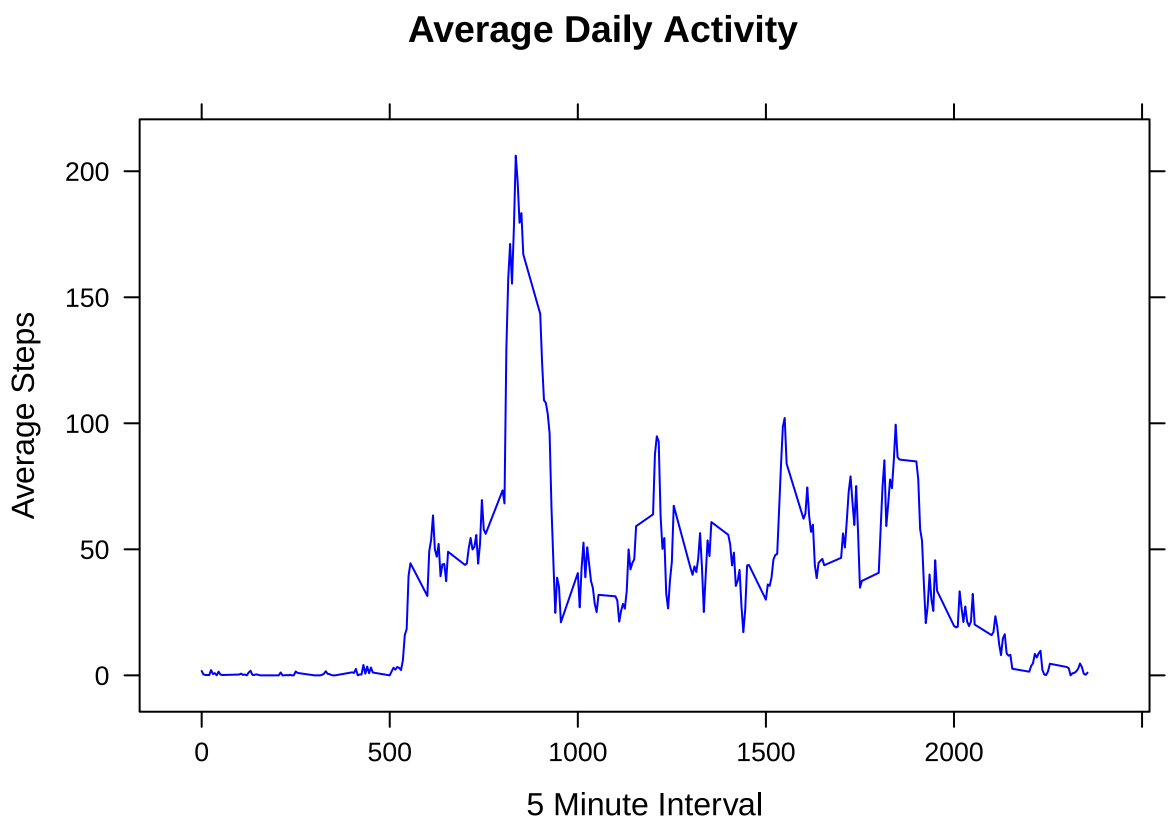
```
## [1] 10765
```

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)

We remove NA values use the xyplot command from the lattice package of type "l" to create a time series plot of daily activity:

```
interval_average_steps <- tapply(activity$steps[non_missing],activity$interval[non_missing],mean)
intervals <- unique(activity$interval)
xyplot(interval_average_steps~intervals,type="l", main="Average Daily Activity",xlab="5 Minute Interval
```



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

We find the index value of the maximum number of average steps and use it to find the interval:

```
max_index <- which.max(interval_average_steps)
max_interval <- intervals[max_index]
max_interval
```

```
## [1] 835
```

Imputing missing values

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

We use the complete.cases command to find missing values and sum them up:

```
num_missing_values <- sum(!complete.cases(activity))
num_missing_values
```

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

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.

We will replace all the missing step values with the mean of the interval of those step values.

