Modeling the Effect of the RapidRide Program on Bus Reliability in Seattle, WA, Using Regression and Bayesian Additive Regression Trees

Peter Silverstein

Submitted in partial fulfillment of the requirements
for the degree of Master of Arts under the Executive
Committee of the Graduate School of Arts and
Sciences

COLUMBIA UNIVERSITY

Advisor: Dr. Edwin Grimsley

2025

© 2025

Peter Silverstein

All Rights Reserved

Contents

0. Abstract

Mass transit systems are a crucial piece in the fabric of any urban environment. They allow people who do not own cars to access the far reaches of a city and provide a more sustainable alternative to single occupancy vehicle traffic. In large part, the success of a transit system is dependent on its ability to get people where they want to go and do so with reliability. The present research examines a transit intervention, the RapidRide bus route upgrade program in Seattle, Washington, and seeks to determine whether it has a positive effect on bus reliability. Linear regression models and a Bayesian Additive Regression Trees model are applied in a causal inference setting to estimate the average treatment effect of the RapidRide treatment on the bus routes that have been upgraded. All models find that the RapidRide treatment improves reliability. Median estimates for this treatment effect vary by model, and range from a 15 second improvement to a nearly 60 second improvement as a result of the intervention. Further, different models provide different uncertainty associated with this effect, indicating that the variability of the treatment effect is also an interesting point of study. Because reliability is an attribute that is highly valued by mass transit riders and because better reliability is associated with a greater ability to make transfers and travel efficiently, these models point to the RapidRide upgrade program being a success.

1. Introduction

Public transit systems shape urban environments by providing residents with affordable, sustainable, and efficient mobility options. A key factor determining the effectiveness of transit systems is accessibility—the ease with which riders can reach desired destinations within a city. Accessibility is often operationalized through the concept of the Space-Time Prism (STP), which measures the range of feasible destinations that individuals can reach in a specific timeframe given constraints such as travel speed, wait times, and transfers. Within this framework, transit reliability—the consistency and predictability of service schedules—is critically important. Unreliable service, characterized by late or early arrivals, negatively impacts riders' ability to successfully transfer

between routes, thereby reducing overall accessibility.

To address reliability issues without incurring the substantial costs and timelines associated with major infrastructure projects like subway or light rail systems, many cities have turned to Bus Rapid Transit (BRT). BRT encompasses operational strategies such as bus-only lanes, off-board fare collection, traffic signal priority, and active headway management to enhance bus service speed and reliability. Seattle's RapidRide program exemplifies a "BRT-lite" approach, implementing only selected BRT features on high-demand bus routes to improve service quality at relatively lower costs.

Despite ongoing investments in RapidRide upgrades across Seattle's transit network, empirical evidence regarding their effectiveness remains limited. This paper seeks to fill this gap by evaluating whether routes upgraded under the RapidRide program exhibit improved reliability compared to comparable routes without such upgrades. Using King County Metro real-time arrival data and Bayesian Additive Regression Trees (BART), this study estimates the causal effect of RapidRide upgrades on schedule reliability. By rigorously accounting for multilevel data structures and potential confounding variables such as traffic conditions and ridership demand, the analysis aims to provide robust insights into the effectiveness of targeted transit interventions in enhancing urban accessibility through improved schedule adherence.

2. Literature Review

2.1 Accessibility and Transit Reliability

Cities are complex spatial relationships between people, places, and services. To a large degree, where an individual is within a city may determine their ability to access various other people, places, and services within that city. The related concepts of access and accessibility (Hansen, 1959) are central to the field of urban planning and represent the extent to which a location/service/opportunity is available to an individual that exists at a given location. The central goal within designing an urban environment with access in mind is to maximize an individual's ability to get where they want or need to go (Levinson & Wu, 2020). This may be done in different

ways—land use planning would seek to place relevant zones (e.g., retail, industrial, residential, etc.) in useful places, and it may be a further goal of an urban planner to ensure equity in access between individuals of different social backgrounds (Hansen, 1959). However, it is the case that cities are not set up equitably, with equal and consistent opportunities available to all people that live there (Gregory, 2024). Because of this, it is necessary to improve accessibility through mobility—using transportation solutions to allow people to easily navigate through the city, enable interactions between people from different areas, and ensure that opportunities for employment and recreation are available to everyone, regardless of race, socioeconomic status, or some other categorization (Antipova et al., 2020).

Within the framework of urban transportation, mass transit systems provide key benefits that make them important and relevant subjects of study and evaluation. Mass transit systems allow people to more easily traverse an urban environment without the need for a personal motor vehicle. This has important socioeconomic ramifications—personal vehicle ownership typically entails higher costs than public transit and the use of mass transit by economically disadvantaged people has been empirically shown (Anderson & Galaskiewicz, 2021). Additionally, mass transit is a more environmentally sustainable transportation solution than personal vehicles, and increased adoption is associated with lower traffic congestion (Dutta, 2013). As such, a well-functioning transit system is crucial to the fabric of an urban environment, and maximizing accessibility (i.e., the extent to which a rider can travel from a starting point to any given end point) is an important aspect of the transit system.

When operationalizing accessibility in a mass transit setting, the conceptual basis is to consider an individual rider at a starting point. From this starting point, there exists a set of possible destinations that the rider can go to, but only a subset of these possible destinations is considered to be accessible to the rider. The subset may be defined in a variety of ways, but the most common definition of transit accessibility is the STP. The STP is a time geography concept that models the extent to which someone can travel, considering a limited time window and various other limiting factors such as available paths and maximum speed (Hägerstrand, 1970). Analyzing accessibility

through the STP involves considering route speed, wait times at stops, number of transfers, etc. (Liu et al., 2023). Comprehensive analyses of the mass transit system accessibility would also account for the desirability of various destinations in the network (i.e., can riders get to where they want or need to go, not just how far can they travel), but this is outside the scope of the present research (Handy, 2020). The analysis in this paper assumes that the transit network is designed with destination desirability in mind (e.g., routes tend to converge in urban centers), and that the goal is simply to improve the accessibility of the network.

An important aspect of transit accessibility is the extent to which pieces of the system interact well with each other. Transfers (moving from one bus route to another) are necessary parts of transit systems. The ease and consistency with which a rider can successfully perform these transfers is essential to STP-based accessibility. That is, needing to wait 15 minutes at a stop during a transfer (compared with 5 minutes) has a detrimental impact on the size of the STP. Schedule-based accessibility uses publicly-available schedules to construct an STP that accounts for bus arrival times and possible wait times between transfers, though this approach is somewhat rudimentary as it does not account for real-time variation in bus arrival times Wessel et al. (2017). Liu et al. (2023) point out that accounting for this real-time variation would produce more accurate estimates of accessibility in their proposal for an STP calculated using "realizable real-time accessibility." They conclude that accounting for real-time uncertainty in their calculations tends to decrease the scope of the STP—that is, uncertainty and deviation from schedule worsen accessibility.

It is intuitive to think about how small changes in reliability can propagate through a trip, causing a rider to arrive at their destination substantially later than planned. The rider may plan on using two routes connected by a transfer. If the gap between the first bus's arrival at the transfer stop and the second bus's departure is small (say, less than 5 minutes), then unreliability from the first bus (e.g., arriving 5 minutes late) can cause the rider to miss the transfer. This exacerbates the effect of the late bus further, especially if the second bus route has relatively infrequent arrivals (there are many routes in Seattle that arrive once every 30 minutes) (Metro, 2025). Missing a bus or a transfer because of service unreliability is incredibly frustrating. This has been captured in survey research

of transit riders, where service reliability is prioritized over raw speed of service (Pulugurtha et al., 2022). That is, riders would prefer a slower, more predictable system over one that is relatively faster but more unpredictable.

The present research proceeds with the assumptions that (a) transit accessibility should be improved, (b) accessibility can be improved through the improvement of the STP, and (c) the STP may be improved through an improvement to schedule adherence. Reliability is the construct that measures schedule adherence: to what extent does a bus arrive on-time? For the purposes of this project, early and late arrivals are considered to be the same—the occurrence of either one can directly cause a missed transfer. Thus, schedule adherence is measured in terms of absolute deviation (AD), the number of seconds that a bus's actual arrival time is different from its scheduled arrival time.

2.2 Antecedents of Reliability

Empirical research has tackled the idea of transit reliability in an effort to better understand its component parts. That is, what makes a transit system unreliable? Factors such as route length, number of intersections with traffic lights, day of the week, time of day, number of prior stops, dwell time (the amount of time buses sit at a stop between arrival and departure), the presence of bus-only lanes, and traffic conditions have all been found to be related to reliability and some combination of these factors are included in most models seeking to explain variation in arrival times or travel times within mass transit literature (Chen, 2024; Huang et al., 2021; Mohamed et al., 2021).

Another important factor in bus reliability is the bunching phenomenon. "Bunching" occurs when a delay of one bus along a route causes later, on-time buses to be delayed when they get stuck behind the original late bus (Diab et al., 2015). Bus bunching propagates lateness through a route (and sometimes a system) in a series of spillover effects. Much of modern transit management is focused on reducing bus bunching through headway management, or the amount of time between consecutive buses on a route (Diab et al., 2015).

It is important to understand that many of the aforementioned antecedents to reliability may be

correlated with each other and/or contain interaction effects. For example, Monday morning rush hour likely reflects increased traffic congestion, which also magnifies the effects of road features like signalized intersections. Additionally, rush hour traffic may increase the relative effect of a busy road or number of signalized intersections on reliability (Mohamed et al., 2021). Further care will be taken in the methods section of this paper to design an analysis strategy that can easily deal with these issues.

2.3 Bus Rapid Transit and RapidRide Service

With the antecedents of transit reliability in mind, transit agencies can identify strategies and infrastructure choices that directly address these common issues. A simple and common (though expensive) solution is to remove mass transit service from the roads in an effort to mitigate the effects of signalized intersections and traffic congestion. Examples of this are subway systems and light rail lines that are either elevated above street level or tunnel beneath street level. As excellent as these solutions are from the perspective of mitigation of street-level effects, they are associated with astronomical investment in new infrastructure and extended timelines for implementation. In Seattle, for example, the initial construction of the Central Link light rail line (called the "1 Line") took from 2003 to 2009 and cost well over $2 billion (further extensions have taken more time and cost more billions of dollars) (Transit, 2011). Additionally, the in-progress West Seattle Link project has estimated costs of over $5 billion and an expected completion date of 2033 (Packer, 2024). These figures should not be taken as a negative value judgement on light rail transit, but merely that cheaper, more agile options provide a good counterbalance for any transit agency looking to improve service.

Bus Rapid Transit (BRT) offers a suite of operation-management strategies designed to upgrade bus service in terms of speed and reliability. BRT is a rather generic term—there are many strategies that may be considered by a transit agency, and the exact choice will depend on the idiosyncrasies of the locality in question (Muňoz Abogabir & Paget-Seekins, 2016). Some key elements of BRT are bus-only lanes, off-board fare payment (riders pay fares at the stop before the bus arrives and

then may board through any door), raised platforms for level boarding (i.e., no need for the bus to kneel for riders to board), traffic signal priority (the bus receives priority at signalized intersections) and active management of headway between buses (Muñoz Abogabir & Paget-Seekins, 2016).

Figure 1

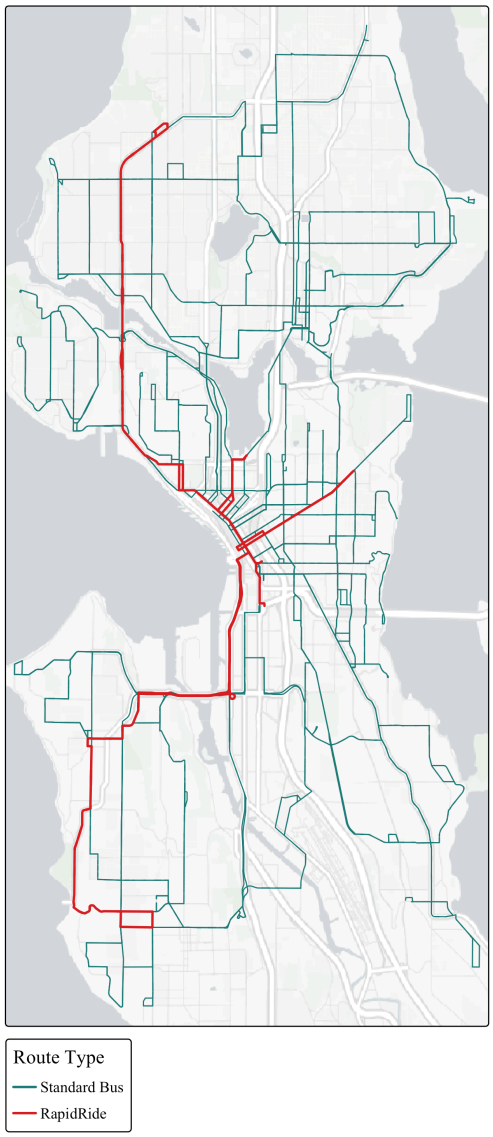

Figure 1: Map of Standard and RapidRide Bus Service in Seattle, WA. Sources: King County Metro, CartoDB.

The implementation of BRT practices does not happen without significant investment of time and money, but this investment is far lower than that required for a major mass transit infrastructure project such as light rail. As such, BRT is a common choice internationally for developing cities as a step to formalize informal bus networks with many independent operators (Muñoz Abogabir & Paget-Seekins, 2016). That said, it is also one of the tools that some cities in the United States have used in an effort to improve municipal transit service.

Seattle introduced the RapidRide program in the early 2010s as an effort to upgrade busy existing bus routes with some of the principles outlined above (Consulting, 2019). It is important to note that, despite characteristics such as off-board fare payment, transit signal priority, headway management, and some bus-only lanes, RapidRide is not technically considered to be a full BRT system by the Institute for Transportation and Development Policy, although the recently-opened G Line and upcoming H, I, and J Lines are closer to full BRT than

previous lines (Orr, 2024; Transportation and Development Policy (ITDP), 2024). Rather, one can think of RapidRide as "BRT-Lite"—an upgrade package for certain bus routes (often the most busy ones) designed to improve reliability and frequency of service (Orr, 2024). Figure 1 shows the bus lines in Seattle, with red lines denoting the RapidRide routes.

As with most public projects, there is a good amount of local discourse surrounding whether RapidRide is a valuable and worthwhile endeavor for the City of Seattle and King County Metro, with criticisms ranging from perceptions of worse service to arguments over whether the investment in RapidRide could be better used on other transit projects (Fesler, 2024; Orr, 2024). This research project does not seek to litigate on most of these issues. The central question to the project remains the somewhat raw effectiveness of RapidRide—does the RapidRide intervention improve reliability on a bus route over standard routes?

2.4 The Social Implications of Transit Performance

Mass transit, in addition to being a policy needed for sustainability goals, is a social justice policy. Given that a good, reliable transit system increases accessibility to the city at large, equitable and efficient dispersion of quality mass transit options allows for diverse groups of people to access more areas of the city, opening up job and recreation opportunities (Covington, 2018). 60% of urban-area zero-vehicle households in the United States are low income, which highlights who is being impacted when public transit is improved (Tomer & Puentes, 2011).

This is an issue that is particularly pertinent in Seattle, a city with a deep history of race-based redlining that carries forward into highly racially homogenized neighborhoods today (Gregory, 2024). In addition, Seattle is a highly vertical city–much longer along the North/South axis than it is wide from West to East. Historically, poorer and less-white neighborhoods tend to be far North and far South, far away from many of the urban centers that contain many high-quality job opportunities and well-funded recreation/entertainment areas (Gregory, 2024).

In essence, the improvement of transit reliability is directly tied to positive outcomes for individuals across the income and race spectrum. It has the potential to get people to their jobs on-time

on a consistent basis, open up more job opportunities to people living further outside the city, and increase accessibility for people and families to quality recreation and entertainment options.

