Reproducible Research: Peer Assessment 1

Loading and preprocessing the data

1. Load `data.table` and `dplyr` packages for later data manipulation, and `ggplot2` plus `scales` for plotting.

```
if (!require(data.table)) {install.packages("data.table"); library(data.table)}
if (!require(dplyr)) {install.packages("dplyr"); library(dplyr)}
if (!require(ggplot2)) {install.packages("ggplot2"); library(ggplot2)}
if (!require(scales)) {install.packages("scales"); library(scales)}
```

2. Download data file and extract the **.zip** file if not already present in **data/** directory
3. Read **.csv** file using `data.table::fread` wrapping the output using the `%>%` operator which pipes the output first into a `data.table` object, then a `dplyr::tbl_dt` object.

```
if (!file.exists("data/activity.csv")) {
    download.file(url =
        "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip",
        method = "curl", destfile = "data/temp.zip", quiet = TRUE)
    unzip(zipfile = "data/temp.zip", exdir = "data")
    file.remove("temp.zip")
    }
activity <- fread("data//activity.csv") %>%
    data.table %>%
    tbl_dt
```

Next we alter the loaded table using `dplyr`'s mutate function

1. Convert date to integer-based `IDate` class from `data.table` package. Though not useful for this smaller data table, it can sort more quickly with very large data sets
2. Likewise, `ITime` from the `interval` column.

- `Interval <integer division>` 100 gives the hours
- `Interval <modulo>` 100 gives the minutes

3. The original interval column is dropped
4. `date, time` is created as a data.table key, thereby accelerating any future access of the data.

```
activity <- activity %>%
    mutate(date=as.IDate(date),
           time=as.ITime(
               sprintf("%02d:%02d",
                       interval %/% 100,
                       interval  %% 100))) %>%
    select(-interval)
setkey(x = activity, date, time)
```

Now we have

```r
tables()
```

```
##      NAME       NROW NCOL MB COLS            KEY
## [1,] activity 17,568    3  1 steps,date,time date,time
## Total: 1MB
```

Which looks like

```r
head(activity)
```

```
##   steps       date     time
## 1    NA 2012-10-01 00:00:00
## 2    NA 2012-10-01 00:05:00
## 3    NA 2012-10-01 00:10:00
## 4    NA 2012-10-01 00:15:00
## 5    NA 2012-10-01 00:20:00
## 6    NA 2012-10-01 00:25:00
```

What is mean total number of steps taken per day?

To summarize the data into total steps taken per day, we could drop all time intervals that include an NA

```
activity %>% filter(!is.na(steps)) %>% group_by(date) %>% summarise(sum(steps))
```

