Reproducible Research: Peer Assessment 1

Hanyu Wang

October 1, 2015

Loading and preprocessing the data

Show any code that is needed to

1. Load the data (i.e. `read.csv()`)
2. Process/transform the data (if necessary) into a format suitable for your analysis

```
## Loading the necessary packages
library(ggplot2)
library(dplyr)
library(scales)
```

```
## Load activity table
act = read.csv("activity.csv", sep = ",", header = TRUE, na.strings = "NA")
```

```
## Translate the date into date syntax.
act$date = as.Date(act$date, format = "%Y-%m-%d")
```

```
## Change the interval into unified time format:
itv = sapply(act$interval, toString)

## Insert 0's to make Interval into standart time format:
for (i in 1:length(itv)){
        itv[i] = paste(paste(rep(0, (4 - nchar(itv[i]))), collapse = ""), itv[i], sep = "")
}

## Add : to separate hour and minute:
itv = gsub('^([0-9]{2})', '\\1:\\', itv)

## Translate the interval into time syntax:
act$interval = as.POSIXct(itv, format = "%H:%M")


## Doing that in a more trivial way:
#for (i in 1:length(itv)){
        #if (nchar(itv[i])  == 1){
        #itv[i] = paste("000", itv[i], sep = "")
        #} else if (nchar(itv[i])  == 2){
        #itv[i] = paste("00", itv[i], sep = "")
        #} else if (nchar(itv[i])  == 3){
        #itv[i] = paste("0", itv[i], sep = "")
        #}
#}
```

What is mean total number of steps taken per day?

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

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

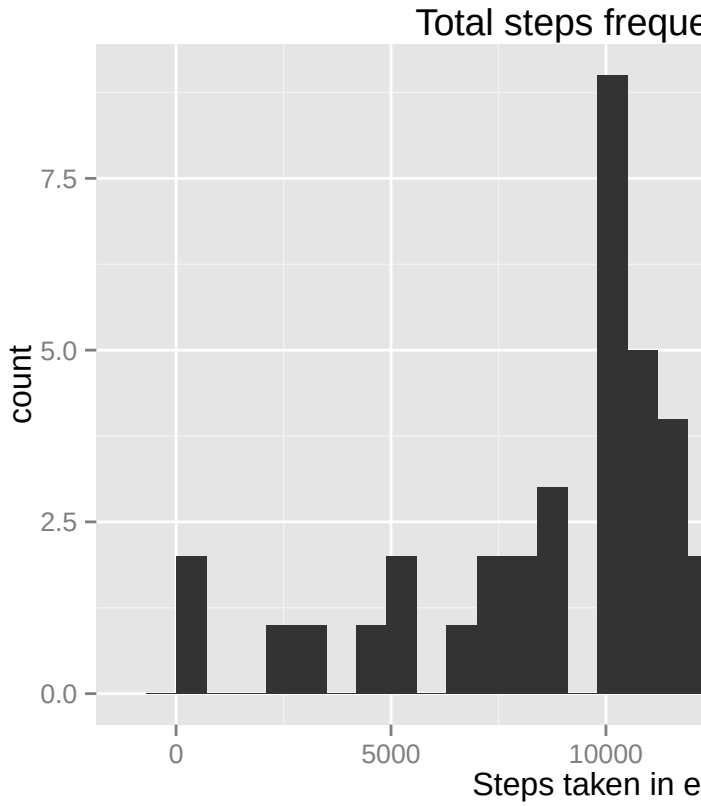
- Calculate the mean and the median for total steps of each day:

```
## Remove the missing values:
actComplete = act[complete.cases(act), ]
str(actComplete)
```

```
## 'data.frame':    15264 obs. of  3 variables:
##  $ steps   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ date    : Date, format: "2012-10-02" "2012-10-02" ...
##  $ interval: POSIXct, format: "2015-10-06 00:00:00" "2015-10-06 00:05:00" ...
```

```
## Split the table by the date, calculate sum of the steps for each days respectively,
## By grouping and summary:
stepDaily = actComplete %>% group_by(date) %>% summarize(steps = sum(steps))



## By employing aggregate function:
#stepDaily = aggregate(steps ~ date, data = actComplete, sum)

##By employing sapply function:
#spltAct = split(actComplete, actComplete$date)
#stepDaily = sapply(spltAct, function(x) return(sum(x$steps)))

## By employing tapply function:
#stepDaily = with(actComplete, tapply(steps, date, sum))
```

```
## Plot the histogram of the sum step for each day:
g_dailyfreqCount = ggplot(stepDaily, aes(x = steps))
g_dailyfreqCount + geom_histogram(binwidth = 700) + labs(title = "Total steps frequency count", x = "St
```



plot the histogram of the total steps count frequency:

```
## Count for mean and median:
meanStepDaily = mean(stepDaily$steps)
medianStepDaily = median(stepDaily$steps)
```