3. Data Overview

Section 3 will outline the dataset used in this study. The dataset is not derived from a single source. Rather, it is an amalgamation of data from several sources, joined at either an individual or group level. Here, I outline data collection for the outcome variable, absolute deviation from schedule (AD), and then pivot to a discussion of which pretreatment covariates were selected and the operationalization of these variables.

3.1 Outcome Data Overview

The dataset started with the outcome variable, AD. Real-time bus arrival times are reported by King County Metro via the General Transit Feed Specification (GTFS), a standard reporting framework for transit agencies. The real-time GTFS feed is available for public access via an application programming interface (API). A request can be made to the API, which will return a list of every stop in the transit network, each of which has a series of real-time timestamps for recent arrivals at that stop. These GTFS data are organized in a roughly hierarchical way. The highest level of this hierarchy is the system as a whole, which can be split into routes. Routes are the geographical multiline segments on which trips run. Trips are unique by day, meaning only one instance of a trip_id is seen per day. Further, trips are directional, meaning it can be seen via an indicator variable in the data which direction along a route the trip travels. Along each trip are stops—pairs of (x, y) coordinates at which buses stop to load and unload passengers. It is important to aggregate stops under trips (rather than routes), because directionality is important. Stop 2 on a route going one direction is the second-to-last stop going the other direction. So, each row in the dataset represents a combination of route, trip, and stop identifiers, which are unique by day. For the data collection in this study, requests were made to the API on an automated schedule (every 15 minutes from 4am to 12am, local time, between 1/29/2025 and 3/31/2025). This procedure resulted in over 4

million rows of observations, from which approximately 54,000 were sampled for model training and test purposes. Scheduled arrival times were joined to the dataset by trip_id/stop_id combination. The final outcome variable, AD, is the absolute difference between the actual arrival time and scheduled arrival time, in seconds. Seconds as a unit were chosen because they represent the most granular unit of reporting available. There was an open question as to whether the outcome variable should discriminate between positive and negative (i.e., whether the bus arrives early or late) or simply be absolute deviation (always positive). The choice of AD was made because to use the positive/negative outcome would have likely required some two-stage modeling with a step to predict whether the arrival delay would be positive or negative, and then a step to predict the magnitude. AD is a simpler construct to model and a bus arriving early or late impacts the rider in much the same way—it often means increased wait times and the possibility for a missed transfer.

3.2 Covariate Data Overview

Length of segment and the RapidRide indicator variables are available via the GTFS static files provided by King County Metro. Length of segment denotes the distance, in meters, that the bus has traveled before any given stop. Length of segment is a directional variable and so was joined to the main dataset by trip_id. The RapidRide indicator was assigned based on route_id. There are three RapidRide routes present in the dataset, accounting for approximately 16 percent of the ~54,000 sampled observations.

Traffic conditions, another important predictor from the literature, were somewhat more difficult to translate into the dataset. Publically-available studies of Seattle traffic either reported traffic levels as a function of time (e.g., traffic levels by hour, by day) or as a function of space (e.g., levels of traffic on major arterials), but not both. Both were added separately to the dataset. Though it is not necessary to standardize variables for use in a BART, both traffic datasets were standardized for better interpretability. Rather than trying to conceptualize how relative vehicle volumes and counts compare to each other, standardization provides a simple intuition where a value of 0 means the observation has an average level of traffic relative to the rest of the city, a 1 indicates the level

of traffic is 1 standard deviation above the mean, and a -1 means the level of traffic is 1 standard deviation below the mean. So, around two-thirds of the data lie within +/- 1 standard deviation from the mean.

The temporal traffic dataset was available on the Seattle Open Data Portal. Each row in this dataset represents a study conducted on a specific date and reports the traffic volume observed during each hour in the day (data.seattle.gov, 2025). The dataset was filtered to keep only studies conducted from 2015 onwards and excluded 2020 and 2021 due traffic disruptions from the COVID-19 pandemic. After these operations, there were 134,170 rows remaining in the dataset. A traffic volume number was appended to each entry in the main dataset, corresponding to the standardized average traffic volume for the weekday and hour that the entry was from.

Spatial traffic data came from a 2018 study (the most recent such study) by the Seattle Department of Transportation and contains traffic volume for each arterial in the city (SDOT, 2018). In order to join these data to the main dataset, spatial overlays were required. King County Metro reports the shapefiles for their bus routes, which are multiline strings (Transit, 2025). One by one, these were overlayed on the spatial traffic geometry (also multiline strings), which were clipped so that only the pieces of the arterials that overlapped with the route shape remained. Then, a weighted average of traffic volume was calculated by multiplying the traffic volume for each arterial by the proportion of the route line it covered. Although all bus routes had some level of coverage from the traffic dataset, not all routes had 100% coverage. In particular, routes that run more often on non-arterial streets would have less coverage. This represents a source of bias in the dataset, but largely a minor one given that most buses run mostly on arterials.

The next set of predictors are four demographic variables: population density, ridership percentage, percentage white, and median household income. Empirical studies have shown that a high volume of ridership tends to predict decreased bus reliability, as a function of increased boarding times, thereby increasing dwell time. Regrettably, the real-time GTFS data used for the outcome variable was not high-resolution enough to report on dwell times (actual arrival and departure times were nearly always identical), so it was necessary to operationalize dwell time in a different way.

The American Community Survey (ACS) 5 year estimates (2019-2023) report the estimated percentage of each census block group that uses the bus to get to work, which gives a rough idea of the demand for buses along each route (Census, 2025). Because transit ridership has been shown to be positively related to areas of lower income, higher non-white percentage, and high density, these variables were also included to better capture variation predicted by high ridership. Again, the route geometry was used—a half-mile buffer was drawn around each route, census block groups with over 50% of their area within the buffer were selected, and a weighted average ridership percentage was calculated using block group population as the weighting variable. The half-mile buffer was selected because it is a commonly used distance threshold for zoning and policy requirements pertaining to bus lines (Council, 2015). For computational resource reasons, these variables were calculated at the full route level, as opposed to the clipping approach used for spatial traffic.

The final covariates in the dataset are weekday and hour. Weekday takes the form of an integer with range 1-7, with 1 corresponding to Sunday. Hour takes the form of an integer with range 1-24. For hour, a 4 denotes the 4:00 to 4:59 time period. These variables were included as key structural elements to the data, primarily designed to handle commuter-based variation in the data that is likely to arise on weekdays during peak hours.

4. Methods

Section 4 will review the conceptual background underpinning causal inference and modeling techniques as they pertain to transit reliability.

4.1 Regression, Causal Inference, and Observational Studies

This study uses a causal inference framework to study the Seattle's RapidRide system, as this methodology is widely used to compare potential outcomes under different applications of some treatment or intervention (Gelman et al., 2023). In this case, the goal is to compare potential outcomes for a route that did receive the RapidRide upgrade "treatment" versus if the same route had not received the treatment. Of course, it is impossible to simultaneously observe a route under

both treatment and non-treatment conditions. Further, assignment of the RapidRide treatment is not random, so the analysis cannot proceed under the assumptions that accompany a randomized experiment. This lack of randomness is made clear in materials produced by King County Metro—routes selected for RapidRide upgrades are among the busiest, high-frequency routes in the city. They tend to run along busy traffic corridors and to/through high residential and job density zones (Metro, 2021). This lack of random assignment creates further work and consideration in the analysis and will be revisited later in this section. For now, the basic formulation of the causal regression model is as follows:

$$Eq.1 : Y \sim \beta X + \theta z + \varepsilon$$

In equation 1, Y represents the outcome (in this case, absolute deviation from schedule, in seconds, for a given bus at a given bus stop). X is the matrix of pretreatment covariates and β is the vector of coefficients for each of these pretreatment variables. Finally, z is the treatment indicator and is equal to 1 for routes who have had the RapidRide treatment applied to them and 0 for routes who have not.

Given that these data are observational, the pretreatment covariates X are particularly important. The regression must adjust for these, otherwise the treatment effect is prone to biases resulting from an imbalanced sample or systematic differences across the treatment and control groups. In this context, it is probable that average traffic congestion, for example, is higher along RapidRide routes than among non-RapidRide routes. The βX term in the regression equation adjusts for such differences and (theoretically) gives an unbiased estimate of the difference in outcome y between the control and treatment groups (henceforth called the treatment effect). Indeed, it can be shown that, in a hypothetical observational study in which there is only one pretreatment variable that impacts both the control and treatment groups, adjustment for this variable will return an unbiased estimate of the treatment effect (Gelman et al., 2023). Of course, in the real world there are many pretreatment variables, not all of which are able to be perfectly accounted for in the analysis. Thus, the adjustment procedures in the regression can only hope to do as much adjustment as possible to

reduce bias. Further weaknesses of the regression models is that (a) they assume a linear relationship unless a different functional form is selected (and the choice of functional form is rife with opportunities for bias) and (b) they only handle interactions between variables at a simplistic level. The following section outlines the model of choice for this study, BARTs, which addresses these weaknesses and will be compared to a traditional multivariate linear regression.

4.2 Bayesian Additive Regression Trees

BART is a nonparametric modeling procedure based on ensemble machine learning methods (Chipman et al., 2010). It provides a nonparametric regression modeling approach through decision trees, avoids overfitting (a common issue with decision trees) through a regularization prior, gives accurate estimates of posterior uncertainty, and elegantly handles heterogeneous treatment effects (Hill, 2011).[1]

Like many Bayesian models, BART uses the Monte Carlo Markov Chain (MCMC) algorithm, a statistical computing algorithm that draws samples from a probability distribution through a random walk, iteratively fitting models. For BARTs, each chain successively builds upon the fit of previous trees, and overfitting is reduced through the regularization prior (Chipman et al., 2005).

4.3 BARTs for Causal Inference

The basic regression tree is a simple machine learning model which creates a series of partitions within the data. The tree begins with a root node, representing a proper subset of the data. From there, the tree performs a series of binary splits (each split is called an interior node), creating a branching series of decision rules that classify the data. At the bottom of the tree, the terminal nodes (also known as leaves) represent the final subgroups and give the mean value of the outcome for observations within these subgroups. A tree is said to have depth 5 if there are 5 decision rules in the longest path from the root node to a terminal node. In regression trees, the algorithm seeks to minimize residual standard error—that is, minimize the average difference between the value of

[1] In a mass transit-focused paper, it is worth stating that Bayesian Additive Regression Trees (BART) should not be confused with Bay Area Rapid Transit (BART).

each terminal node and the values of the observations within those nodes (Chipman et al., 2010). In terms of notation, T represents a binary tree consisting of some number of internal and terminal nodes. $M = mu_1, mu_2, ..., mu_b$ describes the set of parameter values corresponding to each of the b terminal nodes in the tree T. A single regression tree can be expressed in the following way:

$$Eq.2 : Y \sim g(x; T, M) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$$

In equation 2, the outcome Y is given by the function $g(x; T, M)$, which returns a predicted value for an observation with the set of covariates x. The residual error, ε, is assumed to come from a normal distribution with mean 0 and standard deviation sigma. This is analogous to the conditional mean of Y given x, $E(Y|x)$ (Chipman et al., 2010).

A fundamental issue with regression trees is overfitting. Left unchecked, with no limit on depth, the tree will eventually end up with n terminal nodes, where n is the number of observations in the sample, thus fitting the data perfectly. To address this, BARTs use a sum-of-trees approach combined with a regularization prior (the "Additive" and "Bayesian" parts of the BART). This approach seeks to combine a high number of trees while minimizing the amount that each tree contributes to the overall fit through the regularization prior (Chipman et al., 2010).

In BARTs, each tree is considered a "weak learner," meaning the algorithm limits its depth and enforces a strong prior on each terminal node that shrinks its prediction towards 0 (Chipman et al., 2005). The BART model is the sum of many of these trees, and can be expressed with similar notation to that of the single tree:

$$Eq.3 : Y = \sum_{j=1}^{m} g(x; T_j, M_j) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$$

The BART approach treats this sum of trees as the model, and uses the Markov chain Monte Carlo (MCMC) algorithm to iteratively perturb the weak learning trees to improve the fit (Chipman et al., 2005). These perturbations can take the form of adding nodes to the tree (growing), removing nodes from the tree (pruning), or altering a split rule (changing) (Kapelner & Bleich, 2016). It is

worth mentioning that the changing perturbation is only applied to singly internal nodes—ones where both child nodes are terminal nodes (Kapelner & Bleich, 2016). A big relative advantage of BARTs over other ensemble regression tree approaches (e.g., random forest, gradient boosting) is their ability to produce coherent posterior intervals in addition to point estimates (Hill, 2011). The algorithm samples from the posterior distribution via MCMC to quantify uncertainty and behave well. For example, posterior intervals are likely to be wider for predictions at test points farther from the training data set (Chipman et al., 2005).

4.4 Choice of Estimands and Posterior Uncertainty

In cases such as the present research, where multivariate models with interaction effects and non-parametric approaches are used, regression coefficients cannot be directly interpreted as the treatment effect. Rather, an average treatment effect must be calculated by having the model predict the outcome for a sample set of observations and a counterfactual set of these observations (Gelman et al., 2023). Because this study is focused on the effect of the RapidRide intervention, the sample average treatment effect among the treated (SATT) chosen as the causal estimand. Calculating the SATT involves filtering the sample to include only observations that were treated (in this case RapidRide == 1), and using the model to predict fitted values of AD for each observation. Then, the treatment variable is set to the counterfactual (RapidRide == 0) for each of the treated observations, and the model is used to predict fitted values of AD for each of these counterfactual observations. Finally, the counterfactual prediction is subtracted from the factual prediction across each observation and averaged across the sample, producing the SATT.

A major strength of Bayesian approaches is in achieving posterior uncertainty in this estimate. Both the multivariate linear and BART models produce posterior simulation draws (4000 draws for the multivariate linear and 2000 draws for the BART), representing uncertainty in the estimates of the model. The procedure above is repeated for each draw, producing a distribution in individual treatment effects and, thus, a distribution of SATT estimates. The standard deviation of these estimates can be examined to see to what degree the SATT tends to be different from 0 and a confidence

interval can be constructed. In this paper, the standard 95% confidence interval is used, in line with most social science literature.

4.5 Statistical Significance

Before jumping into the analysis and results, it is important to outline what constitutes a meaningful result in the context of this study. Though statistical significance is the status quo when it comes to quantitative analysis, it has many issues that call into question its use in the social sciences. It has been shown that statistical significance is a prerequisite to get research published and that this (a) leads researchers to make decisions designed to achieve statistical significance and (b) tends to bias publishable research to reporting systematically larger effect sizes (Gelman et al., 2023). Further, it is unrealistic to assume a "null" effect (Gelman et al., 2023). In this context, for example, it does not make sense that the RapidRide upgrade would cause absolutely no effect on reliability. The effect may be small (e.g., 5-10 seconds), but this is not a true 0 effect. Rather than looking for whether the estimated parameters and effect sizes in this study cross some arbitrary 95% confidence interval cutoff away from 0 and then deciding that the treatment either works or doesn't, it is better to simply estimate the effect, examine the associated standard error, and interpret the results in the light of this information.

5. Analysis and Results

Section 5 will begin with exploration and discussion of descriptive statistics and assessments of sample balance for causal inference, and then move to model design and evaluation, in which three models will be outlined and tested. Finally, SATT estimates and intervals will be calculated using both Multivariate Linear Regression and BART models.

5.1 Descriptive Statistics

Table 1: Descriptive Statistics

Variable	Mean	Std Dev	Min	25%	Median	75%	Max
RapidRide	0.16	0.37	0.00	0.00	0.00	0.00	1.00
Absolute Deviation	210.64	233.62	0.00	64.00	147.00	276.25	3527.00
Distance Traveled	0.00	1.02	-1.30	-0.80	-0.24	0.62	3.51
Traffic (Day/Hour)	0.00	1.00	-2.74	-0.79	0.25	0.67	1.52
Traffic (Location)	0.01	0.99	-1.51	-0.53	-0.15	0.45	7.81
Population Density	0.00	1.01	-1.82	-0.56	-0.24	0.65	2.49
Route Ridership	-0.01	1.08	-3.54	-0.78	-0.11	0.72	2.95
Percentage White	0.04	0.97	-2.40	-0.35	0.20	0.70	2.04
Median HHI	0.05	0.99	-2.03	-0.76	0.33	0.79	1.68
Weekday	3.79	1.93	1.00	2.00	4.00	5.00	7.00
Hour	14.56	5.08	1.00	10.00	15.00	19.00	24.00

The exploratory data analysis process begins with an overview of the data, variables, and their characteristics. Table 1 contains basic descriptive statistics across the outcome, treatment, and pre-treatment variables in the sample.

