ReproducibleResearch\_CourseProject1

## Loading and Preprocessing Data

Read CSV file in R Studio and stringsAsFactors is set to FALSE so that dates are not factors but strings that will be converted to dates later.

library(readr)  
activity <- read.csv("~/Desktop/Coursera/Reproducible Research/activity.csv", stringsAsFactors = F)

## What is the total number of steps taken per day

suppressMessages(library(dplyr))  
library("dplyr")

Next, since the data contains NA values,Remove NA values using complete.cases function

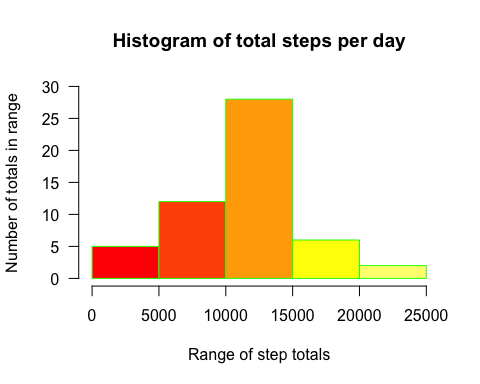
complete\_days\_only <- activity[complete.cases(activity),]

To perform the analysis required for this part of the assignment, use the group\_by and summarise functions from dplyr. The sum can be calculated using summarise once the data has been organized by date using group\_by.

step\_summary <- complete\_days\_only %>%  
 group\_by(date) %>%  
 summarise(daily\_step\_count = sum(steps))

Now plot the results for total number of steps taken per day. The histogram will show the range of totals across the x-axis (from 0 to the maximum number of steps recorded for a single day) while along the y-axis will be shown the number of days that fell within each range (i.e., between 0 and 5000, there were 5 days which had totals in that range).

hist(step\_summary$daily\_step\_count,   
 main = "Histogram of total steps per day",  
 xlab = "Range of step totals",  
 ylab = "Number of totals in range",  
 border = "green",  
 col = heat.colors(5),  
 las = 1,  
 ylim = c(0, 30))



mean(step\_summary$daily\_step\_count)

## [1] 10766.19

median(step\_summary$daily\_step\_count)

## [1] 10765

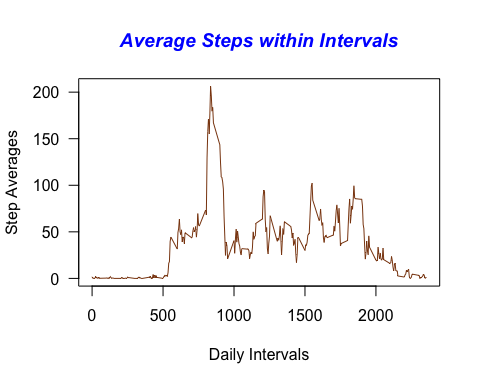
The mean and median can be easily calculated from the daily totals, and the mean of all the steps over the two-month period is 10766.19 while the median is 10765.

## What is the average daily activity pattern?

x <- complete\_days\_only %>%   
 group\_by(interval) %>%   
 summarise(avg\_interval = mean(steps))

Plot the results to see the distribution of the averages. The range of intervals will be plotted along the x-axis and the average number of steps for each interval will be plotted along the y-axis. A time series plot will be used since the data is collected over the course of each day in the month for a 2-month period.

plot(x$interval,   
 x$avg\_interval,   
 type = "l",   
 las = 1,   
 col = "chocolate4",   
 main = "Average Steps within Intervals",  
 col.main = "blue",  
 font.main = 4,  
 xlab = "Daily Intervals",  
 ylab = "Step Averages"  
 )



x[which.max(x$avg\_interval), ]

## # A tibble: 1 x 2  
## interval avg\_interval  
## <int> <dbl>  
## 1 835 206.

The highest average number of steps was found in interval 835 and had the value 206.1698.

## Imputing missing values

The original dataset contained a certain amount of NA values, which were removed prior to doing the previous analyses. Just how many values in the original dataset were NA is easily calculated using the nrow function against both the original and reduced datasets.

nrow(activity)

## [1] 17568

nrow(complete\_days\_only)

## [1] 15264

And so the number of missing observations is the difference between the two, or 2304.

The presence of missing data for certain days in this time sequence may introduce bias into some calculations or summaries of the data, so it is useful to impute values for those observations in this category. There are packages in R which handle imputations but using a simple random number generator in this case should suffice.

