Lyft Data Contest

Emeka Mbazor

9/14/2019

```r
# importing libraries
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang

## -- Attaching packages --------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr)
library(ggplot2)
library(broom)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library(modelr)
```

```
##
## Attaching package: 'modelr'

## The following object is masked from 'package:broom':
##
##     bootstrap
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
# plot settings
theme_set(theme_bw())
options(repr.plot.width=4, repr.plot.height=3)

# loading in datasets
driver <- read.csv("driver_ids.csv")
ride <- read.csv("ride_ids.csv")
ride_stamps <- read.csv("ride_timestamps.csv")
```

```r
str(driver$driver_onboard_date)   # 49 factors
```

```
##  Factor w/ 49 levels "2016-03-28 00:00:00",..: 2 2 9 27 33 40 11 41 30 40 ...
```

```r
str(ride_stamps$timestamp)        # 865827 factors
```

```
##  Factor w/ 865827 levels "","2016-03-28 05:48:18",..: 702756 702760 702800 702801 703012 335582 3355
```

```r
# factors as a data type is not particularly helpful...

# converting driver onboard dates from factors to dates
driver <- driver %>%
  mutate(driver_onboard_date2 = as.Date(as.character(driver_onboard_date)))

# converting ride timestamps from factors to dates
ride_stamps <- ride_stamps %>%
  mutate(timestamp2 = as.Date(as.character(timestamp)))

# converting ride timestamps from factors to date-times
ride_stamps <- ride_stamps %>%
  mutate(timestamp3 = ymd_hms(as.character(timestamp)))

# converting date-times to times,
# hms::as.hms doesn't account for timezones,
# leading to a loss of 4 hours
# 4 hours is added back in after the conversion
four_hours <- 60 * 60 * 4 # four hours in seconds
ride_stamps <- ride_stamps %>%
  mutate(timestamp4 = timestamp3 + four_hours) %>%
  mutate(times = hms::as.hms(timestamp4))
```

```r
# joining all datasets together
ride_info <- left_join(driver,ride, by = "driver_id")
```

```
## Warning: Column `driver_id` joining factors with different levels, coercing
## to character vector
```

```r
ride_info <- left_join(ride_info,ride_stamps, by = "ride_id")
```

```
## Warning: Column `ride_id` joining factors with different levels, coercing
```

```
## to character vector
```

```r
base_fare <- 2
cost_per_mile <- 1.15
cost_per_min <- 0.22
service_fee <- 1.75

# calculating the revenue gained per trip with the equation:
# ((base_fare + (cost_per_mile * ride_distance_miles) + (cost_per_min * ride_duration_min)) * (1 + pri
ride_info_enhanced <- ride_info %>%
  mutate(ride_duration_min = ride_duration / 60.0) %>%
  mutate(ride_distance_miles = ride_distance / 1609.34) %>%
  mutate(prime_decimal = ride_prime_time / 100) %>%
  mutate(price_per_trip = ((base_fare + (cost_per_mile * ride_distance_miles) + (cost_per_min * ride_du

# rides are always at least 5 dollars and are
# capped at 400 dollars
ride_info_enhanced <- ride_info_enhanced %>%
  mutate(price_per_trip_adj = case_when(
    (price_per_trip > 400) ~ 400,
    (price_per_trip < 5) ~ 5,
    TRUE ~ price_per_trip
  ))

# price per trip info
summary(ride_info_enhanced$price_per_trip_adj)
```

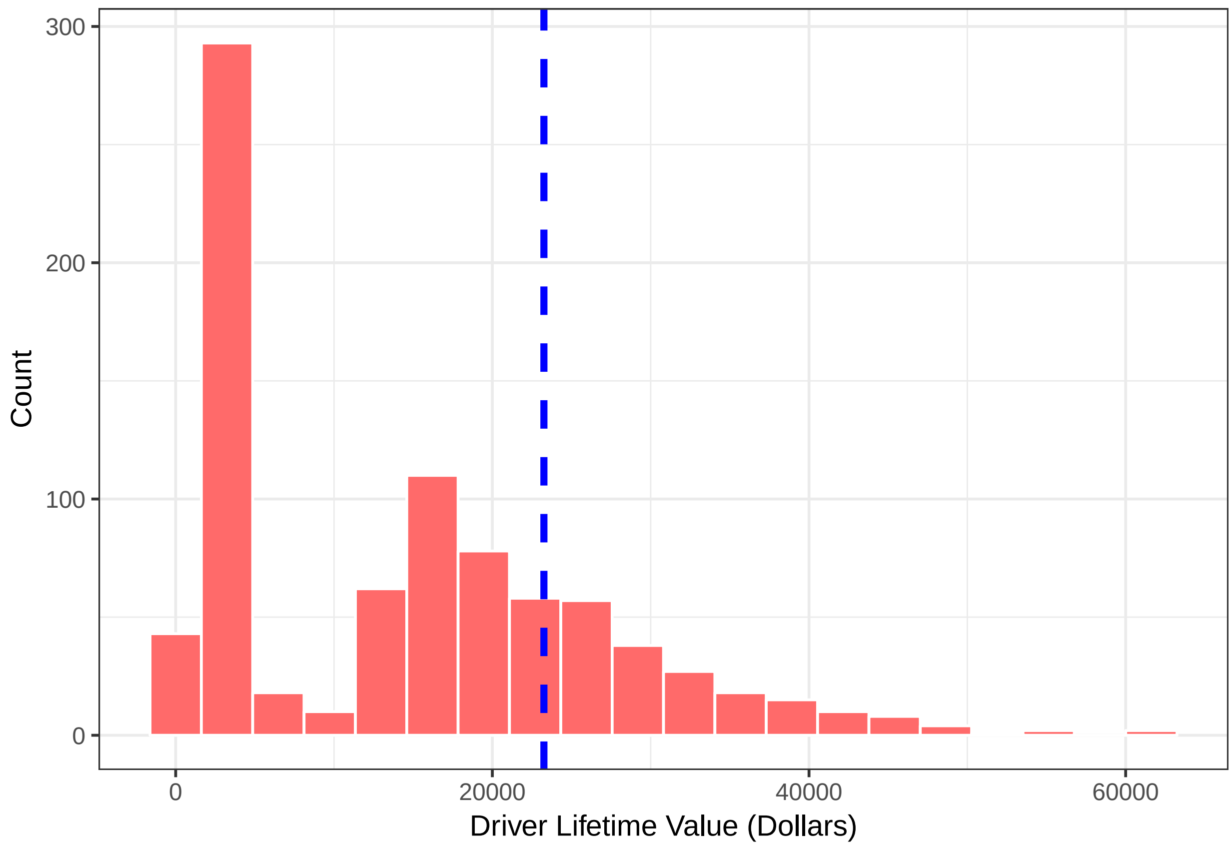
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   5.000   8.037  10.568  13.538  15.111 400.000      83
```

```r
  #  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # 5.000   8.037  10.568  13.538  15.111 400.000      83

# grouped by driver and summed up ride prices to get
# each rider's revenue
ride_info_drivers <- ride_info_enhanced %>%
  group_by(driver_id) %>%
  mutate(driver_revenue = sum(price_per_trip_adj)) %>%
  filter(!(is.na(driver_revenue))) %>%
  ungroup()
# NOTE: We're calculating revenue generated by the drivers but
#       it should still be directly proportional to Lyft's revenue.

# driver revenue info

summary(ride_info_drivers$driver_revenue)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   128.5 16761.5 23256.6 24502.8 31698.1 61751.4
```

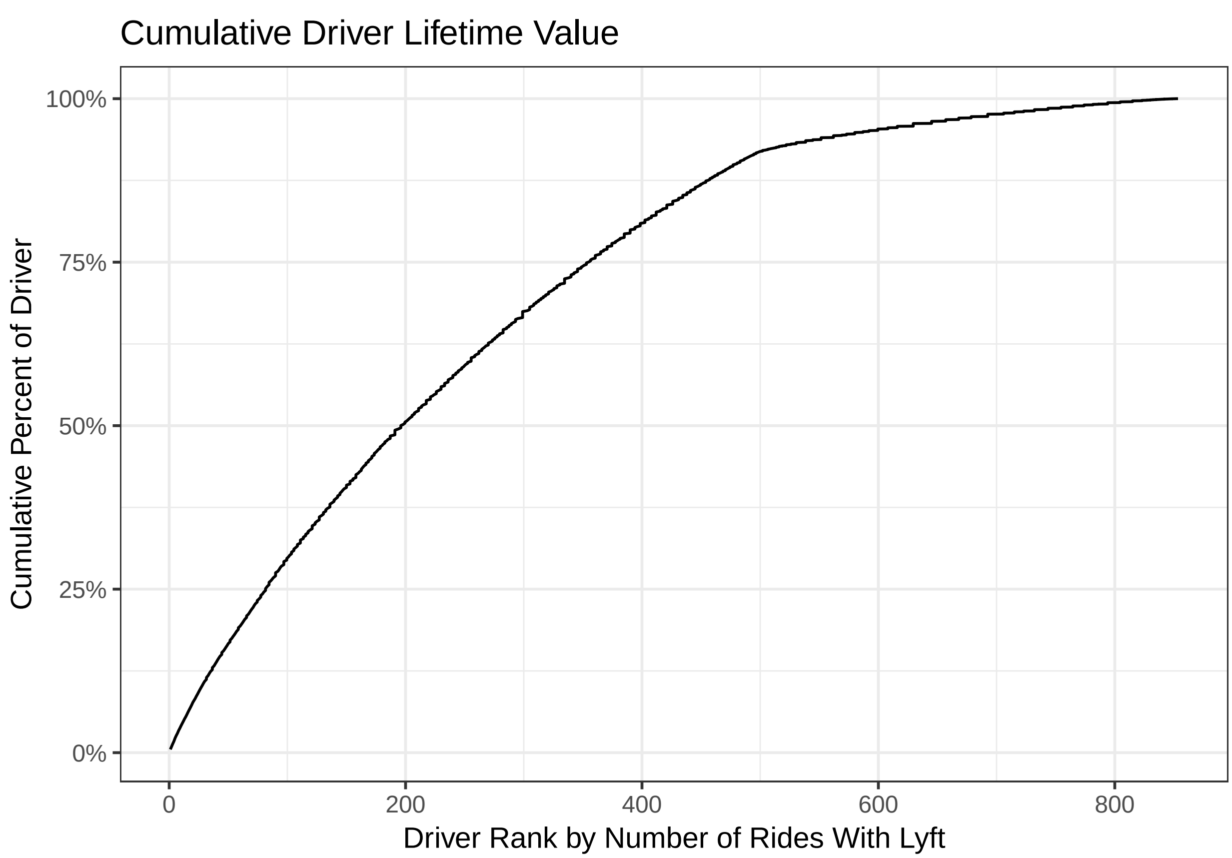
```r
  #  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  # 128.5 16761.5 23256.6 24502.8 31698.1 61751.4

# there seems to be a lot of drivers that don't generate much
# revenue at all...
driver_revenue_median = median(ride_info_drivers$driver_revenue)
```

```r
ride_info_drivers %>%
  group_by(driver_id) %>%
  summarize(driver_revenue = max(driver_revenue))%>%
  ggplot() +
  geom_histogram(aes(x = driver_revenue),
                 color = "white",
                 fill = "indianred1",
                 bins = 20) +
  geom_vline(xintercept = driver_revenue_median, lwd = 1.25,
             linetype = "dashed", color = "blue") +
  xlab("Driver Lifetime Value (Dollars)") +
  ylab("Count")
```




