Reproducible Research: Activity Monitoring

Daniel Escasa

```
{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)
```

Introduction

Excerpts from the original repo from which this was forked

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 project makes use of data from a personal activity monitoring device which collects data at five-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 five-minute intervals each day.

Variable name	Description
steps	Number of steps taking in a five-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 five-minute interval in which measurement was taken

The following are the questions to be addressed:

- What is mean total number of steps taken per day?
- What is the average daily activity pattern?
- Are there differences in activity patterns between weekdays and weekends?

Boring admin stuff

Install ggplot2, knitr, and scales if not yet installed

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

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

```
if (!require("knitr")) {
  message("Installing knitr")
  install.packages("knitr")
}
```

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

```
if (!require("scales")) {
  message("Installing scales")
  install.packages("scales")
}
```

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

Download the zipped dataset if not yet present

```
if (!file.exists("activity.zip")) {
          message("Downloading dataset")
          download.file("https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip",
                        destfile = "activity.zip",
                        method   = "internal",
                        mode     = "wb")
     }
```

Unzip the dataset if not yet present

```
if (!file.exists("activity.csv")) {
          message("Extracting dataset")
          unzip("activity.zip",
                overwrite = FALSE)
     }
```

Read in the data

```
# Shouldn't need error checking here
activity <- read.csv("activity.csv")
```

Load the libraries

```
library(ggplot2)
library(knitr)
library(scales)
```

Now let's get cooking

What is mean total number of steps taken per day?

Compute the total number, mean, and median steps per day. As an aside, take note of the use of the superassignment (`<<-`) instead of the usual assignment operator (`<-`). This is because the `with(){}` creates its own scope. And yes, the scope applies to assigning to `colnames()`.

```
with(data = activity,{
     totalDaySteps  <<- aggregate(steps, by = list(date), FUN = sum)
```

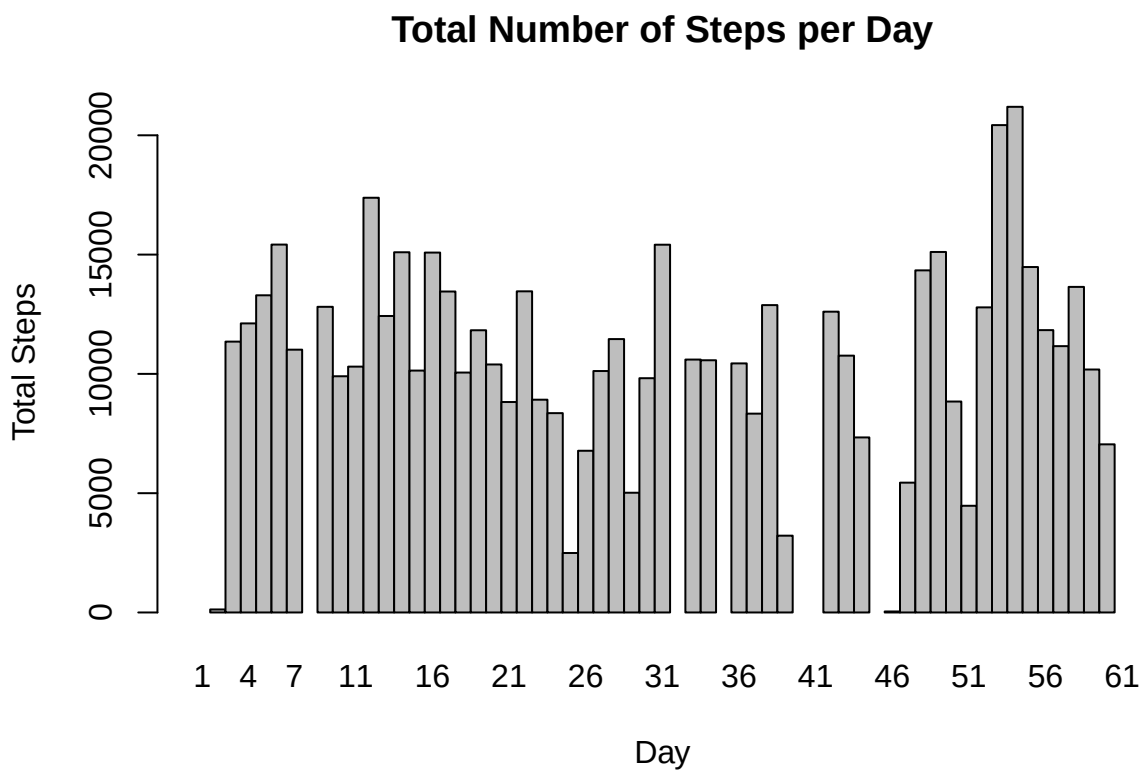
```
     colnames(totalDaySteps) <<- c("date", "steps")
     meanDaySteps   <<- aggregate(steps, by = list(date), FUN = mean)
     colnames(meanDaySteps) <<- c("date", "steps")
     medianDaySteps <<- aggregate(steps, by = list(date), FUN = median)
     colnames(medianDaySteps) <<- c("date", "steps")
})
```

