Advance Features in daru-view

- **PERSONAL DETAILS:**

  - **Name:** Prakriti Gupta
  - **Github:** https://github.com/Prakriti-nith
  - **Email:** prakriti.gupta10@gmail.com
  - **Location:** Ajmer, Rajasthan, India (UTC+0530)
  - **Phone:** (+91) 9882926441

- **Ruby and Sciruby:**
  Ruby is an eloquent language that almost reads like plain english. It is a cleaner object-oriented language with excellent support for functional programming. However, it lacks some scientific libraries that are used in day-to-day work and the support for some data visualization tools.
  This is where my interest for daru and its plugins increased. Through Sciruby I want to improve the support for such libraries which are useful in fields like data science and scientific computing.

- **Science and its area:**
  Science brings to mind a mess of things: high school biology and chemistry classes, college orgo labs, pipettes and finely-tuned scales, physics problems and equations with more constants than possible for real life. However, if you think about anything for long enough, you can find the "science" in everything: from the matter it's composed of to how it obeys the laws of nature to what makes it work.
  I was always intrigued about the mechanism followed by the nature that led to a natural likeness towards physics. Dwelling into it I found myself more comfortable with mathematics.

- **Experience with programming languages:**
  - **Ruby and Javascript:** Proficient in both the languages.
  - **C and C++:** Published various articles in C and C++ on geeksforgeeks.org during the internship (Link).
  - **Android:** Worked as an Android Developer Intern at a Bangalore based startup in India. I have also developed several Android apps. Some of them are: To Do Reminder, TV Shows, Yoga app.
  - **Python:** Intermediate experience with python. Built two projects in Python - Music Player with live lyrics, hack-lastfm.
  - **Octave/Matlab:** Intermediate experience. Made a project for phishing website detection in Matlab.

- **Educational background:**
  I completed my schooling from Cambridge Court High School Jaipur, Rajasthan, India. Currently, I am a junior year Computer Science undergrad at the National Institute of Technology, Hamirpur. The research topic I am interested in is Machine Learning.

- **Past contributions:**
  Here is a list of pull requests I've worked on over time.

- **Other commitments:**
  I have no other commitments aside from GSoC this summer. I have holidays (11th May - 29th July) during most of the coding period and can work full time (45-50 hrs per week) on the project. I have classes in 23rd April - 10th May and 30th July - 14th August, when I'll work 40 - 45 hours.

- **Fun vacations:**
  I am planning to have fun coding this summer.

- **Classes in summer:**
  None

- **Other employment in summer:**
  None, if selected for GSoC.

- **Past GSoC projects:**
  None

PROJECT PLAN FOR GSoC 2018

Data Visualization through daru-view:

Daru-view aims to create interactive plots and tables in Ruby using various adapters available like GoogleCharts, HighCharts, Nyaplot and DataTables. However, the indirect access of these plotting tools through the dependent gems, limits their usage as the dependent gems are not up-to-date. So, my plan for GSoC 2018 broadly focuses on the implementation of advanced features in daru-view. This includes extending the code of these dependent gems and the execution of various features available for Google charts JS, HighCharts and DataTables.

Google Charts:

To enhance the visualization of the data through google charts in daru-view, I want to work on the following five ideas:

- **Implementing the DataView, ChartWrapper and ChartEditor class in daru/view/adapter/googlecharts:**

  DataView:

  - ❏ A DataView is a convenience class that provides a read-only view of a DataTable, with methods to hide or reorder rows or columns quickly without modifying the linked, original data.
  - ❏ Some of its advantages are:
    - ❖ It can hide or show selected columns.
    - ❖ It can sort or filter rows without modifying the underlying data.
  - ❏ For the DataView class in google charts adapter , I will be implementing some of the methods like setColumns(), hideColumns(), setRows(), hideRows() and few others in a way similar to the DataTable class like get_column, get_row and get_cell so that they can be called directly by the `data_view` object.
  - ❏ The default constructor will take the `GoogleVisualr::DataTable` object as argument in order to create the view for it.
  - ❏ The `to_js` method to return the JavaScript equivalent for the `data_view` instance will be implemented as:

```
def to_js
  # code to get js equivalent for data_table instance
  js << "var data_view = new google.visualization.DataView( data_table );"
  # code for the formatters and any manipulation on data_view object to
  # append in js
end
```

  This will further be used in the `draw_js` method to get the full javascript for the dataview.

