Reproducible Research Course Project 1: Peer Graded Assessment

U.Esparza

2020/06/14

1) Loading and pre-processing the data

Firstly, the data must be read:

If not done yet, the zip file must be unzipped first. To do so, the below unzip() function can be used by removing the # symbol from the beginning of the code-line:

```
# unzip(zipfile = "./data/repdata_data_activity.zip", exdir = "./data")
reading <- read.csv("./data/activity.csv", sep=",")
```

Secondly, the dataset is checked using str(), head() and tail() functions:

```
str(reading)
```

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

```
head(reading)
```

```
##   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
```

```
tail(reading)
```

```
##       steps       date interval
## 17563    NA 2012-11-30     2330
## 17564    NA 2012-11-30     2335
## 17565    NA 2012-11-30     2340
## 17566    NA 2012-11-30     2345
## 17567    NA 2012-11-30     2350
## 17568    NA 2012-11-30     2355
```

Thirdly, the dates currently in character format must be transformed into POSIXct format by using strtime() and as.POSIXct():

```
reading$date <- as.POSIXct(strptime(reading$date, "%Y-%m-%d"))
```

---

2) Histogram of the total number of steps taken each day

Firstly, the ggplot2 and dplyr packages must be loaded:

If not done yet, the ggplot2 and dplyr packages must be installed. This can be done by removing the # symbol from the beginning of the function install.packages() in the fisrt two code-lines below:

```
# install.packages("ggplot2")
# install.packages("dplyr")
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Secondly, a new variable summarizing the total steps per day must be created:

```
mySumPerDate <- reading %>% group_by(date) %>% summarize(sumSteps=sum(steps))
```

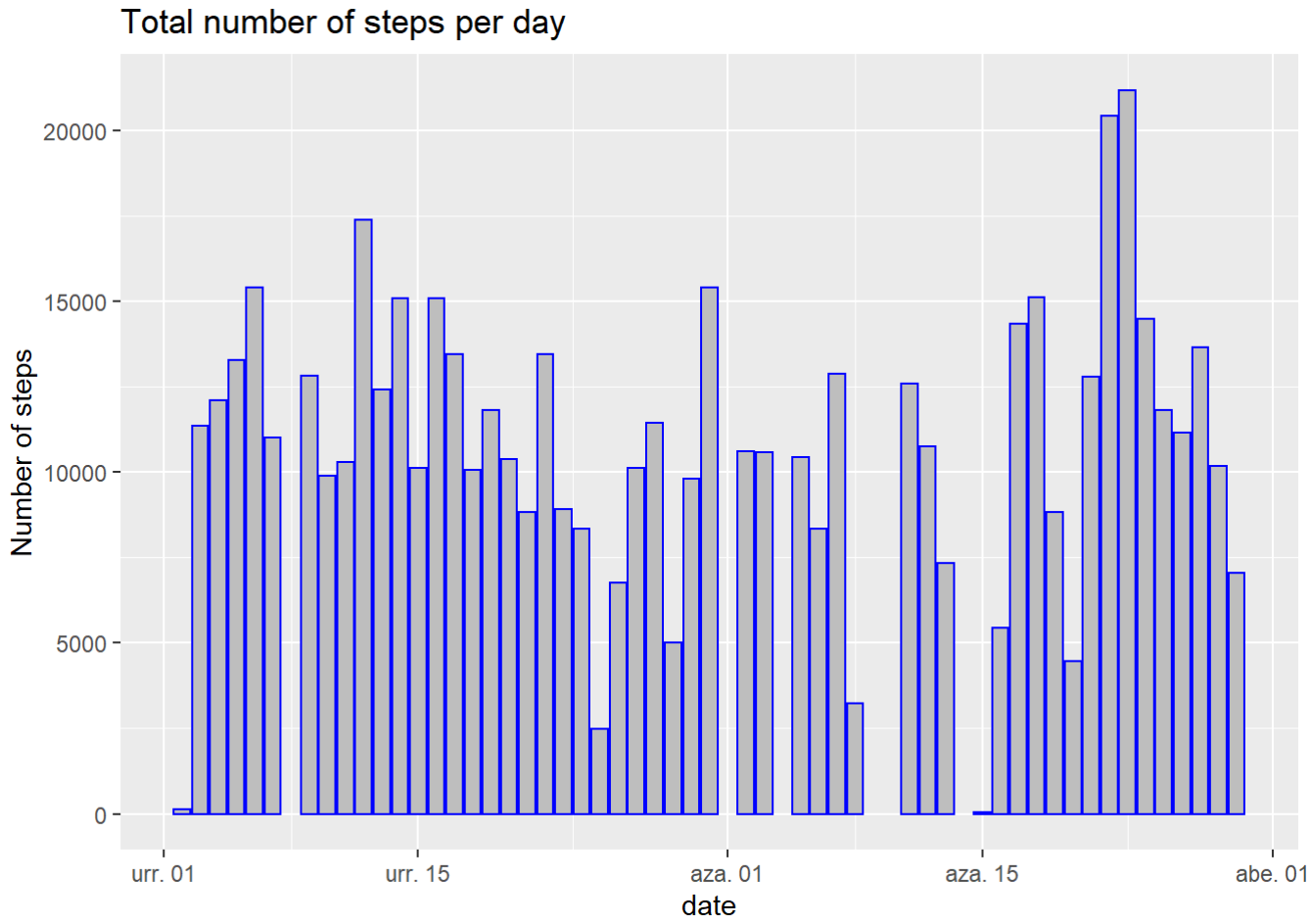
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Thirdly, the histogram with the total number of steps per day is plotted:

```
ggplot(mySumPerDate, aes(x = date, y = sumSteps), stat = "bin") +
geom_histogram(stat = "identity", col = "blue", bg = "gray") +
ggtitle("Total number of steps per day") + ylab("Number of steps")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

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



Remark: my language settings are set to Basque Language. That's why I get x-axis labels "Urr" (short for Urria, i.e. October in Basque), "Aza" (short for Azaroa, i.e. November), "Abe" (short for Abendua, i.e. December) etc.

3) Mean and median number of steps taken each day

The mean() and median() functions are used together with the condition na.rm=TRUE to remove NA values:

```
mean(mySumPerDate$sumSteps, na.rm=TRUE)
```

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

```
median(mySumPerDate$sumSteps, na.rm=TRUE)
```

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

4) Time series plot of the average number of steps taken

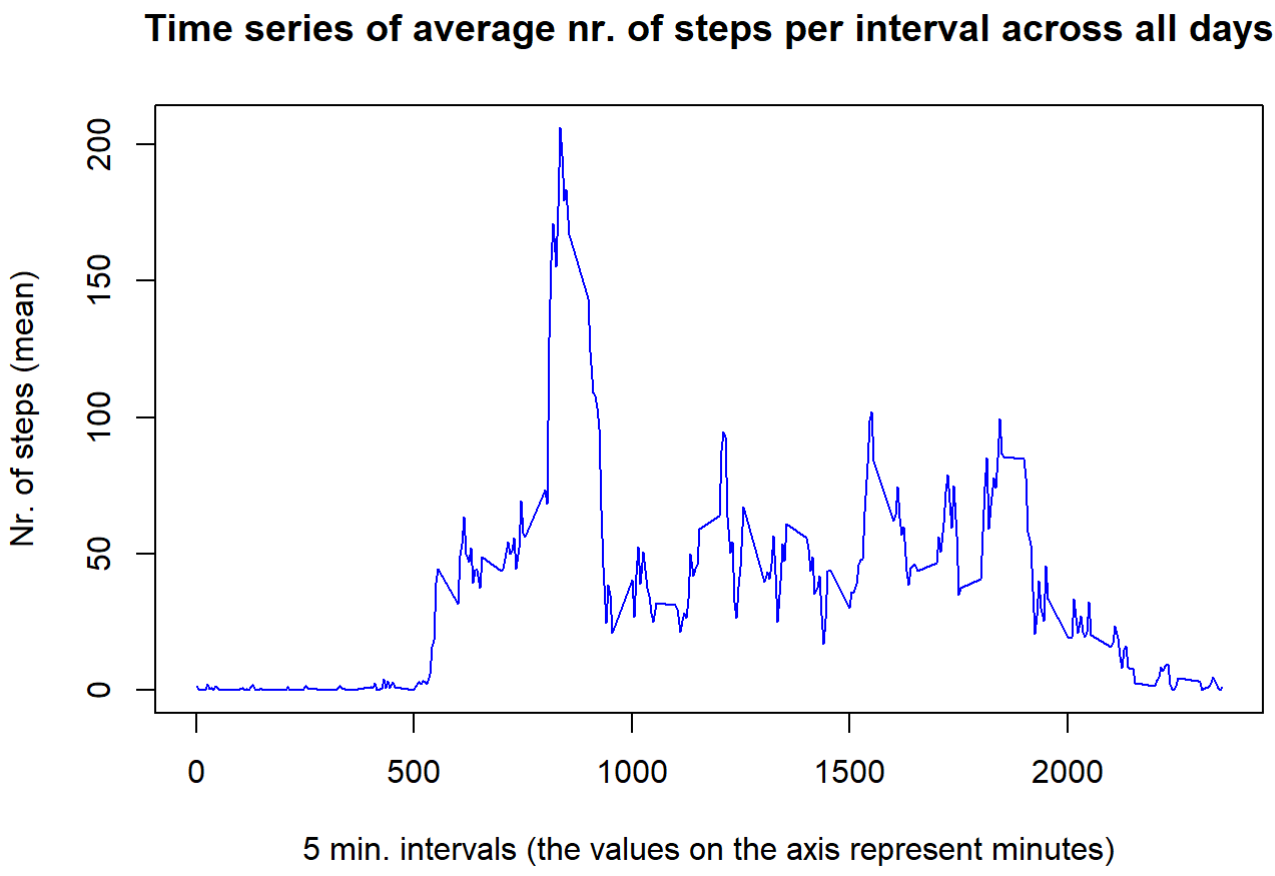
Firstly, a new variable is created summarizing the average number of steps per interval:

```
myMeanPerInt <- reading %>%  group_by(interval) %>% summarize(meanSteps=mean(steps,na.rm=TRUE
))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Secondly, a time series plot is created using the type = “l” condition:

```
plot(x=myMeanPerInt$interval, y=myMeanPerInt$meanSteps, type = "l",
     main = "Time series of average nr. of steps per interval across all days",
     xlab = "5 min. intervals (the values on the axis represent minutes)",
     ylab = "Nr. of steps (mean)",
     col = "blue")
```



5) The 5-minute interval that, on average, contains the maximum number of steps

the max() function is used to get the interval with the maximum nr. of steps, in combination with the row() function to get the complete row (including the interval itself):

```
myMeanPerInt[row(myMeanPerInt)[myMeanPerInt==max(myMeanPerInt$meanSteps)],]
```

```
## # A tibble: 1 x 2
##   interval meanSteps
##      <int>     <dbl>
## 1      835      206.
```

---

6) Code to describe and show a strategy for imputing missing data

The dataset contains some missing values. Let's check first how many NAs exist with the negated (!) version of the complete.cases() function:

```
sum(!complete.cases(reading))
```

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

Data imputation consists on inferring the NAs from the known part of the data, which is often a better strategy than just removing NAs.

The easiest way to carry out an imputation is by using univariate imputing. For instance, NAs can be imputed with a provided constant value, by using statistic calculations as mean, median etc. or by sampling with replacement from non-missing values.

The strategy used here is the UNIVARIATE IMPUTING BY SAMPLING WITH REPLACEMENT from the non-missing values.

In order to use the imputeUnivariate() function, the "missRanger" package must be installed first.To do so, the below install.packages() function can be used by removing the # symbol from the beginning of the code-line:

```
# install.packages("missRanger")
library(missRanger)
readingImputed <- imputeUnivariate(x=reading, v = NULL, seed = NULL)
```

Now we can use the negated (!) version of the complete.cases() function again to check if the imputation actually worked (the result should be zero):

