Stat 27850/30850: Group project # 1

Data The data set for the first project is the Capital Bikeshare data set, from https://www.capitalbikeshare.com/system-data. This data comes from Washington D.C.'s bikeshare program, which records every individual ride taken. Each ride is tagged with its start and end time, start and end station (there are hundreds of locations where bikes can be rented across the city), as well as whether the rider has bought a one-time rental or is a member of the bikeshare program.

On Canvas, we provide a script named `getdata_bikeshare.R` to download and clean the data so that it's ready for R, and organized in a simple format. (A Python script is also available, `getdata_bikeshare.ipynb`.) The variables you will have after running the script are:

- Data for all $n = 1342364$ rides in 2010 & 2011.
- `starttime`, a $n \times 6$ matrix with the start time of each ride. The columns are: year/month/day/hour/minute/second.
- `duration`, a length-n vector indicating the duration of the bike ride, in seconds.
- `station_start` & `station_end`, length-n vector giving the station number where the ride began / ended.
- `stations`, a matrix matching station numbers to physical locations (you can choose to use this information if you would like to incorporate geographic information / calculate distance between stations / restrict to a particular geographic region / etc). Note that stations get added to the program over time (i.e., not all stations were already in use at the beginning of 2010).
- `bike_num`, a length-n vector giving the ID number of the bike used on each ride. Some are NA (unknown).
- `member`, a True/False length-n vector indicating whether each ride was taken by a paid member of the bikeshare program or a one-time purchase.
- `days_since_Jan1_2010`, a length-n vector measuring the date (i.e., the date at the start of the ride) in terms of days since Jan 1 2010.
- `day_of_week`, a length-n vector indicating the day of the week, Monday/Tuesday/etc.

The script will then save these variables in a file.

Assignment Your task is to study the following question: can you detect any routes (i.e., a particular combination of start & end station) where the average time it takes to travel the route, changes over the course of the time period? For example, if a bike lane is added to a major road, this may reduce the travel time for that particular route.

One simple approach we could take, would be to find routes that seem to show an increase (or decrease) in duration over time, then permute the vector of durations for that route and see if the apparent increase (or decrease) is likely to appear equally strong even when the data is randomly shuffled.

However, there are many possible confounders that are not accounted for by a simple permutation scheme. For example, members may ride faster than one time users; the day of the week can affect traffic and therefore change the speed of the ride; difficult weather can slow things down; etc. This is only a suggested list—you're welcome to frame the problem differently, and your group might think of other important confounders as well—be sure to explain your decisions in your report. (For weather—you are welcome to look up weather records from those dates, but this is completely optional and may not be feasible given the limited time—we recommend that you treat weather as unknown but constant over any single day / any single hour / etc.)

Your task is to find a way to reliably identify which routes (if any) show changes. There will be many possible ways to address this. You can think about how to use task-specific permutation strategies (rather than permuting at random), or you can use regression models or other tools instead of permutations. Your final report should give a thoughtful discussion of the issues you have identified, and the strategies and methods you developed to address them. There might be confounding effects you identify that it's not possible to address with the limited data available (or simply due to time constraints); your report can also mention issues that you were not able to address, and assess to what

extent you think it might affect the validity of the analysis. Depending on your approach, it may happen that you are or are not able to detect routes with significant changes. Either outcome is fine, as long as your conclusions and methods are well explained and justified.

Guidelines Groups of size 2, 3, or 4 are allowed for the mini-project. The extent of the project (e.g., the range of questions explored / methods tried / etc) should be proportional to the size of the group. What you hand in:

- Each group should hand in a written report and either include code throughout the report and/or include the code as an appendix. Please designate a single group member to submit everything on Gradescope, and add the other students in the team group members.
- For your code, it should be clearly organized and commented— for example, you may want to label sections of the code so that we can see which part of the report or which plot/table it corresponds to, add comments to explain steps where notation / variable names / nature of the calculation aren't obvious, etc.
- There are no page length or formatting requirements for the written report. Your report should describe the problems and questions you posed, the details of any methods you implemented / models fitted / hypotheses tested, describe your findings and show plots or numerical results as appropriate, and should discuss some interesting issues relating to inference (for example, multiple testing / appropriately controlling for confounding factors / reducing a high dimensional model to a manageable size / etc). You can also include a discussion of open questions and issues that were not addressed (due to time limitations and/or limitations of the available data).

STAT 27850 Project One Report

Tony (Haotian) Zong, Louisa Lyu, Stanley Zhu

2023-02-09

There are three chapters in this report. Chapter one is about a prediction-based approach using linear regression. Chapter two is about the conditional permutation test. Chapter three is the discussion of the performance of different approaches attempted, the significance of our results, and the limitations.

Chapter One: "Regression Test"– A Prediction-Based Approach

Section I: Introduction to the Regression Approach

The gist of this approach is using one year's data to predict the other year's ride durations (the outcome variable), and then comparing the predictions to the actual durations. The intuition is that if there is no significant change in the routes, we should expect to see that the predictions and the true ride durations to be close to each other. And if we observe a statistically significant amount of differences between them for a certain route, this could be a potential evidence that this route had changed between 2010 and 2011.

Before delving into the details of each step of this approach, we want to acknowledge some limitations of this approach in answering the original question of "detecting if there is any change at any point for each route". This approach, as an exploratory analysis, simplifies the question asked. Instead, here we are investigating if there is a change between 2010 and 2011 for each route. Furthermore, due to the limitation of the data (and the fact that this bike share program in D.C. only started in late 2010), we are only comparing Sept.-Dec. 2010 vs. Sept.-Dec. 2011, as there is no data in 2010 from Jan. to Aug.

Again, since the program started in late 2010, there was less data in 2010 than in 2011. Thus, in order to consider more routes in our regression model (a route needs to have enough data to be included in the model; otherwise there can be too much noise), we decided to use 2011 Sept.-Dec.'s data to train the model, and then apply the model on 2010's data for testing.

1.1 Outline of the steps taken

Step 0: Data cleaning. Details will be discussed in the later section.

Step 1: Filter for 2011 Sept.-Dec.'s data. Create a hold-out set for 2011 (for example, 20%). This hold-out set (validation set) will be used to compare with the prediction performance of the model on the test set later.

Step 2: Train a regression model using 2011's training data. Here, we tried two approaches when training the model.

One approach is to train one single model for all the routes, with `route` as a covariate in the model. This will assume that covariates, such as day of week, daily temperature, membership status etc., have the same effects on ride duration across all the routes. While we believe this might be generally true, there definitely can be instances where, for example, some covariate like a low daily temperature (cold weather) has a greater effect on ride duration because of the terrain or geography of some route. i.e. there can be interactions between `route` and other covariates. But we are not including them in the model due to the limitation of sample size.

Another approach is to train a different model for each route we consider. While this may better account for the interaction between `route` and other covariates, since we generally only have hundreds of data points for each route, the relatively low sample size may make the models less robust.

In the later sections, we implemented both approaches.

Step 3: Aggregate the hold-out data set by categorical covariates. Since the data is assumed to be noisy, we decided to aggregate each route's data on an hourly basis (and control for membership status) by taking median for the ride duration. We record how many original samples each aggregated sample represents, and this will be used as weights when making predictions.

Note: We didn't aggregate the data when building the model to allow for a wider range for the feature values, so that it's less likely for extrapolation to happend when making predictions.

Step 4: Apply the model on the 2011 hold-out set. Construct 95% prediction intervals for the hold-out data set, adjusted for the weights mentioned in step 3. Record the non-coverage rate of the prediction intervals for each route.

Step 5: Apply the regression model on the aggregated 2010 data set. Construct prediction intervals and find a new non-coverage rate for each route. Perform a binomial test on the the non-coverage rates we get when applying the model on the 2011 validation set and the 2010 test set.

Step 6: Adjust the p-values from the tests for the multiple testing issue (using FDR control), and then report the routes with significant change in non-coverage rate, i.e. the routes that have statistically significant change between 2010 and 2011.

1.2 Initial data cleaning

We get additional data on daily average temperature in Washington D.C. from North America Land Data Assimilation System. We take a log transformation on ride duration to make it less skewed. Then, we remove outliers in `log_duration` using the 1.5 IQR rule and remove routes with less than 500 rides in the original data set to make out test to be less affected by the outliers. In the end, we exclude the data from Jan. to Aug. in 2011, as discussed earlier in the outline.

```
# Temperature Data
temp_data <-
  read.table("data/temperature-data.txt",
             skip = 1,
             col.names = c('time1', 'time2', 'daily_max', 'daily_min')) %>%
  mutate(year = as.numeric(substring(time2, 1, 4))) %>%
  mutate(month = as.numeric(substring(time2, 6, 7))) %>%
  mutate(day = as.numeric(substring(time2, 9, 10))) %>%
  mutate(daily_temp = (daily_max + daily_min) / 2)

# Remove outliers with IQR range test
outliers <- function(x) {
    q25 <- quantile(x, probs=.25)
    q75 <- quantile(x, probs=.75)
    interval <- q75 - q25
    x > q75 + (interval * 1.5) | x < q25 - (interval * 1.5)
}

# Bike-share Data
load("data/bikedata.RData")
colnames(starttime) = c("year", "month", "day", "hour", "minute", "second")
df <- data.frame(log_duration = log(duration), station_start, station_end,
                 starttime, day_of_week, days_since_Jan1_2010, member) %>%
  # Join with Temperature Data
  left_join(temp_data, by = c('year','month','day')) %>%
  # Add Weekend/weekday
  mutate(weekend = day_of_week %in% c("Saturday", "Sunday")) %>%
  # Add Hour of the Day
  mutate(hour_of_day = cut(hour, c(-1, 6, 12, 18, 24))) %>%
  # Add temperature buckets
  mutate(temp = cut(daily_temp, c(20, 40, 60, 80, 100))) %>%
  # Assign Id to Route
  group_by(station_start, station_end) %>%
```

```
  mutate(route = paste0(station_start, "-", station_end)) %>%
  mutate(route = factor(route)) %>%
  # Filter out routes with less than 500 records
  filter(length(log_duration) >= 500 & !outliers(log_duration)) %>%
  ungroup()
# drop 2011 Jan-Aug data
df <- df %>% filter(month %in% 9:12) %>%
  mutate(month = factor(month)) %>%
  mutate(day_of_week = factor(day_of_week)) %>%
  mutate(day_of_week = fct_relevel(day_of_week, c("Monday","Tuesday","Wednesday",
                                                  "Thursday","Friday","Saturday","Sunday"))) %>%
  mutate(hour = factor(hour))
```

Section II: One Single Model for All Routes, with `route` as a Covariate

In this section, we implemented the test using one single regression model for all routes, with `route` as a covariate.

We want to make sure for each route we train the model on, there is enough data. Since we use 2011 data to build the regression model, we filter for "busy" routes based on 2011's data. Here, we decide to only consider the routes with more than 100 rides during September - December in 2011.

We also filter for routes that have more than 100 routes in 2010. A route with too few rides in the test set can result in too much variation in the non-coverage rate, so we decide to only apply the regression model for routes with enough rides when making predictions.

```
busy_routes_df <- df %>%
  filter(year==2011) %>%
  group_by(route) %>%
  summarise(ride_counts = n()) %>%
  filter(ride_counts >= 100)
busy_routes <- busy_routes_df$route

busy_routes_df_2010 <- df %>%
  filter(year==2010) %>%
  group_by(route) %>%
  summarise(ride_counts = n()) %>%
  filter(ride_counts >= 100)
busy_routes_2010 <- busy_routes_df_2010$route
busy_routes_2010 <- intersect(busy_routes, busy_routes_2010)

# split data by year
df11 <- df %>% filter(year == 2011 & route %in% busy_routes) %>%
  select(c(log_duration, member, year, month, day, hour,
           day_of_week, daily_temp, weekend, route))

df10 <- df %>% filter(year == 2010 & route %in% busy_routes_2010)
```

2.1 Implementation of the Test

Step 1: create a hold-out set for 2011

Here, we use 80% of 2011's data to train the model. The rest of 20% will be validation set.

```
# create hold-out set for 2011
picked = sample(seq_len(nrow(df11)), size = nrow(df11)*0.8)
df11_train =df11[picked,]
df11_holdout =df11[-picked,]
```

Step 2: train a regression model using 2011's training data

Regression formula:

`log_duration` ~ `member` + `month` + `day_of_week` + `factor(hour)` + `daily_temp` + `route`

We regress `log_duration` on membership status, month, day of week, hour, and daily temperature. Additionally, `route` is also included as a covariate in the model.

```
lm_2011 = lm(log_duration ~ member + daily_temp + month + day_of_week + hour + route,
             data = df11_train)
```

Step 3: aggregation on 2011 hold-out set

We perform aggregation on the 2011 hold-out data set by grouping rides for each route from the same hour and membership status together. We do the aggregation by taking the median on `log_duration`. Aggregation supposedly should make the data less noisy. We also keep track of the number of samples represented by each row in this aggregated data frame. This will be used as weights when we construct the prediction intervals in the next step.

```
df11_holdout_agg <- df11_holdout %>%
  select(-c(year, weekend)) %>%
  group_by(route, month, day, hour, member) %>%
  mutate(log_duration = median(log_duration)) %>%
  mutate(counts = n()) %>%
  ungroup() %>%
  distinct() %>%
  arrange(route, month, day, hour, member)
```

Step 4: construct prediction intervals for the hold-out data set

Here, we just want to make sure that there is no route that only appears in the hold-out set, but not in the training set, as this will make the `predict()` function fail. After checking this, we know it didn't happen.

```
train_routes <- unique(df11_train$route)
df11_holdout_agg <- df11_holdout_agg %>%
  filter(route %in% train_routes)
```

Then, we make predictions for the 2011 hold-out set, and construct 95% prediction intervals adjusted by weights.

