Maxwell Rodgers

April 30th, 2024

Did I Miss the Bus or Did it Miss Me?

(UTA Punctuality Analysis)

UTA is Utah's largest public metropolitan transportation service. Affordable for citizens, it provides transportation throughout the Wasatch Front through various bus lines, trolley lines, and trains. Everyday, Utah commuters rely on UTA's services to commute across the state, to and from work, and to go where they please. Transportation is arguably the most important public service that can be offered in a large metropolitan area, so naturally people rely on it for their day to day activities. Since people rely so heavily upon it, it needs to be punctual. Most people cannot change being late from work, missing an exam at university, or possibly not going anywhere at all in the event of a missing bus.

With that in mind, I seek to answer this question: **Can we predict the deviation of UTA's actual bus stop times compared to their scheduled stop times based on time of day?**

This question is important because it will help commuters ensure they plan accordingly so they don't miss their busses and in turn miss important events. Additionally, this information can help improve public transportation by applying for more government funding to improve transportation infrastructure. UTA will also be able to see exactly where their variance has the highest deviations to see exactly where they are failing their commuters. UTA reducing variance in their stop times will help build trust in their service. When people can rely on transportation,

ridership will increase and thus lead to less cars on the road. Getting actual stop arrival time variance down will benefit everyone, even those who choose to drive.

My hypothesis is that UTA bus times vary greatly, and to ensure you catch your bus it is not enough to arrive at the stop at the scheduled time. You must arrive early and stay after your scheduled time, else you chance missing the bus you intended to catch.

One potential solution to this problem could be that, if in my analysis, it is shown such that actual stop arrival times have a peak variance UTA could introduce a dynamic route schedule wherein more buses are introduced to a route at specific times to reduce the variance at that time of day.

Data Collection

To see all the moving parts please view my GitHub repository here: https://github.com/maxdotr/UTA-Punctuality-Analysis

Because the data we need concerns when UTA buses actually pass by stops, and UTA does not record this, we need the locations of buses throughout the day. UTA does not store their buses in a real-time location. Although they do have an API, found at https://webapi.rideuta.com/ with documentation at https://developer.rideuta.com/DataInstructions.aspx, that provides real time coordinates of their busses, to track this for long periods of time costs a lot of memory and maintenance. Due to this issue, I will have to personally store the locations of buses after receiving data from UTA's tracking API. I did this over a twenty day period, April 11th, 2024 - April 22nd, 2024. The following script requests the location data for every bus on the routes specified once a minute from 6:00AM to 9:00PM. To host this script, I used PythonAnywhere.

PythonAnywhere is a remote script hosting service which offers some really neat tools including "Always On" tasks, in which in the case of a crash, the task is immediately restarted. This service is why my script was able to consistently collect data over this 20 day period, ensuring the integrity of the data.

Script: https://github.com/maxdotr/UTA-Punctuality-Analysis/blob/main/getStopTimes.py

Additionally, I used Python's scheduling library to run the script on a timer so as to not overwhelm UTA's API.

It is worth noting that this API is not a streaming API despite whatever the documentation says. In reality, at a certain time, UTA updated their API services such that the documentation doesn't mean much. Their developer tools are so outdated, they are hardly any use. In reality, the API is now a REST API. Once you request information from the servers, the busses send their location. To refresh this data, you must ping the API again.

After running this script for 20 days, I had over 6 million points of data. This data was raw location data taken every minute and told me nothing about when the bus actually arrived at a stop. To figure that out, I had to develop an algorithm that would match the closest location point, in a scheduled interval, on a scheduled route, on a specific day. Mind boggling I know, so I will now provide a brief breakdown of what the algorithm does step by step:

- First we fetch static location data from UTA's public GTFS data set. This is the data set that hosts static information like routes, stops, scheduled times, etc, that UTA gives to transit providers like Google and Apple maps. We need this data so we can assemble a dataset that describes the movement of busses through the day and to get the scheduled

times we will be matching later. This data can be found here https://transitfeeds.com/p/utah-transportation-authority/59

- Clean the data to be in usable formats, like making scheduled stop times as valid date time objects to be used in comparisons.
- For each bus route, we must then merge relevant datasets to consolidate route, stop, and trip information.
- For each scheduled stop time in the dataset we just created, we then need to calculate the interval in which to filter our actual location data to check for distances in this interval. (Ex: Scheduled time is for 9:00 AM at stop X for route 9 going towards Poplar Grove on April 12th, we should only check our actual locations if they are for route 9 going towards Poplar Grove and were recorded between 8:45 AM and and 9:15 AM on April 12th)
- For each filtered bus location point in that interval, we must calculate the distance from that bus to the stop we are looking at. The closest distance is +\-30 seconds to the actual time the bus passed/arrived at the stop. (Busses don't always stop if no one is there)
- Calculate the time difference based on the match we just found
- Add this to a dataframe for logging purposes.

This algorithm is not optimized for efficiency, but it IS very accurate. It is so time inefficient, logging operations substantially increased the runtime to the point where I had to develop an asynchronous multithreaded logging solution just so it would run on a small dataset in a timely manner for testing. (Seen on my GitHub) To process the entire 6 million points, it took about 18 hours and used 8GB of RAM.