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

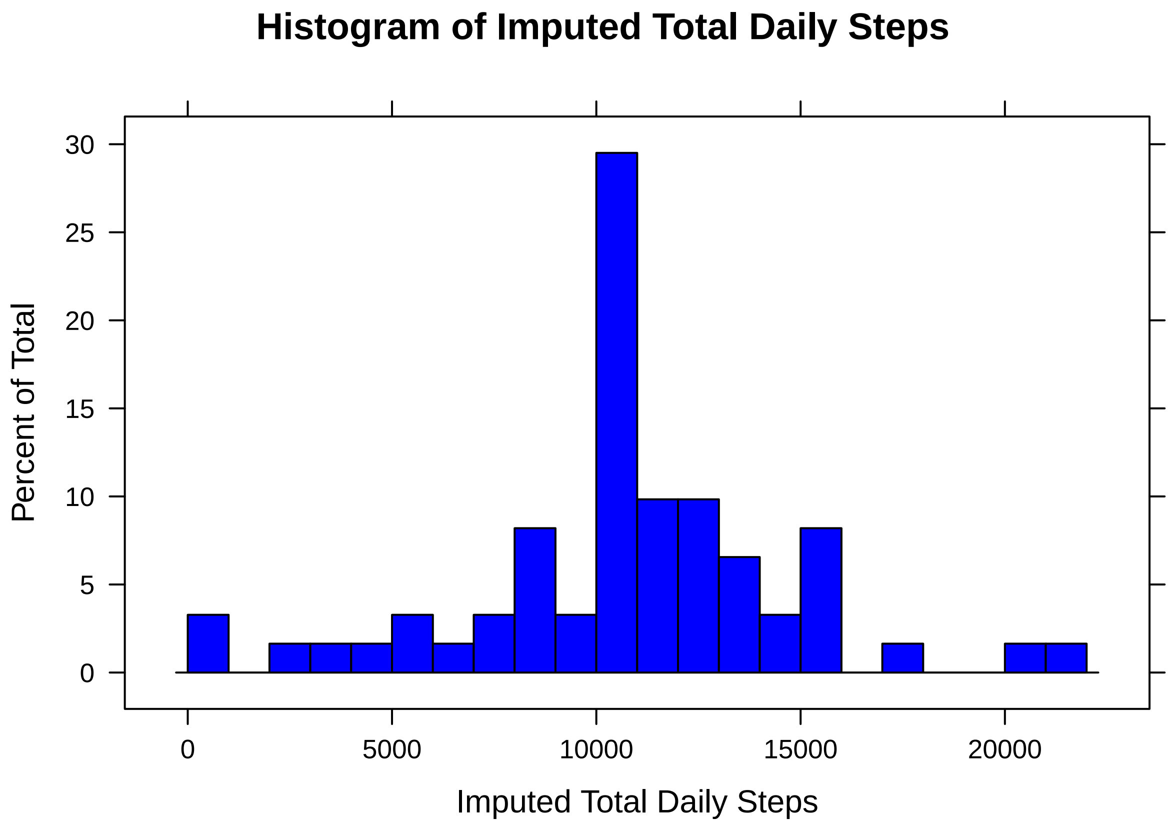
To create the imputed data frame we first index the missing values. We use the missing values index to find the average step value for the missing intervals and place them into a copy of the original data frame using a for loop:

```
missing_values_index <- which(!complete.cases(activity))
interval_index <- unique(activity$interval)
imputed_activity <- data.frame(activity)
for (i in 1 : num_missing_values) imputed_activity$steps[missing_values_index[i]] <- interval_average_s
```

4.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?

We create a histogram of the imputed total daily steps use the histogram function:

```
imputed_daily_steps <- tapply(imputed_activity$steps,imputed_activity$date,sum)
histogram(imputed_daily_steps,breaks=20,main="Histogram of Imputed Total Daily Steps",xlab = "Imputed To
```



We calculate the mean number of imputed steps:

```
mean_imputed_steps <- mean(imputed_daily_steps)
mean_imputed_steps
```

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

We calculate the median number of imputed steps:

```
median_imputed_steps <- median(imputed_daily_steps)
median_imputed_steps
```

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

We observe that the mean is the same as our previous estimate, and the median is now equal to the mean. Imputing missing data has the effect of adding intervals with the average number of steps. Therefore the mean doesn't change, and the median is pushed closer to the mean.