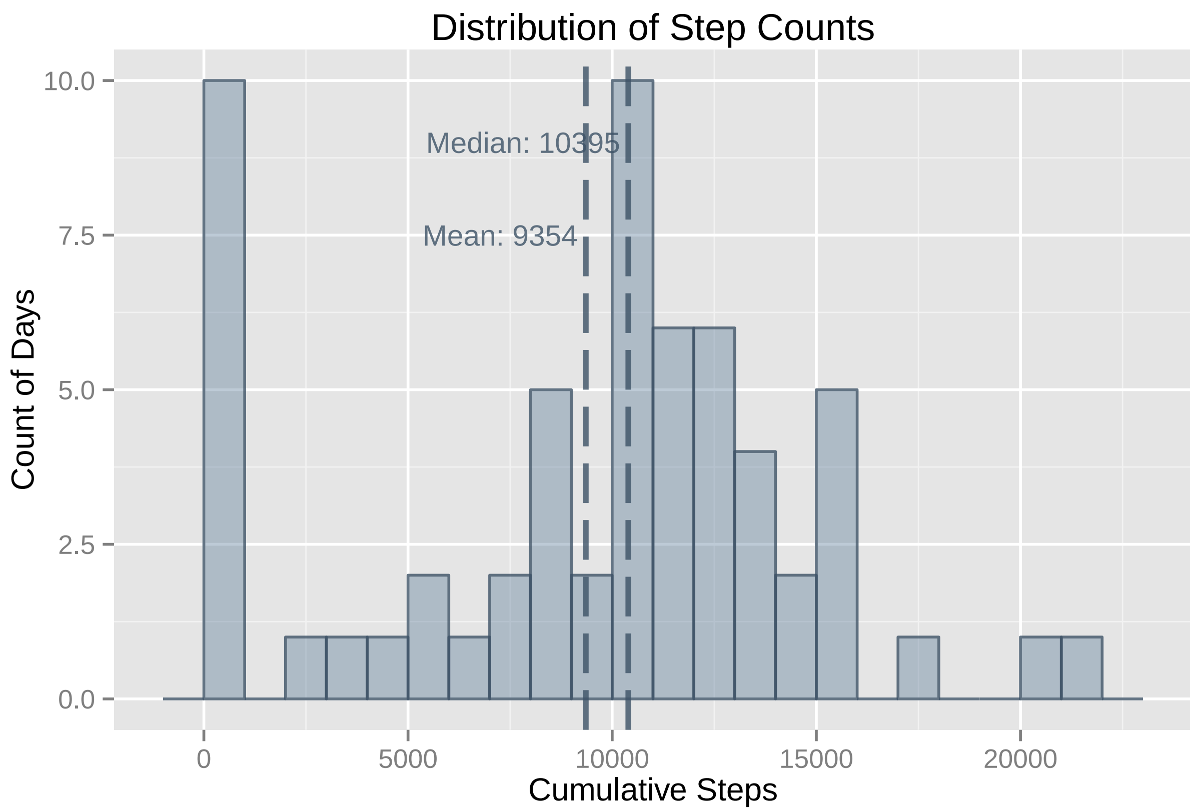
However, that completely drops some dates such as 2012-10-01 and 2012-11-01 which have no valid measurements. At this point in the assignment, rather than imputing missing values, we are *trying* to show the limitations of ignoring the NAs. One way to do this is to read the NAs as zeros which simply do not add to the sum of steps.

```r
activity <- activity %>%
    mutate(stepsZeroNAs = ifelse(test = is.na(steps), yes = 0, no = steps))
#stepsZeroNAs column = Steps with ZERO in place of NAs

stepCounts <- activity %>%
    group_by(date) %>%
    summarise(stepsZeroNAs=sum(stepsZeroNAs))
```

Now, lets build a histogram of the data with ggplot2

```r
Zdata <- ggplot(data = stepCounts)
tempHue <- 0.58
    ColA.light <- hsv(h = tempHue, s = 0.4, v = 0.6, alpha = 0.4)
    ColA.dark <- hsv(h = tempHue, s = 0.4, v = 0.4, alpha = 0.8)
Zaes <- aes(x = stepsZeroNAs)
Zgeom <- geom_histogram(fill=ColA.light, colour=ColA.dark, binwidth = 1000)
meanstepsZeroNAs <- mean(stepCounts$stepsZeroNAs)
medianstepsZeroNAs <- median(stepCounts$stepsZeroNAs)
ZmeanAnnotation <- annotate(
  "text", y = 7.5, x = meanstepsZeroNAs, align = "right", hjust=1, size=4,
  label=sprintf("Mean: %.0f ", meanstepsZeroNAs), color=ColA.dark)
```

```r
ZmedianAnnotation <- annotate(
  "text", y = 9, x = medianstepsZeroNAs, align = "right", hjust=1, size=4,
  label=sprintf("Median: %.0f ", medianstepsZeroNAs), color=ColA.dark, border=1)
Zmean <- geom_vline(
  xintercept=meanstepsZeroNAs, colour=ColA.dark, size=1.0, linetype="longdash")
Zmedian <- geom_vline(
  xintercept=medianstepsZeroNAs, colour=ColA.dark, size=1.0, linetype="longdash")
labels <- labs(
  title = "Distribution of Step Counts",
  x = "Cumulative Steps", y = "Count of Days")
Zdata + Zaes + Zgeom + labels +
    Zmean + ZmeanAnnotation +
    Zmedian + ZmedianAnnotation
```



9354 is the **mean** number of steps taken in a day, as calculated from data where NA is treated like no steps in that time period