Without aggregation, the formula for the prediction interval of a new sample is:

$$\hat{y}_h \pm t_{(1-\alpha/2,n-2)} \times \sqrt{MSE \times \left(1 + \frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum (x_i - \bar{x})^2}\right)}$$

Now, for an aggregated sample i with weight w_i, where w_i represents the number of original samples being aggregated into this aggregated sample, the formula for the prediction interval of a new aggregated sample is:

$$\hat{y}_h \pm t_{(1-\alpha/2,n-2)} \times \sqrt{MSE \times \left(\frac{1}{\sqrt{w_i}} + \frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right)}$$

```
prediction_2011_holdout <- data.frame(predict(lm_2011, newdata = df11_holdout_agg,
                                    interval = "predict", level = 0.95,
                                    weights = df11_holdout_agg$counts)) %>%
  mutate(true_duration = df11_holdout_agg$log_duration)
```

The 95% prediction intervals for the hold-out set as a whole have a non-coverage rate of 0.04799874, which is roughly equal to what we expect (0.05):

```
overall_nc_rate_2011 <- nrow(prediction_2011_holdout %>%
       filter(true_duration>upr | true_duration<lwr)) / nrow(prediction_2011_holdout)
overall_nc_rate_2011
```

```
## [1] 0.04799874
```

Then, we compute the non-coverage rates each route in the 2011 hold-out set. The non-coverage rate for route i is defined as:

$$\frac{number\ of\ rides\ of\ route\ i\ whose\ true\ duration\ lies\ outside\ of\ the\ prediction\ interval}{total\ number\ of\ rides\ of\ route\ i}$$

```
noncoverage_rates_2011 <- prediction_2011_holdout %>%
  mutate(route = df11_holdout_agg$route) %>%
  group_by(route) %>%
  mutate(total_ride_count = n()) %>%
  mutate(noncoverage_count = sum(true_duration>upr | true_duration<lwr)) %>%
  mutate(noncoverage_proportion = noncoverage_count / total_ride_count) %>%
  ungroup() %>%
  select(c(route, total_ride_count, noncoverage_count, noncoverage_proportion)) %>%
  distinct() %>%
  arrange(desc(noncoverage_proportion)) %>%
  filter(route %in% busy_routes_2010)
```

Now, if the 2010's data doesn't have significant change from 2011's data, we should also expect to see a non-coverage rate of roughly 5%.

Step 5: apply the regression model on the aggregated 2010 data set, construct prediction intervals, find non-coverage rate for each route, and then perform binomial tests to see if the non-coverage rate on the testing set is significantly greater than that on the hold-out set

Again, we first aggregate 2010's data by hour and by membership status and compute weights, just like what we did to 2011's hold-out set:

```
df10_agg <- df10 %>%
  select(-c(year, weekend)) %>%
  group_by(route, month, day, hour, member) %>%
  mutate(log_duration = median(log_duration)) %>%
  mutate(counts = n()) %>%
  ungroup() %>%
```

```
  distinct() %>%
  filter(route %in% train_routes) %>%
  arrange(route, month, day, hour, member)
```

Then, we create prediction intervals, adjusted by weights:

```
prediction_2010 <- data.frame(predict(lm_2011, newdata = df10_agg,
                                      interval = "predict", level = 0.95,
                                      weights = df10_agg$counts)) %>%
  mutate(true_duration = df10_agg$log_duration) %>%
  mutate(route = df10_agg$route)
```

Finally, we get an overall non-coverage rate of 0.1022566, which is greater than 0.05.

```
nrow(prediction_2010 %>%
       filter(true_duration>upr | true_duration<lwr)) / nrow(prediction_2010)
```

```
## [1] 0.1022566
```

We can do a binomial test to see if the overall non-coverage rate of the 2010 test set is significantly greater than the overall non-coverage rate of the 2011 hold-out set:

```
binom.test(nrow(prediction_2010 %>% filter(true_duration>upr | true_duration<lwr)),
           nrow(prediction_2010),
           overall_nc_rate_2011)$p.value
```

```
## [1] 7.376648e-274
```

This small p-value indicates that there was some route that had changed between 2010 and 2011. Next, we test for each route whether it had significant change or not.

We first compute the non-coverage rates for the routes in the 2010 testing set:

```
noncoverage_rates_2010 <- prediction_2010 %>%
  group_by(route) %>%
  mutate(total_ride_count = n()) %>%
  mutate(noncoverage_count = sum(true_duration>upr | true_duration<lwr)) %>%
  mutate(noncoverage_proportion = noncoverage_count / total_ride_count) %>%
  ungroup() %>%
  select(c(route, total_ride_count, noncoverage_count, noncoverage_proportion)) %>%
  distinct() %>%
  arrange(desc(noncoverage_proportion))
```

The for each `busy_route`, i.e. each route that had more than 100 rides in both 2010 Sept-Dec and 2011 Sept-Dec, we do a binomial test to see if its non-coverage rates on 2011 hold-out set and 2010 testing set are significantly different than each other:

```
p_vals = c()

for (a_route in busy_routes_2010) {

  route_info_2011 = noncoverage_rates_2011 %>% filter(`route` == a_route)
  route_info_2010 = noncoverage_rates_2010 %>% filter(`route` == a_route)

  noncov_rate_2011 = route_info_2011$noncoverage_proportion

  num_noncoverage_2010 = route_info_2010$noncoverage_count
  total_2010 = route_info_2010$total_ride_count
```

```
  res = binom.test(num_noncoverage_2010,
                   total_2010,
                   noncov_rate_2011,
                   alternative = "greater")

  p_val = res$p.value
  p_vals = append(p_vals, p_val)
}
```

Step 6: Adjust the p-values for multiple testing issue and report the routes with significant change in non-coverage rate, i.e. the routes that have statistically significant change between 2010 and 2011

```
p_vals = p.adjust(p_vals, method = 'fdr')
final1 = data.frame(noncoverage_rates_2010$route, p_vals)
final1 = final1 %>% filter(p_vals < 0.05)
length(final1$noncoverage_rates_2010.route)
```

```
## [1] 76
```

After adjusting for multiple testing issue (we used FDR control), we report 76 significant routes out of 159 `busy_routes` we considered.

Is this result reliable? In the next subsection, we further discuss one potential limitation of this regression prediction-based approach.

2.2 Discussion

One potential limitation of the result above is that, there is a possibility that the composition of the riders of a route (especially for covariates that we could not control for, such as age) changed between 2010 and 2011, which resulted in the change in the ride duration. While we are not able to see if the unobserved confounders did have significant change, we may take a look at the observed covariates and see their distribution in both years, which could give us some evidence of whether this potential limitation has a significant impact or not.

First, we take a look at the overall composition of the covariates of both years.

(i) 2010 and 2011 member vs non-member

Figure 1

(ii) 2010 and 2011 day of week

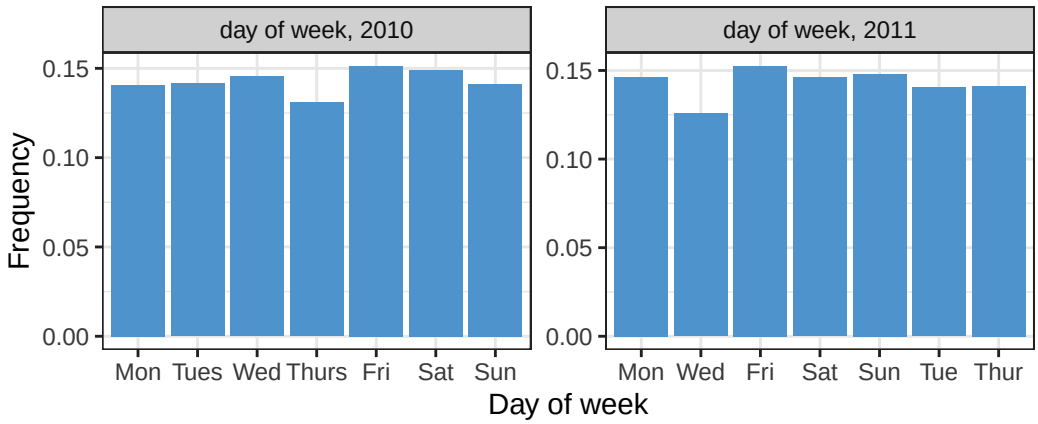


Figure 2

(iii) 2010 and 2011 hour of day

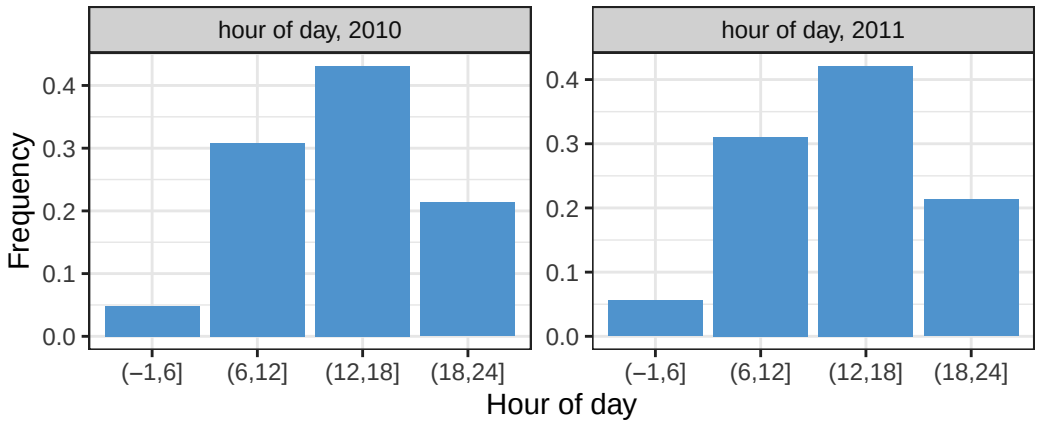


Figure 3

Using the above three plots, we visually inspect and compare the overall distributions of membership, day of week, and hour of day in 2010 and 2011. While there is a little perturbation in the distribution of hour of day, in general we didn't see significant changes in the composition of the observed covariates.

Next, we examine a randomly selected route's composition of the same set of covariates in 2010 and 2011, to see if this still holds on the route-level.

