Reproducible Research- Week 1 Course Project 1

Neelam

29/05/2024

Reproducible Research

Week 1- Project Assignment

This assignment will be described in multiple parts. You will need to write a report that answers the questions detailed below. Ultimately, you will need to complete the entire assignment in a single R markdown document that can be processed by knitr and be transformed into an HTML file.

Throughout your report make sure you always include the code that you used to generate the output you present. When writing code chunks in the R markdown document, always use echo = TRUE echo = TRUE so that someone else will be able to read the code. This assignment will be evaluated via peer assessment so it is essential that your peer evaluators be able to review the code for your analysis.

For the plotting aspects of this assignment, feel free to use any plotting system in R (i.e., base, lattice, ggplot2)

Fork/clone the GitHub repository created for this assignment . You will submit this assignment by pushing your completed files into your forked repository on GitHub. The assignment submission will consist of the URL to your GitHub repository and the SHA-1 commit ID for your repository state.

NOTE: The GitHub repository also contains the dataset for the assignment so you do not have to download the data separately.

Task 1

Loading and preprocessing the data

1. Show any code that is needed to Load the data (i.e. read.csv())

- Load the appropriate packages (install these if you don't already have them) and set your current working directory to where the dataset is

```r
library(rmarkdown)
library(tidyr)
library(dplyr)
library(ggplot2)
library(readr)
setwd("~/Reproducible Research/week 1/RepData_PeerAssessment1/Assignment submission_NI")
```

2. After downloading the data from https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip read it into R using the following code

```r
amd <- read_csv("activity.csv")
```

2. Process/transform the data (if necessary) into a format suitable for your analysis.

```r
str(amd)
```

```
## tibble [17,568 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ steps   : num [1:17568] NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date[1:17568], format: "2012-10-01" "2012-10-01" ...
##  $ interval: num [1:17568] 0 5 10 15 20 25 30 35 40 45 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   steps = col_double(),
##   ..   date = col_date(format = ""),
##   ..   interval = col_double()
##   .. )
```

```
head(amd)
```

```
## # A tibble: 6 x 3
##   steps date       interval
##   <dbl> <date>        <dbl>
## 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
```

```
#after seeing that the variable date is not in the appropriate format
amd <- read_csv("activity.csv", col_types = cols(
  date = col_date(format = "%Y-%m-%d") #  Specify the date formats
  ))

# Coding NaN variables to NA, so all missing values are "NA"

for (col in names(amd)) {
  amd[[col]] <- ifelse(is.nan(amd[[col]]), NA, amd[[col]])
}

print(amd)
```

```
## # A tibble: 17,568 x 3
##    steps  date interval
##    <dbl> <dbl>    <dbl>
##  1    NA 15614        0
##  2    NA 15614        5
##  3    NA 15614       10
##  4    NA 15614       15
##  5    NA 15614       20
##  6    NA 15614       25
##  7    NA 15614       30
##  8    NA 15614       35
##  9    NA 15614       40
## 10    NA 15614       45
## # ... with 17,558 more rows
```

Task 2

What is mean total number of steps taken per day? For this part of the assignment, you can ignore the missing values in the dataset.

The total mean steps per day are 9,354. Calculated from code below.

```
# Summarize steps by date
daily_steps <- amd %>%
  group_by(date) %>%
  summarize(total_steps = sum(steps, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# Print the summarized data
print(daily_steps)
```

```
## # A tibble: 61 x 2
##     date total_steps
##    <dbl>       <dbl>
##  1 15614           0
##  2 15615         126
##  3 15616       11352
##  4 15617       12116
##  5 15618       13294
##  6 15619       15420
##  7 15620       11015
##  8 15621           0
##  9 15622       12811
## 10 15623        9900
## # ... with 51 more rows
```

