The program consists of two parts: javascript and java. Java part consists of all the background logic like data processing, parsing and also interactive UI. The main package contains a class called “webview” which serves as the interface between the imbedded browser and java program. Thus the interactive UI from java controls the various visualizations of data in javascript.

A few things as reminders:

1. The javascript code is contained inside /src/resource folder. Every time anyone wants to modify the javascript code, he/she needs to first modify it in src/resource folder then click “clean project” in eclipse to let the IDE copies the javascript files into target folder so that the new code can be in effect for the program.
2. The library for machine learning is from library which must be configured in each computer first. Here is the procedure to set up the library: <http://docs.opencv.org/doc/tutorials/introduction/java_eclipse/java_eclipse.html>.
3. All code related to javascript interface in java must run in javafx thread. Eclipse does not work well with javafx, E(fx)clipse is a very good option.
4. Double click on the google map will cause a bug and the program to be very slow in about 5 seconds. This is due to the javafx browser. I currently haven’t found a good way to go around it.

Javascript:

The javascript consists of two objects, precinct and rpa, and one main class.

Rpa is a subclass of precinct. They are basically object representations of the rpa and precinct which contain a “google.map.polygon” variable that provides the visualization.

The main class contains two classes of methods. One is GUI control like “showMarker()”, etc. The other is helper method that receives parameters from java and creates an object representation of marker, precinct and rpa, such as “addMarker()”. These objects will be created when the java program starts running but will be visible only after the user selects to do so.

Java:

The java contains six packages. The two most important packages are the main package which has the logic for UI control and the plot-tools for data-processing and object representations of various visualizations. Misellanies contains some useful tools, “test” class is just for me to test the program. CrimeCase is an object representation of crimes. Mypanel contains customized classes of UI for this program. Strategy implements the strategy pattern for “Setfocus()” method. Here is a quick description of the classes:

CrimeCase Package:

1. crimeCase: Takes position and time parameters and create an instance of crimeCase object. Time parameter must be either a long or a date type. Type of the crime is not necessary for the constructor. Contains helper function like “setPrecinct” that determines which precinct the crime belongs to based on geo location.
2. pastCrime: Contains parsing tools for excel file, also contains parsing tools for the txt files that I create as a representation of burglary data. The data is pre-processed so that the program already knows which precinct each crime belongs to. This helps make the program faster. However, txt is not a very good way to represents data in a portable way. I plan to build xmls for crime data soon.

MainPackage:

The main package has a main UI control class: “crimeDist\_init”, five tabbed panel as a subclass of “myComponet” in myPanel, and also a “webview” class.

1. crimeDist\_init: main UI control, the entry point of the program. Initialize webview, data, and add various UI components onto Jframe. Also control visibility of marker and rpas, if the visibility of these two components change, the crimeDist\_init will call “changeStrategy” in webView so that only the elements that are selected to be visible will be shown on google map.
2. Various panels: Controls the panel they corresponds to. Usually calls the plot tools to process and get the data for visualization in webview or chart plots. For past crime and simulated crime, they will call “setFocus()” method to enable the visualizations that belongs to the selected precinct to be visible instead of directly call “ShowMarker()”. Since “setFocus” implements strategy pattern, past crime and simulate crime’s functionality can also change smoothly based on the user’s selection. However, crimes from all precincts and rpas’ internal state will be cached in javascript even though they are not all visible. This will make the transition from one precinct to another faster and also make the state transition faster.
3. WebView, as described before, is an interface between javascript and java, it has “changeStrategy” method which implements strategy pattern for ”Setfocus()”. It also has some useful methods like “addCrime() or addPrecinct()” that takes a java object and transforms it into a javascript object using “executeScript()”. Since using “executeScript()” is error prone, it is recommended that any code related to web should be in javascript if they can.

Miscellanies package:

1. Cookies: A very small database containing values like how many clusters should there be in the Gaussian mixture model.
2. DateFormatter: for formatting date in dateChooser.
3. kmlParser: Parse data from kml, currently can parse rpas and precincts. Returns an array of polygon objects.
4. saveMap: helper class for printing.

MyPanel package:

1. datePanel: Panel for selecting date range.
2. Mycompnent: The super class of all the panels in main package, useful for saving redundant code, has an “addtext” method that add the text to the panel and a “removealltext” method.

PlotTools

1. areaIdentifier: Takes in a point and a polygon and determines if the point is in the polygon. Use beam method that takes a point and creates a very long beam along the x-axis. Then calculate the number of intersections between the beam and the path of the polygon. If there is an odd number of intersections, the point is in the polygon. This method is fast but not accurate for edge conditions.
2. gaussianMixture: Contains the model for Gaussian mixture special crime distribution. Generate a model based on the previous data, must use the opencv library. It has methods called “generateCrimes” to predict crime based on the previous data.
3. HeatMap: Generate a heat map over the area the user is looking at right now, then attach it to the google map. I do this due to two reason. Firstly, calculating and loading a heat map for the Nashville area takes too much calculation. By using the bound of the screen the user is looking at, the resolution can be better without sacrificing the speed of the program. Secondly, google map api contains some useful and convenient method about the area user is looking at. It returns a “bound” object which is also used to attach image overlay. Thus the bound can be used in both places to reduce code. The heatmap class first get density for a 100\*100 square area. Then it rescales all those values to be from 0~1. Finally it generates an image based on a method called color interpolation. I use JheatChart at first but it only supports two colors, so I decide to write my own methods. The detail is in the class.
4. PlotTemporal: Predict the number of temporal crimes based on the parameters from python code right now using Weibull distribution. See the paper for model detail.
5. Polygon: A class for the polygon, containing array of points for all the vertex in the polygons.
6. RpaPolygon: A subclass of polygon for representing rpas, has a variable representing the number of crimes in the polygon.
7. RpaAggregate: An aggregate of rpas, useful for some operations like refreshing the colors on the google map based on the number of crimes in each rpa and assigning crimes to rpas based on the position of crimes.
8. Temporal density: contains some helper method for plot temporal.

Strategy:

Implements strategy for “setFocus()” method. The markerStrategy will show all the markers in a certain precinct if the method is called. The rpaStrategy will show all the rpas in a certain precinct if the method is called. Null strategy does nothing and both strategy does both.