(i) 2010 and 2011 member vs non-member – route `31202-31214`

Figure 4

(ii) 2010 and 2011 day of week – route `31202-31214`

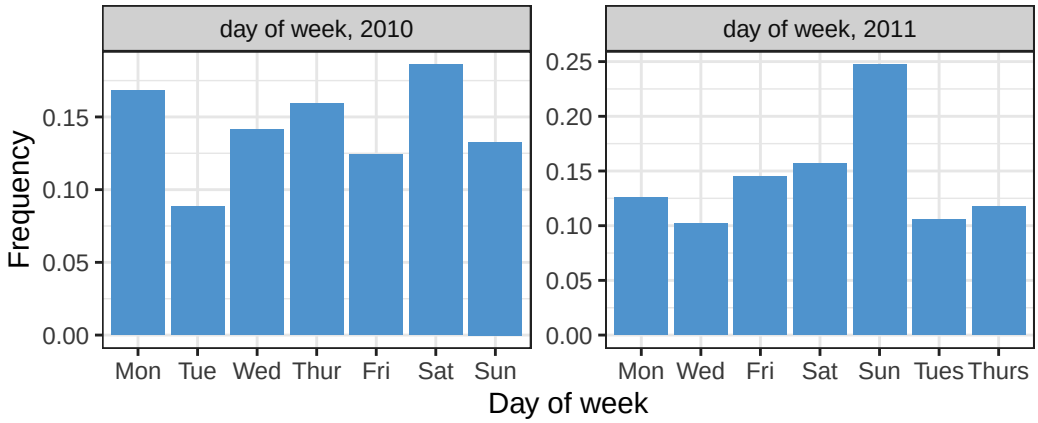


Figure 5

(iii) 2010 and 2011 hour of day – route `31202-31214`

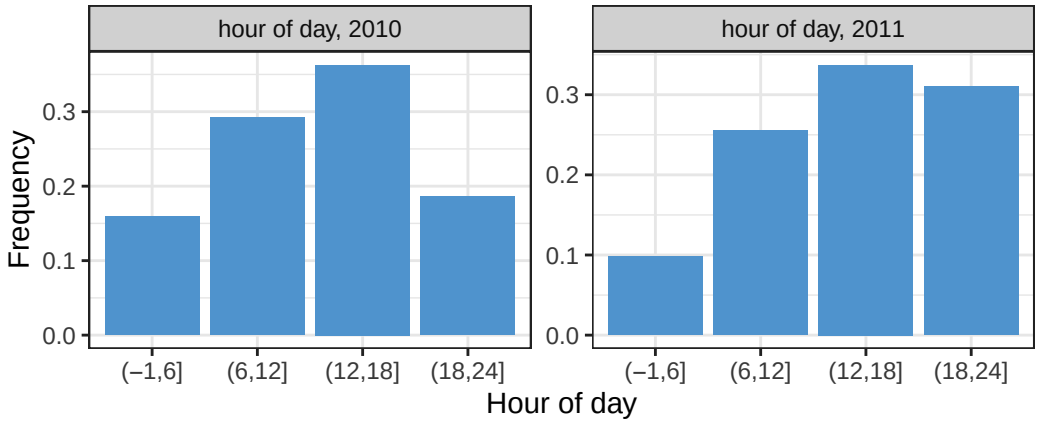


Figure 6

However, when we examine the above plots generated from a randomly selected significant route we identified, there is indeed more variations in the observed covariates between 2010 and 2011. While we controlled for these observed covariates, this suggests that the composition of other unobserved confounders such as age might also change, which can cause the change in the non-coverge rates we found.

Because of the limitation of the data set, we are not able to control for variables such as age, and we can only assume that the composition of the unobserved confounders doesn't change for each route across 2010 and 2011.

Ideally, we could have better control for the effect of the potential change in the composition of the unobserved confounders between 2010 and 2011 by more carefully select the testing set. Instead of using all the 2010 data as the testing set, we may try to find the samples that are "similar" (in terms of the covariates' values) to the samples in the hold-out set. Supposedly, unobserved confounders are more likely to be the same if the observed covariates are the same or at least similar. In this way, we may reduce the effect of the potential change in the composition of the unobserved confounders between 2010 and 2011.

Section III: Different model for different routes

Now we consider the second way of constructing the regression model. That is, instead of letting all the routes share the same coefficients, we construct a separate regression model for each route. Then, we follow the same procedure used in section II to test for the difference in the non-coverage rates on the hold-out set and test set we get when we use the model to construct prediction intervals.

3.1 Implementation of the Test

We perform the same initial data cleaning as in section II, and again we want to make sure for each route we train a model on, there is enough data. That is, we are only considering routes with more than 100 rides in both 2010 Sept-Dec and 2011 Sept-Dec:

```
busy_routes_df_2011 <- df %>%
  filter(year==2011) %>%
  group_by(route) %>%
  summarise(ride_counts = n()) %>%
  filter(ride_counts >= 100)

busy_routes_2011 <- busy_routes_df_2011$route


busy_routes_df_2010 <- df %>%
  filter(year==2010) %>%
  group_by(route) %>%
  summarise(ride_counts = n()) %>%
  filter(ride_counts >= 100)

busy_routes_2010 <- busy_routes_df_2010$route
# we only want to make predictions for routes that have enough
# rides in both years
busy_routes <- intersect(busy_routes_2011, busy_routes_2010)

# split data by year
df11 <- df %>% filter(year == 2011 & route %in% busy_routes) %>%
  select(c(log_duration, member, year, month, day, hour,
           day_of_week, daily_temp, weekend, route))

df10 <- df %>% filter(year == 2010 & route %in% busy_routes)
```

Then, we iterate through the `busy_routes`. For each iteration, we filter for data of the route we are currently considering. Then, follow the same steps in the outline. That is, we

(1) randomly split the 2011 data set into a training set and a hold-out/validation set;

(2) train a model using the training set. In this case `route` is not a covariate anymore (i.e. the regression formula in this case is `log_duration` ~ `member` + `month` + `day_of_week` + `factor(hour)` + `daily_temp`);

(3) aggregate the hold-out set;

(4) apply the model onto the aggregated hold-out set, construct weighted prediction intervals, find non-coverage rate of the prediction intervals on the hold-out set;

(5) apply the model onto the aggregated testing set, construct weighted prediction intervals, find non-coverage rate of the prediction intervals on the testing set;

(6) perform a binomial test on the difference between the two non-coverage rates and get a p-value.

After all the iterations, we adjust the p-values for multiple testing issues (using FDR control).

```
p_vals = c()
```

```
for (a_route in busy_routes) {

  # filter for this route's data
  route_df11 <- df11 %>% filter(route == a_route)
  route_df10 <- df10 %>% filter(route == a_route)

  # create hold-out set for 2011 data
  picked = sample(seq_len(nrow(route_df11)), size = nrow(route_df11)*0.7)
  df11_train =route_df11[picked,]
  df11_holdout =route_df11[-picked,]

  # train a model for this route
  # print(a_route)
  lm_2011 = lm(log_duration ~ member + daily_temp + month + day_of_week + hour,
               data = df11_train)

  # aggregate on 2011 hold-out set
  df11_holdout_agg <- df11_holdout %>%
    select(-c(year, weekend)) %>%
    group_by(month, day, hour, member) %>%
    mutate(log_duration = median(log_duration)) %>%
    mutate(counts = n()) %>%
    ungroup() %>%
    distinct() %>%
    arrange(month, day, hour, member)

  # make sure that the hold-out set doesn't have new levels
  df11_holdout_agg <- df11_holdout_agg %>%
    filter(hour %in% unique(df11_train$hour)) %>%
    filter(day_of_week %in% unique(df11_train$day_of_week))

  # make predictions for the hold-out set
  prediction_2011_holdout <- data.frame(predict(lm_2011, newdata = df11_holdout_agg,
                                      interval = "predict", level = 0.95,
                                      weights = df11_holdout_agg$counts)) %>%
    mutate(true_duration = df11_holdout_agg$log_duration)

  # find non-coverage rate on hold-out set
  holdout_noncover = nrow(prediction_2011_holdout %>%
                            filter(true_duration>upr | true_duration<lwr))
  holdout_total = nrow(prediction_2011_holdout)
  holdout_noncover_rate = holdout_noncover/holdout_total

  # aggregate on 2010 test data
  df10_agg <- route_df10 %>%
    select(-c(year, weekend)) %>%
    group_by(month, day, hour, member) %>%
    mutate(log_duration = median(log_duration)) %>%
    mutate(counts = n()) %>%
    ungroup() %>%
    distinct() %>%
```

```
    arrange(month, day, hour, member)

  # make sure that the 2010 test set doesn't have new levels
  df10_agg <- df10_agg %>%
    filter(hour %in% unique(df11_train$hour)) %>%
    filter(day_of_week %in% unique(df11_train$day_of_week))

  # make predictions on 2010 test set
  prediction_2010 <- data.frame(predict(lm_2011, newdata = df10_agg,
                                        interval = "predict", level = 0.95,
                                        weights = df10_agg$counts)) %>%
    mutate(true_duration = df10_agg$log_duration)

  # find non-coverage rate on test set
  test_noncover = nrow(prediction_2010 %>% filter(true_duration>upr | true_duration<lwr))
  test_total = nrow(prediction_2010)
  test_noncover_rate = test_noncover / test_total

  # do binomial test on the 2 non-coverage rates; find p value
  res = binom.test(test_noncover,
                   test_total,
                   holdout_noncover_rate,
                   alternative = "greater")
  p_val = res$p.value
  p_vals = append(p_vals, p_val)
}
```

```
p_vals = p.adjust(p_vals, method = 'fdr')
final2 = data.frame(busy_routes, p_vals)
final2 = final2 %>% filter(p_vals < 0.05)
length(final2$busy_routes)
```

```
## [1] 66
```

As shown above, when we build a separate model for each route, we identified 66 out of 159 `busy_routes` we considered.

3.2 Discussion

First, the result of this approach (building a separate model for each route) is also subject to the same potential limitation imposed by the change of composition in the unobserved confounders, which has already been discussed in section 2.2.

Now, we can find the intersection of the significant routes detected by these two ways.

```
intersect(final2$busy_routes, final1$noncoverage_rates_2010.route)
```

```
##  [1] "31101-31201" "31101-31213" "31103-31106" "31104-31104" "31104-31106"
##  [6] "31104-31200" "31105-31101" "31106-31101" "31110-31205" "31110-31214"
## [11] "31200-31101" "31200-31110" "31200-31111" "31201-31111" "31205-31205"
## [16] "31205-31229" "31206-31100" "31212-31200" "31213-31201" "31213-31214"
## [21] "31214-31201" "31215-31215" "31217-31217" "31235-31235" "31303-31302"
## [26] "31602-31104" "31613-31606" "31613-31622"
```

```
length(intersect(final2$busy_routes, final1$noncoverage_rates_2010.route))
```

```
## [1] 28
```

We that the overlap between the two sets of discoveries is roughly 40% of each of them. This indicates that whether to build separate model for each route does affect the result, and which method to use would depend on the assumptions we want to make (i.e. whether different routes should share the same set of coefficients for the covariates). On the other hand, the overlapped routes identified by both methods may suggest that they more likely had undertaken changes between 2010 and 2011.

For more discussion on the comparison of different approaches and the limitations of these approaches, please refer to the last chapter of this report.

Chapter Two: Permutation Test

Section I: Introduction to the Permutation Test

This permutation approach finds routes that seem to show change in duration over time, and then permute the duration to see if the change is or is not likely to appear when data is randomly shuffled. For routes that is statistically unlikely to have the actual changes when randomly shuffled, we believe the change is unlikely to be due to random fluctuation in data but indeed an actual change in duration. Thus, these routes will be marked as significant. Because we are testing for hundreds of routes simultaneously, we also used multiple testing correction to control for false discovery rate when computing p values and making inferences.

1.1 Outline of the steps taken

Step 0: Data cleaning, filter out only routes with sufficient number of records (>500) to perform the permutation test.

Step 1: Run the permutation test on each route without controlling for any covariate to guage its efficiency in identifying routes.

Step 2: Run the task specific permutation test, which is the permutation test with control for covariate, on the routes. With control, only data within the same group category get shuffled. For example, morning rides will only be shuffled with other morning rides, and member rides will only be shuffled with other members' rides. This approach preserves original trends and structures in the data so that original trends will be preserved.

Step 3: Evaluate the task specific permutation test's reliability by simulations. We constructed three fake datasets: completely null distributions for all routes, strictly decreasing distribution for all routes, and half null half decreasing distributions for all routes. We then run our task specific permutation test to see if it correctly identified the non-null routes.

Step 4: Explore possible improvements on the task specific permutation test. Here, two approaches are attempted:

Firstly, we tried to use finer bucketing of data on routes with sufficiently large amount of data (>1000) to see it's effectiveness. For these popular routes, we divided covariate like temperature, day of week, and hour of day more meticulously. For example, we split temperature into 10F intervals, compared to before splitting them into 20F intervals.

Secondly, we explored other test statistics, for example by dividing the data in middle for each route and calculate the mean and median differences between the two halves. These method might be able to better identify sudden changes compared to absolute correlation test, which is more effectively at finding long-term changes.

Step 5: Find common significant routes identified by different test statistics in permutation test. Do research to explore the routes on map and see if our results actually correspond to real-world constructions, or explore possible reasons for our significant results.

