Reproducible Research Course Project 1

Jeffrey M. Hunter

20 May, 2019

Contents

Course Project

Reproducible Research Course Project 1

Peer-graded Assignment

- This course project is available on GitHub

  Reproducible Research Course Project 1

Synopsis

It is now possible to collect a large amount of data about personal movement using activity monitoring devices such as a Fitbit, Nike Fuelband, or Jawbone Up. These type of devices are part of the "quantified self" movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. But these data remain under-utilized both because the raw data are hard to obtain and there is a lack of statistical methods and software for processing and interpreting the data.

This assignment makes use of data from a personal activity monitoring device. This device collects data at 5 minute intervals through out the day. The data consists of two months of data from an anonymous individual collected during the months of October and November, 2012 and include the number of steps taken in 5 minute intervals each day.

The variables included in this dataset are:

- **steps**: Number of steps taking in a 5-minute interval (missing values are coded as `NA`)
- **date**: The date on which the measurement was taken in YYYY-MM-DD format
- **interval**: Identifier for the 5-minute interval in which measurement was taken

The dataset is stored in a comma-separated-value (CSV) file and there are a total of 17,568 observations in this dataset.

Environment Setup

Load packages used in this analysis.

```
if (!require(ggplot2)) {
    install.packages("ggplot2")
    library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
if (!require(dplyr)) {
    install.packages("dplyr")
    library(dplyr, warn.conflicts = FALSE)
}
```

```
## Loading required package: 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
```

Display session information.

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] dplyr_0.8.1   ggplot2_3.1.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1       knitr_1.23       magrittr_1.5     tidyselect_0.2.5
##  [5] munsell_0.5.0    colorspace_1.4-1 R6_2.4.0         rlang_0.3.4
##  [9] stringr_1.4.0    plyr_1.8.4       tools_3.6.0      grid_3.6.0
## [13] packrat_0.5.0    gtable_0.3.0     xfun_0.7         withr_2.1.2
## [17] htmltools_0.3.6  assertthat_0.2.1 yaml_2.2.0       lazyeval_0.2.2
## [21] digest_0.6.18    tibble_2.1.1     crayon_1.3.4     purrr_0.3.2
## [25] glue_1.3.1       evaluate_0.13    rmarkdown_1.12   stringi_1.4.3
## [29] compiler_3.6.0   pillar_1.4.0     scales_1.0.0     pkgconfig_2.0.2
```

Questions

Reports will be run to answer specified questions in the homework assignment.

Loading and preprocessing the data

Load and process the dataset

```
setwd("~/repos/coursera/github-assignments/reproducible-research-course-project-1")
activityDataFile <- "data/activity.csv"
if (!file.exists(activityDataFile)) {
    tempFile <- tempfile()
    download.file("https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip", tempFile)
    unzip(tempFile, exdir = "data")
    unlink(tempFile)
}
activityData <- read.csv(activityDataFile, sep = ",")
activityData$date <- as.POSIXct(activityData$date, format = "%Y-%m-%d", tz = "")
```

Display dataset summary

```
names(activityData)
```

```
## [1] "steps"    "date"     "interval"
```

```
str(activityData)
```

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

```
summary(activityData)
```

```
##      steps             date                        interval
##  Min.   :  0.00   Min.   :2012-10-01 00:00:00   Min.   :   0.0
##  1st Qu.:  0.00   1st Qu.:2012-10-16 00:00:00   1st Qu.: 588.8
##  Median :  0.00   Median :2012-10-31 00:00:00   Median :1177.5
##  Mean   : 37.38   Mean   :2012-10-31 00:25:34   Mean   :1177.5
##  3rd Qu.: 12.00   3rd Qu.:2012-11-15 00:00:00   3rd Qu.:1766.2
##  Max.   :806.00   Max.   :2012-11-30 00:00:00   Max.   :2355.0
##  NA's   :2304
```