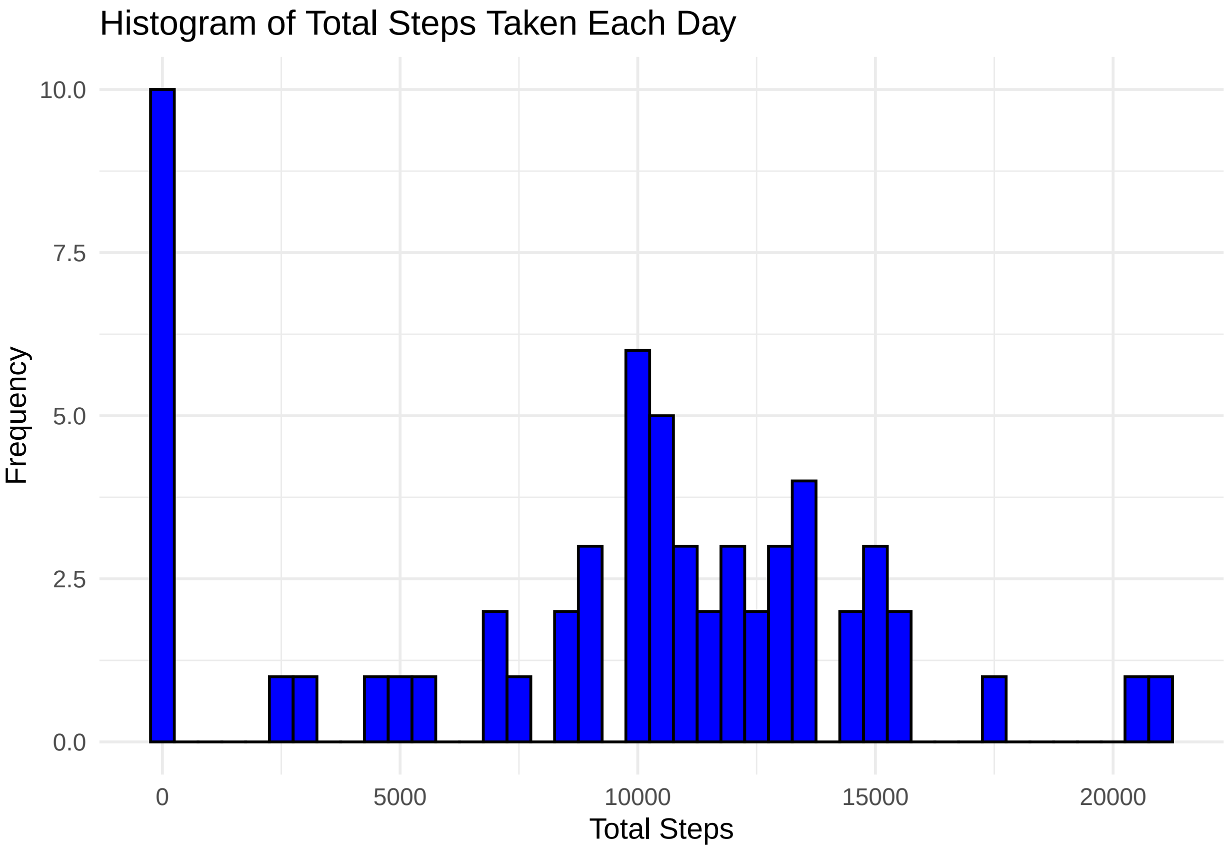
```
# Calculate the mean number of steps per day
totmean_steps_per_day <- mean(daily_steps$total_steps, na.rm = TRUE)

# Print the mean steps per day
print(totmean_steps_per_day)
```

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

Make a histogram of the total number of steps taken each day

```
ggplot(daily_steps, aes(x = total_steps)) +
  geom_histogram(binwidth = 500, fill = "blue", color = "black") +
  labs(title = "Histogram of Total Steps Taken Each Day",
       x = "Total Steps",
       y = "Frequency") +
  theme_minimal()
```



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

Mean total steps per day = 9354 Median total steps per day = 10395

```
totmean_steps_per_day <- mean(daily_steps$total_steps, na.rm = TRUE)
totmedian_steps_per_day <- median(daily_steps$total_steps, na.rm = TRUE)
```

