***Important Files- ones you will be in the most***

1. services folder
   1. venture.js – this is the file that needs to be manipulated the most in order to successfully wire the app to the venture server. In its current form, this service acts as a javascript object (think class in OOP) that contains 7 properties and 5 functions.
      1. Properties
         1. continuous – Boolean that keeps track of whether or not you are running a continuous inference, is used in the sendCmd function to determine whether on success of submitting a command the object needs to bother calling *list\_directives*.
         2. toHandler – used to store the interval loop that is used when running a continuous inference. It is assigned in the inferContinuous function and cleared in the stopInferContinuous function.
         3. commands – and array that holds string primitives. Every time a command is entered it is pushed to this array. Will be used for the ability to scroll through passed commands using the ‘up’ key, and eventually used for exporting all entered commands into a text file.
         4. apiReturns – and array of objects that includes the entered command, the directive id, and when appropriate the value returned by the venture server. I also tacked on an error property that can store an error message and an success property which is just a boolean representing whether venture could interpret the user command. This array is iterated over in the index in the li element under the ‘history’ div.
         5. directives – an array of objects that represent the current directives (the model). I have no real set structure for these objects so feel free to change what I have.
         6. assumes – array of the parameters of your model used in the index.html to generate the radio buttons to choose the x and y.
         7. valueLog – an array of objects that keeps track of the last 100 values of each parameter. The structure of each object includes the symbol of the parameter and an array of objects that represents the list of values and the corresponding unique ids (timestamps of when they were received from the server)
      2. Functions
         1. sendCmd – function that is used when a user enters the command. Will add command to the commands array and then (when operational) will make the correct ajax call to the server and parse the return data, ultimately adding it to the apiReturns array.
         2. getDirectives – this function will be called to send the list\_directives command to the server and parse the return objects. It will pass the return objects to the directives array and also it will filter out the directives with the ‘assume’ instruction and put those in the assumes array.
         3. storeValues – this function gets called every time the getDirectives function is called. Its job is to keep track of up to the last 100 values of each parameter in the model.
         4. inferContinuous – continuously call the getDirectives method.
         5. stopInferContinuous – stop the continuous calling of the getDirectives method.
      3. Next steps
         1. Right now all ajax calls are commented out. The big steps will be to uncomment these and have them send the correct http requests to the venture server.
   2. helpers.js – this service can also be thought of as a class that will just contain functions to handle different parsing responsibilities. The two functions currently defined (a generous term) are placeholders. Feel free to change these methods (and their names). Right now the placeholders are for parsing the return from the ‘list\_directives’ command (the directive function), and parsing the return data sent by the server after a user has issued a command (api function). Feel free to add to these.
   3. controllers/main.js –this file basically acts as a connector between the view (index.html) and the returns from the services defined above. It is largely responsible for event handling and acting and registering the venture service onto the scope variable $scope.app. If you want a variable to be available to the ui, you must register the variable on the $scope variable.

***Semi-important Files – ones you might be in, but if so, not often***

1. directives folder
   1. directive.js – creates an attribute directive that when applied to an element will register an event on pressing the *enter* key.
   2. histo-marginal.js – creates an element directive that handles the creation of the histogram and marginal plot graphic. The directive takes two attributes representing the data to plotted for x and y. It contains 3 functions, one to update the histograms, one to update the scatter plot, and one to listen for changes in the data.
2. index.html – contains all the html tagging for the app. In addition, contains angular template language which can be identified through the use of attributes that begin with ng, such as ng-repeat (which iterates through an array object from the controller layer) and ng-model (which binds data from ui to the controller and vice-versa). If you create new .js files you must add them to the index.html files to ensure it is included when you run the web app. You can add these after the body tag.

***Not really important – should not have to touch these files***

1. style folder –contains stylesheet specs for the html document. Includes main.css which I wrote and bootstrap.min.css which is an open source style component from twitter.
2. fonts folder – contains glyphicons that are used by the bootstrap library.
3. vendor folder – contains third party libraries including d3 (for graphics), angular, and underscore (json manipulation)
4. js/app.js – just initializes the app, I do not foresee anyone needing to make changes to this file for the first version.