10395 is the **median**, using the same data

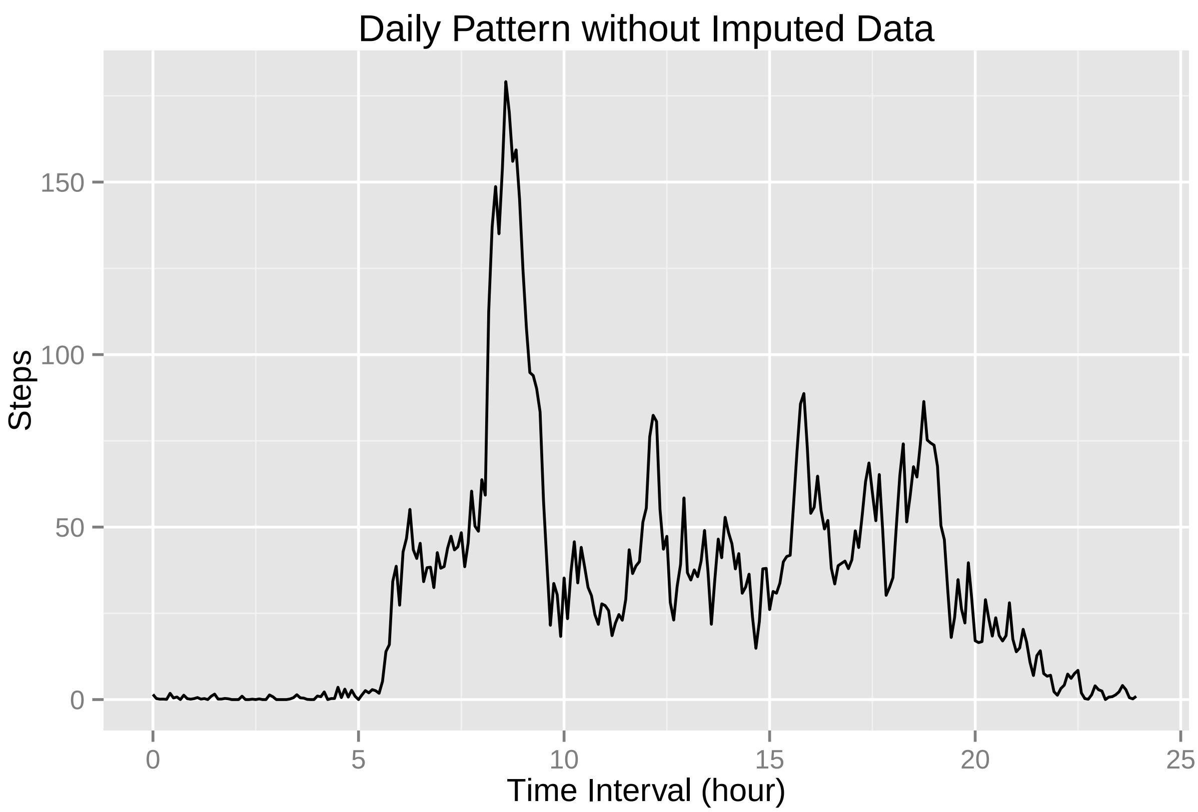
What is the average daily activity pattern?

Calculate typical daily pattern, with NAs ignored and with NAs treated as zero.

```r
dailyPattern <- activity %>%
    group_by(time) %>%
    summarise(stepsZeroNAs=mean(stepsZeroNAs),
              stepsTypical=mean(steps, na.rm = TRUE))
setkey(dailyPattern, time)
```

Then plot the data

```
qplot(data = dailyPattern, y=stepsZeroNAs, x=time/3600, geom="line",
      xlim = c(0,24), xlab="Time Interval (hour)", ylab="Steps",
      main="Daily Pattern without Imputed Data")
```



From this calculation, we can note that typically, the most active 5-minute interval during the day starts at:

```
dailyPattern[stepsZeroNAs==max(dailyPattern$stepsZeroNAs), time]
```

```
## [1] "08:35:00"
```

Imputing missing values

The original data set contained `2304` NAs, out of a total `17568` observations.