1.2 Load Data and Data Cleaning

```
# Remove outliers with IQR range test
outliers <- function(x) {
    q25 <- quantile(x, probs=.25)
    q75 <- quantile(x, probs=.75)
```

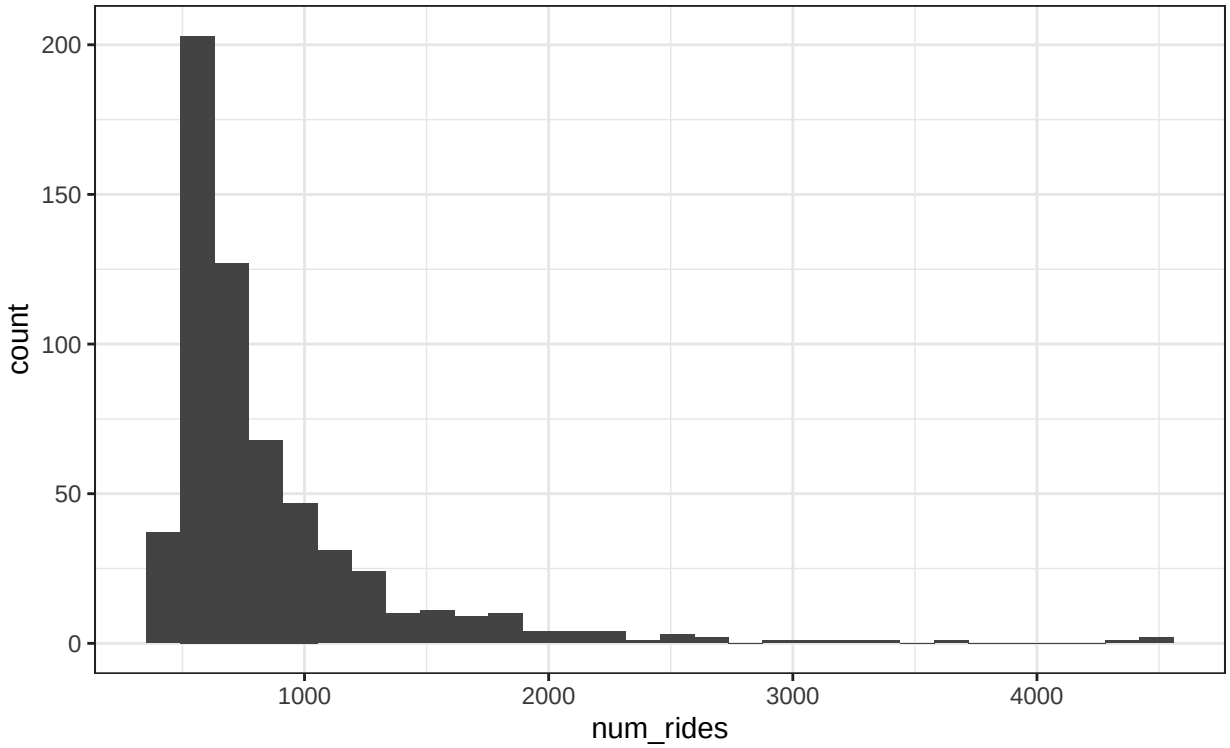
```
    interval <- q75 - q25
    x > q75 + (interval * 1.5) | x < q25 - (interval * 1.5)
}

# Just remove 5% data on the top end.
outliers2 <- function(x) {
  x > quantile(x, probs=0.95)
}

# Temperature Data
temp_data <-
  read.table("data/temperature-data.txt",
             skip = 1,
             col.names = c('time1', 'time2', 'daily_max', 'daily_min')) %>%
  mutate(year = as.numeric(substring(time2, 1, 4)),
         month = as.numeric(substring(time2, 6, 7)),
         day = as.numeric(substring(time2, 9, 10)),
         daily_temp = (daily_max + daily_min) / 2)

# Bike-share Data
load("data/bikedata.RData")
colnames(starttime) = c("year", "month", "day", "hour", "minute", "second")
df <- data.frame(log_duration = log(duration), station_start, station_end,
                 starttime, day_of_week, days_since_Jan1_2010, member) %>%
  # Join with Temperature Data
  left_join(temp_data, by = c('year','month','day')) %>%
  # Add Weekend/weekday
  mutate(weekend = day_of_week %in% c("Saturday", "Sunday")) %>%
  # Add Hour of the Day (with different degree of preciseness)
  mutate(hour_of_day = cut(hour, c(-1, 12, 24))) %>%
  mutate(hour_of_day_2 = cut(hour, c(-1, 6, 12, 18, 24))) %>%
  # Add temperature buckets (with different degree of preciseness)
  mutate(temp = cut(daily_temp, c(20, 40, 60, 80, 100))) %>%
  mutate(temp_2 = cut(daily_temp, c(20, 30, 40, 50, 60, 70, 80, 90, 100))) %>%
  # Assign Id to Route
  group_by(station_start, station_end) %>%
  mutate(route = paste0(station_start, "-", station_end)) %>%
  # Filter out routes with less than 500 records
  filter(length(log_duration) >= 500, !outliers(log_duration)) %>%
  ungroup()

# Drop unnecessary columns
df <- subset(df, select=-c(station_start, station_end, month, day,
                           minute, second, time1, time2, daily_max, daily_min))
```

1.3 Data Distribution

We first check that how much data is left after cleaning - we are left with 603 observations.

```
df %>%
  group_by(route) %>%
  summarize(num_rides = length(log_duration)) %>%
  ggplot(aes(x=num_rides)) +
  geom_histogram()
```

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



Section II: Permutation test without Any Control

2.1 Permutation test code

For each route, we are interested in the question if there is any change in ride duration at any point over the span of 2010-2011. In the language of hypothesis testing, this translates to the hypothesis that ride duration is independent of the days_since_Jan1_2010. Rejecting this hypothesis would mean that ride duration depends on the date - in other words change over time. We observe that the absolute value of correlation between duration and days_since_Jan1_2010 would be relatively small if the hypothesis is true and large otherwise, making it a good candidate statistic for the permutation test. Therefore, we first perform permutation test without any confounder control to gauge how well it works.

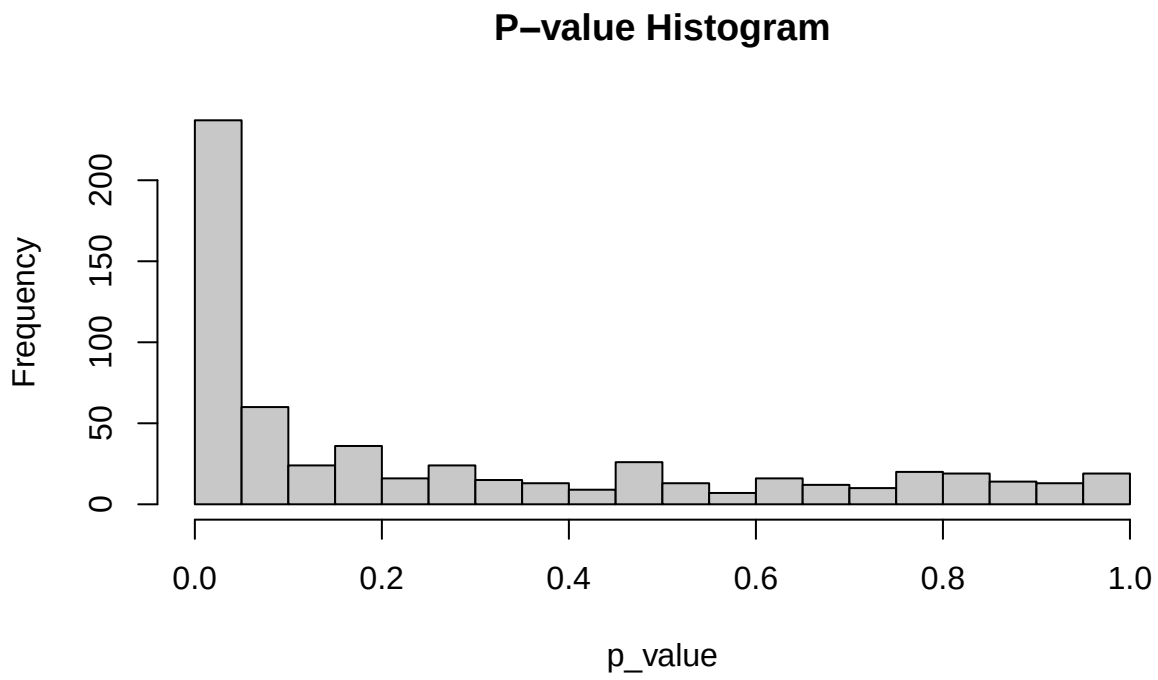
It should be noted in advance that total number of routes left after data cleaning and filtering process is 603.

```
permutation.test <- function(duration, days_from_start, n=1000) {
  T <- abs(cor(duration, days_from_start))
  distribution <- c()
  for (i in 1:n) {
    distribution[i] <- abs(cor(sample(duration, replace=FALSE), days_from_start))
  }
  p_val <- (1 + sum(distribution >= T)) / (1 + n)
  return (p_val)
}
p_vals <- df %>%
  group_by(route) %>%
  summarize(p_val = permutation.test(log_duration, days_since_Jan1_2010))
```

```r
# Adjust p_vals for multiple testing
p_vals$p_adjust = p.adjust(p_vals$p_val, method = 'fdr')
```

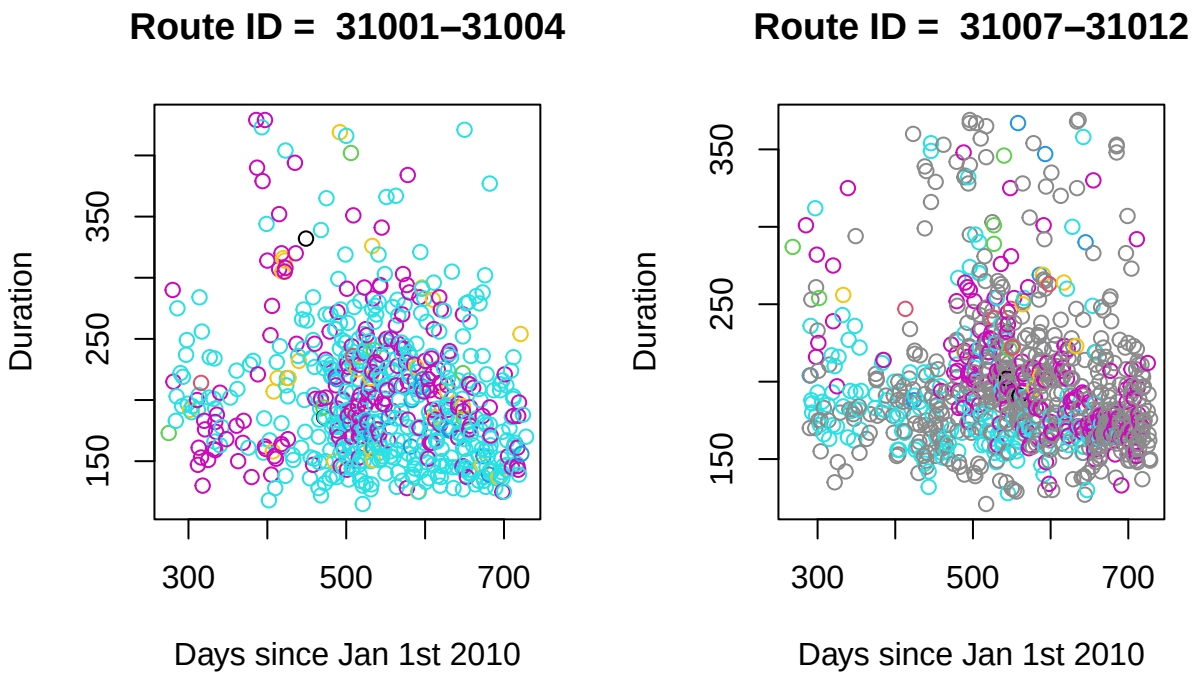
2.2 P-value Histogram and Graphs

```r
hist(p_vals$p_adjust, xlab="p_value", main="P-value Histogram", breaks=20)
```



We plot the top routes that have the strongest signals and see if they indeed exhibit some kind of change. Each color represents a group.

```r
plot_route_change <- function(df, p_vals, n=2) {
  sorted_pvals <- p_vals[order(p_vals$p_adjust), ]
  par(mfrow=c(round(n/2),2))
  for (i in 1:n) {
    route_df <- filter(df, route == sorted_pvals$route[i]) %>%
      mutate(grouping = paste(member, hour_of_day_2))
    plot(route_df$days_since_Jan1_2010, exp(route_df$log_duration),
         xlab="Days since Jan 1st 2010", ylab="Duration",
         col=factor(route_df$grouping),
         main=paste("Route ID = ", sorted_pvals$route[i]))
  }
}
plot_route_change(df, p_vals)
```



For these two plots, we observe a gradual decrease in the duration as time goes on from 2010 to 2011.

Section III: Task-Specific Permutation Tests

However, with confounding variables in the play, a simple permutation scheme will not suffice. For example, it is plausible that members generally ride faster than non-members (who may not be frequent riders), and so the correlation between ride duration and date is confounded by the membership status. To account for these many possible confounders, we use a task-specific permutation testing scheme.

