Course Project 1. Reproducible Research (Week 2)

Christian Jay

2022-06-23

1) Code for reading in the dataset and/or processing the data

Packages loading required for plotting data, mostly using ggplot2

```
library(ggplot2)
library(lattice)
```

Data importing from a csv file and generation of a new data set without NAs

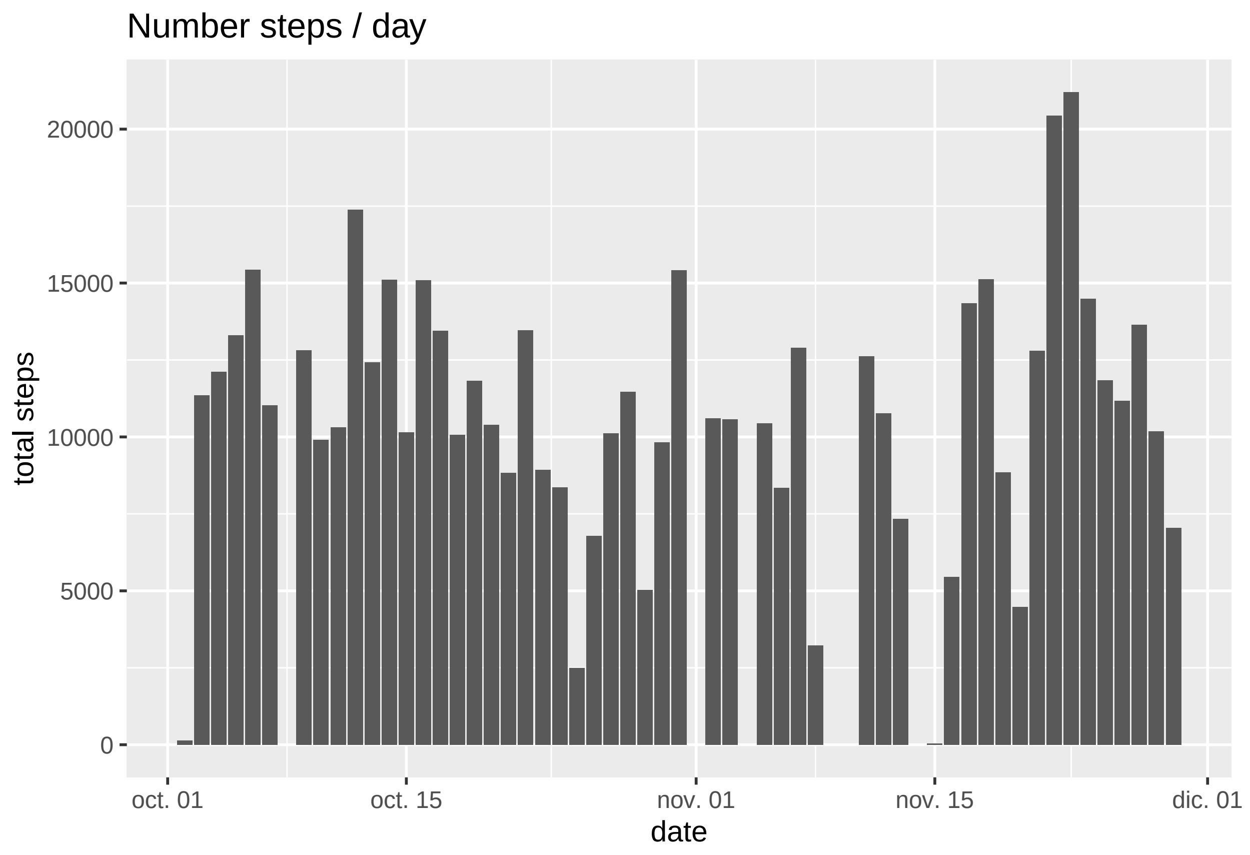
```
data <- read.csv("A:/Project Y06_Pre-Clinical Experiments/R Course/homework/Week 2 Course Project 1/repo

# data <- read.csv("A:/Project Y06_Pre-Clinical Experiments/R Course/homework/Week 2 Course
# Project 1/repdata_data_activity/activity.csv", header = TRUE, sep = ",")

data <- transform(data, date = as.Date(date))
adata <- data[!is.na(data[1]),]
```

