divvy_project

Frank Jian

Last edited February 25, 2022

Contents

SCENARIOS **1**

SIX PHASE OF DATA ANALYSIS **1**

ASK . . . 1

PREPARE . . . 1

PROCESS . . . 3

ANALYZE/SHARE . . . 6

ACT . . . 25

SCENARIOS

You are a junior data analyst working in the marketing analyst team at Cyclistic, a bike-share company in Chicago. The director of marketing believes the company's future success depends on maximizing the number of annual memberships. Therefore, your team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, your team will design a new marketing strategy to convert casual riders into annual members. But first, Cyclistic executives must approve your recommendations, so they must be backed up with compelling data insights and professional data visualizations.

SIX PHASE OF DATA ANALYSIS

ASK

- What is the problem I am trying to solve?
- How can your insights drive business decisions?
- What steps have you taken to ensure that your data is clean?
- What trends or relationships did you find in the data?

PREPARE

Check the existing working directory

```
getwd()
```

```
## [1] "/Users/jianfrank/google_analytics_capstone/raw data/csv_file"
```

Set working directory

```
setwd("/Users/jianfrank/google_analytics_capstone/raw data/csv_file")
```

Import files Find related files ended with "data.*csv", saved it to `myFiles`. Inside `grand_data`, we first use lapply() & fread() to read every list of `myFiles` and then use do.call() to bind all the rows together and save it to `grand_data`.

```
myFiles <- list.files(pattern="data.*csv")
grand_data <- do.call(rbind, lapply(myFiles, fread, na.strings = c("", "NA")))
station_info <- read.csv("Divvy_Bicycle_Stations.csv")
```

Check data structures, variable, variable definition, records and datatype,

```
str(grand_data)
```

```
## Classes 'data.table' and 'data.frame':   5479096 obs. of  13 variables:
##  $ ride_id           : chr  "70B6A9A437D4C30D" "158A465D4E74C54A" "5262016E0F1F2F9A" "BE119628E44F87
##  $ rideable_type     : chr  "classic_bike" "electric_bike" "electric_bike" "electric_bike" ...
##  $ started_at        : chr  "2020-12-27 12:44:29" "2020-12-18 17:37:15" "2020-12-15 15:04:33" "2020-
##  $ ended_at          : chr  "2020-12-27 12:55:06" "2020-12-18 17:44:19" "2020-12-15 15:11:28" "2020-
##  $ start_station_name: chr  "Aberdeen St & Jackson Blvd" NA NA NA ...
##  $ start_station_id  : chr  "13157" NA NA NA ...
##  $ end_station_name  : chr  "Desplaines St & Kinzie St" NA NA NA ...
##  $ end_station_id    : chr  "TA1306000003" NA NA NA ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 41.8 ...
##  $ start_lng         : num  -87.7 -87.7 -87.7 -87.7 -87.6 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.8 ...
##  $ end_lng           : num  -87.6 -87.7 -87.7 -87.7 -87.6 ...
##  $ member_casual     : chr  "member" "member" "member" "member" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
str(station_info)
```

```
## 'data.frame':    841 obs. of  8 variables:
##  $ ID               : num  560 290 644 684 632 ...
##  $ Station.Name     : Factor w/ 841 levels "2112 W Peterson Ave",..: 495 384 807 600 166 53 714 239 2
##  $ Total.Docks      : int  11 15 11 15 15 11 15 15 11 4 ...
##  $ Docks.in.Service: int  11 15 11 15 15 11 15 15 11 4 ...
##  $ Status           : Factor w/ 3 levels "In Service","Not In Service",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Latitude         : num  41.8 41.9 41.9 41.7 41.9 ...
##  $ Longitude        : num  -87.7 -87.7 -87.7 -87.7 -87.7 ...
##  $ Location         : Factor w/ 841 levels "(41.64850076266409, -87.54608988761902)",..: 148 597 333
```

Transform the data adding multiple columns for easier analysis

```r
level <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
level_week <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
divvy_data <- grand_data %>%
  mutate(
    rideable_type = as.factor(rideable_type),
    started_at = as_datetime(started_at),
    ended_at =as_datetime(ended_at),
    member_casual = as.factor(member_casual),
    year = format(started_at, "%Y"),
    month = format(started_at, "%b"),
    week = format(started_at, "%U"),
    weekday = format(started_at, "%A"),
    hour = format(started_at, "%H"),
    year = factor(year),
    month = factor(month, levels = level),
    week = as.character(week),
    weekday = factor(weekday, level = level_week),
    hour_end = format(ended_at, "%H"),
    used_time = round(as.numeric(started_at %--% ended_at, "minutes"),2),
    distance = distHaversine(cbind(start_lng, start_lat),
                             cbind(end_lng, end_lat))
  )
```

PROCESS

Filter out used time is below zero

```r
divvy_data_clean <- divvy_data %>%
  filter(used_time > 0)
```

Check if there's any missing values

```r
divvy_data_clean %>%
  bind_shadow() %>%
  group_by(rideable_type, start_station_name_NA, start_station_id_NA, end_station_name_NA, end_station_
  summarise(n())
```

```
## # A tibble: 11 x 8
## # Groups:   rideable_type, start_station_name_NA, start_station_id_NA,
## #   end_station_name_NA, end_station_id_NA, end_lat_NA [11]
##    rideable_type start_station_name_NA start_station_id_NA end_station_name_NA
##    <fct>         <fct>                 <fct>               <fct>
##  1 classic_bike  !NA                   !NA                 !NA
##  2 classic_bike  !NA                   !NA                 NA
##  3 classic_bike  !NA                   !NA                 NA
##  4 docked_bike   !NA                   !NA                 !NA
##  5 docked_bike   !NA                   !NA                 NA
##  6 electric_bike !NA                   !NA                 !NA
##  7 electric_bike !NA                   !NA                 NA
##  8 electric_bike NA                    !NA                 !NA
##  9 electric_bike NA                    !NA                 NA
## 10 electric_bike NA                    NA                  !NA
```

```
## 11 electric_bike NA                    NA                   NA
## # ... with 4 more variables: end_station_id_NA <fct>, end_lat_NA <fct>,
## #   end_lng_NA <fct>, n() <int>
```

Check if there's any missing values given on different scenarios. We can break it down to 8 scenarios then can apply coping strategy to each of them.

- rideable_type: electric_bike, start_station_name: NA, start_station_id: NA(254320)-add "E_station" to start_station_name

```
divvy_data_clean <- divvy_data_clean %>%
  mutate(start_station_name = ifelse(rideable_type == "electric_bike" &
                                       !is.na(end_station_name) &
                                       !is.na(end_station_id) &
                                       is.na(start_station_id) &
                                       is.na(start_station_name) &
                                       !is.na(end_lat) &
                                       !is.na(end_lng), "E_station", start_station_name))
```

- rideable_type: electric_bike, end_station_name: NA, end_station_id: NA(292335)- add "E_station" to end_station_name

```
divvy_data_clean <- divvy_data_clean %>%
  mutate(end_station_name = ifelse(
    rideable_type == "electric_bike" &
      is.na(end_station_name) &
      is.na(end_station_id) &
      !is.na(start_station_name) &
      !is.na(start_station_id) &
      !is.na(end_lat) &
      !is.na(end_lng) &
      !is.na(start_lat) &
      !is.na(start_lng), "E_station", end_station_name))
```

- rideable_type: electric_bike, start_station_name: NA, start_station_id: NA, end_station_name: NA, end_station_id: NA(397056)-add "E_station" to start_station_name and end_station_name

```
divvy_data_clean <- divvy_data_clean %>%
  mutate(end_station_name = ifelse(rideable_type == "electric_bike" &
                                       is.na(end_station_name) &
                                       is.na(start_station_name) &
                                       is.na(start_station_id) &
                                       !is.na(end_lat) &
                                       !is.na(end_lng), "E_station", end_station_name),
         start_station_name = ifelse(rideable_type == "electric_bike" &
                                       is.na(start_station_id) &
                                       is.na(start_station_name) &
                                       is.na(end_station_id) &
                                       !is.na(end_lat) &
                                       !is.na(end_lng), "E_station", start_station_name))
```

After going through above processes, missing values from electric bikes have been cleaned to a point. Only thing you need to do is impute start_station_name column with value from `station_select` full joining with `divvy_data_clean` by longitude and latitude. If it still contains NA then filter out from the dataset.

