Bertinelli_Gabriele_Rlab01

April 17, 2024

1 RLab01 - Gabriele Bertinelli (2103359)

[3]:
```r
library(tidyverse)
# library(ggplot2)
# library(tibble)
# library(dplyr)
library(geosphere)

Sys.setlocale("LC_TIME", "English")
```

'English_United States.1252'

1)

[ ]:
```r
# read_csv is from tidyverse, faster than read.csv

d1 <- read_csv('./Data_CitiBike/JC-201902-citibike-tripdata.csv')
d2 <- read_csv('./Data_CitiBike/JC-201903-citibike-tripdata.csv')
d3 <- read_csv('./Data_CitiBike/JC-201904-citibike-tripdata.csv')
d4 <- read_csv('./Data_CitiBike/JC-201905-citibike-tripdata.csv')
d5 <- read_csv('./Data_CitiBike/JC-201906-citibike-tripdata.csv')
```

[5]:
```r
str(d1)
```

```
spc_tbl_ [18,565 × 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ tripduration           : num [1:18565] 142 223 106 370 315 …
 $ starttime              : POSIXct[1:18565], format: "2019-02-01 15:35:02"
"2019-02-01 17:00:46" …
 $ stoptime               : POSIXct[1:18565], format: "2019-02-01 15:37:24"
"2019-02-01 17:04:30" …
 $ start station id       : num [1:18565] 3183 3183 3183 3183 3183 …
 $ start station name     : chr [1:18565] "Exchange Place" "Exchange Place"
"Exchange Place" "Exchange Place" …
 $ start station latitude : num [1:18565] 40.7 40.7 40.7 40.7 40.7 …
 $ start station longitude: num [1:18565] -74 -74 -74 -74 -74 …
 $ end station id         : num [1:18565] 3639 3681 3184 3211 3273 …
 $ end station name       : chr [1:18565] "Harborside" "Grand St" "Paulus Hook"
"Newark Ave" …
```

```
 $ end station latitude   : num [1:18565] 40.7 40.7 40.7 40.7 40.7 …
 $ end station longitude  : num [1:18565] -74 -74 -74 -74 -74 …
 $ bikeid                 : num [1:18565] 29677 26234 29588 29250 29586 …
 $ usertype               : chr [1:18565] "Subscriber" "Subscriber" "Subscriber"
"Subscriber" …
 $ birth year             : num [1:18565] 1963 1992 1960 1976 1980 …
 $ gender                 : num [1:18565] 1 2 1 1 1 1 1 1 1 1 …
 - attr(*, "spec")=
  .. cols(
  ..   tripduration = col_double(),
  ..   starttime = col_datetime(format = ""),
  ..   stoptime = col_datetime(format = ""),
  ..   `start station id` = col_double(),
  ..   `start station name` = col_character(),
  ..   `start station latitude` = col_double(),
  ..   `start station longitude` = col_double(),
  ..   `end station id` = col_double(),
  ..   `end station name` = col_character(),
  ..   `end station latitude` = col_double(),
  ..   `end station longitude` = col_double(),
  ..   bikeid = col_double(),
  ..   usertype = col_character(),
  ..   `birth year` = col_double(),
  ..   gender = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

2)

[6]:
```
# bind_rows() is useful for combining data frames wherein the columns are all␣
↪identical
df <- as_tibble(bind_rows(d1, d2, d3, d4, d5))
```

[7]:
```
str(df) # check shape of df
```

```
tibble [150,792 × 15] (S3: tbl_df/tbl/data.frame)
 $ tripduration           : num [1:150792] 142 223 106 370 315 …
 $ starttime              : POSIXct[1:150792], format: "2019-02-01 15:35:02"
"2019-02-01 17:00:46" …
 $ stoptime               : POSIXct[1:150792], format: "2019-02-01 15:37:24"
"2019-02-01 17:04:30" …
 $ start station id       : num [1:150792] 3183 3183 3183 3183 3183 …
 $ start station name     : chr [1:150792] "Exchange Place" "Exchange Place"
"Exchange Place" "Exchange Place" …
 $ start station latitude : num [1:150792] 40.7 40.7 40.7 40.7 40.7 …
 $ start station longitude: num [1:150792] -74 -74 -74 -74 -74 …
 $ end station id         : num [1:150792] 3639 3681 3184 3211 3273 …
 $ end station name       : chr [1:150792] "Harborside" "Grand St" "Paulus Hook"
```

```
"Newark Ave" …
 $ end station latitude   : num [1:150792] 40.7 40.7 40.7 40.7 40.7 …
 $ end station longitude  : num [1:150792] -74 -74 -74 -74 -74 …
 $ bikeid                 : num [1:150792] 29677 26234 29588 29250 29586 …
 $ usertype               : chr [1:150792] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" …
 $ birth year             : num [1:150792] 1963 1992 1960 1976 1980 …
 $ gender                 : num [1:150792] 1 2 1 1 1 1 1 1 1 1 …
```

3)

```
[8]: df <- drop_na(df) # drop rows with NA values
```

```
[9]: str(df)

     # no rows removed due to NA values
```

```
tibble [150,792 × 15] (S3: tbl_df/tbl/data.frame)
 $ tripduration           : num [1:150792] 142 223 106 370 315 …
 $ starttime              : POSIXct[1:150792], format: "2019-02-01 15:35:02"
"2019-02-01 17:00:46" …
 $ stoptime               : POSIXct[1:150792], format: "2019-02-01 15:37:24"
"2019-02-01 17:04:30" …
 $ start station id       : num [1:150792] 3183 3183 3183 3183 3183 …
 $ start station name     : chr [1:150792] "Exchange Place" "Exchange Place"
"Exchange Place" "Exchange Place" …
 $ start station latitude : num [1:150792] 40.7 40.7 40.7 40.7 40.7 …
 $ start station longitude: num [1:150792] -74 -74 -74 -74 -74 …
 $ end station id         : num [1:150792] 3639 3681 3184 3211 3273 …
 $ end station name       : chr [1:150792] "Harborside" "Grand St" "Paulus Hook"
"Newark Ave" …
 $ end station latitude   : num [1:150792] 40.7 40.7 40.7 40.7 40.7 …
 $ end station longitude  : num [1:150792] -74 -74 -74 -74 -74 …
 $ bikeid                 : num [1:150792] 29677 26234 29588 29250 29586 …
 $ usertype               : chr [1:150792] "Subscriber" "Subscriber"
"Subscriber" "Subscriber" …
 $ birth year             : num [1:150792] 1963 1992 1960 1976 1980 …
 $ gender                 : num [1:150792] 1 2 1 1 1 1 1 1 1 1 …
```

4) 4.1)

