Introduction

This analysis uses my personal Strava data from all rides from start to Dec 2021. I use Tableau, Postgres, and R to conduct some basic analyses and to see whether it might be possible to implement a “guess my bike” feature on the Strava app, which could replace the default bike feature for riders who frequently use one of many bikes. This would make uploading rides faster, eliminating or reducing the need to edit the default bike.

SQL Analysis

```
#How many rides were completed by each bike? Sort in descending order of count
SELECT bike_id, COUNT(id) as "Number of Rides by Bike"
FROM strava
GROUP BY bike_id
ORDER BY "Number of Rides by Bike" DESC;
```

	bike_id bigint	Number of Rides by Bike bigint
1	3996174	1519
2	4226605	325
3	3996178	157
4	3996175	81
5	[null]	8

Most rides have taken place with bike 3996174 (Cannondale CAAD10), followed by 4226605 (Salsa Mukluk), 3996178 (Titanium Litespeed), and 3996175 (Cannondale Trail MTB). 8 rides have been done using unnamed bikes.

#Display min and max suffer score by bike

```
SELECT bike_id, MIN(suffer_score), MAX(suffer_score)
FROM strava
WHERE suffer_score>0
GROUP BY bike_id;
```

	bike_id bigint	min integer	max integer
1	3996178	2	980
2	3996174	2	26828
3	4226605	2	617
4	3996175	13	260

This output gives a range of suffer scores, which in itself doesn't reveal much. However, it shows that there could be an outlier for bike 3996174.

#Display the average suffer score by bike in descending order to see if there is a difference

```
SELECT bike_id, AVG(suffer_score) as "Average Suffer Score"
FROM strava
GROUP BY bike_id
HAVING AVG(suffer_score)>0
ORDER BY "Average Suffer Score" DESC;
```

	bike_id bigint	Average Suffer Score numeric
1	3996174	112.4217462932454695
2	3996178	103.8809523809523810
3	4226605	73.9502262443438914
4	3996175	60.5952380952380952

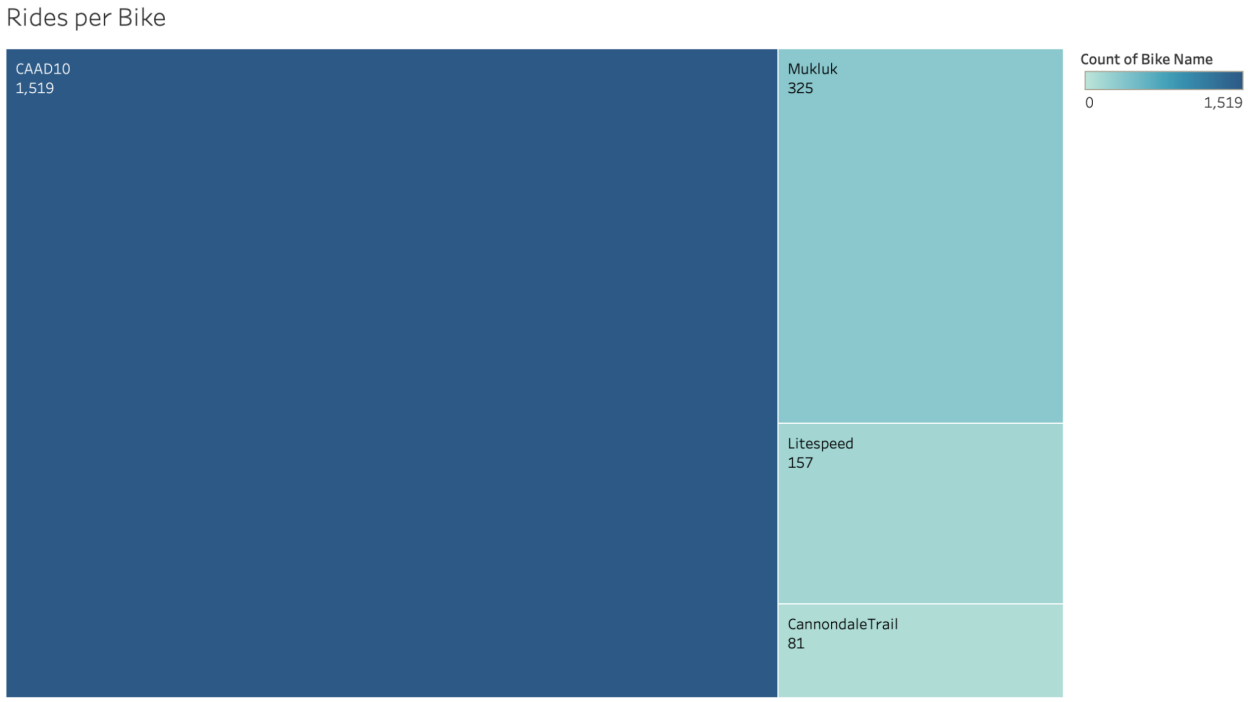
The average suffer score reveals somewhat more information. The CAAD10 has the highest average suffer score, followed by the Litespeed. These are both road bikes and compare to somewhat lower average suffer scores of 74 and 60.6 for my Mukluk and Trail bikes, respectively. This indicates that the road bikes generally have higher suffer scores and warrants further examination.

```
#Which day of the week is a ride most likely to take place?
SELECT start_day, COUNT(start_day) AS "Number of Rides by Day"
FROM strava
GROUP BY start_day
ORDER BY "Number of Rides by Day" DESC;
```