Matching algorithm:

https://github.com/maxdotr/UTA-Punctuality-Analysis/blob/main/getActualArrivalTimes.py

Logging solution is described here, written by Plumber Jack:

http://plumberjack.blogspot.com/2010/09/improved-queuehandler-queuelistener.html

Data Limitations

Since I collected all the data myself, there were some limitations so as to not overwhelm UTA's API, my computer resources, my storage limitations, and the integrity of my scripts. They are as follows:

- Data was only collected for UTA bus routes that pass through either downtown Salt Lake City or the University of Utah. These routes are routes 1, 17, 2, 200, 205, 209, 21, 213, 220, 223, 4, 455, 470, 509, and 9.
- Data was only collected between 6:00 AM and 9:00 PM, even though some routes go later into the night and some start earlier.
- Data was only collected over a 20 day period, April 11th, 2024 - April 22nd, 2024.
- Some route's bus trackers sometimes go offline, which results in inaccurate location data. (Cleaned later)
- Bus routes were taken every minute so actual arrival time I calculated could be +/- 30 seconds. This would happen if the bus passed a stop between the minute interval.

Data Cleaning

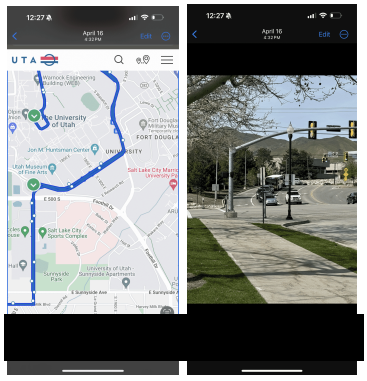
My algorithm is very accurate, but some of the merge operations with UTA's static data caused duplicates. These duplicates in the final dataset I made are caused by duplicate scheduled stop times in the dataset the algorithm was using to create it. They can be removed.

Some of the distances calculated are off because UTA's tracking API isn't always working correctly. If the algorithm matched a stop and the distance to the stop was greater than 100 meters, this is because a bus on that route or the route itself wasn't updating the API correctly. This is a known issue with the API so removing these entries can be done without worry.

Validating Accuracy with Real World Experience

On 04/16 I took these photos with the purpose of proving the accuracy of my algorithm and data collection methods. As you can see, I took the photo at the bus's location at 4:32, screenshotted the tracker information at 4:33, and boarded at 4:35.

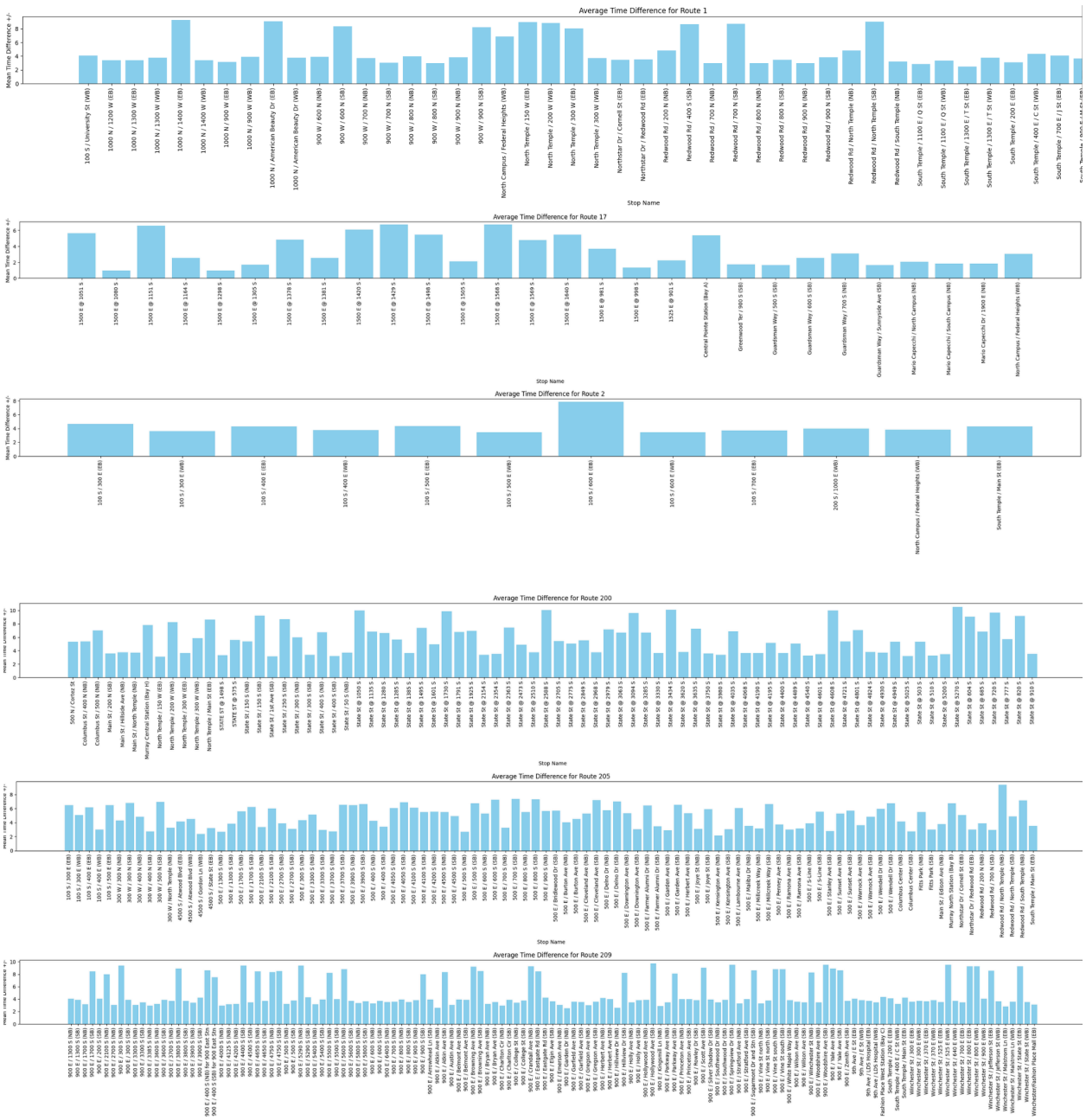
I then searched my match times for this bus, and low and behold the info matches!

