Using BigQuery to perform basic data analytics in R

Virgil

5/3/2022

This is a sample notebook of executing SQL commands in order to analyze and visualize some data, using some basic charts.

We'll explore a BigQuery public data set and reproduce how we might write queries for certain business problems.

Setup

Install relevant packages

Setup to connection to BigQuery

```
con <- dbConnect(
  bigquery(),
  project = "bigquery-public-data",
  dataset = "san_francisco_bikeshare",
  billing = "personal-portfolio-346209"
)
 con
```

```
## <BigQueryConnection>
##   Dataset: bigquery-public-data.san_francisco_bikeshare
##   Billing: personal-portfolio-346209
```

Queries

We'll be using the San Francisco Bikeshares dataset, which contains information around trips for the bikeshare program in San Francisco.

There's 4 tables in this dataset. First, we should look at the schema of all of these tables and see where we might be able to join for insight in future queries.

The tables are

- bikeshare_regions
- bikeshare_station_info
- bikeshare_station_status
- bikeshare_trips

```r
# bikeshare_regions

dbGetQuery(
   con, '
  select *
  FROM bikeshare_regions
  limit 10'
 )
```

```
## ! Using an auto-discovered, cached token.

##   To suppress this message, modify your code or options to clearly consent to
##   the use of a cached token.

##   See gargle's "Non-interactive auth" vignette for more details:

##   <https://gargle.r-lib.org/articles/non-interactive-auth.html>

## i The bigrquery package is using a cached token for 'virgiluwaoma@gmail.com'.

## # A tibble: 6 x 2
##   region_id name
##       <int> <chr>
## 1         3 San Francisco
## 2         5 San Jose
## 3        12 Oakland
## 4        13 Emeryville
## 5        14 Berkeley
## 6        23 8D
```

```r
# bikeshare_station_info
dbGetQuery(
  con, '
  select *
  FROM bikeshare_station_info
  limit 10'
)
```

```
## # A tibble: 10 x 12
##    station_id name      short_name   lat   lon region_id rental_methods capacity
##         <int> <chr>     <chr>      <dbl> <dbl>     <int> <chr>             <int>
##  1        574 West at ~ OK-H3-2     37.8 -122.        NA ['CREDITCARD'~        0
##  2        570 Howard S~ SF-K24-2    37.8 -122.        NA ['CREDITCARD'~       12
##  3        544 Allyne P~ SF-C21      37.8 -122.        NA ['CREDITCARD'~       19
##  4        546 13th St ~ OK-L6-2     37.8 -122.        NA ['CREDITCARD'~       19
##  5        547 North Po~ SF-A26      37.8 -122.        NA ['CREDITCARD'~       19
##  6        571 Irving S~ SF-M8       37.8 -122.        NA ['CREDITCARD'~       19
##  7        573 25th Ave~ SF-M7       37.8 -122.        NA ['CREDITCARD'~       19
##  8        501 Balboa P~ SF-Y16      37.7 -122.        NA ['CREDITCARD'~       23
##  9        568 Alemany ~ SF-Y18      37.7 -122.        NA ['CREDITCARD'~       23
## 10        562 8th St a~ SF-M27      37.8 -122.        NA ['CREDITCARD'~       25
## # ... with 4 more variables: external_id <chr>, eightd_has_key_dispenser <lgl>,
## #   has_kiosk <lgl>, station_geom <wk_wkt>
```

```r
# bikeshare_station_status
dbGetQuery(
  con, '
  select *
  from bikeshare_station_status
  limit 10'
)
```

```
## # A tibble: 10 x 11
##    station_id num_bikes_available num_bikes_disabled num_docks_available
##         <int>               <int>              <int>               <int>
##  1        263                   0                  0                   0
##  2        574                   0                  0                   0
##  3         55                   3                 24                   0
##  4        108                  11                  0                   0
##  5        386                  12                  0                   0
##  6        257                  14                  1                   0
##  7        279                  14                  1                   0
##  8        168                  15                  0                   0
##  9        218                  15                  0                   0
## 10        132                  17                  1                   0
## # ... with 7 more variables: num_docks_disabled <int>, is_installed <lgl>,
## #   is_renting <lgl>, is_returning <lgl>, last_reported <int>,
## #   num_ebikes_available <int>, eightd_has_available_keys <lgl>
```

```r
# bikeshare_station_status
dbGetQuery(
  con, '
  select *
  from bikeshare_trips
  limit 10'
)
```

```
## # A tibble: 10 x 21
##    trip_id   duration_sec start_date          start_station_na~ start_station_id
##    <chr>            <int> <dttm>              <chr>                        <int>
##  1 38762018~          271 2018-04-30 21:21:25 24th St at Chatt~              132
##  2 30892018~          406 2018-04-30 21:12:51 Downtown Berkele~              245
##  3 40702018~         2449 2018-04-30 20:38:02 Valencia St at 2~              127
##  4 34302018~          316 2018-04-30 20:32:39 Berkeley Civic C~              246
##  5 12332018~          392 2018-04-30 20:22:38 Bay Pl at Vernon~              195
##  6 23372018~          198 2018-04-30 20:06:36 Downtown Berkele~              245
##  7 20482018~          914 2018-04-30 19:52:13 Jersey St at Chu~              138
##  8 31262018~          488 2018-04-30 19:25:56 Grand Ave at Web~              181
##  9 28620180~         1827 2018-04-30 19:00:44 Valencia St at 2~              134
## 10 38232018~          639 2018-04-30 19:17:29 Cyril Magnin St ~                4
## # ... with 16 more variables: end_date <dttm>, end_station_name <chr>,
## #   end_station_id <int>, bike_number <int>, zip_code <chr>,
## #   subscriber_type <chr>, c_subscription_type <chr>,
## #   start_station_latitude <dbl>, start_station_longitude <dbl>,
## #   end_station_latitude <dbl>, end_station_longitude <dbl>,
## #   member_birth_year <int>, member_gender <chr>,
## #   bike_share_for_all_trip <chr>, start_station_geom <wk_wkt>, ...
```

