# Information

Title: Crime Victoria Visualisation Application

Authors: Aaron Tillekeratne, Alen Bednjacic

Date: 19 March 2017

Task: Provide a visualisation platform to quickly gain insight into the crime in Victorian suburbs.

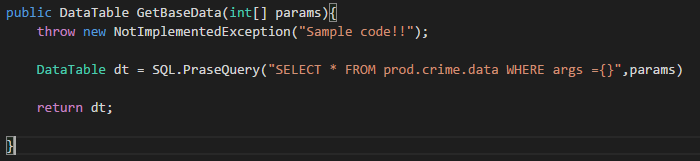
# Description

The main idea behind this visualisation is to provide the end user with a means of data exploration without having to be overly data savvy. The application is to be via web service and will be focused at every day (non-analyst) users who want to see how crime is distributed in Victoria. The application is split into two main components, maps and relationships.

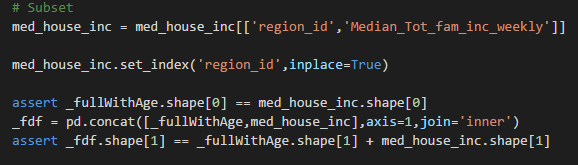
The maps part of the application is targeted to provide quick visualisations with crime data in a spatial setting. The default view is the Victoria map coloured by the number of crimes. This enables any user who’s visiting the page to immediately gauge the crime prone areas. Users can also change the type of visualisation from the map to bubble charts, heat maps and tree maps. These plots enable the visualisation of extra dimensions such as crime type, age groups, education levels, etc.

The correlations screen is aimed at users who indent to do a bit more of a deep dive into the data. Users can drag and drop measures into the worksheet and generate histograms and scatter plots. Users can also generate correlation lines and recover the correlation coefficient to see the strength of the relationships. Typical dimensions are also made available so that users can visualise clusters or groupings if present.

The design pattern used in this application is Model View Controller. A tabular data structure is adopted in the model layer, which the application will access in a read only mode. When any particular view is invoked by the user, the controller will fetch the data from the model with simple select statements. Code example shown below.



Then it is fed into the controller, which will do all transformations in memory as required by the view. This is generally parsed from the application backend language to a python/R interpreter. The python interpreter will reshape the data to what is required (eg segmenting age, etc.) and return a JSON object back to the web application framework code.



Then the web application backend will send the JSON object to the users browser via AJAX, where the visualisation library will render the JSON object to the required visualisation. The visualisation library will be essentially providing a high level interface to theD3.js library.

# Mathematics

Many of the math function calls will be aggregation of summation and mean calculation in the Map view. In the more complex correlations view, there will be linear calls to linear regression functions. Since the application is not meant to encompass all the abilities of an analytics tool, the users will have no direct interaction with the mathematics, rather it’ll be inferred from the visualisation to call the underlying math function.

# Software

There are a number of software and system level requirements to facilitate a service as proposed. Firstly, a database is required to hold the base model data. A web server is also required to host the application itself.

Inside the application, software requirements include, a server side scripting language to interact with the data model as well as carry out data transformations as requested by the view engine. There are numerous languages including C#, python, Java, etc. Since all transformations are done in the controller layer, a data analytics language is also required, usually R or python.

The rendering will all be done in the browser using JavaScript. Visualisation library such as D3.js or C3.js. This keeps the concerns separate and stops the data layer from being manipulated.

# Cost

As with any web app development, there is some non-trivial cost involved. However, due to cloud services, web apps can be serviced with costs as low as $50 month including database and web hosting. Rapid development frameworks such as python’s Django or C#’s Razor MVC enable quick development times even perhaps less than a month.

# Resources

The most important resource for tool such as this is a Data Scientist. The person developing this tool will need to:

1. Scope the traffic and users.
2. Choose infrastructure for database and web service.
3. Develop the database.
4. Write the server side application.
5. Write the data manipulation in the control layer.
6. Write the front end visualisation in JavaScript.
7. Deploy the app.