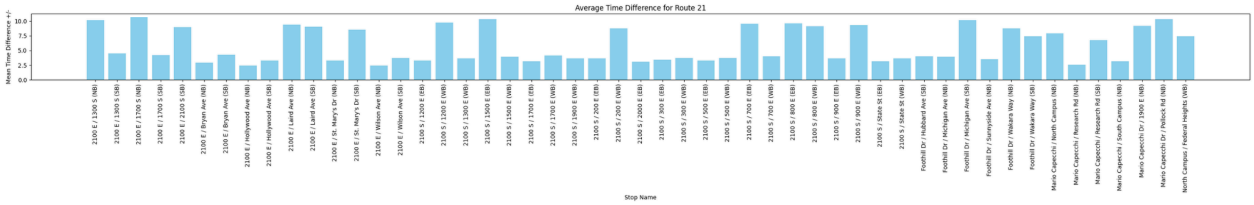


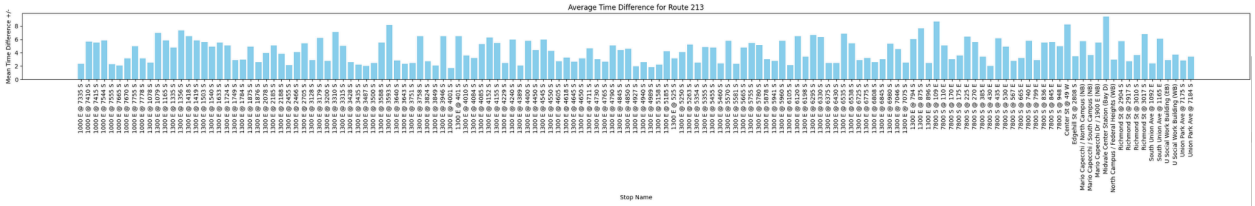
	vehicle_id	stop_id	stop_name	route	destination	scheduled_arrival_time	actual_arrival_time	distance_to_stop	match_date	time_difference
834415	23022	17727	Guardsman Way / Sunnyside Ave (SB)	9	Poplar Grove (Orange St)	16:34:10	16:35:21	45.447899	2024-04-16	1.183333

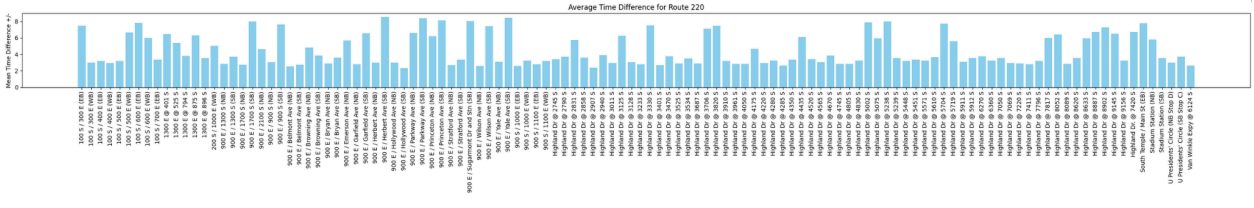
Early Visualizations

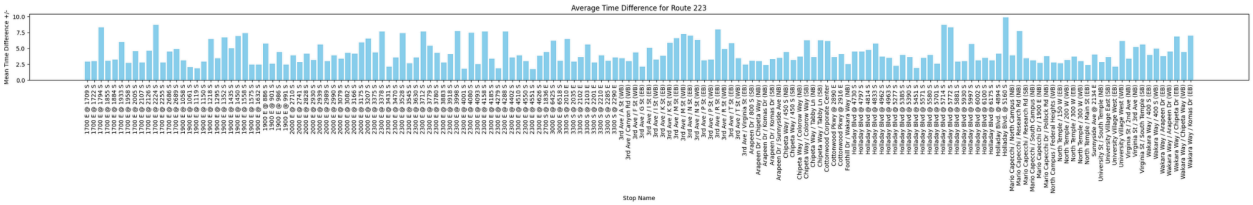
Now that we know our data is correct, I want to get a quick idea where my hypothesis is at. To do this, I will create a bar chart for the average for variance for each stop on each route I tracked. Bar charts are great for comparing categorical dependent variables to independent dynamic variables. Looking at the data, it does seem there are trends in variance!