The following code will generate a series of integral values that spans the range of observations in the original dataset, from the minimum to the maximum value. The entries in this vector will be used as replacement values for the existing NA entries in the original dataset. Using set.seed will result in the same set of integers being produced for subsequent runs and support the concept of reproducible research. The max value is divided by 10 provide reasonable scale to these values, which can get quite large and skew the results in a different way than the missing values can.

set.seed(1234)  
z <- floor(runif(nrow(activity),   
 min = min(activity$steps, na.rm = T),   
 max = max(activity$steps, na.rm = T)/10))

Next the indices of the missing values are determined. The use of a separate vector to hold these results rather than include them inline in the mutation code is done for readability.

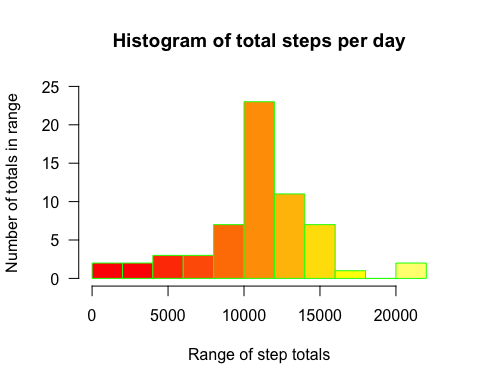
w <- which(is.na(activity$steps))  
activity$steps[w] <- z[w]

How does imputing values for missing data affect the results obtained earlier with the reduced dataset? The same calculations for total steps per day can be calculated against the augmented dataset and the results plotted via a histogram.

complete\_data <- activity %>%   
 group\_by(date) %>%   
 summarise(daily\_step\_count = sum(steps))

And here is the plot:

hist(complete\_data$daily\_step\_count,   
 breaks = 10,  
 main = "Histogram of total steps per day",  
 xlab = "Range of step totals",  
 ylab = "Number of totals in range",  
 border = "green",  
 col = heat.colors(12),  
 las = 1,  
 ylim = c(0, 25))



Once again, the mean and median can be easily calculated from the daily totals, and the mean of all the steps over the two-month period is 10858.38 while the median is 11196. The imputed data had a negligible affect on both the mean and the median calculated values. The data did, however, produce a gap in recorded values for the range between 18000 and 20000.

## Are there differences in activity patterns between weekdays and weekends?

Another reasonable question to ask is whether or not the activity level changes based on the day being a weekday or a weekend. To answer this question, the date variable has its values converted to dates.

activity$date <- as.POSIXct(activity$date)

Next, in order to separate the data into weekday and weekend subsets for analysis, the date variable is input to the weekdays method and that output is checked against a vector of weekend day names. The appropriate text is added as a new column to the dataframe. This column is then converted to a factor using the as.factor method.

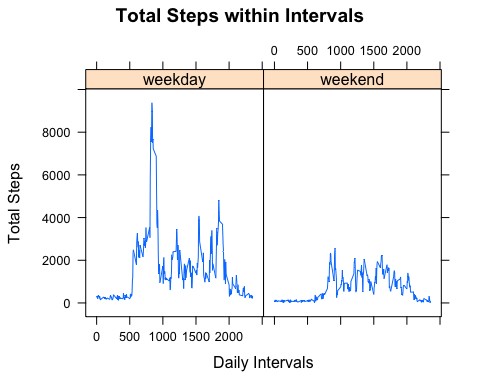
activity$dayType <- ifelse(weekdays(activity$date) %in% c("Saturday", "Sunday"),"weekend","weekday")  
activity$dayType <- as.factor(activity$dayType)

To prepare the data for plotting, the dataset with imputed values is organized first by dayType (weekday, weekend) and then by interval. A sum of the steps recorded within those time intervals for each dayType is then computed.

q <- activity %>%   
 group\_by(dayType, interval) %>%   
 summarise(daily\_step\_count = sum(steps))

Now that the data is ready for plotting, create a time series plot using the lattice package so that the independent/dependent variables can be conditioned by the factor variable dayType.

library(lattice)  
with(q,   
 xyplot(daily\_step\_count ~ interval | dayType,   
 type = "l",   
 main = "Total Steps within Intervals",  
 xlab = "Daily Intervals",  
 ylab = "Total Steps"))



It would appear from the output of the graph that this person spent a great deal of time relaxing on the weekends after walking so much during the week!