SCC Client Report
An Analysis Plan for Evaluating Location of a New Vance County EMS Station

October 12, 2023

1 Background

The Vance County EMS system currently consists of four ambulances and two call stations, one located in the Southern district and one in the Central district. This setup has led to residents in the Northern district being underserved; average response times are noticeably higher than the other two districts, with delays being significant enough to change outcomes in potentially fatal scenarios. Coupling this with a rising demand for EMS services from the North, you are looking to build a northern station in one of two proposed locations; your past call data is being used to inform your decision. The available data originates from your provided written charts. The dates of each trip have been manipulated into a mock dataset to protect patients' privacy. Each row represents a single trip with details such as the dispatch station, the coordinates of the patient's address, various logged times (dispatch, en route, arrival to site, leaving site, arriving at hospital, and clear time), and other details about the trip. You want to use this information to analyze travel times, the associated system load, and assess different station location and vehicle allocation scenarios. You have asked us to help answer the two following research questions: 1) which of the proposed North district station locations would better serve the community?, and 2) how should the 4 available ambulances be allocated to best serve the community?

2 Recommendations

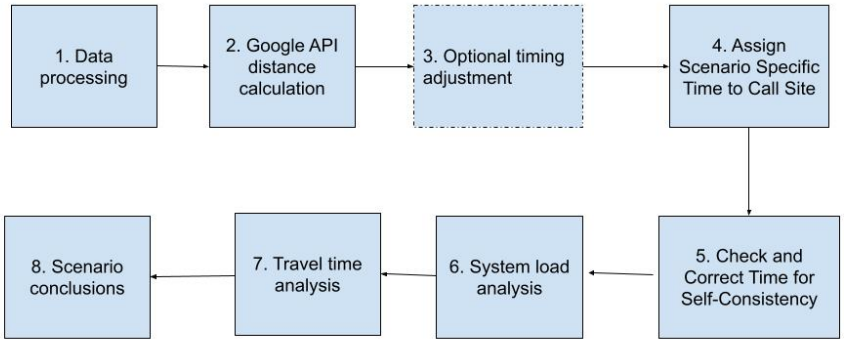
2.1 Overview

Scenarios	S0(Current)	S1	S2	S3	S4
Far North	0	0	1	0	1
Near North	0	1	0	1	0
South	3	3	3	2	2
Central	1	0	0	1	1

We came up with 5 reasonable scenarios to perform our analysis on. We have 4 total ambulances with 4 different stations- Near North, Far North, Central, and South.

Our recommendations rely heavily on using caller coordinates and Google's Distance Matrix API to generate additional travel time estimation features from the possible stations, then constructing

a model using these data to estimate both ambulance travel time and load on the emergency system for each scenario of interest. The proposed analysis pipeline in Fig. 1 begins with data quality checks, including the identification of out-of-county and other atypical trips, followed by the utilization of the Google Distance Matrix API to estimate travel times from all potential station locations. At this point, there is the possibility of constructing additional models to adjust Google API estimates to be more realistic in emergency and non-emergency EMS situations. The next step is to use estimated travel times to construct different real world ambulance dispatch scenarios. The next step is to verify if the timing data is consistent and correct. There are several potential measures of load analysis on the per-call basis which then can be modeled with the aforementioned scenarios in a linear mixed model. The travel time analysis focuses on modeling the dependency of estimated travel times on scenarios, similarly using a linear mixed model. From the results of both these analyses, a conclusion on which scenario to recommend to the county may be formed.



2.2 Preliminary Data Processing and Computations

Prior to proceeding with the analysis, we recommend that you familiarize yourself with the data, check for unexpected features and correct any errors you find. For example:

- Verify all trips within the data fall within Vance county boundaries.
- Decide whether to include calls with atypical features, for example calls where latitude and/or longitude is not within Vance County boundaries.
- Determine how frequently expected dispatch rules are followed, for example all calls are addressed by an ambulance from the closest station if one is available.

- Familiarize yourself with the temporal and spatial patterns in the data, for example:

  - Average distance and travel time from a call to each station.
  - The call volume for each region (North, Central, and South).
  - For each region, call volume and average travel time by day of the week, hour of the day and if attainable, month of the year.

Effort spent upfront identifying and correcting data errors and developing a comprehensive understanding of the data will improve the quality of subsequent analyses.

The Google Distance Matrix API is a service of the Google Maps platform that accepts multiple origin and destination locations and returns a matrix of expected travel times and distances. The API is flexible, and offers a variety of features that a user can incorporate into their estimated travel time for a given route. One key feature that we make use of in our recommendation and we will highlight is the `traffic_model` parameter: a user can input `optimistic`, `pessimistic`, or `best_guess` which corresponds to the API's method of estimating travel time for the given `traffic_model` scenarios. Read more about the Distance Matrix API here.