Date transforming into days format using a loop for easier plotting. Coding steps per steps function

```
get_steps_per_day <- function(d) {
  steps_per_day <- tapply(d$steps, d$date, sum)
  steps_per_day <- data.frame(cbind(day = names(steps_per_day), steps = steps_per_day))

steps_per_day <- transform(steps_per_day, day = as.Date(day))
steps_per_day <- transform(steps_per_day, steps = as.numeric(as.character(steps)))}
```

2) Histogram of the total number of steps taken each day. NAs can be ignored, so data was used instead of adata.

```
## Warning: Removed 2304 rows containing missing values (position_stack).
```



3) Mean and median number of steps taken each day. Coding steps per interval function

```
get_steps_per_interval <- function(d) {
  steps_per_interval <- tapply(d$steps, d$interval, mean)
  steps_per_interval <-
    data.frame(cbind(interval = names(steps_per_interval), steps = steps_per_interval))

  rownames(steps_per_interval) <- NULL

  steps_per_interval <- transform(steps_per_interval, interval = as.numeric(as.character(interval)))
  steps_per_interval <- transform(steps_per_interval, steps = as.numeric(as.character(steps)))
}

total_steps_per_day_adata <- get_steps_per_day(adata)
```

```
mean(total_steps_per_day_adata$steps)
```

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

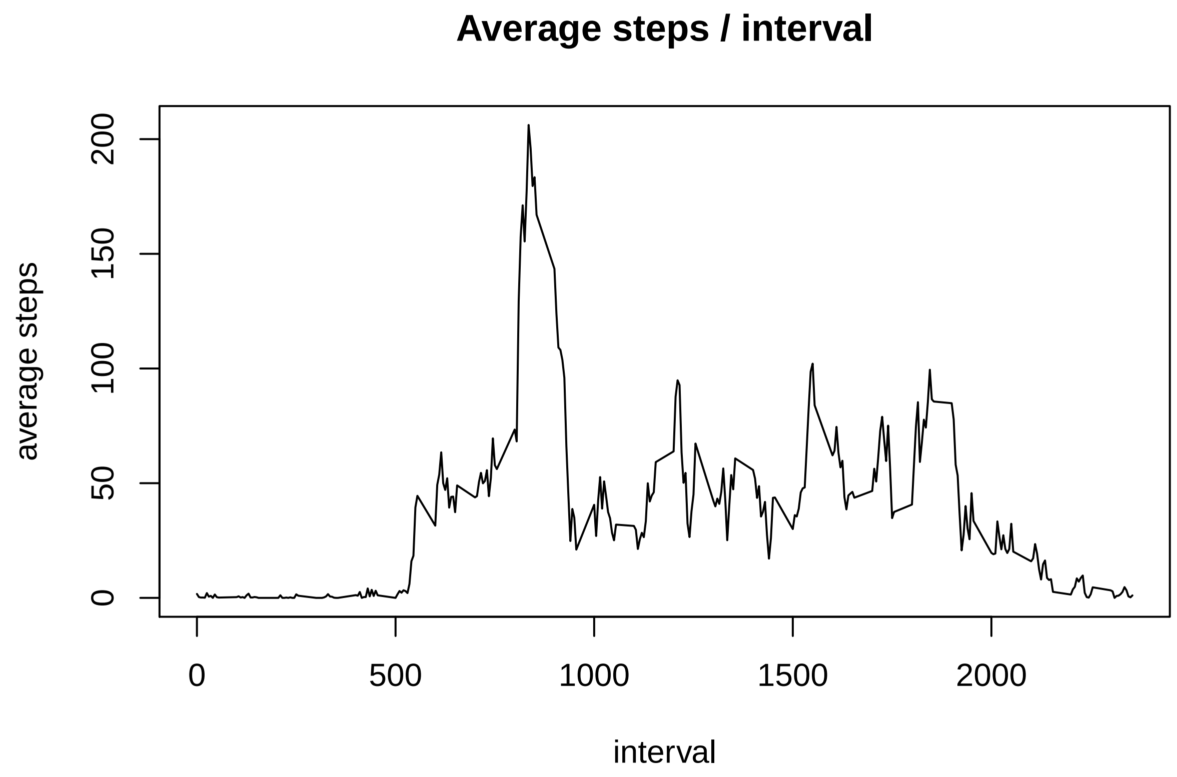
```
median(total_steps_per_day_adata$steps)
```