- rideable_type: electric_bike, start_station_name: NA(3)-add "E_station" to start_station_name and end_station_name

```
station_select <- station_info %>%
  select(ID, Station.Name, Latitude, Longitude) %>%
  mutate(ID = as.factor(ID))
glimpse(station_select)
```

```
## Rows: 841
## Columns: 4
## $ ID           <fct> 560, 290, 644, 684, 632, 640, 690, 600, 650, 143649509660~
## $ Station.Name <fct> Marshfield Ave & 59th St, Kedzie Ave & Palmer Ct, Western~
## $ Latitude     <dbl> 41.78683, 41.92153, 41.86856, 41.72823, 41.94454, 41.9499~
## $ Longitude    <dbl> -87.66621, -87.70732, -87.68623, -87.66752, -87.65468, -8~
```

```
divvy_data_clean <- divvy_data_clean %>%
  full_join(station_select, by = c("start_lat" = "Latitude", "start_lng" = "Longitude")) %>%
  mutate(start_station_name = coalesce(start_station_name, Station.Name)) %>%
  select(-ID, -Station.Name) %>%
  filter(!is.na(ride_id))

divvy_data_clean <- divvy_data_clean %>%
        filter(!is.na(start_station_name), !is.na(start_station_id))
```

- rideable_type: classic_bike, end_station_name: NA, end_station_id: NA(4299)-add "E_station" to end_station_name

```
divvy_data_clean <- divvy_data_clean %>%
  mutate(end_station_name = ifelse(rideable_type == "classic_bike" &
                                     is.na(end_station_name) &
                                     !is.na(start_station_name) &
                                     !is.na(start_station_id) &
                                     !is.na(end_lat) &
                                     !is.na(end_lng) &
                                     !is.na(start_lat) &
                                     !is.na(start_lng), "E_station", end_station_name))
```

- rideable_type: classic_bike, end_station_name: NA, end_station_id: NA, end_lat: NA, end_lng:NA(4460)-filter out
- rideable_type: docked_bike, end_station_name: NA, end_station_id: NA, end_lat: NA, end_lng:NA(4460)-filter out

```
divvy_data_clean <- divvy_data_clean %>%
  filter(!is.na(end_lat),
         !is.na(end_lng))
```

- add "E01" to start_station_id & end_station_id

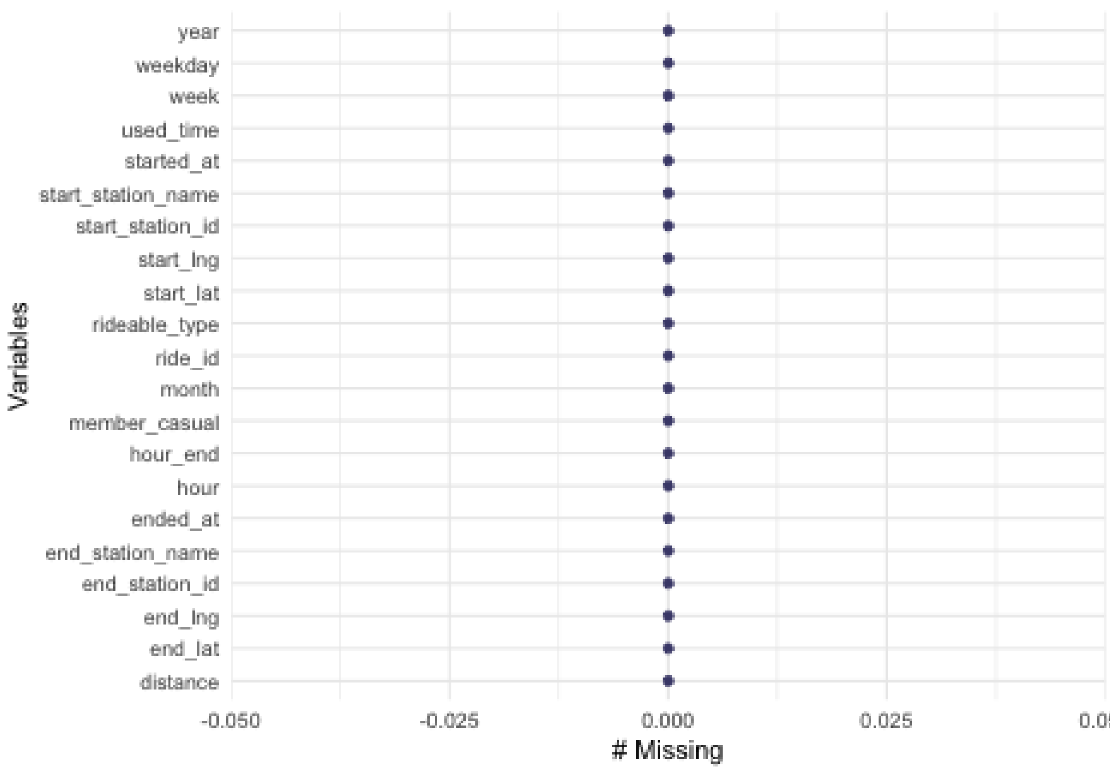
```
divvy_data_clean <- divvy_data_clean %>%
  mutate(start_station_id = ifelse(is.na(start_station_id), "E01", start_station_id),
         end_station_id = ifelse(is.na(end_station_id), "E01", end_station_id))
```

- add "E_station" to start_station_name if `is.na(start_station_name)` is true.
- add "E_station" to end_station_name if `is.na(end_station_name)` is true.

```
divvy_data_clean <- divvy_data_clean %>%
  mutate(start_station_name = ifelse(is.na(start_station_name), "E_station", start_station_name),
         end_station_name = ifelse(is.na(end_station_name), "E_station", end_station_name))
```

Check if there is any missing values

```
gg_miss_var(divvy_data_clean)
```



ANALYZE/SHARE

Once the dataset is all set for analyzing, we first look at what does the dataset look like.

```
str(divvy_data_clean)
```

```
## Classes 'data.table' and 'data.frame':   4821909 obs. of  21 variables:
##  $ ride_id           : chr  "70B6A9A437D4C30D" "726B352441501450" "15F369FDAED4E8E3" "0CFD61DFE00E60
##  $ rideable_type     : Factor w/ 3 levels "classic_bike",..: 1 1 3 3 3 2 1 3 3 3 ...
##  $ started_at        : POSIXct, format: "2020-12-27 12:44:29" "2020-12-12 15:37:11" ...
```

```
##  $ ended_at          : POSIXct, format: "2020-12-27 12:55:06" "2020-12-12 15:46:23" ...
##  $ start_station_name: chr  "Aberdeen St & Jackson Blvd" "Larrabee St & Armitage Ave" "Larrabee St &
##  $ start_station_id  : chr  "13157" "TA1309000006" "TA1309000006" "KA1503000043" ...
##  $ end_station_name  : chr  "Desplaines St & Kinzie St" "E_station" "Wells St & Walton St" "Desplain
##  $ end_station_id    : chr  "TA1306000003" "E01" "TA1306000011" "TA1306000003" ...
##  $ start_lat         : num  41.9 41.9 41.9 41.9 42 ...
##  $ start_lng         : num  -87.7 -87.6 -87.6 -87.6 -87.7 ...
##  $ end_lat           : num  41.9 41.9 41.9 41.9 41.9 ...
##  $ end_lng           : num  -87.6 -87.7 -87.6 -87.6 -87.6 ...
##  $ member_casual     : Factor w/ 2 levels "casual","member": 2 2 2 2 2 1 2 2 2 2 ...
##  $ year              : Factor w/ 2 levels "2020","2021": 1 1 1 1 1 1 1 1 1 1 ...
##  $ month             : Factor w/ 12 levels "Jan","Feb","Mar",..: 12 12 12 12 12 12 12 12 12 12 ...
##  $ week              : chr  "52" "49" "50" "52" ...
##  $ weekday           : Factor w/ 7 levels "Monday","Tuesday",..: 7 6 5 1 1 4 7 1 7 2 ...
##  $ hour              : chr  "12" "15" "13" "17" ...
##  $ hour_end          : chr  "12" "15" "14" "17" ...
##  $ used_time         : num  10.62 9.2 7.83 1.8 16.63 ...
##  $ distance          : num  1494 1363 2147 324 4717 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
head(divvy_data_clean)
```

```
##             ride_id rideable_type          started_at            ended_at
## 1: 70B6A9A437D4C30D  classic_bike 2020-12-27 12:44:29 2020-12-27 12:55:06
## 2: 726B352441501450  classic_bike 2020-12-12 15:37:11 2020-12-12 15:46:23
## 3: 15F369FDAED4E8E3 electric_bike 2020-12-18 13:53:56 2020-12-18 14:01:46
## 4: 0CFD61DFE00E6043 electric_bike 2020-12-28 17:10:25 2020-12-28 17:12:13
## 5: 0B040778F2EF7C84 electric_bike 2020-12-14 17:39:19 2020-12-14 17:55:57
## 6: 244CB936487039B7   docked_bike 2020-12-10 13:36:16 2020-12-10 14:37:03
##            start_station_name start_station_id          end_station_name
## 1: Aberdeen St & Jackson Blvd            13157 Desplaines St & Kinzie St
## 2: Larrabee St & Armitage Ave     TA1309000006                 E_station
## 3: Larrabee St & Armitage Ave     TA1309000006      Wells St & Walton St
## 4:   Kingsbury St & Kinzie St     KA1503000043 Desplaines St & Kinzie St
## 5:      Clark St & Leland Ave     TA1309000014                 E_station
## 6:      Clark St & Leland Ave     TA1309000014     Clark St & Leland Ave
##    end_station_id start_lat start_lng  end_lat   end_lng member_casual year
## 1:   TA1306000003  41.87773 -87.65479 41.88872 -87.64445        member 2020
## 2:            E01  41.91808 -87.64375 41.92000 -87.66000        member 2020
## 3:   TA1306000011  41.91811 -87.64380 41.90013 -87.63445        member 2020
## 4:   TA1306000003  41.88919 -87.63858 41.88910 -87.64248        member 2020
## 5:            E01  41.96713 -87.66745 41.93000 -87.64000        member 2020
## 6:   TA1309000014  41.96710 -87.66743 41.96710 -87.66743        casual 2020
##    month week  weekday hour hour_end used_time distance
## 1:   Dec   52   Sunday   12       12     10.62 1493.655
## 2:   Dec   49 Saturday   15       15      9.20 1362.890
## 3:   Dec   50   Friday   13       14      7.83 2146.519
## 4:   Dec   52   Monday   17       17      1.80  323.600
## 5:   Dec   50   Monday   17       17     16.63 4716.686
## 6:   Dec   49 Thursday   13       14     60.78    0.000
```

```
dim(divvy_data_clean)
```

```
## [1] 4821909      21
```