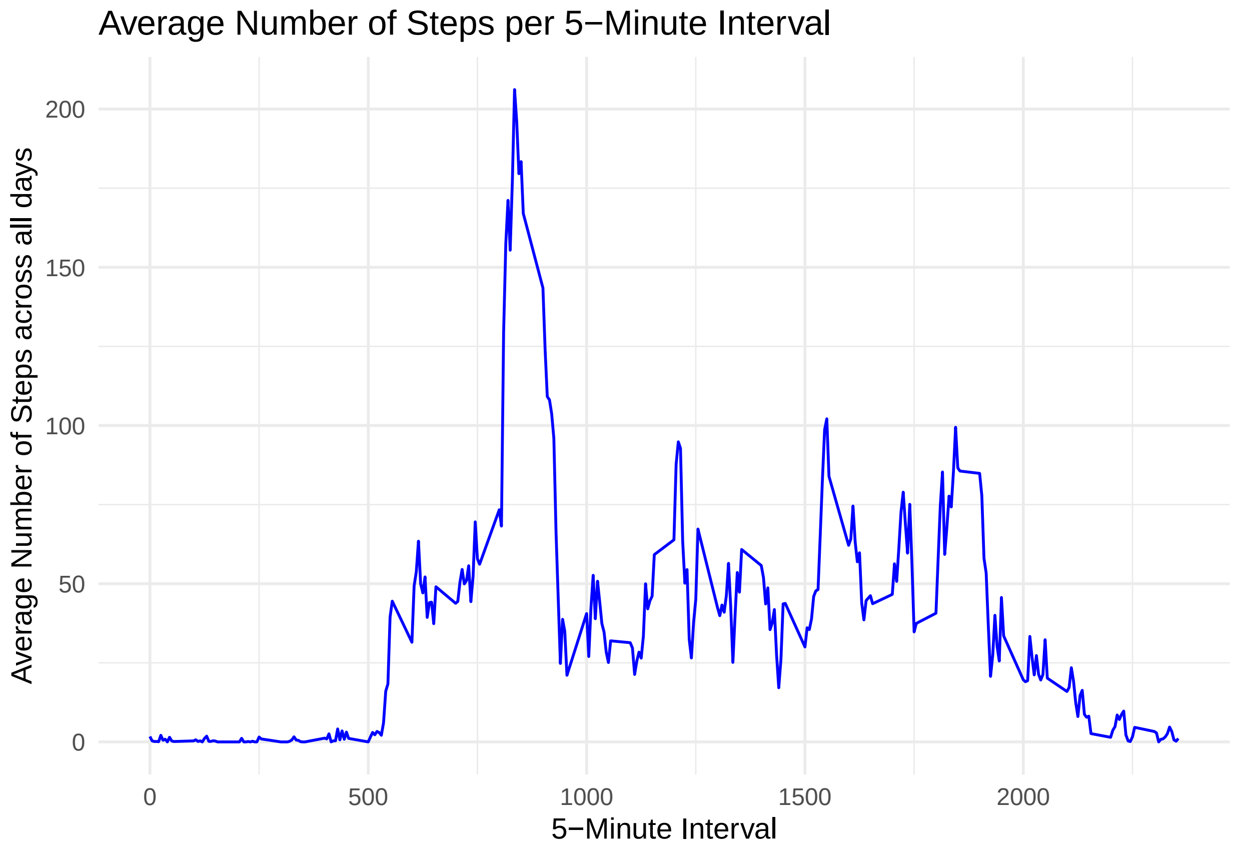
Task 3

What is the average daily activity pattern?

1. 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)

```
# Calculate the average steps per interval
average_steps_per_interval <- amd %>%
  group_by(interval) %>%
  summarize(avg_steps = mean(steps, na.rm = TRUE))

# Create the time series plot
ggplot(average_steps_per_interval, aes(x = interval, y = avg_steps)) +
  geom_line(color = "blue") +
  labs(title = "Average Number of Steps per 5-Minute Interval",
       x = "5-Minute Interval",
       y = "Average Number of Steps across all days") +
  theme_minimal()
```



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

The **835th** 5-minute interval contains the maximum number of steps (206)

```
# Find the interval with the maximum average steps
max_interval <- average_steps_per_interval %>%
  filter(avg_steps == max(avg_steps))

print(max_interval)
```

```
## # A tibble: 1 x 2
##   interval avg_steps
##      <dbl>     <dbl>
## 1      835      206.
```

Task 4

Imputing missing values

Note that 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.