```
sum(!complete.cases(readingImputed))
```

```
## [1] 0
```

---

7) Histogram of the total number of steps taken each day after missing values are imputed

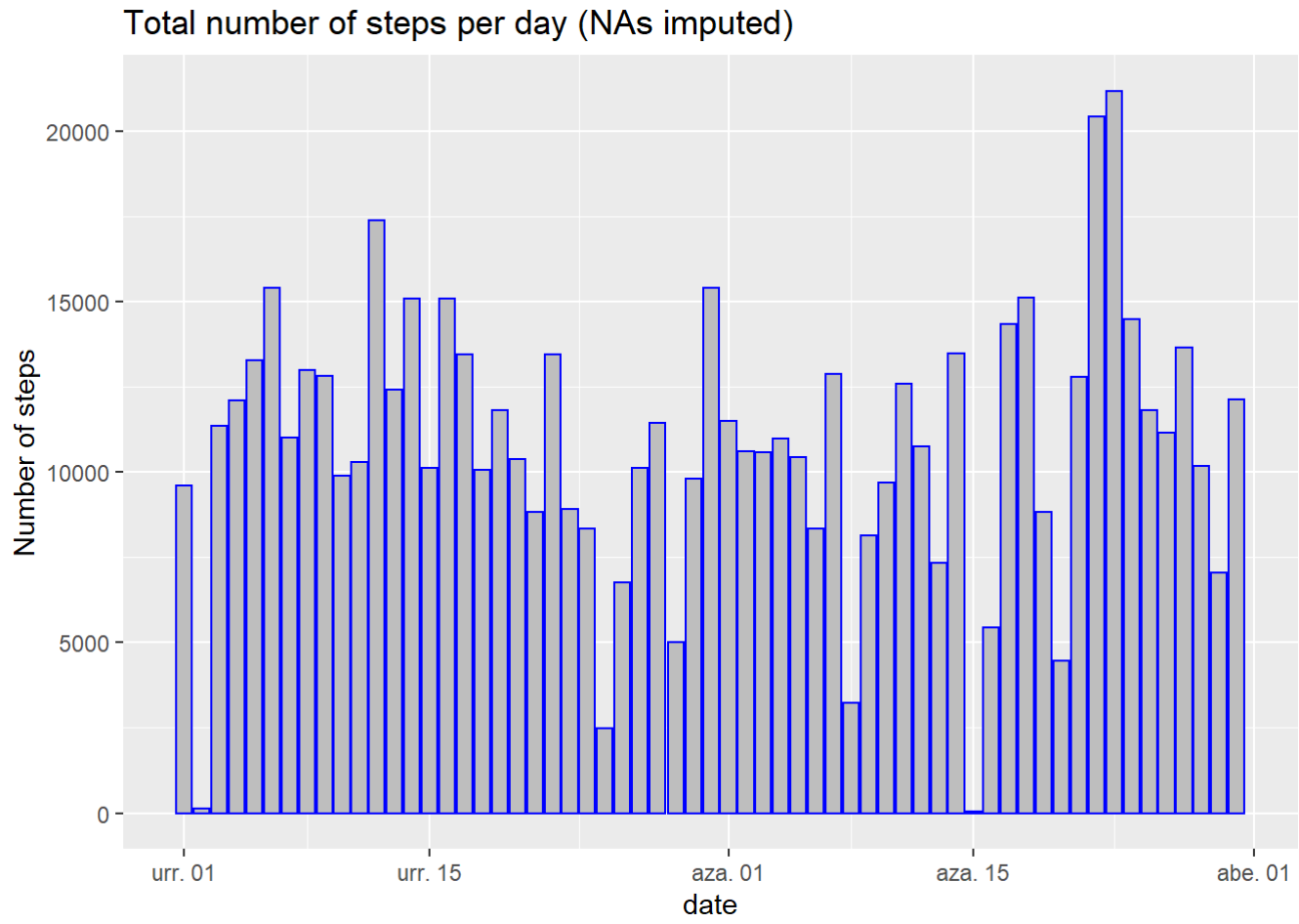
Firstly, the total number of steps per day is calculated and plotted again as done in step 2, but this time using imputed values:

```
readingImputed$date <- as.POSIXct(strptime(readingImputed$date, "%Y-%m-%d"))
mySumPerDate <- readingImputed %>% group_by(date) %>% summarize(sumSteps=sum(steps))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(mySumPerDate, aes(x = date, y = sumSteps), stat = "bin") +
geom_histogram(stat = "identity", col = "blue", bg = "gray") +
ggtitle("Total number of steps per day (NAs imputed)") + ylab("Number of steps")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



As one can see, the plot is slightly different from the one obtained in step 2: as a consequence of the NA imputation process, this time there is no date with missing data.

Secondly, the mean and the median are calculated again as done in step 2, but this time using imputed data:

```
mean(mySumPerDate$sumSteps, na.rm=TRUE)
```

```
## [1] 10806.3
```

```
median(mySumPerDate$sumSteps, na.rm=TRUE)
```

```
## [1] 10995
```

As one can see, the results are slightly different from the ones achieved in step 2.

---

8) Panel plot comparing the average number of steps taken per 5-minute interval across

weekdays and weekends.

Firstly, a factor must be created to differentiate between weekdays and weekends. To do so, the wday() function will be used.

In order to use the wday() function, the “lubridate” package must be installed first. If not done yet, the below install.packages() function can be used by removing the # symbol from the beginning of the code-line:

```
# install.packages("lubridate")
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
day <- wday(readingImputed$date)


for(i in 1:length(day))
if (day[i] == 1 | day[i] == 7) {
  day[i] <- "weekend"
  } else {
    day[i] <- "weekday"
  }

readingDay <- cbind(readingImputed,day)
readingWD <- readingDay[day == "weekday",]
readingWE <- readingDay[day == "weekend",]

myMeanPerIntWD <- readingWD %>%  group_by(interval) %>% summarize(meanStepsWD=mean(steps,na.r
m=TRUE))
```