Calculate the mean and the median for sum steps of each day:

```
    The mean of the total steps is 10766.19 steps.
    The median of the total steps is 10765` steps.
```

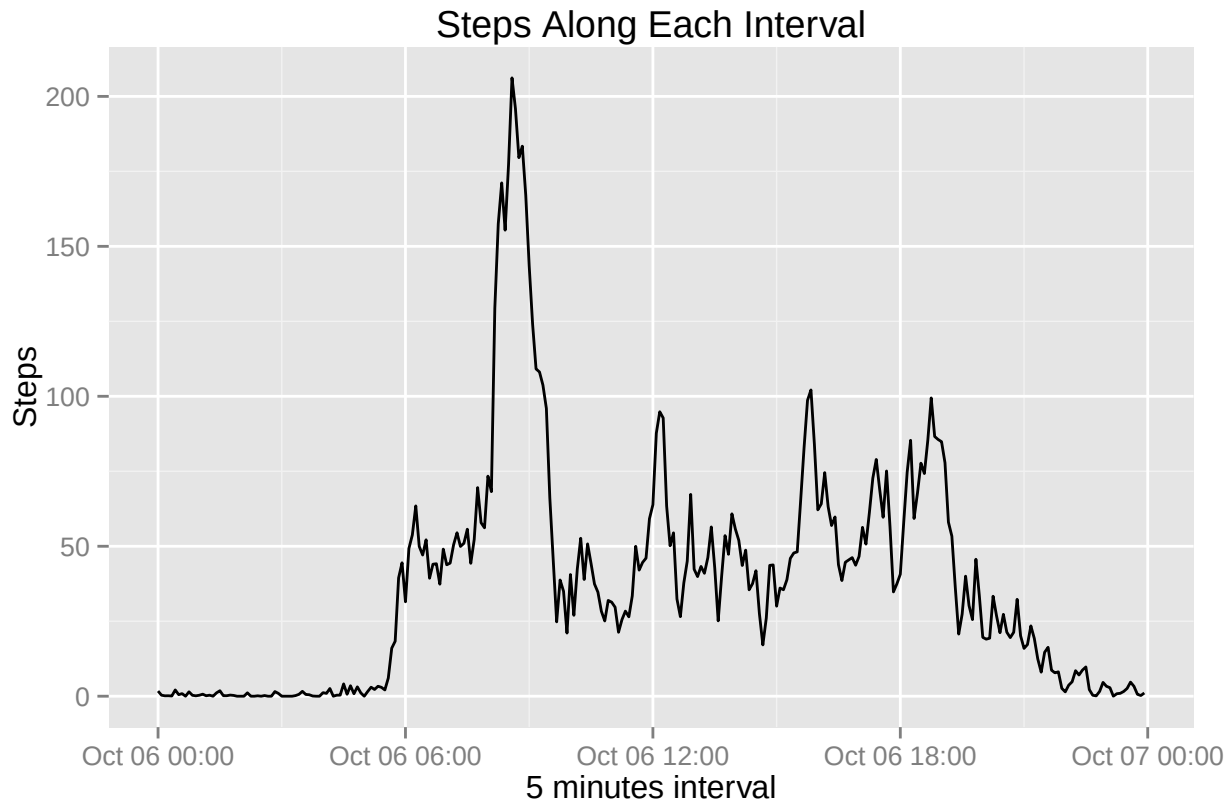
What is the average daily activity pattern?