```r
# driver revenue cumulative distribution function

ride_info_drivers <- ride_info_drivers %>%
  group_by(driver_id) %>%
  mutate(num_rides = n())

cdf <- ride_info_drivers %>%
  select(driver_id, driver_revenue, num_rides) %>%
  group_by(driver_id) %>%
  summarize(driver_revenue = max(driver_revenue),
            num_rides = max(num_rides)) %>%
  arrange(desc(num_rides)) %>%
  unique() %>%
```

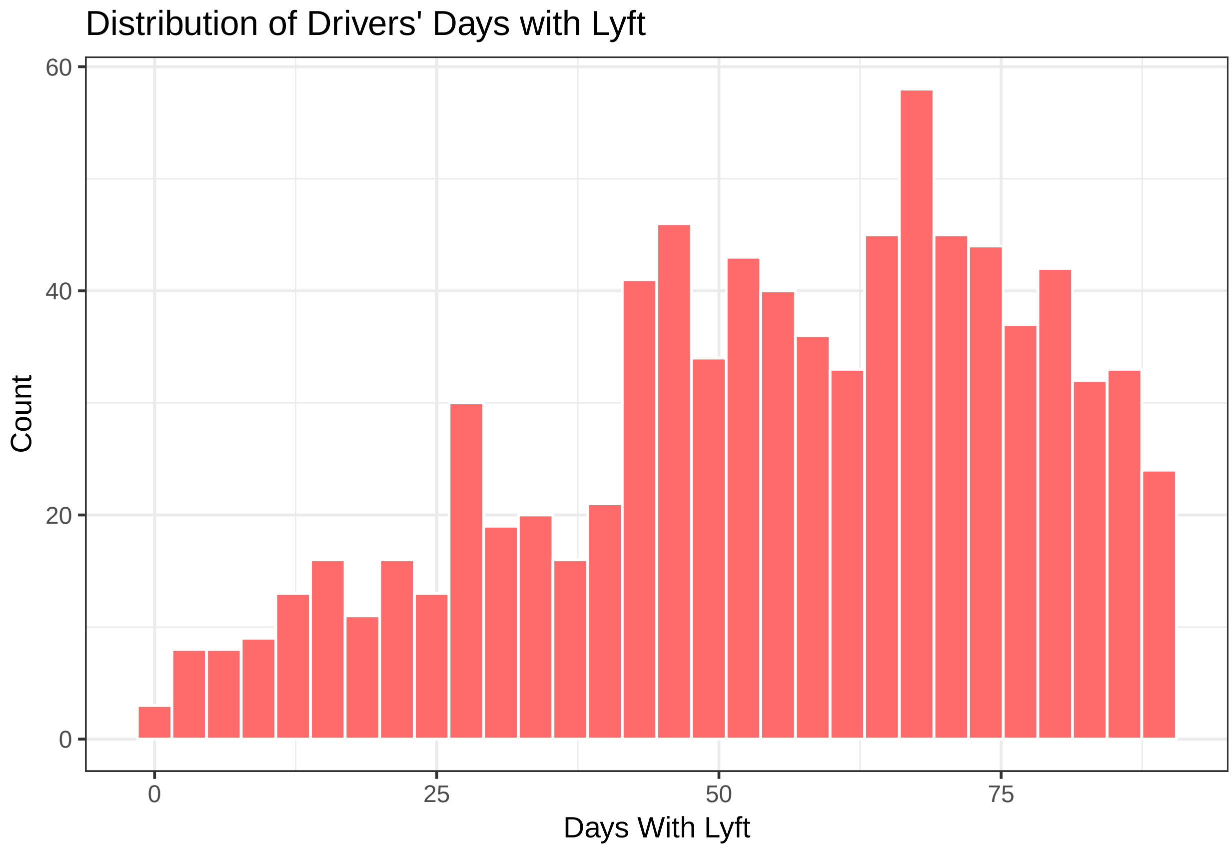
```r
  ungroup() %>%
  mutate(rank = rank(desc(num_rides)),
         cdf = cumsum(driver_revenue) / sum(driver_revenue)
         )
# a majority of the accumulative driver lifetime value is
# earned by drivers who rank 500 and lower in terms of number of rides
# The decrease in the slope also suggests that past this point
# it's economically inefficient to encourage drivers to
# make more rides

cdf %>%
  ggplot(aes(x = rank, y = cdf)) +
  geom_line() +
  scale_y_continuous(label = percent) +
  ylab('Cumulative Percent of Driver') +
  xlab('Driver Rank by Number of Rides With Lyft') +
  ggtitle('Cumulative Driver Lifetime Value')
```




```r
### Data Cleaning + Data Prep

 ## factor generation:

  # driver revenue by week
   weekly_driver_revenue <- ride_info_enhanced %>%
   mutate(event = as.character(event)) %>%
   filter(event == "accepted_at") %>%
```

```r
    mutate(week = week(timestamp2)) %>%
    group_by(driver_id,week) %>%
    mutate(weekly_driver_revenue = sum(price_per_trip_adj)) %>%
    ungroup() %>%
    group_by(driver_id) %>%
    summarize(weekly_driver_revenue = mean(weekly_driver_revenue))


  # time driver has been with Lyft
  ride_info_drivers <- ride_info_drivers %>%
    group_by(driver_id) %>%
    mutate(driver_onboard_date2 = max(driver_onboard_date2),
          last_ride_date = max(timestamp2)) %>%
    mutate(days_with_lyft = last_ride_date - driver_onboard_date2) %>%
    ungroup()

  days_with_lyft <- ride_info_drivers %>%
    select(driver_id, days_with_lyft) %>%
    group_by(driver_id) %>%
    summarize(days_with_lyft = max(days_with_lyft)) %>%
    ungroup()

# -------------------------------------------------------------
    # distribution of drivers' days with lyft
    ride_info_drivers %>%
      group_by(driver_id) %>%
      summarize(days_with_lyft = max(days_with_lyft)) %>%
      mutate(days_with_lyft = as.numeric(days_with_lyft)) %>%
      ungroup() %>%
      ggplot() +
      geom_histogram(aes(x = days_with_lyft),
                     color = "white",
                     fill = "indianred1"
                      ) +
      labs(title = "Distribution of Drivers' Days with Lyft") +
      xlab("Days With Lyft") +
      ylab("Count")
```

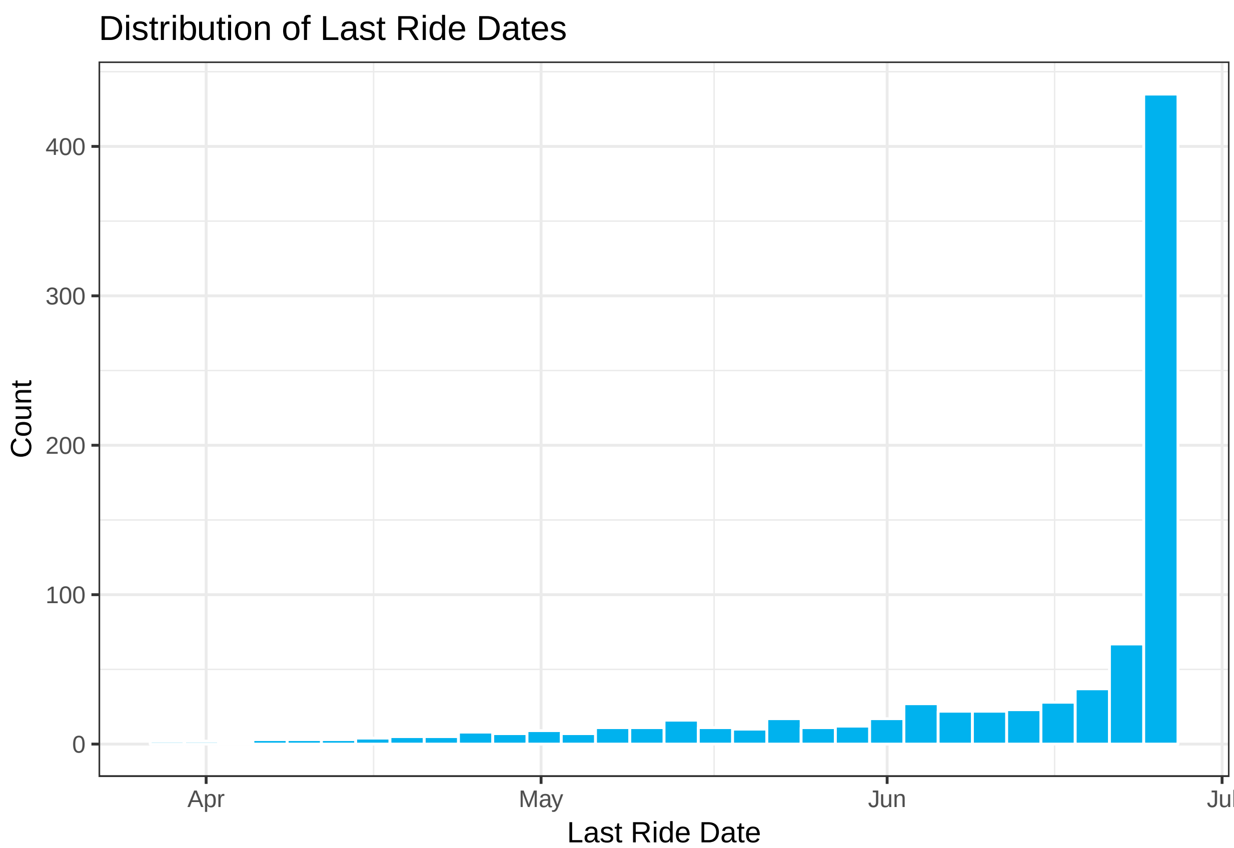
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 18 rows containing non-finite values (stat_bin).
```




```r
ride_info_drivers %>%
  group_by(driver_id) %>%
  summarize(last_ride_date = max(last_ride_date)) %>%
  ggplot(aes(x = last_ride_date)) +
  geom_histogram(color = "white", fill = "deepskyblue2") +
  labs(title = "Distribution of Last Ride Dates") +
  xlab("Last Ride Date") +
  ylab("Count")
```

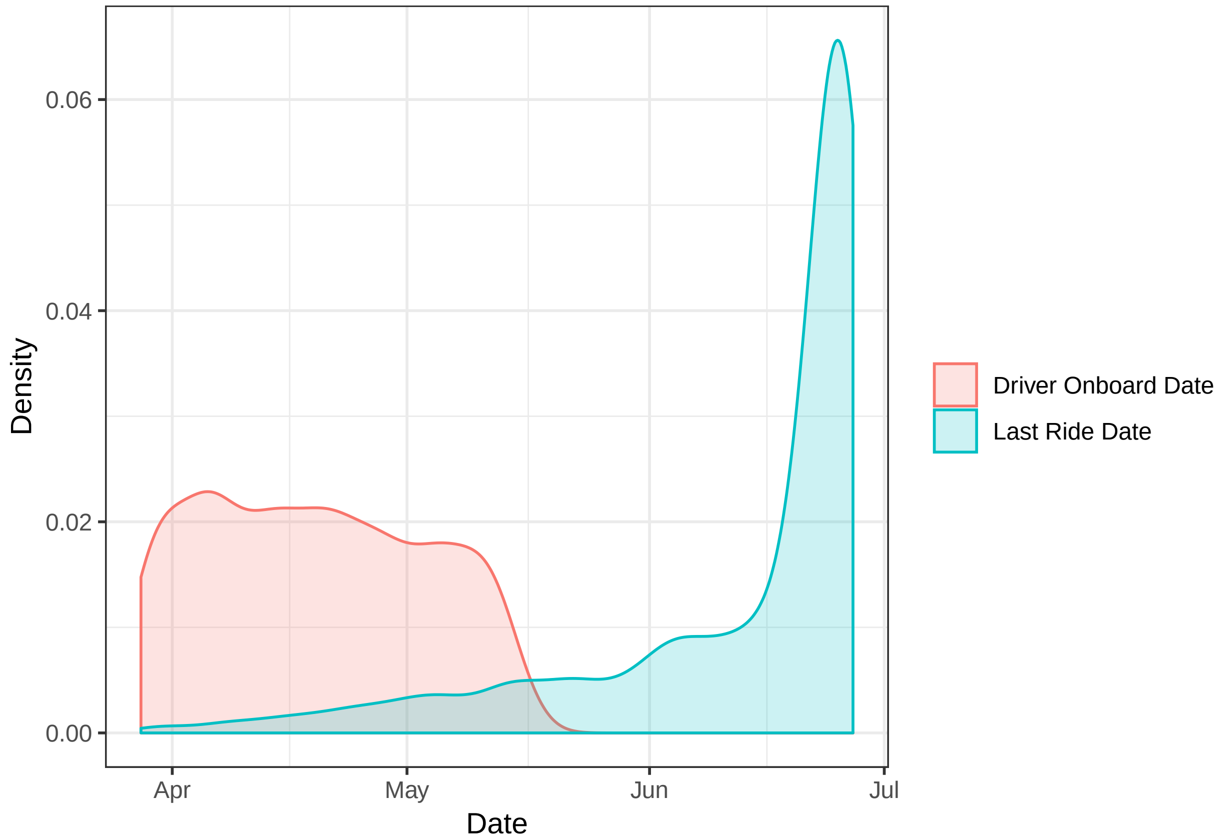
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 18 rows containing non-finite values (stat_bin).
```