```
[10]: avg_trip_duration <- mean(df$tripduration)/60

      median_trip_duration <- median(df$tripduration)/60

      sprintf("Average trip duration: %.2f min", avg_trip_duration)
      sprintf("Median trip duration: %.2f min", median_trip_duration)
```

’Average trip duration: 12.81 min’

’Median trip duration: 5.68 min’

4.2)

```
[11]: min_trip_duration <- min(df$tripduration)/60
max_trip_duration <- max(df$tripduration)/60

sprintf("Min trip duration: %.2f min", min_trip_duration)
sprintf("Max trip duration: %.2f min", max_trip_duration)

sprintf('Max trip duration in hours: %.2f h', max_trip_duration/3600)
```

’Min trip duration: 1.02 min’

’Max trip duration: 28817.00 min’

’Max trip duration in hours: 8.00 h’

There’s a trip that’s really really long ahah. Almost 20 days. Maybe some long excursion… (or, more likely, wrong data)

4.3)

```
[12]: df_clean <- df %>% filter(tripduration <= 3*3600) # filter out trips longer␣
  ↪than 3 hours
```

```
[13]: avg_trip_duration_clean <- mean(df_clean$tripduration)/60

median_trip_duration_clean <- median(df_clean$tripduration)/60

sprintf("Average trip duration: %.2f min", avg_trip_duration_clean)
sprintf("Median trip duration: %.2f min", median_trip_duration_clean)

min_trip_duration_clean <- min(df_clean$tripduration)/60
max_trip_duration_clean <- max(df_clean$tripduration)/60

sprintf("Min trip duration: %.2f min", min_trip_duration_clean)
sprintf("Max trip duration: %.2f min", max_trip_duration_clean)

sprintf('Max trip duration in hours: %.2f h', max_trip_duration_clean/3600)
```

’Average trip duration: 9.22 min’

’Median trip duration: 5.67 min’

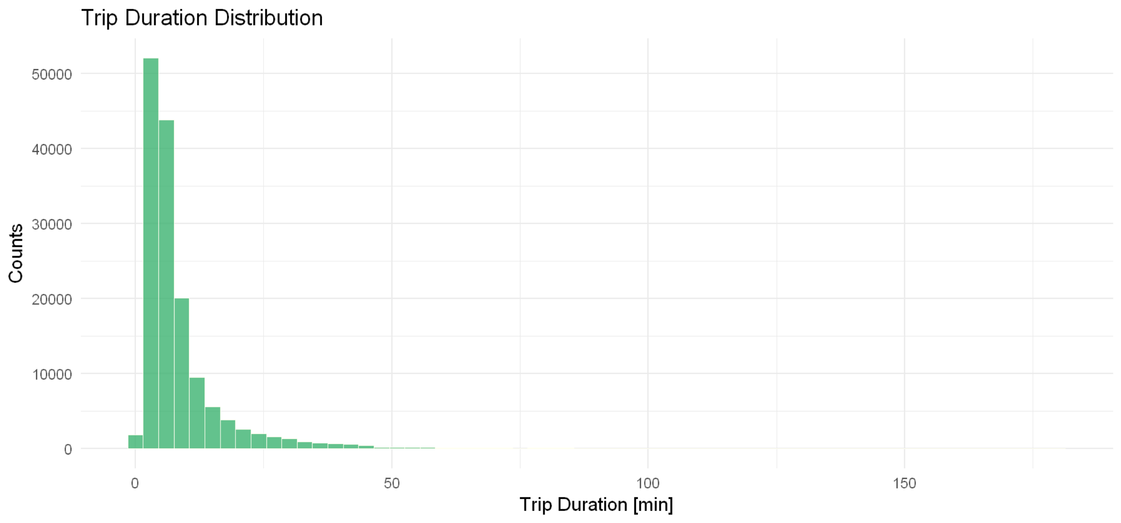
’Min trip duration: 1.02 min’

’Max trip duration: 180.00 min’

’Max trip duration in hours: 0.05 h’

4.4)

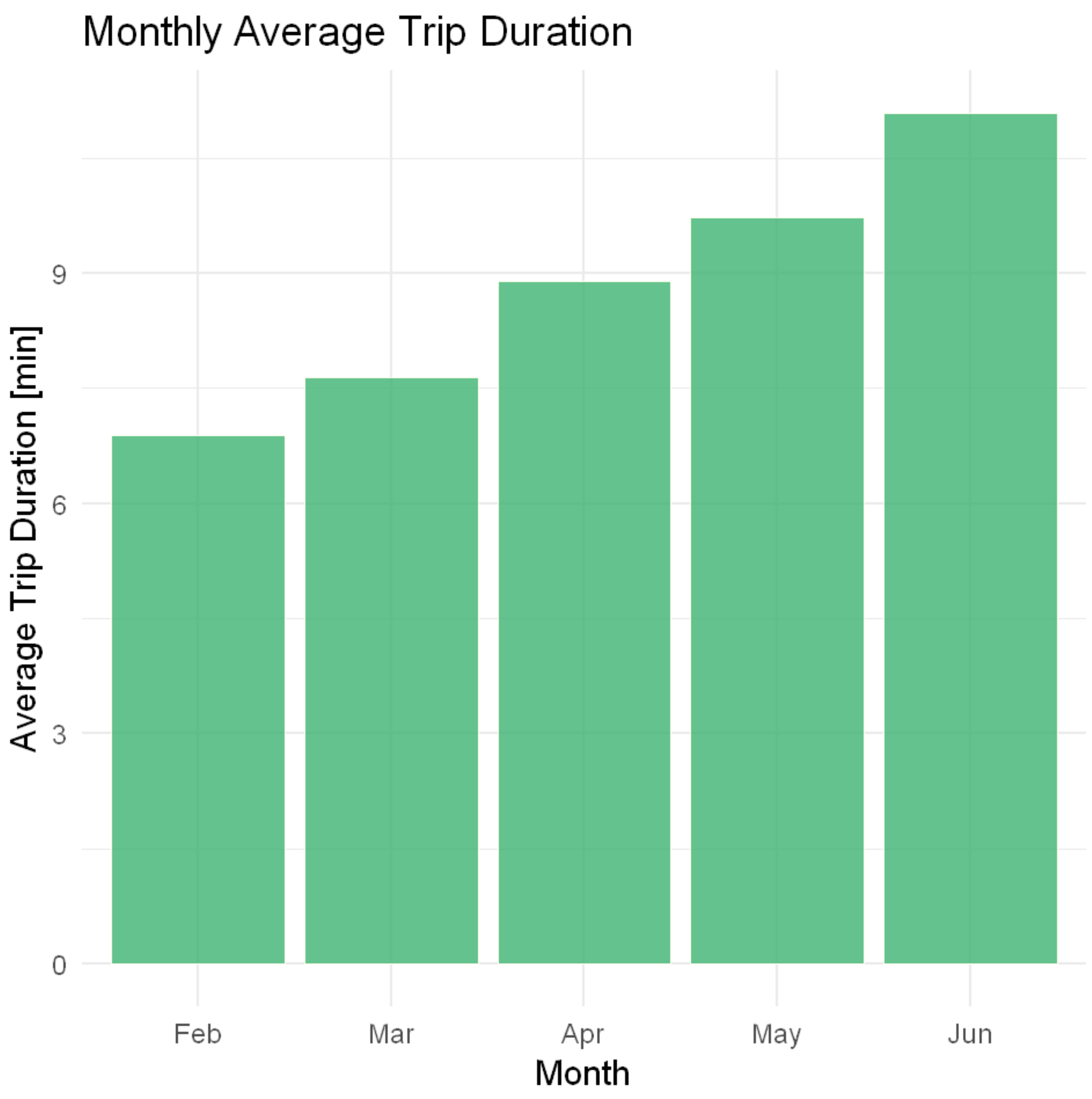
```
[26]: options(repr.plot.width = 15, repr.plot.height = 7) # set plot size