	start_day character varying	Number of Rides by Day bigint
1	Wed	318
2	Tue	311
3	Sun	300
4	Mon	294
5	Thu	293
6	Fri	291
7	Sat	283

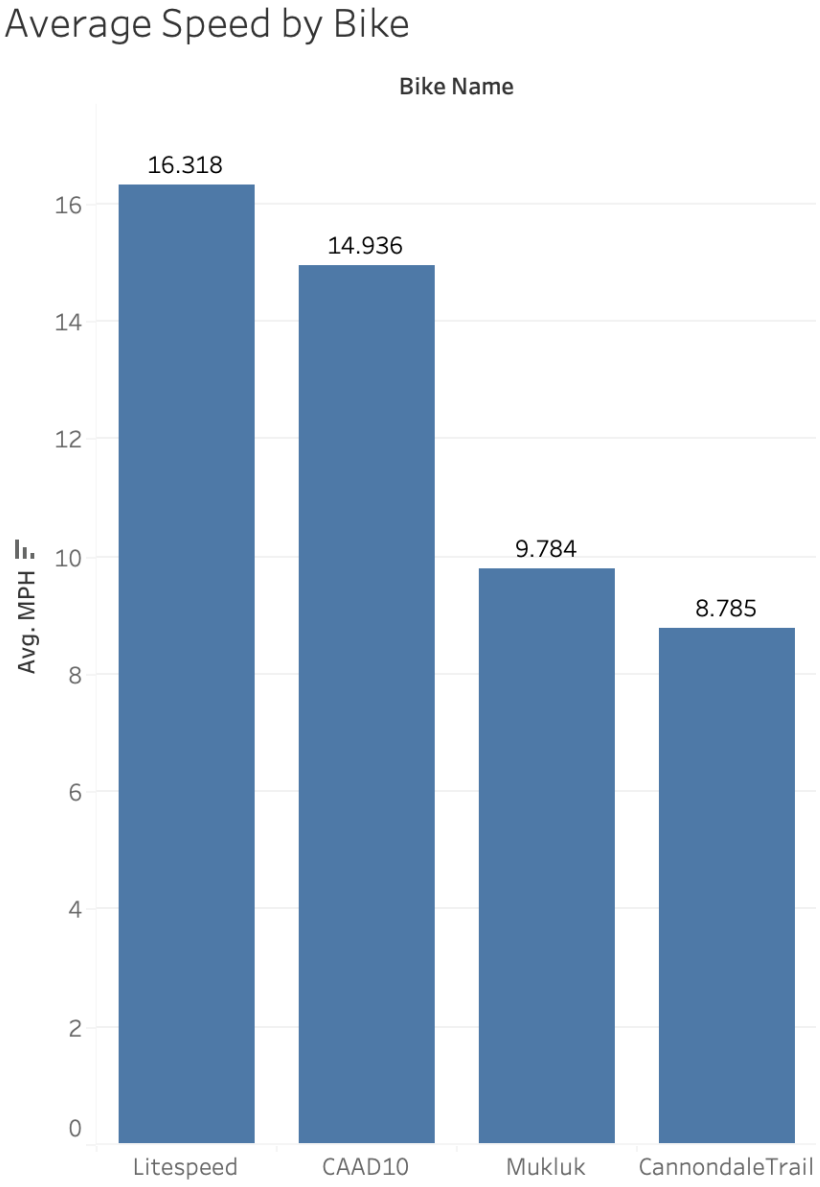
These rides are fairly evenly dispersed, but there are somewhat more rides that have taken place during the week versus the weekend. Interestingly, Saturday has seen the fewest rides.