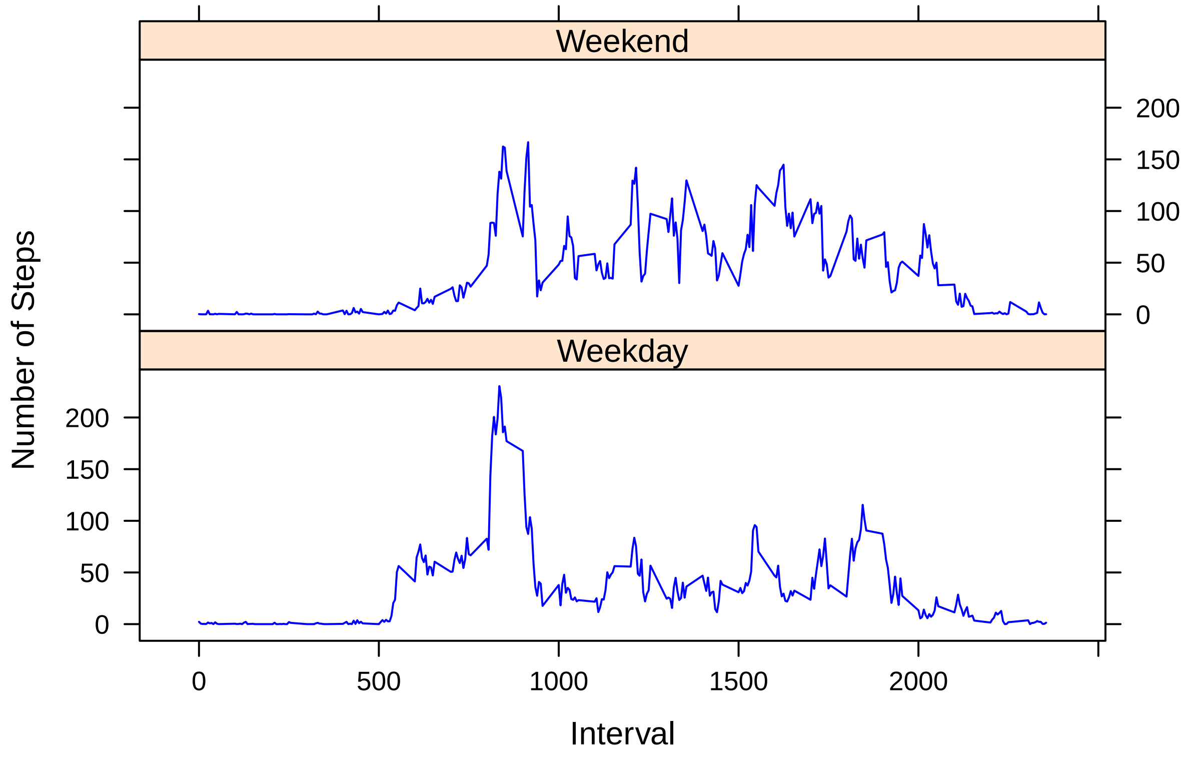
Are there differences in activity patterns between weekdays and weekends?

We create a factor variable with 2 levels "Weekday" and "Weekend" and then split the data frame according to those levels:

```
imputed_activity$day <- factor(sapply(weekdays(imputed_activity$date),function(x) x %in% c("Saturday","S
```

We use the aggregate command to find the mean of the intervals by weekday/weekend and create a time series line plot comparing weekday and weekend average steps per interval:

```
average_steps <- aggregate(imputed_activity$steps ~ imputed_activity$interval + imputed_activity$day, in
names(average_steps) <- c('interval','day','steps')
xyplot(steps~interval|day,average_steps,type='l',layout=c(1,2),xlab='Interval',ylab='Number of Steps', 
```


Weekend
Weekday
Number of Steps
Interval