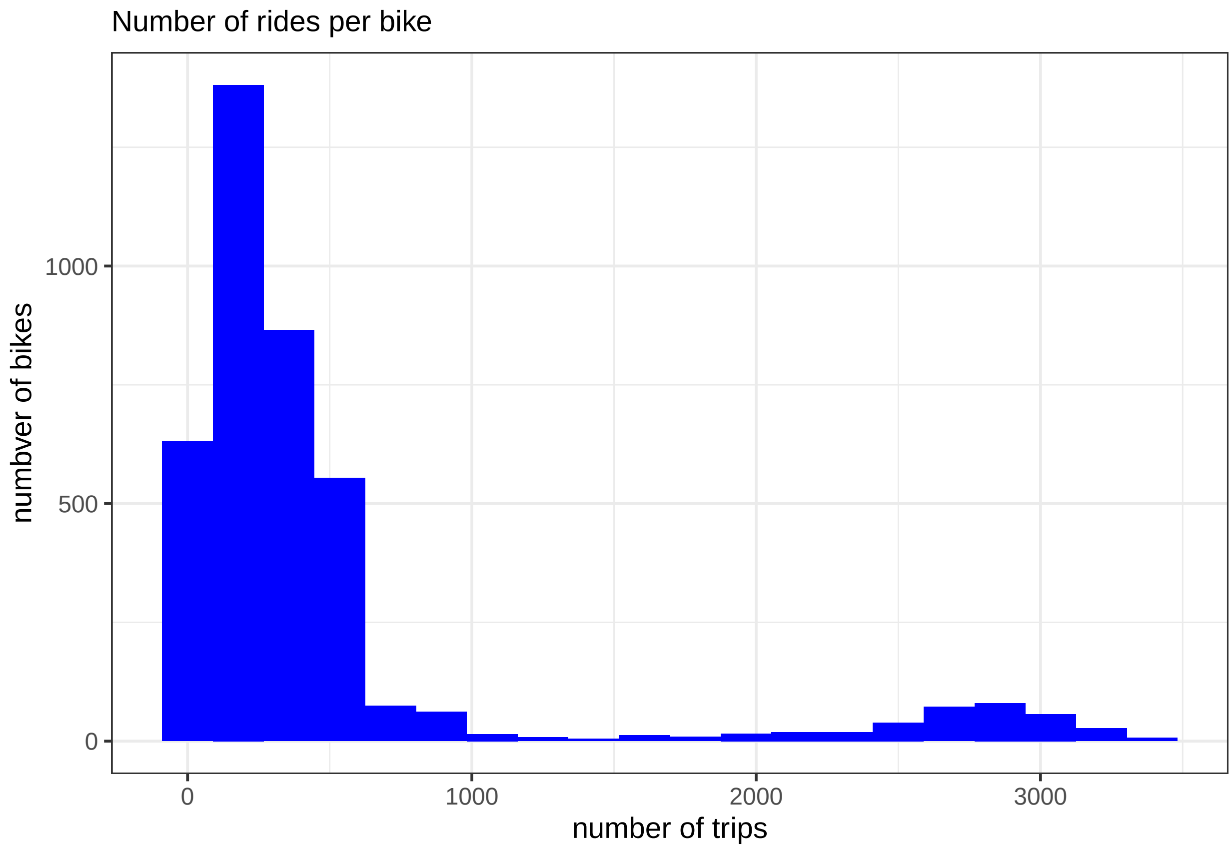
Looking at the schemas, we see that each table gives us some different information. A few things to note:

- We get some interesting information from the station_info table regarding payment types. It may be interesting to look if stations with different payment types are associated with more or less rides.
- The bikeshare_trips table will give us information around ride-by-ride stats and has unique identifiers around customers/members that use them.
  - There is additional information for members, but not for customers
  - This will allow us to take a look at where popular routes might be

At this point, we can start looking at doing some queries for exploratiion, and see where we might be able to answer questions with real business impact.

Let's start out by looking how many rides each of the bikes in our dataset have on them. This might give us an idea how much wear and tear these bikes have.

```
# q1
# Which bikes have been used the most?
query1 <- dbGetQuery(
  con, '
  select count(trip_id) as num_trips, bike_number
  from bikeshare_trips
  group by bike_number
  order by num_trips desc
  '
)

ggplot(query1, aes(x = num_trips)) +
  geom_histogram(bins = 20, fill = "blue") +
  labs(x = "number of trips",
       y = "numbver of bikes",
       subtitle = "Number of rides per bike") +
  theme_bw()
```

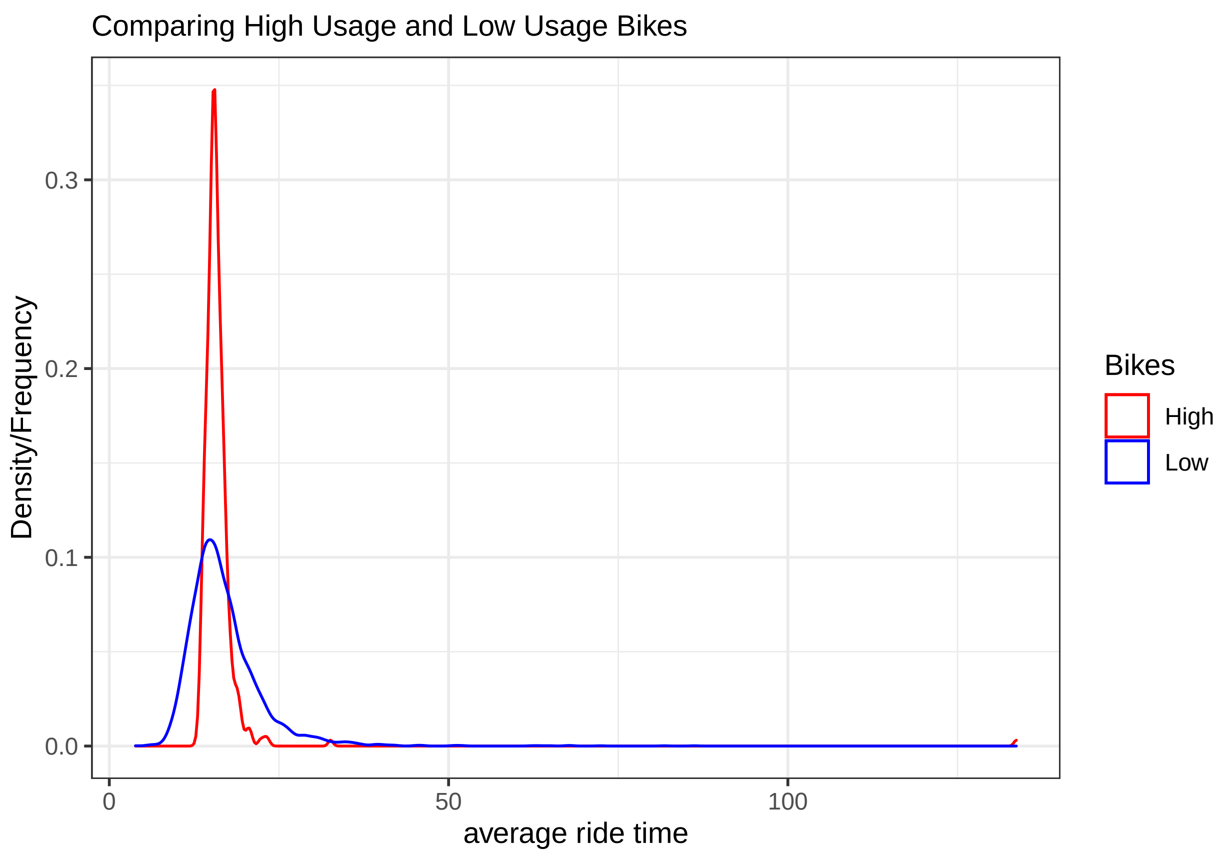


We see that the distribution is not normal, and it looks like there are two fundamental groups. We have one group of bikes that is used less than about 1000 times, and another normal-ish looking distribution centered around 2750. It might be interesting to look at the differences between these two groups of bikes- maybe they tend to be found on different routes? Maybe they have less miles on them, but are used more frequently?