Tableau



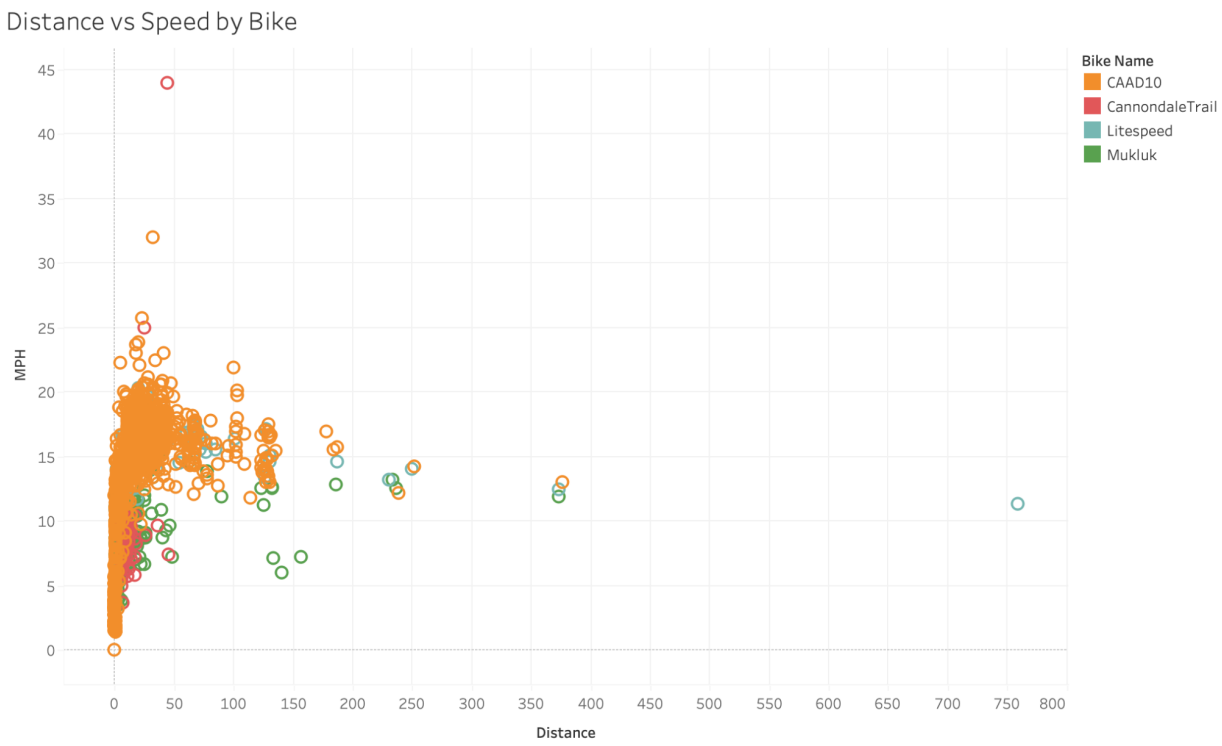
Bike Name and count of Bike Name. Color shows count of Bike Name. Size shows count of Bike Name. The marks are labeled by Bike Name and count of Bike Name.

The first SQL query viewed as a treemap in Tableau shows just how dominant the CAAD10 and Mukluk are in this dataset.