The typical number of steps at each time interval during the day was calculated in the previous section

```
dailyPattern <- activity %>%
    group_by(time) %>%
    summarise(stepsZeroNAs=mean(stepsZeroNAs),
              stepsTypical=mean(steps, na.rm = TRUE))
```

This table can be used to impute typical values for the NAs in the original data set. It covers all time intervals during the day covered by the earlier method of turing NAs into zeroes:

```
length(unique(dailyPattern$stepsTypical)) == length(unique(dailyPattern$stepsZeroNAs))
```

```
## [1] TRUE
```

And, it contains no NAs:

```
sum(is.na(length(unique(dailyPattern$stepsTypical))))
```

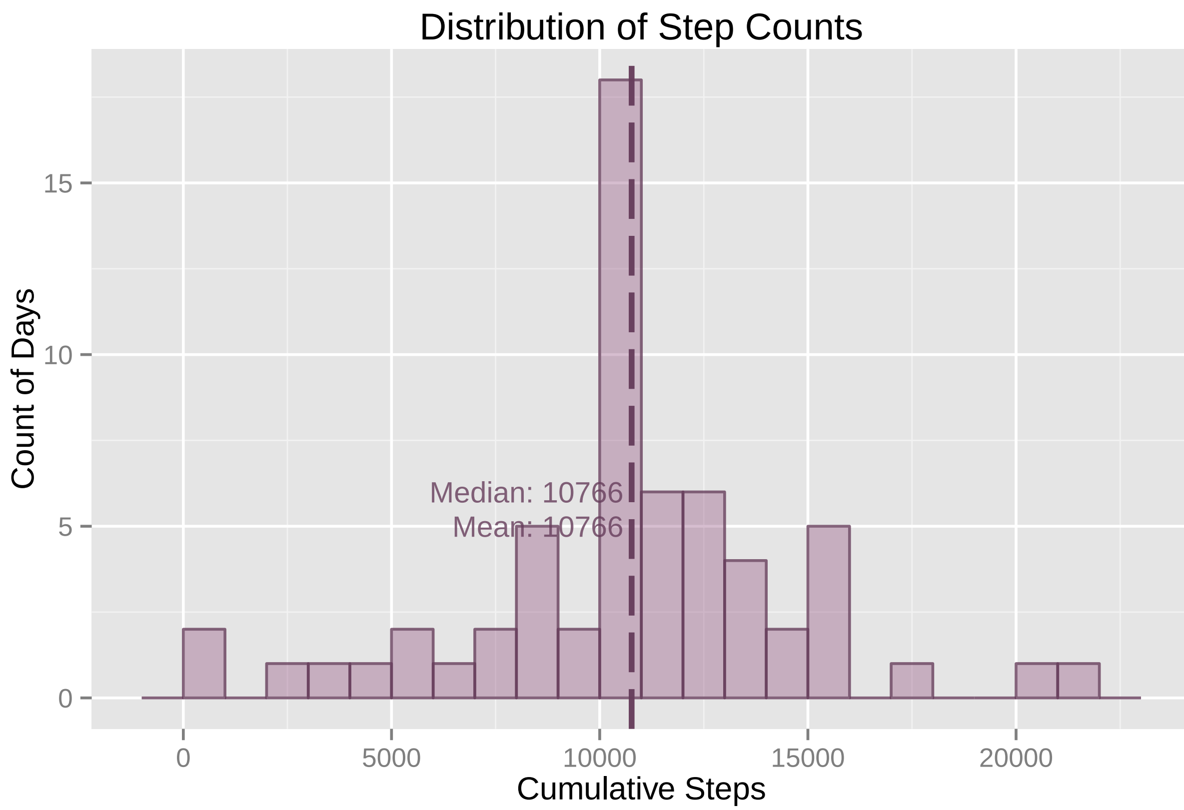
```
## [1] 0
```

This table can be used to create a new column in the original data set that imputes missing values from the typical daily pattern in cases there were NA values previously.

```
activity <- activity %>%
    mutate(stepsTypical = ifelse(
        test = is.na(steps),
        yes = dailyPattern[time==time, stepsTypical],
        no = steps))
```

