Citibike Prediction

Sara Goldberger

2023-06-19

Description

In this report, I use data of Citibike trips from the year 2014 to create a predictive model of the total trips per day for any given year.

The data consists of ymd (day of year as a date type), num_trips (total number of trips on that day), date (the date as a double type), prcp (precipitation), snwd (snow depth), snow (snowfall), tmax (maximum temperature), and tmin (minimum temperature).

These features will be analyzed to determine the relationships and dependencies.

The data will also be modified to include holidays, weekday, and month.

Load the Data

The trips data used is sourced from Citibike published data and weather data from belvedere tower in central park, stored in a file called 'trips_per_day.tsv'.

```
trips_per_day <- read_tsv('trips_per_day.tsv')
```

```
## Rows: 365 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl  (7): num_trips, date, prcp, snwd, snow, tmax, tmin
## date (1): ymd
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(trips_per_day)
```

```
## # A tibble: 6 x 8
##   ymd        num_trips     date  prcp  snwd  snow  tmax  tmin
##   <date>         <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2014-01-01      6059 20140101  0      0     0     3.3   2.4
## 2 2014-01-02      8600 20140102  0.33   0     3.1   3.3   1.8
## 3 2014-01-03      1144 20140103  0.29   5.9   3.3   1.8   0.9
## 4 2014-01-04      2292 20140104  0      5.9   0     2.9   0.8
## 5 2014-01-05      2678 20140105  0.14   3.9   0     4     2.7
## 6 2014-01-06      9510 20140106  0.36   1.2   0     5.5   1.9
```

Part A: Analyzing Additional Predictors

The data loaded does not consider whether the trip day is a holiday, which day of the week it is, or which month it is in. These may be important predictors, so I will consider each and decide whether it is significant enough to add to the data.

I analyze the predictors, find the significance, and modify the data accordingly.

Holiday

Analyze the data

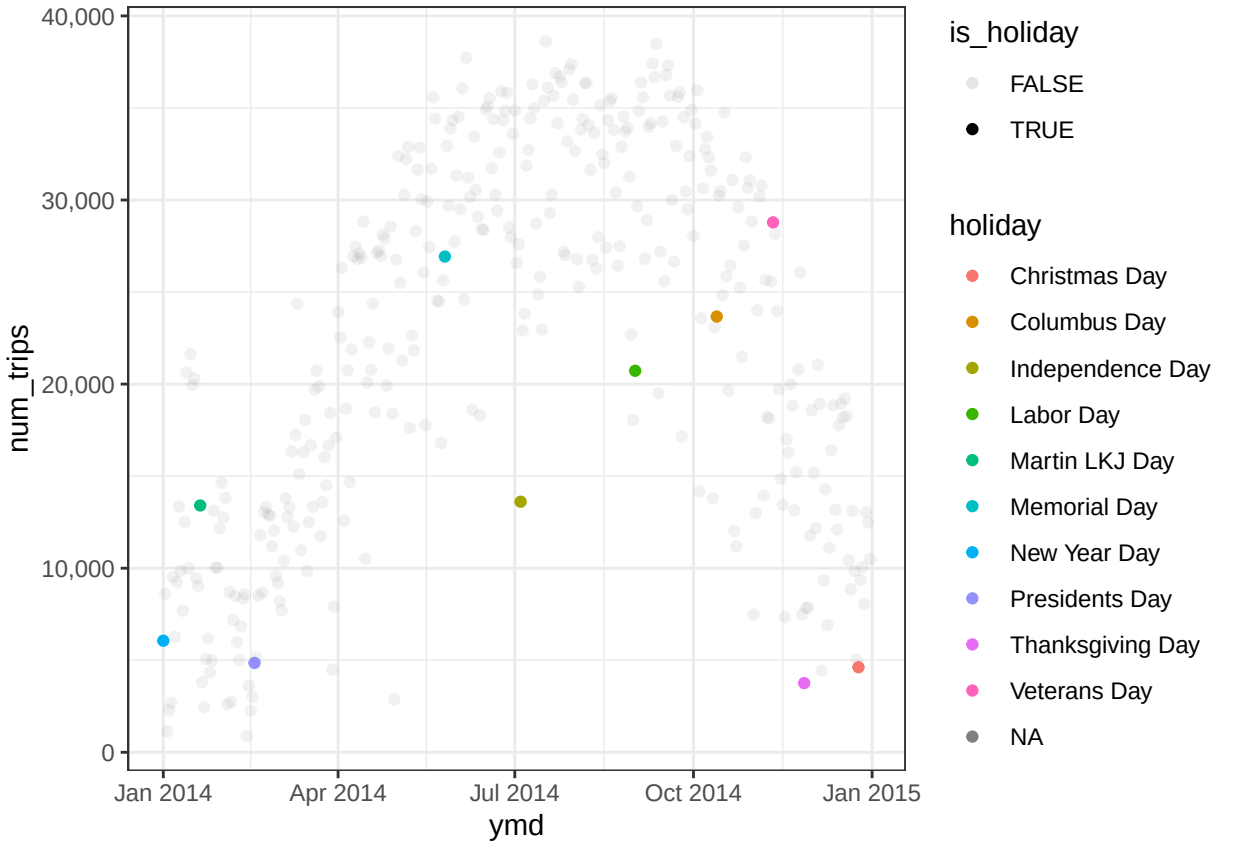
I load data from US_holidays_2014.txt, a text file with the major US holidays. I plot the number of trips per day and highlight the holidays in order to get a feel for where these holiday dates are falling among all the other dates.

```
# Load in the holidays data
holidays <- read_delim('US_holidays_2014.txt', delim = ,)
```

```
## Rows: 10 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): holiday
## date (1): ymd
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Join the trips and holiday data
trips_holidays <- left_join(trips_per_day, holidays, by = 'ymd') %>%
  mutate(month = month(ymd), is_holiday = (ymd %in% holidays$ymd))

# Plot the data
trips_holidays %>%
  group_by(holiday) %>%
  ggplot(aes(x = ymd, y = num_trips, color = holiday, alpha = is_holiday)) +
  geom_point() +
  scale_y_continuous(label = comma)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



Find significant holidays

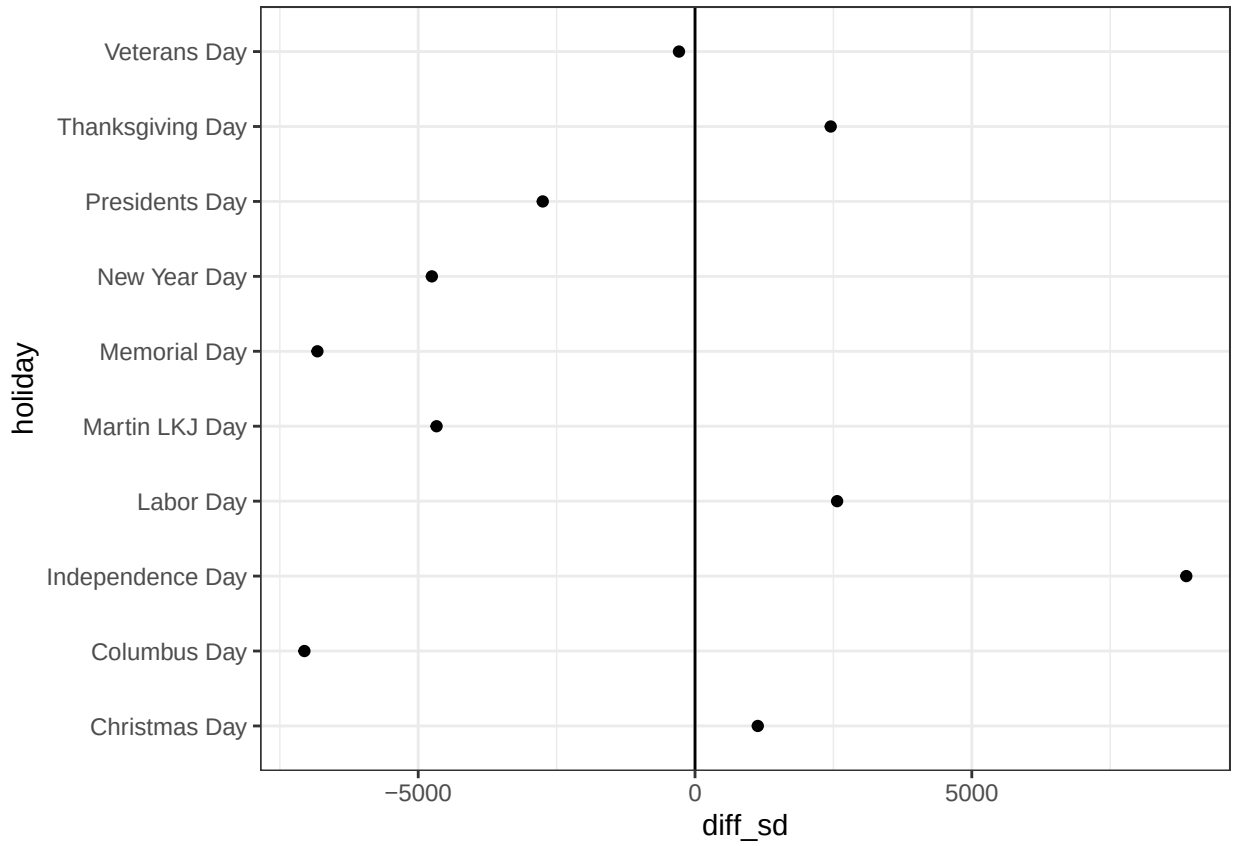
There are a few holidays that clearly fall outside the normal trend, like Independence Day and Labor Day.

To quantify which holidays are significant, I find the average trip count per month, the standard deviation per month, and the difference between each holiday trip count and the average trip count of the month they fall in. Then I take those differences and subtract off one and a half of their month's standard deviation to find how much more the significance of the holiday trip count deviation (if any). If the difference is large, it means that the holiday probably had an influence on trip count, and it should be included in the model. If there is a negative difference, it means the holiday is probably not significant.

```
# Find the difference between the holiday trip count deviation from the monthly average and the standar
trips_holidays_sd <- trips_holidays %>%
  group_by(month) %>%
  mutate(mo_avg = mean(num_trips), mo_sd = sd(num_trips)) %>%
  filter(!is.na(holiday)) %>%
  mutate(diff_avg = num_trips-mo_avg) %>%
  mutate(diff_sd = abs(diff_avg)-(mo_sd*1.5))

summary(trips_holidays_sd$mo_sd)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3949    5310    5590    5858    6551    7623
```

```r
# Plot the deviation of the difference of each holiday trip count from the standard deviation of the mo
trips_holidays_sd %>%
  ggplot(aes(x = diff_sd, y = holiday)) +
  geom_point() +
  geom_vline(xintercept = 0)
```



Add significant holidays to the data

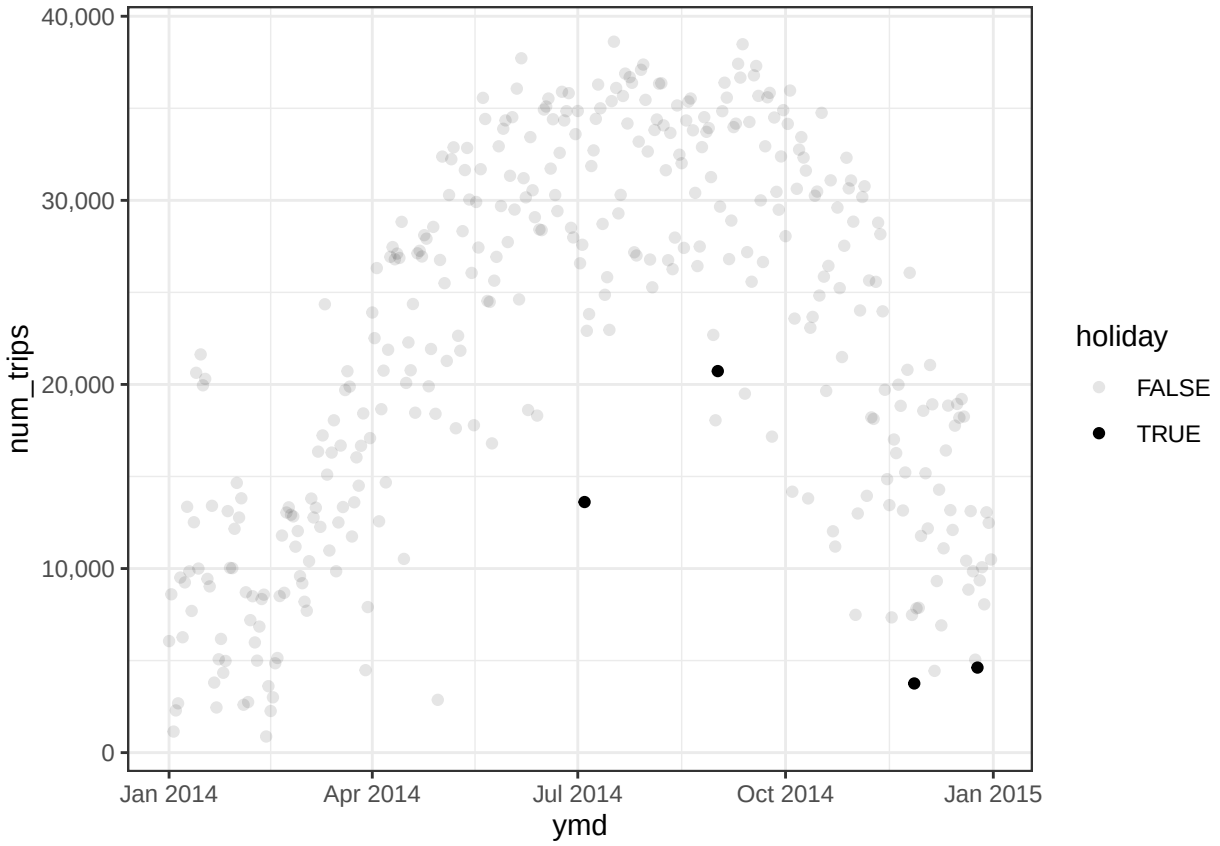
Keep only the holidays that have significantly greater deviations from the average trip count of the month.

All of the holidays found to be significant had lower trip counts than the average of the month. As such, a single column can be added to the trips data to signify whether the day is a holiday or not, and a single variable with a negative coefficient will be implemented in the model to signify a decrease in trips if it is a significant holiday.

```r
# Keep all the holidays to the right of the line, i.e significant differences.
holidays_best <- trips_holidays_sd %>%
  filter(diff_sd > 0)

# Add a logical column to the data to mark whether the day is a holiday (TRUE) or not (FALSE).
trips_per_day <- trips_per_day %>%
  mutate(holiday = (ymd %in% holidays_best$ymd)) %>%
  replace_na(list(holiday=F))

# Now replot the data
```

```
trips_per_day %>%
  ggplot(aes(x = ymd, y = num_trips, alpha = holiday)) +
  geom_point() +
  scale_y_continuous(label = comma)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



Notice that the holidays we are left with are fairly far away from the other points in the month.