The steps to get started can be found by following the table of contents at the link above. Briefly, this will include setting up your Good Cloud project, creating an API key and then calling the Google Distance Matrix API with the `R mapsi` package, which we will describe further below.

The Distance Matrix API allows for usage of built-in parameters to better resemble real-life scenarios. Various inputs include arrays of origin and destination locations, method of travel (walking, transit, biking, driving), as well as specific transit and driving options. We will focus on the `driving_options` parameter. These driving options contain specifics regarding traffic models: best guess, pessimistic, and optimistic. These different traffic models slightly manipulate the times of transportation by adjusting the `duration_in_traffic` metric using historical traffic conditions as well as live traffic information. This is valuable in the context of this project because it allows for better alignment of estimated travel times with both emergency (lights on) and non-emergency scenarios. While the google distance matrix API is helpful for estimating travel times, it only can estimate travel times in the future. For this project, this implies that the dates of the trips in the dataset have to be changed to a date in the future. This could lead to trips being estimated on a different day of the week than they might've taken place and possible distortion from accurate traffic scenarios.

Be mindful of the limited bandwidth available when making calls/requests to the API. Conducting a comprehensive, full analysis will likely require numerous individual requests separated by brief time delays. In all the scenarios used (Best Guess, Pessimistic, Optimistic, Green Light), you will observe the command `Sys.sleep(0.5)`. This command introduces a brief pause at the end of each iteration within the for-loop, allowing the API sufficient processing time. Consequently, it may be necessary for this code to run overnight due to the extended duration required for data collection.

2.3 Estimating Expected Scenario–Specific Trip Assignments and Travel Times

The construction of a multiple regression model can provide insight into whether you need to adjust the Google API estimates prior to analysis. By setting the observed travel times from the station to the site of the call as the response variable, you can use predictor variables, such as various API estimated travel times, time of day, and distance traveled, to check model fit and accuracy regarding precise time estimates. Overall, this optional step is helpful because factors such as traffic patterns, route changes, or even specific local travel conditions could affect the API's ability to accurate capture travel time – in which case, adjustments would be improve the accuracy of the analysis.

For the purpose of your analysis, there are multiple models that you can choose from, such as the 1-predictor, 2-predictor, and 3-predictor linear models, as well as the Bayesian Additive Regression Trees Model, also known as the BART Model. At a high level, the BART model utilizes Bayesian decision tree frameworks to help capture nuanced, non-linear patterns in data [1], which can be helpful with travel time deviations and other factors that could affect your API estimates. Linear models are effective when the relationship between the predictor and response variables are approximately linear, while tree-based methods, such as the BART model, can model intricate non-linear relationships within the data. The use of one model type over the other depends on the nature of your data and the patterns within it.

To get started with building our models, we recommend the construction of a variable based on observed "best guess" travel times retrieved from the Google API. For each trip between the destination and the Central or South station, you will fill in the travel times as recorded by the API. These observed "best guess" travel times, along with additional variables such as time of day, day of the week, season, and the emergency/non-emergency nature of the call, form the basis for the corrective modeling step. If you find this step to be useful, you can use your chosen model to correct the estimated Google travel times to all of the observed call locations from each station.

Key Variables of Interest:

- `dispatch_time`
- `observed_travel_time`
- `api_tt_estimates`
- `best_guess_travel_time`
- `time_of_day`
- `distance_traveled`
- `day_of_week` *
- `season` *
- `emergency_category` *

[1] Bayesian Computation Book, Chapter 7: Bayesian Additive Regression Trees

* Note: suggested variables to introduce additional complexity into the models, since factors like time of the week and seasonal patterns may significantly impact call volume, traffic conditions, and travel time estimates.

The model building on the training data will entail two "sub-models" for each model type, one with the API estimated travel times as one of the predictors, and the other with the best guess travel time substituted in for the API estimated travel times. In order to determine the effectiveness of model fit and which model is the most ideal for your analysis, we can inspect the Root-Mean-Squared-Error (RMSE) for each model on the testing data. The lower the RMSE value, the better the model fit. After generating predictions and RMSE values from each model, you can compare the RMSE values for all of the models using the following interpretation: the model with the lowest RMSE can serve as an appropriate and ideal model fit for our analysis. If the RMSE value of the model using all of Google API's estimates as predictor variables is significantly lower than the models using just the API's best guess estimate, we can infer that the API estimates are fairly accurate. However, if we adjust the Google API estimates and it lowers the model's RMSE on the test dataset, it may be helpful to make adjustments to the travel time estimates before further analysis.