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

2304 total missing values in the dataset

```
# Calculate the total number of rows with any missing values
totNA <- sum(rowSums(is.na(amd)) > 0)
print(totNA)
```

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

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

```
# Step 1: Calculate the mean steps for each 5-minute interval as the first date has no steps recorded t
mean_steps_per_interval <- amd %>%
  group_by(interval) %>%
  summarize(mean_steps = mean(steps, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
# Step 2: Replace missing values in the steps column with the mean steps for the respective interval
filled_data <- amd %>%
  left_join(mean_steps_per_interval, by = "interval") %>%
  mutate(steps = ifelse(is.na(steps), mean_steps, steps)) %>%
  select(-mean_steps)

# Check the resulting data frame
head(filled_data)
```

```
## # A tibble: 6 x 3
##    steps  date interval
##    <dbl> <dbl>    <dbl>
## 1 1.72   15614        0
## 2 0.340  15614        5
## 3 0.132  15614       10
## 4 0.151  15614       15
## 5 0.0755 15614       20
## 6 2.09   15614       25
```

```
print(filled_data)
```

```
## # A tibble: 17,568 x 3
##     steps  date interval
##     <dbl> <dbl>    <dbl>
##  1 1.72   15614        0
##  2 0.340  15614        5
##  3 0.132  15614       10
##  4 0.151  15614       15
##  5 0.0755 15614       20
##  6 2.09   15614       25
##  7 0.528  15614       30
##  8 0.868  15614       35
##  9 0      15614       40
## 10 1.47   15614       45
## # ... with 17,558 more rows
```

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?

Imputing missing data can potentially affect the estimates of the mean and median. By imputing missing data with the mean or median of the respective interval, you are essentially changing the distribution of the data. The impact on the estimates will depend on the amount and distribution of missing data, as well as the method used for imputation.

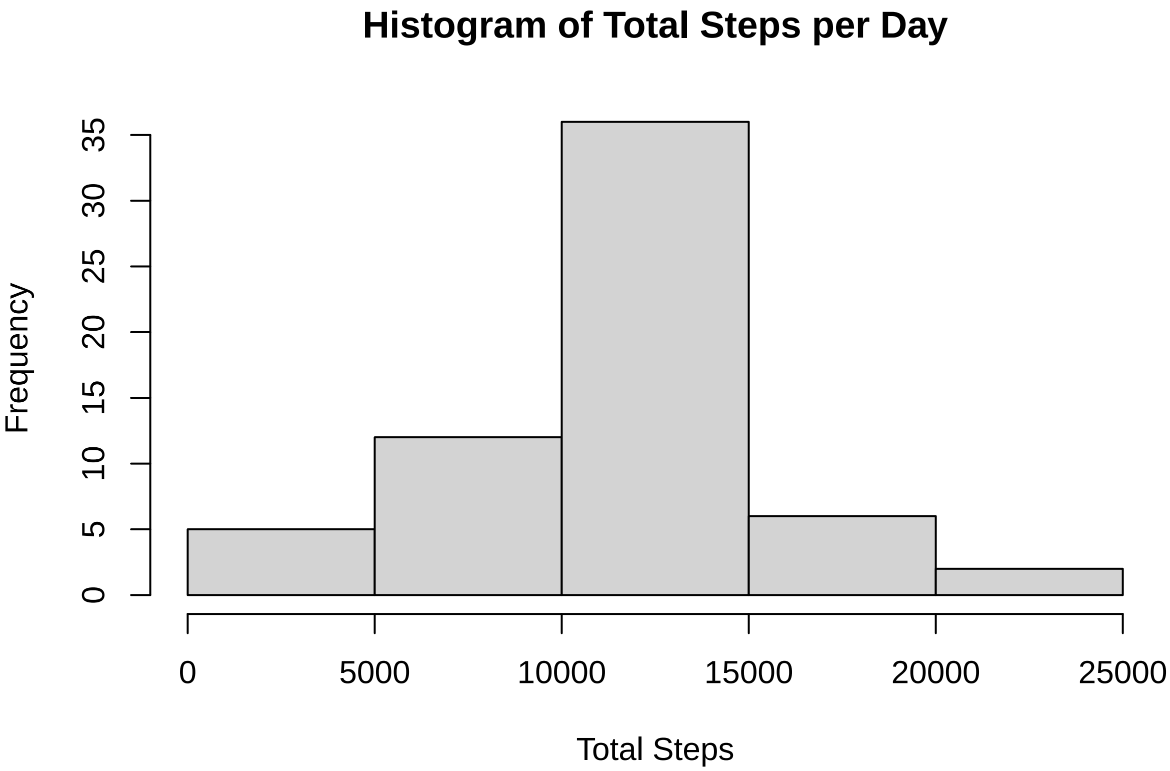
If the missing data is not imputed, days with missing data will have fewer steps counted, which might result in underestimation of the mean and median. Imputing missing data with the mean or median of the respective interval might lead to overestimation or underestimation depending on how the missing data is distributed across intervals and days.