Let's start off by looking at the differences between the average ride time between the "many-rides" group and the "few-rides" group.

```
# q2
# compare average ride times for bikes above/below 1500 bikes
# high rides query
query2_a <- dbGetQuery(
  con, '
  select avg(duration_sec)/60 as avg_trip_length,
  count(trip_id) as num_trips,bike_number
  from bikeshare_trips
  group by bike_number
  having num_trips >= 1500'
)

# low rides query
query2_b<- dbGetQuery(
  con, '
  select avg(duration_sec)/60 as avg_trip_length,
  count(trip_id) as num_trips, bike_number
  from bikeshare_trips
  group by bike_number
```

```
  having num_trips < 1500'
)

# Add a colummn to both data frames to help with visualization
query2_a <- query2_a %>% mutate(Usage="High")
query2_b <- query2_b %>% mutate(Usage="Low")

# Plot
ggplot() +
  geom_density(query2_a, mapping = aes(avg_trip_length, color = Usage) ) +
  geom_density(query2_b, mapping = aes(avg_trip_length, color = Usage) ) +
  labs(
    x = "average ride time",
    y = "Density/Frequency",
    subtitle = "Comparing High Usage and Low Usage Bikes",
  ) +
  theme_bw() +
  scale_color_manual(name= "Bikes",
                     breaks=c('High', 'Low'),
                     values=c('High'='red', 'Low'='blue'))
```



So we can see that our bikes with lower number of trips have a higher variance around the average trip length, whereas the high usage bikes have much lower variance. This gives some evidence to our theory that perhaps the higher usage bikes are going on certain high-traffic routes; that are about 17 minutes.

Note that a density plot instead of comparing histograms. From a data visualization perspective, we want these plotted on the same axes to make this comparison easy to make. Doing overlapping histograms can get cluttered, so we opt instead for the density plot which comes across much cleaner with the same kind of takeaway as the histogram.

Interestingly, the mean of both of these appear to be the same. We'll calculate some basic statics below to confirm.

```
# high usage
stat.desc(query2_a)
```

```
##              avg_trip_length    num_trips  bike_number Usage
## nbr.val          358.0000000 3.580000e+02 3.580000e+02    NA
## nbr.null           0.0000000 0.000000e+00 0.000000e+00    NA
## nbr.na             0.0000000 0.000000e+00 0.000000e+00    NA
## min               13.0698681 1.517000e+03 1.600000e+01    NA
## max              133.6713344 3.394000e+03 8.780000e+02    NA
## range            120.6014663 1.877000e+03 8.620000e+02    NA
## sum             5789.4148535 9.543740e+05 1.570580e+05    NA
## median            15.5456871 2.754000e+03 4.475000e+02    NA
## mean              16.1715499 2.665849e+03 4.387095e+02    NA
## SE.mean            0.3413612 2.161656e+01 7.362189e+00    NA
## CI.mean.0.95       0.6713316 4.251181e+01 1.447871e+01    NA
## var               41.7168389 1.672847e+05 1.940425e+04    NA
## std.dev            6.4588574 4.090046e+02 1.392991e+02    NA
## coef.var           0.3993963 1.534237e-01 3.175202e-01    NA
```

```
# low usage
stat.desc(query2_b)
```

```
##              avg_trip_length    num_trips  bike_number Usage
## nbr.val         3.594000e+03 3.594000e+03 3.594000e+03    NA
## nbr.null        0.000000e+00 0.000000e+00 0.000000e+00    NA
## nbr.na          0.000000e+00 0.000000e+00 0.000000e+00    NA
## min             3.858333e+00 1.000000e+00 9.000000e+00    NA
## max             8.621191e+01 1.471000e+03 4.073000e+03    NA
## range           8.235358e+01 1.470000e+03 4.064000e+03    NA
## sum             6.078550e+04 9.930450e+05 7.736717e+06    NA
## median          1.583771e+01 2.305000e+02 2.175500e+03    NA
## mean            1.691305e+01 2.763063e+02 2.152676e+03    NA
## SE.mean         9.330888e-02 3.420129e+00 1.818306e+01    NA
## CI.mean.0.95    1.829437e-01 6.705588e+00 3.565016e+01    NA
## var             3.129133e+01 4.204003e+04 1.188262e+06    NA
## std.dev         5.593865e+00 2.050367e+02 1.090074e+03    NA
## coef.var        3.307425e-01 7.420628e-01 5.063810e-01    NA
```