Summarized Procedure:
1. Build multiple regression and BART models with response and predictor variables.
2. Calculate RMSE for these predictions. Also, calculate RMSE when only using Google API's "best guess" values.
3. Compare both RMSE values for all of the models and identify which models are the most accurate fits.
4. Proceed by either adjusting API estimates or keeping them as is.

In terms of utilizing the Google-API-to-predicted-time adjustment model suggested above, consider the following scenarios below, where each scenario shows the number of ambulances stationed at each station.

	South	Central	Near North	Far North
Current (0)	1	3	0	0
Scenario 1	0	3	1	0
Scenario 2	0	3	0	1
Scenario 3	1	2	1	0
Scenario 4	1	2	0	1

For the added API-estimated travel time columns, `eTT.BG.So`, `eTT.BG.Ce`, `eTT.BG.NN`, and `eTT.BG.FN`, for each scenario, you can calculate the estimated minimum Google API time as:

$$\min \begin{pmatrix} \texttt{eTT.BG.So} \cdot I_{\text{South}} & \text{if } I_{\text{South}} \neq 0, \\ \texttt{eTT.BG.Cen} \cdot I_{\text{Central}} & \text{if } I_{\text{Central}} \neq 0, \\ \texttt{eTT.BG.NN} \cdot I_{\text{NearNorth}} & \text{if } I_{\text{NearNorth}} \neq 0, \\ \texttt{eTT.BG.FN} \cdot I_{\text{FarNorth}} & \text{if } I_{\text{FarNorth}} \neq 0 \end{pmatrix}$$

For each call in the observed data, you will assign a specific departure location, which will be different for each of the five scenarios above. For instance, the 23rd observed call in the mock data set you provided has `REF.GPS.LAT` of 36.4240 and `REF.GPS.LON` of -78.4538, which is located

at the Northern end of Vance County. So, for this observed call, the ambulance departure location would be Central station, Near North station, Far North station, Near North station, and Far North station (for Current Scenario, Scenario 1, Scenario 2, Scenario 3, and Scenario 4 respectively).

Once the process of assigning scenario-specific departure locations for each observed call is complete, you will choose to use the raw Google best guess times or the corrected times (using the model mentioned above). Note that the scenario-specific ambulance departure location assignment method described above assumes that there is a vehicle available at each of the scenario-specific stations at any given point in time. The observed data show that, depending on the preceding call(s), sometimes a call from the South end of the county may require a vehicle from the Central station rather than the South station for instance.

So, while maintaining the principle of assigning scenario-specific ambulance departure location based on the station closest to `REF.GPS.LAT, REF.GPS.LON` you may want to employ a corrective step that checks for certain cases. It is important that the corrective step accounts for overlapping trips, as it affects the availability of vehicles. This can be done by identifying trips where the dispatch time is earlier than the vehicle clearance time (denoted as `DT AVAILABLE` in the observed data set) of one or more of the previous trips. The end of each trip is scenario-specific and can be calculated by adding the difference between the dispatch and en route times to the dispatch time, the estimated travel time to the site, and the difference between the time of arrival and the time of availability. In other words, you would look into lagged differences in ambulance dispatch times and check that they are non negative - a negative difference could indicate that up to four ambulances (in the case of four total ambulances) are "out in action" when a new call arrives, which would heavily affect the response time only if the corrective step is properly implemented.

It is also important to find and acknowledge the anomalistic trips in the data, as these can easily skew statistics such as the average travel times. Anomalies in the data can present in many different ways. In the mock data, there are a few key outlier trips, which you are assuming will be reflected in the real data. For example, if the location of the call is outside of Vance County, it should not be included in a model that focuses on calls within Vance County. Furthermore, if there are any exceptionally long trips in the observed data, the long travel time might be a product of external factors such as weather and could potentially alter the model as a part of Missing Not at Random data (MNAR), which would need to be addressed.

2.4 Load Analysis

In this section, we explain how to conduct a load analysis to examine how the different scenarios would increase or decrease stress on Vance County's EMS System. Several methods for evaluating system load were considered, including:

- percentage of time an ambulance needs to be dispatched from a suboptimal station
- number of ambulances available at any given time of day
- amount of time each ambulance is in use
- a binary indicator for whether the ambulance at the closest station is available

However, we specifically wanted to highlight the last method–using a binary indicator for whether the ambulance at the closest station is available, and comparing this across the different scenarios–for being comprehensible and having a comparatively straightforward implementation. The value of this binary indicator would be 1 if the ambulance at the closest station is available when the call is placed, and 0 if the ambulance at the closest station is not available.