It's important to consider the potential biases introduced by imputation and to interpret the estimates accordingly. Additionally, sensitivity analysis or comparison with other methods of imputation can help assess the robustness of the estimates.

```r
# Step 1: Calculate the total number of steps taken each day
steps_per_day <- filled_data %>%
  group_by(date) %>%
  summarize(total_steps = sum(steps, na.rm = TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# Step 2: Create a histogram of the total number of steps taken each day
hist(steps_per_day$total_steps, main = "Histogram of Total Steps per Day", xlab = "Total Steps", ylab =
```



```r
# Step 3: Calculate the mean and median of the total number of steps taken per day
mean_steps_per_day <- mean(steps_per_day$total_steps, na.rm = TRUE)
median_steps_per_day <- median(steps_per_day$total_steps, na.rm = TRUE)

# Print the mean and median
print(paste("Mean total number of steps per day:", mean_steps_per_day))
```

```
## [1] "Mean total number of steps per day: 10766.1886792453"
```

```
print(paste("Median total number of steps per day:", median_steps_per_day))
```

```
## [1] "Median total number of steps per day: 10766.1886792453"
```

Task 5

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.*

1. Create a new factor variable in the dataset with two levels – "weekday" and "weekend" indicating whether a given date is a weekday or weekend day.

```
# Convert the numeric date column to Date type
filled_data$date <- as.Date(as.character(filled_data$date), format = "%Y%m%d")

# Determine the day of the week for each date
filled_data$day_of_week <- weekdays(filled_data$date)

# Create a new factor variable indicating weekday or weekend day
filled_data$day_type <- ifelse(filled_data$day_of_week %in% c("Saturday", "Sunday"), "weekend", "weekda
filled_data$day_type <- factor(filled_data$day_type, levels = c("weekday", "weekend"))

# Check the resulting data frame
head(filled_data)
```