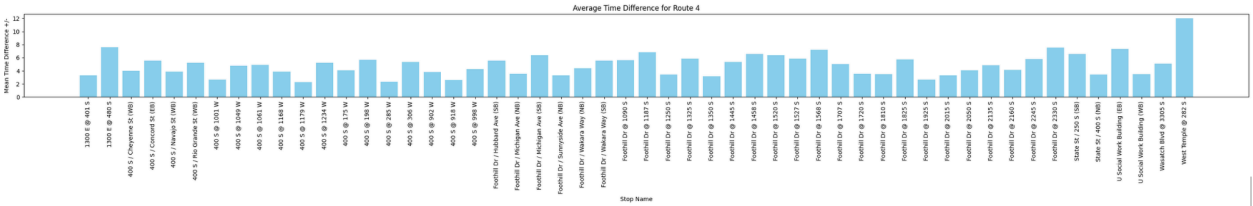


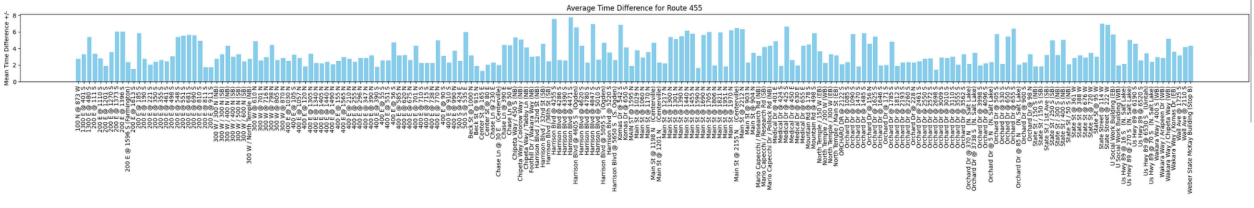

Average Time Difference for Route 21
Mean Time Difference +/-
Stop Name

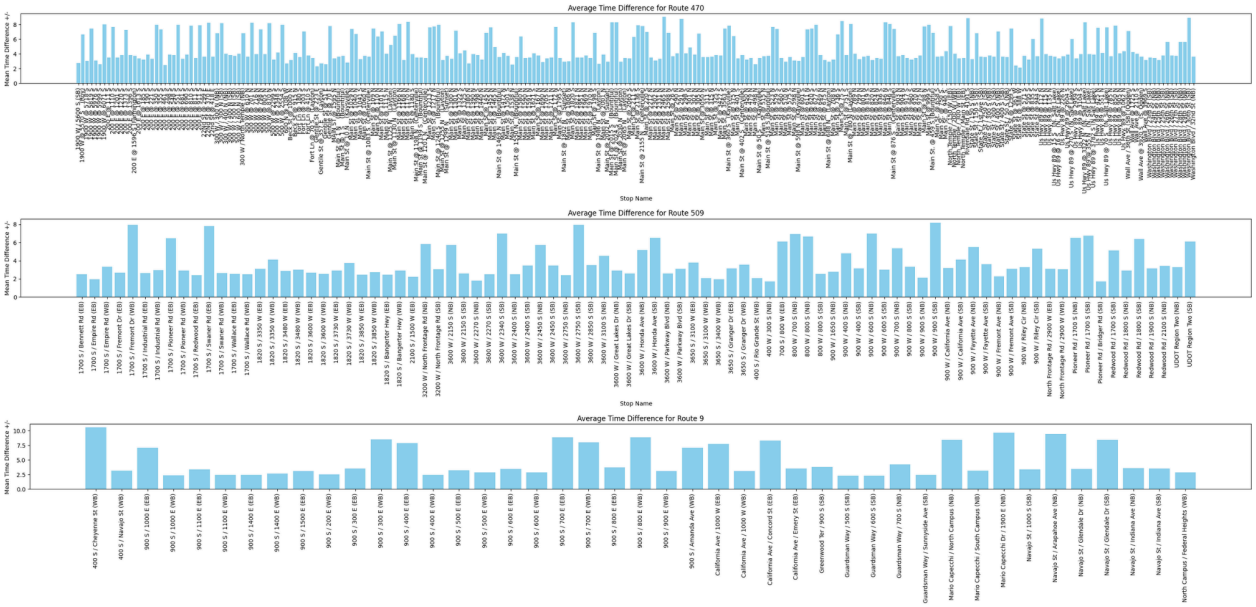

Average Time Difference for Route 213
Mean Time Difference +/-
Stop Name


Average Time Difference for Route 220
Mean Time Difference +/-
Stop Name

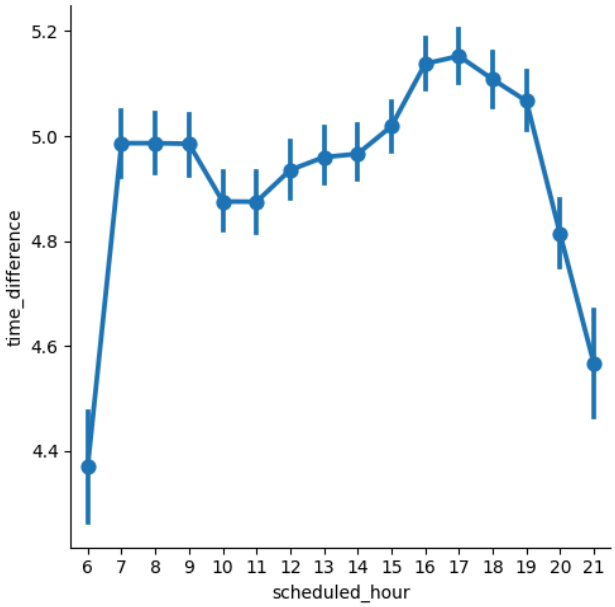

Average Time Difference for Route 223
Mean Time Difference +/-
Stop Name


Average Time Difference for Route 4
Mean Time Difference +/-
Stop Name


Average Time Difference for Route 455
Mean Time Difference +/-



Additionally, I want to create a catplot to see the variance for our entire data set at each hour of the day. Cat Plots are great because they match the curve of the graph and also inform the reader of data points that fall outside of the curve. The trending variance at different times of day seems very strong. This catplot also matches what one would expect, with lower variance in the morning and evening when there are less riders and peaks during the daytime when people go to work.