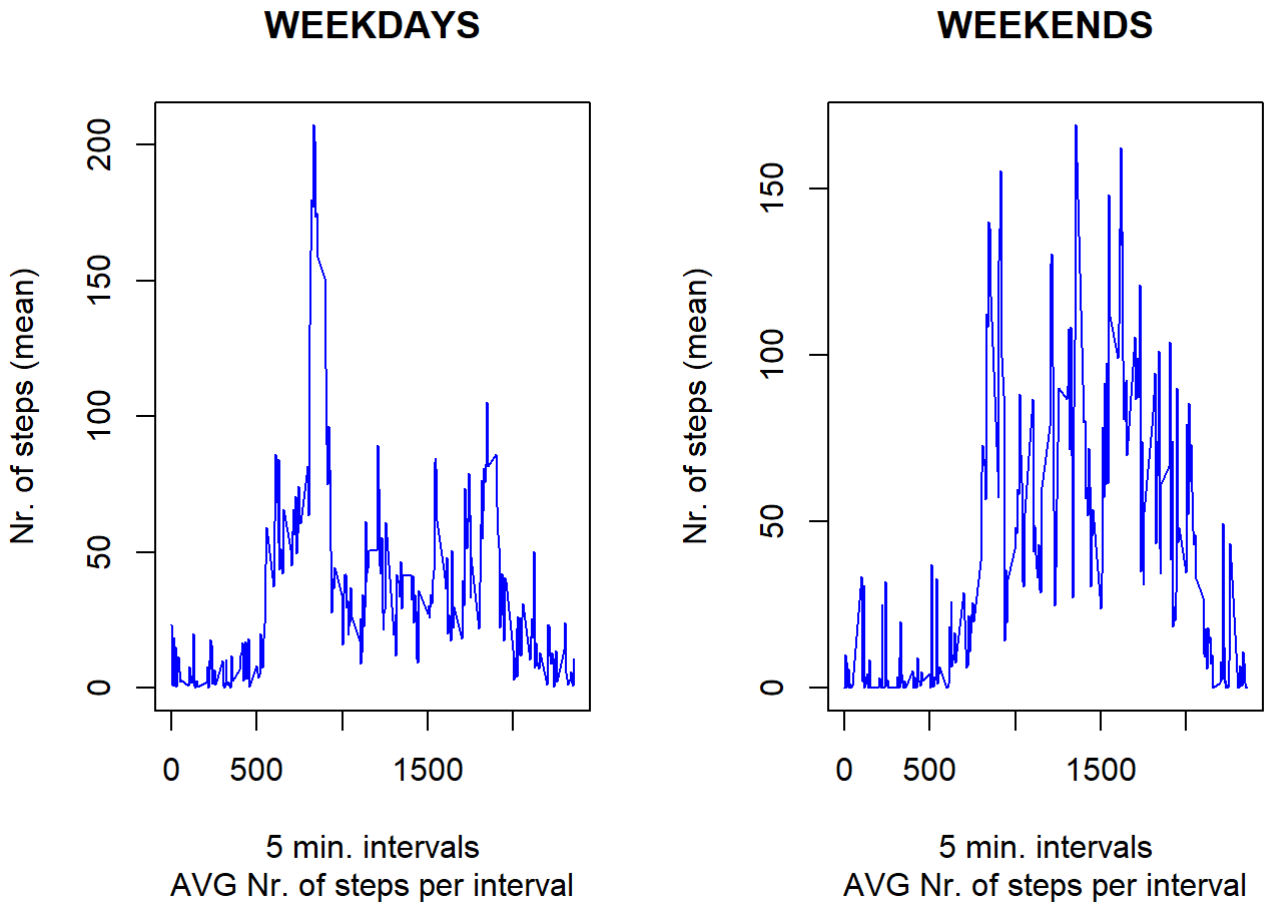
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
myMeanPerIntWE <- readingWE %>%  group_by(interval) %>% summarize(meanStepsWE=mean(steps,na.r
m=TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Secondly, the panel plot is created:

```
par(mfrow = c(1,2))
plot(x=myMeanPerIntWD$interval, y=myMeanPerIntWD$meanStepsWD, type = "l",
     main = "WEEKDAYS", sub = "AVG Nr. of steps per interval",
     xlab = "5 min. intervals",
     ylab = "Nr. of steps (mean)",
     col = "blue")
plot(x=myMeanPerIntWE$interval, y=myMeanPerIntWE$meanStepsWE, type = "l",
     main = "WEEKENDS", sub = "AVG Nr. of steps per interval",
     xlab = "5 min. intervals",
     ylab = "Nr. of steps (mean)",
     col = "blue")
```



9) All of the R code needed to reproduce the results (numbers, plots, etc.) in the report

The R code needed to reproduce the results is described in the previous steps (from 1 to 8).