For convenience, merge the three into one dataframe.

```
dayStats           <- merge(totalDaySteps, meanDaySteps, by = "date")
dayStats           <- merge(dayStats, medianDaySteps, by = "date")
colnames(dayStats) <- c("date", "totalSteps", "meanSteps", "medianSteps")
```

Plot total number of steps taken each day, telling `barplot()` to remove spaces between bars. Also, using `(as.Date(date) - as.Date(totalDaySteps[1, 1]) + 1)` as `names.arg` so labels on x axis are day numbers, not the dates. This is so the plot is neater.

```
with(dayStats,
     barplot(space = 0, totalSteps,
             main  = "Total Number of Steps per Day",
             xlab  = "Day", ylab = "Total Steps",
             names.arg = (as.Date(date) - as.Date(totalDaySteps[1, 1]) + 1)))
```




```
dev.copy(png, "plots/01-totalSteps.png")
```

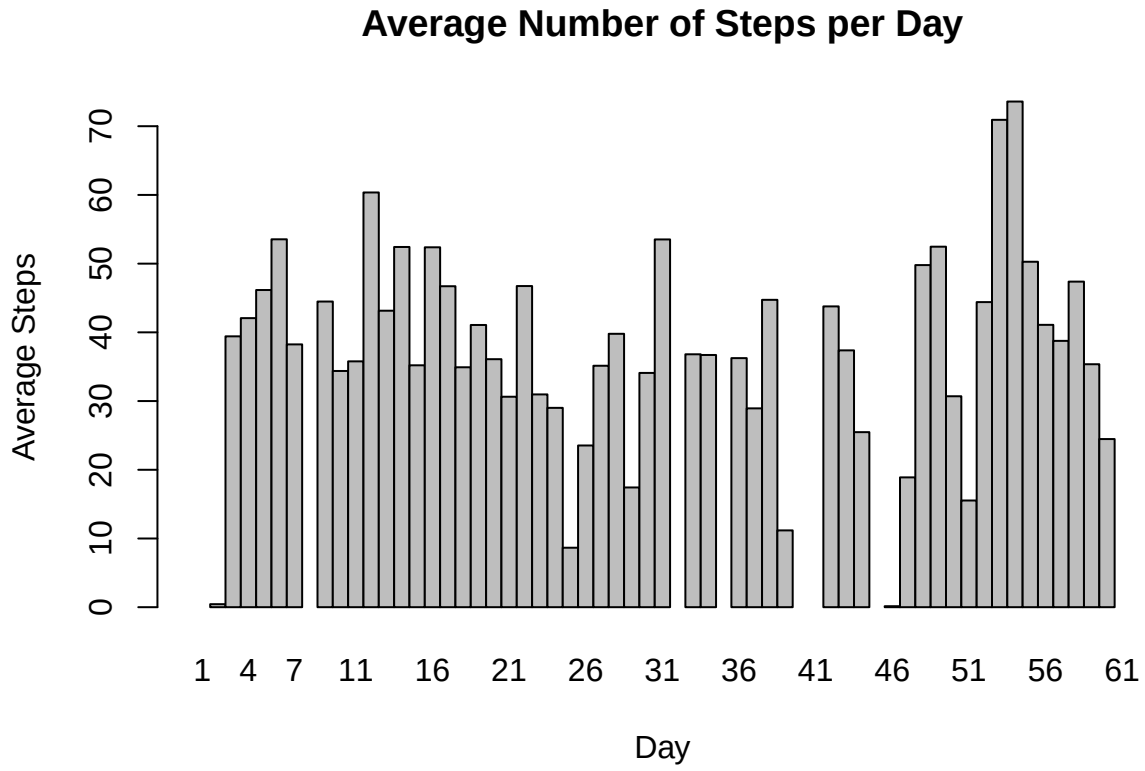
```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

Plot mean of number of steps per day.