RapidRide has a mean of 0.16, indicating that the sample is comprised of 16% of observations occurring at stops along RapidRide routes and 85% of observations occurring along non-RapidRide routes. This distribution is the result of some oversampling of RapidRide observations, as natural incidence rates of RapidRide observations were closer to 8%. This allows the model more information when it comes to estimating the treatment effect but does not grossly overstate the incidence of RapidRide routes in the city, which would be the case if the sampling was done at a 50/50 level.

The average AD for observations in the sample is 210 seconds. In other words, the average bus arrival is 3.5 minutes away from its scheduled arrival time. It is important to note that the median observation is 147 seconds, so the distribution is affected by some extremely late or early

bus arrivals (the greatest of which is nearly 3600 seconds, or about an hour). The standard deviation of 233 seconds indicates very high variability for these outcomes. Though some observations close to the maximum could be considered outliers by the classical definition of $mean + 3 * sd$, they were not removed in order to preserve the accuracy of the sample–sometimes buses run an hour late and it would be a mistake to exclude that information from the analysis.

The following variables were centered and standardized: Distance Traveled, Traffic (Day/Hour), Traffic (Location), Population Density, Route Ridership, Percentage White, and Median HHI. Each of these variables has mean approximately equal to 0 and standard deviation approximately equal to 1, though these are slightly changed as a result of the RapidRide oversampling.

Finally, the roughly similar values of mean and median for Weekday and Hour indicate fairly even distributions of observations across day and hours. For hours, though, the mean being a bit above 12 (the presumed mean), indicates a lack of observations in the early hours of the morning. This is expected since King County Metro does not run buses between around midnight and 4am.

Table 2: Balance Statistics for Causal Inference

Treat	Control	Difference	Ratio
199.04	212.93	-13.89	1.10
0.07	-0.01	0.08	0.84
0.01	0.00	0.01	0.97
0.06	-0.01	0.07	1.20
-0.04	0.01	-0.05	0.96
-0.08	0.01	-0.09	0.52
0.48	-0.05	0.53	4.33
0.60	-0.06	0.66	1.46
3.68	3.81	-0.12	1.03
14.27	14.62	-0.35	1.00

Beyond simple descriptives, it is important to assess balance when dealing with causal inference: do the characteristics of the control and treatment groups looks roughly similar? Table 2 compares the raw means of the two samples in the "treat" and "control" columns, along with the difference. Further, the "ratio" column provides a quick comparison—a value of 1 indicates perfect balance between the treatment and control. The ratio for many covariates is near 1, indicating reasonable balance. There are three variables of slight concern. Route Ridership demand is lower for RapidRide routes than for non-RapidRide routes, and RapidRide routes tend to run though areas that are much whiter and somewhat higher income than non-RapidRide routes. These three variables are correlated with each other (see Figure 2), so imbalance across all three of them is unsurprising. Adjustment within the regression and BART models is designed to account somewhat for these differences, but they do point to possible reasons that the raw difference between the treatment and control may be misleading. Finally, Figure 2 contains a correlation plot for each of the treatment, outcome, and pre-treatment variables. Absolute Deviation is slightly correlated with a number of predictors.

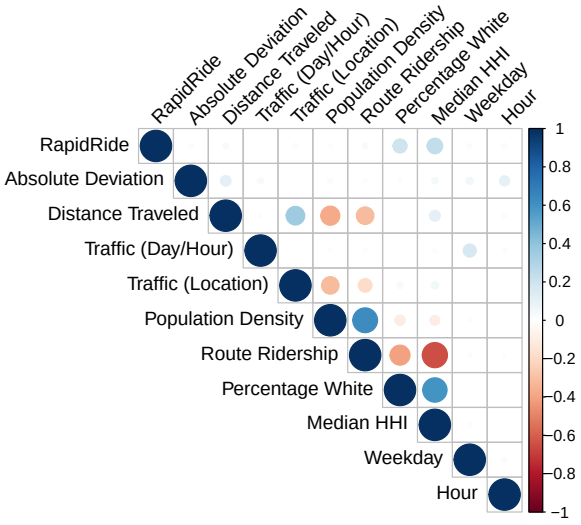

Figure 2: Correlation Plot

5.2 Model Design and Evaluation

The present study considers three models: a simple binary linear regression of AD on RapidRide treatment, a multivariate linear regression of AD on RapidRide treatment and pre-treatment variables, with interactions, and a non-parametric BART model. Section 5.2 will give an overview of the model specifications, assessment of model performance, and a discussion as to which models will be used in the final analysis.

The simple linear regression of AD on the binary RapidRide treatment variable, formalized in equation 4, produces a coefficient that may be interpreted as the difference in means between the control and treatment groups.

$$Eq.4 : AD \sim \theta * RapidRide + \varepsilon$$

As discussed earlier, though, in this observational setting it is key to adjust for pre-treatment variables within the regression model in an effort to account for bias and systematic differences between the groups. Additionally, it is reasonable to think that the RapidRide treatment may be heterogeneous for different subgroups of the data. For example, perhaps it is more effective on certain days or hours, or in areas of relatively higher traffic. Finally, in a simplistic effort to capture possible nonlinearities within the data, two interaction terms between pre-treatment variables were included: distance traveled x traffic (day/hour) and distance traveled x traffic (location), with the hypothesis that traffic effects get more pronounced the further the bus has to go. As such, equation 5 formalizes the regression equation.

$$Eq.5: AD = \beta_0 + \beta_1 RapidRide + \beta_2 X + \beta_3(RapidRide \times X_{subset}) + \beta_4(X_{interaction}) + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

$$X_{subset} = (Weekday,\ Hour,\ Traffic\,(Location),\ Traffic\,(Day/Hour),\ RouteRidership)$$

$$X_{interaction} = Traffic(Location) \times Traffic(Day/Hour) +$$
$$DistanceTraveled \times Traffic(Location) +$$
$$DistanceTraveled \times Traffic(Day/Hour)$$

Finally, the specification for the BART model. As discussed previously, one of the advantages of the BART model is its ability to find nonlinearities within the data and look for heterogeneous treatment effects by subgroup. The specification of the BART model is straightforward, as can be seen in equation 6.

$$Eq.6: AD \sim BART(RapidRide + X)$$

To compare the fit of the BART model to those of the linear regression models, fitted predictions be compared to observed data to calculate the Residual Mean Squared Error (RMSE), where lower RMSE indicates a better fit. Table 5 computes and compares the RMSE for a holdout test set of 50,000 observations. Though the values are relatively similar, there is a clear improvement between the Binary Linear and Multivariate Linear models, as well as the Multivariate and BART models.

Table 3: RMSE by Model

Model	RMSE
Binary Linear	236.36
Multivariate Linear	233.11
Bayesian Additive Regression Trees	225.40

As a final aspect to the model evaluation stage, it is important to examine the posterior residual sigma values for each chain in the BART Model. When the model is well-specified and the algorithm is able to converge on a parameters with relative ease, each of the four chains mixes well with the others, producing similar values of sigma. When the model is not well-specified, the chains often diverge and produce substantially different values of sigma, the standard deviation of the error term (Vehtari et al., 2021). Divergence wastes MCMC chain iterations and can lead to bias in the output of the model. There are two ways to evaluate the model on this front. First, one can simply plot the values of sigma over the iterations of each chain and visually inspect the pattern to see if mixing has occurred. Second, one can calculate r-hat, a metric that compares the variation in sigma within each chain to the variation in sigma across all the chains. A value of 1 is ideal, and a value above 1 indicates that some level of poor mixing has occurred (a value below 1 is not possible). Standard recommendation is that a r-hat value of less than 1.1 is ideal (Vehtari et al., 2021). Figure 3 presents a graph of Sigma Posterior Samples by Chain, along with the estimated r-hat of 1.18. Admittedly, this value of r-hat is above usual recommendations, indicating that additional work should be done to better specify the model or specify better priors to avoid bias. That being said, the later estimates produced by the model are fairly coherent and time in a master's thesis is limited, so the research proceeds with the caveat that the model is sub-optimal.

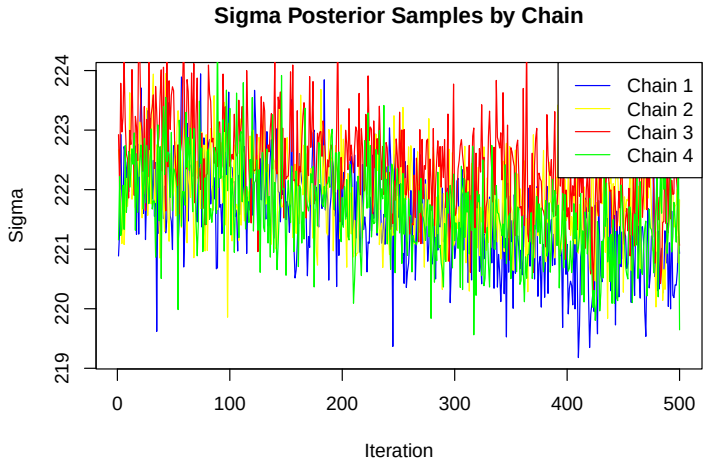

Figure 3: R-Hat = 1.18

5.3 Results

Section 5.3 will begin with a comparison of results for the two regression models. The BART, being nonparametric in nature, returns no regression coefficients and, thus, cannot be compared directly to the regression models in this way. Instead, the BART model and the multivariate linear model will each be used to compute the sample average treatment effect among the treated (SATT), which does allow for a comparison between the two types of model.

5.3.1 Regression Results

Table 4: Linear Regression Coefficients

Parameter	Estimate (Binary)	SE (Binary)	Estimate (Multivariate)	SE (Multivariate)
Intercept	212.98	1.11	216.70	1.15
RapidRide	-13.94	2.74	-105.32	39.29
Distance Traveled	NA	NA	36.09	1.13
Traffic (Day/Hour)	NA	NA	8.84	1.10
Traffic (Location)	NA	NA	-6.77	1.16
Population Density	NA	NA	19.08	1.66
Route Ridership	NA	NA	-7.66	2.56
Percentage White	NA	NA	0.08	1.43
Median HHI	NA	NA	5.46	1.92

Table 4 reports the estimated coefficients for each variable, along with standard error, for the two regression models. Although statistical significance is not a core outcome of interest for this study, multiplying the standard error by 2 provides a reasonable estimate of the upper/lower bounds of a 95% confidence interval. First, it is clear that both models estimate a negative relationship between RapidRide and AD, indicating that RapidRide routes are, on average, more reliable. Further, most predictors in the multivariate model have coefficients that are quite a bit larger than their standard errors, meaning the model estimates they do have an effect on the outcome. Though it is possible to go through the interaction effects and determine the estimated effect of the RapidRide treatment

for each combination of variables, it is simpler to compute the SATT, which will be done in the next section.

5.3.2 Sample Average Treatment Effect among the Treated (SATT)

Because of the less-than-ideal r-hat values for the BART model and the relative similarity between the RMSE values for the BART model and multivariate linear models, both will be used to calculate SATT estimates. The estimates, along with their distributions, can be examined comparatively to get a better idea of what each model is saying about the research question and, together, they should give a decent picture of the underlying effect.

Table 5: SATT, 95% Confidence Interval

Estimate	Lower95	Upper95
-48.70	-84.05	-11.90
-15.36	-24.25	-6.68

To calculate the SATT and include its uncertainty, the dataset was filtered to only include treated observations (RapidRide == 1), resulting in n = 8981 treated observations. Then, the dataset was copied and all of the 1s in the RapidRide column were flipped to 0. This provides the counterfactual—two observations that are identical across all pretreatment variables but that differ on the treatment variable. Each model is then used to estimate fitted predictions for each observation. It is important to note that both models have several thousand posterior sampling draws, so they produce several thousand fitted values per observation. To be precise, the multivariate linear regression produces 4000 estimates per observation and the BART model produces 2000 estimates per observation. These estimates provide the posterior uncertainty associated with the SATT estimates and allow for the construction of confidence intervals. Once the predictions are made, the counterfactual predictions are subtracted from the factual predictions to produce individual estimates of the treatment effect for each observation across each simulation draw. Then, the

individual estimates are averaged for each simulation draw, producing 4000 estimated SATTs for the multivariate linear model and 2000 estimated SATTs for the BART model. These represent the estimated distribution of the SATT. Table 5 provides the estimated median SATT for each model, along with an upper and lower bound consistent with a 95% confidence interval. In this case, a negative estimate indicates a decrease in AD and an improvement in reliability as a result of the RapidRide treatment. Further, both models produce upper bounds that are still negative, indicating high likelihood that the true value of the average treatment effect among the treated is negative.

It is interesting, though, to examine the distributions of the estimates of the SATT between the two models. Figure 4 present density plots of the two distributions, with the BART distribution in blue and the multivariate linear distribution in red. Both look normally distributed, which is expected given that both models rely on the assumption of normally-distributed error terms, but it is clear that the BART estimates are much more variable than the multivariate linear. Given that BARTs are designed to capture complex interactions and nonlinearities within the data, this additional uncertainty is expected. The more important takeaway is the relative agreement of the two models that the effect of the RapidRide intervention is generally an improvement to reliability.

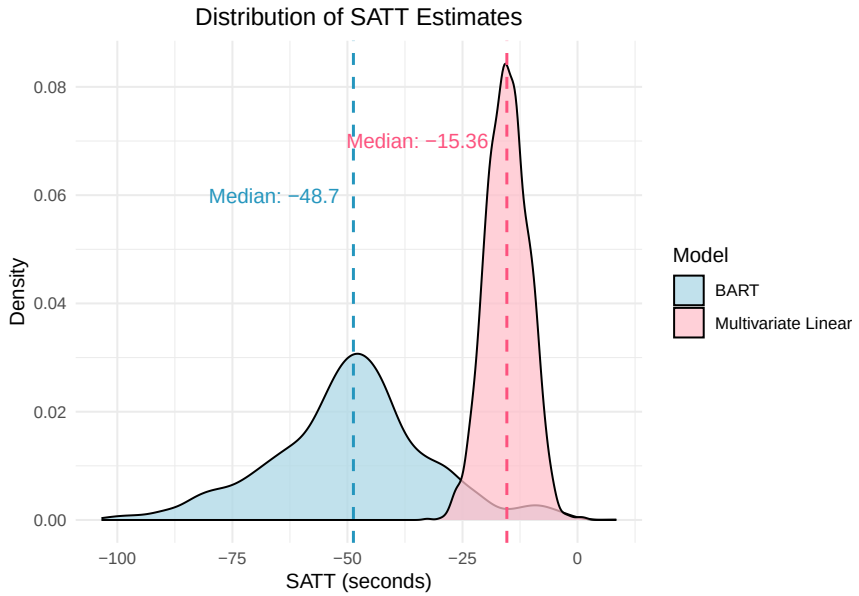

Figure 4: The two models not only differ in their mean estimate of the sample average treatment effect, they also estimate different variability within these estimates.

5.3.3 Model Comparison: Subgroup Estimates

In addition to the single estimate of SATT, it is interesting to examine differing effects for various subgroups of the data. Not only does it provoke interesting discussion about where and when the RapidRide intervention is effective, but it also demonstrates further differences between the models used. For this example, the SATT is calculated separately for each hour of the day in the dataset. There are no observations for hours 2 or 3, so this results in 22 distributions of the SATT for each model. Figure 5 plots these estimates to show the comparison between the BART and multivariate linear models. The circles represent the median estimate, and the error bars represent a 95% confidence interval for the distribution.