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

```
# The mean total steps per day was 10766.19
# The median steps per day was 10765
```

4) Time series plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days (y-axys).

```
average_steps_per_interval_adata <- get_steps_per_interval(adata)
# Subset of 288 obs and 2 variables
```



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

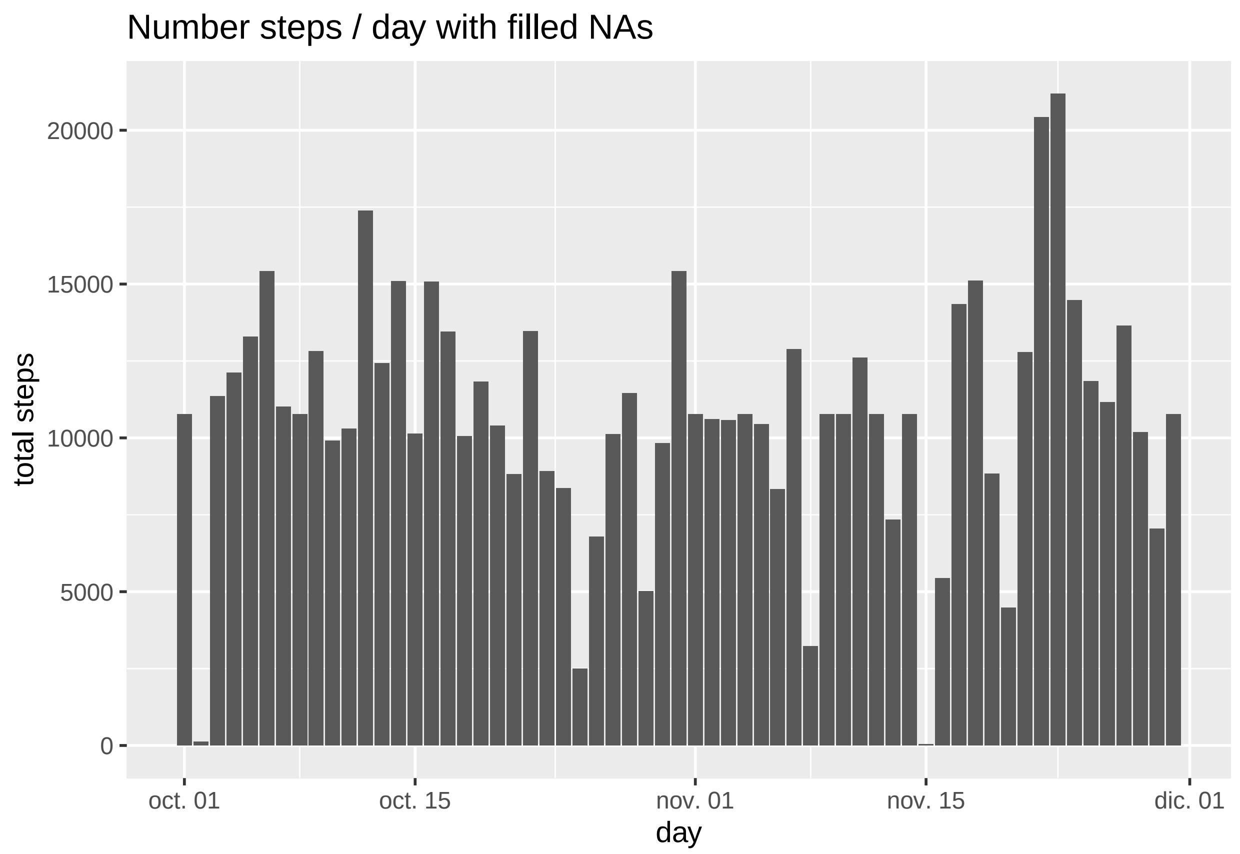
```
is_max <- average_steps_per_interval_adata$steps == max(average_steps_per_interval_adata$steps)

max_per_interval <- average_steps_per_interval_adata[is_max,]
interval_for_max <- max_per_interval$interval
# 835 interval has 206.1698 steps, which correlates with the maximum peak in previous plot
```