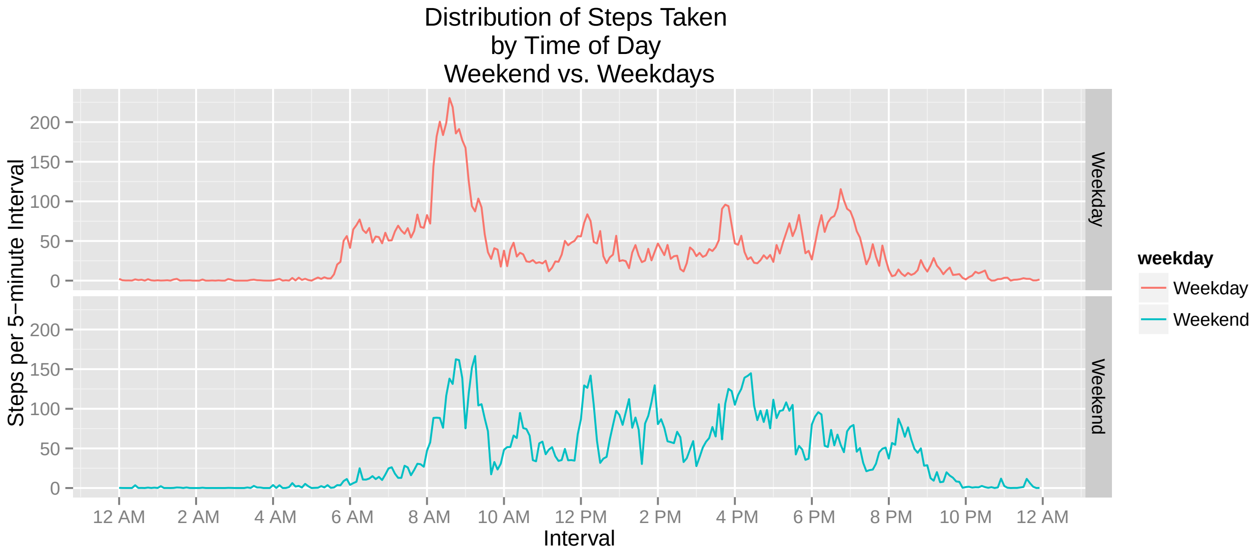
These new data can be plotted like before:

```
stepCounts <- activity %>%
    group_by(date) %>%
    summarise(stepsZeroNAs=sum(stepsZeroNAs),
              stepsImputed=sum(stepsTypical))

Imp.data <- ggplot(data = stepCounts)
tempHue <- 0.88
    ColB.light <- hsv(h = tempHue, s = 0.4, v = 0.6, alpha = 0.4)
    ColB.dark <- hsv(h = tempHue, s = 0.4, v = 0.4, alpha = 0.8)
Imp.aes <- aes(x = stepsImputed)
Imp.geom <- geom_histogram(fill=ColB.light, colour=ColB.dark, binwidth = 1000)
meansteps <- mean(stepCounts$stepsImputed)
mediansteps <- median(stepCounts$stepsImputed)
Imp.meanAnnotation <- annotate("text", y = 5, x = meansteps, align = "right",
                            label=sprintf("Mean: %.0f ", meansteps),
                            color=ColB.dark, hjust=1, size=4)
Imp.medianAnnotation <- annotate("text", y = 6, x = mediansteps, align = "right",
                            label=sprintf("Median: %.0f ", mediansteps),
                            color=ColB.dark, hjust=1, size=4, border=1)
Imp.mean <- geom_vline(xintercept=meansteps, colour=ColB.dark,
                       size=1.0, linetype="longdash")
Imp.median <- geom_vline(xintercept=mediansteps, colour=ColB.dark,
                         size=1.0, linetype="longdash")
labels <- labs(title = "Distribution of Step Counts",
               x = "Cumulative Steps", y = "Count of Days")
Imp.data + Imp.aes + Imp.geom + labels +
    Imp.mean + Imp.meanAnnotation +
    Imp.median + Imp.medianAnnotation
```