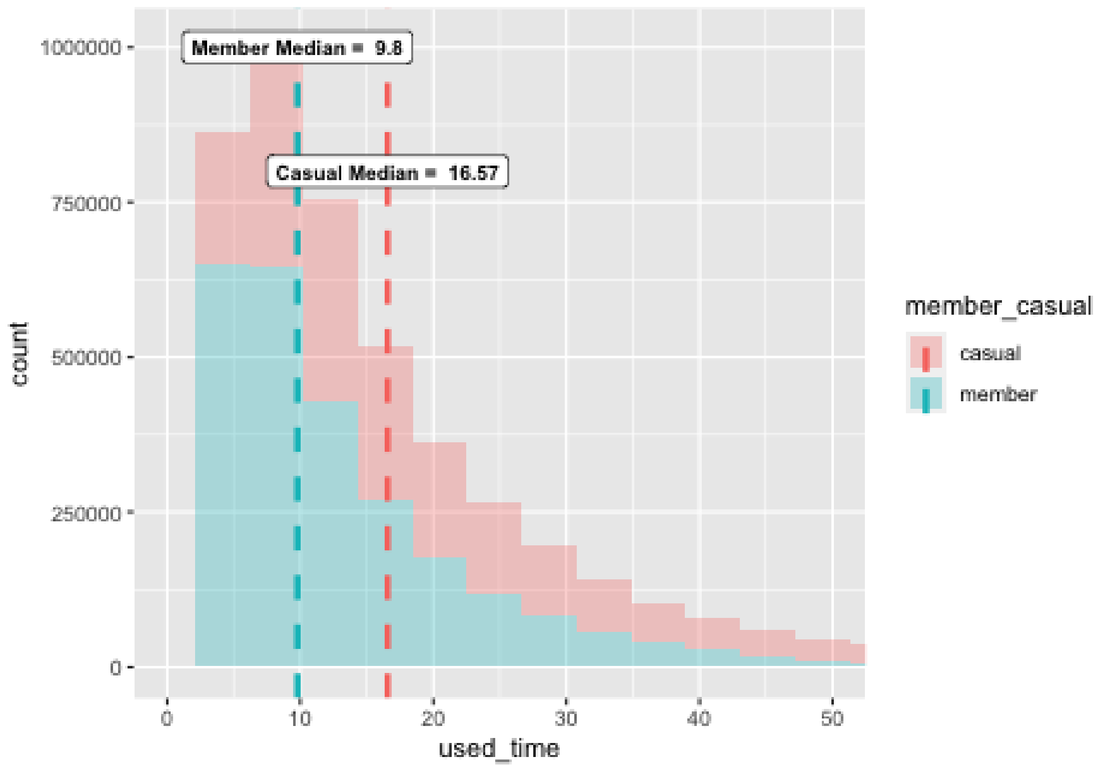
Then we can look at how used time and number of usage differ in group of types of bike and membership. As we see the result, customers who are casual user and uses the docked bike have the most spread_out time on the bike. Standard deviation of casual users riding docked bikes is more than 30 times larger than those who are member users riding docked bikes. We can deep-dive later. But let's see the median used time cross membership and the median used time cross type of bikes and membership first.

```
divvy_data_clean %>%
        group_by(rideable_type, member_casual) %>%
        summarise(sd_used_time = sd(used_time),
                  mean_used_time = mean(used_time),
                  median_used_time = median(used_time),
                  maximum_used_time = max(used_time),
                  minimum_used_time = min(used_time),
                  interquartile_used_time = iqr(used_time),
                  count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%"))
```

```
## # A tibble: 6 x 10
##   rideable_type member_casual sd_used_time mean_used_time median_used_time
##   <fct>         <fct>                <dbl>          <dbl>            <dbl>
## 1 classic_bike  casual                45.4           26.3             16.1
## 2 classic_bike  member                21.8           13.8             10.0
## 3 docked_bike   casual               691.            77.7             28.8
## 4 docked_bike   member                22.4           12.2              8.83
## 5 electric_bike casual                23.9           20.6             13.4
## 6 electric_bike member                15.4           12.7              9.2
## # ... with 5 more variables: maximum_used_time <dbl>, minimum_used_time <dbl>,
## #   interquartile_used_time <dbl>, count <int>, percent <chr>
```

Median for member riders and casual riders

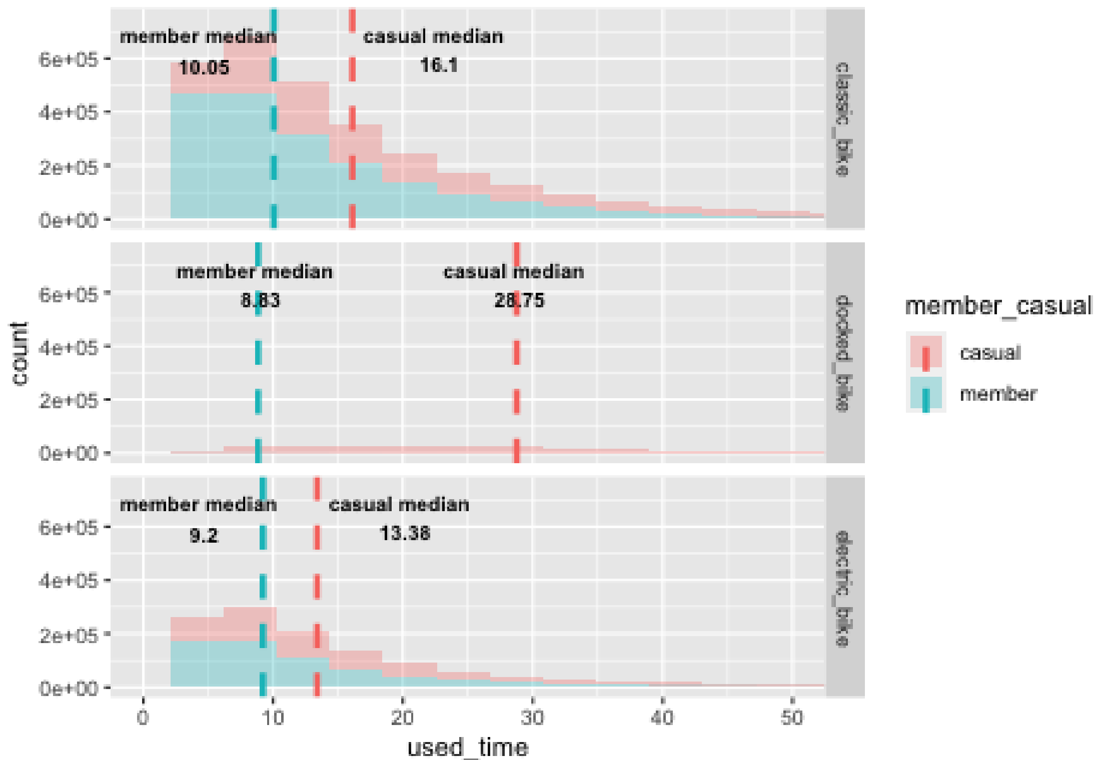
```
used_time_1 <- divvy_data_clean %>%
  group_by(member_casual) %>%
  summarise(median = median(used_time))
used_time_mb <- used_time_1 %>%
  filter(member_casual == "member") %>%
  pull(median)
used_time_ca <- used_time_1 %>%
  filter(member_casual == "casual") %>%
  pull(median)
ggplot(divvy_data_clean, aes(x=used_time, fill= member_casual))+
  geom_histogram(alpha=0.3, size=1.5, bins = 40)+
  scale_x_continuous(limits = c(0,160))+
  coord_cartesian(xlim = c(0,50))+
  geom_vline(data=used_time_1, aes(xintercept = median, color = member_casual),size=1.2, linetype = "da
  annotate(x=used_time_mb, y=1000000, label = paste("Member Median = ", used_time_mb),geom="label",size
  annotate(x=used_time_ca, y=800000, label = paste("Casual Median = ", used_time_ca),geom="label",size=
```