  ChartWrapper: ___

  - ❏ A ChartWrapper class is used to wrap the chart and is used to handle all the loading, drawing and datasource querying for the chart. It also eliminates the need to specify chart libraries.
  - ❏ User can pass additional option of `chart_class` which will take the value as `Chart`, `ChartWrapper` or `ChartEditor`. Default value will be `Chart`. If

the value of the option retrieved is ChartWrapper then the object of ChartWrapper class will be created here in the code.

- ❏ For the implementation of the ChartWrapper class similar to the BaseChart class for GoogleCharts (link), the prominent method that should be implemented is to_js to get the script of the chart (when show_in_iruby or div is called in iruby notebook or web frameworks respectively, to_js method is ultimately called at the lower level).
- ❏ The ChartWrapper constructor will take values as: chartType, dataTable, options and containerId.
- ❏ Various setter methods like set_chart_type, set_options, set_datatable will be implemented that can be used to set specs parameters in the constructor.
- ❏ In the load_js method which will be called from to_js method (eg: link), there is no need to load the packages explicitly, as the chartwrapper class handles the looking up and loading the chart packages.
- ❏ A sample example of the ChartWrapper class is shown here.

ChartEditor: _____

- ❏ The ChartEditor class enables the user to customize the chart through the in-page dialog box. This dialog box provides numerous features to **edit the chart at runtime**.
- ❏ If the value of the option chart_class provided by the user is ChartEditor then the object of ChartEditor class will be created here in the code.
- ❏ We would be implementing the ChartEditor class in the daru/view/adapter/googlecharts by extending the code of google_visualr (similar to the BaseChart - link). Here also the most important method to implement is to_js method.
- ❏ The ChartEditor works on the ChartWrapper object.
- ❏ So, the to_js method will first include the js equivalent for the ChartWrapper class which will further be used to get the js equivalent for the ChartEditor. Further, the callback method of the eventlistener will be added in the to_js method in order to save the chart when the user click on the ok button in the dialog box.
- ❏ I have tried an example to demonstrate the ChartEditor class (link).

- **Export the chart in various formats:**

In google charts the image url can be obtained using chart.getImageURI() which can be used further to export the chart in various formats. APIs like chart.download_pdf, chart.download_image can be created for google charts

which will render the template containing the chart_div to create the chart object and further the script to download the chart.

To download the chart as pdf:

```
var doc = new jsPDF();
doc.addImage(chart.getImageURI(), 0, 0);
doc.save('chart.pdf');
```

To download the chart as image:

```
var a = document.createElement('a');
a.href = "chart.getImageURI()";
a.download = "output.png";
document.body.appendChild(a);
a.click();
document.body.removeChild(a);
```

- **Import data using Google sheets:**
  In order to import the data from the spreadsheet, the spreadsheet must either be visible to everyone or the page must explicitly acquire an end-user credential. If either of the case satisfies, we would then be querying the spreadsheet to retrieve the data.
  - ❏ The daru-view user would send the data in the string format (URL of the spreadsheet). The user can add the optional parameters to the URL as described in the documentation (link).
  - ❏ We need to check if the data is of type String in show_script method here. For that we can pass @data along with the @chart in the methods show_in_iruby (link) and div (link).
  - ❏ If it is of type String then we will call another to_js_spreadsheet method which will contain two methods of js, first to create query and second to handle the response.
  - ❏ The example I tried to import using spreadsheet can be found here.

- **Adding more methods in daru/view/adapter/googlecharts:**
  Google Datatables implemented in google_visualr contains only the basic methods to implement the table. It lacks some of the prominent ones like getNumberOfRows(), getNumberOfColumns(), getSortedRows(), and few more. So, we need to extend the code of google_visualr in daru-view and add some more methods to it.