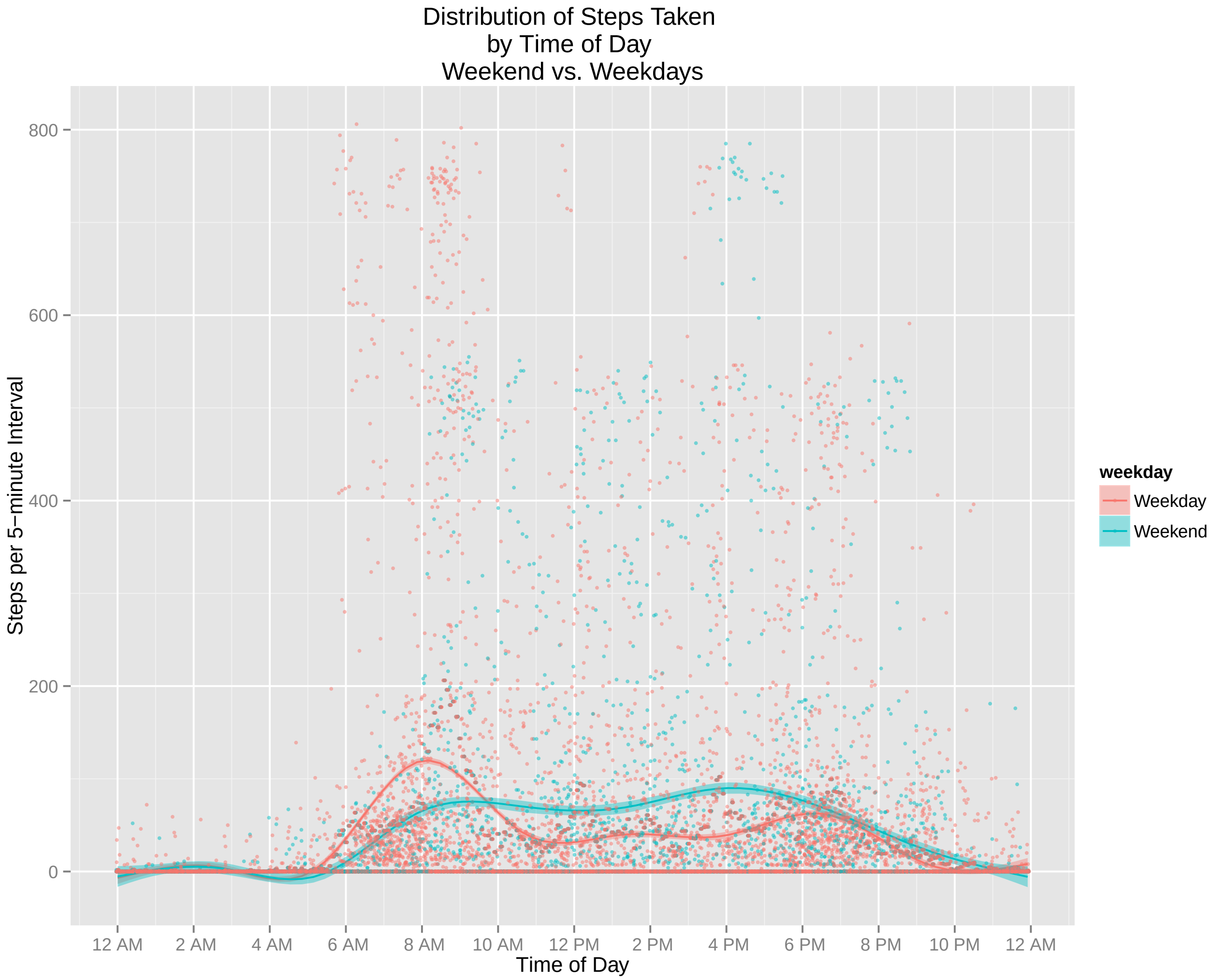
Are there differences in activity patterns between weekdays and weekends?

```
activity <- activity %>%
  mutate(weekday= ifelse(
    weekdays(date) %in% c("Saturday", "Sunday"),
    "Weekend",
    "Weekday"))

dailyPattern.split <- activity %>%
    group_by(time, weekday) %>%
    summarise(time, steps=mean(stepsTypical))

ggplot(data = dailyPattern.split, aes(x = as.POSIXct(time), y = steps, color = weekday)) +
  scale_x_datetime(breaks = date_breaks("2 hour"), labels = date_format("%l %p")) +
  # See documentation for `scales` package to understand `scale_x_datetime`,
  # `date_breaks` and `date_format` transformation of times along x-axis
  facet_grid(weekday ~ .) + geom_line() +
  labs(title = sprintf(
    "Distribution of Steps Taken \nby Time of Day \nWeekend vs. Weekdays"),
    x = "Interval", y = "Steps per 5-minute Interval")
```