Median used time cross type of bikes and membership We can see for member users no matter what type of bike they have use they have pretty much the same median around 9. However, for casual users they have larger median used time especially in docked_bike category which has been used mostly by casual riders.

```
used_time_1 <- divvy_data_clean %>%
  group_by(rideable_type, member_casual) %>%
  summarise(median = median(used_time))

ggplot(divvy_data_clean, aes(x=used_time, fill= member_casual))+
  geom_histogram(alpha=0.3, size=1.5, bins = 40)+
  scale_x_continuous(limits = c(0,160))+
  coord_cartesian(xlim = c(0,50))+
  geom_vline(data=used_time_1, aes(xintercept = median, color = member_casual),size=1.2, linetype = "da
  geom_text_repel(
          data = used_time_1,
          aes(x = median,
              y = 750000,
              label = paste(member_casual, "median \n", median)),
            size = 3,
            fontface = "bold"
  )+
  facet_grid(rideable_type~.)
```



Use graph to see the median of different group

```
divvy_data_clean %>%
  group_by(rideable_type, member_casual) %>%
  summarise(sd_used_time = sd(used_time),
            mean_used_time = mean(used_time),
            median_used_time = median(used_time),
            maximum_used_time = max(used_time),
            minimum_used_time = min(used_time),
            interquartile_used_time = iqr(used_time),
            count = n()) %>%
  ungroup() %>%
  mutate(percent = paste0(round(count/sum(count),2)*100,"%"))
```

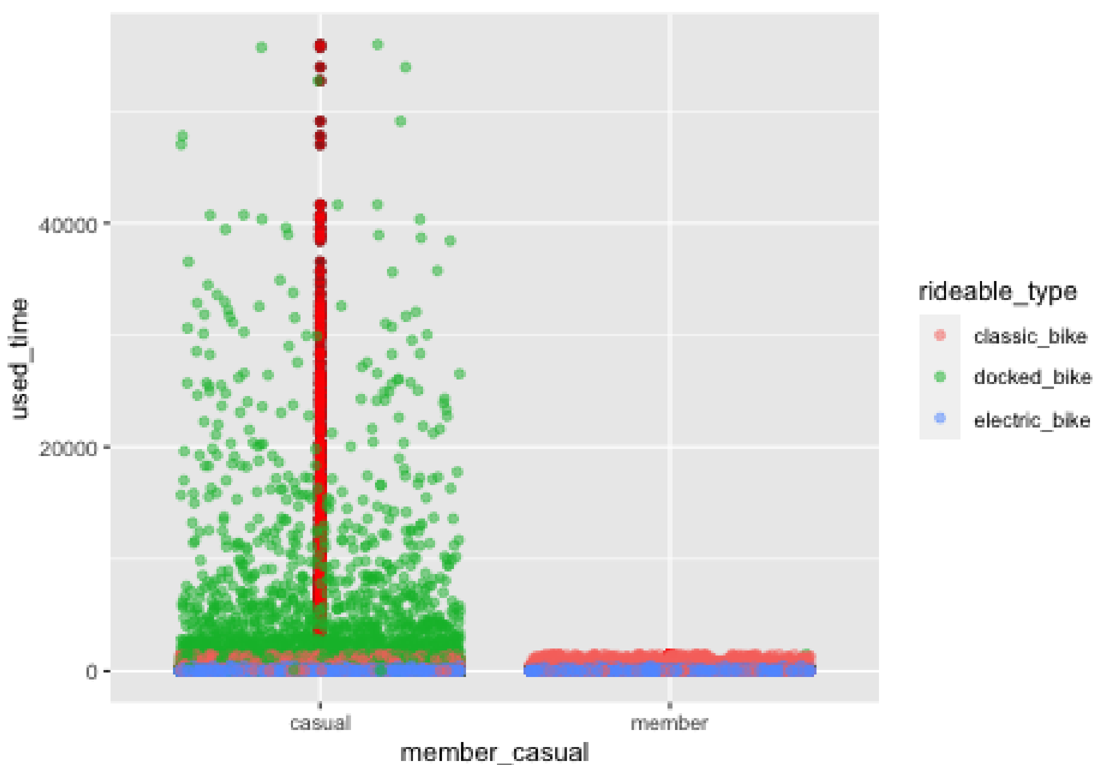
```
## # A tibble: 6 x 10
##   rideable_type member_casual sd_used_time mean_used_time median_used_time
##   <fct>         <fct>                <dbl>          <dbl>            <dbl>
## 1 classic_bike  casual                45.4           26.3            16.1
## 2 classic_bike  member                21.8           13.8            10.0
## 3 docked_bike   casual               691.            77.7            28.8
## 4 docked_bike   member                22.4           12.2             8.83
## 5 electric_bike casual                23.9           20.6            13.4
## 6 electric_bike member                15.4           12.7             9.2
## # ... with 5 more variables: maximum_used_time <dbl>, minimum_used_time <dbl>,
## #   interquartile_used_time <dbl>, count <int>, percent <chr>
```

```
third_quartile <- quantile(divvy_data_clean$used_time, 0.75) %>% unname()
first_quartile <- quantile(divvy_data_clean$used_time, 0.25) %>% unname()
iqr <- IQR(divvy_data_clean$used_time)
```

```r
lower <- first_quartile - 1.5*iqr
higher <- first_quartile + 1.5*iqr

divvy_outlier <- divvy_data_clean %>%
        filter(used_time < lower | used_time > higher)

divvy_data_clean %>%
        ggplot(aes(x = member_casual, y = used_time))+
        geom_boxplot()+
        geom_point(divvy_outlier, mapping = aes(x = member_casual, y = used_time), color = "red", alpha
        geom_jitter(aes(color = rideable_type), alpha = 0.5)
```



We just found that time casual users on a docked bike is the most spread out among all categories. What I mean by spread out is a few days away from time since they first used it. It's highly unlikely that users are on a bike all day long. We probably can assume there might be something wrong with casual users returning their bikes. (follow-up: used time outlier of end time frequency in month & weekday)

```r
divvy_doc_cas <- divvy_data_clean %>%
        filter(rideable_type == "docked_bike", member_casual == "casual")
out <- boxplot.stats(divvy_doc_cas$used_time)$out

divvy_doc_cas %>%
        filter(used_time %in% out) %>%
        count()
```

```
##        n
## 1: 26921
```

```
divvy_doc_cas %>%
        filter(used_time %in% out) %>%
        group_by(start_station_name, end_station_name) %>%
        summarise(count = n()) %>%
        arrange(desc(count))
```

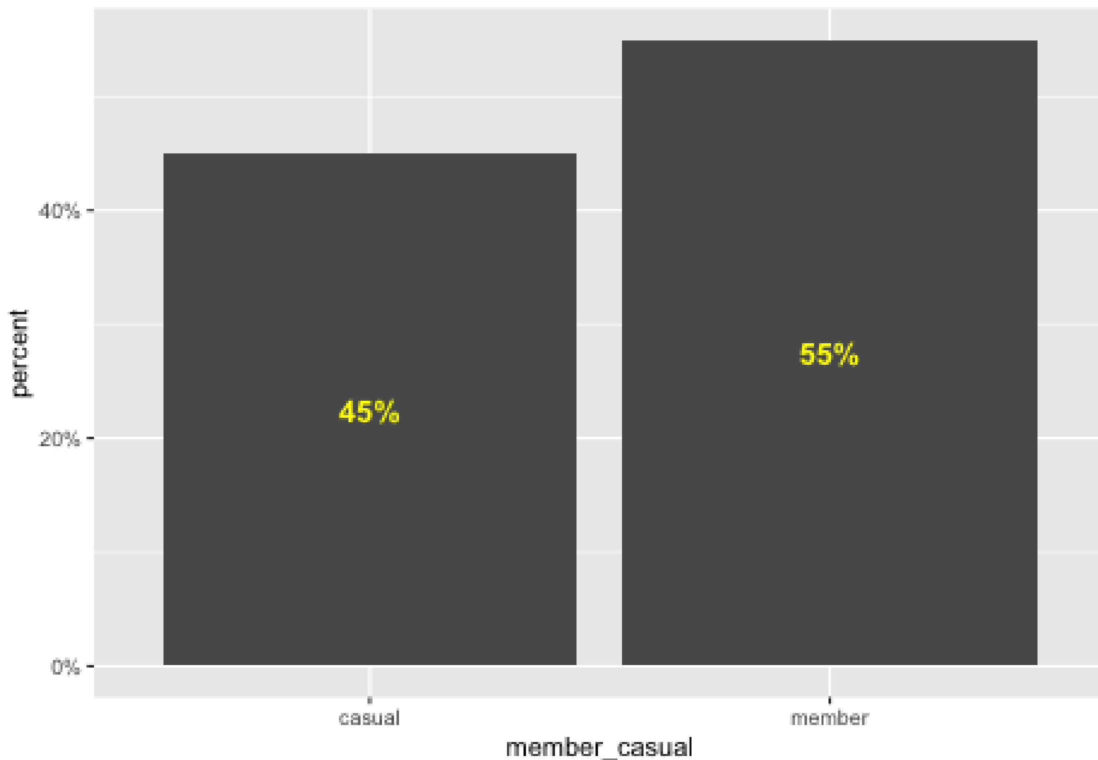
```
## # A tibble: 9,222 x 3
## # Groups:   start_station_name [648]
##    start_station_name         end_station_name           count
##    <chr>                      <chr>                      <int>
##  1 Streeter Dr & Grand Ave    Streeter Dr & Grand Ave      557
##  2 Millennium Park            Millennium Park              284
##  3 Michigan Ave & Oak St      Michigan Ave & Oak St        251
##  4 Montrose Harbor            Montrose Harbor              249
##  5 Fort Dearborn Dr & 31st St Fort Dearborn Dr & 31st St   221
##  6 Lake Shore Dr & Monroe St  Lake Shore Dr & Monroe St    169
##  7 Buckingham Fountain        Buckingham Fountain          151
##  8 Adler Planetarium          Adler Planetarium            149
##  9 Theater on the Lake        Theater on the Lake          144
## 10 Michigan Ave & 8th St      Michigan Ave & 8th St        130
## # ... with 9,212 more rows
```

```
divvy_doc_cas %>%
        filter(used_time %in% out) %>%
        distinct(end_station_name)
```

```
##                end_station_name
##   1:        Morgan St & Polk St
##   2: Indiana Ave & Roosevelt Rd
##   3:        MLK Jr Dr & 63rd St
##   4:      Michigan Ave & Lake St
##   5:      Damen Ave & Foster Ave
##  ---
## 654:       Racine Ave & 61st St
## 655:        Doty Ave & 111th St
## 656:       Halsted St & 96th St
## 657:  Commercial Ave & 100th St
## 658:       Summit Ave & 86th St
```