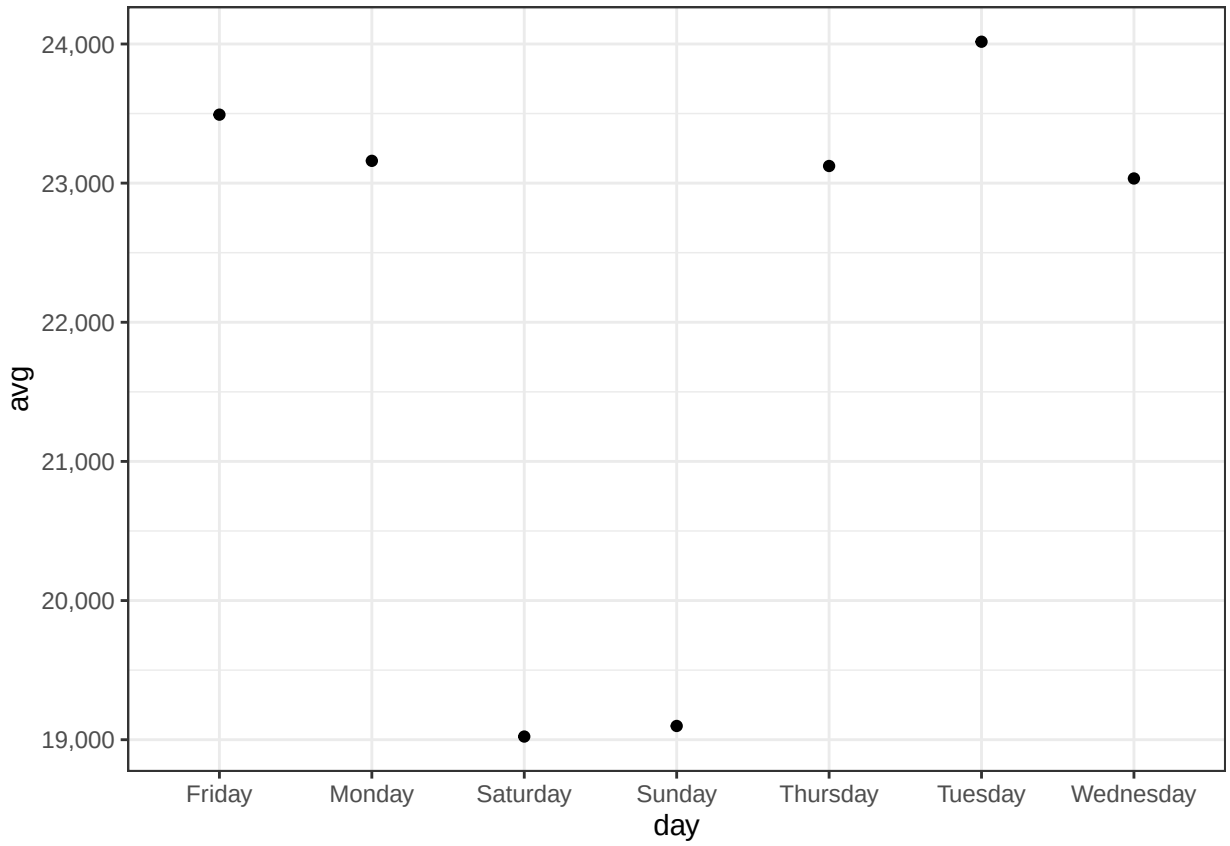
Day of the Week

Analyze the data

The number of trips may be affected by day of the week. Split data by day of the week and see if the plot of number of trips based on day of the week is significant.

```
# Add a column to mark day of week
trips_weekdays <- trips_per_day %>%
  mutate(day = weekdays(ymd))

# Plot the average number of trips based on day of the week
trips_weekdays %>%
  group_by(day) %>%
  summarise(avg = mean(num_trips)) %>%
```

```
  ggplot(aes(x = day, y = avg)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



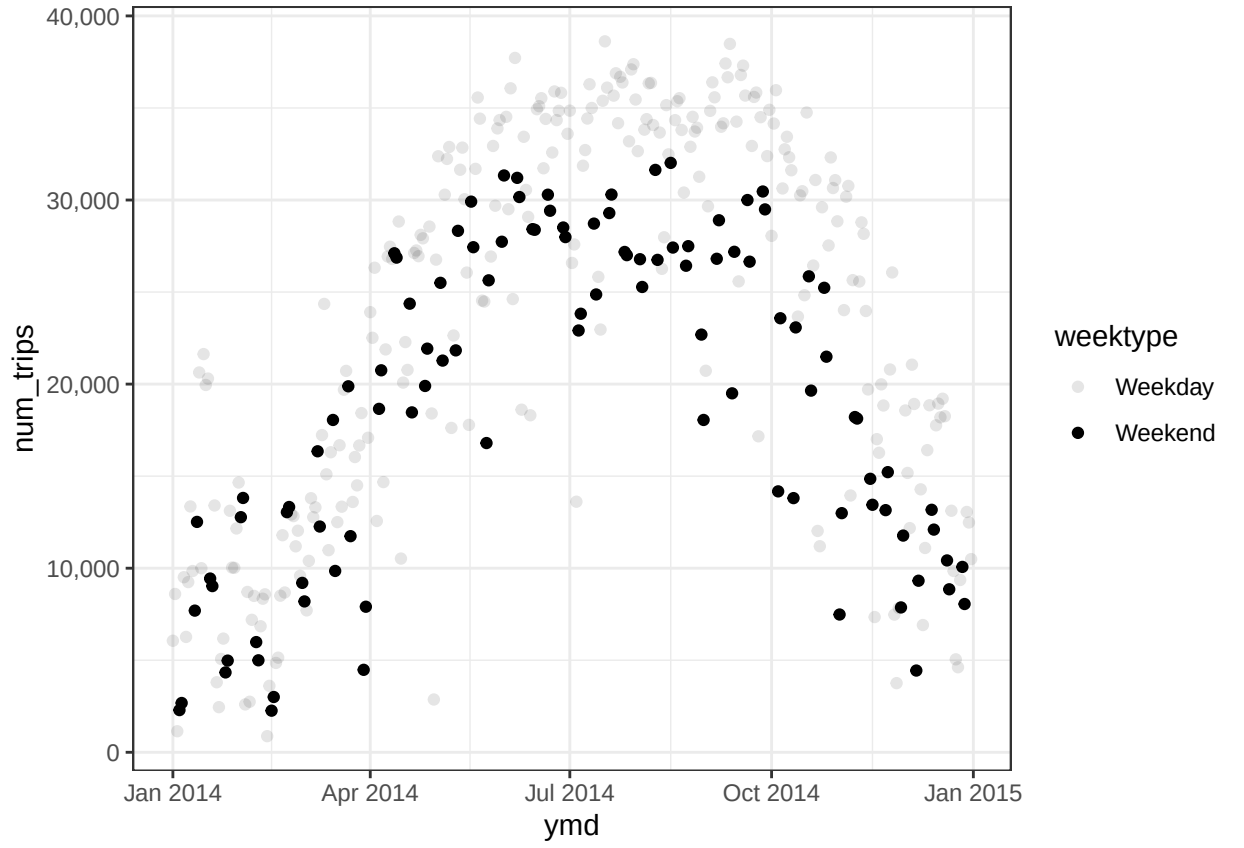
The data shows that if the day of the week is a weekend versus a weekday, number of trips drop considerably.

Add week type to the data

Add a column to the data to signify whether the day is a weekday or weekend.

```
trips_weektype <- trips_weekdays %>%
  mutate(weektype = ifelse(day=="Saturday"|day=="Sunday", "Weekend", "Weekday"))

# Now replot the data, and notice if weekend has on average fewer trips than weekdays
trips_weektype %>%
  ggplot(aes(x = ymd, y = num_trips, alpha = weektype)) +
  geom_point() +
  scale_y_continuous(label = comma)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



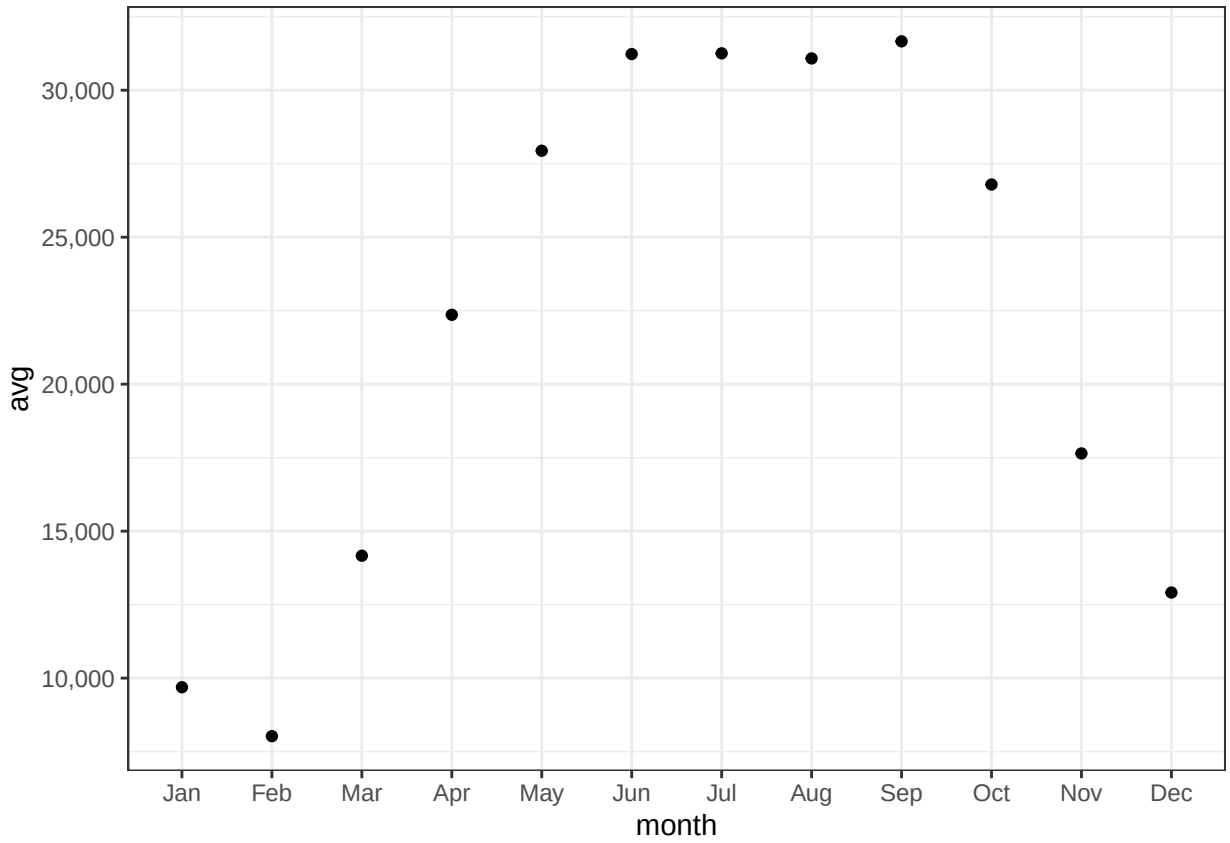

```
# Update the data
trips_per_day <- trips_weektype %>% select(-day)
```

Month

Analyze the data

The number of trips may be affected by month. Split the data by month and see if the plot of number of trips based on month is significant.

```
# Add in a column to mark day of week
trips_months <- trips_per_day %>%
  mutate(month = month(ymd, label = T))

# Plot the average number of trips based on day of the week
trips_months %>%
  group_by(month) %>%
  summarise(avg = mean(num_trips)) %>%
  ggplot(aes(x = month, y = avg)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



The data shows that the summer months tend to peak.

Add month to the data

```
# Update the data
trips_per_day <- trips_months
```

Part B: Split the Data

> Split the data into randomly selected training, validation, and test sets, with 90% of the data for training and validating the model, and 10% for a final test set (to be used only once, towards the end of this exercise).

```
set.seed(18)

num_days <- nrow(trips_per_day)
frac_model <- 0.9
num_model <- floor(num_days * frac_model)

# randomly sample rows for the model set
ndx <- sample(1:num_days, num_model, replace=F)
```

```
# used for the model
trips_per_day_model <- trips_per_day[ndx, ] %>% arrange(ymd)

# used for the final test set
trips_per_day_final_test <- trips_per_day[-ndx, ]

head(trips_per_day_model)
```

```
## # A tibble: 6 x 11
##   ymd        num_trips     date  prcp  snwd  snow  tmax  tmin holiday weektype
##   <date>         <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>   <chr>
## 1 2014-01-01      6059 20140101  0      0     0     3.3   2.4 FALSE   Weekday
## 2 2014-01-02      8600 20140102  0.33   0     3.1   3.3   1.8 FALSE   Weekday
## 3 2014-01-03      1144 20140103  0.29   5.9   3.3   1.8   0.9 FALSE   Weekday
## 4 2014-01-04      2292 20140104  0      5.9   0     2.9   0.8 FALSE   Weekend
## 5 2014-01-05      2678 20140105  0.14   3.9   0     4     2.7 FALSE   Weekend
## 6 2014-01-06      9510 20140106  0.36   1.2   0     5.5   1.9 FALSE   Weekday
## # i 1 more variable: month <ord>
```