```
# density histogram to visualize retention
date_density <- gather(ride_info_drivers, type, date, driver_onboard_date2,last_ride_date)

date_density <- date_density %>%
  select(driver_id, type, date)

    date_density %>%
      group_by(driver_id, type) %>%
      summarize(date = max(date)) %>%
      ggplot(aes(color = type, fill = type)) +
      geom_density(aes(x = date),
                   alpha = 0.2) +
      ylab("Density") +
      xlab("Date") +
      scale_color_discrete(name="",
                           labels=c("Driver Onboard Date",
                                    "Last Ride Date")) +
      scale_fill_discrete(name="",
                           labels=c("Driver Onboard Date",
                                    "Last Ride Date"))
```

```
## Warning: Removed 18 rows containing non-finite values (stat_density).
```



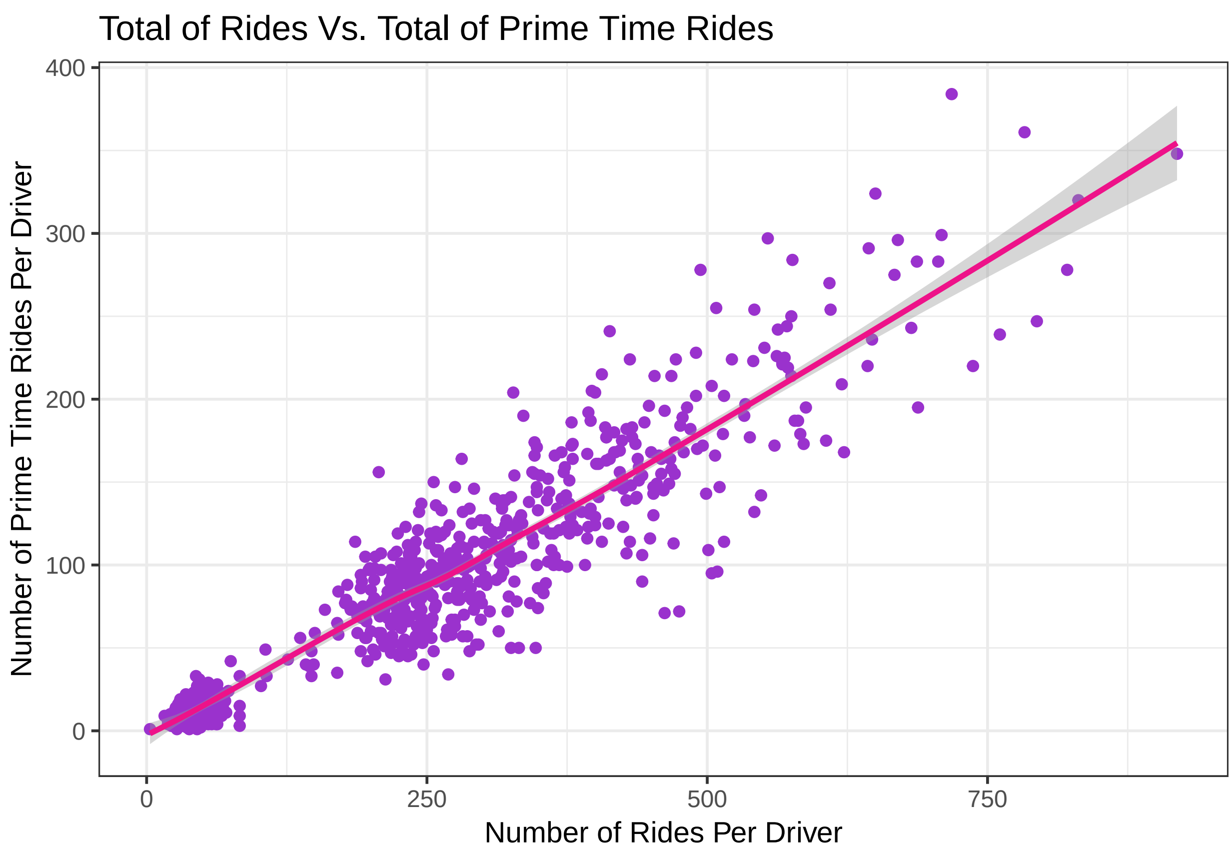

```r
# ------------------------------------------------------------


 # number of rides
   num_rides <- ride_info_drivers %>%
     group_by(driver_id) %>%
     distinct(ride_id) %>%
     summarize(num_rides = n())

 # mean requested times
   requested <- ride_info_drivers %>%
     select(driver_id, event, times) %>%
     filter(event == "requested_at") %>%
     group_by(driver_id) %>%
     summarize(requested_mean = mean(times),
               requested_total = sum(times)) %>%
     ungroup()

 # mean accepted times
   accepted <- ride_info_drivers %>%
     select(driver_id, event, times) %>%
     filter(event == "accepted_at") %>%
     group_by(driver_id) %>%
     summarize(accepted_mean = mean(times),
               accepted_total = sum(times)) %>%
     ungroup()
```

```r
# mean arrived times
  arrived <- ride_info_drivers %>%
    select(driver_id, event, times) %>%
    filter(event == "arrived_at") %>%
    group_by(driver_id) %>%
    summarize(arrived_mean = mean(times),
              arrived_total = sum(times)) %>%
    ungroup()

# mean picked up times
  picked_up <- ride_info_drivers %>%
    select(driver_id, event, times) %>%
    filter(event == "picked_up_at") %>%
    group_by(driver_id) %>%
    summarize(picked_up_mean = mean(times),
              picked_up_total = sum(times)) %>%
    ungroup()

# mean dropped off times
  dropped_off <- ride_info_drivers %>%
    select(driver_id, event, times) %>%
    filter(event == "dropped_off_at") %>%
    group_by(driver_id) %>%
    summarize(dropped_off_mean = mean(times),
              dropped_off_total = sum(times)) %>%
    ungroup()

# requested-arrived time lapse
  requested_arrived <- full_join(requested, arrived, by = "driver_id")

  requested_arrived <- requested_arrived %>%
    mutate(requested_arrived = arrived_mean - requested_mean) %>%
    select(driver_id, requested_arrived)

# accepted - arrived time lapse

  accepted_arrived <- full_join(accepted, arrived, by = "driver_id")

  accepted_arrived <- accepted_arrived %>%
    mutate(accepted_arrived = arrived_mean - accepted_mean) %>%
    select(driver_id, accepted_arrived)

# requested - dropped off time lapse
  # ride duration is only based on the arrived - dropped off
  requested_dropped_off <- full_join(requested, dropped_off, by = "driver_id")

  requested_dropped_off <- requested_dropped_off %>%
    mutate(requested_dropped_off = dropped_off_mean - requested_mean) %>%
    select(driver_id, requested_dropped_off)

# day/night
  day_night <- ride_info_drivers %>%
    filter(event == "accepted_at") %>%
```

```r
    select(driver_id, timestamp3) %>%
    mutate(hour = hour(timestamp3)) %>%
    group_by(driver_id) %>%
    summarize(hour = mean(hour)) %>%
    ungroup() %>%
    mutate(day_night = case_when((hour >= 6 & hour < 12) ~ "morning",
                                 (hour >= 12 & hour < 18) ~ "afternoon",
                                 (hour < 6 ) ~ "night",
                                 TRUE ~ "evening"
                                 )) %>%
    select(driver_id, day_night)

 # average miles per hour
  mph <- ride_info_drivers %>%
    mutate(ride_duration_hour = ride_duration_min / 60) %>%
    group_by(driver_id) %>%
    mutate(mean_mph = ride_distance_miles / ride_duration_hour) %>%
    summarize(mean_mph = mean(mean_mph)) %>%
    ungroup()

  # counts of prime time
  prime_time_counts <- ride_info_drivers %>%
    filter(ride_prime_time != 0) %>%
    group_by(driver_id) %>%
    distinct(ride_id) %>%
    summarize(prime_time_counts = n()) %>%
    ungroup()

# Graphing total number of rides vs. the number of rides
# that had Prime Time applied to them
  num_rides_vs_num_prime <- full_join(num_rides,
                                      prime_time_counts,
                                      by = "driver_id")

  num_rides_vs_num_prime %>%
    ggplot() +
    geom_point(aes(x = num_rides, y = prime_time_counts),
               color = "darkorchid3") +
    geom_smooth(aes(x = num_rides, y = prime_time_counts),
                method = "loess", color = "deeppink2") +
    labs(title = "Total of Rides Vs. Total of Prime Time Rides") +
    xlab("Number of Rides Per Driver") +
    ylab("Number of Prime Time Rides Per Driver")
```

```
## Warning: Removed 3 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 3 rows containing missing values (geom_point).
```




```
# -------------------------------------------------------
```

```
# driver revenue dataset
    driver_revenue <- ride_info_drivers %>%
      group_by(driver_id) %>%
      summarize(driver_revenue = max(driver_revenue))

    driver_revenue <- left_join(driver_revenue, days_with_lyft)
```

```
## Joining, by = "driver_id"
```

```
    driver_revenue <- left_join(driver_revenue, num_rides)
```

```
## Joining, by = "driver_id"
```

```
    driver_revenue <- left_join(driver_revenue, requested)
```

```
## Joining, by = "driver_id"
```

```
    driver_revenue <- left_join(driver_revenue, accepted)
```

```
## Joining, by = "driver_id"
```

```
    driver_revenue <- left_join(driver_revenue, arrived)
```

```
## Joining, by = "driver_id"
```

```
    driver_revenue <- left_join(driver_revenue, picked_up)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue, dropped_off)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue, requested_arrived)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue,
                            accepted_arrived)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue,
                            requested_dropped_off)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue, day_night)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue, mph)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue,
                            prime_time_counts)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- left_join(driver_revenue,
                            weekly_driver_revenue)
```

```
## Joining, by = "driver_id"
```

```r
driver_revenue <- driver_revenue %>%
  mutate(num_prime_num_rides = prime_time_counts / num_rides)
```

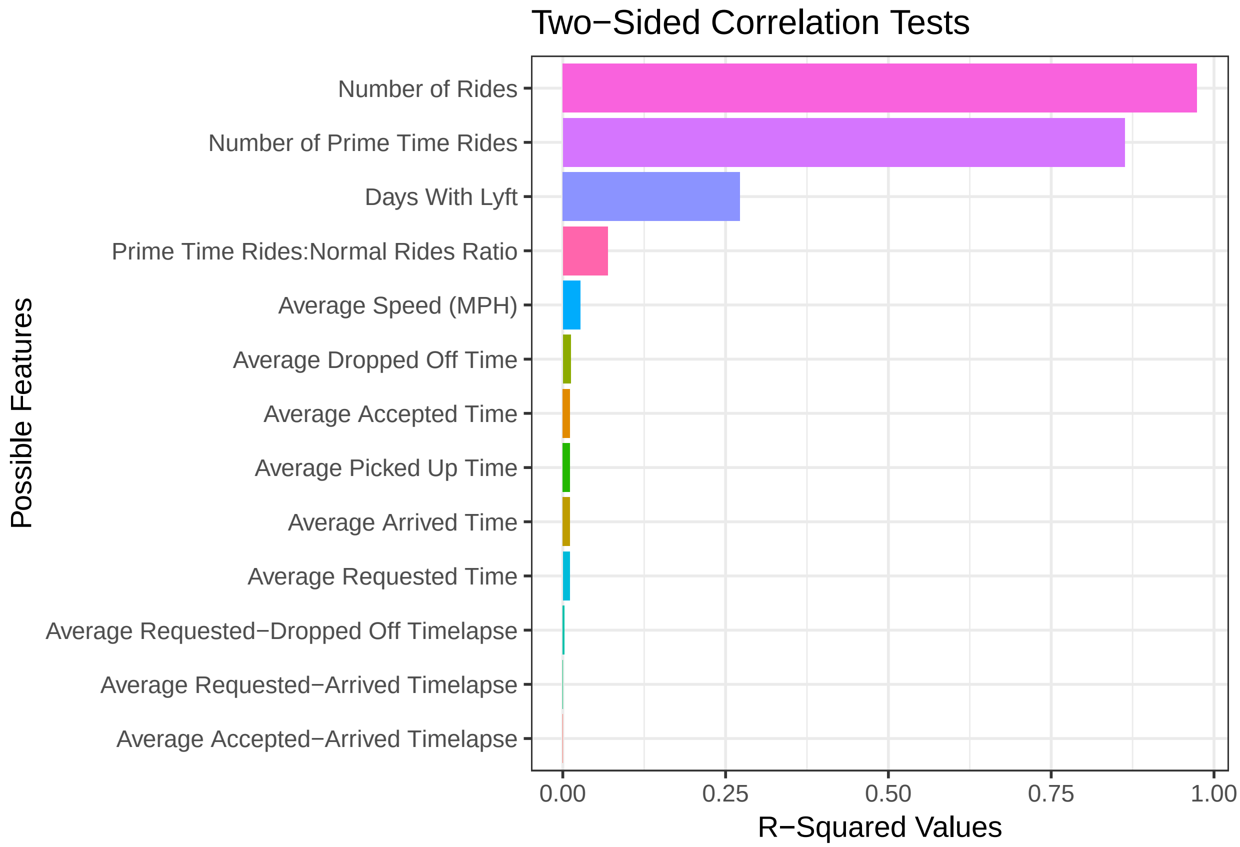
```r
### Statistical Tests (Parametric & Non-Parametric)