A few more things to note…

- We have a lot more bikes in the low usage group compared to high usage group, by about 9x
- The means are pretty close, but the standard deviations are less similar. The higher usage has a higher variance but lower mean.
- We could run a t-test to see if the means are equal, but with such large sample sizes we will likely come to the conclusion that they are different

Let's take a look into the top 25 routes used for each group and see if this explains the differences.

```
# routes for high usage
# q3

query3a <- dbGetQuery(
  con, '
  select sum(num_trips) as trips, start_station_id,
         end_station_id, concat(start_station_id, "_", end_station_id) as route_code

  from (select count(trip_id) as num_trips, start_station_id, end_station_id, bike_number
        from bikeshare_trips
        where bike_number in (select bike_number
                              from (select count(trip_id) as num_trips, bike_number
                                    from bikeshare_trips
                                    group by bike_number
                                    having num_trips >= 1500)
                              )
        group by start_station_id, end_station_id, bike_number
        )

  group by start_station_id, end_station_id
  order by trips desc
  limit 25
  '
)

query3a
```

```
## # A tibble: 25 x 4
##    trips start_station_id end_station_id route_code
##    <int>            <int>          <int> <chr>
##  1  8749               50             60 50_60
##  2  8168               69             65 69_65
##  3  7281               61             50 61_50
##  4  6601               50             61 50_61
##  5  6568               65             69 65_69
##  6  6557               60             74 60_74
##  7  6065               51             70 51_70
##  8  5930               70             50 70_50
##  9  5790               74             61 74_61
## 10  5714               74             70 74_70
## # ... with 15 more rows
```

```
# routes for low usage
# q3
query3b <- dbGetQuery(
  con, '
  select sum(num_trips) as trips, start_station_id,
         end_station_id, concat(start_station_id, "_", end_station_id) as route_code,

  from (select count(trip_id) as num_trips, start_station_id, end_station_id, bike_number
        from bikeshare_trips
```

```
        where bike_number in (select bike_number
                              from (select count(trip_id) as num_trips, bike_number
                                    from bikeshare_trips
                                    group by bike_number
                                    having num_trips < 1500)
                              )
        group by start_station_id, end_station_id, bike_number
        )

  group by start_station_id, end_station_id
  order by trips desc
  limit 25
  '
)

query3b
```

```
## # A tibble: 25 x 4
##    trips start_station_id end_station_id route_code
##    <int>            <int>          <int> <chr>
##  1  4930               15              6 15_6
##  2  3758               28             27 28_27
##  3  3444               27             28 27_28
##  4  3129                4              2 4_2
##  5  3096                2              4 2_4
##  6  2872                6             16 6_16
##  7  2716               81             15 81_15
##  8  2469               32             28 32_28
##  9  2468                6             15 6_15
## 10  2277               15             81 15_81
## # ... with 15 more rows
```

Let's see if there are any common routes between the high usage and low usage bikes

```
high_vol <- query3a

high_vol <- high_vol %>%
  mutate(`in_low?` = route_code %in% query3b$route_code)

high_vol
```

```
## # A tibble: 25 x 5
##    trips start_station_id end_station_id route_code `in_low?`
##    <int>            <int>          <int> <chr>      <lgl>
##  1  8749               50             60 50_60      FALSE
##  2  8168               69             65 69_65      FALSE
##  3  7281               61             50 61_50      FALSE
##  4  6601               50             61 50_61      FALSE
##  5  6568               65             69 65_69      FALSE
##  6  6557               60             74 60_74      FALSE
##  7  6065               51             70 51_70      FALSE
##  8  5930               70             50 70_50      FALSE
```

```
## 9  5790                 74              61 74_61      FALSE
## 10  5714                 74              70 74_70      FALSE
## # ... with 15 more rows
```

Interestingly, none of the top 25 routes for the high volume bikes are in the top 25 low volume bike routes. Although we could drill a bit deeper into this, with an initial analysis we see evidence that bikes that have the most rides are going on different routes for the higher volume bikes and lower volume bikes. This might be an interesting business result if the company is experiencing unequal wear and tear on the bikes - perhaps bikes from the lower volume routes could be moved to the higher ones and vice versa for more equal wear.