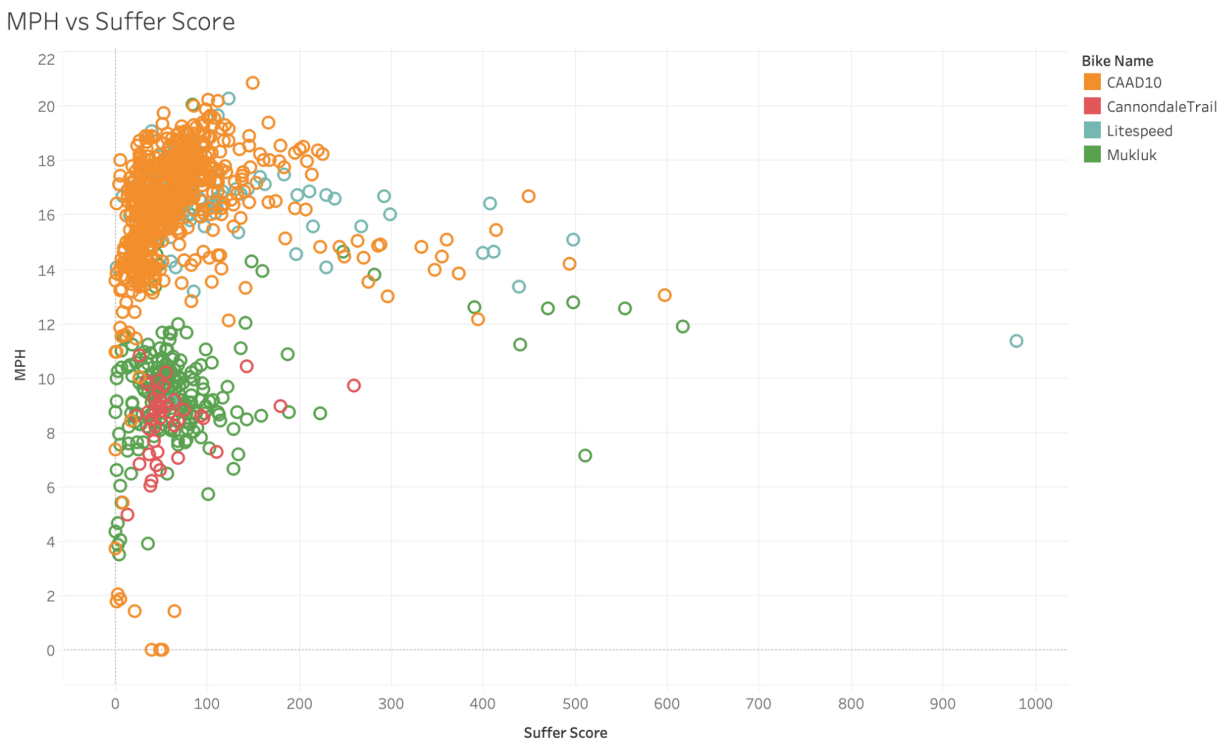
Average of MPH for each Bike Name. The marks are labeled by average of MPH. The view is filtered on Bike Name, which excludes Null.

To see if there is a possible relationship between bike and speed, the bikes and their average speed were plotted on a simple bar graph. These speeds are calculated values and represent the average moving speed of each bike. Surprisingly, the Litespeed comes out on top of the CAAD10 (surprising because the CAAD10 is a race bike, and the Litespeed is a randonneuring bike).