Let X_j be the explanatory variables, and Y be the response, which in our case is the ride duration. Suppose X_0 is the days_since_Jan1_2010, what we are actually interested in is the following hypothesis test:

$$H_0 : Y \perp\!\!\!\perp X_0 \mid X_{-0} = \text{all other explanatory variables}$$

However, due to limitations in the dataset, it is infeasible to control for all explanatory variables, as we don't have enough data for each possible variable combinations, and we have not collected every possible ambient variables, such as weather, and the riders' ages. Therefore, we opted to control for a limit set of possible confounders. In particular, we control for the following variables:

- membership
- day of the week
- hour of the day
- temperature

During the permutation tests, we only permute data that has the same membership status, is on the same day of the week, and so on. If we let \$X = \$ days_from_Jan1_2010, \$Z = \$ (membership status, day, hour, temperature), \$Y = \$ duration, and $\tilde{X}$ be another days_from_Jan1_2010 data that got permuted, then under the null hypothesis that

$$H_0 : Y \perp\!\!\!\perp X \mid Z$$

and an additional assumption that $\mathbb{P}(X \mid Z) = \mathbb{P}(\tilde{X} \mid Z)$, we have

$$
\begin{aligned}
\mathbb{P}(X, Y, Z) &= \mathbb{P}(X \mid Y, Z)\mathbb{P}(Y, Z) \\
&= \mathbb{P}(X \mid Z)\mathbb{P}(Y, Z) \qquad \text{Conditional Independence} \\
&= \mathbb{P}(\tilde{X} \mid Z)\mathbb{P}(Y, Z) \\
&= \mathbb{P}(\tilde{X}, Y, Z)
\end{aligned}
$$

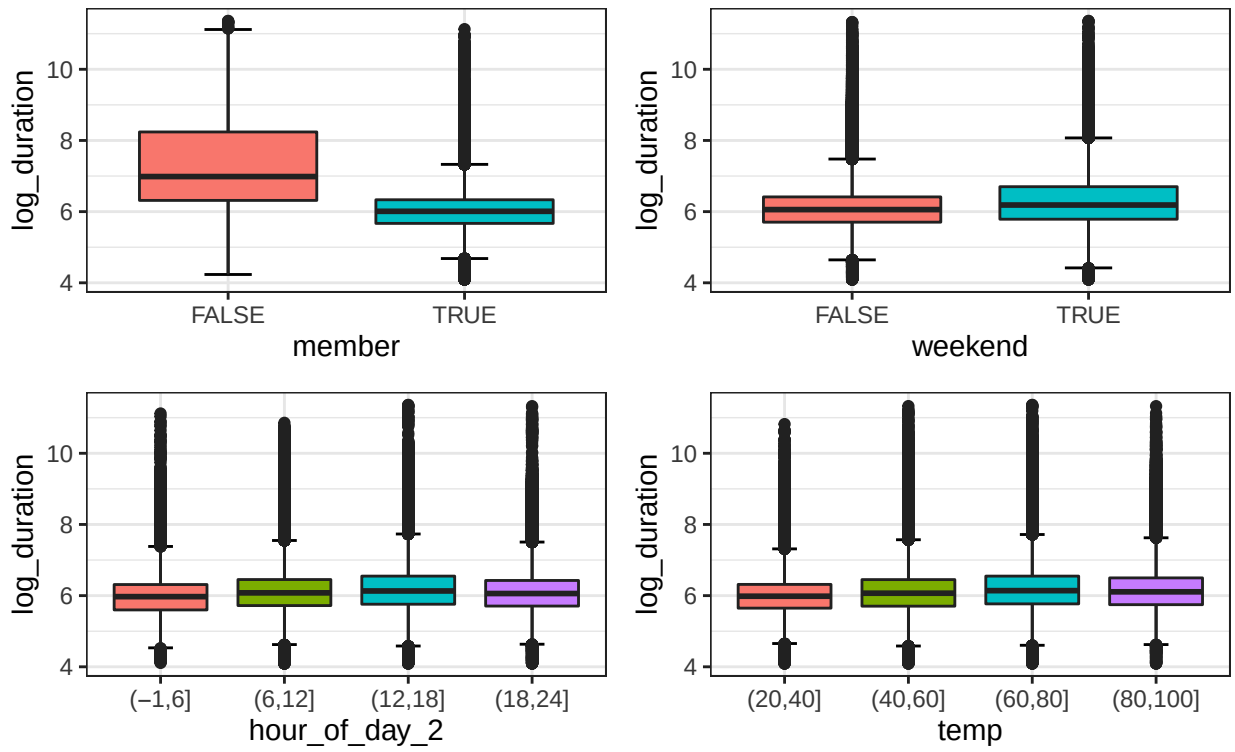
Therefore, the distribution remains unchanged. This additional assumption that $\mathbb{P}(X \mid Z) = \mathbb{P}(\tilde{X} \mid Z)$ is reasonable in this case, as we are only saying that a member, or non-member, has the same probability of riding on any day (no further knowledge). As such, we adjust the permutation test scheme to be task specific to account for confounders.

Membership status, the day of the week (weekend or weekday), hour of the day, and temperature might be possible confounders. Adjusting the permutation test scheme to also account for these factors can ensure certain original trends in the data, for example the proportion of weekend rides, remain unchanged. We believe these are intuitive confounders as we would expect rides on Monday or Saturday, at 3am or 3pm, in 80F or 50F to be different.

3.1 Preliminary data distribution exploration

We first plot to compare between the overall duration for member vs non-member, weekend vs weekday, day vs night, and temperature intervals to see if there's a difference in duration in these group categories.

```
plots <- c()
vars <- c("member", "weekend", "hour_of_day_2", "temp")
for (i in 1:length(vars)) {
  plot <- ggplot(data=df, mapping=aes_string(x=vars[i], y="log_duration", fill=vars[i])) +
    stat_boxplot(geom = "errorbar", width = 0.20) +
    geom_boxplot() + theme(legend.position = "none")
  plots[[i]] <- plot
}
grid.arrange(grobs=plots, ncol = 2)
```



We make the following observations:

- Members overall have lower log_durations than non-members, meaning members spend shorter time riding overall.
- Weekends have slightly higher log_durations compared to the weekdays, suggesting people spend longer time to ride on weekends in general.
- There is little difference in log_duration between different hours, but it seems that durations are a little longer on afternoons.
- There is also little difference in log_duration across different temperatures, but it seems that hotter temperature relates with slightly longer durations.

We will now run the task specific permutation tests.

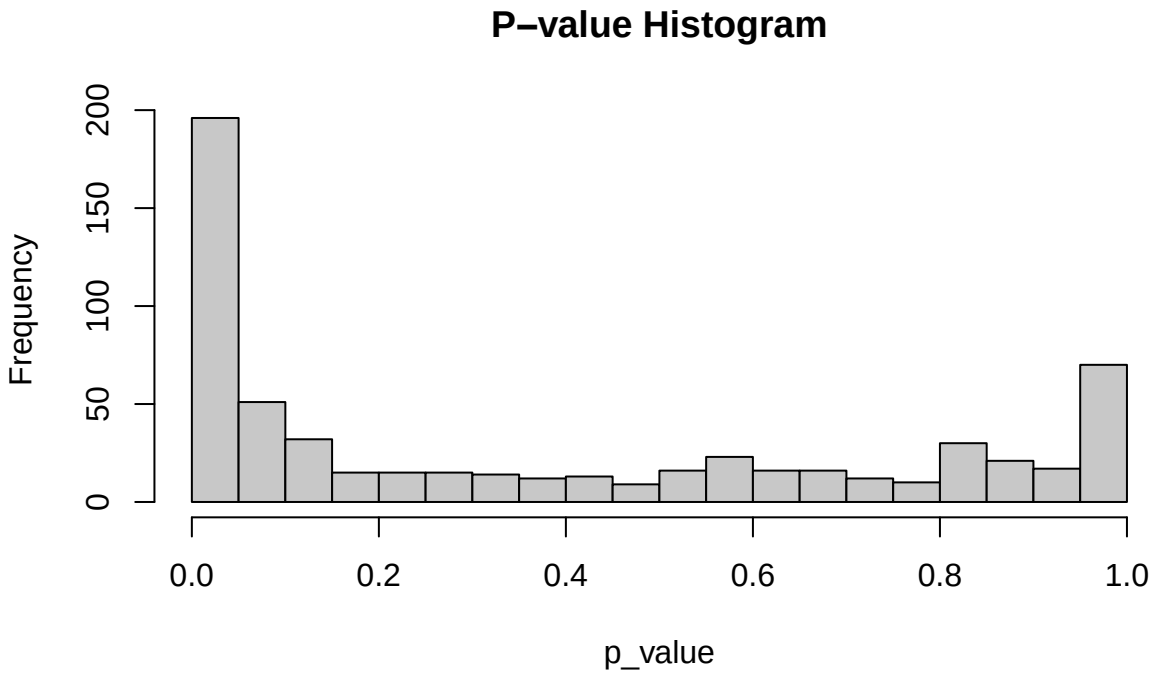
3.2 Task specific permutation tests code

```
# Task specific Permutation Tests
permutation.task_specific_test <- function(duration, days_from_start, compute_t, grouping, n=1000) {
  t <- compute_t(duration, days_from_start)
  distribution <- c()
  for (i in 1:n) {
    permuted_duration <-
      ave(duration, grouping,
          FUN = function(x) if (length(x) == 1) x else sample(x))
    distribution[i] <- compute_t(permuted_duration, days_from_start)
  }
  p_val <- (1 + sum(distribution >= t)) / (1 + n)
  return (p_val)
}

# Absolute value of correlation as the statistic
abscor_t <- function(log_duration, days_from_start) abs(cor(log_duration, days_from_start))

# Control for Membership status, day of the week (2), hour of the day(2), and temperature(4)
# parenthesis represents how many categories there are for each variable
group_controlled_pvals <- df %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010, abscor_t,
                                                   paste(member, weekend, hour_of_day, temp))) %>%
  # Adjust p_vals for multiple testing
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))
```

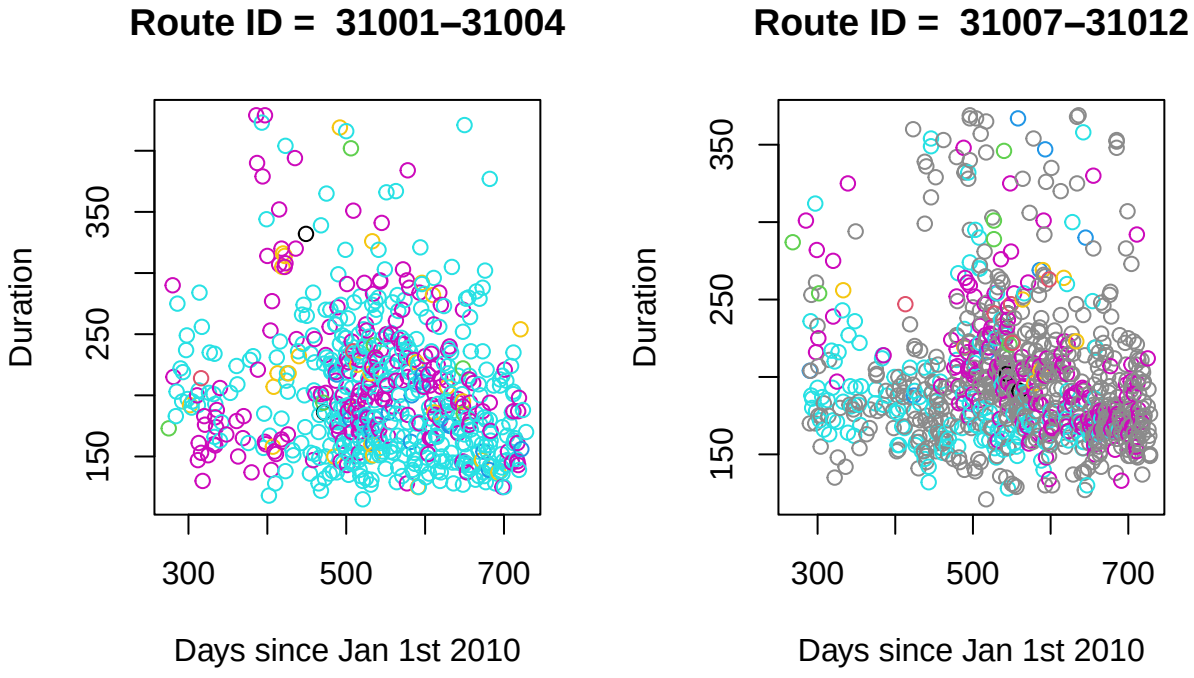
```
## [1] "Number of significant routes using absolute correlation statistics: 196"
```

3.3 Task specific P-value Histogram and Graphs

```
# P-value distribution
hist(group_controlled_pvals$p_adjust,
     xlab="p_value", main="P-value Histogram", breaks=20)
```




```
# Top Routes
plot_route_change(df, group_controlled_pvals, n = 2)
```



Similarly, we observe a decreasing trend in duration from 2010 to 2011, though there is a reduction in the number of discovery due to control.