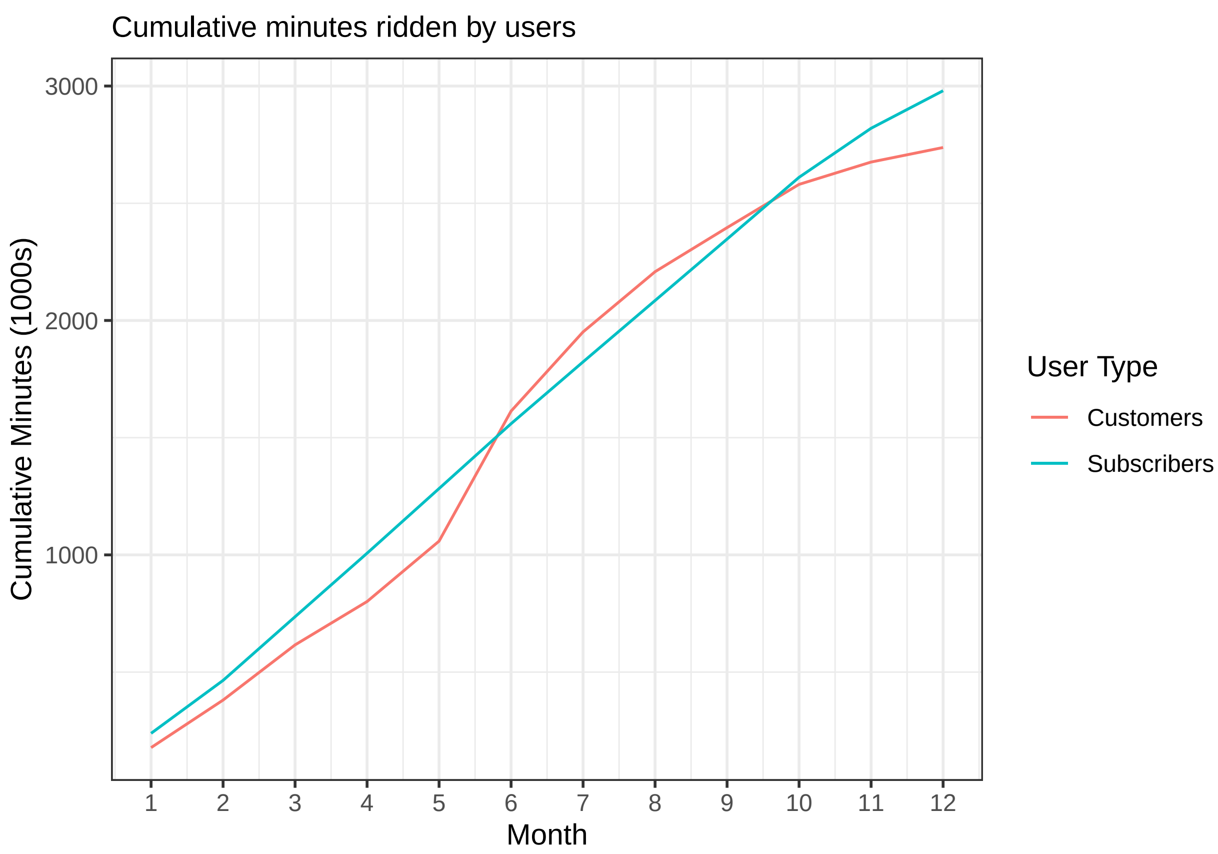
Let's pivot a bit to look at some customers vs subscribers behaviours.

I want to take a look at the cumulative minutes spent on bike rides by our subscribers vs customers on a month-by-month basis for 2015 (only year we have full data).

```r
query4 <- dbGetQuery(
  con, '
  select sum(customer_minutes_sum) over (order by end_month rows unbounded preceding)/1000 as cumulative
          sum(subscriber_minutes_sum) over (order by end_month rows unbounded preceding)/1000 as cumulat
          end_year,
          end_month

  from (select sum (case when subscriber_type = "Customer" then duration_sec/60 else null end) as custor
               sum (case when subscriber_type = "Subscriber" then duration_sec/60 else null end) as subs
               extract(year from end_date) as end_year,
               extract(month from end_date) as end_month
        from bikeshare_trips
        group by end_year, end_month
        having end_year = 2015
        )

  order by end_year, end_month
  '
)

#Plot
query4
```

```
## # A tibble: 12 x 4
##    cumulative_minutes_cust cumulative_minutes_sub end_year end_month
##                      <dbl>                  <dbl>    <int>     <int>
##  1                    178.                   239.     2015         1
##  2                    381.                   465.     2015         2
##  3                    616.                   736.     2015         3
##  4                    801.                  1007.     2015         4
##  5                   1059.                  1283.     2015         5
##  6                   1613.                  1560.     2015         6
##  7                   1951.                  1823.     2015         7
##  8                   2208.                  2085.     2015         8
##  9                   2396.                  2347.     2015         9
## 10                   2581.                  2611.     2015        10
## 11                   2676.                  2820.     2015        11
## 12                   2738.                  2980.     2015        12
```

```
ggplot(query4, mapping = aes(x = end_month)) +
  geom_line(mapping = aes(y = cumulative_minutes_cust, color = "Customers"))+
  geom_line(mapping = aes(y = cumulative_minutes_sub, color= "Subscribers"))+
  labs(
    x = "Month",
    y = "Cumulative Minutes (1000s)",
    color = "User Type",
    subtitle = "Cumulative minutes ridden by users"
  ) +
  scale_x_continuous(breaks = seq(1, 12, by = 1))+
  theme_bw()
```