time_duration_hist <- ggplot(data=df_clean, aes(x=tripduration/60)) +
                                                geom_histogram(binwidth=3,␣
 ↪fill='mediumseagreen', color='ivory', alpha=0.8) +
                                                labs(title='Trip Duration␣
 ↪Distribution', x='Trip Duration [min]', y='Counts') +
                                                # xlim(c(0, 3600)) +
                                                theme_minimal(base_size = 18)

time_duration_hist # plot trip duration distribution
```



5)

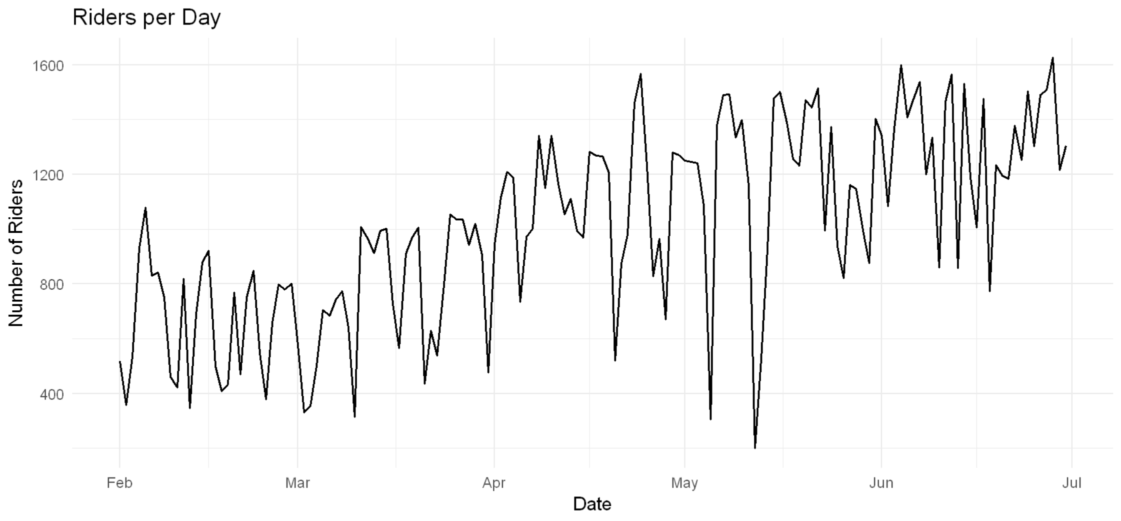
```
[15]: monthly_avg_trip_duration <- df_clean %>%
                                                            mutate(month =␣
 ↪lubridate::month(starttime)) %>% # mutate add a new col. lubridate::month()␣
 ↪returns the month number
                                                            group_by(month)␣
 ↪%>%
                                                            summarize(avg_trip_duration␣
 ↪= mean(tripduration)) # summarize() is used to aggregate data by group

monthly_avg_trip_duration <- monthly_avg_trip_duration %>%
                                                            mutate(month_w␣
 ↪= c('Feb', 'Mar', 'Apr', 'May', 'Jun'))  # add month names col for the plot


options(repr.plot.width = 8, repr.plot.height = 8)
```

```
level_order <- c('Feb', 'Mar', 'Apr', 'May', 'Jun') # order of months for plot
monthly_plot <- ggplot(data= monthly_avg_trip_duration, aes(x= factor(month_w,␣
 ↪level=level_order), y=avg_trip_duration/60)) +
                                        geom_bar(stat='identity',␣
 ↪fill='mediumseagreen', color='ivory', alpha=0.8) +
                                        labs(title='Monthly Average Trip␣
 ↪Duration', x='Month', y='Average Trip Duration [min]') +
                                        theme_minimal(base_size = 18)