```
head(trips_per_day_final_test)
```

```
## # A tibble: 6 x 11
##   ymd        num_trips     date  prcp  snwd  snow  tmax  tmin holiday weektype
##   <date>         <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>   <chr>
## 1 2014-01-16     19953 20140116  0      0     0     4.2   3.6 FALSE   Weekday
## 2 2014-02-03      2600 20140203  1.17   1.2   8     4.3   2.7 FALSE   Weekday
## 3 2014-02-13       876 20140213  1.78  11.8   9.5   3.6   2.4 FALSE   Weekday
## 4 2014-03-27     16666 20140327  0      0     0     4.4   2.2 FALSE   Weekday
## 5 2014-04-06     20750 20140406  0      0     0     6.1   3.6 FALSE   Weekend
## 6 2014-04-08     21884 20140408  0.34   0     0     6.4   4.6 FALSE   Weekday
## # i 1 more variable: month <ord>
```

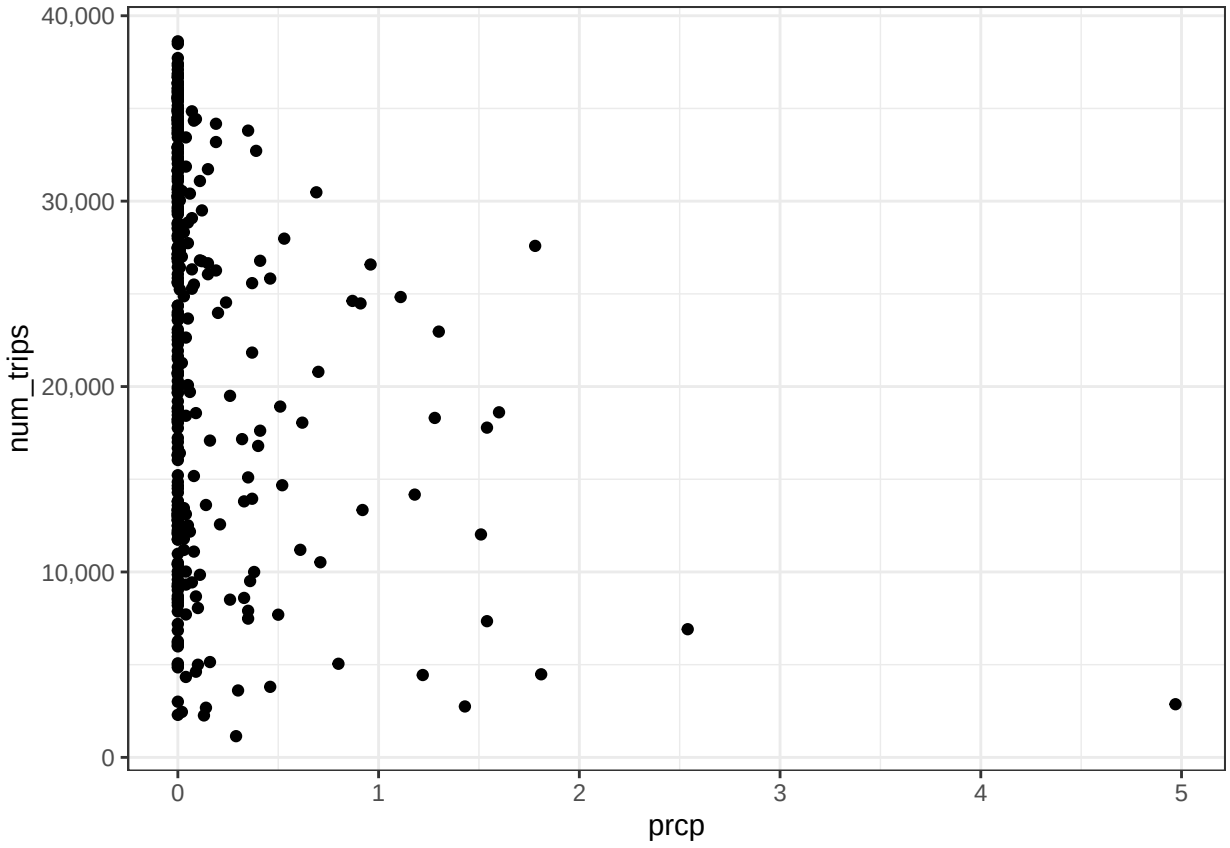
Part C: Analyze the Features

> Now that I have separated the training data I can begin building my model. I start by analyzing my features.

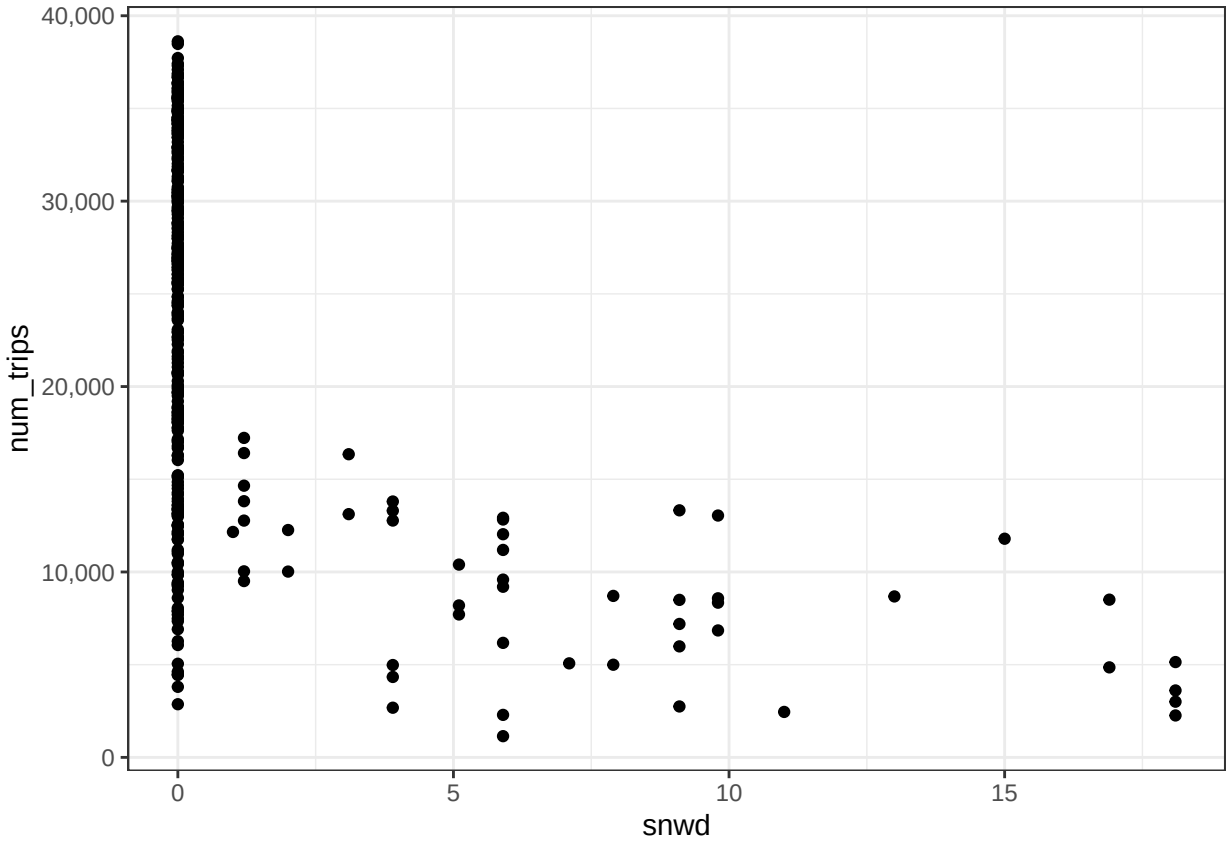
Plot the Features

Plot num_trips against each feature and analyze the relationship.

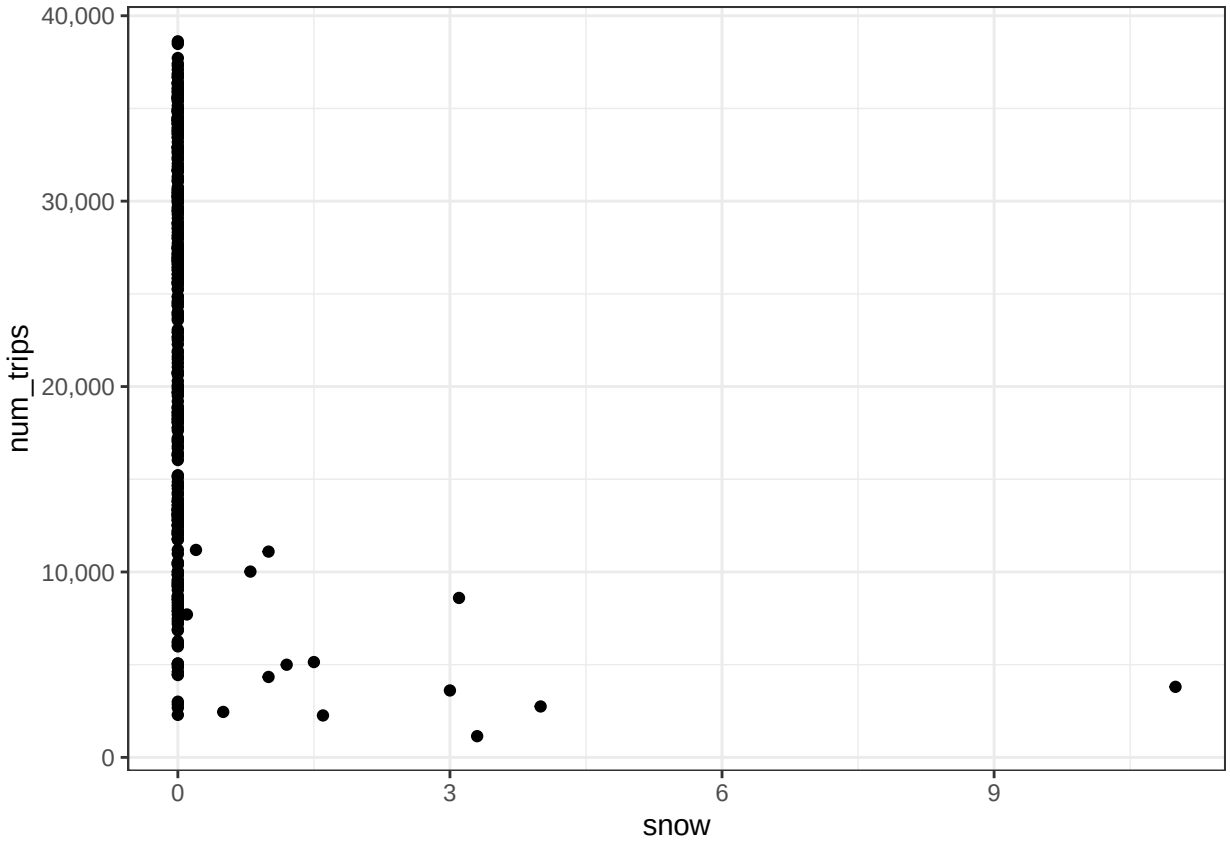
```
# features: prcp, snwd, snow, tmax, tmin, holiday, weektype, month

ggplot(trips_per_day_model, aes(x = prcp, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



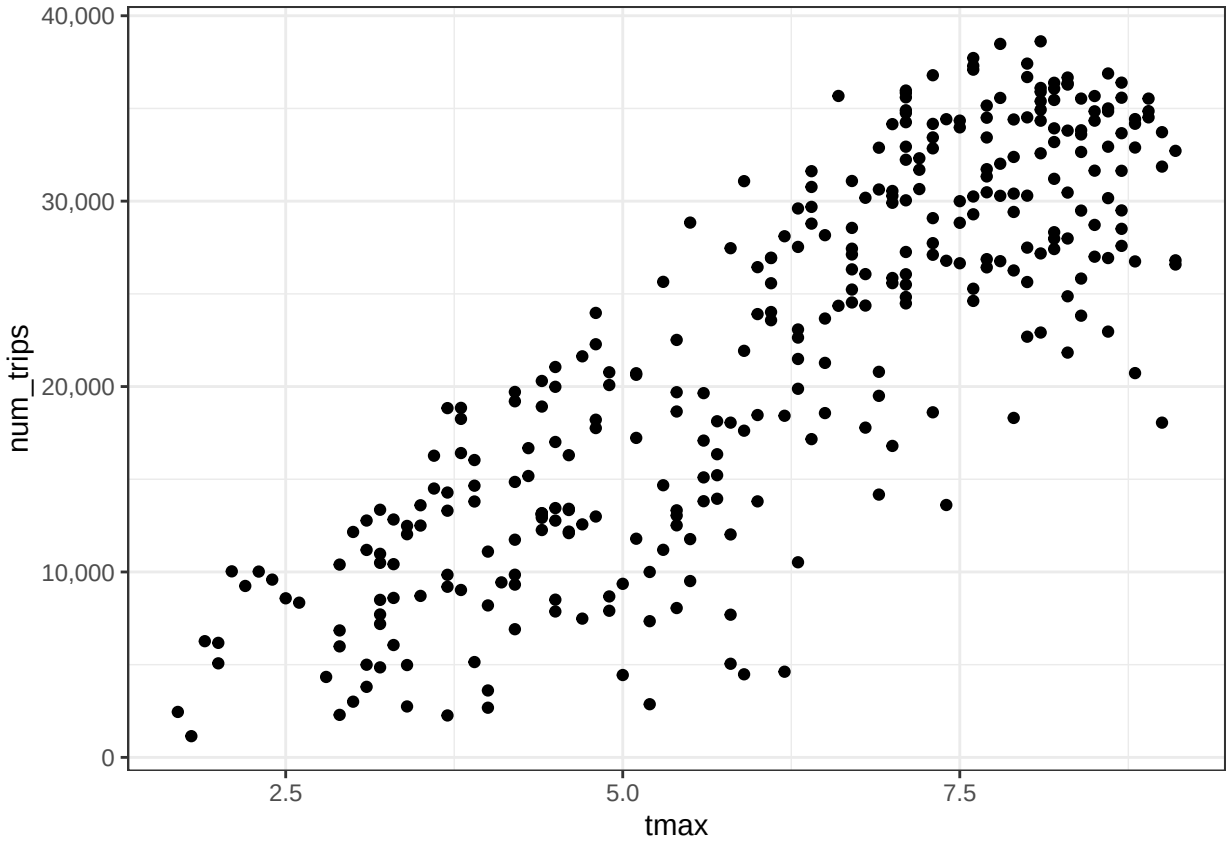

```
ggplot(trips_per_day_model, aes(x = snwd, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



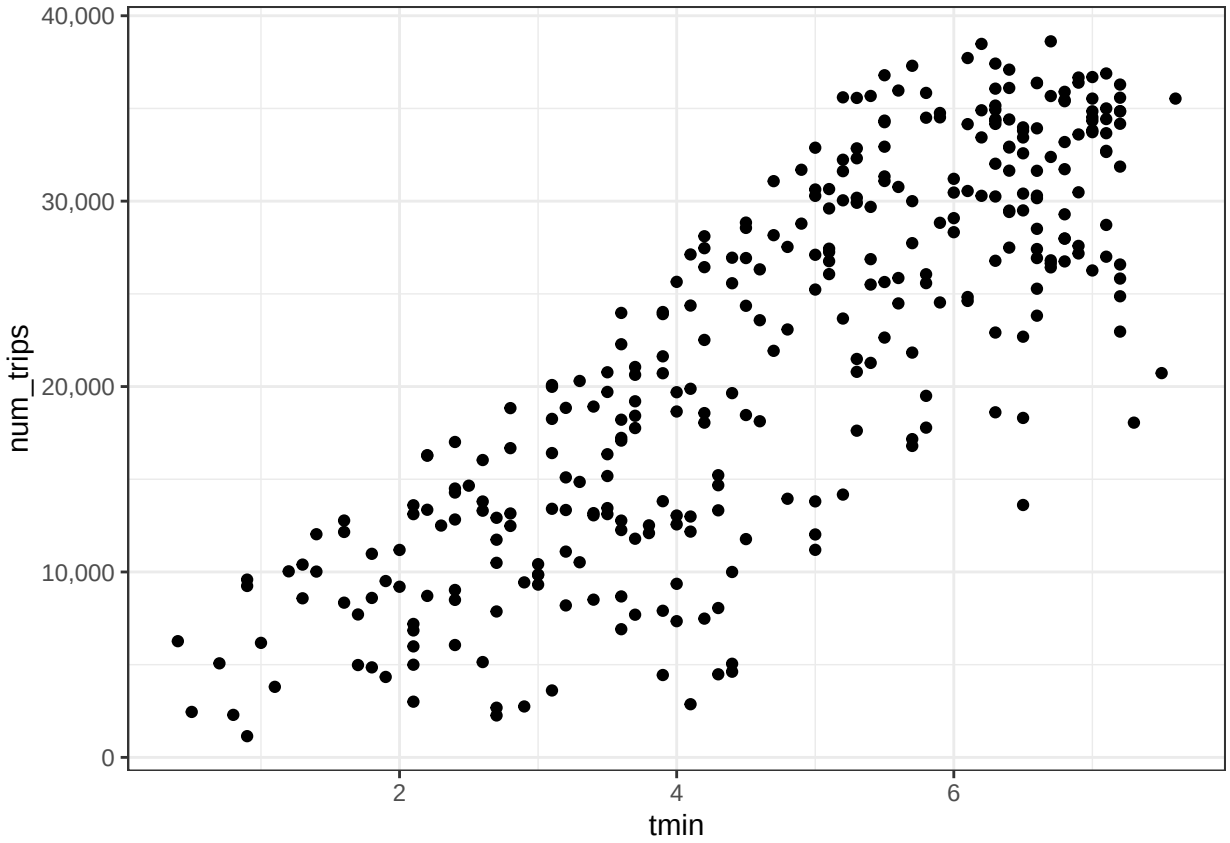

