Analytic Engine Client Documentation

Description and Features:

The analytic engine is a software suite that uses a variety of methods to help analyze metrics from a given database of simulations and real time resource data based on general traffic to and from a given source. This could range from CPU usage of a computer to the average pull requests from a webpage. The current version of the analytics engine, performs multiple statistical analysis methods including Covariance, Correlation, Deviation and Threshold analysis, Normalization, Null cleaning, and Deep Learning methods such as Entailment search. We also have give the user the ability to use the program as either a command line interface on your local machine linked to a server, or remotely through our webpage UI.

Excluded Features and Future Plans:

The analytic engine supports a few more features, but were unable to be implemented fully due to time constraints. Such features are: outlier removal/smoothing of data, finding of local max and min, more comprehensive interpolation of points, removing nulls (rather than treating them as 0), and other statistical analysis methods such as different correlation/covariance formulas, the ability to store data to decrease result analysis time via mongoDB implementation and as well general performance improvements such as asynchronous calls for all features in the engine and multi-core threading for increased throughput.

Possible ways to approach creating a more robust interpolation:

- Isolate points of interest (local minimums and maximums in a subset of the given set of points) and use these to match up with points at the same time in the comparison metric. This avoids using too many interpolated points. (we have started creating a search for local minima and maxima as mentioned above)
- Find a way to create a parametric equation for a given set of points and use its derivative to discover possible points of interest (not implemented)

Making statistical analysis more robust:

Using linux timestamps was causing issues with our analysis, so what we do now is assume that data is at a consistent interval in a given range and interpolate the other data points at that same interval. Including time and value with each datapoint would lead to better analysis and let interpolation create uneven spaces in the data. This lets us use more real points and lets us

drop the ‘NA’ values rather than regard them as 0. However, some ‘NA’ values are recording 0 events in a timeframe so it’s not always incorrect to use this method before analysing the points.

Performance gains:

Multi-thread the search functionality

NodeJS was not the correct choice for our analysis-- while we wrap R for the actual number crunching, this is creating an instance of R for each calculation and destroying it. This should be an async process, taking advantage of as many instances of R that the server can run at a time but the R wrapper library for NodeJS seems to have some bugs when scaled up. Likely from NodeJS being asynchronous language and the package and methods we used to make some aspects asynchronous were incompatible with our current implementation of NodeJS, like the aforementioned R wrapper.