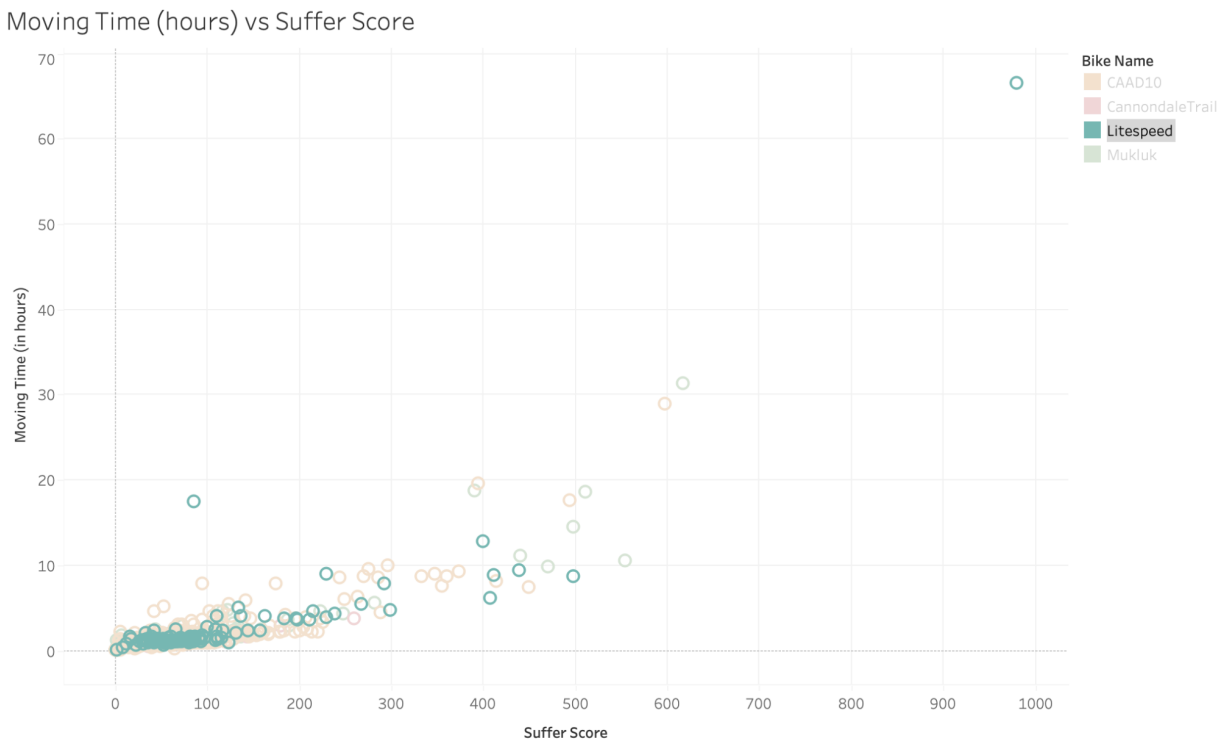
Distance vs. MPH. Color shows details about Bike Name. The view is filtered on Bike Name and MPH. The Bike Name filter excludes Null. The MPH filter keeps non-Null values only.

This chart reveals that there are a considerable amount of rides on the CAAD10 that are low distance/low speed. Not easily seen in this chart are that the Mukluk and Trail are grouped relatively similarly to each other, and the Litespeed data points are grouped more closely with the majority of CAAD10 values, which is to be expected given the previous bar chart.



Suffer Score vs. MPH. Color shows details about Bike Name. The view is filtered on MPH, Suffer Score and Exclusions (MPH,Suffer Score). The MPH filter keeps non-Null values only. The Suffer Score filter keeps non-Null values only. The Exclusions (MPH,Suffer Score) filter keeps 2,038 members.

Because the average suffer score calculated in SQL revealed some counterintuitive findings, it follows to investigate the relationship between suffer score and other variables. Here, the suffer score calculation is visualized, since MPH and bike are essentially synonymous.



Suffer Score vs. Moving Time (in hours). Color shows details about Bike Name. The view is filtered on Suffer Score and Exclusions (Moving Time (in hours),Suffer Score). The Suffer Score filter keeps non-Null values only. The Exclusions (Moving Time (in hours),Suffer Score) filter keeps 1,919 members.

There appears to be a modest relationship between moving time and suffer score, but the relationship is not exactly clear through a visual analysis (the Mukluk has many data points that seem to overlap the Litespeed, including some very high suffer scores).

Suffer Score vs. Distance. Color shows details about Bike Name. The data is filtered on Exclusions (Elevation Gain,Suffer Score), which keeps 1,371 members. The view is filtered on Suffer Score, which keeps non-Null values only.

Distance, though basically an analog of moving time, also appears to be a reliable indication of suffer score.

R Analysis

Data was imported to R and filtered down to numeric data and calculated variables include MPH and stopped time.

	term	bike_id	distance	moving_time_raw	elapsed_time_raw	stopped_time	elevation_gain	suffer_score	mph
1	bike_id	*NA*	-0.09278706	-0.01644068	-0.01645726	-0.002379562	0.04819794	-0.01673458	-0.65035695
2	distance	-0.092787059	*NA*	0.13109864	0.16863953	0.636512297	0.92997694	0.05759865	0.10304929
3	moving_time_raw	-0.016440682	0.13109864	*NA*	0.99818120	0.098060709	0.13441311	0.99542482	-0.12260463
4	elapsed_time_raw	-0.016457265	0.16863953	0.99818120	*NA*	0.157876973	0.17216409	0.98991574	-0.12450399
5	stopped_time	-0.002379562	0.63651230	0.09806071	0.15787697	*NA*	0.64040514	0.03655619	-0.04705829
6	elevation_gain	0.048197940	0.92997694	0.13441311	0.17216409	0.640405143	*NA*	0.06334248	-0.06413075
7	suffer_score	-0.016734580	0.05759865	0.99542482	0.98991574	0.036556186	0.06334248	*NA*	-0.11358288