```
with(dayStats,
     barplot(space = 0, meanSteps,
             main  = "Average Number of Steps per Day",
             xlab  = "Day", ylab = "Average Steps",
             names.arg = (as.Date(date) - as.Date(totalDaySteps[1, 1]) + 1)))
```




```
dev.copy(png, "plots/02-meanSteps.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

And here's the dataframe of average number of steps per day:

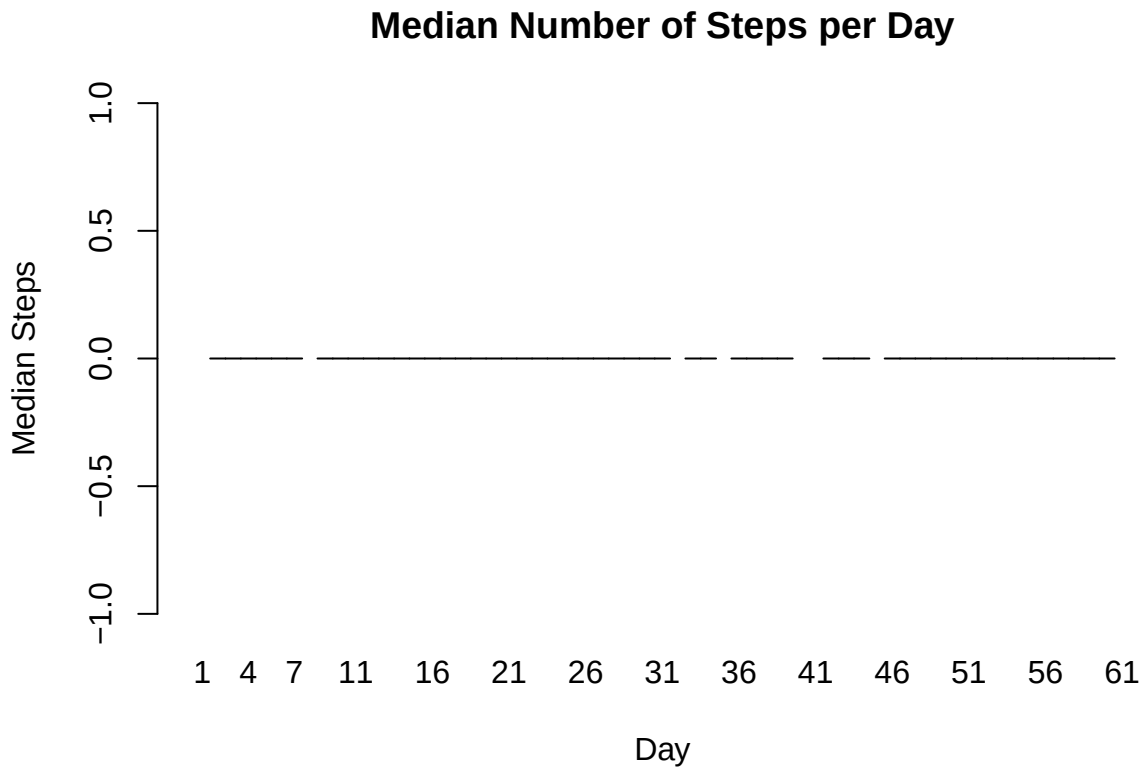
```
kable(dayStats[, c(1, 3)],
      col.names = c("date", "mean"))
```

