Repro_activity

ting

2025-06-09

```
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this: reading the CSV file, name as Df1 install package

1. What is mean total number of steps taken per day?

```
library(tidyr)
library(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
```

```
library(ggplot2)

    setwd("C:/Users/xiaot/datasciencecoursera/Reproduce analysis/repdata_data_activity")
    Df1<- read.csv("activity.csv",header = TRUE )# Equivalent explicit version
    head(Df1)
```

```
##   steps       date interval
## 1    NA 2012-10-01        0
## 2    NA 2012-10-01        5
## 3    NA 2012-10-01       10
## 4    NA 2012-10-01       15
## 5    NA 2012-10-01       20
## 6    NA 2012-10-01       25
```

```
# check the missng percent
    missing_percent <- colMeans(is.na(Df1)) * 100
    missing_percent
```

```
##    steps     date interval 
## 13.11475  0.00000  0.00000
```

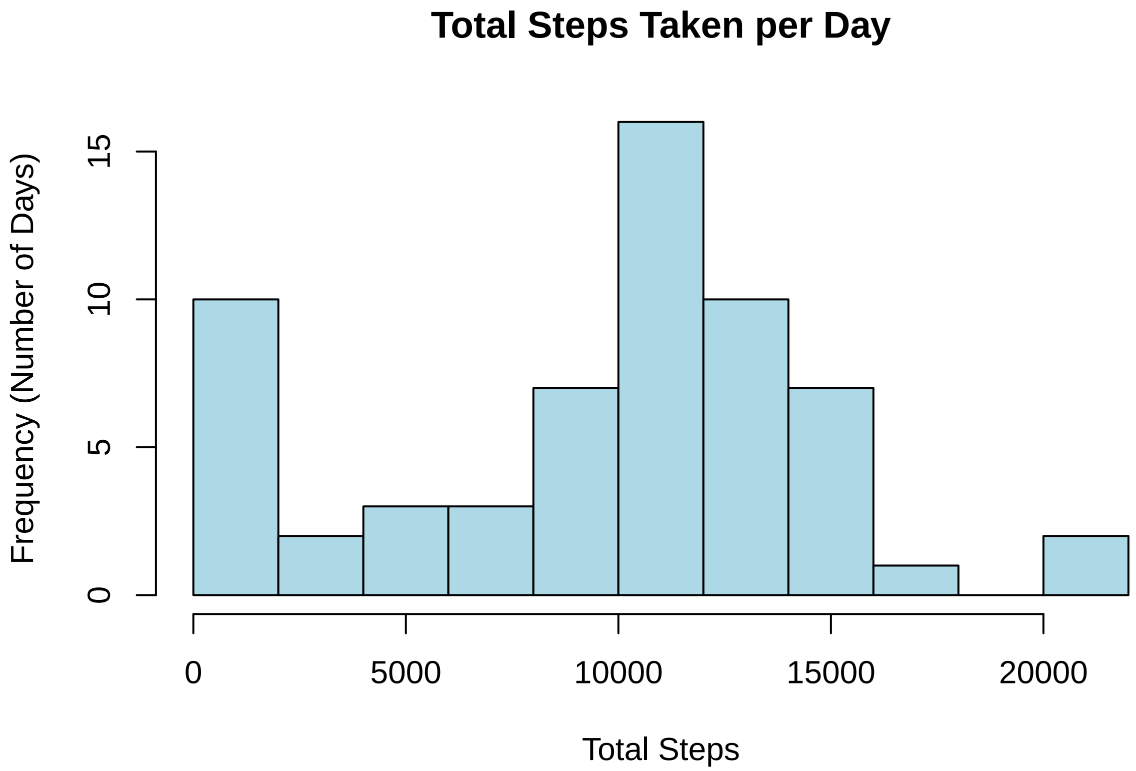
```
    steps_per_day <- Df1 %>%
  group_by(date) %>%
  summarise(total_steps = sum(steps, na.rm = TRUE))