The most obvious difference between the models is the variability of the estimates. Using the linear regression model alone, there would be strong evidence that the treatment effect is more pronounced between the hours of 6 and 10, and that it disappears or even changes sign in the afternoon and evening hours. Conversely, the BART model shares information between subgroups, and partially-pools the estimates, resulting in much more stable estimates of the treatment effect. While it seems unlikely that the treatment effect of the RapidRide intervention changes sign for the last few hours of the day, it is not inconceivable. This is somewhat corroborated by the BART estimates—the 95% confidence intervals for the afternoon and evening hours often include 0. Another interesting difference is that the confidence intervals for the multivariate linear model are smaller than the BART for subgroups with larger samples, but the opposite is true for hours with low amounts of observations, such as 1am and 4am. This is, again, due to partial pooling. That said, rather than making a strict value judgement about which model is better than the other, it is likely a better strategy to synthesize what each model says and proceed as if both provide insights.

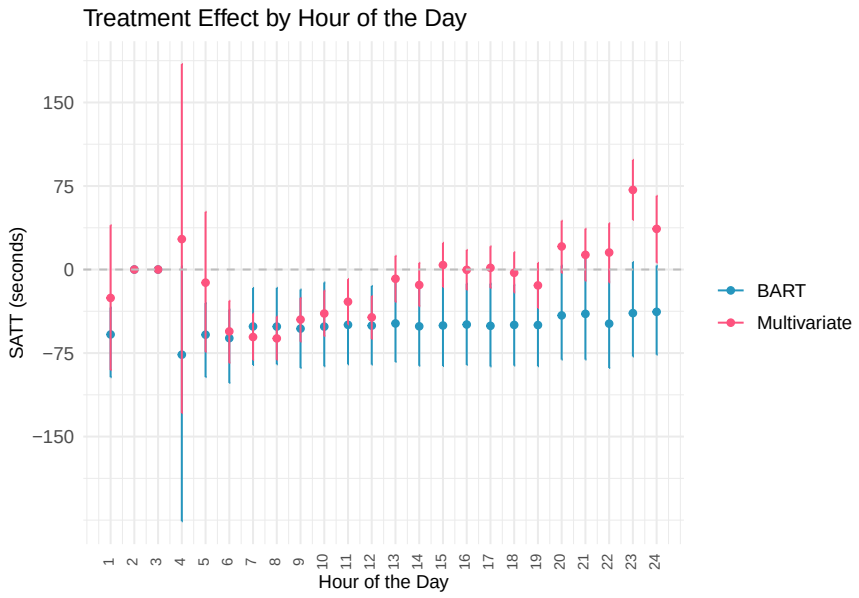

Figure 5: The models vary in their estimated treatment effect by hour, with the BART estimating more homogeneous effects, while the multivariate linear model's estimates are more variable with greater variation in their uncertainty.

6. Discussion

6.1 Implication for Policy

At a very basic level, these results are a good sign for the RapidRide system. Though it has many stated goals, an improvement in reliability of service is a fundamental win for the system. That being said, the magnitude of the improvement is a question that came up time and time again through the analysis of these results. Take, for example, the multivariate linear results, which point to an effect of about 15 seconds. Is this worth it? While certainly better than no change or a decrease in reliability, it is difficult to say whether an average improvement of 15 seconds is a worthwhile investment of several million dollars. On the other hand, the BART estimate of almost 60 seconds seems to be a more enticing effect. To ground these numbers in reality, a 2021 report from King County Metro entitled "Transit Speed & Reliability Guidelines and Strategies" highlights that buses arriving more than a minute early or more than 5-and-a-half minutes late violate the rules of "on-time performance" for standard bus routes and that the goal for RapidRide buses is arrival within 2 minutes on either side of the scheduled arrival time (Metro, 2021). In the context of these figures,

the nearly 60-second average improvement seen for the RapidRide intervention does appear to be impactful relative to existing goals.

Moving beyond simple reliability, mass transit is an important social equity tool that should be used by the city to break down boundaries between neighborhoods in Seattle, a city with a long history of redlining and racial/socioeconomic segregation by neighborhood. A 2019 King County Metro report outlines recommendations from the Mobility Equity Cabinet and identifies increasing and improving bus service in areas of high density and high proportion low-income and people of color as their top recommendation for investment (Metro, 2019). Given that transit should be used to increase access to the wider city from neighborhoods of a diverse range, this evidence that the RapidRide system functions well is a signal that King County and the City of Seattle should continue to fund the program. As was clear from the analysis of balance between RapidRide and non-RapidRide routes, the RapidRide routes within the city disproportionately lie in areas of higher income and higher percentage white residents. Future routes should focus on areas of lower wealth and higher proportions of minority residents.

Another theme that arose through the project was the idea of granularity. In local matters especially, every dollar counts. Policy decisions may start at the 30,000-foot, strategic priority level, but they filter down into fundamental, small-scale decisions about allocation of time and funding to extremely specific areas. This is a major weakness of a lot of traditional (i.e., non-Bayesian, non-pooling) regression methods—when the subgroups get really small, the uncertainty gets really big. There is a common heuristic that you need around 16x the sample size to effectively estimate an interaction effect as compared to a main effect, and this heuristic only refers to a single interaction, not to speak of varying treatment effects by hour of the day, day of the week, and on a specific bus route. Through partial-pooling and Bayesian approaches, these subgroups can at least be estimated with a higher degree of precision, even if the smallest subgroups are pooled in towards the overall mean. BART models are designed to handle this sort of thing, while providing the ability to estimate non-linearities easily, making them a really valuable tool for causal inference moving forward.

6.2 Limitations

Modeling is defined in large part by its limitations, and this project is no different. Broadly, the limitations of the project can be placed into three categories: data quality, generalizability, and model specification.

Some of the data for this project is extremely accurate. The GTFS real-time feeds and data on route distances/RapidRide treatment are highly accurate and suffer from essentially no missingness whatsoever. The main data quality issue for the dataset is in the application of covariates to individual observations. First, an ideal dataset would include real-time traffic estimates for each observation, rather than aggregate estimates for time and place based on historical data. This change, simple to outline and difficult to actually implement, would likely improve the predictive accuracy of the model by a large margin. Additionally, the operationalization of the ridership variable is imperfect—the census variables would be better replaced by route-level ridership data combined with daily ridership numbers, in an ideal world. Further, the variable set is likely insufficient to satisfy the assumption that all confounding covariates are adjusted for within the model.

Further, the inferences associated with this study are only truly generalizable to the time period in which the data was collected (the first few months of 2025). It may be reasonable to generalize the findings to other months and years, but it is hard to say this without additional evidence.

Lastly, there are the previously outlined issues with model specification. The BART model did not mix particularly well, and it is likely that the addition of some priors or some other model specification would improve the fit and, thus, the inference. Additionally, there are more complex versions of the BART model that allow for concretely defined hierarchical structure to the data (Dorie et al., 2022). These data are structured by day and hour, as well as by stop-trip-and-route. Accounting for these explicitly in a multilevel structure and then using a BART term for the remaining covariates would likely provide better causal inference than the current models.

6.3 Future Directions

With the limitations above addressed, this model becomes a powerful tool for public policy researchers and decision-makers. The first future direction for this research is a wider-scale study involving samples from throughout a year, with further adjustment for seasonal/holiday trends. Further, there are additional RapidRide bus routes outside of the Seattle city limits. While all routes outside the city limits (RapidRide and otherwise) were omitted from the current research because they lacked spatial traffic data, a more inclusive study would provide better inference for the project as a whole.

A huge possibility for this research is in cross-city comparison. Many cities have bus rapid transit systems, and inferences across cities could be combined for a better understanding of the intervention as a whole. The standardized GTFS structure of real-time transit data used in cities across the US and around the world makes data comparable across cities, which is a big boon to modeling. Further, one weakness of the present research is that it fails to discriminate between different aspects of the RapidRide intervention—that is, it is unknown which part(s) of the intervention is responsible for the positive impact on reliability. Because many cities implement BRT differently, it may be possible to subclassify BRT programs based on the specifics of their upgrades, and isolate which of traffic signal priority, decreased dwell times, headway management, etc. are most effective in improving transit reliability.

Lastly, the SATT estimates for this study were calculated on RapidRide routes, but the procedure can be reversed, so the counterfactual turns non-RapidRide routes into RapidRide ones. BARTs are well-suited to this task as well—they account for uncertainty when balance and support between the treatment and control groups are not good, meaning inference for control observations without solid support and balance from the treatment group would have more uncertainty associated with them. This could be used by policymakers to identify routes with the greatest potential for reliability improvements, alongside traditional indicators that make a route attractive for an upgrade.

7. Conclusion

Causal inference in observational settings is hard, and the case of RapidRide in Seattle is no different. The treatment is demonstrably applied in a non-random way, creating a fundamental violation of the principle of ignorability. Further, it is more interesting to examine the current state of the program, rather than trying to achieve a pre-post setup when the intervention was implemented, given that it changes and evolves over time. So, the question of what effect the RapidRide program has on the reliability of a bus route in 2025 is a question that can only be answered with modeling and adjustment. BARTs and multivariate linear regression both provide solutions, and the inferences that come from these models can help inform policymakers as to whether the intervention is successful. The present study provides an initial result that the RapidRide intervention has the intended effect on reliability and reduces absolute deviation from schedule on a bus route by an average of nearly 90 seconds per minute. Further results in subgroups provide future areas for research—granular understanding of these patterns is paramount to better-allocated spending on mass transit in the future.

8. References

Anderson, K. F., & Galaskiewicz, J. (2021). Racial/Ethnic Residential Segregation, Socioeconomic Inequality, and Job Accessibility by Public Transportation Networks in the United States. *Spatial Demography*, *9*(3), 341–373. https://doi.org/10.1007/s40980-021-00093-8

Antipova, A., Sultana, S., Hu, Y., & Rhudy, J. P. (2020). Accessibility and Transportation Equity. *Sustainability*, *12*(9), 3611. https://doi.org/10.3390/su12093611

Census, U. (2025). *American Community Survey (ACS)*. https://www.census.gov/programs-surveys/acs

Chen, X. (2024). *Bayesian inference and forecasting methods in public transit systems*.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2005). *BART: Bayesian Additive Regression Trees*.

Chipman, H. A., George, E. I., & McCulloch, R. E. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, *4*(1). https://doi.org/10.1214/09-AOAS285

Consulting, S. S. (2019). *Bus Reliability*.

Council, P. S. R. (2015). *Transit-Supportive Densities and Land Uses*.

Covington, K. L. (2018). Overcoming Spatial Mismatch: The Opportunities and Limits of Transit Mode in Addressing the Black–White Unemployment Gap. *City & Community*, *17*(1), 211–235. https://doi.org/10.1111/cico.12278

data.seattle.gov. (2025). *Traffic Count Studies by Hour Bins*. https://catalog.data.gov/dataset/traffic-count-studies-by-hour-bins

Diab, E., Bertini, R., & El-Geneidy, A. (2015). *Bus transit service reliability: Understanding the impacts of overlapping bus service on headway delays and determinants of bus bunching*.

Dorie, V., Perrett, G., Hill, J. L., & Goodrich, B. (2022). Stan and BART for Causal Inference: Estimating Heterogeneous Treatment Effects Using the Power of Stan and the Flexibility of Machine Learning. *Entropy*, *24*(12), 1782. https://doi.org/10.3390/e24121782

Dutta, P. K. (2013). Taking the Car out of Carbon. In J. L. Renne & B. Fields (Eds.), *Transport Beyond Oil* (pp. 126–140). Island Press/Center for Resource Economics. https://doi.org/10.5822/978-1-59726-242-2_8

Fesler, S. (2024). *The Case Against RapidRide and For Funding Massive Transit Service Expansion Now*. https://www.theurbanist.org/2024/02/29/the-case-against-rapidride-and-for-funding-massive-transit-service-expansion-now/

Gelman, A., Hill, J., & Vehtari, A. (2023). *Regression and Other Stories*.

Gregory, J. (2024). Homes for Some: Seattle's History of Housing and Racial Exclusion. *Pacific Northwest Quarterly*, *115*(1), 26–36. http://ezproxy.cul.columbia.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=31h&AN=182205574&site=ehost-live&scope=site

Hägerstrand, T. (1970). What about people in regional science. *Transport Sociology: Social Aspects of Transport Planning*, 143–158.

Handy, S. (2020). Is accessibility an idea whose time has finally come? *Transportation Research Part D: Transport and Environment*, *83*, 102319. https://doi.org/10.1016/j.trd.2020.102319

Hansen, W. G. (1959). How Accessibility Shapes Land Use. *Journal of the American Institute of Planners*, *25*(2), 73–76. https://doi.org/10.1080/01944365908978307

Hill, J. L. (2011). Bayesian Nonparametric Modeling for Causal Inference. *Journal of Computational and Graphical Statistics*, *20*(1), 217–240. https://doi.org/10.1198/jcgs.2010.08162

Huang, Y. P., Chen, C., Su, Z. C., Chen, T. S., Sumalee, A., Pan, T. L., & Zhong, R. X. (2021). Bus arrival time prediction and reliability analysis: An experimental comparison of functional data analysis and Bayesian support vector regression. *Applied Soft Computing*, *111*, 107663. https://doi.org/10.1016/j.asoc.2021.107663

Kapelner, A., & Bleich, J. (2016). **bartMachine** : Machine Learning with Bayesian Additive Regression Trees. *Journal of Statistical Software*, *70*(4). https://doi.org/10.18637/jss.v070.i04

Levinson, D., & Wu, H. (2020). Towards a general theory of access. *Journal of Transport and Land Use*, *13*(1), 129–158. https://www.jstor.org/stable/26967239

Liu, L., Porr, A., & Miller, H. J. (2023). Realizable accessibility: Evaluating the reliability of public transit accessibility using high-resolution real-time data. *Journal of Geographical Systems*, *25*(3), 429–451. https://doi.org/10.1007/s10109-022-00382-w

Metro, K. C. (2019). *King County Metro Mobility Framework Recommendations Summary*. https://cdn.kingcounty.gov/-/media/king-county/depts/metro/documents/about/policies/2019-10-mobility-framework-recommendations-attachment-a.pdf?rev=42a41e95d3534c43b16caea89d258a6a&hash=974840868AA30E276B3BDB6269D0816F

Metro, K. C. (2021). *TRANSIT SPEED & RELIABILITY GUIDELINES & STRATEGIES*. https://kingcounty.gov/~/media/depts/metro/about/planning/speed-reliability-toolbox.pdf

Metro, K. C. (2025). *Schedules and maps*. https://kingcounty.gov/en/dept/metro/routes-and-service/schedules-and-maps

Mohamed, A. H., Adwan, I. A. I., Ahmeda, A. G. F., Hrtemih, H., & Al-MSari, H. (2021). Identification of Affecting Factors on the Travel Time Reliability for Bus Transportation. *Knowledge-Based Engineering and Sciences*, *2*(1), 19–30. https://doi.org/10.51526/kbes.2021.2.1.19-30

Muňoz Abogabir, J. C., & Paget-Seekins, L. (2016). *Restructuring public transport through bus rapid transit: An international and interdisciplinary perspective*. Policy press.