monthly_plot # barplot of monthly average trip duration
```

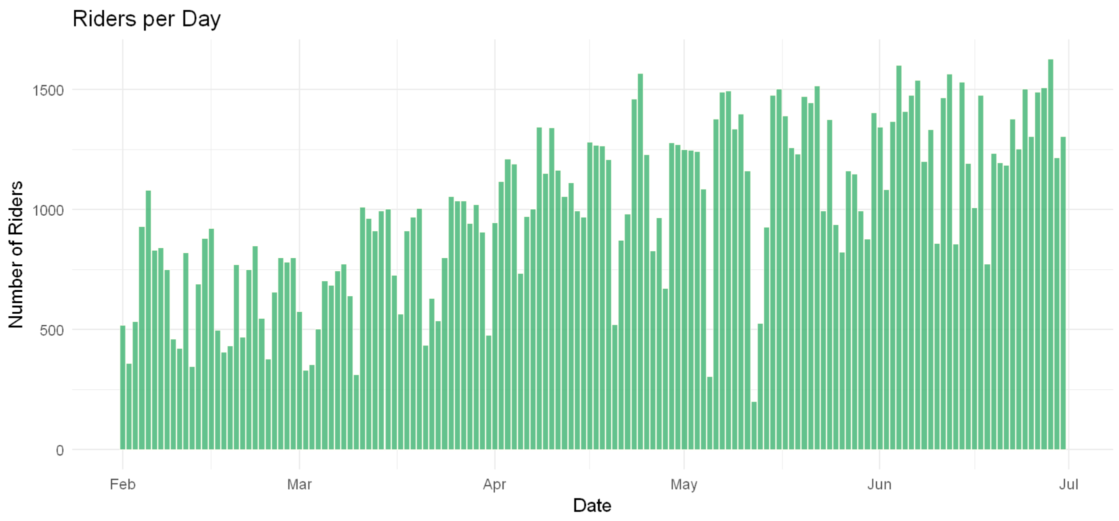


6) 6.1)

[16]:
```
riders_per_day <- df_clean %>%
                                        mutate(date = as.Date(starttime,␣
 ↪locale="EN-us")) %>% # convert starttime to date obj
                                        group_by(date) %>%
                                        summarize(riders_cnt = n()) # count␣
 ↪number of riders per day


options(repr.plot.width = 15, repr.plot.height = 7)

riders_per_day_plot <- ggplot(data=riders_per_day, aes(x=date, y=riders_cnt)) +
                                                geom_line(color='black', lwd=1)␣
 ↪+
                                                labs(title='Riders per Day',␣
 ↪x='Date', y='Number of Riders') +
                                                theme_minimal(base_size = 18)

riders_per_day_plot # plot ridership per day
```



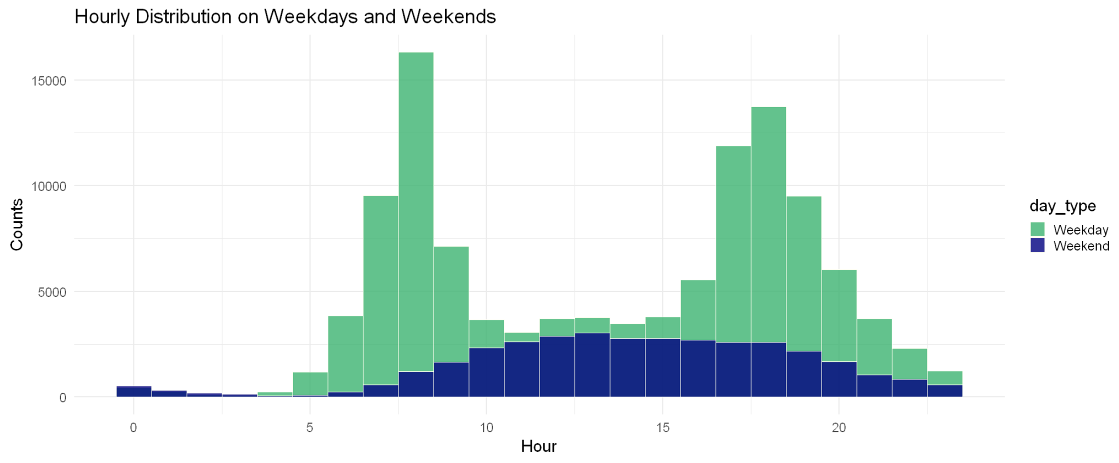
[17]:
```
options(repr.plot.width = 15, repr.plot.height = 7)

riders_per_day_barplot <- ggplot(data=riders_per_day, aes(x=date,␣
 ↪y=riders_cnt)) +
                                                      geom_bar(stat='identity',␣
 ↪fill='mediumseagreen', color='ivory', alpha=0.8) +
                                                      labs(title='Riders per␣
 ↪Day', x='Date', y='Number of Riders') +
                                                         theme_minimal(base_size␣
↪= 18)