# simple correlation tests in order to see what factors
# matter the most when it comes to driver revenue
# r - linear correlation (+1 is a strong positive relationship,
#                         -1 is a strong negative relationship,
#                         0 means no relationship)
# R^2 - how much of the variation of the dependent variable
#       can be attributed to the independent variable


## Factors Being Considered
# days_with_lyft
# num_rides
# requested
# accepted
# arrived
# picked_up
# dropped_off
# requested_arrived
```

```
# accepted_arrived
# requested_dropped_off
## day_night <- not being included with correlation
# mean_mph
# prime_time_counts
# num_prime_num_rides


# removing NAs
driver_revenue2 <- na.omit(driver_revenue)

# calulating the r-sq values in order to represent how much the
# variation in driver revenue can be explained
# by each individual factor
rsq <- rep(0,13)
p_values <- rep(0,13)

possible_features <- c("Days With Lyft",
                       "Number of Rides",
                       "Average Requested Time",
                       "Average Accepted Time",
                       "Average Arrived Time",
                       "Average Picked Up Time",
                       "Average Dropped Off Time",
                       "Average Requested-Arrived Timelapse",
                       "Average Accepted-Arrived Timelapse",
                       "Average Requested-Dropped Off Timelapse",
                       "Average Speed (MPH)",
                       "Number of Prime Time Rides",
                       "Prime Time Rides:Normal Rides Ratio")

rsq[1] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$days_with_lyft)))^2
rsq[2] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$num_rides)))^2
rsq[3] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_mean)))^2
rsq[4] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$accepted_mean)))^2
rsq[5] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$arrived_mean)))^2
rsq[6] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$picked_up_mean)))^2
rsq[7] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$dropped_off_mean)))^2
rsq[8] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_arrived)))^2
rsq[9] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$accepted_arrived)))^2
rsq[10] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_dropped_off)))^2
rsq[11] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$mean_mph)))^2
rsq[12] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$prime_time_counts)))^2
rsq[13] <- (cor(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$num_prime_num_rides)))^2

# p-values of each correlation test
# all of them are statistically significant
p_values[1] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$days_with_lyft))$p.va
p_values[2] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$num_rides))$p.value
p_values[3] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_mean))$p.va
p_values[4] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$accepted_mean))$p.val
p_values[5] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$arrived_mean))$p.valu
p_values[6] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$picked_up_mean))$p.va
```

```r
p_values[7] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$dropped_off_mean))$p
p_values[8] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_arrived))$p
p_values[9] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$accepted_arrived))$p
p_values[10] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$requested_dropped_of
p_values[11] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$mean_mph))$p.value
p_values[12] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$prime_time_counts))$
p_values[13] <- cor.test(driver_revenue2$driver_revenue, as.numeric(driver_revenue2$num_prime_num_rides)

cor_dfr <- data_frame(possible_features, rsq, p_values)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

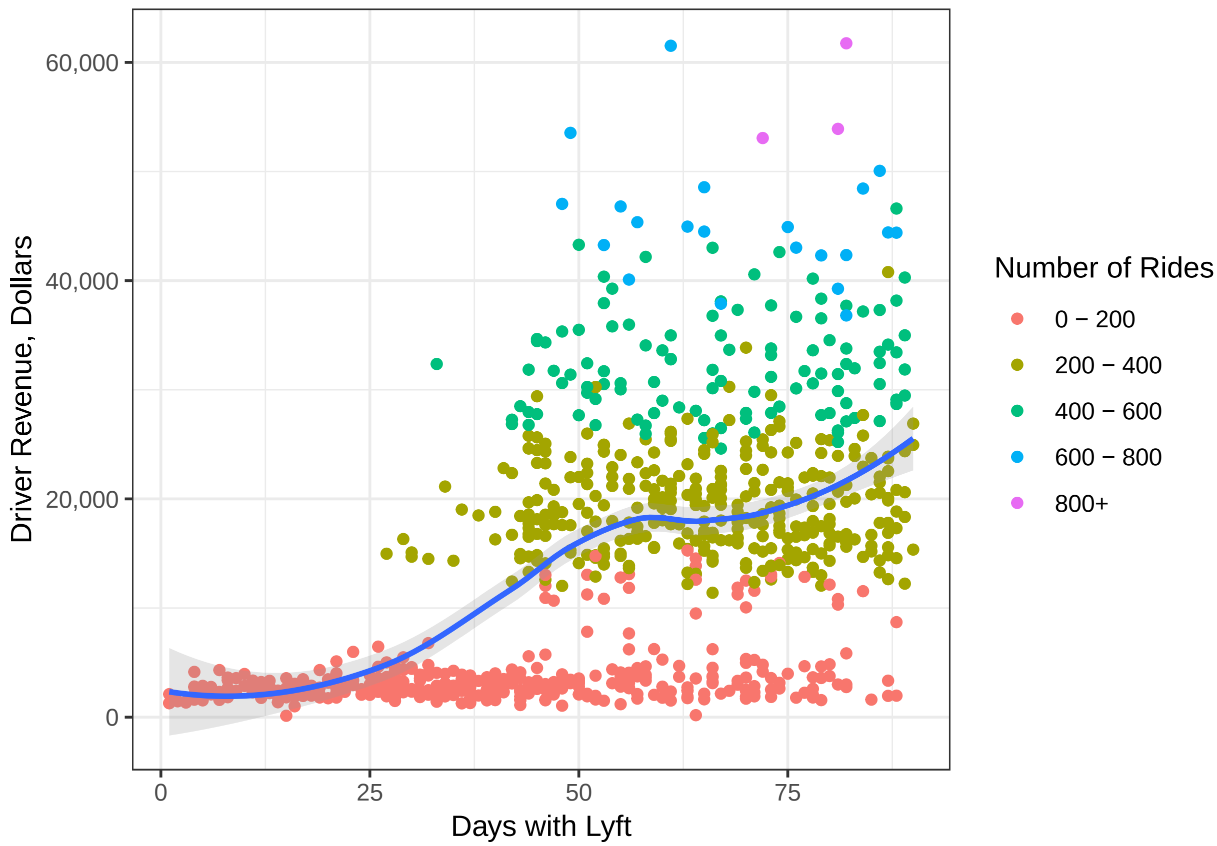
```r
# bar plot showing each factors Rsq value
cor_dfr %>%
  ggplot() +
  geom_col(aes(x = reorder(possible_features, rsq), y = rsq, fill = possible_features)) +
  coord_flip() +
  ylab("R-Squared Values") +
  xlab("Possible Features") +
  labs(title = "Two-Sided Correlation Tests") +
  theme(legend.position = "none")
```




```r
# plot visualizing the signicant correlation between the number
# of rides have on the variation in driver revenue
driver_revenue %>%
  mutate(num_rides_group = case_when((num_rides < 200) ~ "0 - 200",
```

```
                                      (num_rides < 400) ~ "200 - 400",
                                      (num_rides < 600) ~ "400 - 600",
                                      (num_rides < 800) ~ "600 - 800",
                                      TRUE ~ "800+"
  )) %>%
  mutate(num_rides_group = as.factor(num_rides_group)) %>%
  ggplot(aes(x = days_with_lyft, y = driver_revenue)) +
  geom_point(aes(color = num_rides_group)) +
  ylab("Driver Revenue, Dollars") +
  xlab("Days with Lyft") +
  geom_smooth(aes(x = days_with_lyft, y = driver_revenue), group = 1, alpha = 0.25) +
  scale_y_continuous(labels = comma) +
  labs(color = "Number of Rides")
```

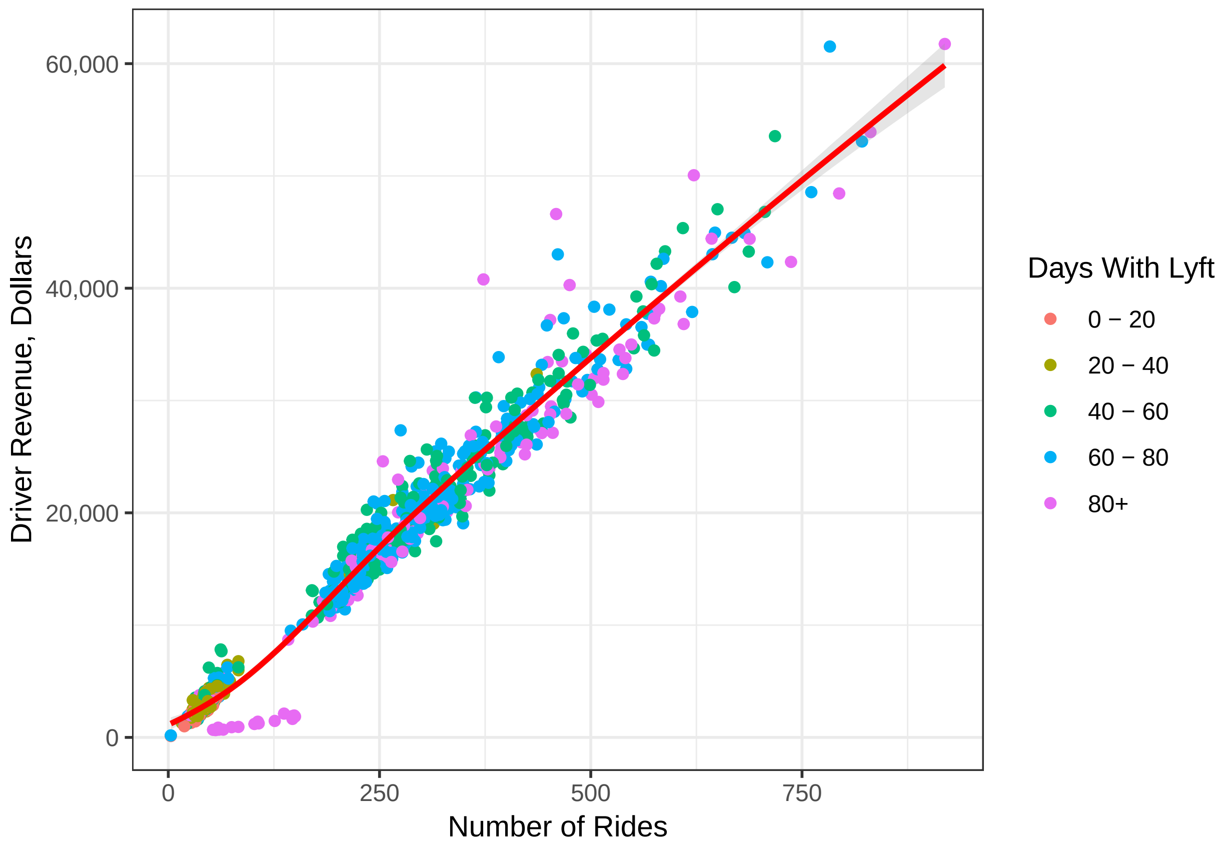
```
## Don't know how to automatically pick scale for object of type difftime. Defaulting to continuous.
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning: Removed 18 rows containing non-finite values (stat_smooth).
## Warning: Removed 18 rows containing missing values (geom_point).
```




```
# another plot visualizing the same thing in another way
driver_revenue %>%
  mutate(days_with_lyft = as.numeric(days_with_lyft)) %>%
  mutate(days_with_lyft_group = case_when((days_with_lyft < 20) ~ "0 - 20",
                                      (days_with_lyft < 40) ~ "20 - 40",
                                      (days_with_lyft < 60) ~ "40 - 60",
```