date	mean
2012-10-01	NA
2012-10-02	0.4375000
2012-10-03	39.4166667
2012-10-04	42.0694444
2012-10-05	46.1597222
2012-10-06	53.5416667
2012-10-07	38.2465278
2012-10-08	NA
2012-10-09	44.4826389
2012-10-10	34.3750000
2012-10-11	35.7777778
2012-10-12	60.3541667
2012-10-13	43.1458333
2012-10-14	52.4236111
2012-10-15	35.2048611
2012-10-16	52.3750000
2012-10-17	46.7083333
2012-10-18	34.9166667
2012-10-19	41.0729167
2012-10-20	36.0937500
2012-10-21	30.6284722
2012-10-22	46.7361111
2012-10-23	30.9652778
2012-10-24	29.0104167
2012-10-25	8.6527778
2012-10-26	23.5347222
2012-10-27	35.1354167
2012-10-28	39.7847222
2012-10-29	17.4236111
2012-10-30	34.0937500
2012-10-31	53.5208333
2012-11-01	NA
2012-11-02	36.8055556
2012-11-03	36.7048611
2012-11-04	NA
2012-11-05	36.2465278
2012-11-06	28.9375000
2012-11-07	44.7326389
2012-11-08	11.1770833
2012-11-09	NA
2012-11-10	NA
2012-11-11	43.7777778
2012-11-12	37.3784722
2012-11-13	25.4722222
2012-11-14	NA
2012-11-15	0.1423611
2012-11-16	18.8923611
2012-11-17	49.7881944
2012-11-18	52.4652778
2012-11-19	30.6979167
2012-11-20	15.5277778
2012-11-21	44.3993056

date	mean
2012-11-22	70.9270833
2012-11-23	73.5902778
2012-11-24	50.2708333
2012-11-25	41.0902778
2012-11-26	38.7569444
2012-11-27	47.3819444
2012-11-28	35.3576389
2012-11-29	24.4687500
2012-11-30	NA

Lastly, plot the median number of steps per day.

```
with(dayStats,
     barplot(space = 0, medianSteps,
             main  = "Median Number of Steps per Day",
             xlab  = "Day", ylab = "Median Steps",
             names.arg = (as.Date(date) - as.Date(totalDaySteps[1, 1]) + 1)))
```




```
dev.copy(png, "plots/03-medianSteps.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
```

```
##   2
```

Median is zero because of the large number of zero steps. `NA`s may also account for that.

Since the median is zero for all days, there's no point in presenting the dataframe.

What is the average daily activity pattern?

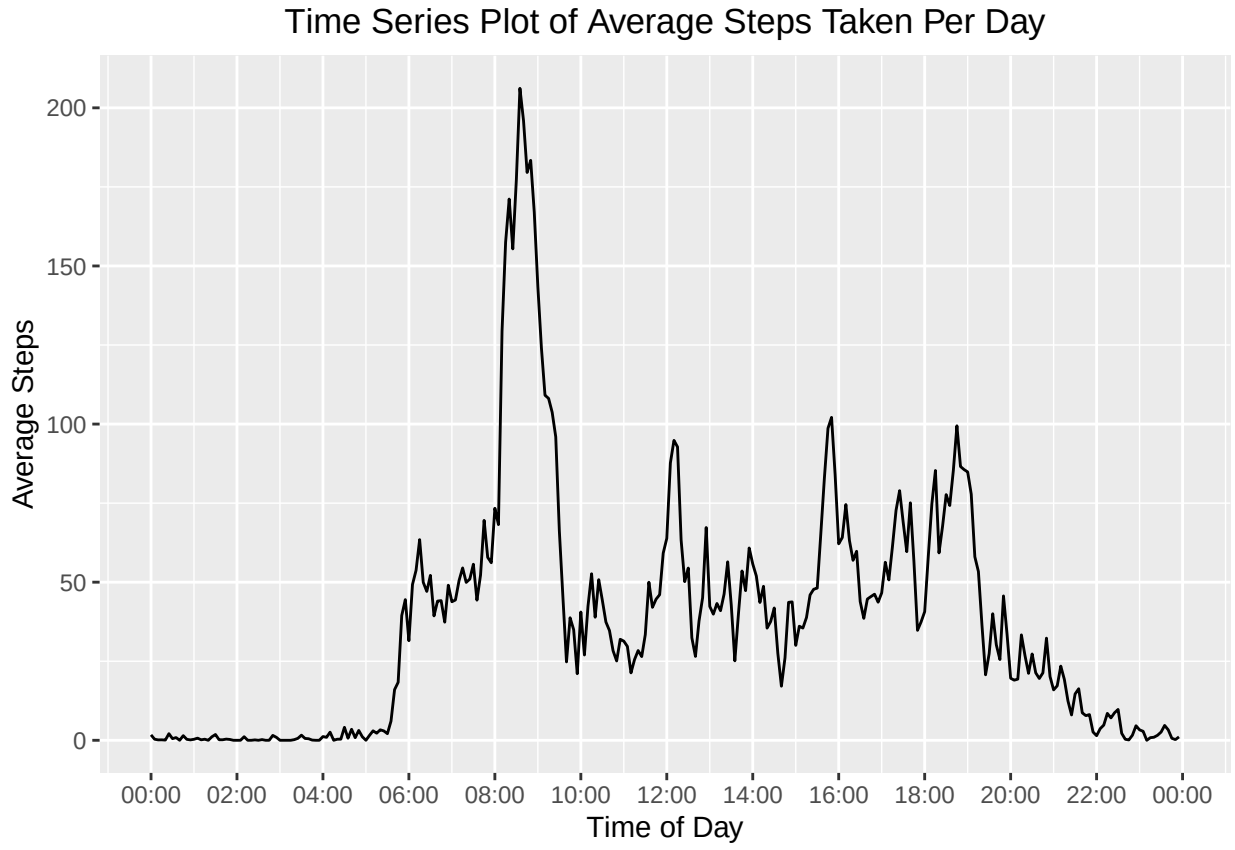
Create a dataframe with steps per interval, and plot the line.