6) Code for imputing missing data. Calculate and report the total number of NAs in the dataset. Devise a strategy for filling in all of NAs. You can use the mean/median for that day or the mean for that 5-minute interval. Create a new dataset equal to the original but with the NAs filled in

```
total_Nas <- sum(is.na(data$steps))
# 2304 NAs in data. This is correct because data - NAs = adata (17568 - 2304 = 15264obs)

filled_data <- data
```

```r
na_index <- which(is.na(filled_data$steps))

for(i in na_index) {
  interval <- filled_data$interval[i]
  filled_data$steps[i] <- average_steps_per_interval_adata[average_steps_per_interval_adata$interval
                                                         == interval,]$steps
}

# The new object filled_data, or fdata for short, has the same number of elements than the
# original data set with NAs, data. NAs have been replaced with average steps per interval for
# each NAs. Filled data have the same number of observations as data has: 17568 obs and 3 variables.
```

7) Histogram of the total number of steps taken each day after NAs are imputed. Calculate the mean and median total number of steps taken per day. What is the impact of imputing missing data on the estimates of the total daily number of steps?



```r
mean(tot_steps_day_fdata$steps)
```

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

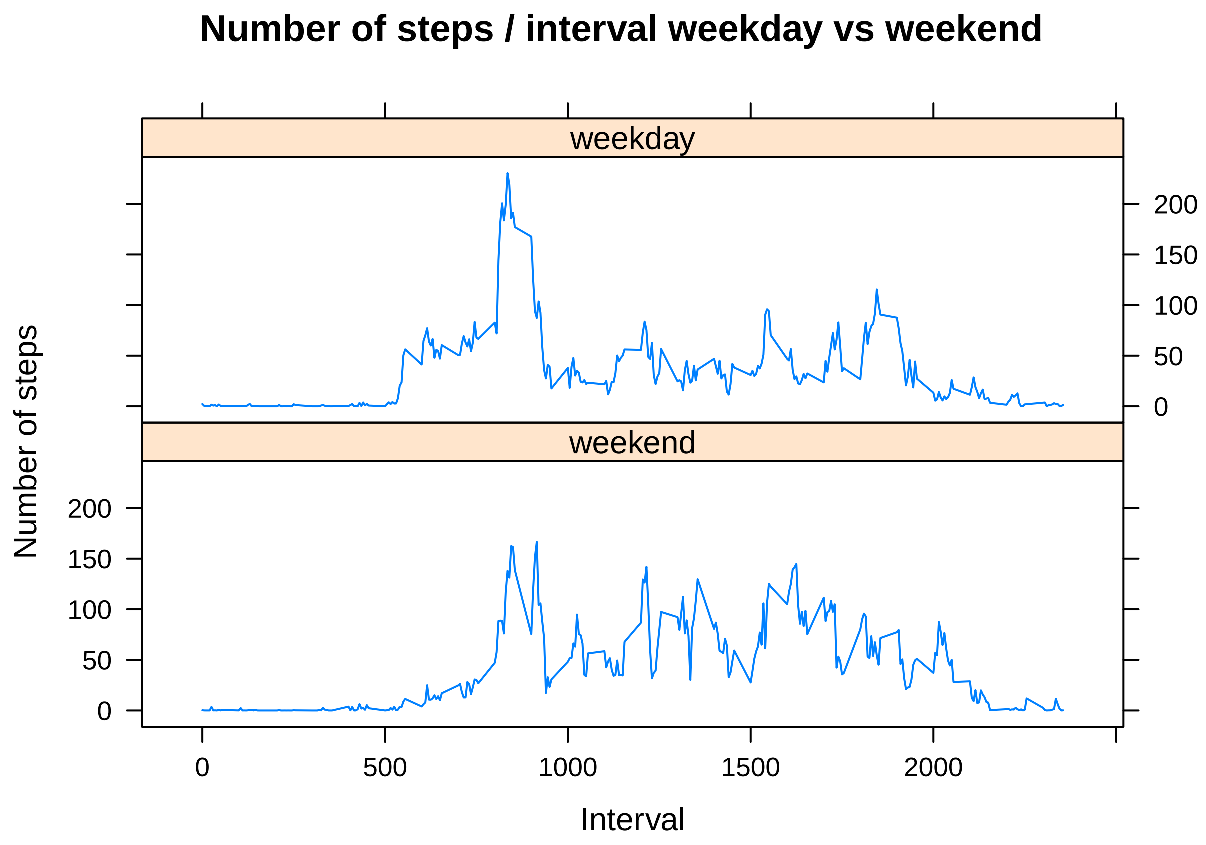
```r
median(tot_steps_day_fdata$steps)
```

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

```
# The mean of steps with filled NAs is 10766.19 vs 10766.19, which is the exact number
# The median of steps with filled NAs is 10766.19 vs 10765, which is almost the exact number

# Filling missing values with the average steps per interval does not impact the total daily number
# of steps, despite the plots between data and fdata look different
```

8) Panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends.Are there differences in activity patterns between weekdays and weekends? Use weekdays() function and the data set with the filled-in missing values for this part. Create a new factor variable in the data set 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 of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all weekday days or weekend days (y-axis).

```
weekdays_list <- c("lunes", "martes", "miércoles", "jueves", "viernes")
# Days were written in Spanish due to R version using weekdays() function

filled_data$day_of_week <- weekdays(filled_data$date)
filled_data$type_of_day <- factor(filled_data$day_of_week %in% weekdays_list,
                                  levels = c(TRUE, FALSE),
                                  labels = c("weekday", "weekend"))

# First of all, new variables need to be included in our fdata: day of week and week type
# and type of day

weekday_data <- filled_data[filled_data$type_of_day == "weekday",]
weekend_data <- filled_data[filled_data$type_of_day == "weekend",]

weekday_steps <- get_steps_per_interval(weekday_data)
weekend_steps <- get_steps_per_interval(weekend_data)

weekday_steps$type_of_day <- factor(TRUE, levels = c(FALSE, TRUE), labels = c("weekend", "weekday"))
weekend_steps$type_of_day <- factor(FALSE, levels = c(FALSE, TRUE), labels = c("weekend", "weekday"))

weekday_weekend_steps <- rbind(weekday_steps, weekend_steps)
xyplot(steps ~ interval | type_of_day, data = weekday_weekend_steps, layout = c(1,2), type = "l",
       xlab = "Interval", ylab = "Number of steps", main = "Number of steps / interval weekday vs weeke
```



```
# After assigning these variables, we subset the number of steps for weekdays and weekends
# and proceed to performed the desired time plot of number of steps for each interval in weekdays
# vs weekend (Figure 4)
```