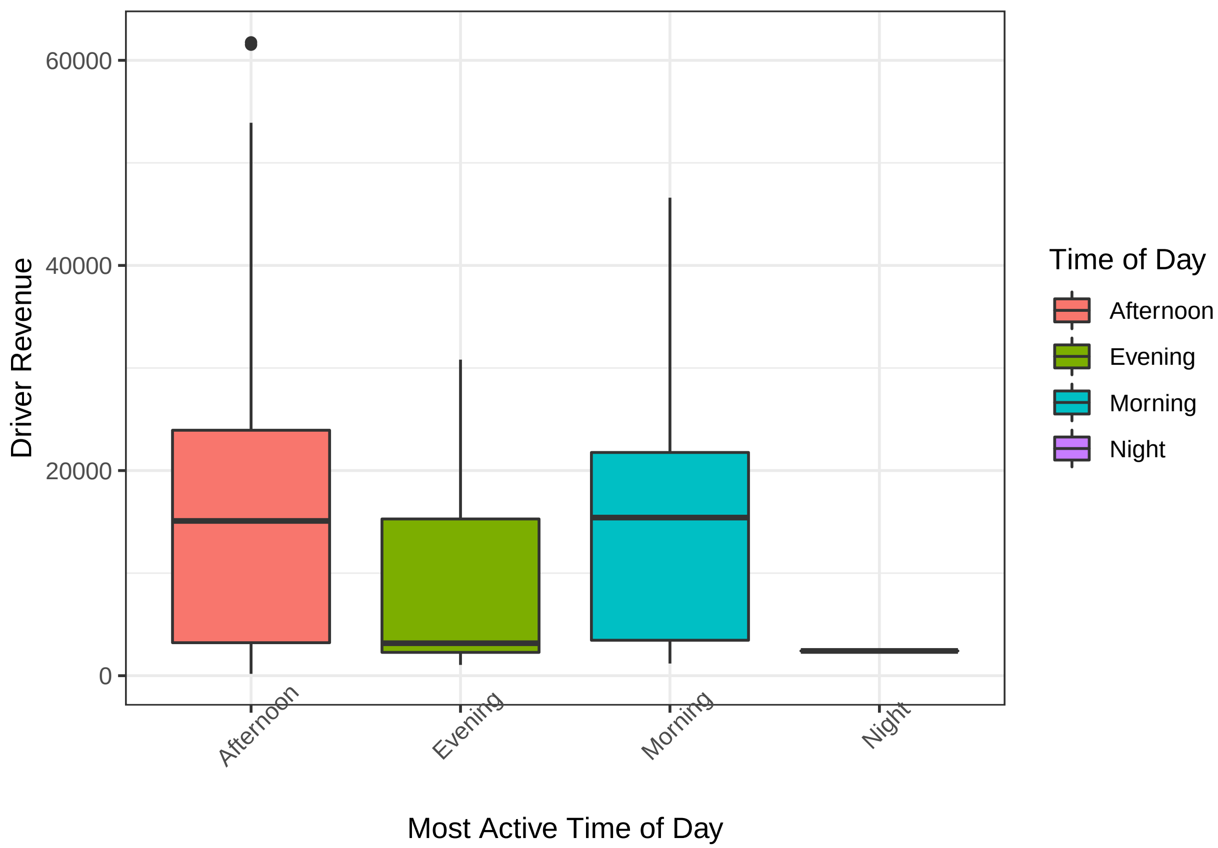
```
                                            (days_with_lyft < 80) ~ "60 - 80",
                                            TRUE ~ "80+"
  )) %>%
  mutate(days_with_lyft_group = as.factor(days_with_lyft_group)) %>%
  ggplot(aes(x = num_rides, y = driver_revenue)) +
  geom_point(aes(color = days_with_lyft_group)) +
  ylab("Driver Revenue, Dollars") +
  xlab("Number of Rides") +
  geom_smooth(aes(x = num_rides, y = driver_revenue), group = 1, alpha = 0.25, color = "red") +
  scale_y_continuous(labels = comma) +
  labs(color = "Days With Lyft")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

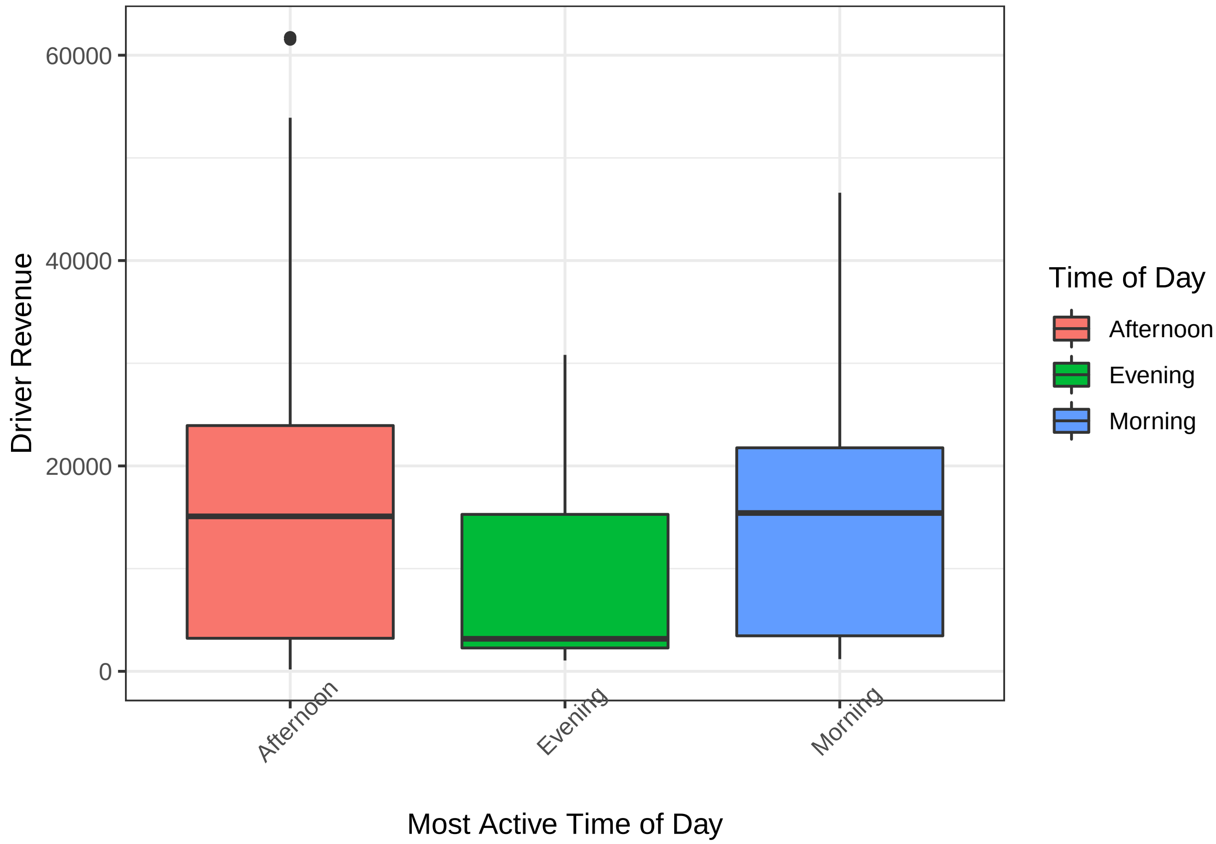



```
revenue <- driver_revenue$driver_revenue
day_night <- as.factor(driver_revenue$day_night)

# visualizing the distributions of driver revenue among
# different groups of drivers
# drivers are grouped by the time of day they are most active
driver_revenue2 %>%
  mutate(day_night = case_when((day_night == "morning") ~ "Morning",
                               (day_night == "afternoon") ~ "Afternoon",
                               (day_night == "evening") ~ "Evening",
                               (day_night == "night") ~ "Night"
```

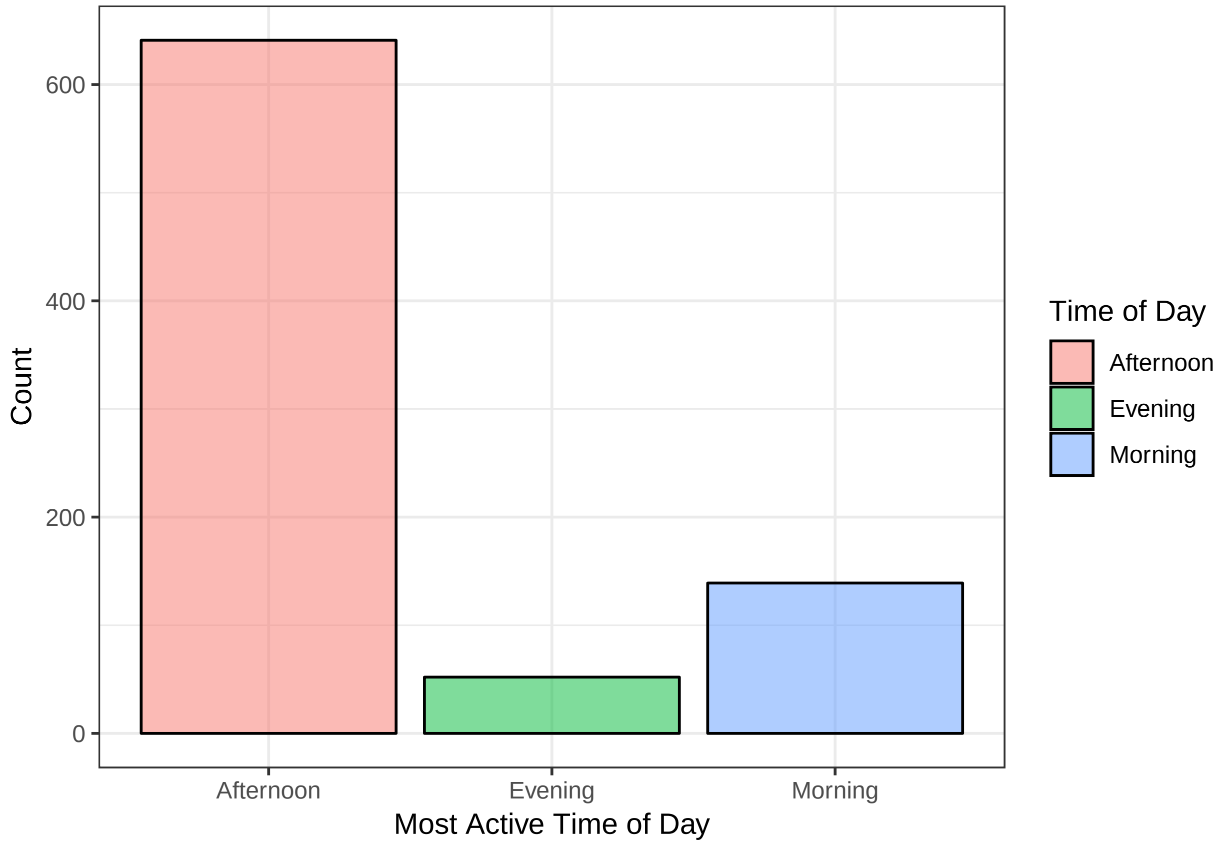
```
  )) %>%
  ggplot() +
  geom_boxplot(aes(x = as.factor(day_night), y = driver_revenue,
                   fill = as.factor(day_night))) +
  theme(axis.text.x = element_text(angle = 45)) +
  xlab("Most Active Time of Day") +
  ylab("Driver Revenue") +
  scale_fill_discrete(name = "Time of Day")
```



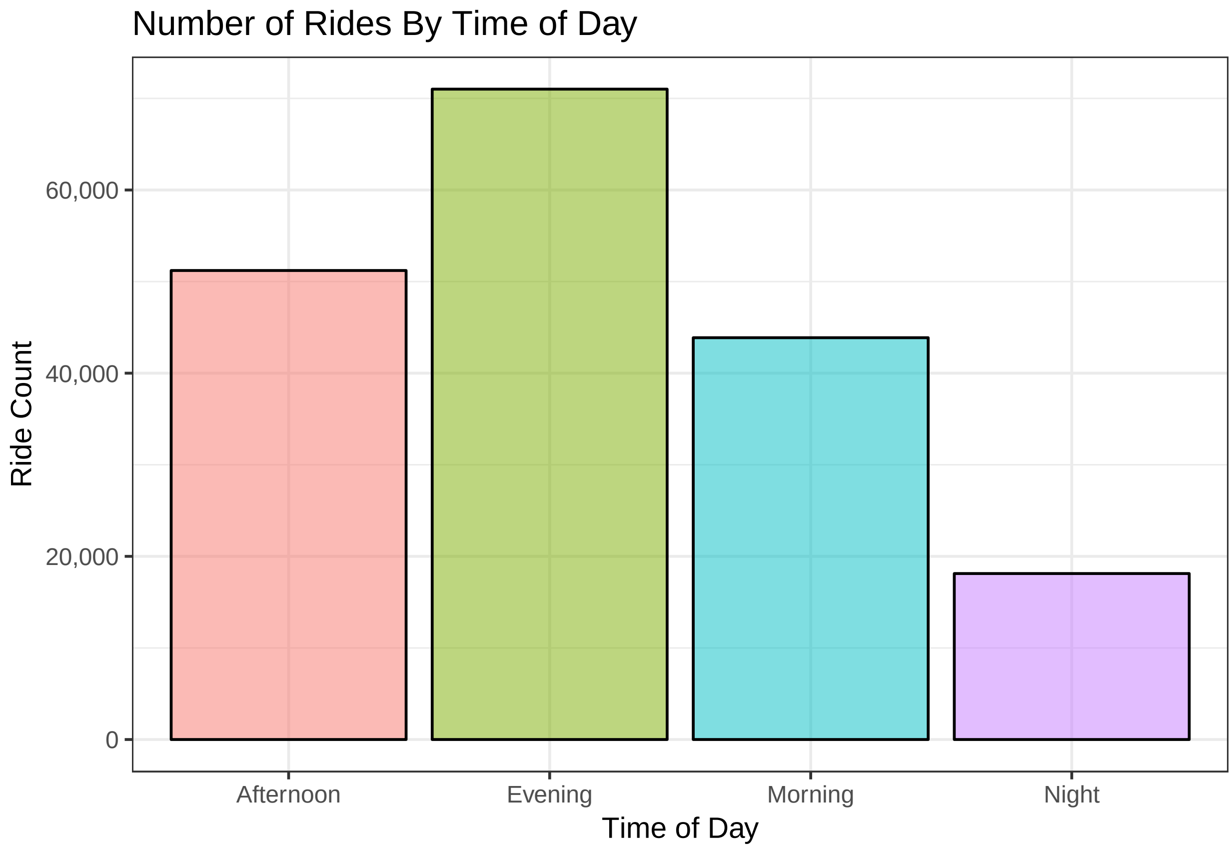