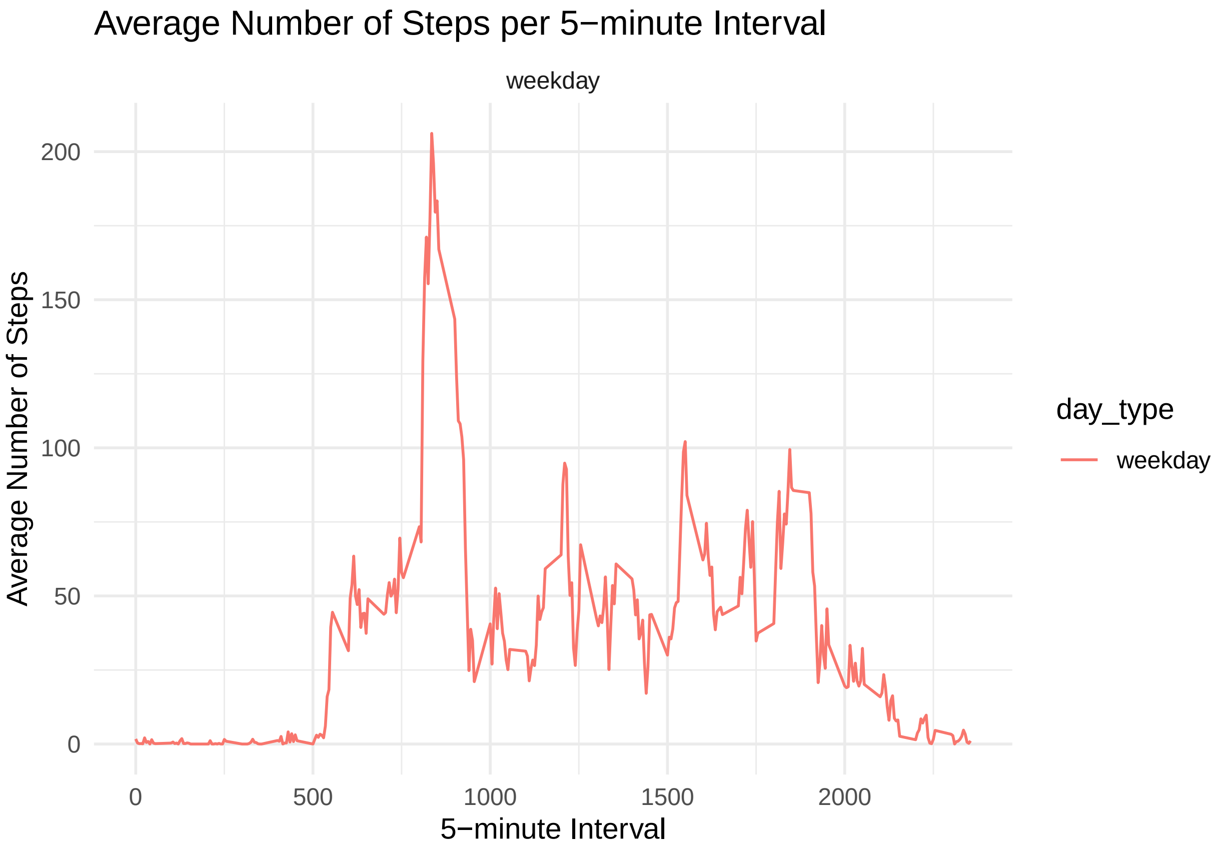
```
## # A tibble: 6 x 5
##    steps date       interval day_of_week day_type
##    <dbl> <date>        <dbl> <chr>       <fct>
## 1 1.72   NA                0 <NA>        weekday
## 2 0.340  NA                5 <NA>        weekday
## 3 0.132  NA               10 <NA>        weekday
## 4 0.151  NA               15 <NA>        weekday
## 5 0.0755 NA               20 <NA>        weekday
## 6 2.09   NA               25 <NA>        weekday
```

2. 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). *See the README file in the GitHub repository to see an example of what this plot should look like using simulated data.*

```
# Calculate the average number of steps taken for each 5-minute interval, separately for weekdays and w
avg_steps_per_interval <- filled_data %>%
  group_by(interval, day_type) %>%
  summarize(avg_steps = mean(steps, na.rm = TRUE))
```

```
## `summarise()` regrouping output by 'interval' (override with `.groups` argument)
```

```
# Create a panel plot
ggplot(avg_steps_per_interval, aes(x = interval, y = avg_steps, group = day_type, color = day_type)) +
  geom_line() +
  facet_wrap(~ day_type, ncol = 1) +
  labs(x = "5-minute Interval", y = "Average Number of Steps", title = "Average Number of Steps per 5-mi
  theme_minimal()
```


Average Number of Steps per 5−minute Interval
weekday
Average Number of Steps
200
150
100
50
0
0
500
1000
1500
2000
5−minute Interval
day_type
weekday