3.4 Sanity check of task specific permutation test

To check if our permutation test is working properly, we generate null, non-null, half null and half non-null datasets to check if the permutation test will give us the expected answers.

```
# run sanity checks
sanity_check <- function(df) {
  # run the permutation task specific test on the null dataset
  pvalues <- df %>%
    group_by(route) %>%
    summarize(p_val = permutation.task_specific_test(random, days_since_Jan1_2010, abscor_t,
                                                     paste(member, weekend, hour_of_day, temp),
                                                     n=100)) %>%
    # Adjust p_vals for multiple testing
    mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

  # Identify routes with p-val < 0.05
  return (nrow(pvalues %>% filter(p_adjust < 0.05)))
}

# null dataset
df_null <- df %>%
  group_by(route) %>%
  mutate(random = rnorm(n(), mean = mean(log_duration), sd = sd(log_duration))) %>%
  ungroup()
print(paste0("Number of significant routes for null dataset: ", sanity_check(df_null)))
```

```
## [1] "Number of significant routes for null dataset: 0"
```

```
# signal dataset
df_nonnull <- df %>%
  group_by(route) %>%
  mutate(random = n():1) %>%
  ungroup()
print(paste0("Number of significant routes for signal dataset: ", sanity_check(df_nonnull)))
```

```
## [1] "Number of significant routes for signal dataset: 603"
```

```
# half signal half null dataset
df_halfnull <- df %>%
  group_by(route) %>%
  filter(route %in% group_controlled_pvals$route[1:round(nrow(group_controlled_pvals)/2)]) %>%
  mutate(random = n():1)

df_halfnonnull <- df %>%
  group_by(route) %>%
  filter(route %in% group_controlled_pvals$route[(round(nrow(group_controlled_pvals)/2) + 1):nrow(group_co
  mutate(random = rnorm(n(), mean = mean(log_duration), sd = sd(log_duration)))

df_halfhalf <- rbind(df_halfnull, df_halfnonnull)
print(paste0("Number of significant routes for half/half dataset: ", sanity_check(df_halfhalf)))
```

```
## [1] "Number of significant routes for half/half dataset: 307"
```

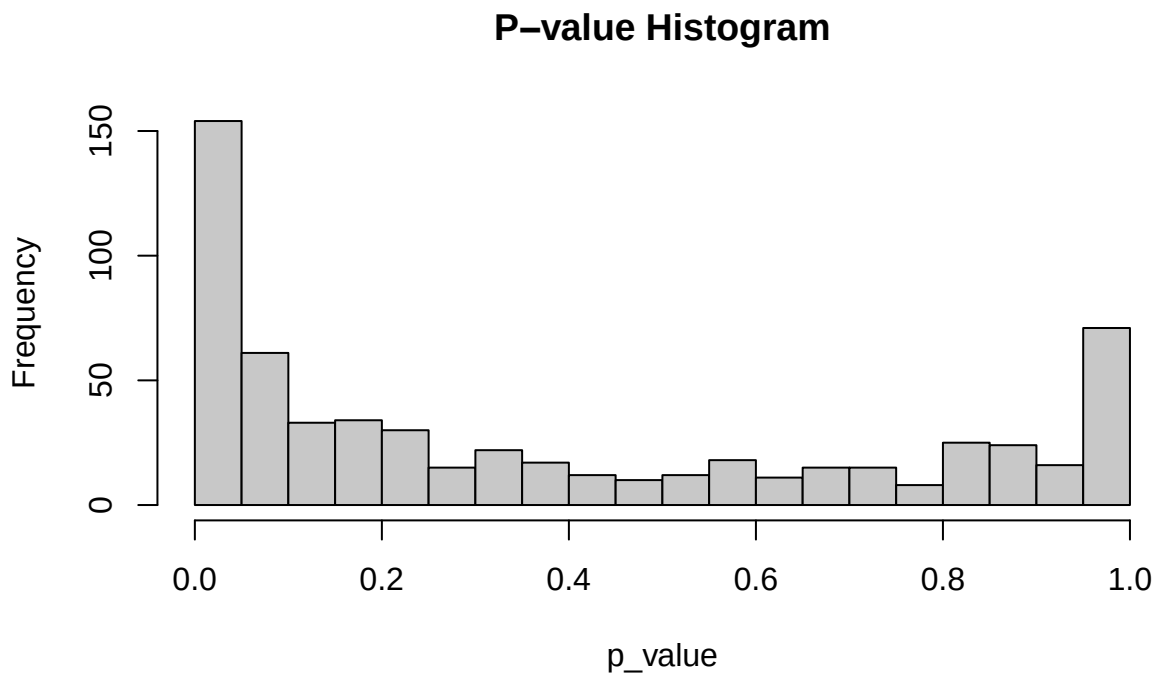
We see that there is no significant route for the null dataset generated by random normal distribution for each route, with mean being the original mean of the route's log_duration and standard deviation being the original standard deviation of the route's log_duration. Similarly, All the 603 routes are significant for the all signal dataset. The signal dataset is generated by decreasing each route's log_duration as time goes on.

On the other hand, for the data with half of the routes following random normal distribution and half of the routes following a decreasing trend, we found 302 routes to be significant by the correlation test. By checking with null, signal, and half-null half-signal dataset, we are able to conclude that our task specific permutation test is working properly.

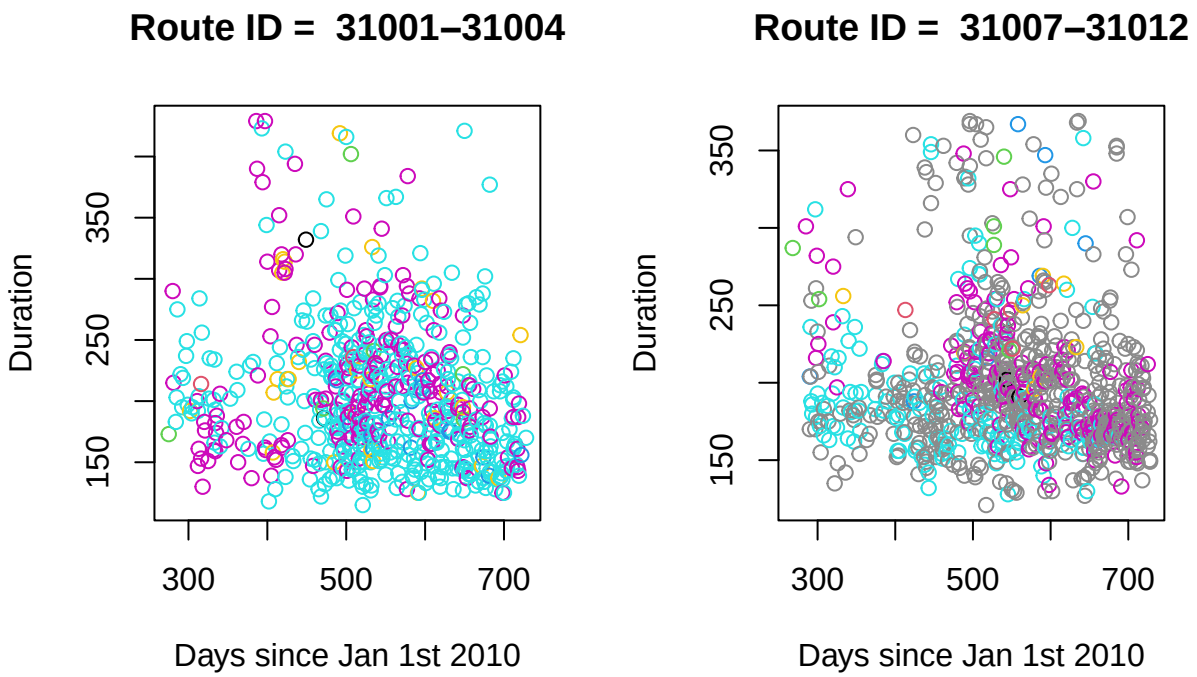
3.5 Finer Bucketing of Data with task specific permutation test

We then experiment with a finer bucketing of data, where weekdays changed from 2 categories (weekday/weekend) to 7 categories, temperature changed from 20F intervals to 10F intervals, and hour changed from 2 12-hour groups to 4 6-hour groups.

```
# FINER control for Membership status, day of the week (7), hour of the day (6), and temperature (8)
# parenthesis represents how many categories are for each variable
group_controlled_pvals2 <- df %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010, abscor_t,
                                                   paste(member, day_of_week, hour_of_day_2, temp_2),
                                                   n=100)) %>%
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

# Plot P-value histogram and top routes
hist(group_controlled_pvals2$p_adjust,
     xlab="p_value", main="P-value Histogram", breaks=20)
```



```
plot_route_change(df, group_controlled_pvals)
```



We see that the number of significant routes reduces, as we are employing a finer control, which likely leads to less power.

Section IV: Exploration of Better Statistics to Detect Sudden Changes

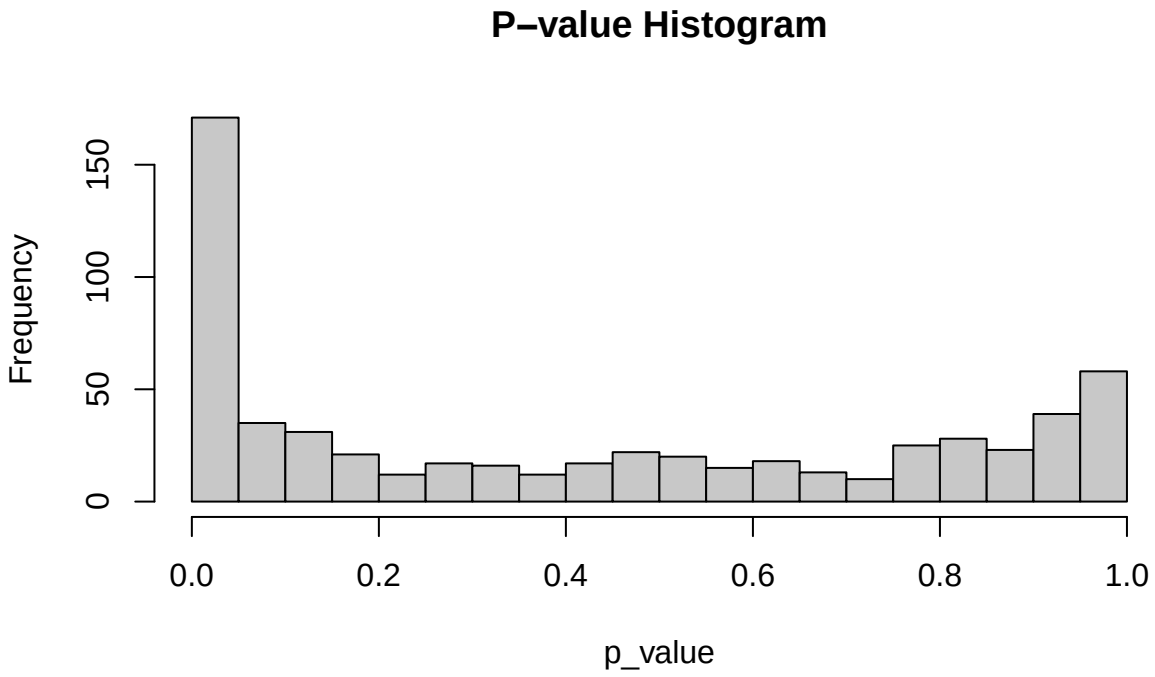
As correlation is more suitable in detecting long-term gradual change in data and not as powerful for detecting bumps, we explore other statistics to better identify bumps and changes in the data.

We perform the same task specific permutation test, but change the test statistics to mean or median instead of correlation. We split the data into the "earlier half" and the "later half" based on days_since_Jan_1st_2010. We then compute and compare the mean or median of the two halves. Permutation test will then be performed to see if the difference between the two halves is indeed statistically significant (for each route).