We move our focus to the number of rides between casual and member users. For number of rides itself, member riders use divvy more than casual user by 10%.

```
divvy_data_clean %>%
        group_by(member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = round(count/sum(count),2)) %>%
        ggplot(aes(x = member_casual, y = percent))+
        geom_bar(stat = "identity", position = "stack")+
        geom_text(aes(y = percent/2, label = paste0(percent*100, "%")), color = "yellow", fontface = "bo
        scale_y_continuous(labels = scales::percent)
```



Next, let's see what the numbers look like if we delve it in on monthly basis.

```
divvy_data_clean %>%
        filter(member_casual == "member") %>%
        group_by(month) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(count))
```

```
## # A tibble: 12 x 3
##    month  count percent
##    <fct>  <int> <chr>
##  1 Aug   350322 13%
##  2 Sep   346645 13%
##  3 Jul   340491 13%
##  4 Jun   321456 12%
##  5 Oct   311608 12%
##  6 May   246573 9%
##  7 Nov   202726 8%
##  8 Apr   184889 7%
##  9 Mar   134716 5%
## 10 Dec    92953 4%
## 11 Jan    72207 3%
## 12 Feb    36198 1%
```

```
divvy_data_clean %>%
        filter(member_casual == "casual") %>%
```

```
        group_by(month) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(count))
```

```
## # A tibble: 12 x 3
##    month  count percent
##    <fct>  <int> <chr>
##  1 Jul   393845 18%
##  2 Aug   364762 17%
##  3 Jun   327264 15%
##  4 Sep   315691 14%
##  5 May   230811 11%
##  6 Oct   210855 10%
##  7 Apr   125983 6%
##  8 Nov    81695 4%
##  9 Mar    78753 4%
## 10 Dec    26412 1%
## 11 Jan    15894 1%
## 12 Feb     9160 0%
```

Let's plot it on the graph to see differences in number of rides between member riders and casual riders. Overall, there's a noticeable trends that it climbed from January, peaked in July/August and declined ever since to the end of year. But you can tell from the plot that casual riders surpass member riders in numbers of rides from June to August.

```
divvy_data_clean %>%
        group_by(month, member_casual) %>%
        summarise(count = n())  %>%
        ggplot(aes(x = month, y = count, group = member_casual, color = member_casual))+
        geom_line()
```

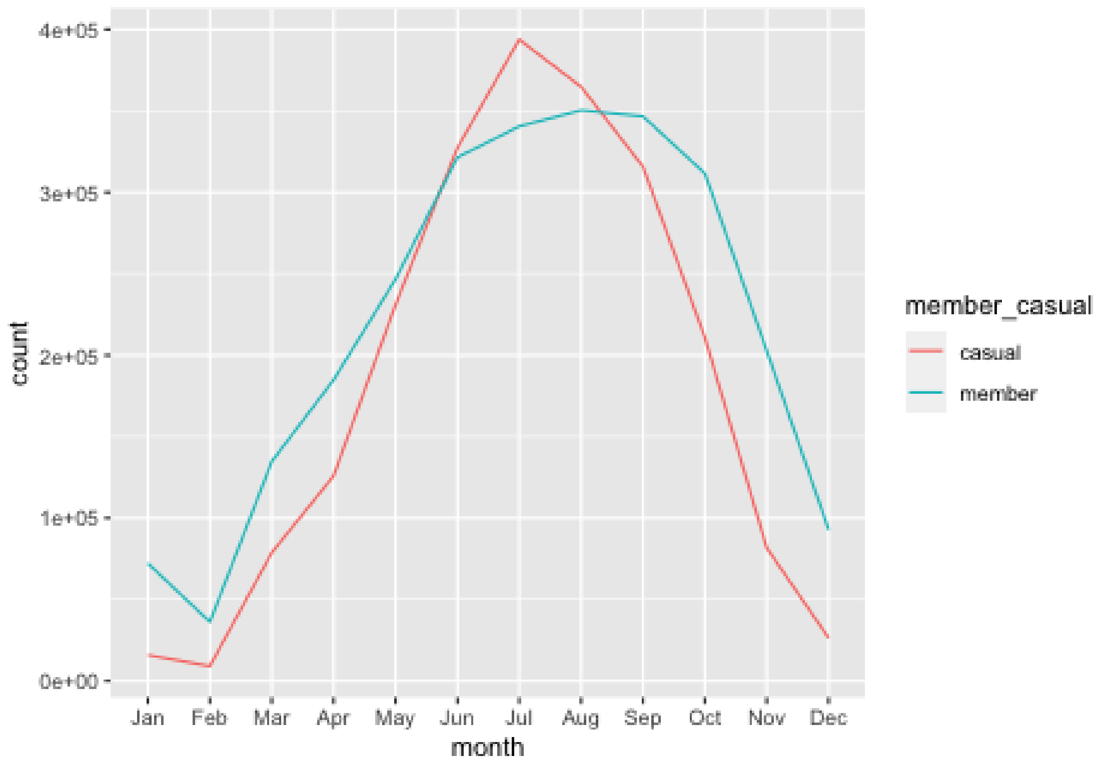


Table of number of rides cross type of bikes and membership over month

```
count_m <- divvy_data_clean %>%
        group_by(month, member_casual, rideable_type) %>%
        summarise(count = n()) %>%
        spread(month, count)
kable(count_m)
```

member_casual	rideable_type	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
casual	classic_bike	8237	5637	45416	70585	123602	187694	240883	229399	195025	105356	31764	11288
casual	docked_bike	2105	1271	15657	24713	43352	51715	57698	45065	35337	22689	7560	4935
casual	electric_bike	5552	2252	17680	30685	63857	87855	95264	90298	85329	82810	42371	10189
member	classic_bike	53358	29158	106955	143754	185012	246569	265253	272876	266615	210470	122114	59248
member	docked_bike	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	7772
member	electric_bike	18848	7040	27761	41135	61561	74887	75238	77446	80030	101138	80612	25933