riders_per_day_barplot # barplot of ridership per day
```



6.2)

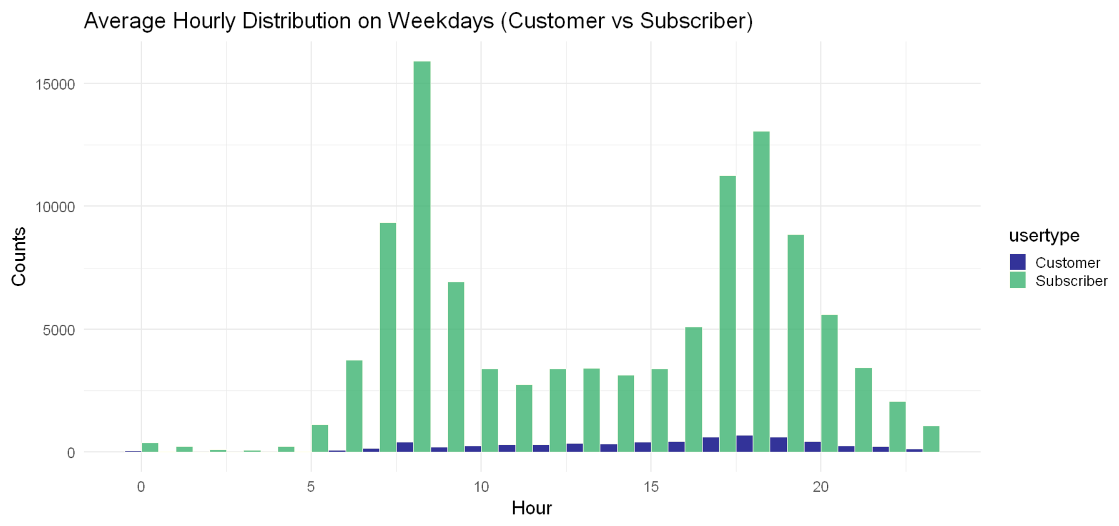
[18]:
```
# Extract the day of the week from the starttime col
df_clean$weekday <- lubridate::wday(df_clean$starttime, label = TRUE)

# Create a new col to separate between weekdays and weekends
df_clean$day_type <- ifelse(df_clean$weekday %in% c("Sat", "Sun"), "Weekend",␣
↪"Weekday")

# Extract the hour from the starttime column
df_clean$hour <- lubridate::hour(df_clean$starttime)


options(repr.plot.width = 17, repr.plot.height = 7)

hourly_distribution <- ggplot(data = df_clean, aes(x = hour, fill = day_type)) +
                                                  geom_histogram(binwidth = 1,␣
↪position = "identity", alpha = 0.8, color='ivory') +
                                                  labs(title = "Hourly␣
↪Distribution on Weekdays and Weekends", x = "Hour", y = "Counts") +
                                                  scale_fill_manual(values =␣
↪c("Weekday" = "mediumseagreen", "Weekend" = "navy")) +
                                                  theme_minimal(base_size=18)
```

```
hourly_distribution # hourly distribution on weekdays and weekends
```



6.3)

[19]:
```
# Filter the data for weekdays
df_weekdays <- df_clean %>% filter(day_type == "Weekday")

# Group the data by hour and user type
hourly_distribution <- df_weekdays %>%
                                                group_by(hour, usertype) %>%
                                                # summarize(avg_count =␣
 ↪mean(tripduration)) %>%
                                                ungroup()

options(repr.plot.width = 15, repr.plot.height = 7)

hourly_plot <- ggplot(data = hourly_distribution, aes(x = hour, fill =␣
 ↪usertype)) +
                                geom_histogram(binwidth = 1, position =␣
 ↪"dodge", alpha = 0.8, color='ivory') +
                                labs(title = "Average Hourly Distribution on␣
 ↪Weekdays (Customer vs Subscriber)",
                                                x = "Hour", y = "Counts") +
                                scale_fill_manual(values = c("Customer" =␣
 ↪"navy", "Subscriber" = "mediumseagreen")) +
                                theme_minimal(base_size = 18)

hourly_plot # hourly distribution on weekdays, separating customer and␣
 ↪subscriber users
```

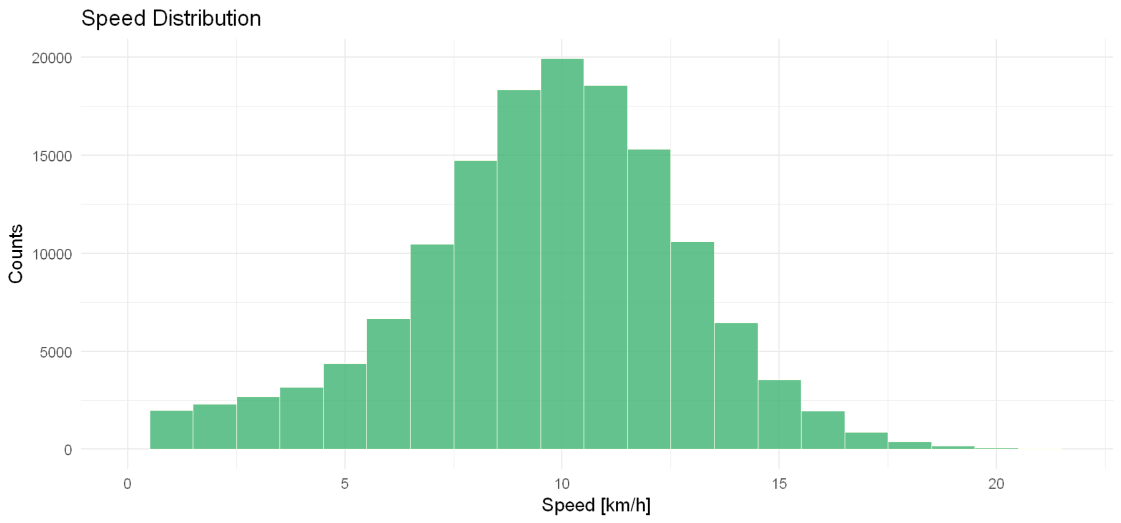


7) 7.1)

[36]:
```r
df_vel <- df_clean %>% filter(tripduration <= 3600) # filter out trips longer␣
 ↪than 1 hour