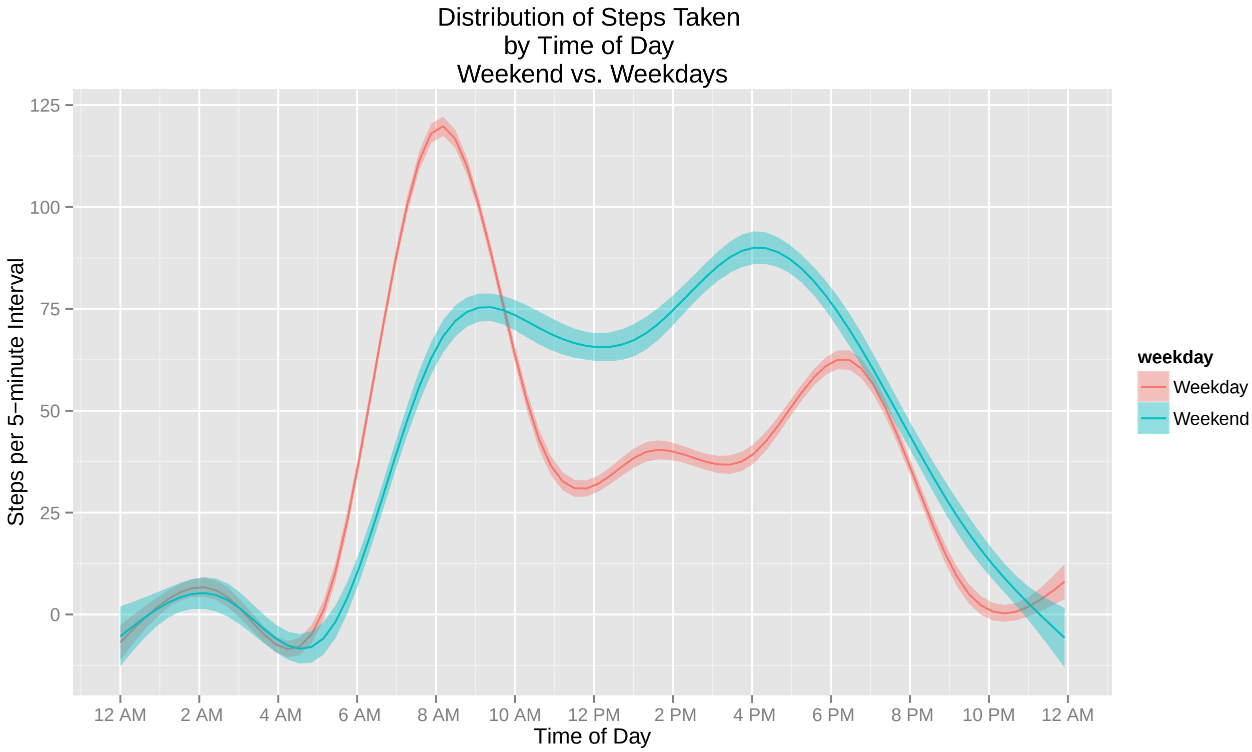
And, just for fun here is an additional plot of the data

```
ggplot(data = activity, aes(x = as.POSIXct(time), y = stepsTypical, colour = weekday, fill = weekday)) +
  scale_x_datetime(breaks = date_breaks("2 hour"), labels = date_format("%l %p")) +
  geom_smooth(level = 0.8) + geom_point(alpha=0.5, size=1, position = "jitter") +
  labs(title = sprintf(
    "Distribution of Steps Taken \nby Time of Day \nWeekend vs. Weekdays"),
    x = "Time of Day", y = "Steps per 5-minute Interval")
```



And, zooming in on the smoothed data

```
ggplot(data = activity,
       aes(x = as.POSIXct(time), y = stepsTypical, colour = weekday, fill = weekday)) +
  scale_x_datetime(breaks = date_breaks("2 hour"), labels = date_format("%l %p")) +
  geom_smooth(level = 0.6) +
  labs(title = sprintf("Distribution of Steps Taken \nby Time of Day \nWeekend vs. Weekdays"),
       x = "Time of Day", y = "Steps per 5-minute Interval")
```


Distribution of Steps Taken
by Time of Day
Weekend vs. Weekdays
Steps per 5−minute Interval
125
100
75
50
25
0
12 AM
2 AM
4 AM
6 AM
8 AM
10 AM
12 PM
2 PM
4 PM
6 PM
8 PM
10 PM
12 AM
Time of Day
weekday
Weekday
Weekend