```
ggplot(trips_per_day_model, aes(x = snow, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



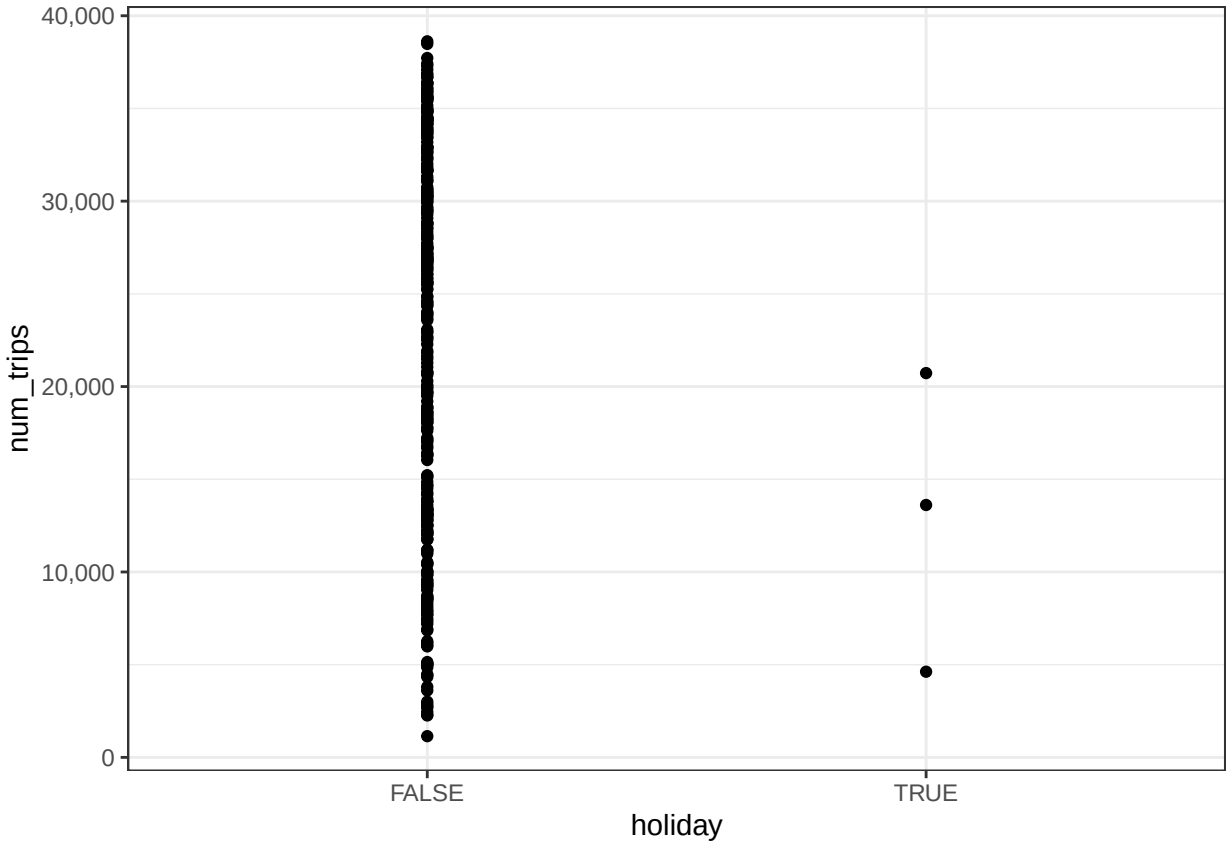

```
ggplot(trips_per_day_model, aes(x = tmax, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



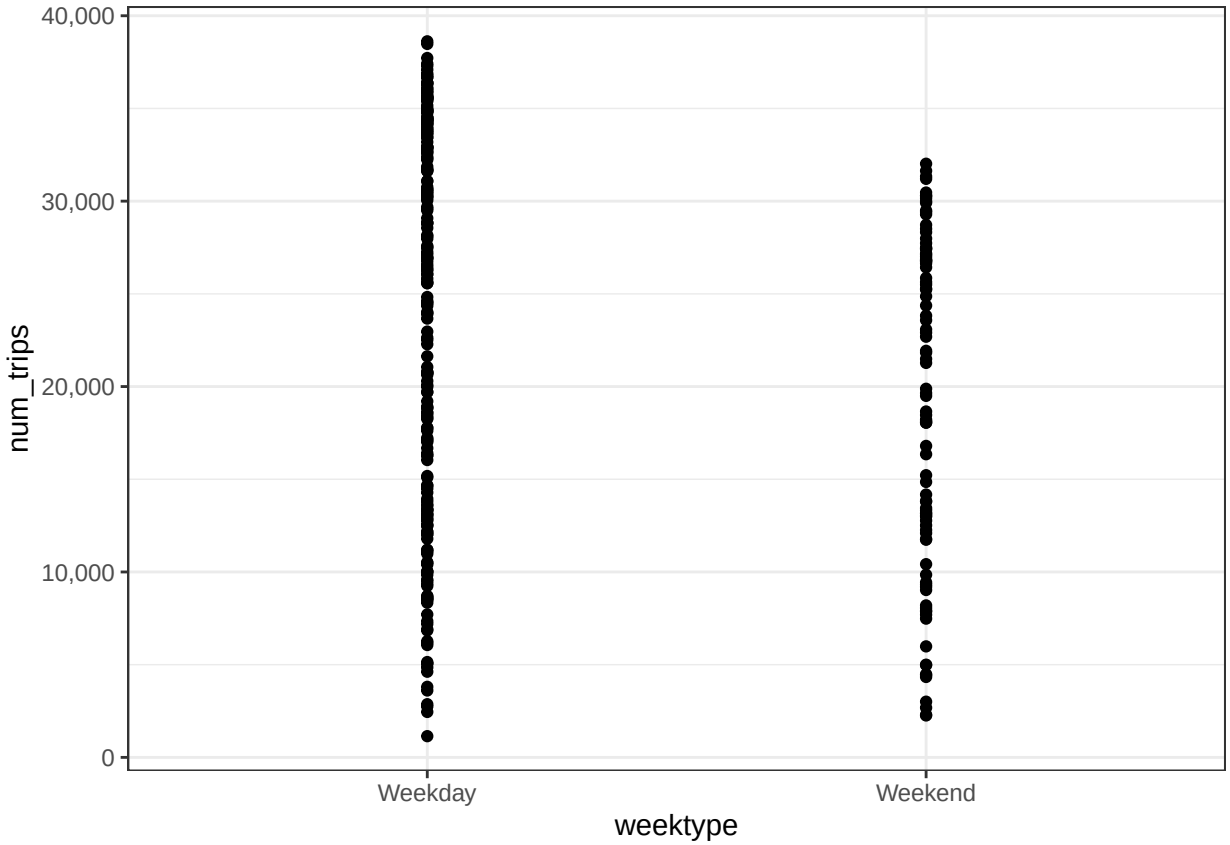

```
ggplot(trips_per_day_model, aes(x = tmin, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



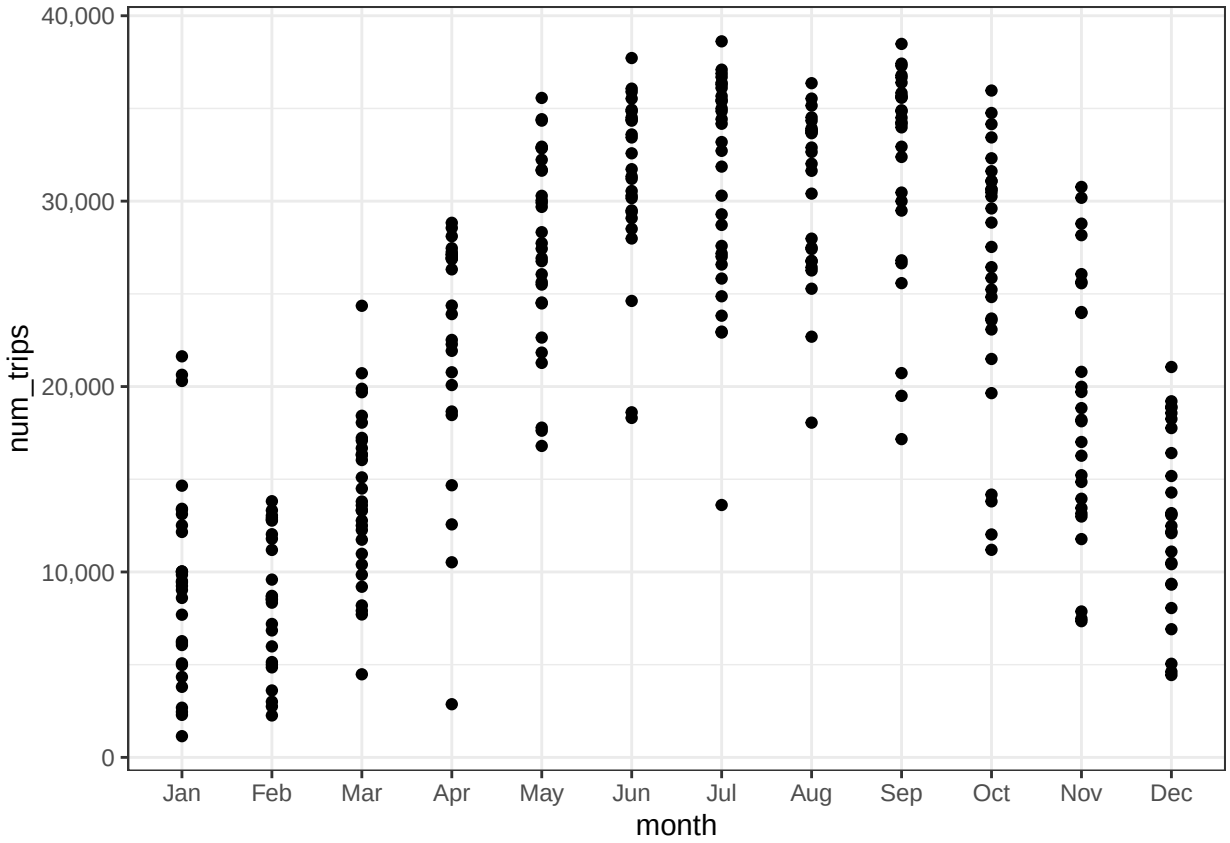

```
ggplot(trips_per_day_model, aes(x = holiday, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```




```
ggplot(trips_per_day_model, aes(x = weektype, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```




```
ggplot(trips_per_day_model, aes(x = month, y = num_trips)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



Analyze the plots

The plots for tmax, tmin, and month show a strong correlation, and further on I explore whether the relationship is linear or polynomial.

Since holiday and weektype are logical columns, the plots are not too informative. However, based on the analysis of these features previosuly I am confident that they are significant.

The plots for prcp, snwd, and snow are interesting. There is no clear relationship to trip count, which indicates to me that I may be dealing with features that are interdependent.

Logically, I consider what correlations there may be in these features:

If it is raining (prcp) we expect trip counts to drop. But if it is raining and it is cold (tmin) then trip count may drop more than if it is raining and it is hot (tmax)

If it is snowing (snow) we would expect trip counts to drop if there is enough snow that there is snow on the ground (snwd) and if the snow will last a while i.e. if it is very cold (tmin).

If there is snow on the ground (snwd) but it isn't very cold, maybe the snow will melt and people are fine to bike. But if there is snow on the ground (and there is snow falling) and it is very cold, the ground will remain covered in snow longer and we would expect trips to drop.

Based on these observations, these are the relationships I will explore: prcp & tmin, prcp & tmax, snwd & tmin, snow & tmin, snwd & snow.

Part D: Compare Models

I plot the simplest model, and build on it using my observations about the features as well as k-fold cross-validation to compare the models as I build on them, quantifying the performance of the models using root mean-squared error.

K-fold Cross-validation

Reshuffle the data and add folds

```
set.seed(42)
num_folds <- 5
num_days <- nrow(trips_per_day_model)
num_train <- floor(num_days * (1-(1/num_folds)))

ndx <- sample(1:num_days, num_train, replace=F)

trips_per_day_model <- trips_per_day_model[ndx, ] %>%
  mutate(fold = (row_number() %% num_folds) + 1)
```

Create a function for the k-fold plotting to avoid redundancy later in the code

```
plot_function <- function(K, avg_validate_err, se_validate_err, title) {
# plot the validate error, highlighting the value of k with the lowest average error
plot_data <- data.frame(K, avg_validate_err, se_validate_err)
ggplot(plot_data, aes(x=K, y=avg_validate_err)) +
  geom_pointrange(aes(ymin=avg_validate_err - se_validate_err,
                      ymax=avg_validate_err + se_validate_err,
                      color=avg_validate_err == min(avg_validate_err))) +
  geom_line(color = "red") +
  scale_x_continuous(breaks=1:12) +
  theme(legend.position="none") +
  xlab(title) +
  ylab('RMSE on validation data')
}
```

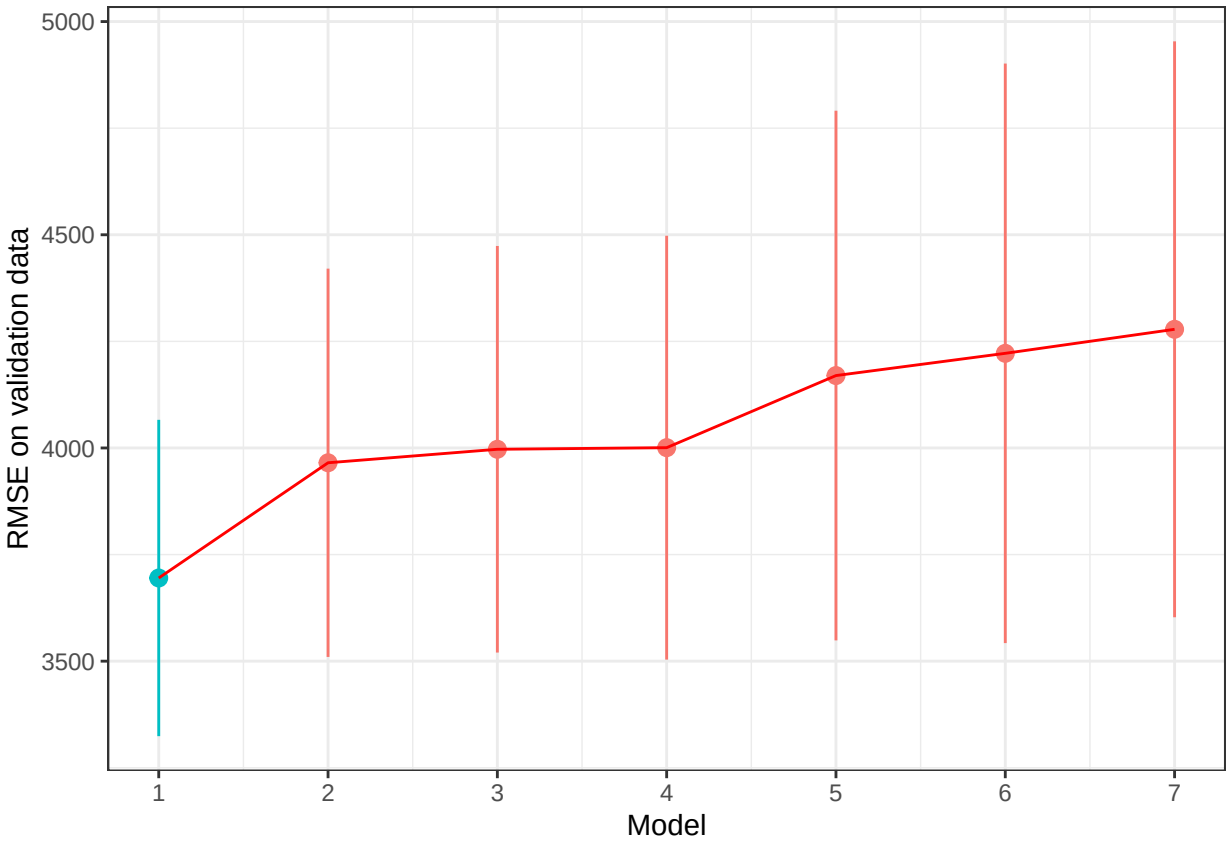
Create a function for the k-fold model cross-validation (this doesn't work for k-fold polynomial degree testing) to avoid redundancy later in the code.