# View the result
    steps_per_day
```

```
## # A tibble: 61 x 2
##    date       total_steps
##    <chr>            <int>
##  1 2012-10-01           0
##  2 2012-10-02         126
##  3 2012-10-03       11352
##  4 2012-10-04       12116
##  5 2012-10-05       13294
##  6 2012-10-06       15420
##  7 2012-10-07       11015
##  8 2012-10-08           0
##  9 2012-10-09       12811
## 10 2012-10-10        9900
## # i 51 more rows
```

```
# Summary (min, median, mean, max)
    summary(steps_per_day$total_steps)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       0    6778   10395    9354   12811   21194
```

```
##Make a histogram of the total number of steps taken each day
    ## Create histogram
   Step_per_day_p<-  hist(
  steps_per_day$total_steps,
  main = "Total Steps Taken per Day",
  xlab = "Total Steps",
  ylab = "Frequency (Number of Days)",
  col = "lightblue",
  breaks = 10   # Adjust number of bins
    )
```



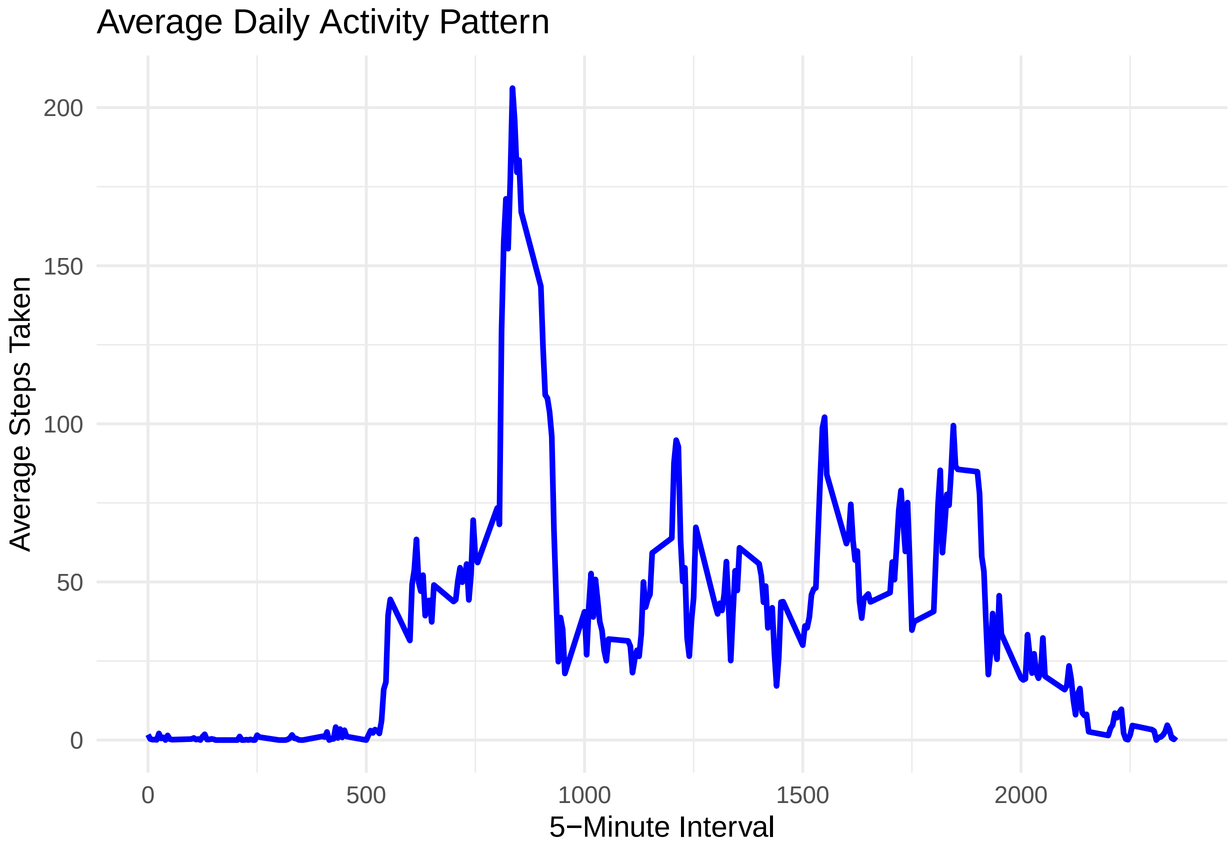
```
print(Step_per_day_p)
```