  Advantages:
  - ❏ Using Daru::DataFrame for the purpose will require a number of requests to the database. Here, I would be implementing these methods at the frontend which will work faster.

- ❏ Also, this will provide the daru-view user to work directly with the table rather than switching again to Daru::DataFrame.
- ❏ Currently, we are not using Daru::DataFrame as the only data container (data Array as well). So, there is a need for such methods.

- **Show multiple charts:**
  To show multiple charts, i.e., in an iruby notebook displaying two charts in the same cell and in web frameworks displaying two charts in the table row, we can do the following:
  - ❏ First, Daru-view user would create two Daru::View::Plot objects and one another Daru::View::Plot object that will combine the other two by passing the data in the new object as an array of the other two objects. For example:

```
line_chart = Daru::View::Plot.new(data1)
line_basic_chart = Daru::View::Plot.new(data2)
combined_chart =  Daru::View::Plot.new([line_chart, line_basic_chart])
combined_chart.show_in_iruby #For iruby notebook
```

  - ❏ @chart object (link) will also contain the plot objects' array. So, in show_in_iruby method in googlecharts.rb, we can call th show_in_iruby in display.rb by any of the array object (preferably using the first object for the purpose)
  - ❏ Now, we can use the data in to_html method to know if the data is an array of Daru::View::Plot objects then we would use another template to render the charts as:

```
<table class="columns">
  <tr>
    <td><div id='<%= id1 %>'></div></td>
    <td><div id='<%= id2 %>'></div></td>
  </tr>
</table>
<script>
<%= chart_script1 %>
<%= chart_script2 %>
</script>
```

  - ❏ In this we can easily get the individual script of each chart by placing a loop in to_html method.

HighCharts:

- **Implementation of HighStock and HighMaps features in daru/view/adapters/highcharts:**

HighStock:

HighStock is based on HighCharts, meaning it has all the core functionalities of HighCharts, plus some additional features as:

- ❏ Navigator
- ❏ Range Selector
- ❏ Scrollbar
- ❏ Crosshair (Crosshairs are implemented in Highcharts (not enabled by default), but not in `lazy_high_charts` gem)

In order to implement these features, we need to extend the code of `lazy_high_charts` gem in daru-view.

For that we can create a separate class to create HighStock objects. This class will add more attributes (further addition to HighCharts' attributes) to the `LazyHighCharts::HighStock.options`.

`high_stock` method (similar to high_chart method) to get the script of the chart has already been implemented in the `lazy_high_charts` gem.

I tried some examples to exhibit the features of HighStock which are shown here.

HighMap:

For the implementation of HighMaps in daru-view :

- ❏ Dependency for HighMaps (using CDN):
  <script src="https://code.highcharts.com/maps/highmaps.js"></script>

  We need to add the dependencies of HighMaps and its modules and update the rake task for HighCharts to include the dependencies for HighMaps also as:

  ```
  task :update => [:core, :stock, :map]
  ```

  where `task :map` will grab the HighCharts Map JS from Upstream.

- ❏ We need to create a `high_map` method in LayoutHelper module (just as the `high_chart` method here) to get the javascript of the chart by extending the code of `lazy_high_charts` in daru-view.
- ❏ Just as the HighChart class is created in lazy_high_charts here, we also need to create a separate class for HighMap with constructor to initialize the map and further methods like default_options and others in daru/view/adapter/highcharts which will include module LayoutHelper.

- For instance, HighMaps like highlighted areas, categorized area, distribution maps and others can be generated using the above implementation as:
  - @chart object will be created by LazyHighCharts::HighMap.new.
  - In to_html method high_map method will be called.
  - In the init_script method dependent_js of HighMaps will be added further.

  A working example of the highmap is shown here.
- However, there are some advanced HighMaps like this, which are quite complex. I will add the support for some advanced HighMaps further in the timeline.