O'Sullivan, D., Morrison, A., & Shearer, J. (2000). Using desktop GIS for the investigation of accessibility by public transport: An isochrone approach. *International Journal of Geographical Information Science*, *14*(1), 85–104. https://doi.org/10.1080/136588100240976

Orr, M. (2024). *No More RapidRide*. https://seattletransitblog.com/2024/03/03/no-more-rapidride/

Packer, R. (2024). *West Seattle Link Cost Estimates Jump $1.6 Billion*. https://www.theurbanist.org/2024/09/13/west-seattle-link-cost-estimates-jump/

Pulugurtha, S. S., Mishra, R., University of North Carolina at Charlotte, Jayanthi, S. L., & University of North Carolina at Charlotte. (2022). *Does Transit Service Reliability Influence Ridership?* Mineta Transportation Institute. https://doi.org/10.31979/mti.2022.2118

SDOT. (2018). *2018 Traffic Flow Counts*. https://data-seattlecitygis.opendata.arcgis.com/datasets/SeattleCityGIS::2018-traffic-flow-counts/explore?location=47.625471%2C-122.341696%2C11.76

Tomer, A., & Puentes, R. (2011). *Transit Access and Zero- Vehicle Households*. https://www.infrastructureusa.org/wp-content/uploads/2011/08/0818_transportation_tomer.pdf

Transit, S. (2011). *Final major construction contract for Central Link ready for close-out*. https://www.soundtransit.org/get-to-know-us/news-events/news-releases/final-major-construction-contract-central-link-ready

Transit, S. (2025). *Open Transit Data (OTD)*. https://www.soundtransit.org/help-contacts/business-information/open-transit-data-otd/otd-downloads

Transportation and Development Policy (ITDP), I. for. (2024). *The Bus Rapid Transit Standard*. https://itdp.org/library/standards-and-guides/the-bus-rapid-transit-standard/

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., & Bürkner, P.-C. (2021). Rank-Normalization, Folding, and Localization: An Improved R^ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis*, *16*(2). https://doi.org/10.1214/20-BA1221

Wessel, N., Allen, J., & Farber, S. (2017). Constructing a routable retrospective transit timetable from a real-time vehicle location feed and GTFS. *Journal of Transport Geography*, *62*, 92–97. https://doi.org/10.1016/j.jtrangeo.2017.04.012

Wessel, N., & Farber, S. (2019). On the accuracy of schedule-based GTFS for measuring accessibility. *Journal of Transport and Land Use*, *12*(1), 475–500. https://www.jstor.org/stable/26911278

Appendices

Appendix A: Full Regression Results Table

Table 6: Linear Regression Coefficients

Parameter	Estimate (Binary)	SE (Binary)	Estimate (Multivariate)	SE (Multivariate)
Intercept	212.98	1.11	216.70	1.15
RapidRide	-13.94	2.74	-105.32	39.29
Distance Traveled	NA	NA	36.09	1.13
Traffic (Day/Hour)	NA	NA	8.84	1.10
Traffic (Location)	NA	NA	-6.77	1.16
Population Density	NA	NA	19.08	1.66
Route Ridership	NA	NA	-7.66	2.56
Percentage White	NA	NA	0.08	1.43
Median HHI	NA	NA	5.46	1.92
avg_traffic_dayhour:spatial_congestion	NA	NA	0.10	1.05
shape_dist_traveled:avg_traffic_dayhour	NA	NA	4.69	1.06
shape_dist_traveled:spatial_congestion	NA	NA	-11.55	1.21
rapid_ride:factor(g_weekday)2	NA	NA	31.59	10.70
rapid_ride:factor(g_weekday)3	NA	NA	44.80	11.33
rapid_ride:factor(g_weekday)4	NA	NA	51.83	11.36
rapid_ride:factor(g_weekday)5	NA	NA	41.49	11.88
rapid_ride:factor(g_weekday)6	NA	NA	48.57	12.80
rapid_ride:factor(g_weekday)7	NA	NA	55.07	11.24
rapid_ride:factor(g_hr)4	NA	NA	29.19	84.83
rapid_ride:factor(g_hr)5	NA	NA	3.48	46.31
rapid_ride:factor(g_hr)6	NA	NA	-25.96	35.60
rapid_ride:factor(g_hr)7	NA	NA	-16.51	35.34
rapid_ride:factor(g_hr)8	NA	NA	5.18	37.95
rapid_ride:factor(g_hr)9	NA	NA	31.98	39.68
rapid_ride:factor(g_hr)10	NA	NA	33.74	38.16
rapid_ride:factor(g_hr)11	NA	NA	39.44	37.87
rapid_ride:factor(g_hr)12	NA	NA	29.30	37.40
rapid_ride:factor(g_hr)13	NA	NA	64.96	38.99
rapid_ride:factor(g_hr)14	NA	NA	62.70	38.66
rapid_ride:factor(g_hr)15	NA	NA	83.67	40.35
rapid_ride:factor(g_hr)16	NA	NA	85.16	40.76
rapid_ride:factor(g_hr)17	NA	NA	95.47	42.09
rapid_ride:factor(g_hr)18	NA	NA	90.48	41.11
rapid_ride:factor(g_hr)19	NA	NA	65.12	40.18
rapid_ride:factor(g_hr)20	NA	NA	88.39	37.38
rapid_ride:factor(g_hr)21	NA	NA	68.81	35.74
rapid_ride:factor(g_hr)22	NA	NA	66.97	36.68
rapid_ride:factor(g_hr)23	NA	NA	111.42	35.60
rapid_ride:factor(g_hr)24	NA	NA	63.80	37.13
rapid_ride:spatial_congestion	NA	NA	-9.01	3.34
rapid_ride:route_ridership	NA	NA	9.70	2.35
rapid_ride:avg_traffic_dayhour	NA	NA	-19.89	7.85

Appendix B: Python Code

Code and datasets for this project can be found on GitHub: https://github.com/Peter-Silverstein/bus-delay-modeling

```
# <---------- API PULL FOR REAL-TIME GTFS DATA ---------->

# Used with AWS Lambda for automation
import requests
from google.transit import gtfs_realtime_pb2
from google.protobuf.json_format import MessageToDict
import boto3
import json
from datetime import datetime
import logging

def lambda_handler(event, context):
    # Define API details
    API_KEY = "2c97496e-e814-4cd6-bb23-14413a2a480d"
    FEED_URL = f"""
    http://api.pugetsound.onebusaway.org/api/gtfs_realtime/trip-
    updates-for-agency/1.pb?key={API_KEY}"""

    logger = logging.getLogger()
    logger.setLevel(logging.INFO)

    # Main workflow
    try:
        # Fetch and parse feed
        feed_content = fetch_gtfs_realtime(FEED_URL)
        parsed_feed = parse_gtfs_feed(feed_content)

        # Extract relevant data
        trips = extract_trip_data(parsed_feed)

        # Convert trips to JSON string
        json_data = json.dumps(trips, indent=4)

        # Setting name for file
        current_time = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        OBJECT_NAME = f"realtime_trip_data_{current_time}.json"
        BUCKET_NAME = "gtfs-data-run1"

        # Save JSON data to S3
        s3_client = boto3.client('s3')
```

```
        s3_client.put_object(
            Bucket=BUCKET_NAME,
            Key=OBJECT_NAME,
            Body=json_data,
            ContentType='application/json'
        )

        logger.info(f"""
        Data successfully saved to S3 bucket '{BUCKET_NAME}' as '{OBJECT_NAME}'
        """)
    except Exception as e:
        logger.error(f"Error occurred: {e}")

# Function to fetch GTFS-Realtime feed
def fetch_gtfs_realtime(feed_url):
    response = requests.get(feed_url)
    response.raise_for_status()  # Raise an exception for HTTP errors
    return response.content

# Function to parse GTFS-Realtime feed
def parse_gtfs_feed(feed_content):
    feed = gtfs_realtime_pb2.FeedMessage()
    feed.ParseFromString(feed_content)
    return MessageToDict(feed)

# Function to extract trip data
def extract_trip_data(parsed_feed):
    trip_data = []
    for entity in parsed_feed.get("entity", []):
        if "tripUpdate" in entity:
            trip_update = entity["tripUpdate"]
            trip_id = trip_update.get("trip", {}).get("tripId", "")
            route_id = trip_update.get("trip", {}).get("routeId", "")
            agency_id = trip_update.get("trip", {}).get("agencyId", "")
            stop_time_updates = trip_update.get("stopTimeUpdate", [])

            for stop_time_update in stop_time_updates:
                stop_id = stop_time_update.get("stopId", "")
                arrival_time = stop_time_update.get("arrival", {}).get(
                  "time", "")
                departure_time = stop_time_update.get("departure", {}).get(
                  "time", "")

                # Append relevant data as a dictionary
```

```
                trip_data.append({
                    "trip_id": trip_id,
                    "route_id": route_id,
                    "agency_id": agency_id,
                    "stop_id": stop_id,
                    "arrival_time": arrival_time,
                    "departure_time": departure_time,
                })
    return trip_data
```

```
# <---------- PULLING JSON FROM AWS S3 STORAGE ---------->

import boto3
import os
import pandas as pd
import json

def download_all_files(bucket_name, local_dir):
    s3 = boto3.client('s3')
    paginator = s3.get_paginator('list_objects_v2')
    pages = paginator.paginate(Bucket = bucket_name)

    for page in pages:
        if 'Contents' in page:
            for obj in page['Contents']:
                key = obj['Key']
                local_file_path = os.path.join(local_dir, key)

                # Create directories if they don't exist already
                os.makedirs(os.path.dirname(local_file_path), exist_ok = True)

                # Download the file
                s3.download_file(bucket_name, key, local_file_path)

# Use function
bucket_name = 'gtfs-data-run1'
local_dir = 'raw_from_awsS3'
download_all_files(bucket_name, local_dir)

print("All files downloaded")

# Function to extract trip data
def extract_trip_data(parsed_feed):
    trip_data = []
```

```
    for entity in parsed_feed.get("entity", []):
        if "tripUpdate" in entity:
            trip_update = entity["tripUpdate"]
            trip_id = trip_update.get("trip", {}).get("tripId", "")
            route_id = trip_update.get("trip", {}).get("routeId", "")
            agency_id = trip_update.get("trip", {}).get("agencyId", "")
            stop_time_updates = trip_update.get("stopTimeUpdate", [])

            for stop_time_update in stop_time_updates:
                stop_id = stop_time_update.get("stopId", "")
                arrival_time = stop_time_update.get("arrival", {}).get(
                  "time", "")
                departure_time = stop_time_update.get("departure", {}).get(
                  "time", "")

                # Append relevant data as a dictionary
                trip_data.append({
                    "trip_id": trip_id,
                    "route_id": route_id,
                    "agency_id": agency_id,
                    "stop_id": stop_id,
                    "arrival_time": arrival_time,
                    "departure_time": departure_time,
                })
    return trip_data

# Function to iterate over the folder
def json_to_df(folder_path):
    all_data = [] # Empty list to store all our dataframes

    # Iterate through all JSON files in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith('.json'):
            file_path = os.path.join(folder_path, filename)

            # Open and load the JSON file
            with open(file_path, 'r') as f:
                parsed_feed = json.load(f)

            # Extract trip data
            trip_data = extract_trip_data(parsed_feed)

            # Convert to dataframe
            df = pd.DataFrame(trip_data)
```

```
            # Add a column for the source filename
            df['source_file'] = filename

            # Append to the list of dataframes
            all_data.append(df)

    # Combine all dataframes into one
    combined_df = pd.concat(all_data, ignore_index = True)

    return combined_df

# Apply the function
local_dir = 'raw_from_awsS3'
combined_df = json_to_df(local_dir)

print(combined_df.head(30))
```

```
# <---------- SAVING TO POSTGRESQL DATABASE FOR FUTURE USE ---------->

import os
import pandas as pd
import numpy as np
import json
import psycopg2
from datetime import datetime
from datetime import timedelta
from datetime import timezone
from psycopg2 import sql
from io import StringIO
from zoneinfo import ZoneInfo

# Extract Trip Data function
def extract_trip_data(parsed_feed):
    trip_data = []
    # Check if parsed_feed is a list
    if isinstance(parsed_feed, list):
        for item in parsed_feed:
            # Directly access the trip data from each item
            trip_id = item.get('trip_id', '')
            route_id = item.get('route_id', '')
            stop_id = item.get('stop_id', '')
            arrival_time = item.get('arrival_time', '')
            departure_time = item.get('departure_time', '')
```

```
            # Append relevant data as a dictionary
            trip_data.append({
                'trip_id': trip_id,
                'route_id': route_id,
                'stop_id': stop_id,
                'arrival_time': arrival_time,
                'departure_time': departure_time,
            })
    return trip_data

# Function to iterate over the folder and process JSON files
def json_to_df(folder_path):
    all_data = []  # List to store all dataframes

    # Iterate through all JSON files in the folder
    for filename in os.listdir(folder_path):
        if filename.endswith('.json'):
            file_path = os.path.join(folder_path, filename)

            # Open and load the JSON file
            with open(file_path, 'r') as f:
                parsed_feed = json.load(f)

            # Extract trip data using the updated function
            trip_data = extract_trip_data(parsed_feed)

            # Convert to dataframe
            df = pd.DataFrame(trip_data)

            # Add a column for the source filename
            df['source_file'] = filename

            # Append to the list of dataframes
            all_data.append(df)

    # Combine all dataframes into one
    combined_df = pd.concat(all_data, ignore_index=True)

    return combined_df

# Function to convert Unix time to date and seconds after midnight in PST
def convert_to_date_and_time(unix_time):
    if unix_time:
        utc_time = datetime.fromtimestamp(int(unix_time), tz = timezone.utc)
```

```
        # Adjust for PST (UTC-8)
        pst_time = utc_time.astimezone(ZoneInfo("America/Los_Angeles"))
        combined_datetime = pst_time.strftime('%Y-%m-%d %H:%M:%S')
        return combined_datetime
    return None

# Application
local_dir = 'raw_from_awsS3'
combined_df = json_to_df(local_dir)

# Cleaning Data

# Converting columns to more useful types
combined_df = combined_df.astype({"source_file": "string"})

# Replace empty strings with NaN in some columns
columns_to_replace = ['trip_id', 'route_id', 'stop_id']
combined_df[columns_to_replace] = combined_df[columns_to_replace].replace(
  '', pd.NA)

# Remove rows where the arrival_time column has an empty value
combined_df = combined_df.dropna(subset=['arrival_time'])

# Apply the conversion function to arrival and departure times
combined_df["arrival_datetime"] = combined_df["arrival_time"].apply(
  convert_to_date_and_time)
combined_df["departure_datetime"] = combined_df["departure_time"].apply(
  convert_to_date_and_time)

# Use the source_file column to extract the day/time of the pull
combined_df['pull_datetime'] = combined_df['source_file'].str.extract(
  r'_(\d{4}-\d{2}-\d{2}_\d{2}-\d{2}-\d{2})\.json$')
combined_df[['pull_date', 'pull_time']] = combined_df[
  'pull_datetime'].str.split('_', expand=True)
combined_df['pull_time'] = combined_df['pull_time'].str.replace('-', ':')
combined_df['pull_datetime'] = pd.to_datetime(
  combined_df['pull_date'] + ' ' + combined_df['pull_time'],
  errors='coerce', utc = True)
combined_df['pull_datetime'] = combined_df['pull_datetime'].dt.tz_convert(
  ZoneInfo("America/Los_Angeles"))

# Compare the pull and arrival day/time to check if the time is forecasted
combined_df["projection"] = np.where((
  combined_df["arrival_datetime"] > combined_df["pull_datetime"]), 1, 0)
```

```
# Removing the source_file,
combined_df = combined_df.drop(["arrival_time",
                                "departure_time",
                                "source_file",
                                "pull_date",
                                "pull_time"],
                                axis = 1)

# Re-typing the columns; replacing NA values with None
combined_df.replace({pd.NA: None, pd.NaT: None}, inplace=True)

combined_df = combined_df.astype({
    "trip_id": "string",
    "route_id": "string",
    "stop_id": "string",
    "arrival_datetime": "datetime64[ns, America/Los_Angeles]",
    "departure_datetime": "datetime64[ns, America/Los_Angeles]",
    "pull_datetime": "datetime64[ns, America/Los_Angeles]"
})

# Print to check
print(combined_df.dtypes)
print(combined_df.head())

# Writing data to PostgreSQL (!!)
# Function to create the table if it does not exist
def create_table_if_not_exists(conn):
    with conn.cursor() as cur:
        cur.execute("""
            CREATE TABLE IF NOT EXISTS sea_gtfs_data (
                unique_id SERIAL PRIMARY KEY,
                trip_id TEXT,
                route_id TEXT,
                stop_id TEXT,
                arrival_datetime TIMESTAMPTZ,
                departure_datetime TIMESTAMPTZ,
                pull_datetime TIMESTAMPTZ,
                projection BOOLEAN
            )
        """)
        conn.commit()

# Function to bulk load a DataFrame into the PostgreSQL table
def bulk_insert_dataframe(conn, df, table_name):
```

```
    buffer = StringIO()
    df.to_csv(buffer, index=False, header=False)
    buffer.seek(0)

    columns = ["trip_id", "route_id", "stop_id",
               "arrival_datetime", "departure_datetime",
               "pull_datetime", "projection"]

    with conn.cursor() as cur:
        cur.copy_expert(
            sql.SQL("COPY {} ({}) FROM STDIN WITH CSV").format(
                sql.Identifier(table_name),
                sql.SQL(', ').join(map(sql.Identifier, columns))
            ),
            buffer
        )
        conn.commit()

# Main script
# MODIFY THESE LINES IN YOUR CODE
# In the main script section:

def main():
    # Database connection parameters
    db_params = {
        "dbname": "sea-gtfs-data",
        "user": "postgres",
        "password": "Parkour",
        "host": "localhost",
        "port": 5432
    }

    # Connect to the database
    conn = psycopg2.connect(**db_params)

    try:
        # Create table if not exists
        create_table_if_not_exists(conn)

        # NEW: Process data in batches
        chunk_size = 50000  # Adjust based on your system's capacity
        total_rows = len(combined_df)

        for start in range(0, total_rows, chunk_size):
```

```
            end = min(start + chunk_size, total_rows)
            chunk = combined_df.iloc[start:end]

            print(f"Processing rows {start+1}-{end} of {total_rows}")

            # NEW: Clear buffers after each chunk
            with conn:
                with conn.cursor() as cur:
                    buffer = StringIO()
                    chunk.to_csv(buffer, index=False, header=False, columns=[
                        "trip_id", "route_id", "stop_id",
                        "arrival_datetime", "departure_datetime",
                        "pull_datetime", "projection"
                    ])
                    buffer.seek(0)

                    copy_sql = sql.SQL("""
                        COPY sea_gtfs_data (
                            trip_id, route_id, stop_id,
                            arrival_datetime, departure_datetime,
                            pull_datetime, projection
                        ) FROM STDIN WITH CSV
                    """)

                    cur.copy_expert(copy_sql, buffer)
                    conn.commit()

                    # Explicitly clean up resources
                    buffer.close()
                    del buffer

    finally:
        conn.close()


if __name__ == "__main__":
    main()
```