Analysis

Above, we can see that our data displays patterns, but it is not linear. This means that normal linear regression models or multivariate linear regression models will not accurately represent the effect of time of day on the variance of the actual arrival times. To get a better idea we will use a spline linear regression model. Spline linear regression models split our data into segments, called splines, where we can then perform linear regression analyses on the data. After we perform our analyses, we will be able to look at the effect each spline has on the model as a whole!

Generalized Linear Model Regression Results

Dep. Variable:	time_difference	No. Observations:	298363
Model:	GLM	Df Residuals:	298356
Model Family:	Gaussian	Df Model:	6
Link Function:	Identity	Scale:	15.982
Method:	IRLS	Log-Likelihood:	-8.3680e+05
Date:	Tue, 30 Apr 2024	Deviance:	4.7683e+06
Time:	22:18:05	Pearson chi2:	4.77e+06
No. Iterations:	3	Pseudo R-squ. (CS):	0.0009383
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	4.1166	0.117	35.317	0.000	3.888	4.345
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[0]	0.9061	0.158	5.722	0.000	0.596	1.216
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[1]	0.8763	0.109	8.012	0.000	0.662	1.091
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[2]	0.6763	0.132	5.117	0.000	0.417	0.935
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[3]	1.1014	0.120	9.208	0.000	0.867	1.336
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[4]	1.0540	0.134	7.865	0.000	0.791	1.317
bs(minutes_since_midnight, knots=(480, 720, 960), degree=3, include_intercept=False)[5]	0.3103	0.131	2.369	0.018	0.054	0.567

After creating our model, we can see some really valuable information. The pearson chi squared test gave us a value extremely close to zero, 0.0000047. The chi squared test is a test that determines the probability of the independent variable, hour of the day, not having an effect on the variance of the time. With such low probability, we can accept our hypothesis, which is that **we <u>can</u> predict the variance of a bus based on the time of day** and reject the null hypothesis that the time of day doesn't have an effect on the variance of actual arrival times.

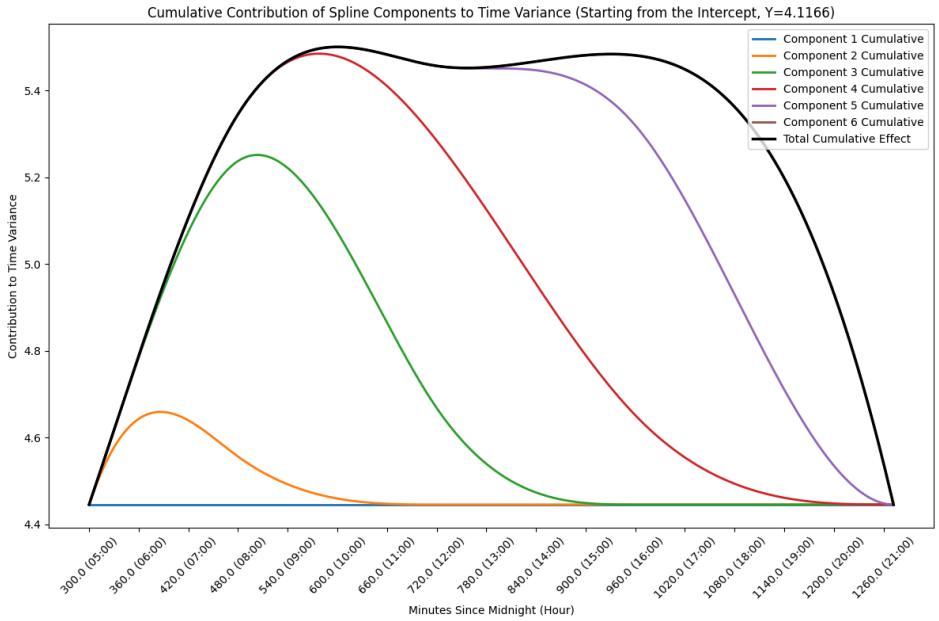
Additionally, this test allows us to see each spline's analyses as well. For each spline, we got p values all lower than 0.05. This means that for each spline component we can accept our hypothesis and reject the null hypothesis. We also are able to see the coefficient of each spline, which is how much the spline component affects the prediction on time difference. The z value for each spline is high, each higher than 2, which indicates the coefficient is significantly different from zero at conventional levels. The confidence interval shows us a range of coefficients for each spline where we can be 95% sure contains the true effect size.