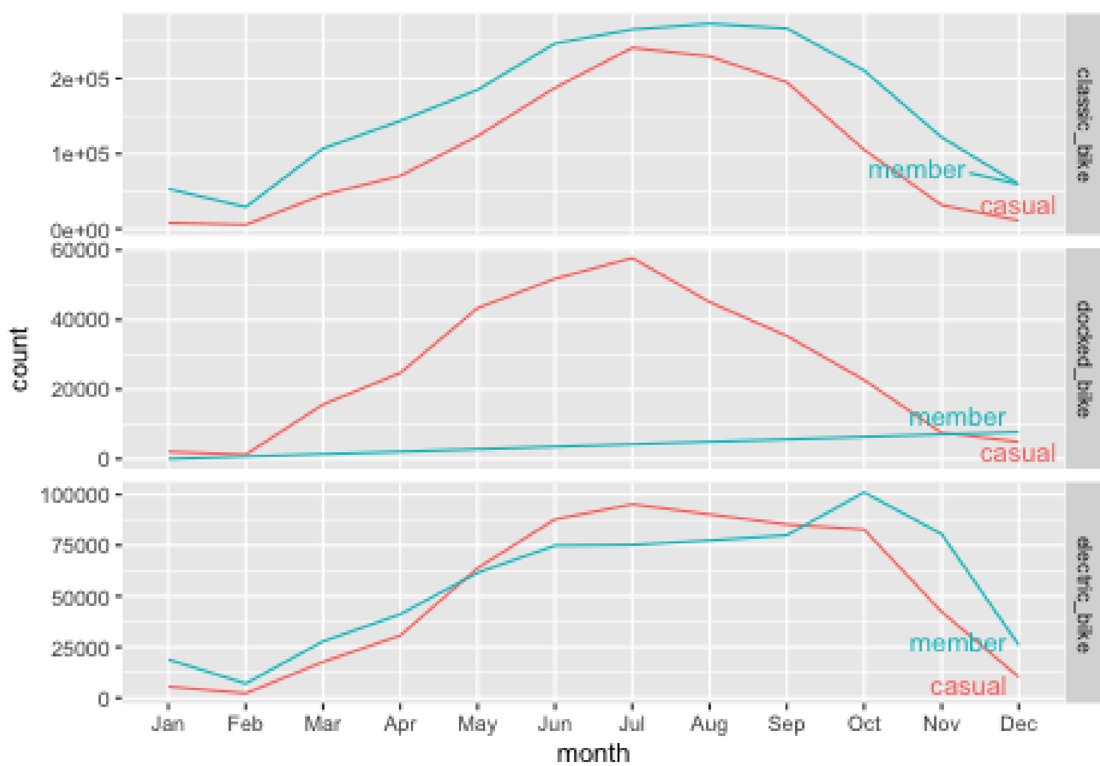
Table of median used time cross type of bikes and membership over month

```
median <- divvy_data_clean %>%
        group_by(month, member_casual, rideable_type) %>%
        summarise(median = median(used_time)) %>%
        spread(month, median)
kable(median)
```

member_casual	rideable_type	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
casual	classic_bike	12.82	16.800	18.05	17.35	18.52	17.13	16.18	15.73	15.23	14.43	12.42	14.05
casual	docked_bike	22.95	27.520	30.48	31.57	31.04	29.95	28.95	27.72	27.63	26.33	23.21	21.48
casual	electric_bike	10.15	12.015	13.82	13.73	15.23	14.70	14.23	13.93	13.28	11.87	10.02	11.15
member	classic_bike	8.95	10.450	10.25	10.65	10.85	10.88	10.55	10.20	9.85	9.02	8.23	9.27
member	docked_bike	2.63	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	8.83
member	electric_bike	8.22	9.220	9.30	9.83	10.00	9.97	9.95	9.77	9.50	8.52	7.58	8.55

Plot it on the graph. But if you dive deeper, you will notice that most of the gap is actually coming from docked bike category.

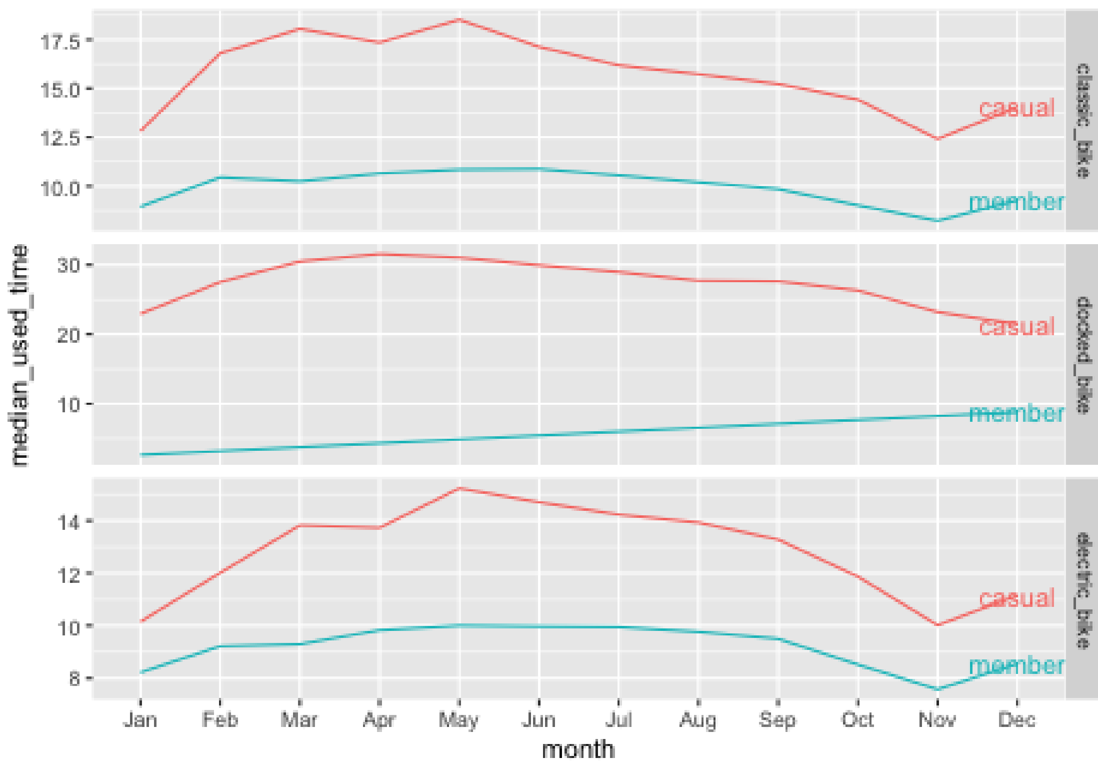
```
divvy_data_clean %>%
        group_by(month, member_casual, rideable_type) %>%
        summarise(count = n()) %>%
        mutate(label = ifelse(month == "Dec", as.character(member_casual), NA_character_)) %>%
        ggplot(aes(x = month, y = count, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text_repel(aes(label = label, x = "Dec", y = count), na.rm = T)+
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```



Given on median used time for different category, it's likely who used docked bike buy day pass service which provides unlimited 3-hour ride a day. For those we use docked bike, it might be ahrd to tell what they are using the bike for, but one thing can be for sure is they are coming from different group of people.

```
divvy_data_clean %>%
        group_by(member_casual, month, rideable_type) %>%
```

```
        summarise(median_used_time = median(used_time)) %>%
        mutate(label = ifelse(month == "Dec", as.character(member_casual), NA_character_)) %>%
        ggplot(aes(month, median_used_time, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text(aes(label = label, x = "Dec", y = median_used_time), na.rm = TRUE) +
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```



We dive in to know how have the bikes been used by member and casual riders in a day order by number of times they have been used from highest to lowest.

```
divvy_data_clean %>%
        filter(member_casual == "member") %>%
        group_by(hour, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

```
## # A tibble: 24 x 4
##    hour  member_casual  count percent
##    <chr> <fct>          <int> <chr>
##  1 18    member        238212 9%
##  2 16    member        224286 8%
##  3 15    member        172256 7%
##  4 08    member        151441 6%
##  5 12    member        153817 6%
##  6 13    member        150826 6%
```

```
##   7 14    member        149054 6%
##   8 19    member        168546 6%
##   9 07    member        132238 5%
##  10 11    member        132686 5%
## # ... with 14 more rows
```

```
divvy_data_clean %>%
        filter(member_casual == "casual") %>%
        group_by(hour, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

```
## # A tibble: 24 x 4
##    hour  member_casual  count percent
##    <chr> <fct>          <int> <chr>
##   1 18    casual        187774 9%
##   2 16    casual        178202 8%
##   3 13    casual        150884 7%
##   4 14    casual        155294 7%
##   5 15    casual        162710 7%
##   6 19    casual        143987 7%
##   7 12    casual        141496 6%
##   8 11    casual        118929 5%
##   9 20    casual        104754 5%
##  10 10    casual         90754 4%
## # ... with 14 more rows
```