```
kfold_model_test <- function(model_list) {
  K <- 1:length(model_list)
  avg_validate_err <- c()
  se_validate_err <- c()
  for (k in K) {

    # do 5-fold cross-validation within each value of k
    validate_err <- c()
    for (f in 1:num_folds) {
      # fit on the training data
      trips_per_day_train <- filter(trips_per_day_model, fold != f)
      model <- lm(model_list[k], data=trips_per_day_train)
```

```
      # evaluate on the validation data
      trips_per_day_validate <- filter(trips_per_day_model, fold == f)
      validate_err[f] <- sqrt(mean((predict(model, trips_per_day_validate) - trips_per_day_validate$num
    }

    # compute the average validation error across folds
    # and the standard error on this estimate
    avg_validate_err[k] <- mean(validate_err)
    se_validate_err[k] <- sd(validate_err) / sqrt(num_folds)
  }
  plot_function(K, avg_validate_err, se_validate_err, 'Model')
}
```

Test simple models

```
model_list <- c("num_trips ~ prcp + snwd + snow + tmax + tmin + holiday + weektype +
                  month",
                "num_trips ~ prcp*tmin + prcp*tmax + snwd*tmin + snow*tmin +
                  snow*snwd + holiday + weektype + month",
                "num_trips ~ prcp*tmin + prcp*tmax + snow*snwd*tmin + holiday +
                  weektype + month",
                "num_trips ~ prcp*tmin + prcp*tmax + snwd*tmin + snow*tmin +
                  snow*snwd + holiday + weektype + month + tmin*tmax",
                "num_trips ~ prcp*tmin + prcp*tmax + snwd*tmin + snow*tmin +
                  snow*snwd + holiday + weektype + month + snwd*tmax + snow*tmax",
                "num_trips ~ prcp*tmin + prcp*tmax + snwd*tmin + snow*tmin +
                  snow*snwd + holiday + weektype + month + snwd*tmax + snow*tmax +
                  tmin*tmax",
                "num_trips ~ prcp*tmin + prcp*tmax + snow*snwd*tmin + holiday +
                  weektype + month + snwd*tmax + snow*tmax + tmin*tmax")
kfold_model_test(model_list)
```



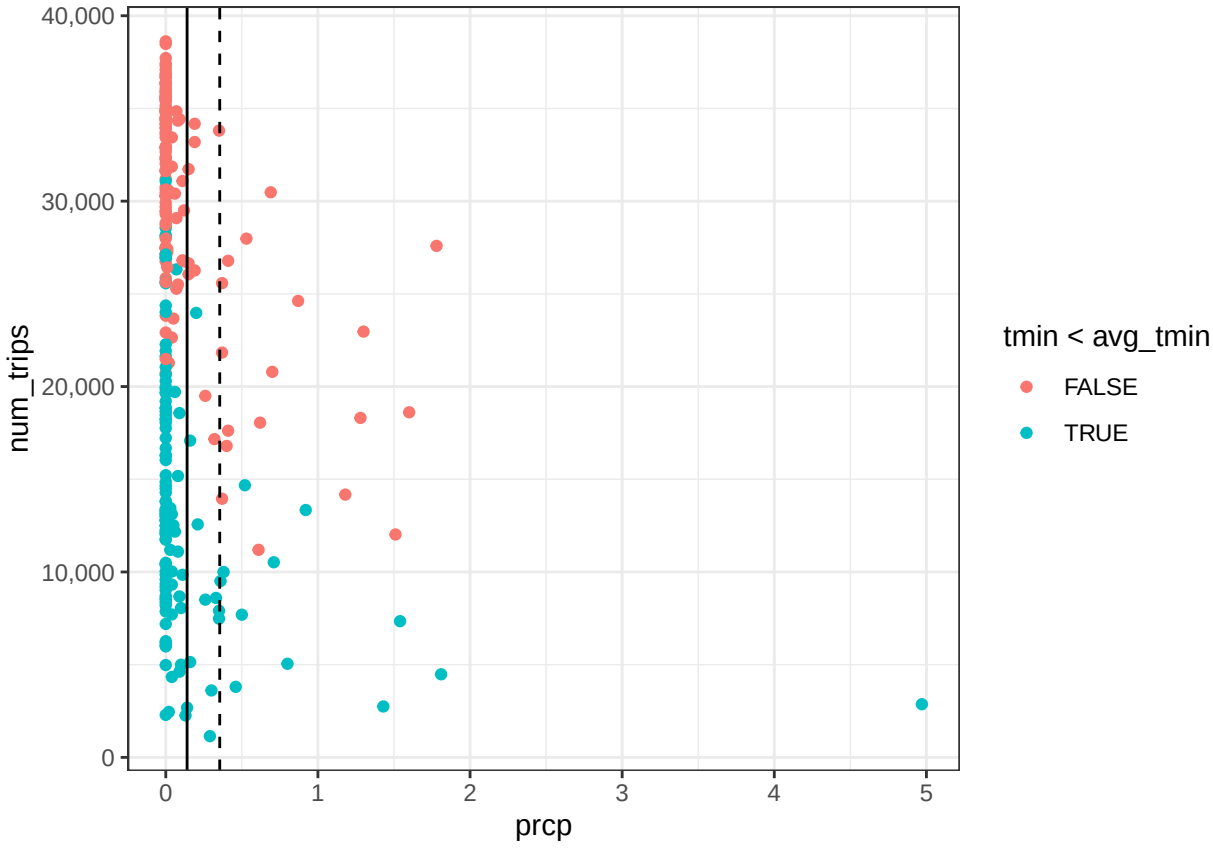
Model number one and two give us the best RMSE, but I am doubtful that the first model is really the best so I will proceed with the second.

Analyze prcp relationship with tmin and tmax

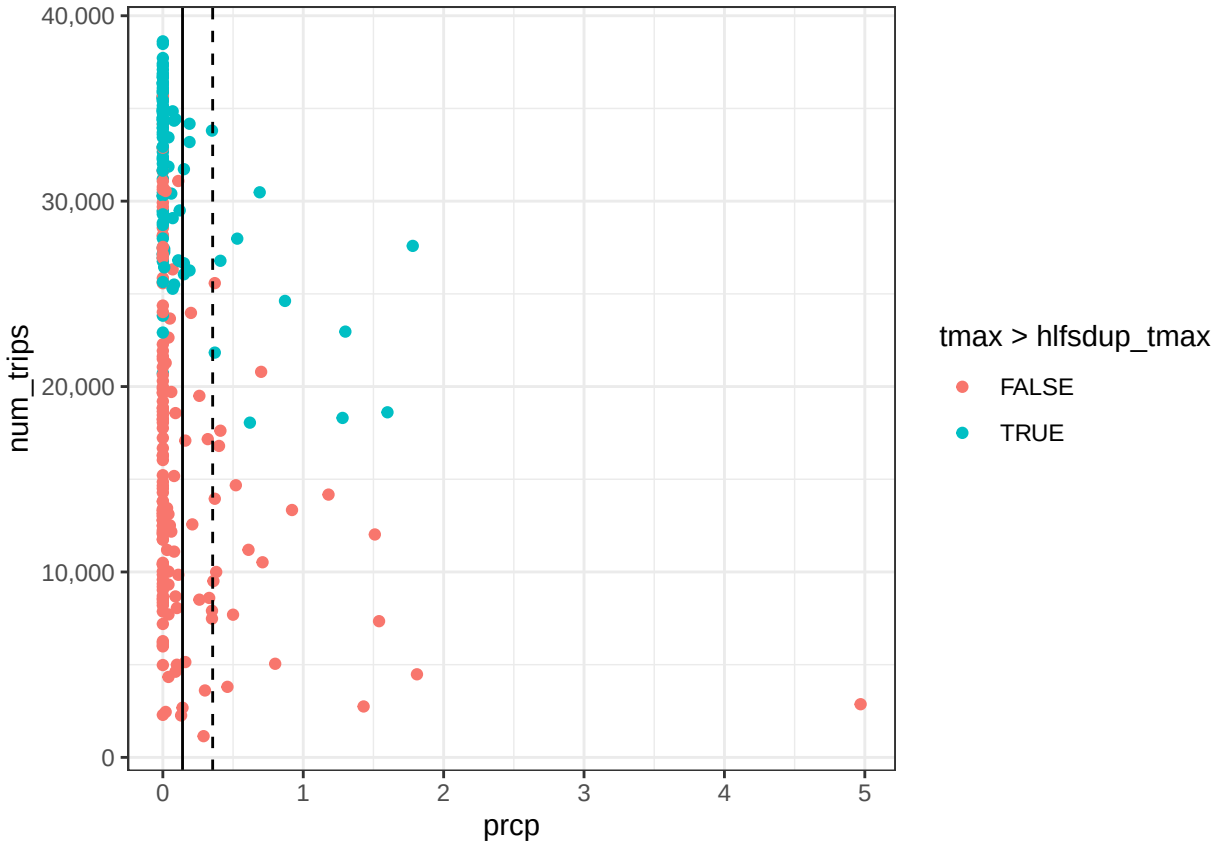
Now I play with the relationship between prcp with tmin and tmax.

Previously I hypothesized that high prcp and low tmin will decrease trip count a lot, while high prcp and high tmax will decrease trip count less. But what constitutes high prcp? I explore two different benchmarks for prcp; average prcp and half a standard deviation above the average prcp.

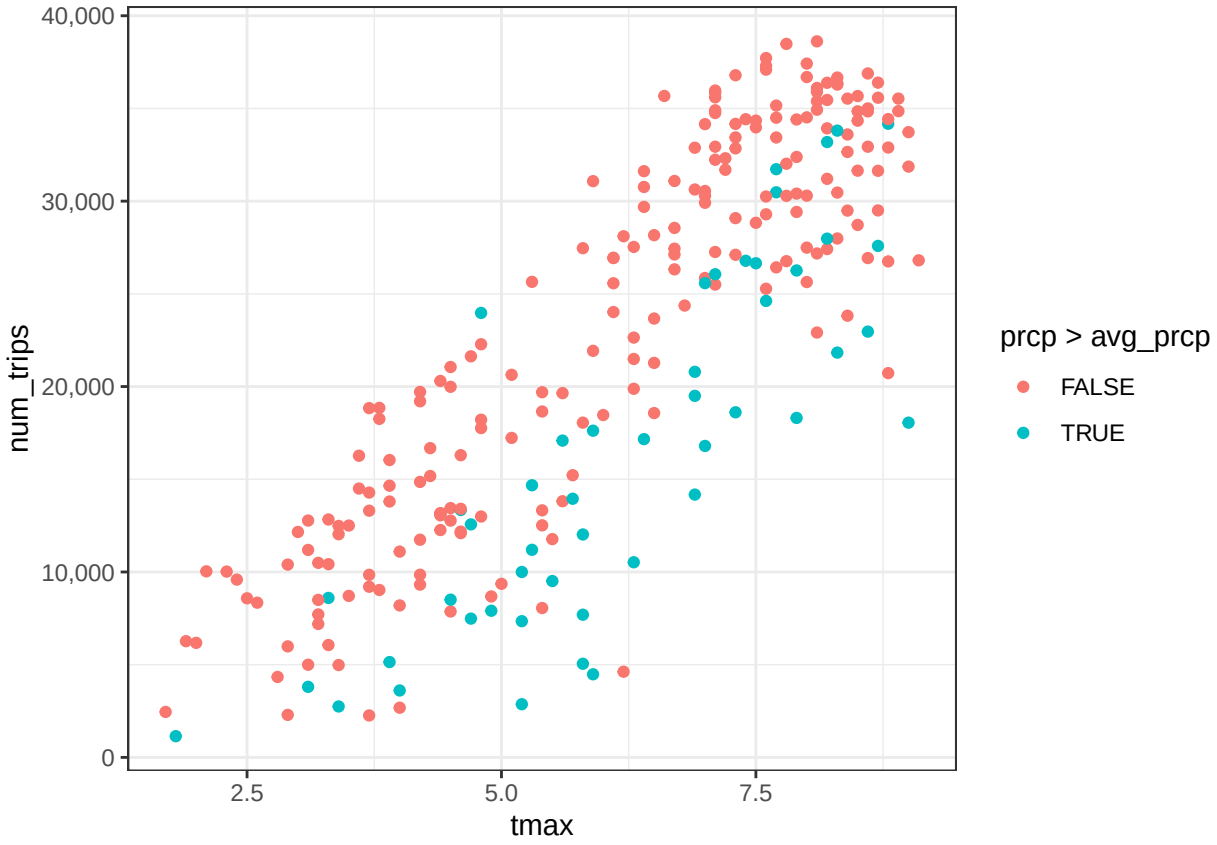
```
# Compute the benchmarks
avg_tmin <- mean(trips_per_day_model$tmin)
hlfsdup_tmax <- mean(trips_per_day_model$tmax)+(sd(trips_per_day_model$tmax)/2)
avg_prcp <- mean(trips_per_day_model$prcp)
hlfsdup_prcp <- mean(trips_per_day_model$prcp)+(sd(trips_per_day_model$prcp)/2)