Limitations of Analysis

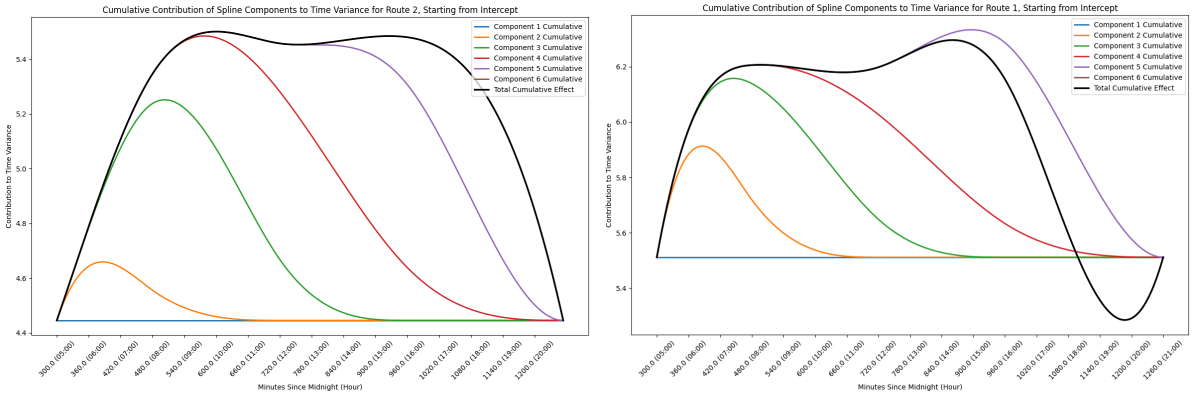
The biggest issue with my analysis was that I only had the processing power to track my data over 20 days. This is an issue because, in Utah, the winter time drastically changes road conditions which could have an effect on actual arrival time variance. I tracked my data with late spring to early summer road conditions. A more in depth analysis would track bus locations over a year. To do this however, would take incredible amounts of processing power even if the algorithm was refactored to be multithreaded. Additionally, it may split the data into a couple of categories such as “Winter” or “University of Utah Academic Semester”, where these categories have a chance to affect the time variance of actual arrival times.

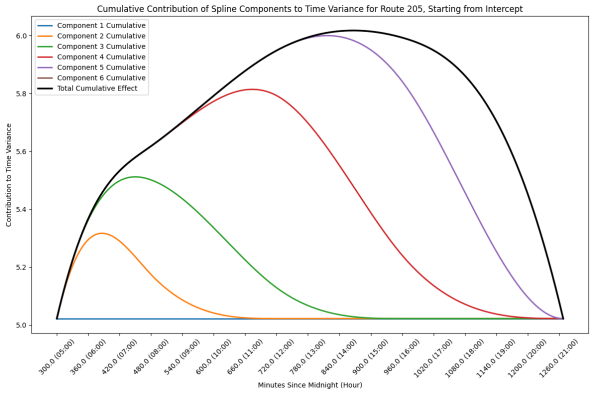
Graphing Our Results

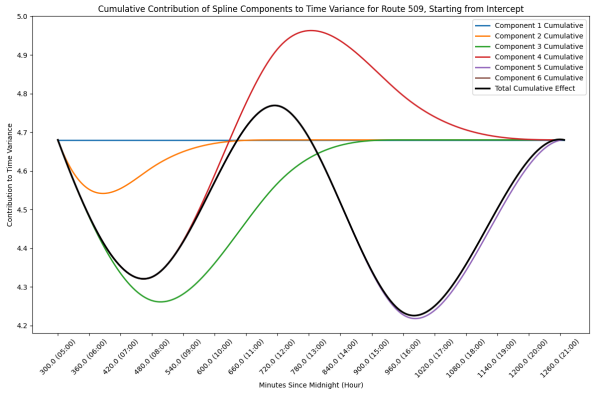
Now that we have accepted our hypothesis, we can graph the results of the spline linear regression model. To visualize the model, we will graph each spline component's effect on the variance of the actual arrival time of buses through the period in which I collected data, 6AM-9PM. We will then add another line which shows the cumulative effect of each spline component on the variance of the time.

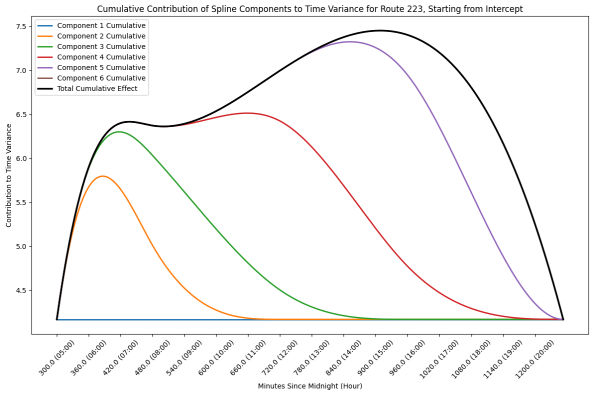


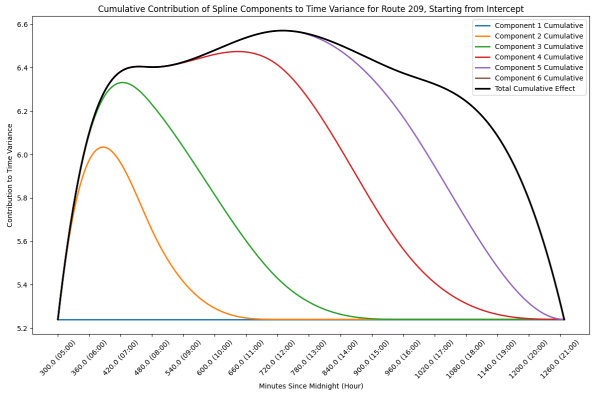
This graph shows the effect of each spline component on the entire model. Since this graph is a model for all our data, in other words each tracked route, it is a very generalized model that will be accurate for most buses at most times. However, we can filter our data by route and then perform the same graphing analyses to get even more specific variance data.