# calculate distance between start and end stations
df_vel$dist <- distHaversine(cbind(df_vel$`start station longitude`,␣
 ↪df_vel$`start station latitude`),
                                                          cbind(df_vel$`end␣
 ↪station longitude`, df_vel$`end station latitude`))

df_vel$speed <- (df_vel$dist / df_vel$tripduration)*3.6

options(repr.plot.width = 15, repr.plot.height = 7)

speed_hist <- ggplot(data=df_vel, aes(x=speed)) +
                                geom_histogram(binwidth=1,␣
 ↪fill='mediumseagreen', color='ivory', alpha=0.8) +
                                labs(title='Speed Distribution', x='Speed [km/
 ↪h]', y='Counts') +
                                xlim(c(0, 6*3.6)) +
                                # ylim(c(0, 2500)) +
                                theme_minimal(base_size = 18)

speed_hist
```

```
Warning message:
"Removed 26 rows containing non-finite outside the scale range
(`stat_bin()`)."
```

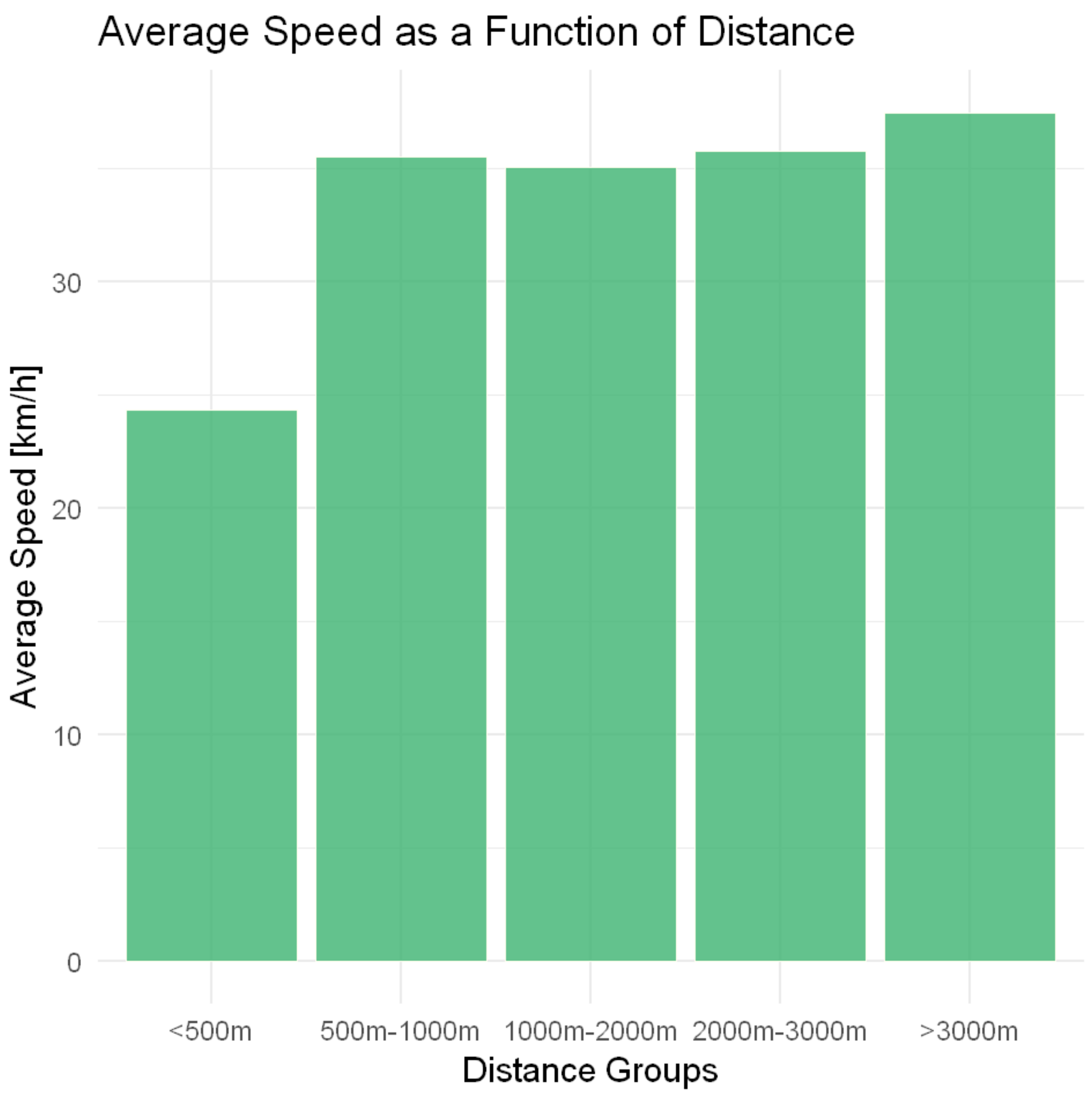
```
Warning message:
"Removed 2 rows containing missing values or values outside the scale
range
(`geom_bar()`)."
```



7.2)

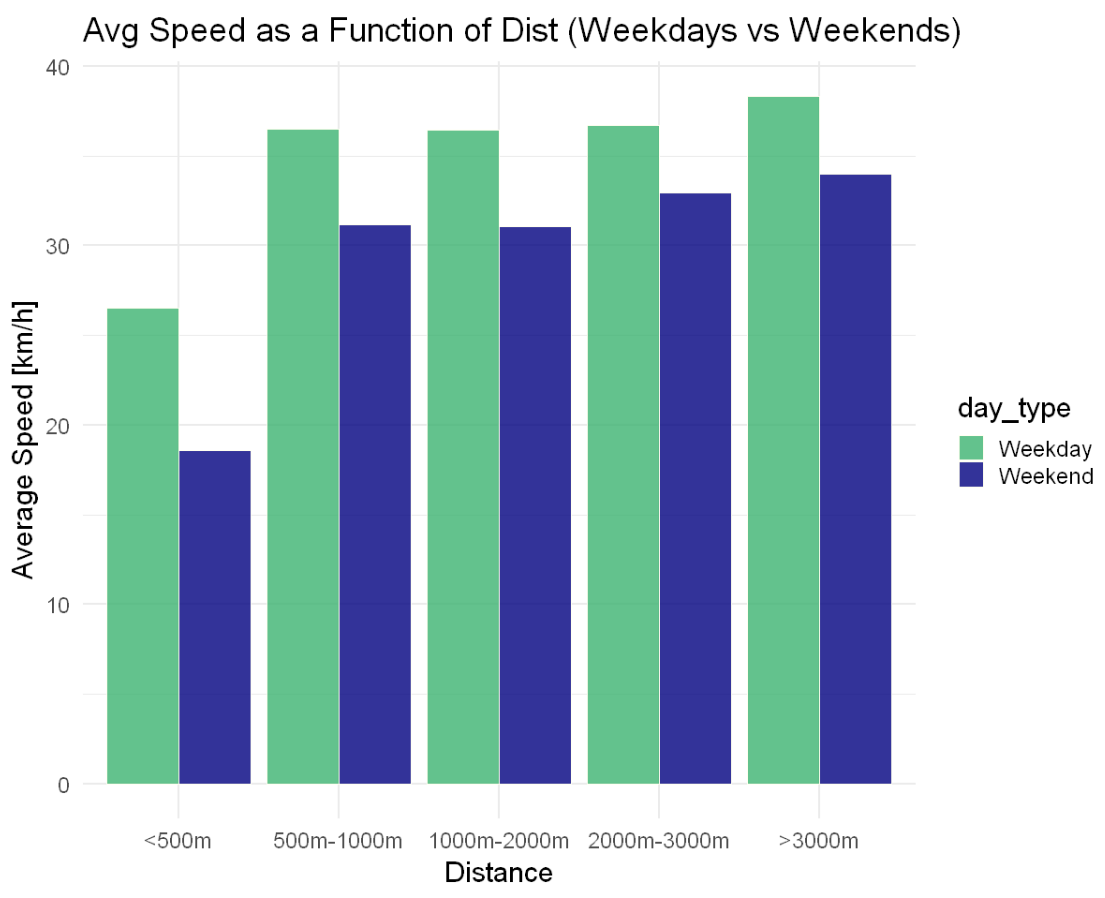
[37]:
```
# Filter the data for distance groups
df_group1 <- df_vel %>% filter(dist < 500)
df_group2 <- df_vel %>% filter(dist >= 500 & dist < 1000)
df_group3 <- df_vel %>% filter(dist >= 1000 & dist < 2000)
df_group4 <- df_vel %>% filter(dist >= 2000 & dist < 3000)
df_group5 <- df_vel %>% filter(dist >= 3000)