```
head(activityData)
```

steps	date	interval
NA	2012-10-01	0
NA	2012-10-01	5
NA	2012-10-01	10
NA	2012-10-01	15
NA	2012-10-01	20
NA	2012-10-01	25

What is mean total number of steps taken per day?

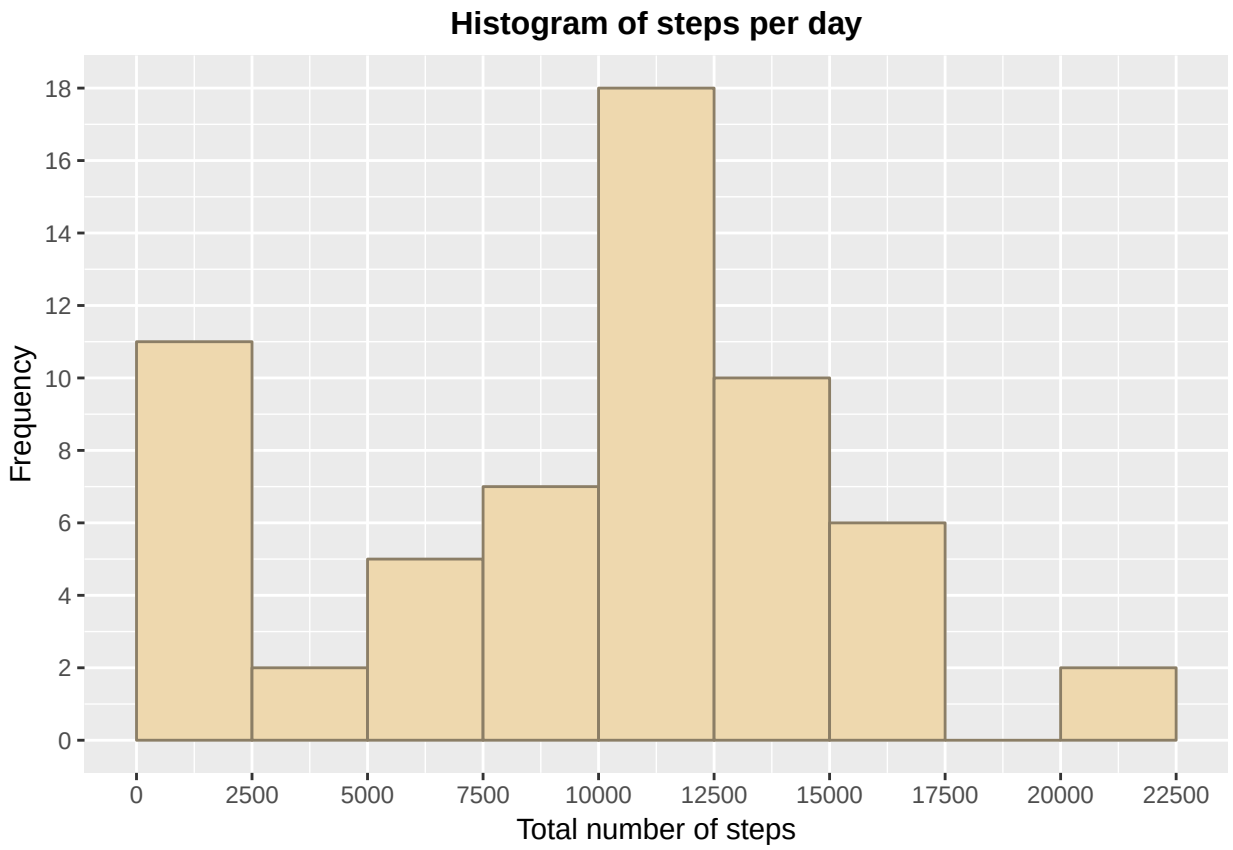
Calculate steps per day (excluding missing values)

```
stepsPerDay <- with(activityData, aggregate(steps, list(date), FUN = sum, na.rm = TRUE))
colnames(stepsPerDay) <- c("date", "steps")
head(stepsPerDay)
```

date	steps
2012-10-01	0
2012-10-02	126
2012-10-03	11352
2012-10-04	12116
2012-10-05	13294
2012-10-06	15420

Display histogram

```
g <- ggplot(stepsPerDay, aes(stepsPerDay$steps))
g + geom_histogram(boundary = 0, binwidth = 2500, col = "wheat4", fill = "wheat2") + ggtitle("Histogram
```



Mean and median number of steps taken each day

Mean (excluding missing values)

```
mean(stepsPerDay$steps, na.rm = TRUE)
```

```
## [1] 9354.23
```

Median (excluding missing values)