Appendix C: R Code

```
# Loading Libraries

# General Use
```

```r
library(tidyverse)
library(ggplot2)
library(here)
library(patchwork)
library(modelsummary)
library(knitr)
library(keyring)
library(lubridate)
library(data.table)

# Modeling
library(stan4bart)
library(bartCause)
library(rstanarm)
library(bayesplot)

# PostgreSQL
library(DBI)
library(RPostgres)

# GIS and Mapping
library(sf)
library(tmap)
```

```r
# <---------- GENERAL DATA LOADING, CLEANING, MANAGEMENT ---------->

# Loading Data
con <- dbConnect(RPostgres::Postgres(),
                 dbname = "sea-gtfs-data",
                 host = "localhost",
                 port = 5432,
                 user = "postgres",
                 password = "Parkour")

gtfs_realtime <- dbReadTable(con, "sea_gtfs_data")
gtfs_realtime <- tibble(gtfs_realtime)
```

```r
# Filtering dataset to work with my spatial congestion dataset +
# include RapidRide C, D, G lines

# Current CRS NAD83
kc_route_shp <- st_read(here("Predictor Data Sets",
            "KCMetro_Transit_Lines",
            "Transit_Routes_for_King_County_Metro___transitroute_line.shp")) %>%
```

```
  select(ROUTE_ID, geometry) %>%
  distinct(ROUTE_ID, .keep_all = TRUE)

routes <- read.csv(here("Predictor Data Sets",
                        "gtfs-static-files/routes.txt")) %>%
  filter(agency_id == 1) %>% # Filtering to only include King County Metro
  select(route_id, route_short_name) %>%
  mutate(rapid_ride = case_when(
    str_detect(route_short_name, "Line") ~ 1,
    TRUE ~ 0
  )) %>%
  replace_na(list(rapid_ride = 0)) %>%
  mutate(route_id = as.numeric(route_id),
         rapid_ride = as.factor(rapid_ride)) %>%
  select(route_id, rapid_ride, route_short_name)
```

```
# Joining
routes_shp <- kc_route_shp %>%
  left_join(routes,
            by = c("ROUTE_ID" = "route_id")) %>%
  st_transform(crs = 2285)

ylims <- c(184191.9, 271524.6)
xlims <- c(1250336, 1293480)
box_coords <- tibble(x = xlims, y = ylims) %>%
  st_as_sf(coords = c("x", "y")) %>%
  st_set_crs(2285)

bounding_box <- st_bbox(box_coords) %>% st_as_sfc()

routes_subset <- st_filter(routes_shp, bounding_box, .predicate = st_within)
routes_inbb <- routes_subset$ROUTE_ID

gtfs_realtime <- gtfs_realtime %>%
  filter(route_id %in% routes_inbb)
```

```
# Filtering out duplicates and future projections
gtfs_main <- gtfs_realtime %>%
  filter(projection == FALSE) %>%
  distinct(trip_id, stop_id, arrival_datetime, .keep_all = TRUE) %>%
  select(trip_id, route_id, stop_id, arrival_datetime, departure_datetime,
         pull_datetime) %>%
  mutate(trip_id = as.factor(trip_id),
         route_id = as.factor(route_id),
```

```
         stop_id = as.factor(stop_id)) %>%
  mutate(arrival_datetime = with_tz(arrival_datetime, "America/Los_Angeles"),
         departure_datetime = with_tz(departure_datetime, "America/Los_Angeles"),
         pull_datetime = with_tz(pull_datetime, "America/Los_Angeles"))

print(paste("Number of rows reduced from", nrow(gtfs_realtime), "to",
            nrow(gtfs_main)))
```

```
# Helper function for dealing with hh > 23 in schedule file
roll_over <- function(time_str) {
  # Split the time string into hours, minutes, seconds
  parts <- as.numeric(strsplit(time_str, ":")[[1]])
  total_seconds <- parts[1] * 3600 + parts[2] * 60 + parts[3]
  # Use modulo operator to get seconds within a day
  remainder <- total_seconds %% 86400
  # Convert remainder seconds back into hh:mm:ss format
  sprintf("%02d:%02d:%02d",
          remainder %/% 3600,
          (remainder %% 3600) %/% 60,
          remainder %% 60)
}
```

```
# Importing scheduled stop times
stop_times <- read_csv(here("Predictor Data Sets",
                            "gtfs-static-files/stop_times.txt")) %>%
  select(trip_id, arrival_time, departure_time, stop_id, stop_sequence,
         shape_dist_traveled) %>%
  mutate(trip_id = as.factor(trip_id),
         stop_id = as.factor(stop_id),
         arrival_time = as.character(arrival_time),
         departure_time = as.character(departure_time)) %>%
  rename(sched_arrival_time = arrival_time,
         sched_departure_time = departure_time) %>%
   mutate(
    sched_arrival_time = sapply(sched_arrival_time, roll_over),
    sched_departure_time = sapply(sched_departure_time, roll_over),
    sched_arrival_time = hms::as_hms(sched_arrival_time),
    sched_departure_time = hms::as_hms(sched_departure_time)
  ) %>%
  distinct(trip_id, stop_id, .keep_all = TRUE)

# Joining to main
gtfs_main_withdelays <- gtfs_main %>%
  left_join(stop_times, by = c("trip_id" = "trip_id",
```

```
                                "stop_id" = "stop_id")) %>%
  mutate(actual_arrival_time = hms::as_hms(format(with_tz(
    arrival_datetime, "America/Los_Angeles"), "%H:%M:%S")),
         actual_departure_time = hms::as_hms(format(with_tz(
           departure_datetime, "America/Los_Angeles"), "%H:%M:%S")),
         date = as.POSIXct(format(with_tz(
           arrival_datetime, "America/Los_Angeles"), "%Y-%m-%d"))) %>%
  mutate(arrival_delay = as.numeric(
    actual_arrival_time - sched_arrival_time)) %>%
  mutate(arrival_delay = ifelse(
    arrival_delay < 80000, arrival_delay,
    arrival_delay - 86400
    )) %>%
  select(date,
         trip_id,
         route_id,
         stop_id,
         sched_arrival_time,
         sched_departure_time,
         actual_arrival_time,
         actual_departure_time,
         arrival_delay,
         stop_sequence,
         shape_dist_traveled,
         pull_datetime)
```

```
# Loading data
congestion_temporal <- read_csv(here("Predictor Data Sets",
                                     "Traffic_Count_Studies_by_Hour_Bins-2.csv"))

# Converting times, setting up day/hour lookup
congestion_dayhour <- congestion_temporal %>%
  filter(TOTAL > 0) %>%
  mutate(datetime = as.POSIXct(ADD_DTTM,
                               format = "%m/%d/%Y %I:%M:%S %p",
                               tz = "America/Los_Angeles")) %>%
  filter(datetime > as.POSIXct("01-01-2015 00:00:00",
                               format = "%m-%d-%Y %H:%M:%S",
                               tz = "America/Los_Angeles")) %>%
  filter(datetime < as.POSIXct("01-31-2020 00:00:00",
                               format = "%m-%d-%Y %H:%M:%S",
                               tz = "America/Los_Angeles") |
           datetime > as.POSIXct("12-31-2021 23:59:59",
                               format = "%m-%d-%Y %H:%M:%S",
```

```
                                tz = "America/Los_Angeles")) %>%
  group_by(WEEKDAY) %>%
  summarize(HR01 = mean(HR01_TOTAL),
            HR02 = mean(HR02_TOTAL),
            HR03 = mean(HR03_TOTAL),
            HR04 = mean(HR04_TOTAL),
            HR05 = mean(HR05_TOTAL),
            HR06 = mean(HR06_TOTAL),
            HR07 = mean(HR07_TOTAL),
            HR08 = mean(HR08_TOTAL),
            HR09 = mean(HR09_TOTAL),
            HR10 = mean(HR10_TOTAL),
            HR11 = mean(HR11_TOTAL),
            HR12 = mean(HR12_TOTAL),
            HR13 = mean(HR13_TOTAL),
            HR14 = mean(HR14_TOTAL),
            HR15 = mean(HR15_TOTAL),
            HR16 = mean(HR16_TOTAL),
            HR17 = mean(HR17_TOTAL),
            HR18 = mean(HR18_TOTAL),
            HR19 = mean(HR19_TOTAL),
            HR20 = mean(HR20_TOTAL),
            HR21 = mean(HR21_TOTAL),
            HR22 = mean(HR22_TOTAL),
            HR23 = mean(HR23_TOTAL),
            HR24 = mean(HR24_TOTAL)) %>%
  mutate(WEEKDAY_NAME = case_when(
    WEEKDAY == 1 ~ "Monday",
    WEEKDAY == 2 ~ "Tuesday",
    WEEKDAY == 3 ~ "Wednesday",
    WEEKDAY == 4 ~ "Thursday",
    WEEKDAY == 5 ~ "Friday",
    WEEKDAY == 6 ~ "Saturday",
    WEEKDAY == 7 ~ "Sunday")) %>%
  mutate(AVG_VOL = select(., HR01:HR24) %>% rowMeans(na.rm = TRUE)) %>%
  relocate(WEEKDAY_NAME, .after = WEEKDAY) %>%
  relocate(AVG_VOL, .after = WEEKDAY_NAME)

# Setting up longer format
congestion_dh_longer <- congestion_dayhour %>%
  select(WEEKDAY_NAME, HR01:HR24) %>%
  pivot_longer(cols = -WEEKDAY_NAME,
               names_to = "HOUR",
               values_to = "avg_traffic_dayhour") %>%
```

```
  mutate(HOUR = as.numeric(gsub("[^0-9]","", HOUR)))

# Joining
gtfs_main_withcongestion <- gtfs_main_withdelays %>%
  mutate(WEEKDAY = weekdays(date),
         HR = (as.numeric(sched_arrival_time) %/% 3600) + 1) %>%
  left_join(select(.data = congestion_dayhour, WEEKDAY_NAME, AVG_VOL),
            by = c("WEEKDAY" = "WEEKDAY_NAME")) %>%
  left_join(congestion_dh_longer,
            by = c("WEEKDAY" = "WEEKDAY_NAME",
                   "HR" = "HOUR")) %>%
    rename("weekday" = "WEEKDAY",
         "hr" = "HR",
         "avg_traffic_day" = "AVG_VOL")

seq_standardized <- read.csv(here("Predictor Data Sets",
                                  "gtfs-static-files/stop_times.txt")) %>%
  select(trip_id, stop_sequence) %>%
  group_by(trip_id) %>%
  arrange(stop_sequence) %>%
  mutate(new_seq = row_number()) %>%
  ungroup() %>%
  arrange(trip_id, stop_sequence) %>%
  mutate(stop_sequence = as.numeric(stop_sequence),
         trip_id = as.factor(trip_id))

gtfs_main_final <- gtfs_main_withcongestion %>%
  inner_join(seq_standardized,
            by = c("stop_sequence" = "stop_sequence",
                   "trip_id" = "trip_id"))

gtfs_main_final <- gtfs_main_final[, c("date",
                                       "route_id",
                                       "trip_id",
                                       "stop_id",
                                       "sched_arrival_time",
                                       "sched_departure_time",
                                       "actual_arrival_time",
                                       "actual_departure_time",
                                       "arrival_delay",
                                       "stop_sequence",
                                       "new_seq",
                                       "shape_dist_traveled",
                                       "pull_datetime",
```

```
                                        "weekday",
                                        "hr",
                                        "avg_traffic_day",
                                        "avg_traffic_dayhour")]

gtfs_main_final
```

```
routes <- read.csv(here("Predictor Data Sets",
                        "gtfs-static-files/routes.txt")) %>%
  filter(agency_id == 1) %>% # Filtering to only include King County Metro
  select(route_id, route_short_name) %>%
  mutate(rapid_ride = case_when(
    str_detect(route_short_name, "Line") ~ 1,
    TRUE ~ 0
  )) %>%
  replace_na(list(rapid_ride = 0)) %>%
  mutate(route_id = as.factor(route_id),
         rapid_ride = as.factor(rapid_ride)) %>%
  select(route_id, rapid_ride)

gtfs_main_withrapidride <- gtfs_main_final %>%
  left_join(routes,
            by = "route_id")

stop_predictors <- gtfs_main_withrapidride %>%
  select(route_id, trip_id, stop_id, rapid_ride, arrival_delay,
         shape_dist_traveled, weekday, hr, avg_traffic_dayhour) %>%
  filter(!is.na(arrival_delay)) %>%
  mutate(weekday = as.factor(weekday))

write_csv(stop_predictors, "../predictor_tables/stop_predictors.csv")
```

```
set.seed(50)

# Train/Test Partition
subset_size <- 100000
subset_indices <- sample(seq_len(nrow(stop_predictors)), size = subset_size)

subset <- stop_predictors[subset_indices, ]
rest <- stop_predictors[-subset_indices, ]
```

```
# Loading trip data (directionality)
trips <- read.csv("../Predictor Data Sets/gtfs-static-files/trips.txt") %>%
```

```
  select(route_id, trip_id, direction_id, shape_id) %>%
  filter(route_id %in% routes_inbb)

# Loading spatial data for routes
shapes <- read.csv("../Predictor Data Sets/gtfs-static-files/shapes.txt") %>%
  arrange(shape_id, shape_pt_sequence) %>%
  st_as_sf(coords = c("shape_pt_lon", "shape_pt_lat"), crs = 4326) %>%
  group_by(shape_id) %>%
  summarise(do_union = FALSE) %>%
  st_cast("LINESTRING") %>%
  st_transform(2285)

# Loading stop sequence for each trip
stop_times <- read.csv("../Predictor Data Sets/gtfs-static-files/
                       stop_times.txt") %>%
  select(trip_id, stop_id, shape_dist_traveled, arrival_time) %>%
  mutate(shape_dist_traveled = shape_dist_traveled) %>%
  mutate(stop_id = as.character(stop_id),
         trip_id = as.character(trip_id)) %>%
  distinct(stop_id, trip_id,
           .keep_all = TRUE)

subset_directionality <- subset %>%
  left_join(trips,
            by = "trip_id")

# Return Direction-Conscious Linestring for TripIDs
subset_withshapes <- subset_directionality %>%
  left_join(shapes,
             by = c("shape_id" = "shape_id")) %>%
  mutate(geometry = case_when(
    direction_id == 1 ~ st_reverse(geometry),
    TRUE ~ geometry),
    trip_id = as.character(trip_id)
  ) %>%
  st_as_sf() %>%
  st_set_crs(st_crs(shapes))

# Clipping route lines
subset_clipped <- subset_withshapes %>%
  mutate(trip_id = as.character(trip_id),
         shape_dist_traveled = case_when(
           shape_dist_traveled == 0 ~ 1,
           TRUE ~ shape_dist_traveled
```

```
        )) %>%
  select(!route_id.y) %>%
  rename("route_id" = "route_id.x") %>%
  mutate(
    total_length = as.numeric(st_length(geometry)),
    # Normalize distance to [0,1] fraction
    to_fraction = pmin(shape_dist_traveled / total_length, 1)
  ) %>%
  rowwise() %>%
  mutate(
    geometry = lwgeom::st_linesubstring(
      geometry,
      from = 0,
      to = to_fraction,
      normalize = FALSE
    )
  ) %>%
  ungroup() %>%
  st_as_sf() %>%
  st_set_crs(st_crs(2285))

get_weighted_traffic_average <- function(geometry) {
  buffered_route <- st_buffer(geometry, dist = 1)
  intersection <- st_intersection(congestion_spatial, buffered_route)