```
## $breaks
##  [1]     0  2000  4000  6000  8000 10000 12000 14000 16000 18000 20000 22000
##
## $counts
##  [1] 10  2  3  3  7 16 10  7  1  0  2
##
## $density
##  [1] 8.196721e-05 1.639344e-05 2.459016e-05 2.459016e-05 5.737705e-05
##  [6] 1.311475e-04 8.196721e-05 5.737705e-05 8.196721e-06 0.000000e+00
## [11] 1.639344e-05
##
## $mids
##  [1]  1000  3000  5000  7000  9000 11000 13000 15000 17000 19000 21000
##
## $xname
## [1] "steps_per_day$total_steps"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

2. What is the average daily activity pattern?

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

```
## Step 1: Calculate Average Steps per Interval
    avg_steps_per_interval <- Df1 %>%
    group_by(interval) %>%
    summarise(avg_steps = mean(steps, na.rm = TRUE))  # Handle missing values
    ggplot(avg_steps_per_interval, aes(x = interval, y = avg_steps)) +
  geom_line(color = "blue", linewidth = 1) +
  labs(
    title = "Average Daily Activity Pattern",
    x = "5-Minute Interval",
    y = "Average Steps Taken"
  ) +
  theme_minimal()
```



3. Imputing missing values Calculate and report the total number of missing values in the data set

```
    # check the missng percent
    missing_percent <- colMeans(is.na(Df1)) * 100
    missing_percent
```

```
##    steps     date interval 
## 13.11475  0.00000  0.00000
```

```r
# Check if any days have all NAs (resulting in NaN means)
    daily_means <- Df1 %>%
    group_by(date) %>%
    summarise(daily_mean = mean(steps, na.rm = TRUE))
    print(daily_means)
```

```
## # A tibble: 61 x 2
##    date       daily_mean
##    <chr>           <dbl>
##  1 2012-10-01    NaN
##  2 2012-10-02      0.438
##  3 2012-10-03     39.4
##  4 2012-10-04     42.1
##  5 2012-10-05     46.2
##  6 2012-10-06     53.5
##  7 2012-10-07     38.2
##  8 2012-10-08    NaN
##  9 2012-10-09     44.5
## 10 2012-10-10     34.4
## # i 51 more rows
```