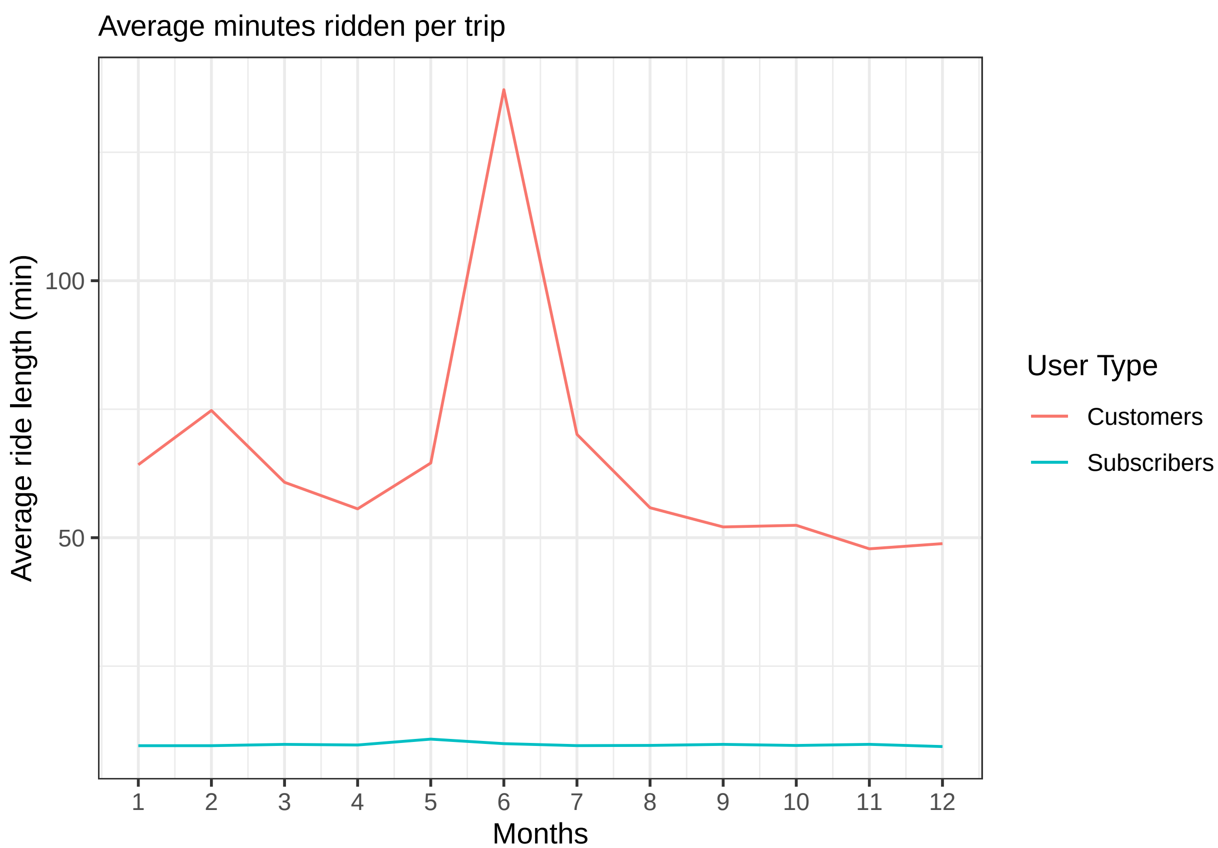


We see something interesting here. Subscribers, people that pay for longer-term memberships, are using the bikes at a decently consistent rate throughout the year. The customers, people that don't intend to use the bikes very often, really use them a lot more in the summer months, months 6 - 8. Overall, the subscribers will spend more time on the bikes over the year with their relatively more consistent usage.