You will use two models. The first model will calculate the probability of whether a call's closest station varies between the different scenarios. As an example of this, a call from the south would not be impacted by whether the north station is placed at the near north or far north location, whereas a call in the north would be impacted by that decision. We suggest you create a 0-1 binary indicator (dummy) variable for the event that load is not sensitive to scenario (i.e., constant across scenarios). This will identify the calls for which ambulance placement does make a difference versus those where it does not. A logistic regression model can then be constructed to identify time- or location-related variables–season, hour of day, event ID, etc.–that are associated with this variable, and the model would likewise be useful for descriptive analysis. Each of the aforementioned covariates will have a fixed effect coefficient, where a coefficient of 0 means this variable does not affect the log odds of a call's closest station varying across scenarios.

The equation for this first model will measure the log odds of whether a call's closest station varies between scenarios. The log odds method of estimation is used because you are modeling a binary outcome. For more information on logistic regression and interpreting model parameters see here.

The second model will use only the calls that are predicted to vary across scenarios, and it will calculate the dependence of the load metric on scenario (i.e., how the load metric varies between different scenarios). As a trivial example, a scenario with one north, two central, and one south station will likely have a greater number of closest station ambulance dispatches than a scenario with one central and three north stations. The independent variable in this model will be the scenario. This model will be an indicator for whether a call was dispatched from its closest ambulance station under a given scenario. The equation for this second model will measure the number of calls where the ambulance at the closest station was available for each scenario. You can then compare the different scenarios' numbers to determine which scenario has the greatest amount of closest station ambulance dispatches, and which scenario has the least. In this model, each call will be evaluated for all five scenarios, so the data will represent 'repeated measures' and the model will include a random effect for call ID to account for this, in addition to the fixed effect for scenario. The resulting model will be a logistic link Generalized Linear Mixed Model, which you can read more about here.

For both of these models, R can be used to estimate both the fixed and random effects and any additional model parameters. Finally, if desired, we can finetune the model's fit by adding other variables or interactions between variables.

2.5 Travel Time Analysis

Ambulance travel time, a significant component of response time, is the metric you aim to optimize by comparing the average travel time across various ambulance distribution scenarios. The initial step in modeling travel time is checking whether the estimated travel times exhibit variation across different scenarios, which enables the identification of instances where travel time is influenced by

the scenario under consideration, and those where it remains the same. A binary outcome, hereby referred to as "differ," is formulated, wherein "1" denotes a variation in travel time across scenarios, and "0" indicates an absence of variation. This differentiation is modeled using a logistic regression model, specifically utilizing the glm() function in R, due to the binary nature of the outcome. The response variable is the binary outcome "differ," and the predictor variables, or covariates, encompass factors such as season, hour of the day, day of the week, priority, and system load. This model calculates the probability of observing a variation in travel times across scenarios, contingent upon the covariates. For instance, during peak summer seasons or specific hours of the day, the model may unveil a heightened probability of variation in travel times across various scenarios, potentially attributable to factors such as escalated traffic or road closures.

Upon recognizing instances where travel times show discernible variations across different scenarios (i.e. when "differ" is predicted to be 1 in the logistic model mentioned in the last paragraph), it becomes vital to model these times, particularly focusing on their dependency on the respective scenarios, given the existence of such variations. The Linear Mixed Model (LMM), especially employing the lme4::lmer() function in R, stands out as a robust tool for this endeavor (Bates et al., 2015). In this model, travel time is designated as the response variable you want to predict; the scenario operates as a fixed effect, symbolizing the specific ambulance allocation strategy being examined and quantifying the impact of various deployment strategies on travel time; a term based on event ID should also incorporated to capture the random effect, allowing each event to have its unique baseline travel time, thereby addressing the unobserved heterogeneity and inherent correlations within each event's measurements; and covariates, which include additional fixed effects like time of day, season, and system load, as previously mentioned, offer a mechanism to control and adjust for these variables in the model. Regarding the model's interpretation, since the primary focus is on the impact of ambulance distribution on travel time, you should identify the baseline scenario and examine the offset term for each scenario in the model output. The scenario presenting the smallest or most negative offset term corresponds to the shortest travel time predicted by this linear mixed-effect model and would be the scenario of interest.

References

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software*, 67(1), 1–48.*doi* : 10.18637/*jss.v*067.*i*01.