# Plot prcp against trip count, seperating with color by high and low tmin, and plot lines to signify t
ggplot(trips_per_day_model, aes(x = prcp, y = num_trips, color = tmin < avg_tmin)) +
  geom_point() +
  geom_vline(xintercept = avg_prcp) +
  geom_vline(xintercept = hlfsdup_prcp, linetype = 'dashed') +
  scale_y_continuous(label = comma)
```



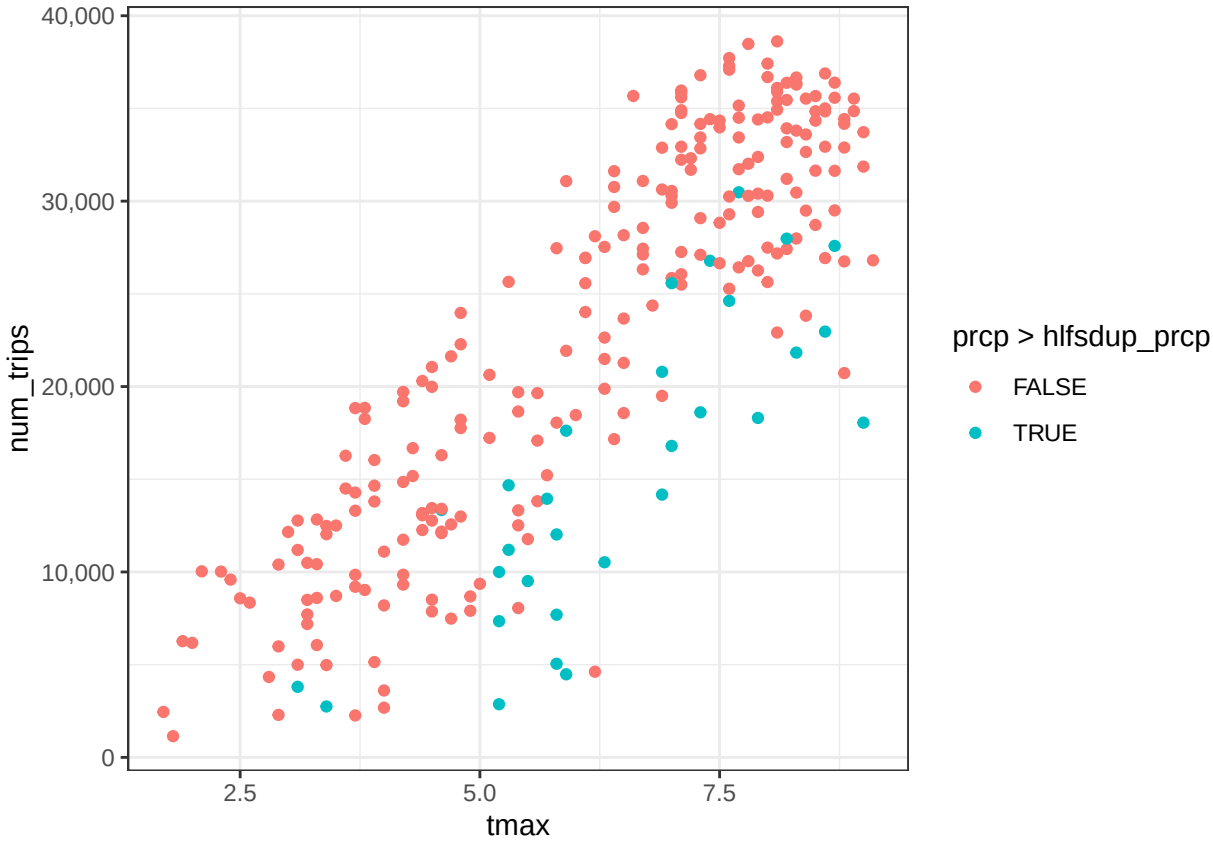

```
# Plot prcp against trip count, seperating with color by high and tmax, and plot lines to signify the b
ggplot(trips_per_day_model, aes(x = prcp, y = num_trips, color = tmax > hlfsdup_tmax)) +
  geom_point() +
  geom_vline(xintercept = avg_prcp) +
  geom_vline(xintercept = hlfsdup_prcp, linetype = 'dashed') +
  scale_y_continuous(label = comma)
```



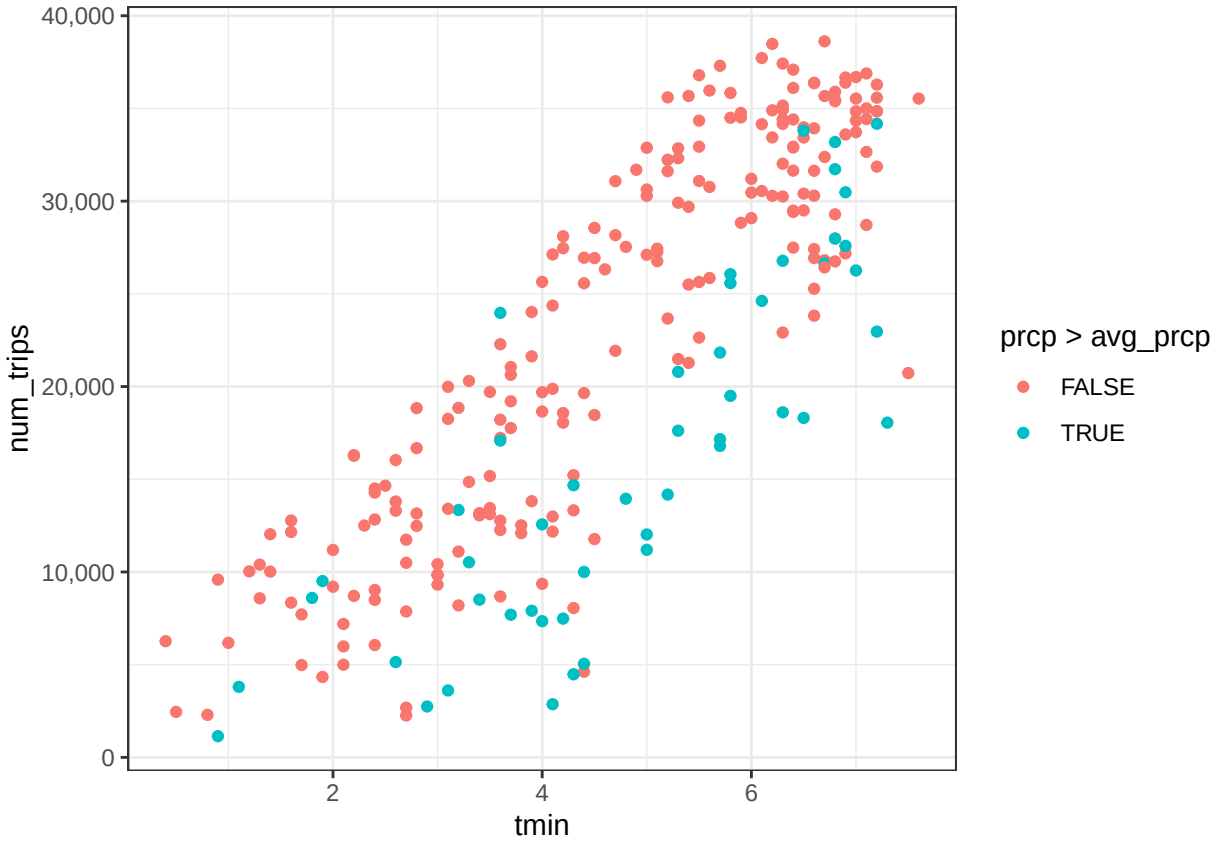

```
# Plot tmax against trip count, separating with color by high and low prcp based on the two different b
ggplot(trips_per_day_model, aes(x = tmax, y = num_trips, color = prcp > avg_prcp)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



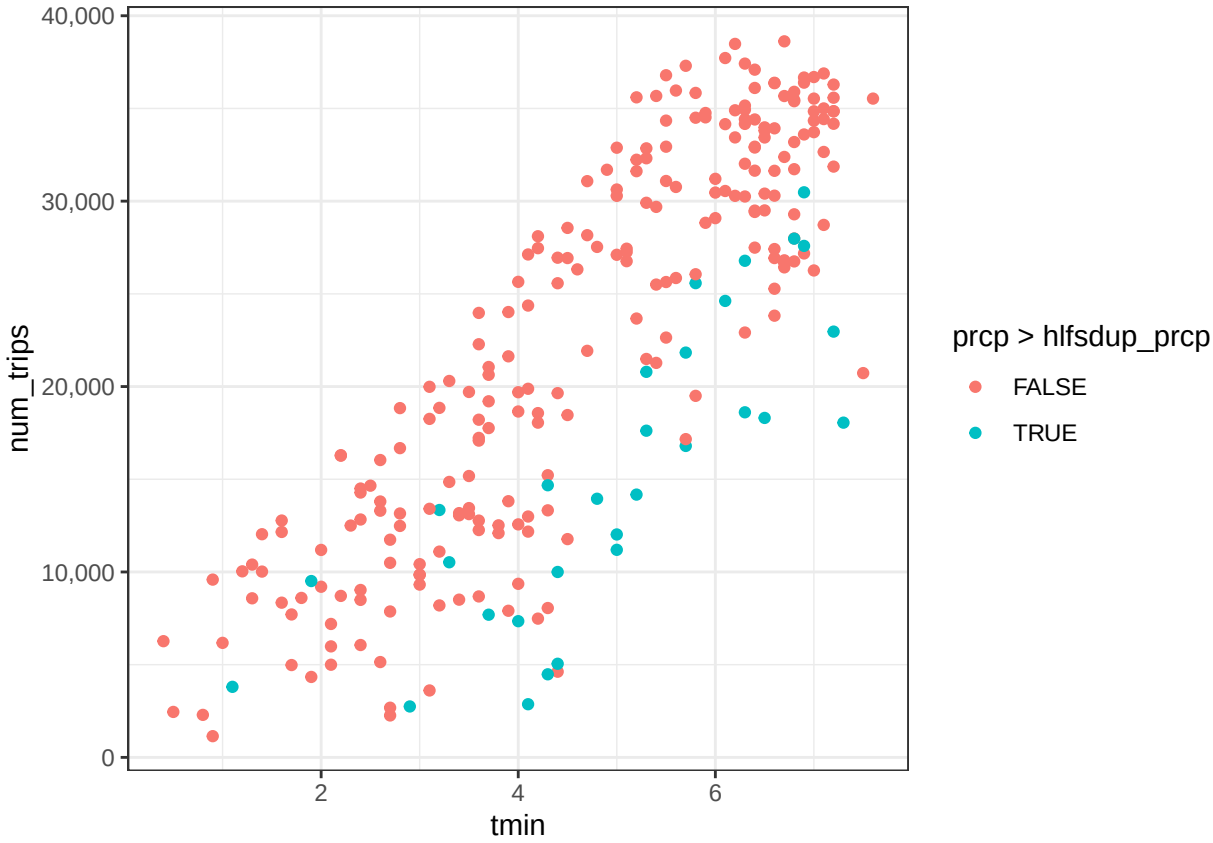

```
ggplot(trips_per_day_model, aes(x = tmax, y = num_trips, color = prcp > hlfsdup_prcp)) +
  geom_point() +
  scale_y_continuous(label = comma)
```




```
# Plot tmin against trip count, separating with color by high and low prcp based on the two different b
ggplot(trips_per_day_model, aes(x = tmin, y = num_trips, color = prcp > avg_prcp)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



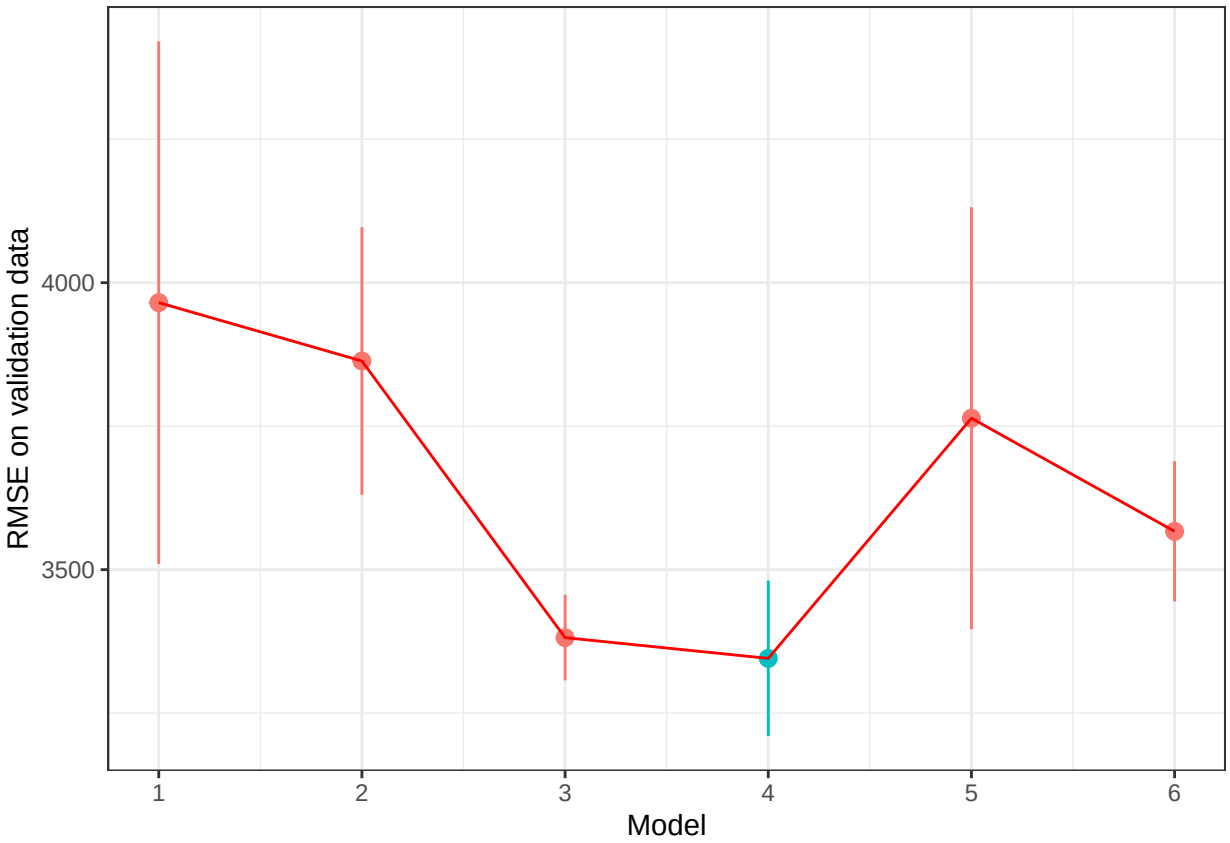

```
ggplot(trips_per_day_model, aes(x = tmin, y = num_trips, color = prcp > hlfsdup_prcp)) +
  geom_point() +
  scale_y_continuous(label = comma)
```



These seem like fair benchmarks. I extend my model in a few different ways to include these benchmarks and compare. I kep the model I chose from the last test to see if adding benchmarks is an improvement.

Test models with prcp benchmarks