  intersection_line <- st_collection_extract(intersection, "LINESTRING")

  # Note this is avg of traffic data we have, so not all routes
  # have complete coverage
  processed_intersection <- intersection_line %>%
    # Calculate length of each segment
    mutate(length = units::set_units(
      st_length(geometry), "ft", mode = "standard")) %>%
    # Filtering out any segment less than 5ft in length to remove noise
    filter(length > units::set_units(
      5, "ft", mode = "standard"))

  total_length <- sum(processed_intersection$length)

  processed_intersection <- processed_intersection %>%
    mutate(wgt_traffic = AWDT * (length / total_length))

  segment_traffic <- as.numeric(sum(processed_intersection$wgt_traffic))
  return(segment_traffic)
}
```

```r
# CRS is NAD83
congestion_spatial <- st_read(here("Predictor Data Sets",
                                   "2018_Traffic_Flow_Counts-shp",
                                   "2018_Traffic_Flow_Counts.shp")) %>%
  select(AWDT, geometry) %>%
  st_transform(crs = 2285)

subset_spatialcongestion <- subset_clipped %>%
  rowwise() %>%
  mutate(spatial_congestion = get_weighted_traffic_average(geometry)) %>%
  ungroup()

subset_fixed <- subset_spatialcongestion %>%
  left_join(trips %>% select(trip_id, route_id),
            by = "trip_id") %>%
  select(!route_id.x) %>%
  rename("route_id" = "route_id.y") %>%
  mutate(route_id = as.factor)

# Ridership
get_weighted_acs <- function(route_id) {
   route_shape <- routes_subset %>%
     filter(ROUTE_ID == route_id)
  # approximate 0.5 mile buffer (in feet)
  buffered_route <- st_buffer(route_shape, dist = 2640)

  # Filter ACS polygons to include only ones with over 50% within buffer
   blocks_filtered <- kcacs_blocks %>%
     filter(lengths(st_intersects(., buffered_route)) > 0) %>%
     rowwise() %>%
     mutate(
       inter_geom = list(st_intersection(geometry, buffered_route)),
       inter_area = {
         ig <- inter_geom[]
         if(length(ig) == 0 || all(st_is_empty(ig))) {
           0
         } else {
           sum(st_area(ig))
         }
       },
       total_area = st_area(geometry),
       overlap_ratio = as.numeric(inter_area / total_area)
       ) %>%
     ungroup() %>%
```

```
    filter(overlap_ratio >= 0.5)

  # Weighted average
  total_population <- sum(blocks_filtered$tot_popE)
  total_buffer_area <- sum(st_area(blocks_filtered$geometry))

  blocks_filtered <- blocks_filtered %>%
    mutate(wgt_ridership = transp_mthd_public_perc * (
      tot_popE/total_population),
           wgt_percwhite = white_perc * (tot_popE/total_population),
           wgt_medHHI = median_HHI * (tot_popE/total_population))

  list(
    pop_density = total_population/total_buffer_area,
    route_ridership = as.double(sum(blocks_filtered$wgt_ridership)),
    perc_white = as.double(sum(blocks_filtered$wgt_percwhite)),
    median_hhi = as.double(sum(blocks_filtered$wgt_medHHI))
  )
}
```

```
# Using keyring package to keep my API key hidden
tidycensus_api_key <- key_get(service = "tidycensus_API",
                              username = "my_tidycensus")
census_api_key(tidycensus_api_key)

ACSlist <- load_variables(2022, "acs5", cache = TRUE)

# Projection is NAD83(!!)
kingcounty_acs_blocks <- get_acs(state = "WA",
                                 county = "King",
                                 geography = "block group",
                                 variables = c(tot_pop = "B01003_001",
                                               transp_basetotal = "B08134_001",
                                               transp_mthd_public = "B08134_061",
                                               race_base = "B02001_001",
                                               race_white = "B02001_002",
                                               median_HHI = "B19013_001"),
                                 geometry = TRUE,
                                 keep_geo_vars = TRUE,
                                 year = 2023,
                                 output = "wide") %>%
  filter(ALAND != 0) %>% # Filter tracts that are 100% water
  mutate(GEOID = as.double(GEOID))
```

```
kcacs_blocks <- kingcounty_acs_blocks %>%
  mutate(ALAND_miles = ALAND/2589988) %>% # Converting sq meters to sq miles
  mutate(transp_mthd_public_perc = transp_mthd_publicE / transp_basetotalE,
         pop_density = tot_popE / ALAND_miles,
         white_perc = race_whiteE / race_baseE,
         median_HHI = median_HHIE) %>%
  filter(!is.na(median_HHI)) %>%
  select(tot_popE,
         pop_density,
         transp_mthd_public_perc,
         white_perc,
         median_HHI,
         geometry) %>%
  st_transform(crs = 2285)

route_demos <- routes_subset %>%
  rowwise() %>%
  mutate(
     acs_data = list(get_weighted_acs(ROUTE_ID))
   ) %>%
  mutate(
    pop_density = acs_data$pop_density, # POP PER SQUARE FOOT
    route_ridership = acs_data$route_ridership,
    perc_white = acs_data$perc_white,
    median_hhi = acs_data$median_hhi
  ) %>%
  ungroup() %>%
  select(!acs_data, !geometry) %>%
  filter(!is.na(ROUTE_ID))

route_demos <- route_demos %>%
  select(ROUTE_ID, pop_density, route_ridership, perc_white, median_hhi) %>%
  rename("route_id" = "ROUTE_ID") %>%
  mutate(pop_density = as.double(pop_density),
         route_id = as.factor(route_id))

subset_withacs <- subset_fixed %>%
  select(!geometry) %>%
  tibble() %>%
  left_join(route_demos,
            by = "route_id")
```

```r
# Standardize function
standardize <- function(x) {
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
}

subset_standardized <- subset_withacs %>%
  select(route_id,
         stop_id,
         trip_id,
         rapid_ride,
         arrival_delay,
         shape_dist_traveled,
         avg_traffic_dayhour,
         spatial_congestion,
         pop_density,
         route_ridership,
         perc_white,
         median_hhi,
         weekday,
         hr) %>%
  mutate(across(c("shape_dist_traveled",
                  "avg_traffic_dayhour",
                  "spatial_congestion",
                  "pop_density",
                  "route_ridership",
                  "perc_white",
                  "median_hhi"),
                standardize)) %>%
  mutate(abs_dev = abs(arrival_delay))
```

```r
# Setting some definitions
weekdays <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
weekends <- c("Saturday", "Sunday")
peak <- c(6, 7, 8, 9, 16, 17, 18, 19) # weekdays only
non_peak <- c(1, 2, 3, 4, 5, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 24)

data_final <- subset_standardized %>%
  mutate(g_weekend = case_when(
    weekday %in% weekdays ~ 0,
    weekday %in% weekends ~ 1
  ),
  g_peak = case_when(
    weekday %in% weekdays & hr %in% peak ~ 1,
    TRUE ~ 0
```

```
  )) %>%
  dplyr::select(route_id,
         stop_id,
         trip_id,
         rapid_ride,
         arrival_delay,
         abs_dev,
         shape_dist_traveled,
         avg_traffic_dayhour,
         spatial_congestion,
         pop_density,
         route_ridership,
         perc_white,
         median_hhi,
         weekday,
         hr,
         g_peak,
         g_weekend) %>%
  mutate(
    weekday = case_when(
      weekday == "Monday" ~ 1,
      weekday == "Tuesday" ~ 2,
      weekday == "Wednesday" ~ 3,
      weekday == "Thursday" ~ 4,
      weekday == "Friday" ~ 5,
      weekday == "Saturday" ~ 6,
      weekday == "Sunday" ~ 7
    )) %>%
  mutate(rapid_ride = as.factor(rapid_ride),
         weekday = as.numeric(weekday),
         hr = as.numeric(hr)) %>%
  rename("g_routeid" = "route_id",
         "g_weekday" = "weekday",
         "g_hr" = "hr")

write_csv(data_final, "../predictor_tables/final_data.csv")
```

```
# Train/Test Split
set.seed(50)

data_final <- read_csv("../predictor_tables/final_data.csv") %>%
  mutate(
    abs_dev = ifelse(
    abs_dev < 80000, abs_dev,
```

```
    abs(abs_dev - 86400)
    ))

# Train/Test Partition
smp_size <- floor(0.5 * nrow(data_final))
train_indices <- sample(seq_len(nrow(data_final)), size = smp_size)

train <- data_final[train_indices, ]
test <- data_final[-train_indices, ]

write_csv(train, "../cloud-scripts/train_data.csv")
write_csv(test, "../cloud-scripts/test_data.csv")
```

```
# <---------- LINEAR REGRESSION MODELING ---------->

train_df <- read_csv(here("cloud-scripts","partial-pooling","train_data.csv"))
test_df <- read_csv(here("cloud-scripts","partial-pooling","test_data.csv"))
df_combo <- rbind(train_df, test_df[test_df$rapid_ride == 1, ])

# A simple comparison of means
fit_binary <- stan_glm(abs_dev ~ rapid_ride,
                       data = df_combo,
                       refresh = FALSE,
                       cores = 4)

# A model with interactions between related variables and
# allow treatment effects to vary by group
fit_int <- stan_glm(abs_dev ~ rapid_ride +
                        shape_dist_traveled +
                        avg_traffic_dayhour +
                        spatial_congestion +
                        pop_density +
                        route_ridership +
                        perc_white +
                        median_hhi +
                        spatial_congestion:avg_traffic_dayhour +
                        shape_dist_traveled:avg_traffic_dayhour +
                        shape_dist_traveled:spatial_congestion +
                        rapid_ride:factor(g_weekday) +
                        rapid_ride:factor(g_hr) +
                        rapid_ride:spatial_congestion +
                        rapid_ride:route_ridership +
                        rapid_ride:avg_traffic_dayhour,
                      data = df_combo,
```

```
                        refresh = FALSE,
                        cores = 4)

# Table of Coefficients
print(fit_binary, digits = 5)
print(fit_int, digits = 5)

save(fit_binary, file = "../models/fit_binary.RData")
save(fit_int, file = "../models/fit_int.RData")
```

```
# <---------- BART MODELING (USED GOOGLE CLOUD & DOCKER) ---------->

# Vanilla BART

# Loading libraries
library(dbarts)
library(bartCause)
library(readr)
library(processx)
library(dplyr)

print(installed.packages())

# Get environment variables set by Vertex AI
model_dir <- "gs://bus-delay-modeling-stan4bart-models/models/"
print(paste("AIP_MODEL_DIR:", model_dir))

#remove trailing slashes.
model_dir <- gsub("/+$", "", model_dir)

if (model_dir == "") {
  model_dir <- "models"  # Fallback for local testing
  dir.create(model_dir, recursive = TRUE, showWarnings = FALSE)
} else {
  #create local model folder to save model to before copying to gs.
  dir.create("local_model_dir", recursive = TRUE, showWarnings = FALSE)
}

# Loading and preparing data
df <- read_csv("train_data.csv")
df_test <- read_csv("test_data.csv")

df_combo <- rbind(df, df_test[df_test$rapid_ride == 1, ])
```

```r
df_cf <- df_combo
df_cf$rapid_ride <- ifelse(df_cf$rapid_ride == 1, 0, 1)

# Training the stan4bart model
fit <- bart2(abs_dev ~
                rapid_ride +
                shape_dist_traveled +
                avg_traffic_dayhour +
                spatial_congestion +
                pop_density +
                route_ridership +
                perc_white +
                median_hhi +
                g_weekday +
                g_hr,
              data = df_combo,
              test = df_cf,
              keepTrees = TRUE,
              seed = 50
              )

# Save the model to a local directory.
save(fit, file = file.path("local_model_dir",
                           "vanillabart_rapidride_model_cf_combo.RData"))

# Copy the model to Google Cloud Storage using gsutil.
if(model_dir != "models"){
  local_file_path <- file.path("local_model_dir",
                               "vanillabart_rapidride_model_cf_combo.RData")
  gs_destination <- file.path(model_dir,
                              "vanillabart_rapidride_model_cf_combo.RData")

  # Run gsutil cp command
  result <- processx::run("gsutil", c("cp", local_file_path, gs_destination))

  if (result$status == 0) {
    cat("Model training completed and saved to", gs_destination, "\n")
  } else {
    cat("Error copying model to Google Cloud Storage.\n")
    cat("gsutil output:\n", rawToChar(result$stdout), "\n")
    cat("gsutil error:\n", rawToChar(result$stderr), "\n")
  }

} else {
```

```
  cat("Model training completed and saved locally to", file.path(
    "local_model_dir", "vanillabart_rapidride_model_cf_combo.RData"), "\n")
}
```

```
# <---------- FINAL ANALYSIS AND VISUALIZATIONS ---------->
library(tidyverse)
library(ggplot2)
library(here)
library(sf)
library(tmap)
library(rstanarm)
library(bartCause)
library(knitr)
library(skimr)
library(corrplot)
library(posterior)
library(bayesplot)
library(dbarts)
library(broom.mixed)
library(kableExtra)
library(extrafont)

extrafont::loadfonts(device = "pdf", quiet = TRUE)
font_import()

# Loading combined final dataset
data <- read_csv(here("predictor_tables","final_data.csv")) %>%
  mutate(
    g_routeid = as.factor(g_routeid),
    stop_id = as.factor(stop_id),
    trip_id = as.factor(trip_id),
    rapid_ride = as.numeric(rapid_ride),
    arrival_delay = as.numeric(arrival_delay),
    abs_dev = as.numeric(abs_dev),
    shape_dist_traveled = as.numeric(shape_dist_traveled),
    avg_traffic_dayhour = as.numeric(avg_traffic_dayhour),
    spatial_congestion = as.numeric(spatial_congestion),
    pop_density = as.numeric(pop_density),
    route_ridership = as.numeric(route_ridership),
    perc_white = as.numeric(perc_white),
    median_hhi = as.numeric(median_hhi),
    g_weekday = as.numeric(g_weekday),
