MBTA Alerts and Performance Proposal

1: Vision and Goals Of The Project:

Currently the MBTA offers a service to riders that will text or email them whenever an abnormal event occurs that they believe may cause delay (I.E. detour, junction not switched properly, etc). Every text or email is considered to be a service alert. This is a human entered metric without any actual data to backup whether or not these abnormalities actually cause a delay. The MBTA also has performance metrics of all of its trains, but has nothing linking alerts and performance together to determine the actual effect of these abnormalities. In this paper we will refer to alerts being the human entered messages and performance as the data given to us by the MBTA. The goals and vision for the project are detailed below.

- Final state of the project
  - Create an interface for MBTA employees and possible customers which allows them to quickly see the location of potential problem areas and delays and provides statistics on past delays and current conditions.
- Key goals of the project
  - Analyse archival MBTA data
  - Collect and integrate real time data with the given data (from GTFS API calls or from MBTA API)
  - Create tools to interpolate the data to estimate real time conditions in the MBTA system
  - Create a user friendly visualization of the effectiveness and / or a distribution model
- Context of decision making
  - What data analysis will be useful in terms of interpolating subway performance
- Shared vision of all team members
  - Using a combination of the MBTA and GTFS data, we want to create a platform that correlates the efficacy of MBTA alerts with delays in performance

2: Scope and Features Of The Project:

The scope for this project revolves around the subway system of the MBTA. We are focussing on reporting the most accurate delays we possibly can.
This includes pulling in external data we believe may have an effect on delays like weather or traffic.

Within Scope:

- Easy to use interface for customers and MBTA employees
  - Including visualization using D3
- Use of external data that we believe will improve accuracy of delays such as traffic and weather conditions
- Real time analysis of MBTA alerts and accurate prediction of potential delay

- Exploring how a delay can cause a chain reaction up the line
- relationships between single train delays and the line as a whole

Outside of scope:

- Applying what we've learned to other areas controlled by MassDOT such as I90 traffic patterns and boats
- Applying what we've learned to other city transit systems.
- Using our findings to suggest improvements to the transit system

3: Solution Concept

Global Architectural Structure Of the Project and Walkthrough:

Below is a description of the system components which make up the structure of the project. The following methods and technologies are chosen to achieve a solution:

- **Data aggregation:** Collect and parse all data streams available in order to produce and prepare a working dataset for processing.
    - SQL database to congregate the data coming from various sources.
    - R library capable of reading MBTA alert performance from a SQL database.
    - Python library capable of downloading, and parsing the realtime GTFS updates from the MBTA system and incorporate the data in the SQL database.
- **Data interpolation:** Find correlations between the various factors such as the delay alerts, subway performance, weather, traveler congestion, and ultimately determine the causal factors in performance.
    - R library can run various econometric and regression models on the data aggregated in the SQL database
    - Node.js interface has ability to call the the R library
- **Data prediction:** Extend the current dataset to include the ability to make future performance predictions based on past models and current conditions.
    - R library interpolates MBTA performance based on predictors found during interpolation
    - Interface for MBTA employees to call functions in the R program and return results.
- **Data visualization:** Create a method of conveying the performance of the subways and effectiveness of the alerts to the user. Also incorporate a way to display the accuracy of real time performance predictions.
    - Node.js application interfaces with the R library and performs API calls to other data sources to read historical and live data in json format.
    - Javascript web application renders the relevant historical data and live data for the end user to view and interact with. A front-end framework like Angular can be used to take advantage of the two-way databinding.

  - D3.js library is used to create useful data visualizations to convey key findings to the end user.

Below is the system diagram which provides an overview of the different elements of of the project.