Define a function that will take an integer – in this case, the interval – and convert it to a `time` object.

```
int2Time <- function(hhmm) {
     # separate the integer into its minute and hour components
     # assumes an integer of the form HHMM, where HH and MM are the hour
     # (between 0-23) and minute (between 0-59) components, respectively, of the
     # time.
     # Does not do parameter validation
     startHr  <- hhmm %% 100
     startMin <- hhmm %/% 100
     # and put them back together
     return(paste0(substring(paste0("0", startMin), nchar(startMin)),
                   ":",
                   substring(paste0("0", startHr), nchar(startHr))))
}
```

Average the number of steps over each five-minute interval and plot the time series.

```
intervalSteps <- aggregate(steps ~ interval, data = activity, mean)
intervalSteps$time <- as.POSIXct(int2Time(intervalSteps$interval), format = "%H:%M", tz = "MST")
ggplot(intervalSteps, aes(x = time, y = steps)) +
     geom_line() +
     scale_x_datetime(labels = date_format("%H:%M", tz = "MST"),
                      date_breaks = "2 hours") +
     labs(title = "Time Series Plot of Average Steps Taken Per Day",
          x = "Time of Day", y = "Average Steps") +
     theme(plot.title = element_text(hjust = 0.5))
```




```
dev.copy(png, "plots/04-intervalSteps.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

So, from the plot, on average there is negligible activity from midnight (see below) to around 5:30 in the morning, slowly climbing at around 7:00, and a spike of over 200 steps at around 8:30.

From then on, there is a minimum of 25 to a maximum of 100 steps, and activity dies down starting close to 11:30 PM.

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

The starting time and corresponding average number of steps are in `intervalSteps[which.max(intervalSteps$steps), 1]` and `intervalSteps[which.max(intervalSteps$steps), 2]` respectively. I thought I'd include the end time of that interval by adding five minutes.

```
intervalStart <- intervalSteps[which.max(intervalSteps$steps), 1]
intervalEnd   <- intervalSteps[which.max(intervalSteps$steps), 1] + 5
averageSteps  <- intervalSteps[which.max(intervalSteps$steps), 2]
```

Since the steps from interval 0 to interval 600 are mostly zero, I'm betting that the measurements start midnight. If they don't, I'm screwed. Or not, maybe just have to make adjustments in the conversions.

What I'm fairly sure of is that the intervals are of the form `hhmm`, where hh and mm are the hour and minute components of the interval. As evidence, `interval` on the 24th row is 155, and on the 25th is 200. Examine the 36th and 37th rows, and `interval` goes from 255 to 300.

Output will be in 24-hour format.

```
sprintf("Maximum number of steps on average taken from %s to %s, number of steps = %f",
        int2Time(intervalStart), int2Time(intervalEnd), averageSteps)
```

```
## [1] "Maximum number of steps on average taken from 08:35 to 08:40, number of steps = 206.169811"
```

Imputing missing values

Are there differences in activity patterns between weekdays and weekends?