```
# removes 'night' for a cleaner graph
driver_revenue2 %>%
  filter(day_night != "night") %>%
  mutate(day_night = case_when((day_night == "morning") ~ "Morning",
                               (day_night == "afternoon") ~ "Afternoon",
                               (day_night == "evening") ~ "Evening",
                               (day_night == "night") ~ "Night"

  )) %>%
  ggplot() +
  geom_boxplot(aes(x = as.factor(day_night), y = driver_revenue,
                   fill = as.factor(day_night))) +
  theme(axis.text.x = element_text(angle = 45)) +
  xlab("Most Active Time of Day") +
  ylab("Driver Revenue") +
  scale_fill_discrete(name = "Time of Day")
```




```
# two graphs showing how many rides are completed in each time
# of day
driver_revenue2 %>%
  filter(day_night != "night") %>%
  mutate(day_night = case_when((day_night == "morning") ~ "Morning",
                               (day_night == "afternoon") ~ "Afternoon",
                               (day_night == "evening") ~ "Evening",
                               (day_night == "night") ~ "Night"

  )) %>%
  ggplot() +
  geom_bar(aes(x = day_night, fill = day_night),
           color = "black",
           alpha = 0.5) +
  xlab("Most Active Time of Day") +
  ylab("Count") +
  scale_fill_discrete(name = "Time of Day")
```




```
ride_info_drivers %>%
  filter(event == "accepted_at") %>%
      select(driver_id, timestamp3) %>%
      mutate(hour = hour(timestamp3)) %>%
      mutate(day_night = case_when((hour >= 6 & hour < 12) ~ "Morning",
                             (hour >= 12 & hour < 18) ~ "Afternoon",
                             (hour < 6 ) ~ "Night",
                             TRUE ~ "Evening"
                             )) %>%
  ggplot(aes(x = day_night)) +
  geom_bar(aes(fill = day_night),
           color = "black",
           alpha = 0.5) +
  scale_y_continuous(labels = comma) +
  xlab("Time of Day") +
  ylab("Ride Count") +
  theme(legend.position = "none") +
  ggtitle("Number of Rides By Time of Day")
```

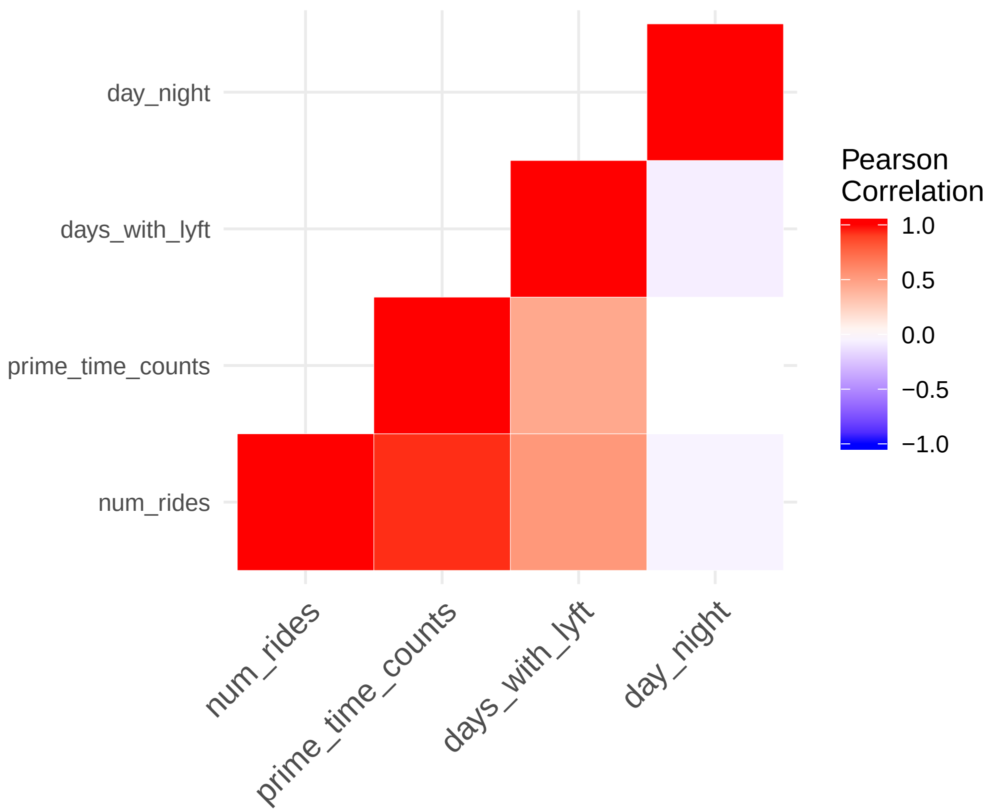



```r
# compact dataframe is made to create
# a correlation matrix heatmap, to check for interdependent
# variables
driver_revenue3 <- driver_revenue2 %>%
  select(num_rides,prime_time_counts, days_with_lyft, day_night) %>%
  mutate(days_with_lyft = as.numeric(days_with_lyft)) %>%
  mutate(day_night = case_when((day_night == "morning") ~ 1,
                               (day_night == "afternoon") ~ 2,
                               (day_night == "evening") ~ 3,
                               (day_night == "night") ~ 4

  ))

cormat <- round(cor(driver_revenue3),2)

get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

# upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}
```

```r
upper_tri <- get_upper_tri(cormat)
upper_tri
```

```
##                  num_rides prime_time_counts days_with_lyft day_night
## num_rides                1              0.94           0.53     -0.05
## prime_time_counts       NA              1.00           0.45      0.00
## days_with_lyft          NA                NA           1.00     -0.07
## day_night               NA                NA             NA      1.00
```

```r
# melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)

# correlation matrix heatmap
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1)) +
  coord_fixed() +
  xlab("") +
  ylab("")
```




```r
# linear regression model to calculate projected lifetime of a driver
```

```r
# response variable: days_with_lyft

# possible features:
# driver revenue
# average number of rides per week *
# average trip_duration *
# average trip_distance *
# average requested_arrived timelapse
# average accepted_arrived timelapse
# average accepted requested_dropped_off timelapse
# day_night * (might need to convert to numbers)
# num_prime_rides:num_rides ratio

driver_revenue2 <- driver_revenue2 %>%
  mutate(days_with_lyft = as.numeric(days_with_lyft)) %>%
  mutate(weeks_with_lyft = days_with_lyft / 7) %>%
  mutate(mean_rides_per_week = num_rides / weeks_with_lyft)

total_trip_stats <- ride_info %>%
  group_by(driver_id) %>%
  summarize(total_duration = sum(ride_duration),
         total_distance = sum(ride_distance))

driver_revenue2 <- full_join(total_trip_stats, driver_revenue2,
                             by = "driver_id")

driver_revenue2 <- driver_revenue2 %>%
  mutate(mean_duration = total_duration / num_rides,
         mean_distance = total_distance / num_rides)

model1 <- lm(days_with_lyft ~ (weekly_driver_revenue +
                              mean_rides_per_week +
                              (mean_duration *
                              mean_distance *
                                requested_dropped_off) +
                                (requested_arrived *
                                accepted_arrived) +
                                day_night +
                                num_prime_num_rides
),
data = driver_revenue2)

summary(model1)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     (mean_duration * mean_distance * requested_dropped_off) +
##     (requested_arrived * accepted_arrived) + day_night + num_prime_num_rides),
##     data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.397 -14.059   1.257  14.852  63.974
```

```
##
## Coefficients:
##                                                Estimate Std. Error
## (Intercept)                                   3.805e+01  1.742e+01
## weekly_driver_revenue                         5.412e-02  4.682e-03
## mean_rides_per_week                          -5.498e-01  5.138e-02
## mean_duration                                 6.081e-03  4.063e-03
## mean_distance                                -2.913e-05  4.384e-04
## requested_dropped_off                         1.746e-02  8.150e-03
## requested_arrived                            -1.704e-02  1.765e-02
## accepted_arrived                              9.505e-03  1.794e-02
## day_nightevening                              7.642e-01  2.915e+00
## day_nightmorning                              3.869e+00  1.867e+00
## day_nightnight                               -1.470e+01  1.952e+01
## num_prime_num_rides                          -1.103e+01  7.055e+00
## mean_duration:mean_distance                  -6.171e-08  9.191e-08
## mean_duration:requested_dropped_off          -4.817e-06  1.994e-06
## mean_distance:requested_dropped_off          -1.899e-07  2.168e-07
## requested_arrived:accepted_arrived           -4.802e-06  1.271e-06
## mean_duration:mean_distance:requested_dropped_off  5.762e-11  4.445e-11
##                                              t value Pr(>|t|)
## (Intercept)                                    2.184 0.029264 *
## weekly_driver_revenue                         11.560  < 2e-16 ***
## mean_rides_per_week                          -10.700  < 2e-16 ***
## mean_duration                                  1.496 0.134912
## mean_distance                                 -0.066 0.947047
## requested_dropped_off                          2.143 0.032428 *
## requested_arrived                             -0.965 0.334604
## accepted_arrived                               0.530 0.596471
## day_nightevening                               0.262 0.793273
## day_nightmorning                               2.072 0.038593 *
## day_nightnight                                -0.753 0.451777
## num_prime_num_rides                           -1.563 0.118441
## mean_duration:mean_distance                   -0.671 0.502178
## mean_duration:requested_dropped_off           -2.416 0.015898 *
## mean_distance:requested_dropped_off           -0.876 0.381305
## requested_arrived:accepted_arrived            -3.779 0.000169 ***
## mean_duration:mean_distance:requested_dropped_off   1.296 0.195225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.42 on 816 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.2154, Adjusted R-squared:  0.2001
## F-statistic: 14.01 on 16 and 816 DF,  p-value: < 2.2e-16
```