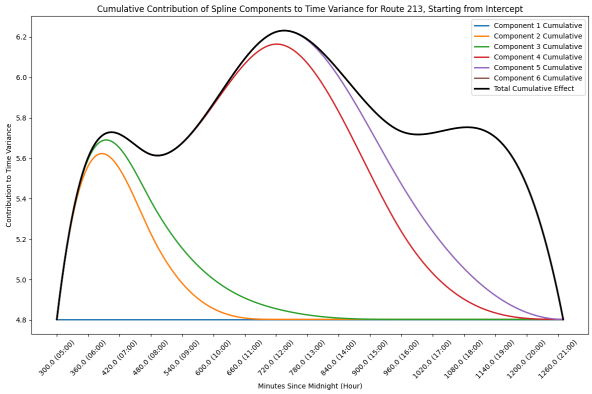


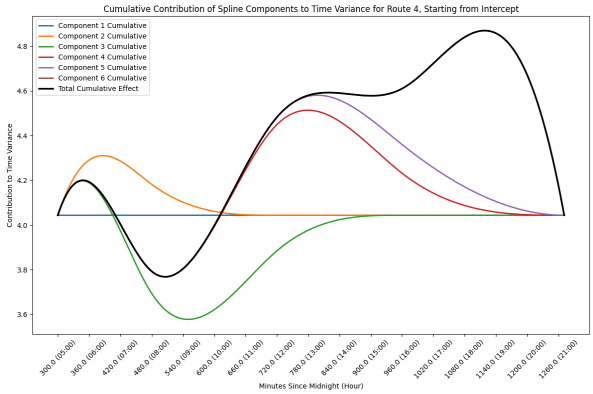

Cumulative Contribution of Spline Components to Time Variance for Route 205, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

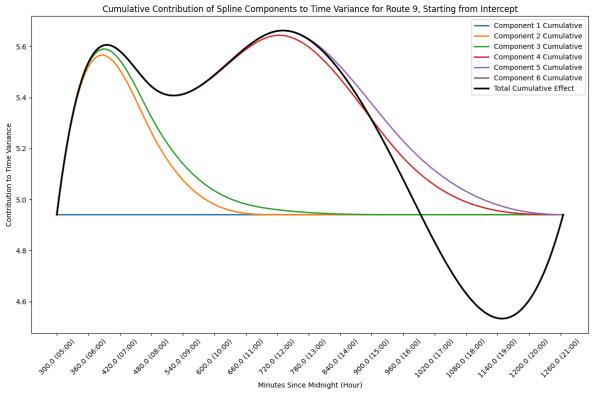

Cumulative Contribution of Spline Components to Time Variance for Route 509, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

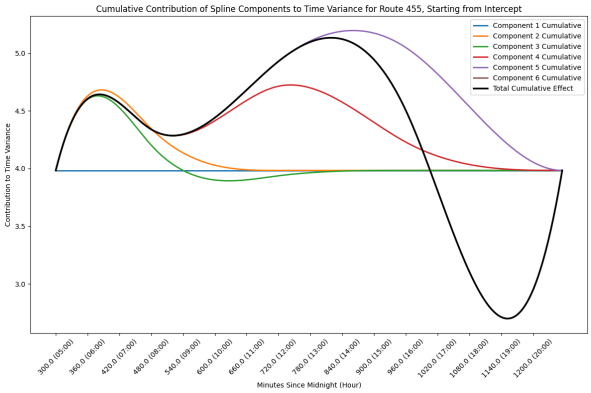

Cumulative Contribution of Spline Components to Time Variance for Route 223, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

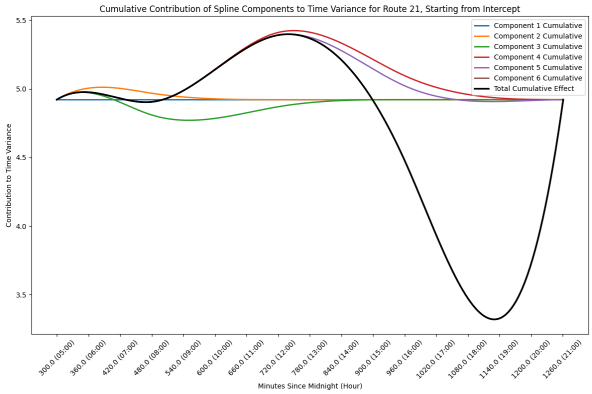

Cumulative Contribution of Spline Components to Time Variance for Route 209, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

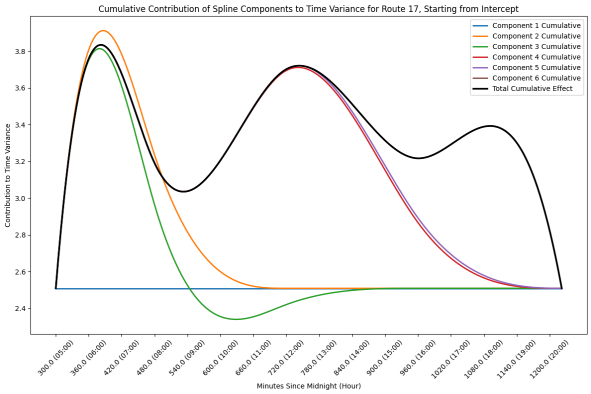

Cumulative Contribution of Spline Components to Time Variance for Route 213, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