    g_hr = as.numeric(g_hr),
    g_peak = as.factor(g_peak),
```

```
    g_weekend = as.factor(g_weekend)
  ) %>%
  mutate(
    abs_dev = ifelse(
    abs_dev < 80000, abs_dev,
    abs(abs_dev - 86400)
    ))

train_df <- read_csv(here("cloud-scripts","partial-pooling","train_data.csv"))
test_df <- read_csv(here("cloud-scripts","partial-pooling","test_data.csv"))
df_combo <- rbind(train_df, test_df[test_df$rapid_ride == 1, ])

load("models/fit_binary.RData")
load("models/fit_int.RData")
load("models/models-vanillabart_rapidride_model_cf_combo.RData")
fit_cf_combo <- fit
load("models/models-vanillabart_rapidride_model.RData")
```

```
# Loading Data
routes_inscope <- unique(data$g_routeid)
stops_inscope <- unique(data$stop_id)
routes_rapidride <- data %>%
  filter(rapid_ride == 1)
routes_rapidride <- unique(routes_rapidride$g_routeid)

routes_shp <- st_read(here("Predictor Data Sets",
          "KCMetro_Transit_Lines",
          "Transit_Routes_for_King_County_Metro___transitroute_line.shp")) %>%
  filter(ROUTE_ID %in% routes_inscope)

routes_rapidride_shp <- routes_shp %>%
  filter(ROUTE_ID %in% routes_rapidride)

stops_shp <- st_read(here("Predictor Data Sets",
                "KCMetro_Transit_Stops",
                "Transit_Stops_for_King_County_Metro___transitstop_point.shp"))
```

```
# Mapping
route_map <- tm_shape(routes_shp) + tm_lines(col = "#1D7D7A", lwd = 1) +
  tm_shape(routes_rapidride_shp) + tm_lines(col = "#D71D24", lwd = 2) +
  tm_basemap("CartoDB.PositronNoLabels") +
  tm_add_legend(type = "line",
                col = c("#1D7D7A", "#D71D24"),
                labels = c("Standard Bus", "RapidRide"),
```

```
                title = "Route Type",
                lwd = 2) +
  tm_layout(fontfamily = "Times New Roman",
            legend.text.size = 0.8,
            legend.title.size = 1)
tmap_save(route_map, filename = "route_map.png")
```

```
# Creating descriptive statistics table
data_descriptives = df_combo %>%
  select(
    rapid_ride,
    abs_dev,
    shape_dist_traveled,
    avg_traffic_dayhour,
    spatial_congestion,
    pop_density,
    route_ridership,
    perc_white,
    median_hhi,
    g_weekday,
    g_hr
  )

overall_descriptives <- skim(data_descriptives) %>%
  select(skim_variable,
         numeric.mean,
         numeric.sd,
         numeric.p0,
         numeric.p25,
         numeric.p50,
         numeric.p75,
         numeric.p100) %>%
  rename(
    "Variable" = "skim_variable",
    "Mean" = "numeric.mean",
    "Std Dev" = "numeric.sd",
    "Min" = "numeric.p0",
    "25%" = "numeric.p25",
    "Median" = "numeric.p50",
    "75%" = "numeric.p75",
    "Max" = "numeric.p100",
  ) %>%
  mutate(Variable = recode(Variable,
                           "rapid_ride" = "RapidRide",
```

```
                            "abs_dev" = "Absolute Deviation",
                            "shape_dist_traveled" = "Distance Traveled",
                            "avg_traffic_dayhour" = "Traffic (Day/Hour)",
                            "spatial_congestion" = "Traffic (Location)",
                            "pop_density" = "Population Density",
                            "route_ridership" = "Route Ridership",
                            "perc_white" = "Percentage White",
                            "median_hhi" = "Median HHI",
                            "g_weekday" = "Weekday",
                            "g_hr" = "Hour"))
```

```
# Assessing balance and overlap for causal inference
# From https://github.com/gperrett/stan4bart-study/blob/master/get_balance.R
# Linked in Dorie et al 2022
get_balance <- function(rawdata, treat,estimand="ATT"){
  if(missing(rawdata)) stop("rawdata is required")
  if(missing(treat)) stop("treatment vector (treat) is required")
  cat("Balance diagnostics assume that the estimand is the",estimand,"\n")
  #
  #raw.dat <- data.frame(rawdata, treat = treat)
  covnames <- colnames(rawdata)
  if (is.null(covnames)){
    cat("No covariate names provided.  Generic names will be generated.")
    covnames = paste("v",c(1:ncol(rawdata)),sep="")
  }
  K <- length(covnames)
  diff.means <- matrix(NA, K, 5)
  var.t <- numeric(K)
  var.c <- numeric(K)
  std.denom <- numeric(K)
  binary <- rep(1,K)

  for (i in 1:K) {
    # separate means by group
    diff.means[i, 1] <- mean(rawdata[treat==1, i])
    diff.means[i, 2] <- mean(rawdata[treat==0, i])
    # separate variances by group == only used as input to calculations below
    var.t[i] <- var(rawdata[(treat == 1), i])
    var.c[i] <- var(rawdata[(treat == 0), i])
    # denominator in standardized difference calculations
    if(estimand=="ATE"){std.denom[i] <- sqrt((var.t[i]+var.c[i])/2)}
    else{
      std.denom[i] <- ifelse(estimand=="ATT",sqrt(var.t[i]),sqrt(var.c[i]))
    }
```

```
    # difference in means
    diff.means[i, 3] <- diff.means[i, 1] - diff.means[i, 2]
    # standardized difference in means (sign intact)
    diff.means[i, 4] <- abs(diff.means[i, 3]/std.denom[i])
    if(length(unique(rawdata[,covnames[i]]))>2){
      binary[i] = 0
      diff.means[i, 5] <- sqrt(var.c[i]/var.t[i])
    }
  }

  dimnames(diff.means) <- list(covnames, c("Treat", "Control", "Difference",
                                           "abs.std.diff", "Ratio"))
  return(diff.means)
}

# Setting up data (covariates in a matrix, treatment vector)
# Removed factor vars
X <- as.matrix(data_descriptives %>% select(!rapid_ride))
y <- data_descriptives$rapid_ride

# Running the function
balance_table <- get_balance(rawdata = X, treat = y, estimand = "ATT")
balance_table <- as_tibble(balance_table) %>%
  dplyr::select(Treat, Control, Difference, Ratio)
rownames(balance_table) <- c("Absolute Deviation", "Distance Traveled",
                             "Traffic (Day/Hour)", "Traffic (Location)",
                             "Population Density", "Route Ridership",
                             "Percentage White", "Median HHI",
                             "Weekday", "Hour")
```

```
# Correlation table
corr_table <- data_descriptives %>%
  rename("RapidRide" = "rapid_ride",
         "Absolute Deviation" = "abs_dev",
         "Distance Traveled" = "shape_dist_traveled",
         "Traffic (Day/Hour)" = "avg_traffic_dayhour",
         "Traffic (Location)" = "spatial_congestion",
         "Population Density" = "pop_density",
         "Route Ridership" = "route_ridership",
         "Percentage White" = "perc_white",
         "Median HHI" = "median_hhi",
         "Weekday" = "g_weekday",
         "Hour" = "g_hr")
corr <- cor(corr_table)
```

```r
actuals <- test_df$abs_dev

# Coefficients for Linear Regressions
binary_tidy <- tidy(fit_binary)
int_tidy <- tidy(fit_int)

merged_df <- full_join(
  binary_tidy %>% select(term, estimate, std.error),
  int_tidy %>% select(term, estimate, std.error),
  by = "term",
  suffix = c("_model1", "_model2")
)

# RMSE
binary_pred <- predict(fit_binary, newdata = test_df)
int_pred <- predict(fit_int, newdata = test_df)
bart_pred <- fitted(fit, type = "ev", sample = "test")

binary_rmse <- sqrt(mean((binary_pred - actuals)^2))
int_rmse <- sqrt(mean((int_pred - actuals)^2))
bart_rmse <- sqrt(mean((bart_pred - actuals)^2))

rmse_df <- tibble(
  Model = c("Binary Linear", "Multivariate Linear",
            "Bayesian Additive Regression Trees"),
  RMSE = c(binary_rmse, int_rmse, bart_rmse)
)
```

```r
# Assessing R-Hat for BART model
sigma_draws <- as_draws_array(fit_cf_combo$sigma)
rhat_values <- round(posterior::rhat(sigma_draws), 2)

df_combo <- rbind(train_df, test_df[test_df$rapid_ride == 1, ])
rr_indices <- df_combo$rapid_ride == 1

# BART SATT
factual_pred <- extract(fit_cf_combo, type = "ev", sample = "train")
counterfactual_pred <- extract(fit_cf_combo, type = "ev", sample = "test")

treated_factual_pred <- factual_pred[, rr_indices]
treated_counterfactual_pred <- counterfactual_pred[, rr_indices]

ind_effects <- treated_factual_pred - treated_counterfactual_pred
```

```r
satt_dist <- rowMeans(ind_effects)

satt_est <- median(satt_dist)
satt_ci <- quantile(satt_dist, probs = c(0.025, 0.975))

satt_df <- tibble(
  Estimate = round(satt_est, 2),
  Lower95 = round(satt_ci[1], 2),
  Upper95 = round(satt_ci[2], 2)
  )
```

```r
# Multivariate SATT
df_combo_rr_f <- df_combo[df_combo$rapid_ride == 1, ]
df_combo_rr_cf <- df_combo_rr_f
df_combo_rr_cf$rapid_ride <- 0

int_factual_pred <- posterior_predict(fit_int, newdata = df_combo_rr_f)
int_counterfactual_pred <- posterior_predict(fit_int, newdata = df_combo_rr_cf)

int_ind_effects <- int_factual_pred - int_counterfactual_pred

int_satt_dist <- rowMeans(int_ind_effects)

int_satt_est <- median(int_satt_dist)
int_satt_ci <- quantile(int_satt_dist, probs = c(0.025, 0.975))

satt_df <- tibble(
  Estimate = c(round(satt_est, 2), round(int_satt_est, 2)),
  Lower95 = c(round(satt_ci[1], 2), round(int_satt_ci[1], 2)),
  Upper95 = c(round(satt_ci[2], 2), round(int_satt_ci[2], 2))
  )
```

```r
# Histplot
satt_dist_df <- tibble(satt_dist)
int_satt_dist_sample <- sample(int_satt_dist, size = 2000)
satt_histplot <- ggplot(data = satt_dist_df, aes(x = satt_dist)) +
  geom_density(aes(fill = "BART"), alpha = 0.75) +
  geom_density(aes(x = int_satt_dist_sample, fill = "Multivariate Linear"),
               alpha = 0.75) +
  geom_vline(xintercept = satt_est, linetype = "dashed",
             color = "#2596be", linewidth = 0.75) +
  geom_vline(xintercept = int_satt_est, linetype = "dashed",
             color = "#fd527e", linewidth = 0.75) +
  theme_minimal() +
```

```
  annotate("text",
           x = satt_est - 3,
           y = 0.06,
           label = paste("Median:", round(satt_est, 2)),
           color = "#2596be",
           hjust = 1) +
  annotate("text",
           x = int_satt_est - 4,
           y = 0.07,
           label = paste("Median:", round(int_satt_est, 2)),
           color = "#fd527e",
           hjust = 1) +
  labs(title = "Distribution of SATT Estimates",
       # subtitle = paste("Median BART SATT =", round(satt_est, 2)),
       x = "SATT (seconds)",
       y = "Density") +
  scale_fill_manual(name = "Model",
                    values = c("BART" = "lightblue",
                               "Multivariate Linear" = "pink")) +
  theme(
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5)
  )
```

```
# Calculate SATT by hour
n_groups <- 24
hr_codes <- seq(from = 1, to = 24)
hr_means <- rep(NA, n_groups)
hr_upper <- rep(NA, n_groups)
hr_lower <- rep(NA, n_groups)

for (hr in 1:n_groups) {
  indices <- df_combo$g_hr[df_combo$rapid_ride == 1]

  y1_pred_hr <- treated_factual_pred[, indices == hr]
  y0_pred_hr <- treated_counterfactual_pred[, indices == hr]

  if (ncol(y1_pred_hr) > 0) {

  ind_effects_hr <- y1_pred_hr - y0_pred_hr
  satt_dist_hr <- rowMeans(ind_effects_hr)

  satt_est_hr <- mean(satt_dist_hr)
  satt_ci_upper <- quantile(satt_dist_hr, probs = c(0.975))
```

```
  satt_ci_lower <- quantile(satt_dist_hr, probs = c(0.025))

  hr_means[hr] <- satt_est_hr
  hr_upper[hr] <- satt_ci_upper
  hr_lower[hr] <- satt_ci_lower
  }
  else {
    hr_means[hr] <- 0
    hr_upper[hr] <- 0
    hr_lower[hr] <- 0
  }
}

hr_ests <- tibble(hr_codes, hr_means, hr_upper, hr_lower)

# Multivariate for comparison
# Calculate SATT by day
n_groups <- 24
int_hr_codes <- seq(from = 1, to = 24)
int_hr_means <- rep(NA, n_groups)
int_hr_upper <- rep(NA, n_groups)
int_hr_lower <- rep(NA, n_groups)
n_draws <- 1000


for (hr in 1:n_groups) {
  hrtreated_df <- df_combo %>%
    filter(rapid_ride == 1) %>%
    filter(g_hr == hr)

  if (nrow(hrtreated_df > 0)) {
    hrcounter_df <- hrtreated_df
    hrcounter_df$rapid_ride <- 0

    y1_pred_hr <- posterior_epred(fit_int, newdata = hrtreated_df)
    y0_pred_hr <- posterior_epred(fit_int, newdata = hrcounter_df)
    ind_effects_hr <- y1_pred_hr - y0_pred_hr
    satt_dist_hr <- rowMeans(ind_effects_hr)

    satt_est_hr <- mean(satt_dist_hr)
    satt_ci_upper <- quantile(satt_dist_hr, probs = c(0.975))
    satt_ci_lower <- quantile(satt_dist_hr, probs = c(0.025))

    int_hr_means[hr] <- satt_est_hr
```

```
    int_hr_upper[hr] <- satt_ci_upper
    int_hr_lower[hr] <- satt_ci_lower
  }
  else {
    int_hr_means[hr] <- 0
    int_hr_upper[hr] <- 0
    int_hr_lower[hr] <- 0
  }
}

int_hr_ests <- tibble(int_hr_codes, int_hr_means, int_hr_upper, int_hr_lower)
```

```
hourly_comparison_graph <- ggplot(data=hr_ests) +
  # BART
  geom_point(aes(x=hr_codes, y=hr_means,
                     color = "BART")) +
  geom_errorbar(aes(ymin=hr_lower,
                     ymax=hr_upper,
                     x=hr_codes,
                     color = "BART"), alpha=1, width = 0) +
  # MULTIVARIATE
  geom_point(aes(x=int_hr_codes, y=int_hr_means,
                     color = "Multivariate")) +
  geom_errorbar(aes(ymin=int_hr_lower,
                     ymax=int_hr_upper,
                     x=int_hr_codes,
                     color = "Multivariate"), alpha=1, width = 0) +
  scale_color_manual(name="",
                     values=c("Multivariate"="#fd527e", "BART" = "#2596be")) +
  geom_hline(yintercept = 0, linetype='dashed', col = 'gray')+
  theme_minimal() +
  scale_x_continuous(breaks = seq(min(hr_ests$hr_codes),
                                  max(hr_ests$hr_codes), by = 1)) +
  scale_y_continuous(breaks = c(-150, -75, 0, 75, 150)) +
  labs(title="Treatment Effect by Hour of the Day",
       x="Hour of the Day",
       y="SATT (seconds)")+
  theme(axis.title=element_text(size=10),
        axis.text.y=element_text(size=10),
        axis.text.x=element_text(angle=90,size=8, vjust=0.3),
        legend.title=element_text(size=10),
        legend.text=element_text(size=10))
```