```r
# highest non-interaction insignificant p-value is
# mean_distance, removing it for model 2

model2 <- lm(days_with_lyft ~ (weekly_driver_revenue +
                               mean_rides_per_week +
                               (mean_duration *
                                 requested_dropped_off) +
                                 (requested_arrived *
```

```r
                                    accepted_arrived) +
                                    day_night +
                                    +
                                    num_prime_num_rides
),
data = driver_revenue2)

summary(model2)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     (mean_duration * requested_dropped_off) + (requested_arrived *
##     accepted_arrived) + day_night + +num_prime_num_rides), data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -45.731 -14.730   0.723  15.149  59.409
##
## Coefficients:
##                                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                            5.022e+01  6.583e+00   7.629 6.58e-14
## weekly_driver_revenue                  5.255e-02  4.619e-03  11.377  < 2e-16
## mean_rides_per_week                   -5.210e-01  5.041e-02 -10.335  < 2e-16
## mean_duration                         -1.311e-05  1.461e-03  -0.009   0.9928
## requested_dropped_off                  6.172e-03  3.345e-03   1.845   0.0653
## requested_arrived                     -1.561e-02  1.768e-02  -0.883   0.3773
## accepted_arrived                       7.451e-03  1.797e-02   0.415   0.6784
## day_nightevening                      -7.045e-02  2.896e+00  -0.024   0.9806
## day_nightmorning                       3.363e+00  1.859e+00   1.809   0.0708
## day_nightnight                        -1.578e+01  1.958e+01  -0.806   0.4205
## num_prime_num_rides                   -2.206e+00  6.282e+00  -0.351   0.7256
## mean_duration:requested_dropped_off   -1.434e-06  8.189e-07  -1.751   0.0803
## requested_arrived:accepted_arrived    -5.204e-06  1.267e-06  -4.109 4.37e-05
##
## (Intercept)                           ***
## weekly_driver_revenue                 ***
## mean_rides_per_week                   ***
## mean_duration
## requested_dropped_off                 .
## requested_arrived
## accepted_arrived
## day_nightevening
## day_nightmorning                      .
## day_nightnight
## num_prime_num_rides
## mean_duration:requested_dropped_off .
## requested_arrived:accepted_arrived  ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.49 on 820 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.206,  Adjusted R-squared:  0.1944
```

```
## F-statistic: 17.73 on 12 and 820 DF,  p-value: < 2.2e-16
```

```r
# highest non-interaction insignificant p-value is
# mean_duration
# removing it for model3

model3 <- lm(days_with_lyft ~ (weekly_driver_revenue +
                               mean_rides_per_week +
                               requested_dropped_off +
                                 (requested_arrived *
                                 accepted_arrived) +
                                 day_night +
                                 num_prime_num_rides
),
data = driver_revenue2)

summary(model3)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     requested_dropped_off + (requested_arrived * accepted_arrived) +
##     day_night + num_prime_num_rides), data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.269 -14.387   0.848  14.968  58.810
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         4.969e+01  2.426e+00  20.481  < 2e-16
## weekly_driver_revenue               5.264e-02  4.469e-03  11.779  < 2e-16
## mean_rides_per_week                -5.185e-01  4.915e-02 -10.549  < 2e-16
## requested_dropped_off               3.529e-04  6.867e-04   0.514   0.6075
## requested_arrived                  -1.763e-02  1.763e-02  -1.000   0.3177
## accepted_arrived                    8.834e-03  1.794e-02   0.493   0.6225
## day_nightevening                    8.687e-02  2.870e+00   0.030   0.9759
## day_nightmorning                    3.233e+00  1.858e+00   1.740   0.0822
## day_nightnight                     -1.483e+01  1.957e+01  -0.758   0.4488
## num_prime_num_rides                -1.827e+00  6.277e+00  -0.291   0.7711
## requested_arrived:accepted_arrived -5.600e-06  1.248e-06  -4.486 8.31e-06
##
## (Intercept)                        ***
## weekly_driver_revenue              ***
## mean_rides_per_week                ***
## requested_dropped_off
## requested_arrived
## accepted_arrived
## day_nightevening
## day_nightmorning                   .
## day_nightnight
## num_prime_num_rides
## requested_arrived:accepted_arrived ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 19.51 on 822 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.2027, Adjusted R-squared:  0.193
## F-statistic:  20.9 on 10 and 822 DF,  p-value: < 2.2e-16
```

```
# highest non-interaction insignificant p-value(s) is
# time of day, removing it for model4

model4 <- lm(days_with_lyft ~ (weekly_driver_revenue +
                                mean_rides_per_week +
                                requested_dropped_off +
                                  (requested_arrived *
                                  accepted_arrived) +
                                  num_prime_num_rides
),
data = driver_revenue2)

summary(model4)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     requested_dropped_off + (requested_arrived * accepted_arrived) +
##     num_prime_num_rides), data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.260 -14.438   0.753  15.130  58.807
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        5.031e+01  2.371e+00  21.214  < 2e-16
## weekly_driver_revenue              5.303e-02  4.427e-03  11.979  < 2e-16
## mean_rides_per_week               -5.221e-01  4.912e-02 -10.629  < 2e-16
## requested_dropped_off              4.760e-04  6.825e-04   0.697    0.486
## requested_arrived                 -1.732e-02  1.764e-02  -0.982    0.326
## accepted_arrived                   8.694e-03  1.794e-02   0.485    0.628
## num_prime_num_rides               -2.547e+00  6.214e+00  -0.410    0.682
## requested_arrived:accepted_arrived -5.423e-06  1.245e-06  -4.355  1.5e-05
##
## (Intercept)                        ***
## weekly_driver_revenue              ***
## mean_rides_per_week                ***
## requested_dropped_off
## requested_arrived
## accepted_arrived
## num_prime_num_rides
## requested_arrived:accepted_arrived ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.51 on 825 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.1991, Adjusted R-squared:  0.1923
```

```
## F-statistic:  29.3 on 7 and 825 DF,  p-value: < 2.2e-16
```

```r
# highest non-interaction insignificant p-value is
# the ratio of the number of prime time rides to number of
# non-prime time rides , removing it for model5

model5 <- lm(days_with_lyft ~ (weekly_driver_revenue +
                               mean_rides_per_week +
                               requested_dropped_off +
                                 (requested_arrived *
                                 accepted_arrived)
),
data = driver_revenue2)

summary(model5)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     requested_dropped_off + (requested_arrived * accepted_arrived)),
##     data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.745 -14.525   0.787  15.065  59.063
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         4.957e+01  1.532e+00  32.357  < 2e-16
## weekly_driver_revenue               5.268e-02  4.343e-03  12.130  < 2e-16
## mean_rides_per_week                -5.212e-01  4.904e-02 -10.627  < 2e-16
## requested_dropped_off               4.866e-04  6.816e-04   0.714    0.475
## requested_arrived                  -1.739e-02  1.763e-02  -0.987    0.324
## accepted_arrived                    8.890e-03  1.792e-02   0.496    0.620
## requested_arrived:accepted_arrived -5.359e-06  1.235e-06  -4.339 1.61e-05
##
## (Intercept)                        ***
## weekly_driver_revenue              ***
## mean_rides_per_week                ***
## requested_dropped_off
## requested_arrived
## accepted_arrived
## requested_arrived:accepted_arrived ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.5 on 826 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.199,  Adjusted R-squared:  0.1931
## F-statistic: 34.19 on 6 and 826 DF,  p-value: < 2.2e-16
```

```r
# highest non-interaction insignificant p-value is
# the requested-dropped off timelapse, removing it for model6

model6 <- lm(days_with_lyft ~ (weekly_driver_revenue +
```

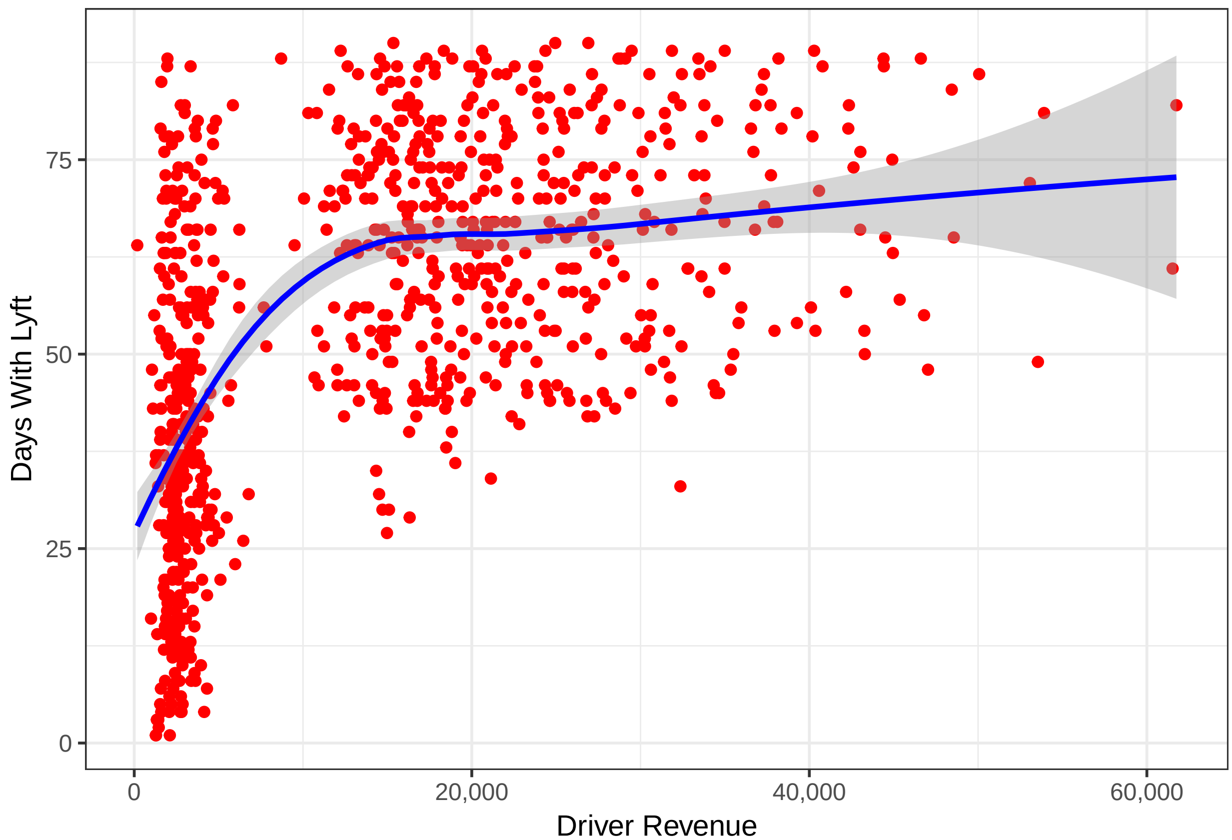
```r
                                mean_rides_per_week +
                                  (requested_arrived *
                                  accepted_arrived)
),
data = driver_revenue2)

summary(model6)
```

```
##
## Call:
## lm(formula = days_with_lyft ~ (weekly_driver_revenue + mean_rides_per_week +
##     (requested_arrived * accepted_arrived)), data = driver_revenue2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.395 -14.839   0.681  14.919  59.295
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          4.950e+01  1.529e+00  32.383  < 2e-16
## weekly_driver_revenue                5.276e-02  4.341e-03  12.154  < 2e-16
## mean_rides_per_week                 -5.209e-01  4.903e-02 -10.626  < 2e-16
## requested_arrived                   -1.601e-02  1.752e-02  -0.914    0.361
## accepted_arrived                     8.609e-03  1.791e-02   0.481    0.631
## requested_arrived:accepted_arrived -5.148e-06  1.199e-06  -4.295 1.96e-05
##
## (Intercept)                        ***
## weekly_driver_revenue              ***
## mean_rides_per_week                ***
## requested_arrived
## accepted_arrived
## requested_arrived:accepted_arrived ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.5 on 827 degrees of freedom
##   (104 observations deleted due to missingness)
## Multiple R-squared:  0.1985, Adjusted R-squared:  0.1936
## F-statistic: 40.96 on 5 and 827 DF,  p-value: < 2.2e-16
```