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

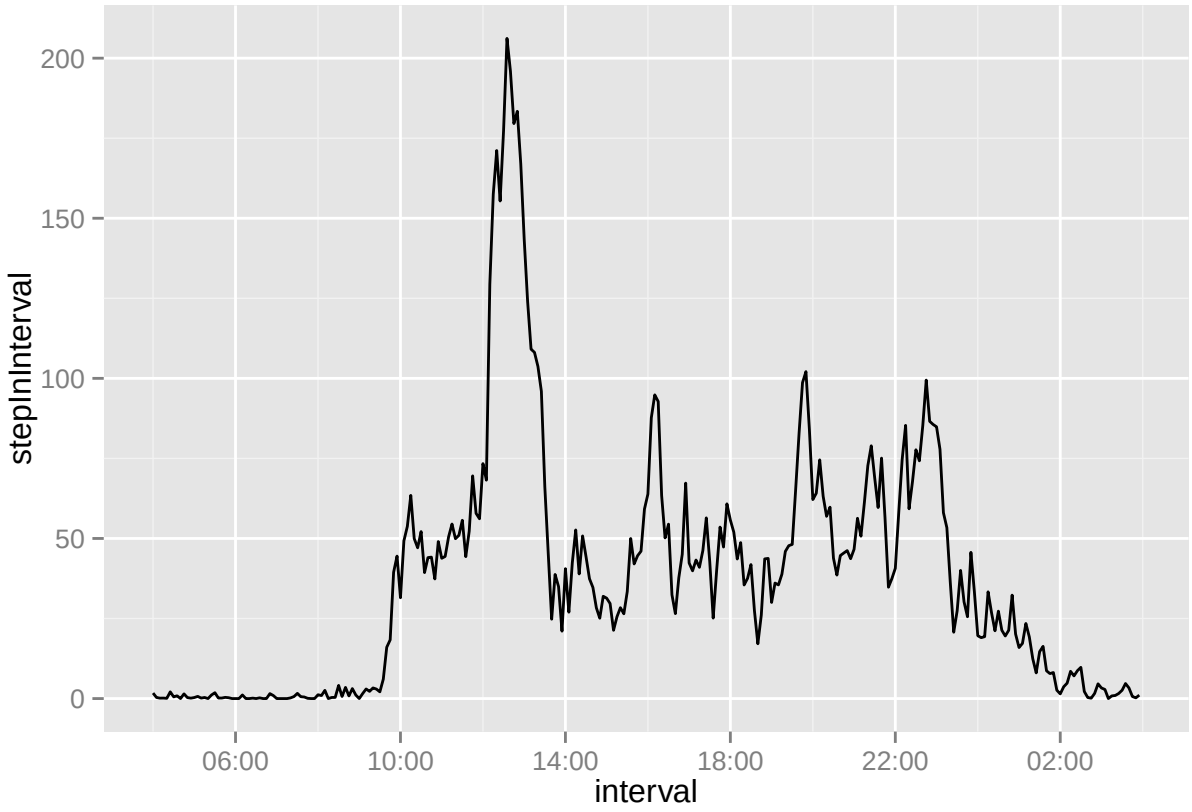
```
## Split the table by the date, calculate sum of the steps for each time interval respectively,
## By grouping and summary:
stepIntveral = actComplete %>% group_by(interval) %>% summarize(stepInInterval = mean(steps))
```

```
## By aggregate function:
#stepIntveral = aggregate(steps ~ interval, data = actComplete, mean)
```

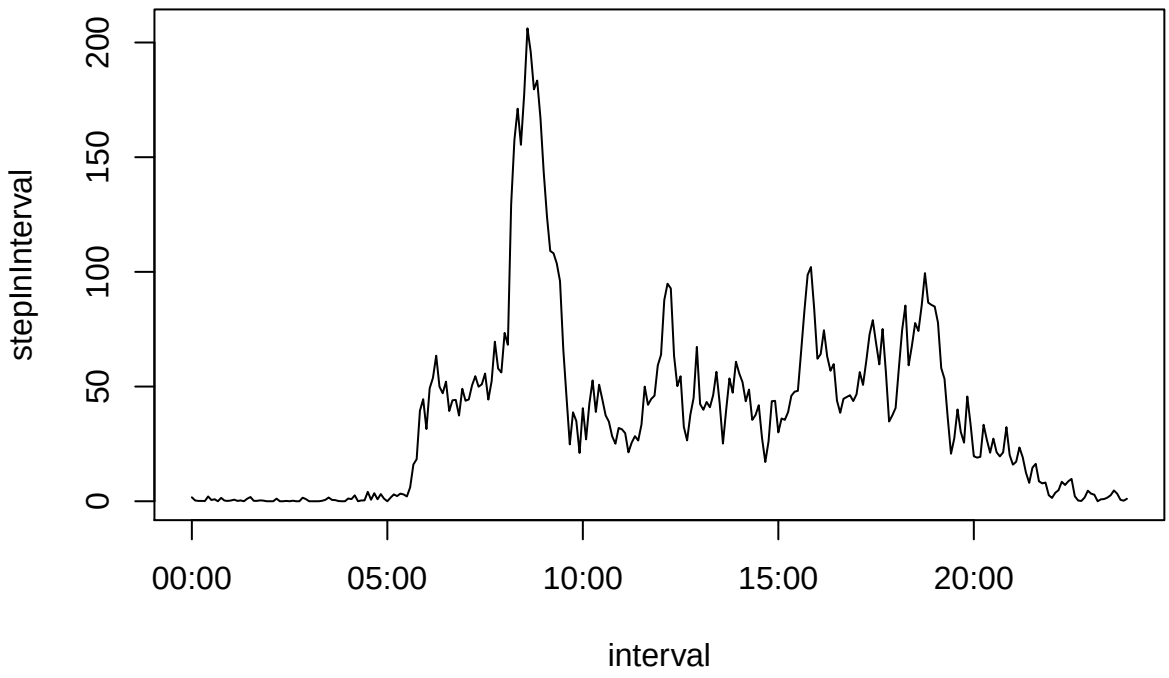
```
## Plot the time series of the sum step for interval:
g_intervalstep = ggplot(stepIntveral, aes(x = interval, y = stepInInterval))

g_intervalstep + geom_line() + labs(title = "Steps Along Each Interval", x = "5 minutes interval", y = "
```




```
### Cannot solve the incorrect time label so far:
g_intervalstep + geom_line() + scale_x_datetime(breaks = date_breaks("4 hours"), minor_breaks = date_bre
```




```
## Comparison:
with(stepIntveral, plot(interval, stepInInterval, type = "l"))
```




```
## Arrange the stepInterval table by stepInInterval in descending order:
maxStepIntveral = stepIntveral %>% arrange(desc(stepInInterval))
```

```
    The 2015-10-06 08:35:00 interval across all 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 (coded as `NA`). The presence of missing days may introduce bias into some calculations or summaries of the data.

1. Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with `NA`s)
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.
3. Create a new dataset that is equal to the original dataset but with the missing data filled in.
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?