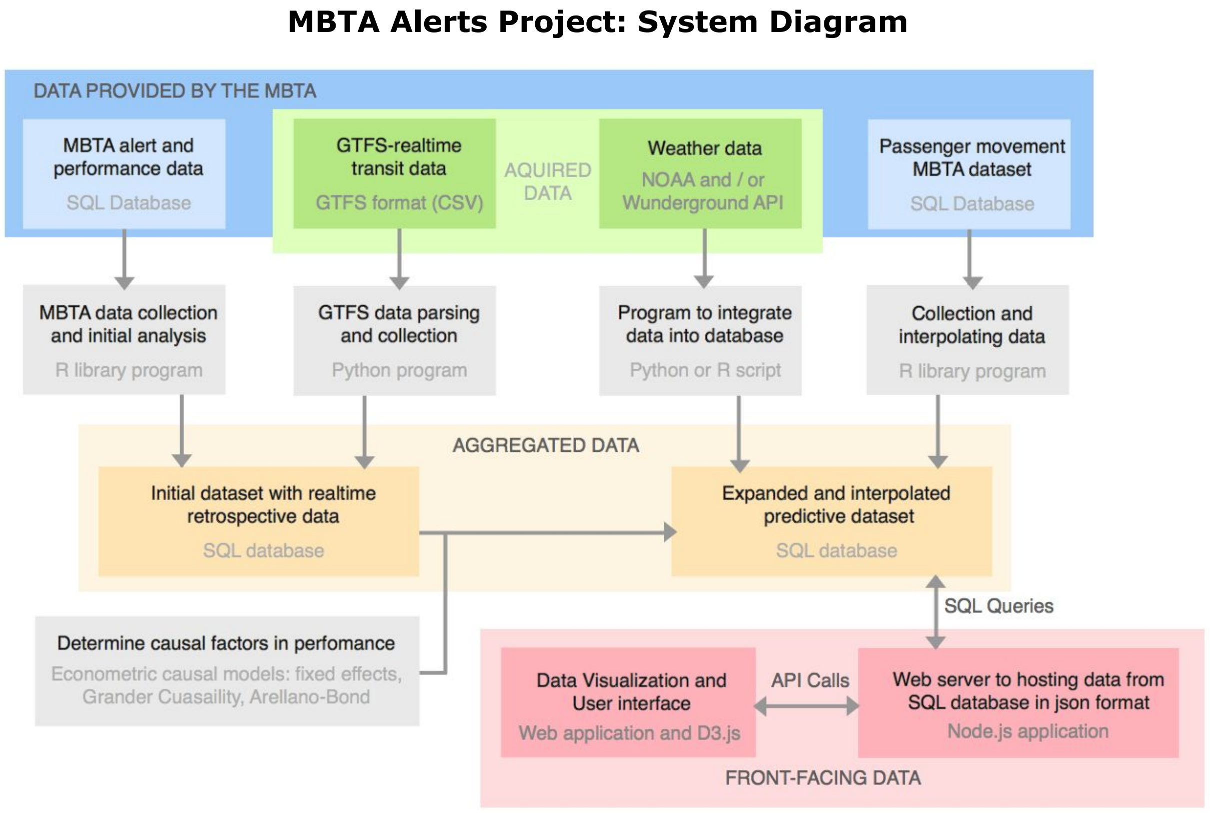
MBTA Alerts Project: System Diagram
DATA PROVIDED BY THE MBTA
MBTA alert and performance data
SQL Database
GTFS-realtime transit data
GTFS format (CSV)
AQUIRED DATA
Weather data
NOAA and / or Wunderground API
Passenger movement MBTA dataset
SQL Database
MBTA data collection and initial analysis
R library program
GTFS data parsing and collection
Python program
Program to integrate data into database
Python or R script
Collection and interpolating data
R library program
AGGREGATED DATA
Initial dataset with realtime retrospective data
SQL database
Expanded and interpolated predictive dataset
SQL database
SQL Queries
Determine causal factors in perfomance
Econometric causal models: fixed effects, Grander Cuasaility, Arellano-Bond
Data Visualization and User interface
Web application and D3.js
API Calls
Web server to hosting data from SQL database in json format
Node.js application
FRONT-FACING DATA

Figure 1: MBTA Alerts project architecture. Items in blue represent data provided by the customer, MBTA. Green represents acquired data from secondary sources, yellow represents the aggregated data, and red represents the program which the end user will experience and interact with. Gray represents methods used to process the data.