```
median(stepsPerDay$steps, na.rm = TRUE)
```

```
## [1] 10395
```

Note: adding `na.rm = TRUE` was not necessary since the `stepsPerDay` data frame already excluded `NA` values.

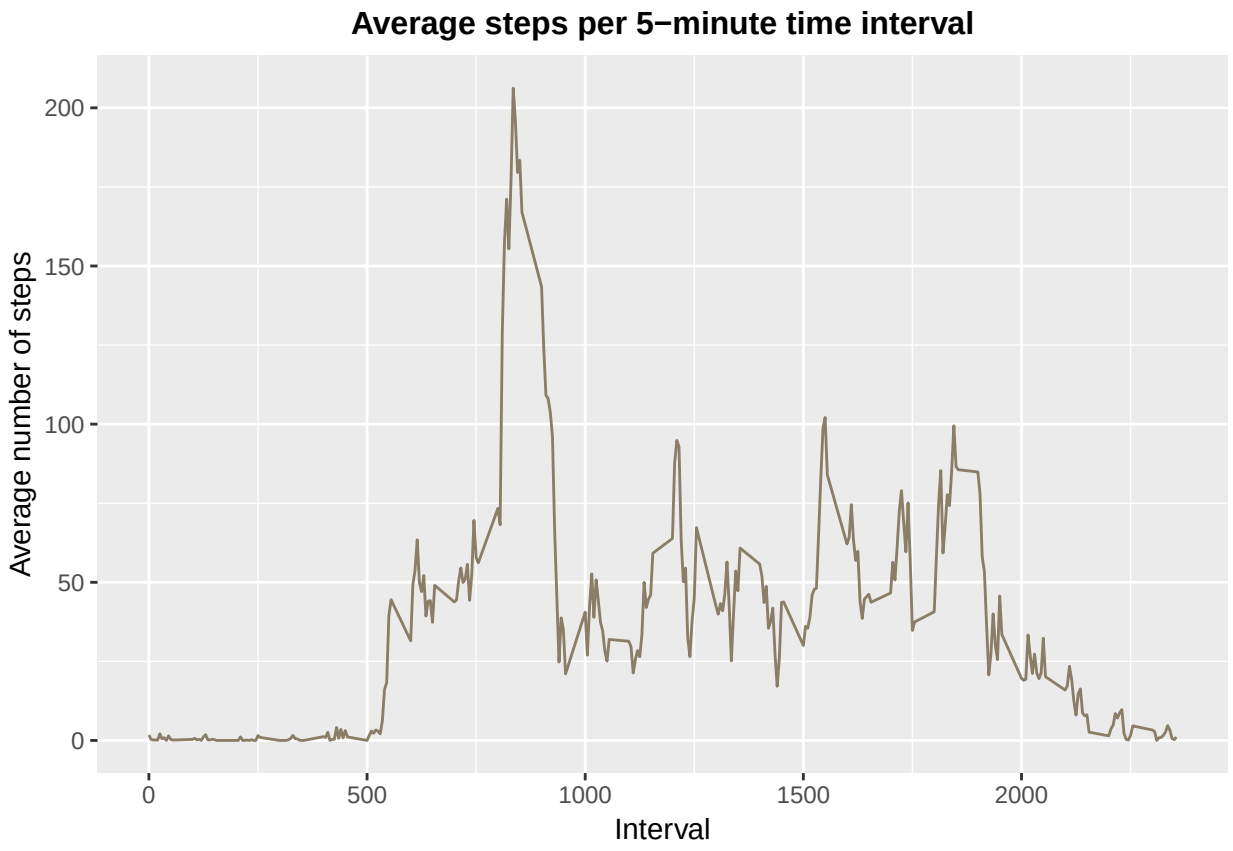
What is the average daily activity pattern?

Calculate steps per time interval

```
stepsPerIntervalAvg <- aggregate(steps ~ interval, data = activityData, FUN = mean, na.action = na.omit)
colnames(stepsPerIntervalAvg) <- c("interval", "steps")
```

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)

```
g <- ggplot(stepsPerIntervalAvg, aes(stepsPerIntervalAvg$interval, stepsPerIntervalAvg$steps))
g + geom_line(col = "wheat4") + ggtitle("Average steps per 5-minute time interval") + xlab("Interval") +
```



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

```
activityData %>% group_by(interval) %>%
    summarize(meanByInterval = mean(steps, na.rm = TRUE)) %>%
    filter(meanByInterval == max(meanByInterval))
```

interval	meanByInterval
835	206.1698

Imputing missing values

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.

Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with `NA`s)

```
sum(is.na(activityData$steps) == TRUE)
```

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

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.

Strategy: Add a new column to the origin dataset named **stepsCompleted** that replaces missing values with the rounded average of the 5-minute interval.