- **Custom styling CSS in HighCharts:**

  - To incorporate the **styled** mode of HighCharts in daru-view, we first need to add some javascript dependencies which are available under *js/* folder on code.highcharts.com. However, from these files presentational code is removed, and CSS is required to style the chart. The default CSS file for **styled** mode is available as */css/highcharts.css* on the root of code.highcharts.com.
  - To incorporate dependent CSS files:
    - In init_script method (link) append dependent css in style tags as:

      ```
      js << "\n<style type='text/css'>"
      js << Daru::DataTables.generate_init_code_css(dependent_css)
      js << "\n</style>"
      ```

    - In web frameworks dependent CSS will be added in the head as dependent js is added.
    - I am searching for the way to incorporate CSS files in IRuby notebook just like js (IRuby.javascript(js)). I have also opened an issue for the same. (link)
  - We further need to include the CSS code (in <style> tags) provided by the user as:
    - We will create another attr_accessor css_data just like series_data so that the user can provide the CSS as:

      ```
      @graph.chart.css_data = css_data_user
      ```

    - Just as high_chart method is created to generate the js script of the chart, hight_chart_css method will be created to generate the CSS of the chart. It will be called from the to_html method (link).
    - In high_chart method CSS can be generated directly as:

```
css = “”
css << “<style>”
css << # retrieve css data from object.css_data
css << “</style>”
```

- ❖ Thus, to_html will then return the combined code of CSS and js of the chart.

- ❏ I have tried the styled mode of CSS in a demo rails app (link).
- ❏ Various styling methods can be seen here.

- **Export the chart in various formats:**

  - ❏ In order to export the chart in different formats, we can create APIs like chart.export_png, chart.export_pdf, chart.export_svg and chart.export_jpg so that we can directly download the chart from the code.
  - ❏ The APIs created will further render the template(s) containing the code to first create the chart object and then the code to export the chart.
  - ❏ For that we can use the exportChart method contained in the exporting module as:

```
var chart = $('#container').highcharts();
chart.exportChart({
        type: 'application/pdf',
        filename: 'my-pdf'
});
```

  Possible values of type are image/png, image/jpeg, application/pdf and image/svg+xml, so we can export the chart in all these formats.

- **Adding more features of HighCharts in daru-view:**

  - ❏ There are charts like sankey diagrams, wind-barb charts, sunburst charts, x-range charts and many others which uses additional js dependencies. Also I would be adding more of the examples in daru-view to demonstrate these charts.
  - ❏ The required dependencies are:

```
<script src="https://code.highcharts.com/modules/sankey.js"></script>
<script src="https://code.highcharts.com/modules/windbarb.js"></script>
<script src="https://code.highcharts.com/modules/sunburst.js"></script>
<script src="https://code.highcharts.com/modules/xrange.js"></script>
```

DataTables:

- **Adding more features of DataTables in daru-view:**

The current implementation of the DataTables in daru-view using the gem daru-data_tables includes only the basic initialization features of the DataTables. So there is a lot more scope to add more features to it. Some of them that I want to implement are:

- **Highlighting rows and columns:** It is useful for drawing attention to where the user's cursor is in a table. To add this feature:
  - In the `to_js` method, CSS code necessary for this feature will be added in the style tags such that the table_script (link) will contain first the CSS code for highlighting feature and then the table script.
  - Further, in the draw_js method (link), following js code will be appended.

```
$('#example tbody').on( 'mouseenter', 'td', function () {
        var colIdx = table.cell(this).index().column;
        $( table.cells().nodes() ).removeClass( 'highlight' );
        $( table.column( colIdx ).nodes() ).addClass( 'highlight' );
} );
```

  - A sample html implementation of the above feature is shown here.
- **Retrieve the row data on click:** We can use DataTables `row().data()` API method to retrieve information about the selected row and we can show it in the alert message as:

```
$('#example tbody').on('click', 'tr', function () {
    var data = table.row( this ).data();
    alert( 'You clicked on '+data[0]+'\'s row' );
} );
```

  - User can pass an additional option of `row_click` as true or false. We will then retrieve the value in a variable which can be used to decide whether or not to show this feature.
  - The above code will be implemented in a function as a string which will be included in `draw_js` method if the option is set to true.
  - A sample html implementation of the above feature is shown here.