avg_speed_group1 <- mean(df_group1$speed)
avg_speed_group2 <- mean(df_group2$speed)
avg_speed_group3 <- mean(df_group3$speed)
avg_speed_group4 <- mean(df_group4$speed)
avg_speed_group5 <- mean(df_group5$speed)


avg_speed_dist <- data.frame(group_dist = c("<500m", "500m-1000m",␣
 ↪"1000m-2000m", "2000m-3000m", ">3000m"),
                                                                avg_speed =␣
 ↪c(avg_speed_group1, avg_speed_group2, avg_speed_group3,

 ↪avg_speed_group5)*3.6)

options(repr.plot.width = 8, repr.plot.height = 8)
```

```r
level_order <- c("<500m", "500m-1000m", "1000m-2000m", "2000m-3000m", ">3000m")
speed_distance_plot <- ggplot(data = avg_speed_dist, aes(x = factor(group_dist,␣
 ↪level=level_order), y = avg_speed)) +
                                                geom_bar(stat = "identity",␣
 ↪fill = "mediumseagreen", color = "ivory", alpha = 0.8) +
                                                labs(title = "Average Speed as␣
 ↪a Function of Distance",
                                                x = "Distance Groups", y =␣
 ↪"Average Speed [km/h]") +
                                                theme_minimal(base_size=18)

speed_distance_plot # Plot the average speed as a function of distance
```



7.3)

[38]:
```
# Filter the data for weekdays
df_weekday <- df_vel %>% filter(day_type == "Weekday")

# Filter the data for weekends
df_weekend <- df_vel %>% filter(day_type == "Weekend")

# same operation as before, but now for weekdays and weekends

df_group1d <- df_weekday %>% filter(dist < 500)
df_group2d <- df_weekday %>% filter(dist >= 500 & dist < 1000)
df_group3d <- df_weekday %>% filter(dist >= 1000 & dist < 2000)
df_group4d <- df_weekday %>% filter(dist >= 2000 & dist < 3000)
df_group5d <- df_weekday %>% filter(dist >= 3000)

df_group1w <- df_weekend %>% filter(dist < 500)
df_group2w <- df_weekend %>% filter(dist >= 500 & dist < 1000)
df_group3w <- df_weekend %>% filter(dist >= 1000 & dist < 2000)
df_group4w <- df_weekend %>% filter(dist >= 2000 & dist < 3000)
df_group5w <- df_weekend %>% filter(dist >= 3000)


avg_speed_group1d <- mean(df_group1d$speed)
avg_speed_group2d <- mean(df_group2d$speed)
avg_speed_group3d <- mean(df_group3d$speed)
avg_speed_group4d <- mean(df_group4d$speed)
avg_speed_group5d <- mean(df_group5d$speed)

avg_speed_group1w <- mean(df_group1w$speed)
avg_speed_group2w <- mean(df_group2w$speed)
avg_speed_group3w <- mean(df_group3w$speed)
avg_speed_group4w <- mean(df_group4w$speed)
avg_speed_group5w <- mean(df_group5w$speed)


avg_speed_distd <- data.frame(group_dist = c("<500m", "500m-1000m",␣
 ↪"1000m-2000m", "2000m-3000m", ">3000m"),
                                                                avg_speed =␣
 ↪c(avg_speed_group1d, avg_speed_group2d, avg_speed_group3d,

 ↪avg_speed_group5d)*3.6,
                                                                day_type =␣
 ↪'Weekday')

avg_speed_distw <- data.frame(group_dist = c("<500m", "500m-1000m",␣
 ↪"1000m-2000m", "2000m-3000m", ">3000m"),
                                                              avg_speed =␣
↪c(avg_speed_group1w, avg_speed_group2w, avg_speed_group3w,