```
activityData$stepsCompleted <- ifelse(is.na(activityData$steps), round(stepsPerIntervalAvg$steps[match(a
```

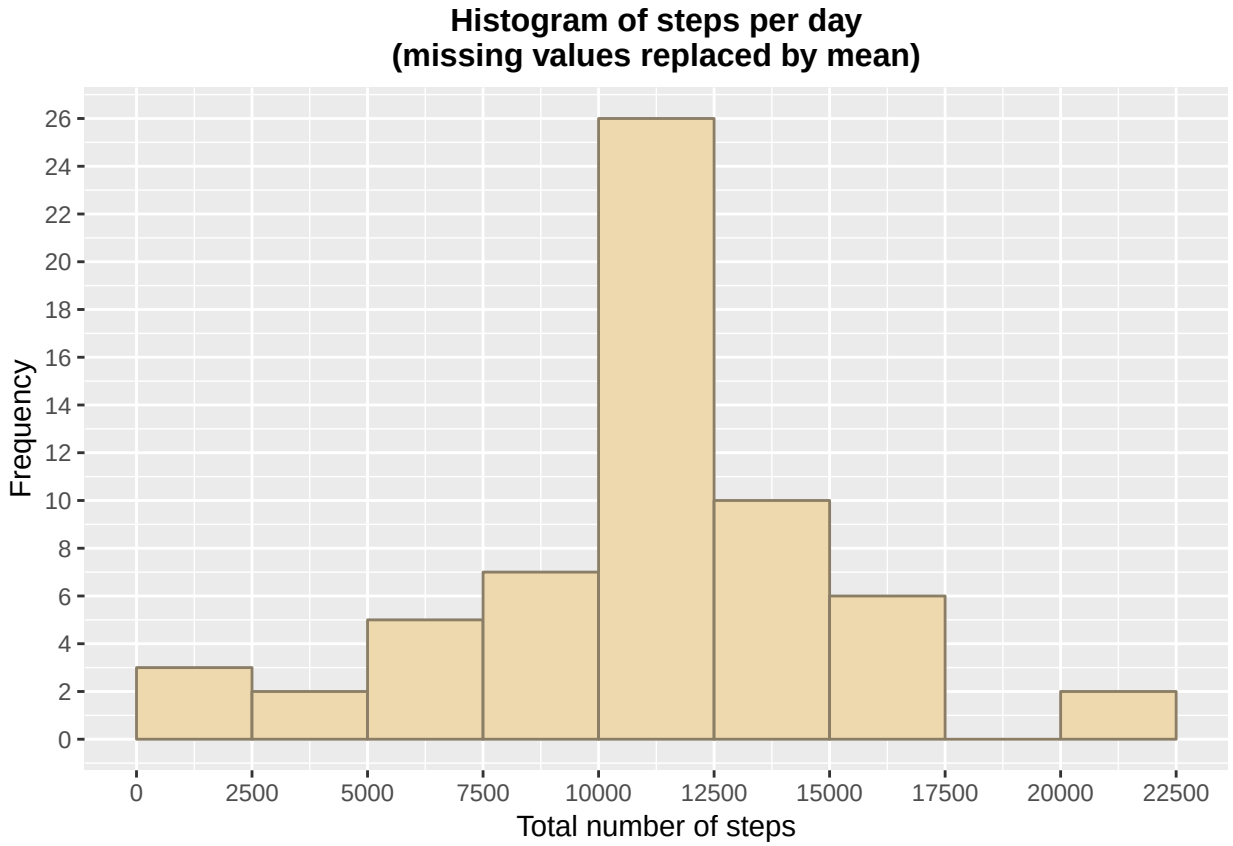
Create a new dataset that is equal to the original dataset but with the missing data filled in.