```
model_list <- c("num_trips ~ prcp*tmin + prcp*tmax + snwd*tmin + snow*tmin +
                  snow*snwd + holiday + weektype + month",
                "num_trips ~ I(prcp > hlfsdup_prcp)*tmin +
                  I(prcp > hlfsdup_prcp)*tmax + snwd*tmin + snow*tmin + snow*snwd +
                  holiday + weektype + month",
                "num_trips ~ I(prcp > avg_prcp)*tmin + I(prcp > hlfsdup_prcp)*tmax +
                  snwd*tmin + snow*tmin + snow*snwd + holiday + weektype + month",
                "num_trips ~ I(prcp > 0)*tmin + I(prcp > hlfsdup_prcp)*tmax +
                  snwd*tmin + snow*tmin + snow*snwd + holiday + weektype + month",
                "num_trips ~ prcp*I(tmin < avg_tmin) + prcp*I(tmax > hlfsdup_tmax) +
                  snwd*tmin + snow*tmin + snow*snwd + holiday + weektype + month",
                "num_trips ~ I(prcp > 0)*I(tmin < avg_tmin) +
                  I(prcp > hlfsdup_prcp)*I(tmax > hlfsdup_tmax) + snwd*tmin +
                  snow*tmin + snow*snwd + holiday + weektype + month")
kfold_model_test(model_list)
```



The third and fourth models give me the lowest RMSE. While the fourth model may appear to have a slightly lower RMSE, the third model has a much smaller standard error bar, so I will proceed with this model.

Now I continue to improve my model by analyzing the features tmin, tmax, and month.

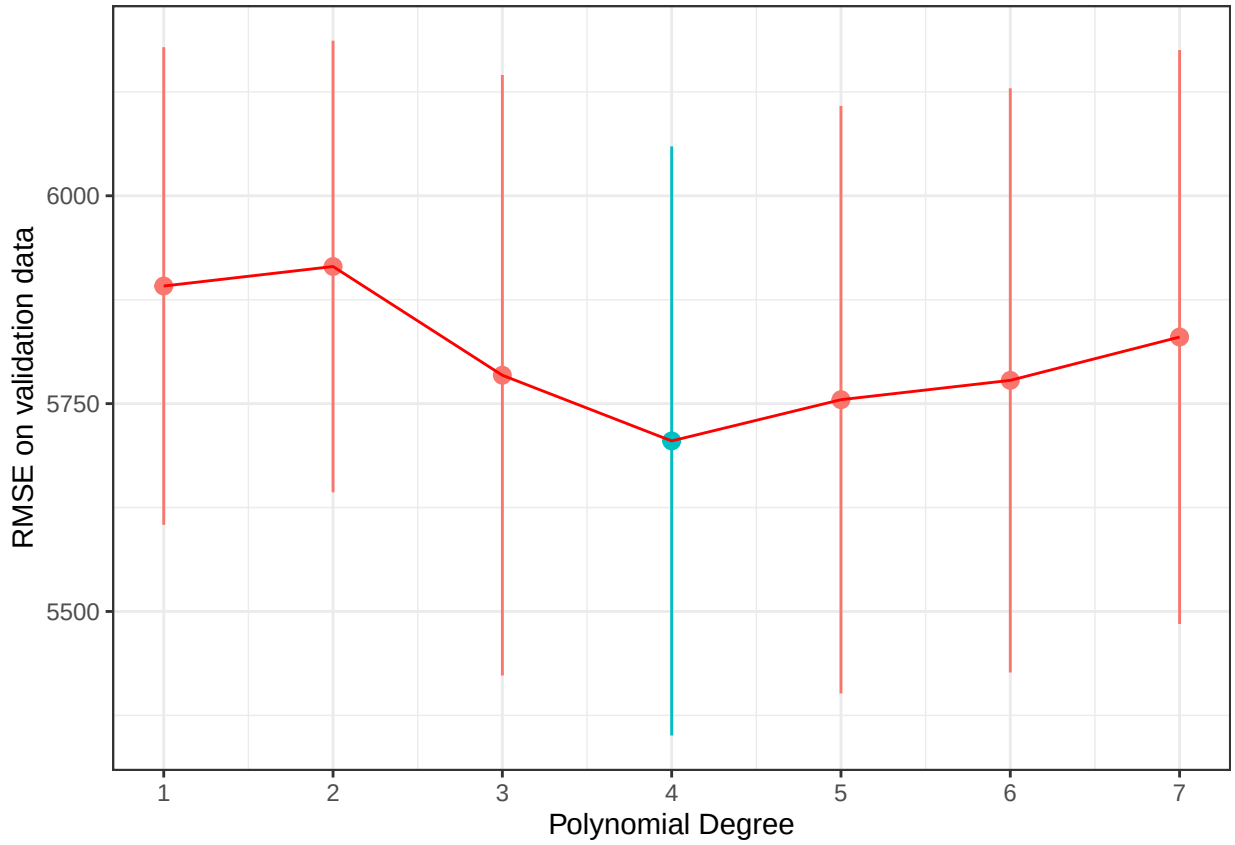
Polynomial relationship of tmin, tmax, and month

Find the best k poly value for tmin, tmax, and month.

tmin

```
# fit a model for each polynomial degree
K <- 1:7
avg_validate_err <- c()
se_validate_err <- c()
for (k in K) {

  # do 5-fold cross-validation within each value of k
  validate_err <- c()
  for (f in 1:num_folds) {
    # fit on the training data
    trips_per_day_train <- filter(trips_per_day_model, fold != f)
    model <- lm(num_trips ~ poly(tmin, k, raw=T), data=trips_per_day_train)
```

```
    # evaluate on the validation data
    trips_per_day_validate <- filter(trips_per_day_model, fold == f)
    validate_err[f] <- sqrt(mean((predict(model, trips_per_day_validate) - trips_per_day_validate$num_t
  }

  # compute the average validation error across folds
  # and the standard error on this estimate
  avg_validate_err[k] <- mean(validate_err)
  se_validate_err[k] <- sd(validate_err) / sqrt(num_folds)
}

# plot the validate error, highlighting the value of k with the lowest average error
plot_function(K, avg_validate_err, se_validate_err, 'Polynomial Degree')
```

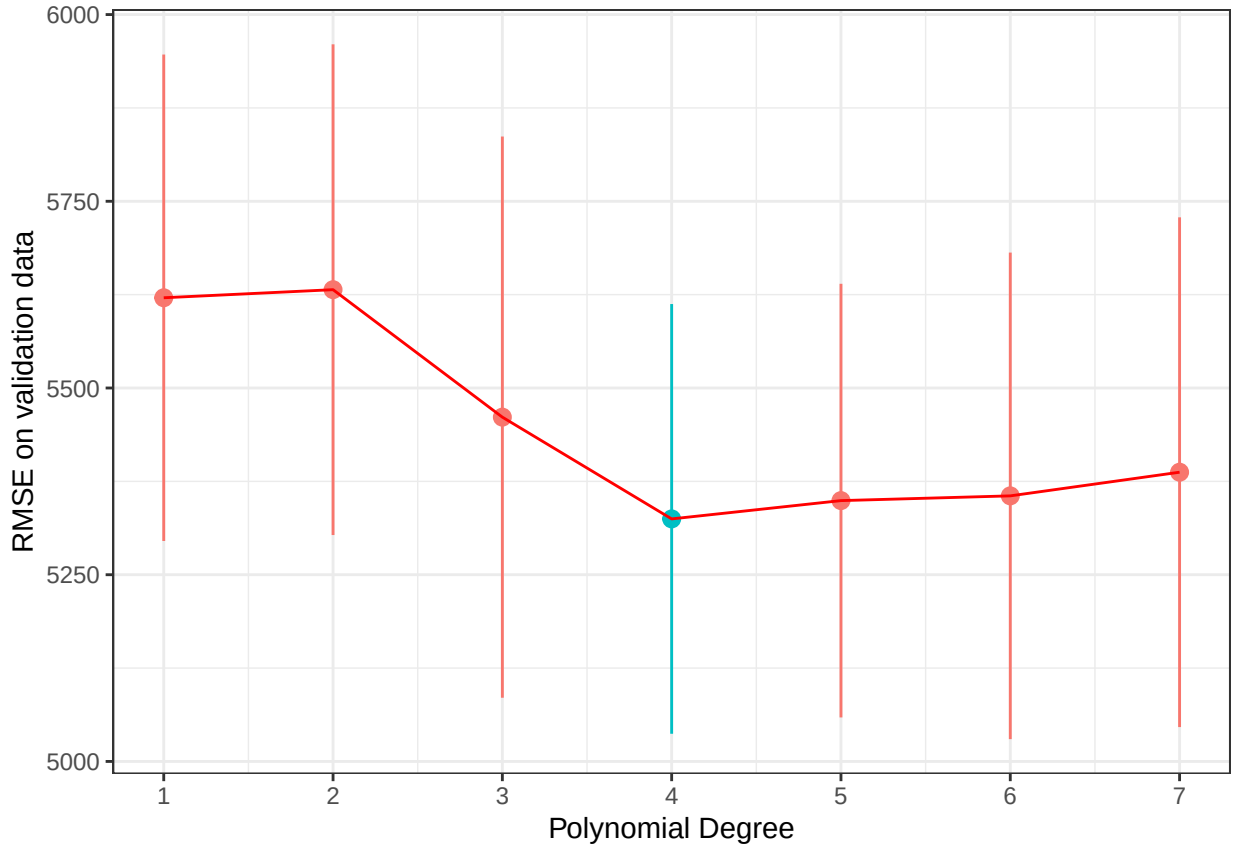


The best degree is 4. I will improve my model accordingly.

tmax

```
# fit a model for each polynomial degree
K <- 1:7
avg_validate_err <- c()
se_validate_err <- c()
for (k in K) {
```

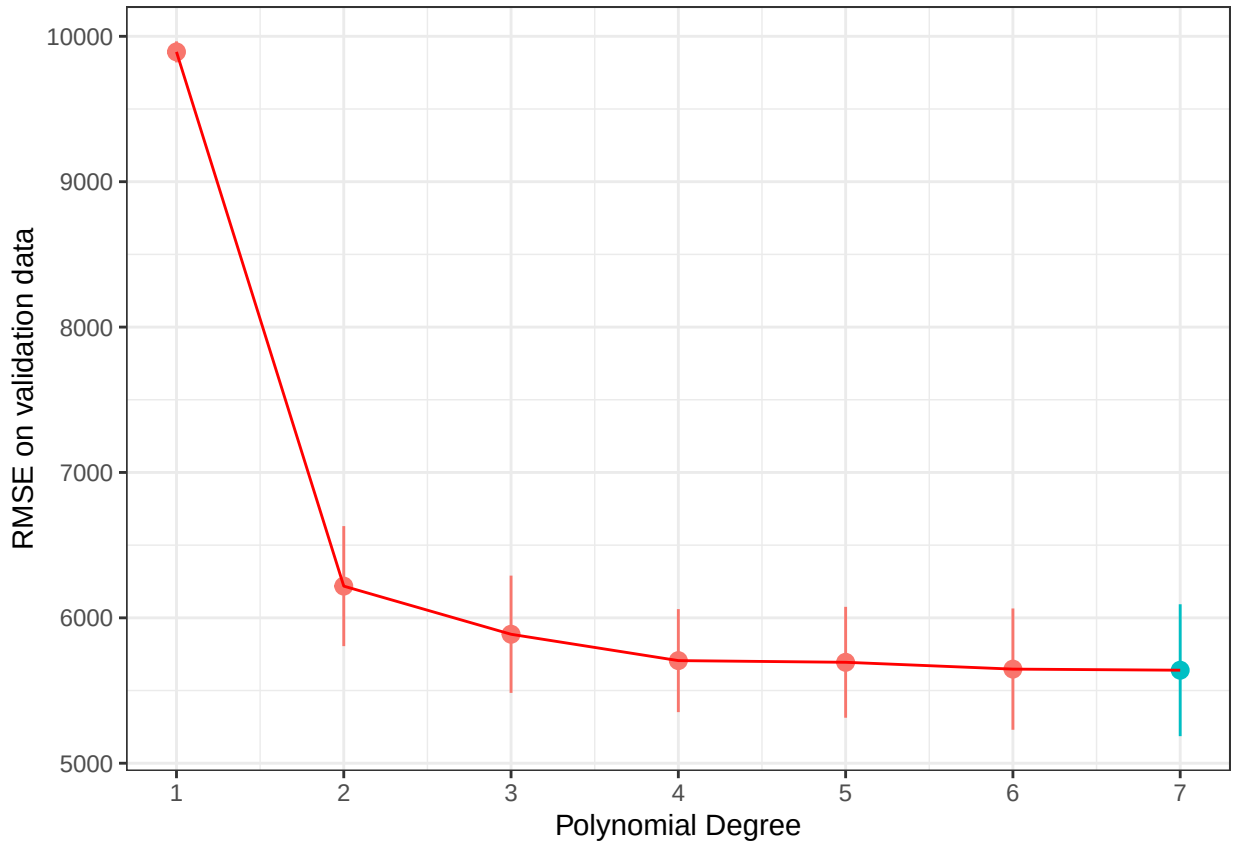
```r
  # do 5-fold cross-validation within each value of k
  validate_err <- c()
  for (f in 1:num_folds) {
    # fit on the training data
    trips_per_day_train <- filter(trips_per_day_model, fold != f)
    model <- lm(num_trips ~ poly(tmax, k, raw=T), data=trips_per_day_train)

    # evaluate on the validation data
    trips_per_day_validate <- filter(trips_per_day_model, fold == f)
    validate_err[f] <- sqrt(mean((predict(model, trips_per_day_validate) - trips_per_day_validate$num_t
  }

  # compute the average validation error across folds
  # and the standard error on this estimate
  avg_validate_err[k] <- mean(validate_err)
  se_validate_err[k] <- sd(validate_err) / sqrt(num_folds)
}

# plot the validate error, highlighting the value of k with the lowest average error
plot_function(K, avg_validate_err, se_validate_err, 'Polynomial Degree')
```



The best degree is 4. I will improve my model accordingly.

month

```
# fit a model for each polynomial degree
K <- 1:7
avg_validate_err <- c()
se_validate_err <- c()
for (k in K) {

  # do 5-fold cross-validation within each value of k
  validate_err <- c()
  for (f in 1:num_folds) {
    # fit on the training data
    trips_per_day_train <- filter(trips_per_day_model, fold != f)
    model <- lm(num_trips ~ poly(month, k, raw=T), data=trips_per_day_train)

    # evaluate on the validation data
    trips_per_day_validate <- filter(trips_per_day_model, fold == f)
    validate_err[f] <- sqrt(mean((predict(model, trips_per_day_validate) - trips_per_day_validate$num_t
  }

  # compute the average validation error across folds
  # and the standard error on this estimate
  avg_validate_err[k] <- mean(validate_err)
  se_validate_err[k] <- sd(validate_err) / sqrt(num_folds)
}