```
## Summarize the count of missing values:
missingCount = sum(!complete.cases(act))
missingProp = missingCount / nrow(act)
```

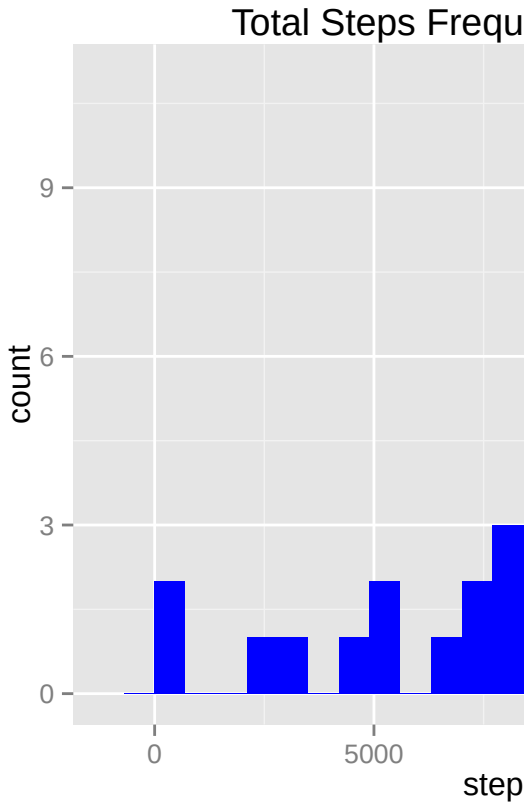
```
    There are 2304 missing values in the dataset. Which is 13.11475% of the entire dataset.
```

One strategy filling out the missing value is that find the average value for the same time interval at the same weekday:

```
## Mutate a weekday variable by the date:
actDay = act %>% mutate(day = factor(weekdays(date), levels = c("Sunday", "Monday", "Tuesday", "Wednesda


## grouping and summarizing steps along weekdays during each interval:
actdaySummary = actDay %>% na.omit() %>% arrange(day, interval) %>% group_by(day, interval) %>% summari
```

```
## Merge the original dataset and the missing values removed summary table:
act_NA = merge(x = actDay, y = actdaySummary, by.x = c("interval", "day"), by.y = c("interval", "day"))



## Replaced the missing value by the averaged value,
## By forming ifelse loop:
act_NA = act_NA %>% mutate(stepsRemoveNAs = ifelse(test = is.na(steps), yes = avgSteps, no = steps))

## By traditional method:
#for (i in 1:nrow(act_NA)) {
        #if (is.na(act_NA[i, "steps"])) {
                #act_NA[i, "steps"] = act_NA[i, "avgSteps"]
        #}
#}
#act_removeNA = act_NA