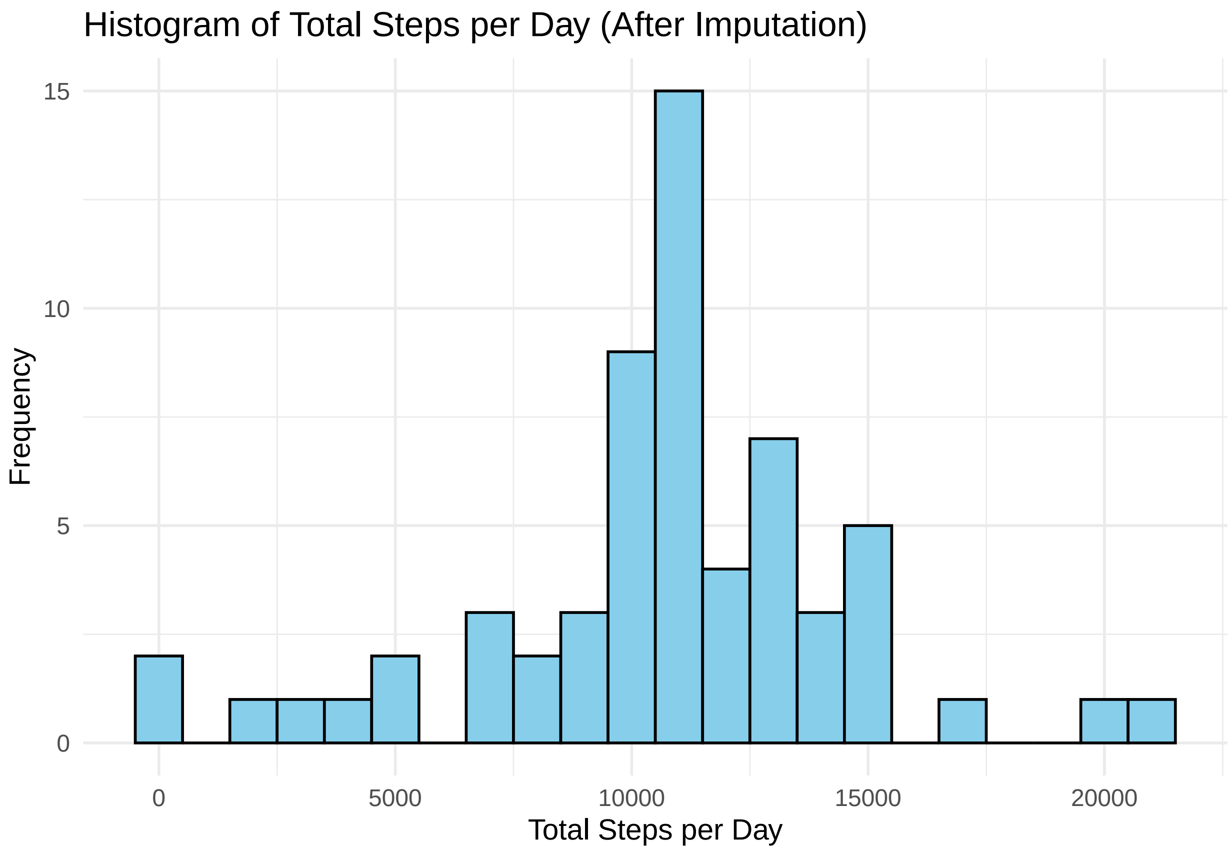
```r
    daily_means %>% filter(is.nan(daily_mean))  # Problematic dates
```

```
## # A tibble: 8 x 2
##   date       daily_mean
##   <chr>           <dbl>
## 1 2012-10-01        NaN
## 2 2012-10-08        NaN
## 3 2012-11-01        NaN
## 4 2012-11-04        NaN
## 5 2012-11-09        NaN
## 6 2012-11-10        NaN
## 7 2012-11-14        NaN
## 8 2012-11-30        NaN
```

```r
## Add a global mean since some days dairy mean are missing
    global_mean <- mean(Df1$steps, na.rm = TRUE)  # Fallback if daily mean is NaN

    Df1_imputed <- Df1 %>%
    left_join(daily_means, by = "date") %>%
    mutate(
    steps = coalesce(steps, daily_mean, global_mean)  # Replaces NA → daily mean → global mean
  ) %>%
  select(-daily_mean)  # Clean up
##Verify All NAs Are Replaced
    sum(is.na(Df1_imputed$steps))  # Should be 0
```

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

```r
## Calculate Total Daily Steps (After Imputation)
    daily_totals <- Df1_imputed %>%
    group_by(date) %>%
    summarise(total_steps = sum(steps))
##Create Histogram
    ggplot(daily_totals, aes(x = total_steps)) +
    geom_histogram(binwidth = 1000, fill = "skyblue", color = "black") +
    labs(title = "Histogram of Total Steps per Day (After Imputation)",
       x = "Total Steps per Day",
       y = "Frequency") +
    theme_minimal()
```



```
## Calculate Mean and Median
    mean_median <- daily_totals %>%
    summarise(
    mean_steps = mean(total_steps),
    median_steps = median(total_steps)
  )

print(mean_median)
```

```
## # A tibble: 1 x 2
##   mean_steps median_steps
##        <dbl>        <dbl>
## 1      10766.       10766.
```