```r
# despite there being non-significant p-values present for the
# requested-arrived and accepted-arrived timelapses,
# their interaction remains highly statistically significant

# weekly driver_revenue has the largest coefficient in the
# model, making it the biggest factor on how many days a
# driver stays with lyft
# # graphing weekly driver_revenue against days_with_lyft
# # to see if polynomial regression is necessary.
# # a loess fit line is used as a guide
#
driver_revenue2 %>%
  arrange(days_with_lyft) %>%
  ggplot(aes(x = driver_revenue, y = days_with_lyft)) +
```

```
  geom_point(color = "red") +
  geom_smooth(color = "blue",
              method = "loess") +
  xlab("Driver Revenue") +
  ylab("Days With Lyft") +
  scale_x_continuous(labels = comma)
```

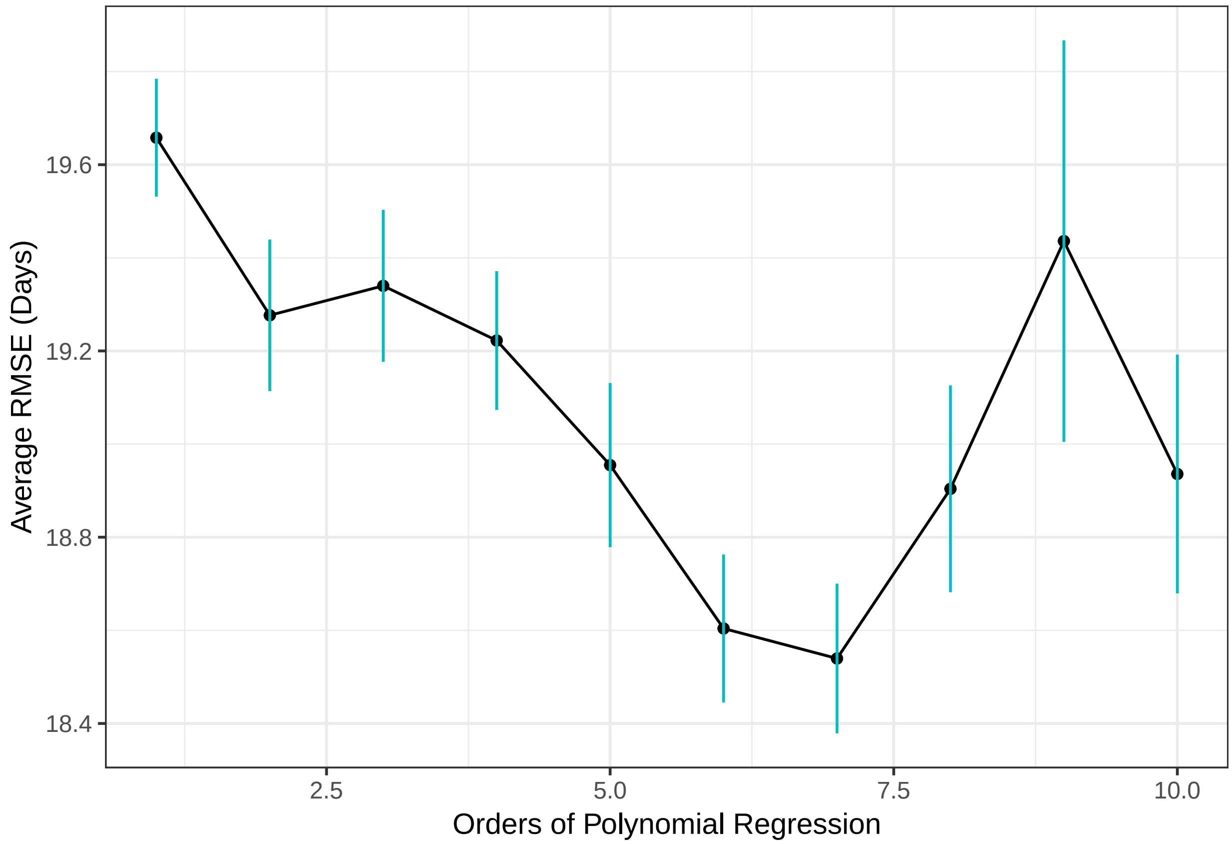
```
## Warning: Removed 104 rows containing non-finite values (stat_smooth).

## Warning: Removed 104 rows containing missing values (geom_point).
```




```
#
# # since the loess fit line has heavy curvature polynomial
# # regression will allow the model to represent the data
# # better
#
#
#
```

```
# in order to determine which order to use for polynomial
# regression, k-fold validation is performed.
# For five loops the data is randomly split into 5 sections.
# In each loop, four sections is used to train the model
# and one section is used to validate the model
# the RMSE value is taken to assess model performance
# and for each polynomial order, the average RMSE is recorded
```

```r
# the model with the lowest RMSE is recommended

sampled_data <- sample_frac(driver_revenue2,1) %>%
  mutate(count = row_number())

fold <- 5
k <- 10
average_rmse <- rep(0,k)
average_se <- rep(0,k)

for(i in 1:k){

  rmse <- rep(0,fold)
  for(j in 1:fold){

  training_setj <- sampled_data %>%
  mutate(cross = (row_number() + 5) %% 5) %>%
  filter(cross != j - 1)

  validation_setj <- sampled_data %>%
  mutate(cross = (row_number() + 5) %% 5) %>%
  filter(cross == j - 1)


  rmse[j] <- rmse(lm(days_with_lyft ~ (poly(weekly_driver_revenue, i, raw = TRUE) +
                              mean_rides_per_week +
                                (requested_arrived *
                                accepted_arrived)
),
data = training_setj), validation_setj)



  }

  average_rmse[i] <- mean(rmse)
  average_se[i] <- sd(rmse)/length(rmse)
}

orders <- c(1:10)

data.frame(orders, average_rmse, average_se) %>%
  ggplot(aes(x = orders)) +
  geom_point(aes(y = average_rmse), color = 'black') +
  geom_line(aes(y = average_rmse), color = 'black') +
  geom_linerange(aes(ymin = average_rmse - average_se,
                     ymax = average_rmse + average_se), color = '#00BFC4') +
  ylab("Average RMSE (Days)") +
  xlab("Orders of Polynomial Regression")
```




```
match(min(average_rmse), average_rmse)
```

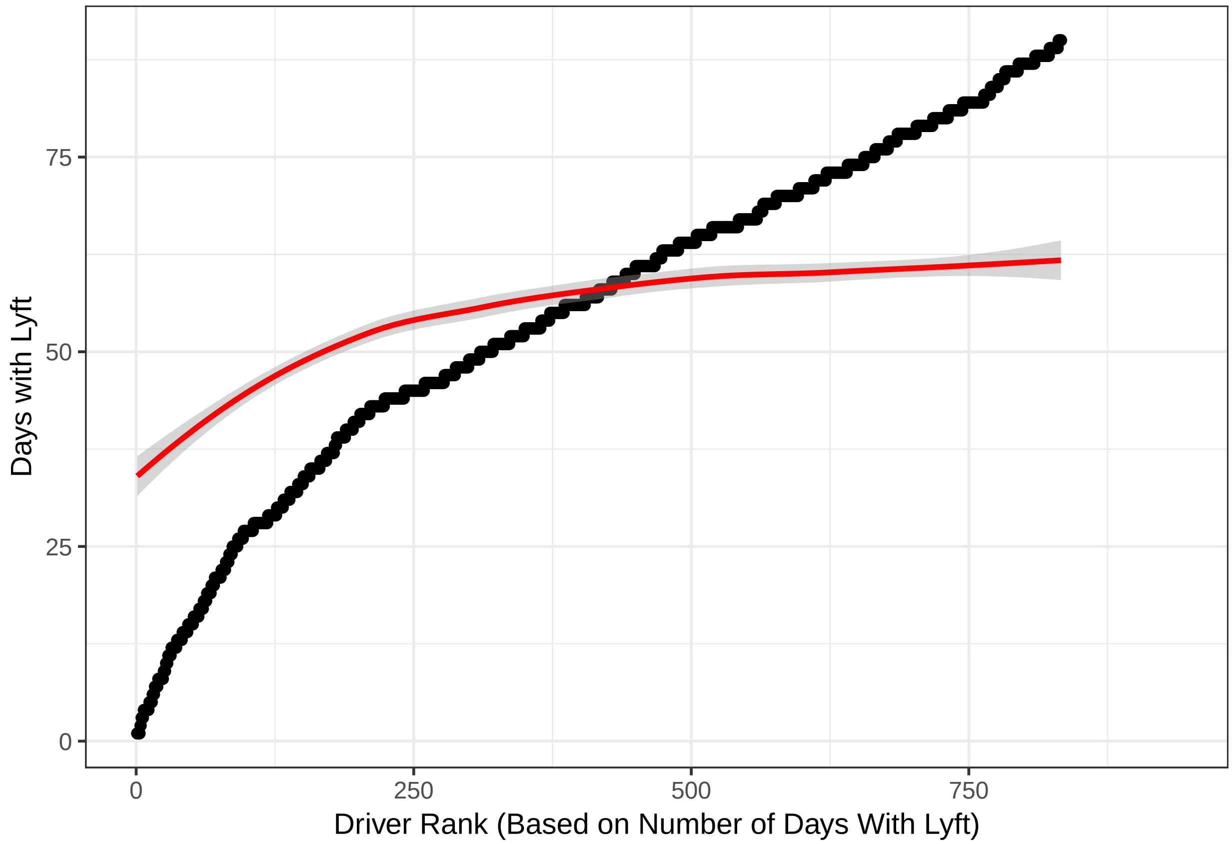
```
## [1] 7
```

```
average_rmse[match(min(average_rmse), average_rmse)]
```

```
## [1] 18.53955
```

```
driver_revenue2 %>%
  arrange(days_with_lyft) %>%
  mutate(rank = row_number()) %>%
  add_predictions(lm(days_with_lyft ~ (poly(weekly_driver_revenue, 7, raw = TRUE) +
                              mean_rides_per_week +
                                (requested_arrived *
                                accepted_arrived)
),
data = driver_revenue2)) %>%
  ggplot(aes(x = rank, y = days_with_lyft)) +
  geom_point() +
  geom_smooth(aes(y = pred), color = "red") +
  ylab("Days with Lyft") +
  xlab("Driver Rank (Based on Number of Days With Lyft)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Removed 104 rows containing non-finite values (stat_smooth).

## Warning: Removed 104 rows containing missing values (geom_point).
```




```
# In order to calculate average driver's lifetime, we
# applied a total of 180 variable scenarios over the
# timeframe of 55 days, and
# attempted to account for the outcomes of the total amount
# of interactions over a span of 45 years (the expected
# work duration of any individual from the age of 20 - 65)

driver_revenue2 %>%
  add_predictions(lm(days_with_lyft ~ (poly(weekly_driver_revenue, 7, raw = TRUE) +
                                         mean_rides_per_week +
                                           (requested_arrived *
                                           accepted_arrived)
),
data = driver_revenue2)) %>%
  summarize(projected_lifetime = mean(pred, na.rm = TRUE)) %>%
  mutate(projected_lifetime = projected_lifetime * 180)
```

```
## # A tibble: 1 x 1
##   projected_lifetime
##                <dbl>
## 1              9938.
```

```
# Projected Driver Lifetime: ~9938 days or ~27 years
```