Design Implications and Discussion:

Key design decisions and motivation behind them:

- The dataset we will be using for analysis and prediction will be on a SQL database. The data provided by the MBTA is already in SQL format, and there are many options for different languages to interface with the SQL database.
- R will be used as the primary data analytics package. R is preferable over competitors like Stata or SAS because it is open source and highly extensible, allowing us to create specific libraries for the MBTA that can more easily interact with our user interface.

- Historical weather data will be aggregated over the same time period as the MBTA data to hopefully find correlations between the weather and delays. Either R or Python can collect data from free APIs from sources like the National Oceanic and Atmospheric Administration (NOAA) or the Wunderground API.
- A node.js server will be used to host a web application which will be used as the user interface with the data for the customer (and potentially MBTA customers). Specific data findings will be collected and converted to JSON file format to be used by the node.js server.
- The collection and parsing of the GTFS-realtime data will be done using Python. Google offers support for Python, and there are existing open-source libraries that can be used in addition to our own python code.
- The front-end web application will be using a front-end framework such as Angular.js to take advantage of the two-way databinding. In addition, the data visualization library D3.js can be used as it offers many tools optimized for visualizing data on a HTML SVG or Canvas object.

4: Acceptance criteria

As per the MBTA's request, the minimum acceptance criteria is to find correlations between the alerts and actual performance of the subway system in Boston based on the given historical data.

Stretch goals:

- What is the difference between train-based and passenger-based performance metrics?
- Including weather and other datasets in the analysis

5: Release Planning

https://trello.com/b/Gyy4ChBf/mbta-alerts-and-performance-anlaysis

We intend to release our platform for monitoring alert performance over 5 iterations.

Release 1 (due week 5):

A basic R library capable of reading MBTA alert and performance data from a SQL database, and returning to the user descriptive statistics. These should include:

- Frequency of alerts over time
- Total number of alerts in a given time period and/or for a given service
- Classification of alerts
- Histograms of MBTA performance over time and by specific services

A Python program that will fetch and parse GTFS updates from the MBTA system, and incorporate these updates into the SQL database used by the R library.

<u>Release 2 (due week 7):</u>
Add the ability to find correlates of performance alerts, as well as the ability of those alerts to interpolate future MBTA performance. This should include:

- The ability for a user to run cross-sectional regression models on MBTA data
- Identify the strongest predictors of MBTA performance, and the ability of alerts to interpolate future performance.
- Econometric causal models (such as Granger Causality, Arellano-Bond, and fixed effects) to determine causal factors in MBTA performance, and alert reliability.

Implement a node.js interface with the ability to call functions in the R library and query the SQL database.

<u>Release 3 (due week 9):</u>
Extend the R package to include the ability to make future performance predictions based on past models and current conditions.

- Using previously identified predictors of reduced performance, implement a system that will interpolate MBTA performance by observing when those predictors arise in real time.
- Compare these predictions to those made by the traditional method of issuing alerts.

Add R library methods to create json formatted files to send to the node.js application.
Create a web application frontend in Javascript that will allow MBTA employees (and possibly customers) to easily call functions in the R program, and return the results.

<u>Release 4 (due week 11):</u>
Add new data sources to the existing set available to the R library. The MBTA has asked us to make predictions based on their GTFS data. We believe we might be able to improve predictions further by including other data sources, such as:

- Passenger movement data. The MBTA has estimates of the number of passengers that use particular services at particular times. Can we incorporate a network analysis of movements between stations?
- Weather data. As we know from last winter, the weather can dramatically impact MBTA performance. If we include the NOAA or Wunderground API in our data, can we improve our interpolations of MBTA performance?

Implement data visualization of the dataset using D3 that will quickly convey performance data to the user.
Create API backend which allows the web application to query the SQL database.

Release 5 (due week 13):

Refine and finalize the data visualizations and user interface, and incorporate the ability to display real time performance predictions.

- Provide visualizations of the current best and worse performance in the MBTA system.
- Provide a map of current performance issues, as well as our estimates of future performance issues. This should also be capable of displaying current alerts, for easy comparison.