↪avg_speed_group5w)*3.6,
                                                              day_type =␣
↪'Weekend')

df_avg_speed_dist <- rbind(avg_speed_distd, avg_speed_distw) # combine the two␣
↪dataframes

options(repr.plot.width = 10, repr.plot.height = 8)

level_order <- c("<500m", "500m-1000m", "1000m-2000m", "2000m-3000m", ">3000m")

# Plot the average speed as a function of distance
speed_distance_plotwd <- ggplot(data = df_avg_speed_dist, aes(x =␣
↪factor(group_dist, level=level_order), y = avg_speed, fill=day_type)) +
                                                        geom_bar(position='dodge',␣
↪stat = "identity", alpha = 0.8, color='ivory') +
                                                        labs(title = "Avg Speed␣
↪as a Function of Dist (Weekdays vs Weekends) ",
                                                        x = "Distance", y =␣
↪"Average Speed [km/h]") +
                                                        scale_fill_manual(values␣
↪= c("Weekday" = "mediumseagreen", "Weekend" = "navy")) +
                                                        theme_minimal(base_size␣
↪= 18)

speed_distance_plotwd
```



8) 8.1)

[23]:
```
start_station <- df_clean %>% group_by(`start station name`) %>%␣
  ↪summarize(count = n()) %>% arrange(desc(count))

end_station <- df_clean %>% group_by(`end station name`) %>% summarize(count =␣
  ↪n()) %>% arrange(count)

sprintf('Most common start station --> %s', start_station$`start station␣
  ↪name`[1])
sprintf('Least popular end station --> %s', end_station$`end station name`[1])
```

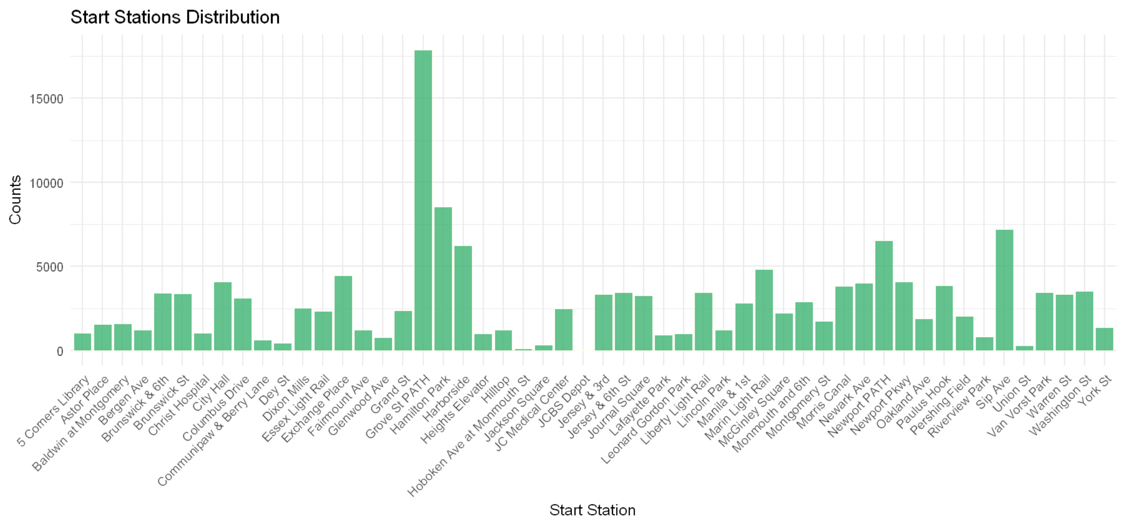
'Most common start station –> Grove St PATH'

'Least popular end station –> 1 Ave & E 16 St'

8.2)

[24]:
```
start_station <- df_clean %>% group_by(`start station name`) #%>%␣
  ↪summarize(count = n())
```

```
# start_station

options(repr.plot.width = 15, repr.plot.height = 7)

# start_distr <- ggplot(data=start_station, aes(x=`start station name`)) +
#                   geom_histogram(stat='count', binwidth = 1,␣
 ↪fill='mediumseagreen', color='ivory', alpha=0.8) +
#                   labs(title='Start Station Distribution', x='Start Station',␣
 ↪y='Frequency')
#                   theme_minimal()+
#                   theme(axis.text.x = element_text(angle = 45, hjust = 1))

start_distr <- ggplot(data=start_station, aes(x=`start station name`)) +
                                geom_bar(fill='mediumseagreen', color='ivory',␣
 ↪alpha=0.8) +
                                labs(title='Start Stations Distribution',␣
 ↪x='Start Station', y='Counts') +
                                theme_minimal(base_size = 15) +
                                theme(axis.text.x = element_text(angle = 45,␣
 ↪hjust = 1))

start_distr
```



8.3)

```
[25]: common_route <- df_clean %>% group_by(`start station name`, `end station name`)␣
 ↪%>% summarize(count = n()) %>% arrange(desc(count))
```

```
least_route <- df_clean %>% group_by(`start station name`, `end station name`)␣
↪%>% summarize(count = n()) %>% arrange(count)

cat('\n3 most common routes:\n')
for (i in 1:3) {
        print(sprintf('%s --> %s, counts: %d', common_route$`start station␣
↪name`[i], common_route$`end station name`[i], common_route$count[i]))
}

cat('\n3 least popular routes:\n')
for (i in 1:3) {
        print(sprintf('%s --> %s, counts: %d', least_route$`start station␣
↪name`[i], least_route$`end station name`[i], least_route$count[i]))
}
```

```
`summarise()` has grouped output by 'start station name'. You can
override
using the `.groups` argument.
`summarise()` has grouped output by 'start station name'. You can
override
using the `.groups` argument.


3 most common routes:
[1] "Hamilton Park --> Grove St PATH, counts: 3037"
[1] "Grove St PATH --> Hamilton Park, counts: 2318"
[1] "Brunswick & 6th --> Grove St PATH, counts: 1915"

3 least popular routes:
[1] "5 Corners Library --> Dixon Mills, counts: 1"
[1] "5 Corners Library --> Grand St, counts: 1"
[1] "Astor Place --> Brunswick & 6th, counts: 1"
```

```
[ ]:
```