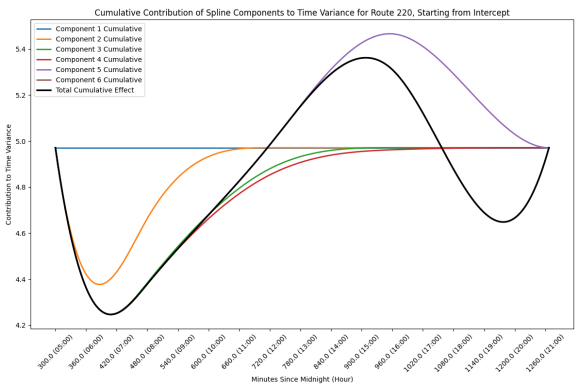

Cumulative Contribution of Spline Components to Time Variance for Route 4, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

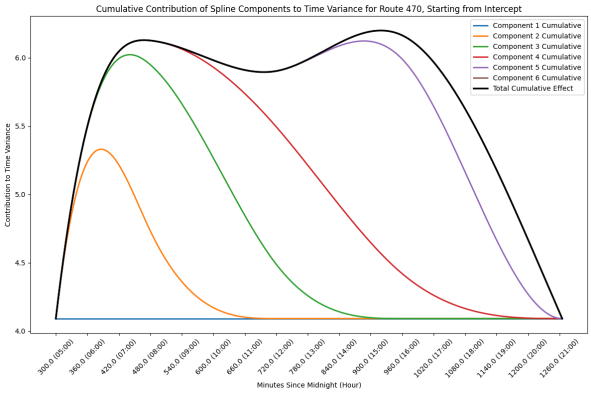

Cumulative Contribution of Spline Components to Time Variance for Route 9, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)

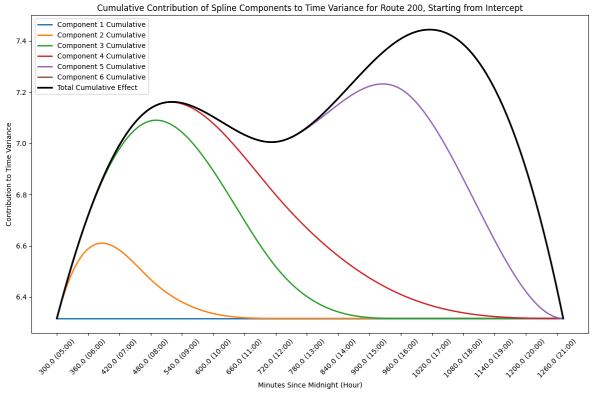

Cumulative Contribution of Spline Components to Time Variance for Route 455, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)


Cumulative Contribution of Spline Components to Time Variance for Route 21, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)


Cumulative Contribution of Spline Components to Time Variance for Route 17, Starting from Intercept
Component 1 Cumulative
Component 2 Cumulative
Component 3 Cumulative
Component 4 Cumulative
Component 5 Cumulative
Component 6 Cumulative
Total Cumulative Effect
Contribution to Time Variance
Minutes Since Midnight (Hour)







How to Use These Graphs

These graphs predict the variance of buses for each route tracked at most times of day. To use this data, as a commuter, to plan my trips I will choose the graph for the route I want, go to the X axis at the time my stop is schedules, and find the corresponding Y value on the black line to know when I should go to my stop, and how long I should plan to stay after. As an example, if I was boarding a bus on route 9 at 12AM, I should plan to arrive 5.6 minutes early and stay 5.6 minutes late to ensure I caught my bus.

To Be Sure... (Potential counterarguments)

"Tracking isn't accurate to real times" - Through personal experience I have proven that UTA's bus tracking API is reasonably accurate for real time location data. However, in the case that it is

off by a couple minutes this would not affect the results of our analysis as results are graphed on a curve over hours so shifting the graph nominally wouldn't do anything significant.

"The bus tracker for route X went down during tracking hours for X amount of time on X day" - Location data was taken over a period of 20 days in which time tracking data never went down for more than a couple hours. This means that each bus on each route was tracked for 240 hours, so a nominal decrease in tracked hours would not statistically significantly affect our data.

"Matching algorithm to get the actual arrival time is unverifiably accurate" - I can assure the algorithm is very accurate. To test it, in the GitHub for this analysis, I included a small sample of data so it can be ran without extensive run times. Additionally, if formatted correctly, you can alter the script to work with whatever test data you please. Additionally, the script has logging statements for ease of analysis.

"The matching algorithm matched a bus with a time saying it was early when it was late." - Lateness or earliness is a matter of perspective for the rider. This analysis was done using absolute variance for this reason. This means that a bus actual arrival time was always correctly matched, just to the scheduled time in the interval examined it was closest to.

Conclusion

In this analysis we have proven that UTA buses are reliably not punctual. We have also proven that the difference in which their actual arrival times vary from their scheduled times can be predicted.

One potential solution to reduce variance could be dynamic route planning. For example, route 200 has increasing variance from 1PM to 6PM. During this time UTA could add buses to the route to reduce this expected variance. UTA can also apply for more funding from the government showing that their current funds are not enough to provide reliable schedules for the amount of riders they serve. Also, UTA can now see what routes have the worst variance and change strategies to take specific action to rectify these routes.

Commuters that ride UTA buses can now also plan their schedules accounting for variance in actual arrival times. Although in an ideal world scheduled times would be accurate, while we strive for this ideal, commuters can use this analysis to better reliably get to where they need to be.

Public transportation is incredibly important. It reduces C02 emissions, is affordable, and can usually be relied upon. More people riding buses benefits everyone. However, the first step to increase ridership is to increase trust in the service. Riders cannot rely on scheduling where there is such variance. UTA would greatly benefit from decreasing this variance.