# plot the validate error, highlighting the value of k with the lowest average error
plot_function(K, avg_validate_err, se_validate_err, 'Polynomial Degree')
```



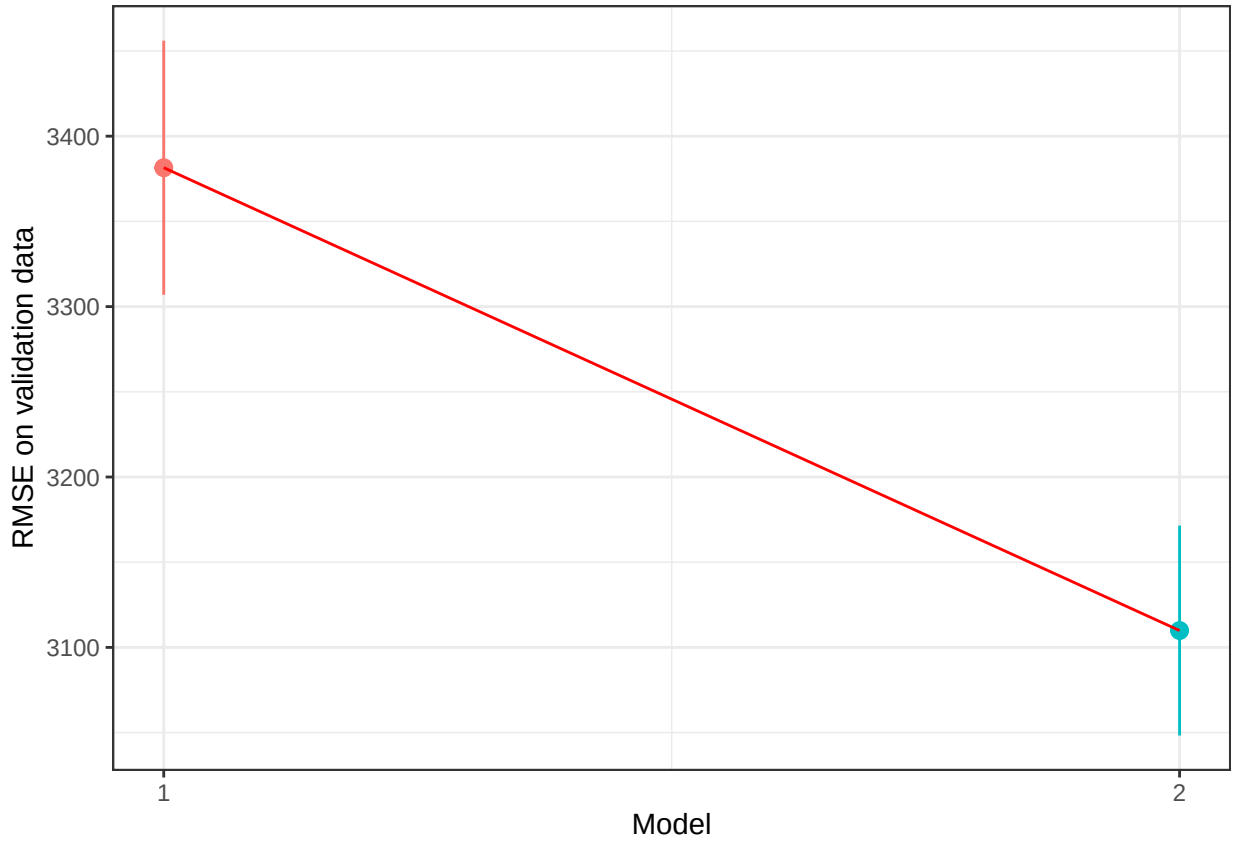
The best degree seems to be 7, but I can just as easily choose 4 and be in pretty good shape. I don't want to risk overfitting my model, so I will stick with 4 and improve my model accordingly.

Test the model with added polynomial relationships

```
model_list <- c("num_trips ~ I(prcp > avg_prcp)*tmin + I(prcp > hlfsdup_prcp)*tmax +
                  snwd*tmin + snow*tmin + snow*snwd + holiday + weektype + month",
                "num_trips ~ I(prcp > avg_prcp)*tmin + I(prcp > hlfsdup_prcp)*tmax +
                  snwd*tmin + snow*tmin + snow*snwd + holiday + weektype +
                  poly(month, 4, raw=T) + poly(tmax, 4, raw=T) + poly(tmin, 4, raw=T)")
kfold_model_test(model_list)
```

```
## Warning in predict.lm(model, trips_per_day_validate): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(model, trips_per_day_validate): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(model, trips_per_day_validate): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases

## Warning in predict.lm(model, trips_per_day_validate): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```

```
## Warning in predict.lm(model, trips_per_day_validate): prediction from
## rank-deficient fit; attr(*, "non-estim") has doubtful cases
```



My exploration is complete! Below is my final model, and it is looking great! My RMSE is now just above 2800 and I have super high R-squared and adjusted R-squared values of around 92%!

```
model <- lm(num_trips ~ I(prcp > 0.15)*tmin + I(prcp > 0.36)*tmax +
              snwd*tmin + snow*tmin + snow*snwd + holiday + weektype +
              poly(month, 4, raw=T) + poly(tmax, 4, raw=T) + poly(tmin, 4, raw=T),
            data=trips_per_day)

rmse(model, trips_per_day)  # 2835.046
```

```
## Warning in predict.lm(model, data): prediction from rank-deficient fit; attr(*,
## "non-estim") has doubtful cases
```

```
## [1] 2835.046
```

```
summary(model)
```

```
##
## Call:
## lm(formula = num_trips ~ I(prcp > 0.15) * tmin + I(prcp > 0.36) *
##     tmax + snwd * tmin + snow * tmin + snow * snwd + holiday +
```

```
##     weektype + poly(month, 4, raw = T) + poly(tmax, 4, raw = T) +
##     poly(tmin, 4, raw = T), data = trips_per_day)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9798.3 -1454.3   350.2  1740.9 13257.2
##
## Coefficients: (2 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -2.007e+04  1.204e+04  -1.667 0.096338 .
## I(prcp > 0.15)TRUE         -4.281e+03  1.853e+03  -2.310 0.021476 *
## tmin                       -4.683e+03  5.703e+03  -0.821 0.412167
## I(prcp > 0.36)TRUE         -8.064e+03  2.868e+03  -2.812 0.005212 **
## tmax                        2.950e+04  1.237e+04   2.385 0.017632 *
## snwd                       -3.045e+02  2.078e+02  -1.466 0.143648
## snow                        7.624e+02  5.282e+02   1.443 0.149835
## holidayTRUE                -1.650e+04  1.509e+03 -10.938  < 2e-16 ***
## weektypeWeekend            -5.770e+03  3.507e+02 -16.454  < 2e-16 ***
## poly(month, 4, raw = T)1 -6.348e+02  1.944e+03  -0.327 0.744244
## poly(month, 4, raw = T)2  4.511e+02  6.285e+02   0.718 0.473453
## poly(month, 4, raw = T)3 -3.380e+01  7.096e+01  -0.476 0.634117
## poly(month, 4, raw = T)4 -6.458e-04  2.598e+00   0.000 0.999802
## poly(tmax, 4, raw = T)1           NA         NA      NA       NA
## poly(tmax, 4, raw = T)2  -9.798e+03  3.573e+03  -2.743 0.006419 **
## poly(tmax, 4, raw = T)3   1.400e+03  4.367e+02   3.206 0.001475 **
## poly(tmax, 4, raw = T)4  -6.824e+01  1.912e+01  -3.569 0.000409 ***
## poly(tmin, 4, raw = T)1           NA         NA      NA       NA
## poly(tmin, 4, raw = T)2   3.446e+03  2.339e+03   1.473 0.141545
## poly(tmin, 4, raw = T)3  -6.651e+02  3.944e+02  -1.686 0.092651 .
## poly(tmin, 4, raw = T)4   3.940e+01  2.322e+01   1.697 0.090672 .
## I(prcp > 0.15)TRUE:tmin  -9.345e+01  3.901e+02  -0.240 0.810829
## I(prcp > 0.36)TRUE:tmax   6.022e+02  4.528e+02   1.330 0.184435
## tmin:snwd                -2.182e+01  7.182e+01  -0.304 0.761397
## tmin:snow                -3.871e+02  2.806e+02  -1.379 0.168692
## snwd:snow                 2.719e+01  3.776e+01   0.720 0.471927
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2933 on 341 degrees of freedom
## Multiple R-squared:  0.9234, Adjusted R-squared:  0.9182
## F-statistic: 178.7 on 23 and 341 DF,  p-value: < 2.2e-16
```

Part E: Plot the Model

Now I plot the final best fit model in two different ways. First with the date on the x-axis and the number of trips on the y-axis, showing the actual values as points and predicted values as a line. Second as a plot where the x-axis is the predicted value and the y-axis is the actual value, with each point representing one day.

Plot the data

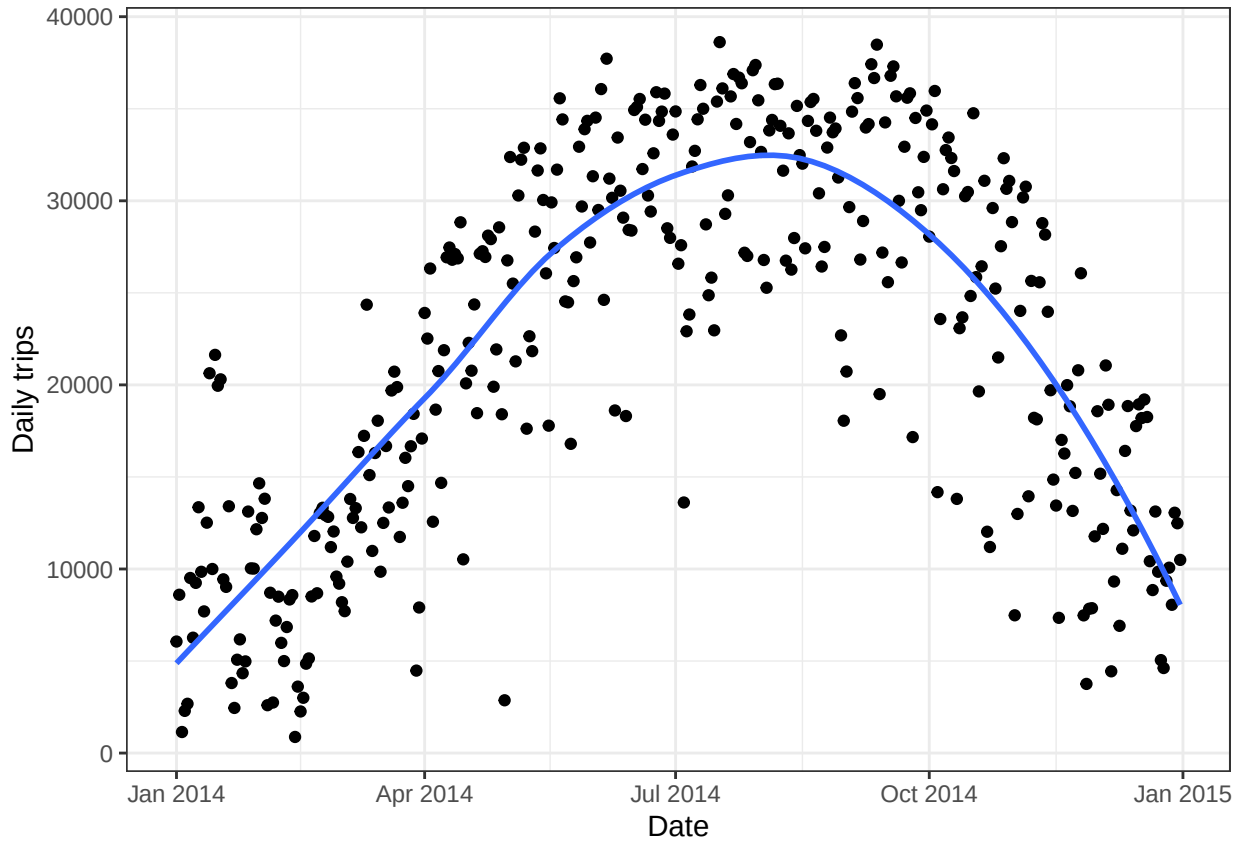
Plot the actual data as points and predicted values as a line.

```
plot_2014 <- trips_per_day %>%
  add_predictions(model) %>%
  ggplot(aes(x = ymd, y = num_trips)) +
  geom_point() +
  geom_smooth(aes(y = pred), se=F) +
  labs(x = 'Date', y = 'Daily trips') +
  scale_y_continuous()
```

```
## Warning in predict.lm(model, data): prediction from rank-deficient fit; attr(*,
## "non-estim") has doubtful cases
```

```
plot_2014
```

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




```
save(model, plot_2014, file = 'predict_citibike.Rdata')
```

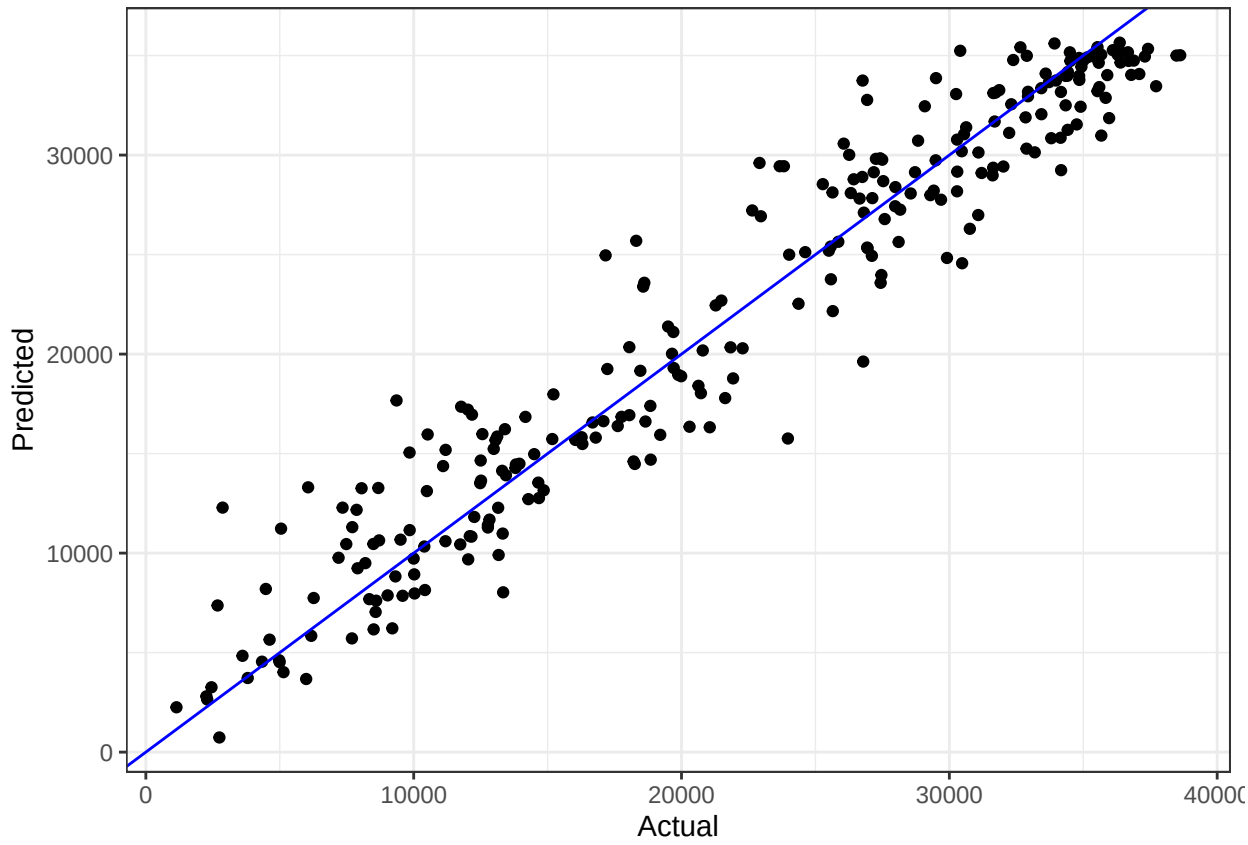
Plot the correlation

Plot the correlation between the predicted values and the actual values.

```
trips_model <- trips_per_day_model %>%
  add_predictions(model)
```

```
## Warning in predict.lm(model, data): prediction from rank-deficient fit; attr(*,
## "non-estim") has doubtful cases
```

```
trips_model %>%
  ggplot(aes(x = num_trips, y = pred)) +
  geom_point() +
  geom_abline(color = 'blue') +
  labs(x = 'Actual', y = 'Predicted') +
  scale_y_continuous()
```



The trend is super linear, which bodes well for my model.