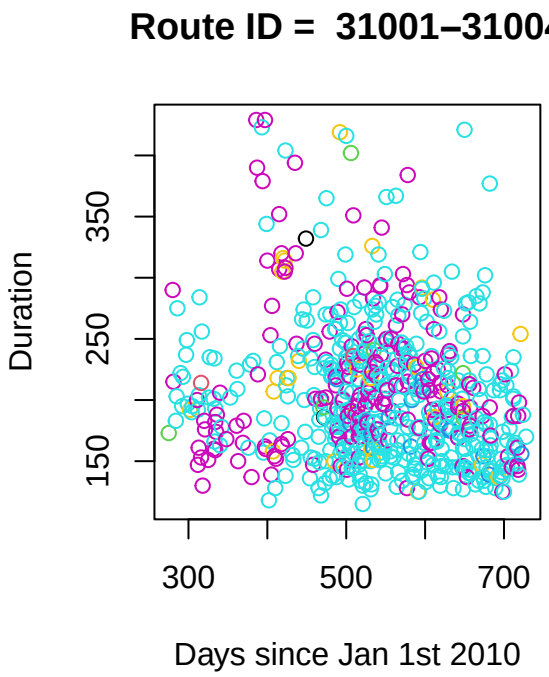
4.1 Using the mean statistic

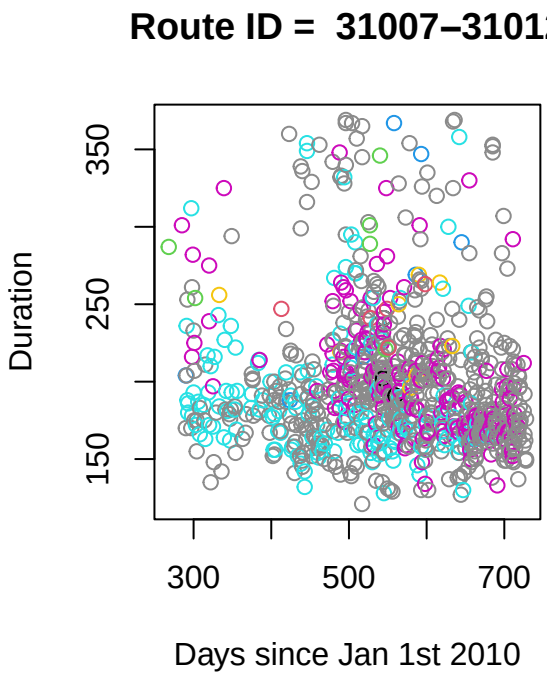
```
# Task specific permutation tests for mean
mean_t <- function(duration, days_from_start) {
  first_half <- duration[1:round(length(duration)/2)]
  second_half <- duration[(round(length(duration)/2) + 1):length(duration)]
  abs(mean(first_half) - mean(second_half))
}

group_controlled_mean_pvals <- df %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010, mean_t,
```

```
                                                    paste(member, weekend, hour_of_day, temp))) %>%
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

hist(group_controlled_mean_pvals$p_adjust,
     xlab="p_value", main="P-value Histogram", breaks=20)
```




```
plot_route_change(df, group_controlled_mean_pvals)
```

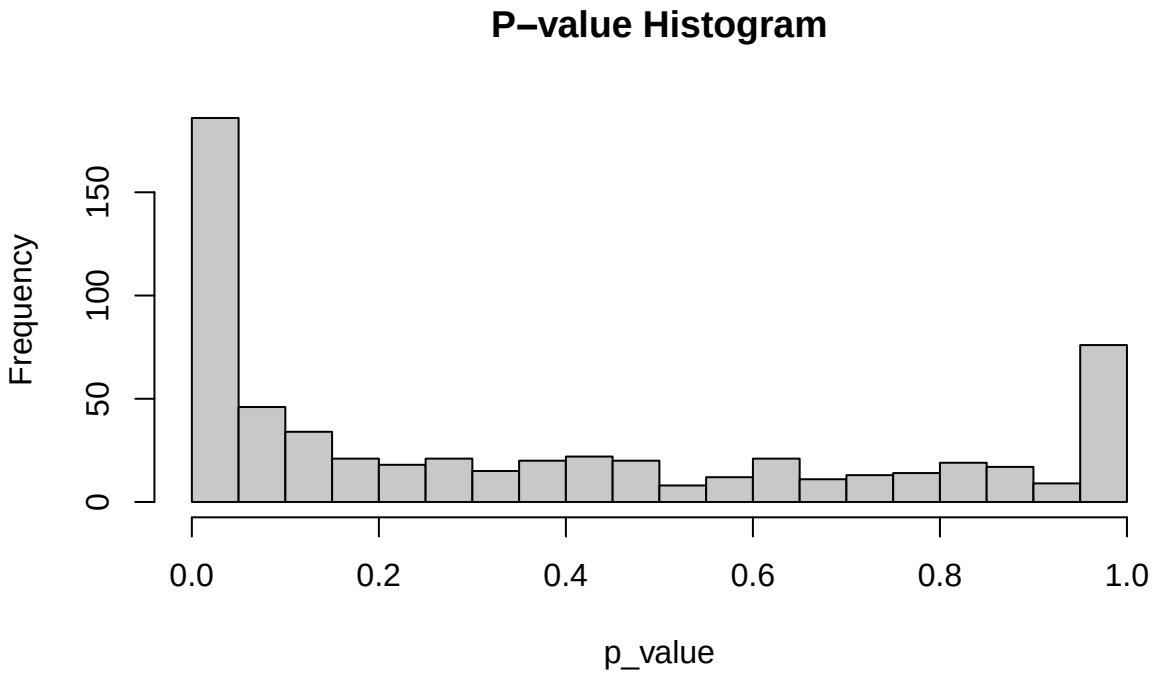




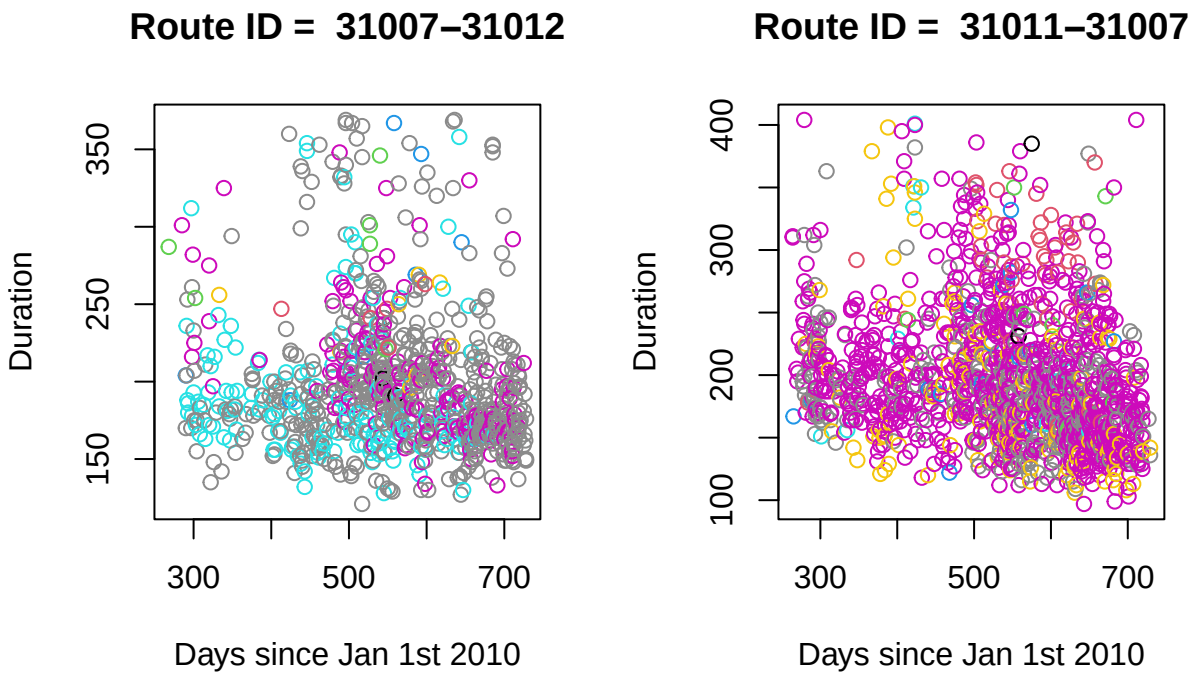

```
## [1] "Number of significant routes using mean statistics: 171"
```

4.2 Using the median statistic

```
# Task specific permutation tests for median
median_t <- function(duration, days_from_start) {
  first_half <- duration[1:round(length(duration)/2)]
  second_half <- duration[(round(length(duration)/2) + 1):length(duration)]
  median(first_half) - median(second_half)
}

group_controlled_median_pvals <- df %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010, median_t,
                                                   paste(member, weekend, hour_of_day, temp))) %>%
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

hist(group_controlled_median_pvals$p_adjust,
     xlab="p_value", main="P-value Histogram", breaks=20)
```




```
plot_route_change(df, group_controlled_median_pvals, n=2)
```




```
## [1] "Number of significant routes using median statistics: 186"
```

4.3 Significant routes with all 3 statistics

We check to see how many routes are marked as having a statistically significant change using all 3 statistics (absolute correlation, mean, and median) with the task specific permutation test.

```
# check the common routes detected by correlation, mean, and median methods
corr_routes <- subset(group_controlled_pvals, p_adjust < 0.05)
mean_routes <- subset(group_controlled_mean_pvals, p_adjust < 0.05)
median_routes <- subset(group_controlled_median_pvals, p_adjust < 0.05)

common_routes <- intersect(intersect(mean_routes$route, median_routes$route), corr_routes$route)
length(common_routes)
```

```
## [1] 124
```

```
head(common_routes, 10)
```

```
##  [1] "31001-31004" "31007-31012" "31007-31013" "31010-31011" "31011-31007"
##  [6] "31011-31010" "31013-31007" "31014-31225" "31100-31201" "31101-31202"
```

4.4 Significant routes exploration

```
stations = c()
for (route in common_routes) {
  for (st in strsplit(route, "-")) {
    stations = c(stations, st)
  }
}
station_counts <- sort(table(stations), decreasing = TRUE)
head(station_counts, 3)
```

```
## stations
## 31200 31623 31228
##    13    12     9
```

These stations correspond to the Dupont circle (31200 & 31201) and the Columbus circle (31623). However, upon inspecting the news & google map, it is unclear if there has been any change to the actual routes during the period. Both stations are among the top used stations per Capital Bikeshare reports. We conjecture that the gradual decrease over time might be more and more people become more familiar with the locations since the program opens in 2010.

Chapter Three: Discussion

Section I: Compare Permutation Test with Regression Test results

1.1 Permutation Test only on data from September to December (both 2010 and 2011)

We then re-run the correlation test with only the data from September to December in 2010 and 2011 to compare with the result from our regression model. Only data from September to December is used because our regression model also only used these data since 2010 does not have data from January to August.

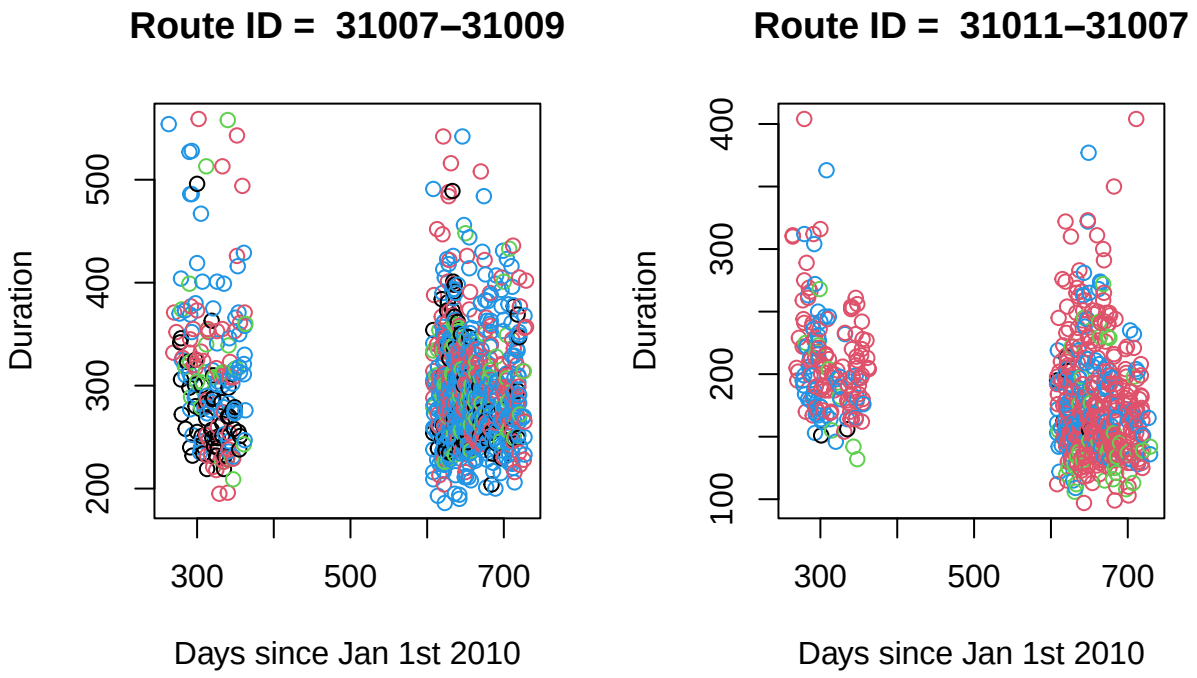
```
# filter out only data from september to december in 2010 and 2011
df_sepdec <- df %>%
  filter(member == TRUE) %>%
  filter(between(days_since_Jan1_2010, 243, 364) | between(days_since_Jan1_2010, 608, 729))

# filter out routes that only have data for 2010 or only have data for 2011
busy_routes <- read.csv("busy_routes.csv")
df_sepdec <- df_sepdec %>% filter(route %in% busy_routes$x)

# test statistics to compare the median between 2010 sep-dec and 2011 sep-dec
median_t_sepdec <- function(duration, days_from_start) {
  first_half <- duration[1:(which(days_from_start > 608)[1] - 1)]
  second_half <- duration[which(days_from_start > 608)[1]:length(duration)]
  median(first_half) - median(second_half)
}

group_controlled_sepdec_pvals <- df_sepdec %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010,
                                                   median_t_sepdec,
                                                   paste(member, weekend, hour_of_day, temp))) %>%
  # Adjust p_vals for multiple testing
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

# Identify routes with p-val < 0.05
significant_routes_sepdec_median <- group_controlled_sepdec_pvals %>%
  filter(p_adjust < 0.05)

# use absolute correlation on the 2010 sep-dec and 2011 sep-dec
group_controlled_sepdec_pvals2 <- df_sepdec %>%
  group_by(route) %>%
  summarize(p_val = permutation.task_specific_test(log_duration, days_since_Jan1_2010,
                                                   abscor_t,
                                                   paste(member, weekend, hour_of_day, temp))) %>%
  # Adjust p_vals for multiple testing
```

```
  mutate(p_adjust = p.adjust(p_val, method = 'fdr'))

# Identify routes with p-val < 0.05
significant_routes_sepdec_abscor<- group_controlled_sepdec_pvals2 %>%
  filter(p_adjust < 0.05)
```

```
## [1] "Number of significant routes using median statistics (only Sep to Dec): 59"
```

```
## [1] "Number of significant routes using absolute correlation (only Sep to Dec): 66"
```

```
plot_route_change(df_sepdec, group_controlled_sepdec_pvals)
```