8	mph	-0.650356946	0.10304929	-0.12260463	-0.12450399	-0.047058287	-0.06413075	-0.11358288	*NA*

Bike id is not strongly correlated with any particular numeric variable beside mph. In fact, suffer score seems less a function of bike ridden and more a function of moving time with an almost perfect correlation of 0.99. Suffer score can be ruled out as a valuable predictor of bike id. Despite this, it may be useful to run a regression on some of these variables and see if there is any predictive capacity.

For the purpose of this analysis, we will attempt to see whether we can predict if the default bike was ridden. Using this as the dependent variable and distance, moving time, stopped time, mph, feet of gain per mile, and suffer score as the independent variables, a multiple regression model was constructed. All variables, with the exception of suffer score are statistically significant with p-values under 0.05.

Mph and distance are the most important variables, both of which increase the probability of a bike other than the default when they increase in value. It can be concluded that with further analysis, it could be possible to construct a classification model to predict a user's bike based on ride data.

	term	**estimate**	**std.error**	**statistic**	**p.value**
1	(Intercept)	-0.4230196472	0.8104559750	-0.5219527	6.017033e-01
2	distance	0.2406606501	0.0439937423	5.4703382	4.491776e-08
3	moving_time_raw	-0.0009029357	0.0001569268	-5.7538670	8.722481e-09
4	stopped_time	0.0307443019	0.0078798758	3.9016226	9.555003e-05

5	mph	0.2056283727	0.0564088998	3.6453179	2.670615e-04
6	feet_per_mile	-0.0232615305	0.0067391202	-3.4517162	5.570333e-04
7	suffer_score	-0.0063224729	0.0037919052	-1.6673605	9.544274e-02

R Code

```
#Instructions for downloading Strava data
#https://scottpdawson.com/export-strava-workout-data/
#https://gist.github.com/scottpdawson/74f85f60a7cf7fcc8ee527592dadf498
#
#This analysis seeks to determine if it is possible to build a model that can
#predict whether the default bike was used for an activity, or if the app
#should prompt the user to select a bike. Such a feature would save the user time
#from having to manually edit the activity after it has been uploaded.


library(tidyverse)
library(readr)

ridedata=read_csv("/Users/jimmyaspras/Downloads/result.csv")
as_tibble(ridedata)

spec(ridedata)

#Compute mph
ridedata$mph=ridedata$distance/(ridedata$moving_time_raw/3600)
ridedata$mph<-as.numeric(ridedata$mph)

#Compute stopped time
ridedata$stopped_time=(ridedata$elapsed_time_raw-ridedata$moving_time_raw)/60

#Compute feet per mile, elevation gain
ridedata$feet_per_mile=ridedata$elevation_gain/ridedata$distance

#Run correlation to determine relationships
#Isolate numeric columns
ridedatanumeric<-ridedata[c("bike_id","distance","moving_time_raw","elapsed_time_raw",
                            "stopped_time","elevation_gain","suffer_score","mph")]
ridedatanumeric<-na.omit(ridedatanumeric)
library(corrr)
cormat<-correlate(ridedatanumeric)
cormat

#The most significant relationships are the most intuitive ones - suffer score/time
#distance/elevation gain, time/elevation gain.

#Can a model be built to predict the bike ridden based on some simple ride data?
```

```
#
#Code the dependent variable (bike_id) to binary categorical

ridedata$bike_id_binary<-ifelse(ridedata$bike_id=="4226605","Default","Other")
ridedata$bike_id_binary<-as.factor(ridedata$bike_id_binary)

#Binary model
library(tidymodels)
bikemod = logistic_reg() %>%
  set_engine("glm") %>%
  fit(bike_id_binary ~ distance + moving_time_raw +
        stopped_time + mph + feet_per_mile +
        suffer_score, data = ridedata)
bikemod
library(broom)
bikemod2<-tidy(bikemod)
bikemod2
glance(bikemod)
```