## Clean up the table:
act_removeNA = act_NA %>% select(stepsRemoveNAs, date, interval) %>% arrange(date, interval)
```

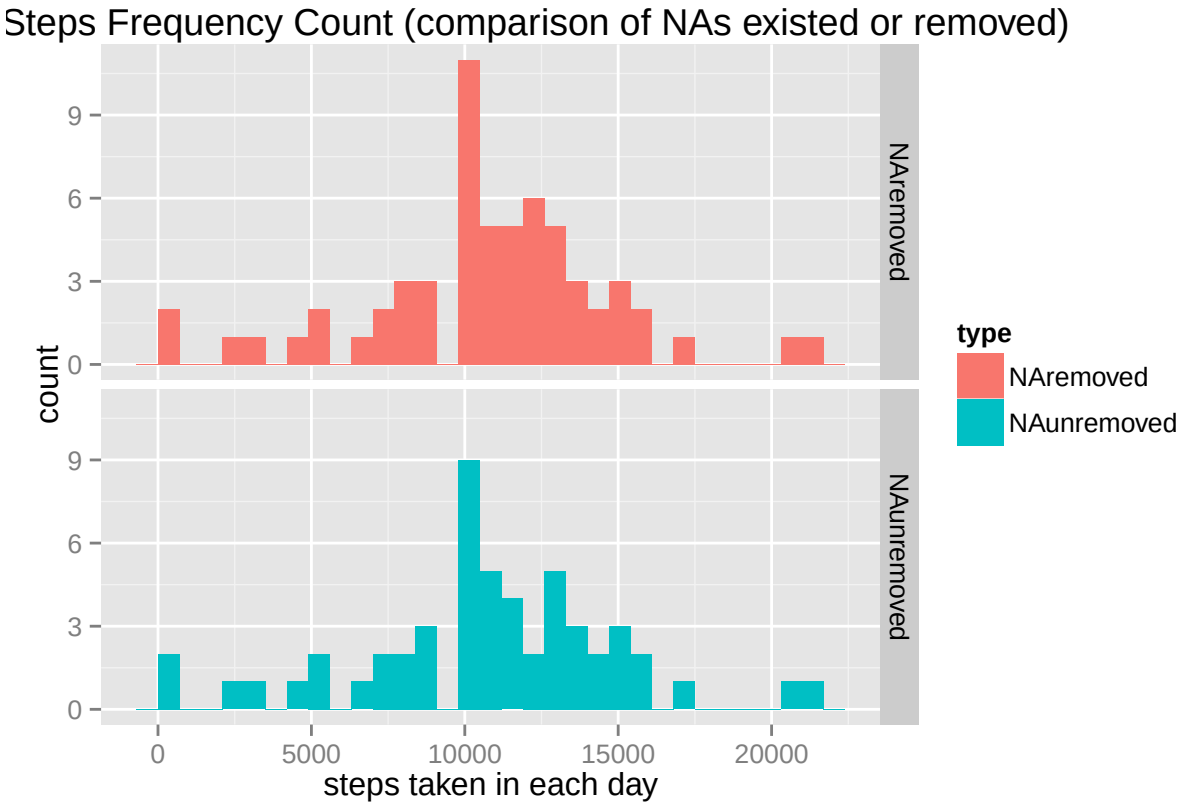
```
## Summarize steps occuring frequency:
stepDaily_removeNA = act_removeNA %>% group_by(date) %>% summarize(steps = sum(stepsRemoveNAs))


## Plot the histogram of the sum step for each day:
g_dailyfreqCountRemoveNAs = ggplot(stepDaily_removeNA, aes(x = steps))
g_dailyfreqCountRemoveNAs + geom_histogram(binwidth = 700, fill = "BLUE") + labs(title = "Total Steps Fr
```



plot the histogram of the total steps count frequency without NAs:

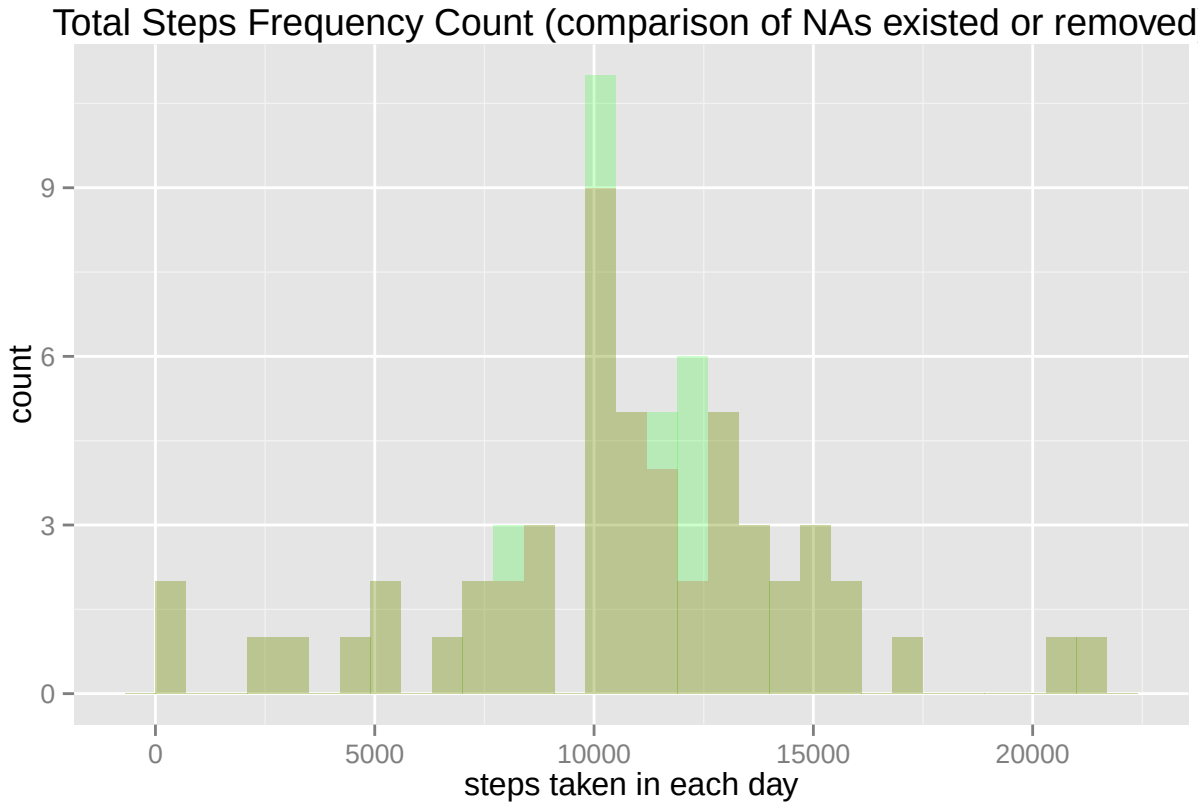
To make a comparison:

```
## Summarize steps occuring frequency:
stepDaily_removeNA_comparsion = act_NA %>% group_by(date) %>% summarize(stepsNA = sum(steps), stepsRemov

## melt and combine into one table:
stepDaily_withNA = stepDaily_removeNA_comparsion[ ,1:2]
stepDaily_withNA$type = "NAunremoved"

stepDaily_withNA = rename(stepDaily_withNA, steps = stepsNA)

stepDaily_withoutNA = stepDaily_removeNA_comparsion[ ,c(1,3)]
stepDaily_withoutNA$type = "NAremoved"
stepDaily_withoutNA = rename(stepDaily_withoutNA, steps = stepsRemove_NAs)

stepDaily_NA_comparsion = rbind(stepDaily_withNA, stepDaily_withoutNA)

## Clean the table
stepDaily_NA_comparsion$type = factor(stepDaily_NA_comparsion$type, order = TRUE)


## Plot the histogram of the sum step for each day:
g_dailyfreqCountComparison = ggplot(stepDaily_NA_comparsion, aes(x = steps, fill = type))
g_dailyfreqCountComparison + geom_histogram(binwidth = 700,) + facet_grid(type ~ .) + labs(title = "Tota
```



Or:

```
## Plot the histogram of the sum step for each day:
g_dailyfreqCountComparison = ggplot(stepDaily_NA_comparsion, aes(x = steps))
g_dailyfreqCountComparison + geom_histogram(data = subset(stepDaily_NA_comparsion, type == "NAunremoved
```




```
## Count for mean and median:
meanStepDaily_removeNA = mean(stepDaily_removeNA$steps)
medianStepDaily_removeNA = median(stepDaily_removeNA$steps)
```

Calculate the mean and the median for sum steps of each day:

```
After removing all NAs, The mean of the total steps is 10821.21 steps.
The median of the total steps is 11015.00` steps.
As we can conclude from it, after removing all missing values, the mean and the median is slightly
```

Are there differences in activity patterns between weekdays and weekends?

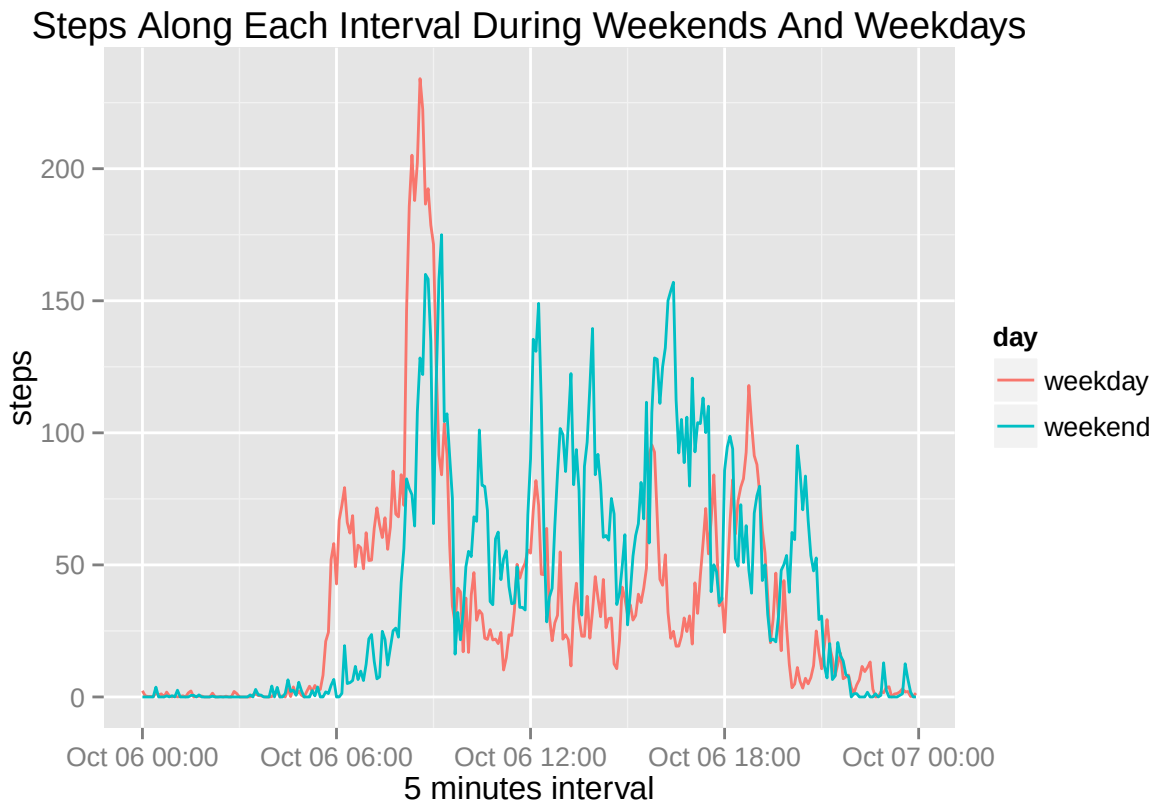
For this part the `weekdays()` function may be of some help here. Use the dataset with the filled-in missing values for this part.

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.
2. 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).

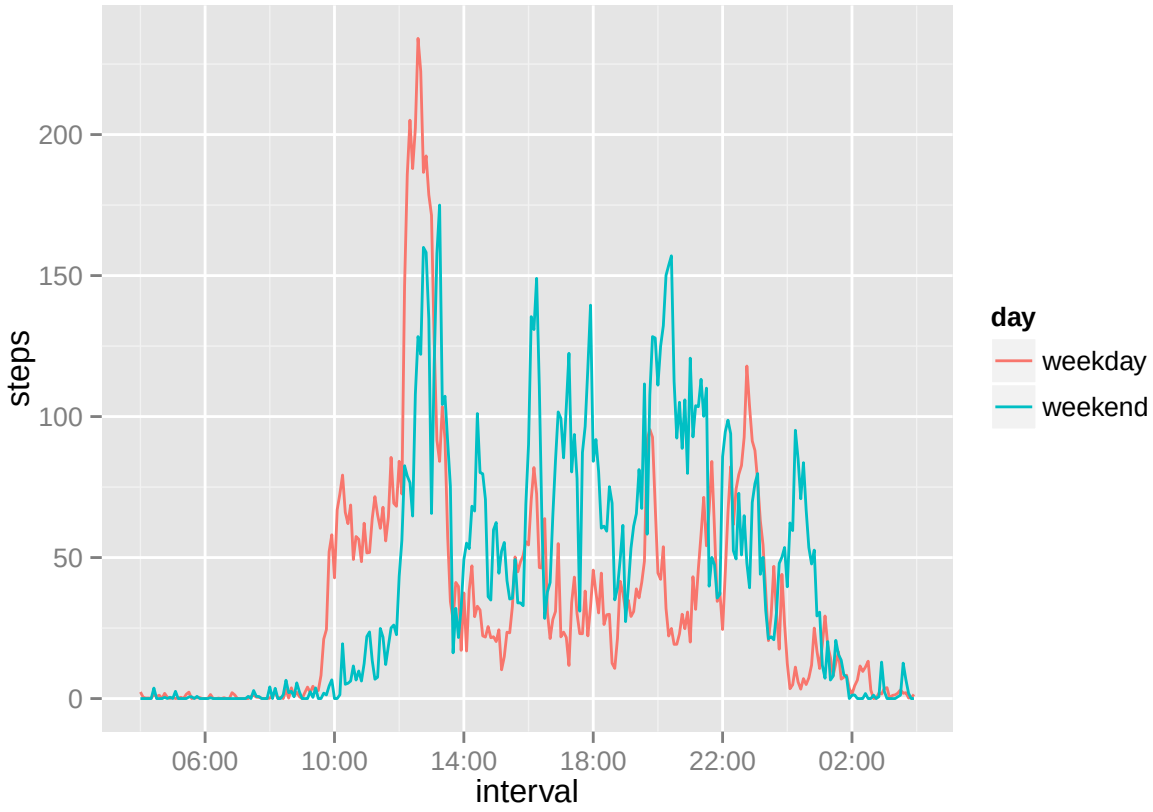
```
## Mutate a weekday variable:
actweekday = act %>% na.omit() %>% mutate(day = ifelse(test = (weekdays(date) %in% c("Sunday", "Saturday
## Factorize the day variable:
actweekday$day = factor(actweekday$day, levels = c("weekday", "weekend"), ordered = TRUE)
```