We dive in to know how have the bikes been used by member and casual riders in a week ordered by number of times they have been used from highest to lowest.

```
divvy_data_clean %>%
        filter(member_casual == "member") %>%
        group_by(weekday, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

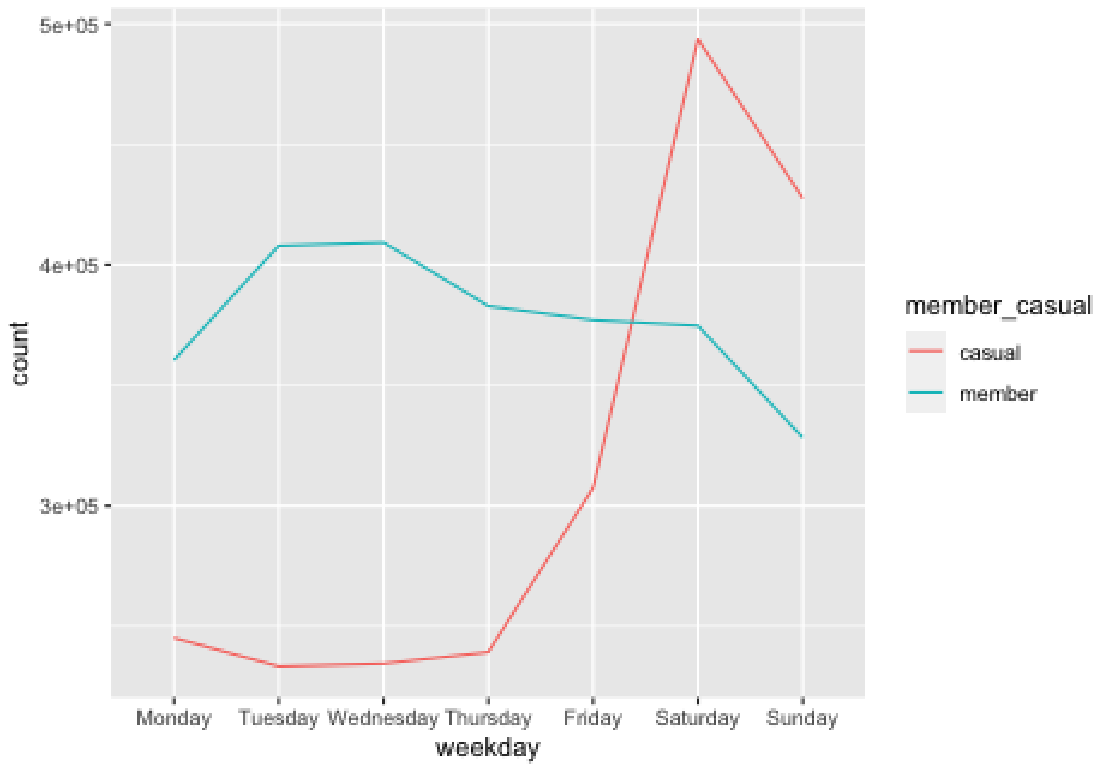
```
## # A tibble: 7 x 4
##   weekday   member_casual  count percent
##   <fct>     <fct>          <int> <chr>
## 1 Tuesday   member        407925 15%
## 2 Wednesday member        409300 15%
## 3 Thursday  member        382929 15%
## 4 Monday    member        360475 14%
## 5 Friday    member        377133 14%
## 6 Saturday  member        374743 14%
## 7 Sunday    member        328279 12%
```

```
divvy_data_clean %>%
        filter(member_casual == "casual") %>%
        group_by(weekday, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

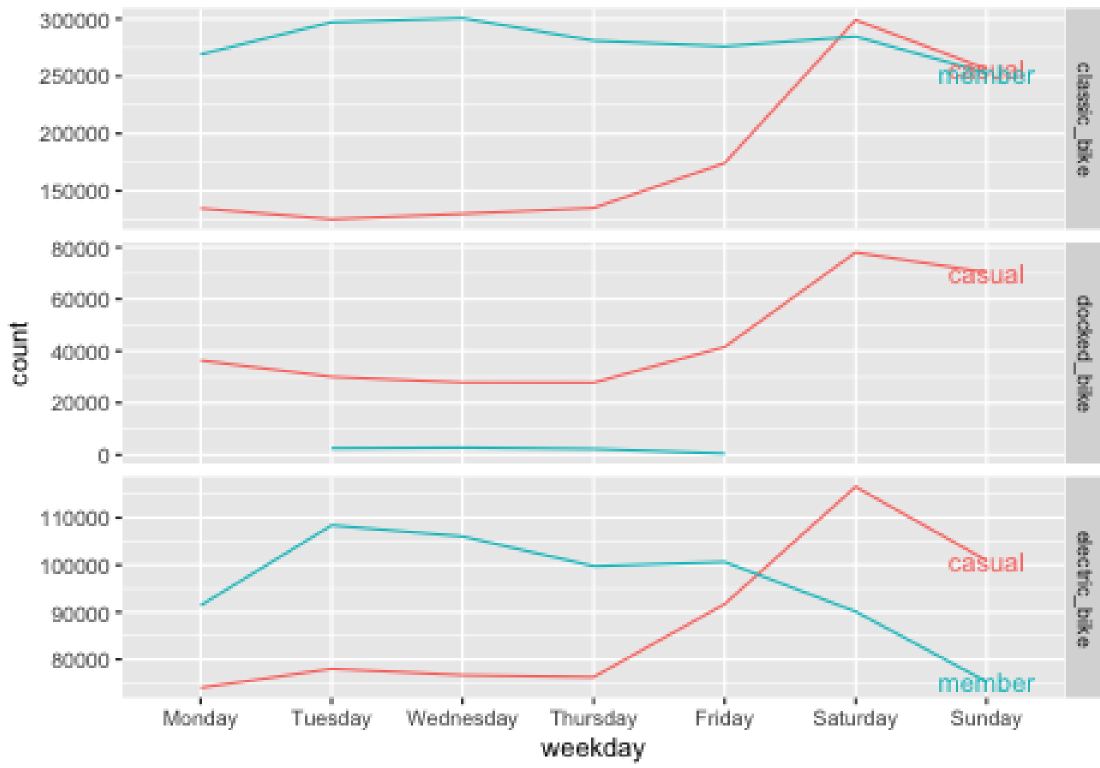
```
## # A tibble: 7 x 4
##   weekday   member_casual  count percent
##   <fct>     <fct>          <int> <chr>
## 1 Saturday  casual        493624 23%
## 2 Sunday    casual        427630 20%
## 3 Friday    casual        307535 14%
## 4 Monday    casual        245126 11%
## 5 Tuesday   casual        233413 11%
## 6 Wednesday casual        234532 11%
## 7 Thursday  casual        239265 11%
```

Let's plot it. From the result below, you can tell memberuser have consistent number of usages each day in a week. However, casualuser have remarkable number of usages on weekends(Saturday & Sunday).

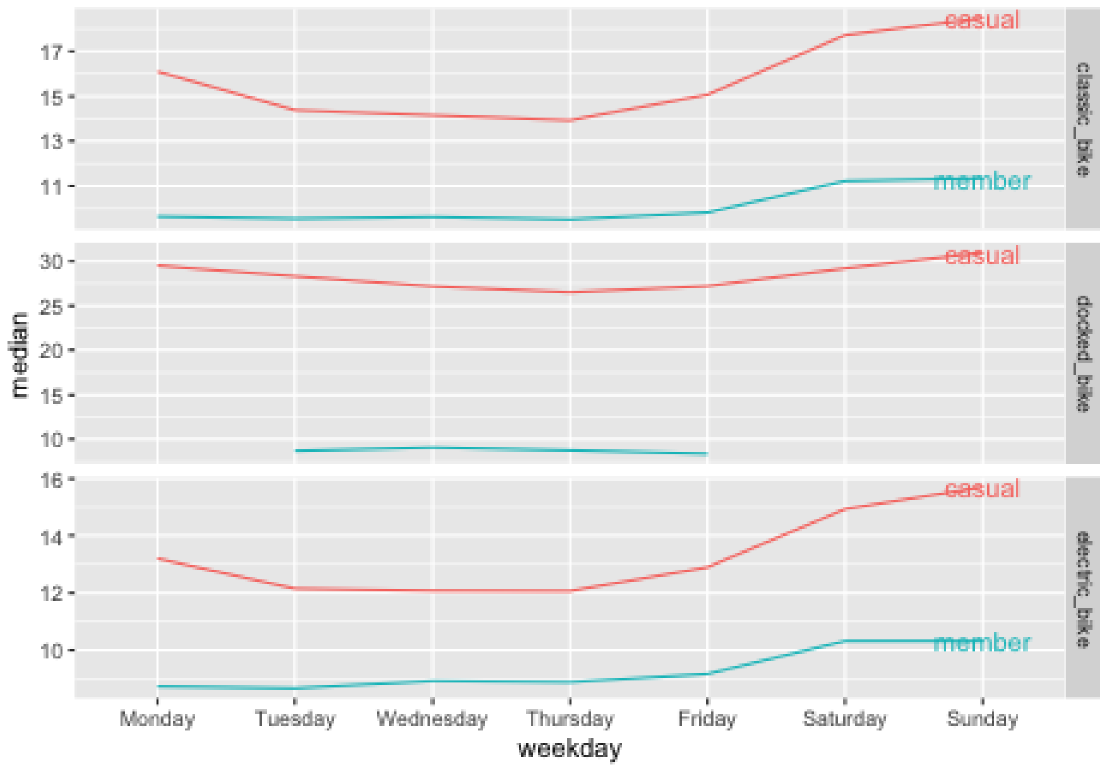
```
divvy_data_clean %>%
        group_by(weekday, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        ggplot(aes(weekday, count, group = member_casual, color = member_casual))+
        geom_line()
```




```
divvy_data_clean %>%
  group_by(weekday, member_casual, rideable_type) %>%
  summarise(count = n()) %>%
  mutate(label = ifelse(weekday == "Sunday", as.character(member_casual), NA_character_)) %>%
  ggplot(aes(weekday, count, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text(aes(label = label, x = "Sunday", y = count), na.rm = TRUE) +
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```