1.2 Common routes between Permutation and Regression Test

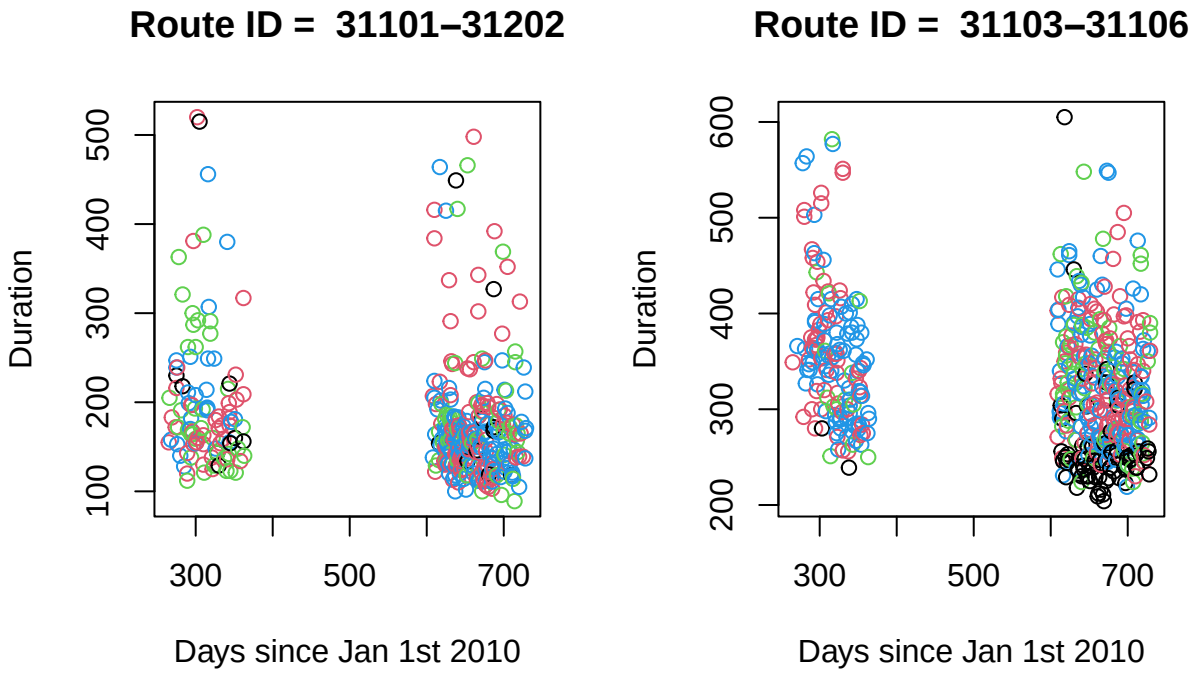
We believe the median/mean test statistics is the most relevant or comparable test statistics with the regression test as both treated 2010 and 2011 data separately and compared the two with each other. The median test statistics would be better than mean as it will not be affected by outliers. Therefore, we find the common routes identified by the permutation (median test statistics) with the regression test to see if the two methods give common results.

```
# Find common routes identified by permutation (median) and regression test
regression <- read.csv("single.csv")
lst <- intersect(significant_routes_sepdec_median$route, regression$noncoverage_rates_2010.route)
lst
```

```
##  [1] "31101-31202" "31103-31106" "31103-31214" "31104-31106" "31104-31110"
##  [6] "31105-31101" "31105-31203" "31106-31103" "31110-31104" "31110-31205"
## [11] "31110-31212" "31110-31214" "31200-31110" "31200-31111" "31200-31229"
## [16] "31201-31110" "31201-31202" "31201-31600" "31202-31200" "31203-31110"
## [21] "31205-31229" "31212-31110" "31218-31108" "31229-31111" "31229-31200"
## [26] "31602-31101" "31602-31107" "31602-31200" "31613-31606" "31613-31622"
## [31] "31623-31611"
```

Among the 76 routes identified as significant by the regression test, and the 59 routes found by the permutation median test, there are 31 routes in common. On one hand, this suggests that differences exist in the two approaches as the result does not match very well. On the other hand, by intersecting the common significant routes found by the two different approaches, these 31 routes' validity are improved as they survived both significance tests: they are more likely to have underwent some change throughout time compared to other routes that are only found to be significant by one test.

```
# plot the first two in the intersection between permutation (median) and regression test
significant_routes_sepdec_common <- significant_routes_sepdec_median %>%
  filter(significant_routes_sepdec_median$route %in% lst)

plot_route_change(df_sepdec, significant_routes_sepdec_common)
```



Above plots show the top two most significant routes identified by permutatation test (median) and regression test in common. Both tests used only data from September to December in 2010 and 2011, thus the above plots only show datapoints used in the tests (2011 January - August data taken out). With different color representing different group categories, we can vaguely see that group composition for each route might be different in the two halves. For example, for route 31103-31106, more black points appeared in the 2011 data (on the right), while there is much less black dots in the 2010 data. This change in group composition might affect the validity of our result: the change in duration might be due to change in group compositions.

For example, hypothetically, duration might shortened because there's more member riding than before and members ride faster in general. However, when doing the permutation test, we did not control for the groupings in each year (2010 and 2011 separately): instead, we shuffled all the data together. This is because there's insufficient data for most routes, especially in 2010. Therefore, there might not be enough data within the year for our permutation test that controls for four covariate (2 * 2 * 2 * 4 = 32 groups).

We then find common routes identified by permutation (median), permutation (absolute correlation), and regression test to identify routes that are shown to be significantly changed by all three test statistics.

```
# Find common routes identified by permutation (median) and regression test
intersect(intersect(significant_routes_sepdec_median$route,
                    significant_routes_sepdec_abscor$route),
          regression$noncoverage_rates_2010.route)
```

```
##  [1] "31101-31202" "31103-31106" "31103-31214" "31104-31106" "31104-31110"
##  [6] "31105-31101" "31105-31203" "31110-31104" "31110-31205" "31110-31214"
## [11] "31200-31110" "31200-31111" "31200-31229" "31201-31110" "31202-31200"
## [16] "31203-31110" "31205-31229" "31212-31110" "31218-31108" "31229-31111"
## [21] "31229-31200" "31602-31107" "31613-31606" "31613-31622"
```

Section II: Limitations

2.1 Limitations of the Regression Test

1) The true relationship between duration and the explanatory variables might not be linear. As such, linear regression might not be the best choice to produce prediction intervals.

2) We are assuming that the data distribution of the hold-out set and the test set is similar. For instance, there is no sudden change in the number of member riders in the test set since that might affect the coverage rate. We are also assuming there is no change in the composition of unobserved confounders between 2010 and 2011. Ideally, it would be the best to have the covariates of the hold-out set and test set from the same distribution.

3) We are assuming that there are no interaction terms between the covariates, which might not be true. However, due to limitations in the sample sizes, it is unfeasible by separating the data into very fine groups like what we did with the permutation data set, and regressing over the rest of the explanatory variables.

4) This regression prediction-based approach only aims to detect whether there's any change in duration between Sep-Dec 2010 and Sep-Dec 2011. It's possible that a change occurred between the two time periods (such as a road construction that started in Jan 2011 and finished in Aug 2011) and our linear regression approach will not be able to detect that. Whereas, on the other hand, permutation tests with the test statistics being the absolute correlation between time and duration should be able to catch these changes.

5) As we drop data from Jan 2011 to Aug 2011, a large portion of the original data, we inevitably lose some power in our detection procedure. Apart from the problem in 4), we are also losing on more sample size. However, this is necessary to get two relatively comparable data sets (e.g. we don't need to worry about the season acting as a confounder).

2.2 Limitations of the Permutation Test

Our permutation test is effective in identifying long-term changes in trend, and its validity is further improved by using three test statistics (correlation, mean, and median) to find significant routes in common.

Nevertheless, there are a few limitations due to the nature of the test statistics and the data itself that should be noted.

1) by plotting out the duration data for the significant routes that the permutation test identified, we noticed that there's an increase in the amount of data with time. This is probably because the record for Washington D.C.'s bikeshare program started at around 2010 September, thus there's more people participating in the program as time progresses. This might bias our test statistics due to the larger amount of data points towards the end (in time): with larger sample size, it is more likely that random fluctuations in the data might leads to spurious correlations.

2) though the permutation test considered membership, day of week, hour of day, and temperature, there are still many other relevant variables that are not considered but worth considering. For example, the age of the rider is likely to influence durations—we would normally expect a 80-year-old to ride slower compared to a 20-year-old. Also, we did not control for weather—we would expect people to take longer riding in rain compared to a sunny or cloudy day. These are not considered due to either limitation in the dataset itself or due to the limited time and resources we have.

Related to the second point, as shown above in the route plots, the group compositions (morning/night rides, weekday/weekend rides, member/non-member rides, rides in different temperatures) for each route might change throughout time. For example, it is conceivable that as more people get familiarized with the bikeshare program, they might be more likely to ride on a daily basis in weekdays instead of just "having fun" on weekends; they might also more likely to be enrolled and become a member. Though our permutation test shuffled data within groups, but we could not control for changes in group compositions. Therefore, our significant routes detected might be due to changes in group compositions: there are more members, and as members ride faster in general, duration is shortened (when we are computing for median or mean). It is worth noting that in the description of the assignment, we were asked "can you detect any routes where the average time it takes to travel the route, changes over the course of the time period?". Therefore, change in duration due to change in group composition is also a valid change over the course of the time period. We believe one possible reason for the change in duration of our detected routes might be due to change in group compositions.

3) our test statistics is not perfect in identifying some abrupt or sudden changes in the data: it captures long-term changes better. We did not find a perfect test statistics that could allow us to detect such sudden changes. Our best conjecture is to chop the data into smaller sections and compare the mean/median between each two consecutive sections. However, due to the limited data for each route, we might not be able to do so while controlling for all the other relevant variables that we should control for (membership, day of week, hour of day, and temperature). However, the absolute correlation approach does attempted and is suitable to answer the question of "detecting if there is any change at any point for each route".

4) we noticed that each "route" is specified by a starting station number and an ending station number. However, when looking up on google maps, we noticed that sometimes there exist multiple possible path between two stations. Different path takes slightly different amount of time, but they are all considered to be the same "route" due to the same starting and ending station number. This might bias our result—the correlation might arise due to fluctuations in frequencies people take different paths. Technically, it should not be quantified as one route if the paths are very different. We tried to improve this problem by removing outliers for each route to exclude some extreme cases (i.e., start from a station, ride around D.C. for 2hrs, and return to a station nearby the starting station), but the result might still be biased by paths that take not so different, but still different amount of times.