```
## Summarize the steps along intervals for weekdays and weekends respectively:
actWeekdaySummary = actweekday %>% group_by(day, interval) %>% summarize(steps = mean(steps)) %>% arrang
```

```
## Plot the steps count along intervals for weekdays and weekends respectively:
g_weekdayIntervalSteps = ggplot(actWeekdaySummary, aes(x = interval, y = steps, color = day))
g_weekdayIntervalSteps + geom_line() + labs(title = "Steps Along Each Interval During Weekends And Week
```

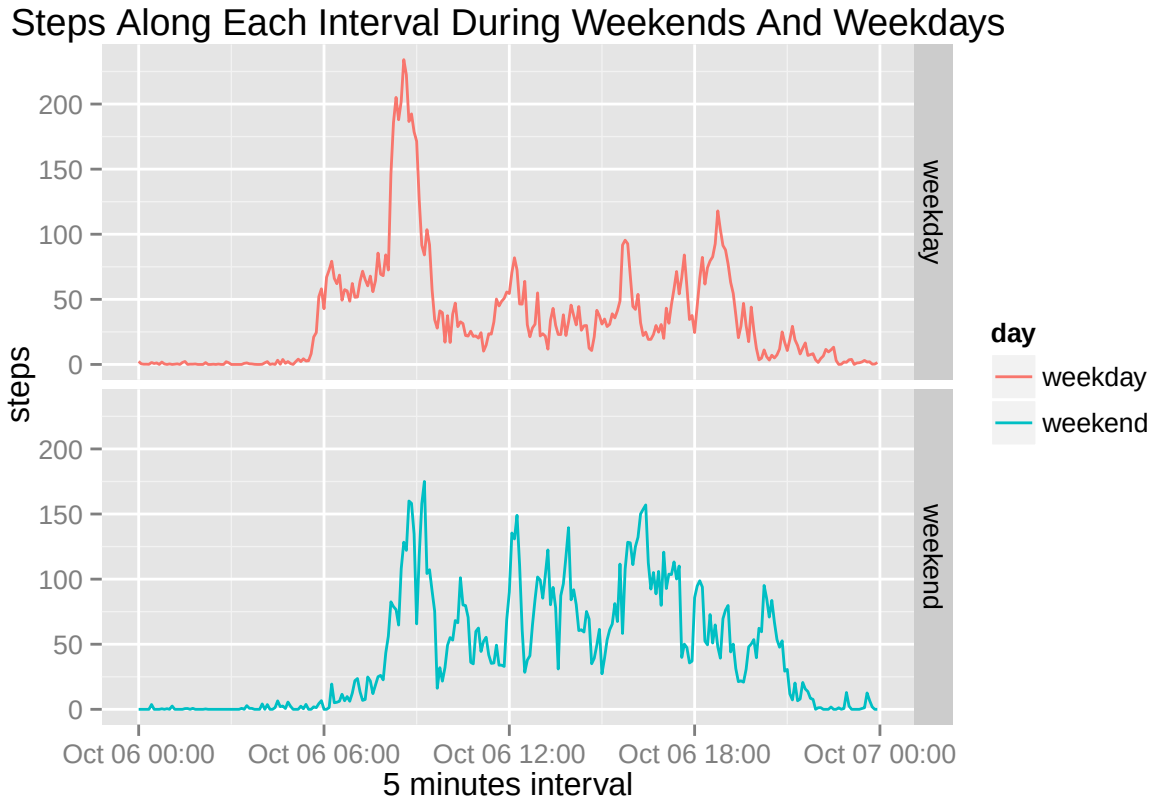



```
### Cannot solve the incorrect time label so far:
g_weekdayIntervalSteps + geom_line() + scale_x_datetime(breaks = date_breaks("4 hours"), minor_breaks =
```



Or:

```
g_weekdayIntervalSteps2 = ggplot(actWeekdaySummary, aes(x = interval, y = steps, color = day))

g_weekdayIntervalSteps2 + geom_line() + facet_grid(day ~ .) + labs(title = "Steps Along Each Interval Du
```


Steps Along Each Interval During Weekends And Weekdays
steps
200
150
100
50
0
weekday
weekend
Oct 06 00:00
Oct 06 06:00
Oct 06 12:00
Oct 06 18:00
Oct 07 00:00
5 minutes interval
day
weekday
weekend