```
divvy_data_clean %>%
  group_by(weekday, member_casual, rideable_type) %>%
  summarise(median = median(used_time)) %>%
  mutate(label = ifelse(weekday == "Sunday", as.character(member_casual), NA_character_)) %>%
  ggplot(aes(weekday, median, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text(aes(label = label, x = "Sunday", y = median), na.rm = TRUE) +
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```



We dive in to know how have the bikes been used by member and casual riders in a day order by number of times they have been used from highest to lowest.

```
divvy_data_clean %>%
        filter(member_casual == "member") %>%
        group_by(hour, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

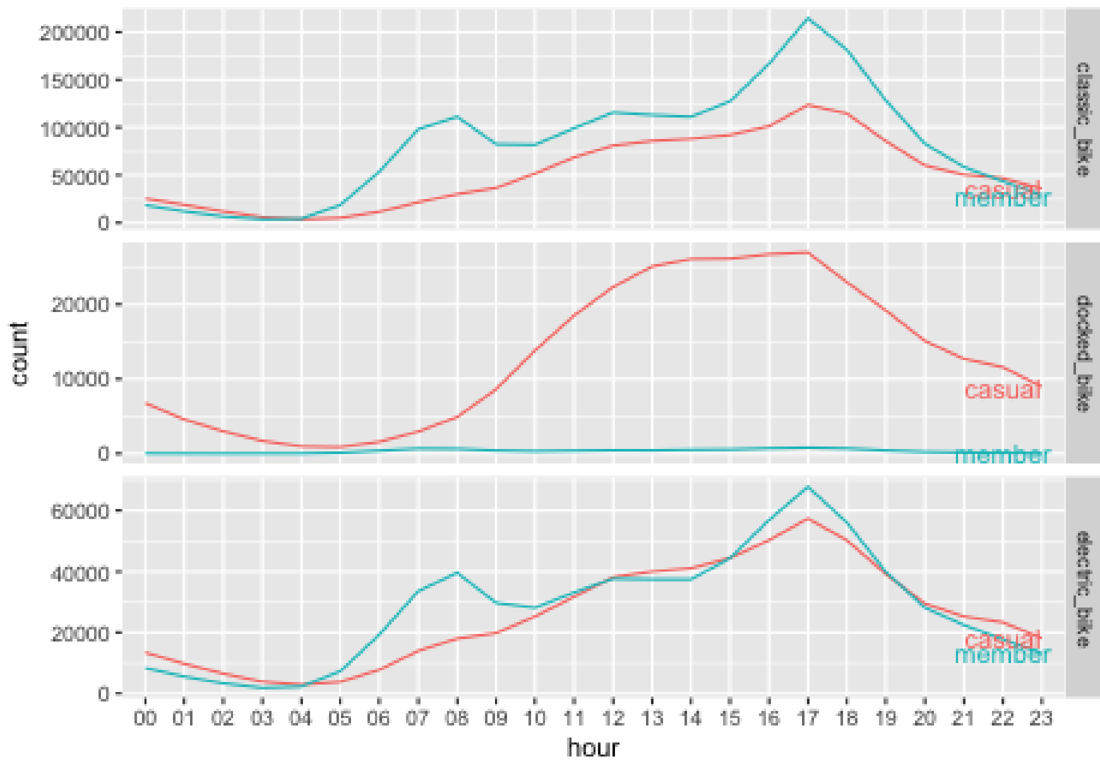
```
## # A tibble: 24 x 4
##    hour  member_casual  count percent
##    <chr> <fct>          <int> <chr>
##  1 18    member        238212 9%
##  2 16    member        224286 8%
##  3 15    member        172256 7%
##  4 08    member        151441 6%
##  5 12    member        153817 6%
##  6 13    member        150826 6%
##  7 14    member        149054 6%
##  8 19    member        168546 6%
##  9 07    member        132238 5%
## 10 11    member        132686 5%
## # ... with 14 more rows
```

```
divvy_data_clean %>%
        filter(member_casual == "casual") %>%
```

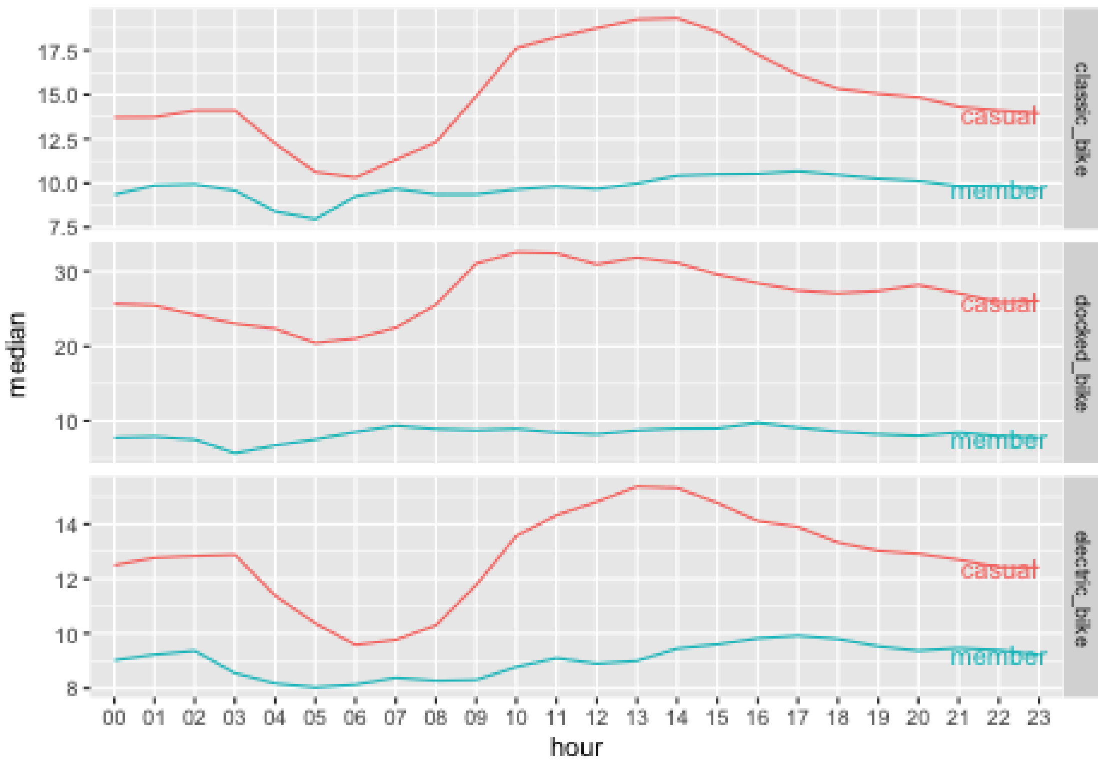
```
        group_by(hour, member_casual) %>%
        summarise(count = n()) %>%
        ungroup() %>%
        mutate(percent = paste0(round(count/sum(count),2)*100,"%")) %>%
        arrange(desc(percent))
```

```
## # A tibble: 24 x 4
##    hour  member_casual  count percent
##    <chr> <fct>          <int> <chr>
##  1 18    casual        187774 9%
##  2 16    casual        178202 8%
##  3 13    casual        150884 7%
##  4 14    casual        155294 7%
##  5 15    casual        162710 7%
##  6 19    casual        143987 7%
##  7 12    casual        141496 6%
##  8 11    casual        118929 5%
##  9 20    casual        104754 5%
## 10 10    casual         90754 4%
## # ... with 14 more rows
```

```
divvy_data_clean %>%
  group_by(hour, member_casual, rideable_type) %>%
  summarise(count = n()) %>%
  mutate(label = ifelse(hour == 23, as.character(member_casual), NA_character_)) %>%
  ggplot(aes(hour, count, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text(aes(label = label, x = 23, y = count), na.rm = TRUE) +
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```




```r
divvy_data_clean %>%
  group_by(hour, member_casual, rideable_type) %>%
  summarise(median = median(used_time)) %>%
  mutate(label = ifelse(hour == 23, as.character(member_casual), NA_character_)) %>%
  ggplot(aes(hour, median, group = member_casual, color = member_casual))+
        geom_line()+
        geom_text(aes(label = label, x = 23, y = median), na.rm = TRUE) +
        facet_grid(rideable_type~., scales = "free")+
        theme(legend.position = "none")
```



ACT

RECOMMENDATION

- As the analysis shows, there are someone who act like member users but haven't subscribed to it yet. Focus on this group as target customer, we can try to attract them with even 3-month-trial or larger dicount than existing members. In the meantime, we can try to figure out why they don't want to join the membership to validate our strategy.
- For those who use divvy frequently on weekends especially on docked bike, we can create a member package called "Weekends premium" to attract them join divvy. But since we have limited data to prove whether they are one time users or frequent users. We just assume that they are frequent users.