```
activityDataNoNA <- data.frame(steps = activityData$stepsCompleted, interval = activityData$interval, da
head(activityDataNoNA, n = 10)
```

steps	interval	date
2	0	2012-10-01
0	5	2012-10-01
0	10	2012-10-01
0	15	2012-10-01
0	20	2012-10-01
2	25	2012-10-01
1	30	2012-10-01
1	35	2012-10-01
0	40	2012-10-01
1	45	2012-10-01

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?

```
stepsPerDayCompleted <- aggregate(activityDataNoNA$steps, list(activityDataNoNA$date), FUN = sum)
colnames(stepsPerDayCompleted) <- c("date", "steps")
g <- ggplot(stepsPerDayCompleted, aes(stepsPerDayCompleted$steps))
g + geom_histogram(boundary = 0, binwidth = 2500, col = "wheat4", fill = "wheat2") + ggtitle("Histogram
```



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

Mean

```
mean(stepsPerDayCompleted$steps)
```

```
## [1] 10765.64
```

Median

```
median(stepsPerDayCompleted$steps)
```

```
## [1] 10762
```

1. Do these values differ from the estimates from the first part of the assignment?

   They do differ, but not significantly when looking at the mean and the median of the total daily number of steps.

2. What is the impact of imputing missing data on the estimates of the total daily number of steps?

   Reviewing the histogram, the only two bins that were impacted are the intervals 0 - 2500 and 10000 - 12500 steps; the latter of which grew from a frequency of 18 to a frequency of 26 (44%). Instead of replacing missing values with the mean, such as the mode or median, may have produced different results.

   Statistic

   Original Dataset

   Imputed Dataset

   Difference

mean

9,354.23

10,765.64

1,411.41 (15%)

median

10,395

10,762

367 (3%)

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.

In this example, I created the following two factor variables:

- **weekDay**: indicate the day of the week (non-abbreviated)
- **dayType**: indicate whether the date is a weekday or a weekend

The current date variable in the dataset has already been formatted to represent a date value using the `as.POSIXct()` function.

```
# create a factor variable indicating the day of the week
weekDay <- weekdays(activityDataNoNA$date, abbreviate = FALSE)
activityDataNoNA <- cbind(activityDataNoNA, weekDay)
names(activityDataNoNA)[4] <- "weekDay"

# create a factor variable indicating weekday or weekend
dayType <- ifelse(activityDataNoNA$weekDay == 'Saturday' | activityDataNoNA$weekDay == 'Sunday', 'weeke
activityDataNoNA <- cbind(activityDataNoNA, dayType)
names(activityDataNoNA)[5] <- "dayType"

# let's see the first 10 observations
head(activityDataNoNA, n = 10)
```

steps	interval	date	weekDay	dayType
2	0	2012-10-01	Monday	weekday
0	5	2012-10-01	Monday	weekday
0	10	2012-10-01	Monday	weekday
0	15	2012-10-01	Monday	weekday
0	20	2012-10-01	Monday	weekday
2	25	2012-10-01	Monday	weekday
1	30	2012-10-01	Monday	weekday
1	35	2012-10-01	Monday	weekday
0	40	2012-10-01	Monday	weekday
1	45	2012-10-01	Monday	weekday

Prepare data:

- create a data frame `stepsPerTime` that represents average steps per time interval by weekday/weekend
- add a `time` variable to display the time interval average in hours

```
stepsPerTime <- aggregate(steps ~ interval + dayType, data = activityDataNoNA, FUN = mean, na.action = r
stepsPerTime$time <- stepsPerTime$interval/100
head(stepsPerTime, n = 10)
```

interval	dayType	steps	time
0	weekday	2.2888889	0.00
5	weekday	0.4000000	0.05
10	weekday	0.1555556	0.10
15	weekday	0.1777778	0.15
20	weekday	0.0888889	0.20
25	weekday	1.5777778	0.25
30	weekday	0.7555556	0.30
35	weekday	1.1555556	0.35
40	weekday	0.0000000	0.40
45	weekday	1.7333333	0.45

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).

```
g <- ggplot(stepsPerTime, aes(time, steps))
g + geom_line(col = "wheat4") + ggtitle("Average steps per time interval\n(weekdays vs. weekends)") + xl
```