Let's change the last query slightly and look how the average ride length changes over months.

```
query5 <- dbGetQuery(
  con, '
  select avg(case when subscriber_type = "Customer" then duration_sec/60 else null end) as customer_minu
         avg(case when subscriber_type = "Subscriber" then duration_sec/60 else null end ) as subscribe
         extract(year from end_date) as end_year,
         extract(month from end_date) as end_month,
```

```
  from bikeshare_trips

  group by end_year, end_month
  having end_year = 2015
  order by end_year, end_month
  '
)

# Plot
ggplot(query5, mapping = aes(x = end_month)) +
  geom_line(mapping = aes(y = customer_minutes_avg, color = "Customers")) +
  geom_line(mapping = aes(y = subscriber_minutes_avg, color = "Subscribers"))+
  labs(
    x="Months",
    y="Average ride length (min)",
    color = "User Type",
    subtitle = "Average minutes ridden per trip"
  ) +
  theme_bw()+
  scale_x_continuous(breaks = seq(1, 12, by = 1))
```



The above graph very clearly shows the phenomenon that we showed with the other query- the average ride length skyrockets over the summer as presumably more casual customers find good weather to take longer bike rides. The subscribers are likely commuting for the most part, or at least keeping their habits very consistent. We also note that the average ride length is much longer for the customers regardless of the spike, indicating again that customers might use the bikes for leisure purposes much more than subscribers.

Given that the average is so much lower and the previous chart looks the way it did, we can infer that the volume of subscriber rides to customer rides is many times higher.

We'll take a look at the origin stations popular with customers and subscribers and see if the capacities seem different for each. To do this we utilize data from multiple tables.

```
# q6
query6_sub <- dbGetQuery(
  con, '
  select sum(case when trips.subscriber_type = "Subscriber" then trips.trip else null end) as sub_trips
         info.station_id as station,
         info.capacity as cap

  from bikeshare_station_info as info
       inner join (select start_station_id, subscriber_type, count(trip_id) as trip
                   from bikeshare_trips
                   group by start_station_id, subscriber_type) as trips
       on info.station_id = trips.start_station_id

  group by station, cap
  order by sub_trips desc
  limit 25
  '
)

head(query6_sub)
```

```
## # A tibble: 6 x 3
##   sub_trips station   cap
##       <int>   <int> <int>
## 1     73053      70    31
## 2     53694      69    31
## 3     42909      50    39
## 4     39733      61    27
## 5     39537      55    27
## 6     38271      74    27
```

```
query6_cust <- dbGetQuery(
  con, '
  select sum(case when trips.subscriber_type = "Customer" then trips.trip else null end) as sub_trips,
         info.station_id as station,
         info.capacity as cap

  from bikeshare_station_info as info
       inner join (select start_station_id, subscriber_type, count(trip_id) as trip
                   from bikeshare_trips
                   group by start_station_id, subscriber_type) as trips
       on info.station_id = trips.start_station_id

  group by station, cap
  order by sub_trips desc
  limit 25
  '
)
```

```r
head(query6_cust)
```

```
## # A tibble: 6 x 3
##   sub_trips station   cap
##       <int>   <int> <int>
## 1     14831      60    31
## 2     13661      50    39
## 3      9563       6    23
## 4      8013      15    38
## 5      7317      70    31
## 6      6774      76    19
```

Right away, we can see that the most frequent stations to start a trip for both subscribers and customers include station 70, indicating this must be an area that a lot of people go to in general.

We'll take a look at the mean and standard deviation around each capacity in the 25 stations for both sides.

```r
mean_cust <- mean(query6_cust$cap)
sd_cust <- sd(query6_cust$cap)
mean_sub <- mean(query6_sub$cap)
sd_sub <- sd(query6_sub$cap)
```

```r
print(paste("Mean of top 25 customer stations capacity:", mean_cust))
```

```
## [1] "Mean of top 25 customer stations capacity: 28.24"
```

```r
print(paste("Mean of top 25 subscriber stations capacity:", mean_sub))
```

```
## [1] "Mean of top 25 subscriber stations capacity: 28.08"
```

```r
print(paste("Standard Deviation of top 25 customer stations capacity:", sd_cust))
```

```
## [1] "Standard Deviation of top 25 customer stations capacity: 6.54013251649638"
```

```r
print(paste("Standard Deviation of top 25 subscriber stations capacity:", sd_sub))
```

```
## [1] "Standard Deviation of top 25 subscriber stations capacity: 5.91551632009695"
```

We don't really see a big difference here. We might look into doing some kind of hypothesis test in the future to dig into this in the future, but we can leave the analysis here for now.