After the addition of these features, I will work on some more features if the time remains.

- **Load large set of data piece by piece:**

- ❏ To load large set of data piece by piece, we need to use server-side processing in DataTables.
- ❏ Server-side processing can be used to show large data sets, with the server being used to do the data processing, and scroller optimising the display of the data in a scrolling viewport.
- ❏ To implement the server-side processing we would be using the `ajax` option of the DataTables. Ajax option will be used to *fake* the data to show Scroller's ability to show large data sets.
- ❏ We would be implementing the ajax option of the DataTables in the `draw_js` method by explicitly adding the `serverSide` option as true and ajax option as described here.
- ❏ To load the data provided by the user into the ajax function, we could do something like this.
- ❏ I have tried an example for 12 million rows which is working fine (link).

- **Display of DataTable in iruby cell:**

  Currently, `daru-data_tables` gem is not able to show DataTables in iruby notebook. The possible reasons could be:
  - ❏ DataTables' js might not be working on loading and css has not been loaded yet.
  - ❏ There might be a problem with iruby notebook to display output when html input is given.

  I am currently working on this issue and would resolve it as soon as possible.

Updating js files:

- **Remove a bunch of lines at the source html file:**

  To make daru-view workable offline, it is loading the JS files in iruby notebook and web application. But when we see the html source code, we see a bunch of lines at the source html file because of js.

  - ❏ In rails, we can add the following lines in the application.js file:

```
//= require daru-view/highcharts/highcharts
//= require daru-view/highcharts/highcharts-more
//= require daru-view/highcharts/highstock
//= require daru-view/googlecharts
```

  A working example for the same is shown here.

- ❏ In nanoc, we can use `nanoc-javascript-concatenator` gem to concatenate JavaScript files in Nanoc.
  Similar to rails, javascript files can be loaded easily using this.
- ❏ In sinatra, we just have to move our static files(js/css) into a folder named public since sinatra looks there with default settings.

Questions:

- Can we create advanced charts like this by extending the daru-view/highcharts code? (i.e. creating methods. How will the data be send? How those options will be set to modify the charts?)

  Yes, we can create a chart like that by extending the daru-view/highcharts code. Everything is similar except:

  - ❏ The callback() function: We can create another `attr_accessor callback_options` in HighChart class which can be used to build `callback_js`. A hash consisting of path, attr and translate keys can be passed to `callback_options`.
    In the build_html_output method we have to create the `callback_js` also which takes empty string if there is no need for callback.
    We need to change the encapsulate_js method so that it includes `callback_js` also.
  - ❏ Positioner in the tooltip can be passed as "`function (labelWidth, labelHeight) { …}`".`js_code`. Similarly, the `backgroundColor` and the `borderColor` can be passed.
  - ❏ Options and series_data can be passed as normal.

    ```
    gauge.chart.options = opts;
    gauge.chart.series_data = series_dt
    ```

- Is it good idea to have some plugin (plugin in the sense, user can add more available adapters for plotting, using some command like daru-view add_adapter abc_charts)?

  Yes, it is a very good idea to have a plugin like that. However, the implementation of a such a command to add a new adapter is complex and needs time. So, I will be implementing the basic structure of it for now which includes:

  - ❏ Creating the rake task (for developers) which will generate the sample template for the adapter. The template will contain all the necessary methods to add an adapter like div, init_script, init_ruby and export_html_file as TODO.

The generation of the html output will depend on the syntax of the adapter. I will further work on it post the GSoC period.

TIMELINE

Community Bonding [23 April - 14 May]

- Spend this time trying to formalize what exactly I need to code and discuss design decisions with mentor to make sure that no bad decisions are made early in the process. A deliverable here would be an exact spec of what code needs to be written.

[14 May - 07 June]

- This period involves the implementation of HighCharts related work. This includes:
  - ❏ Adding more features in daru-view using HighMaps and HighStock.
  - ❏ Custom styling CSS in HighCharts.
  - ❏ Exporting HighCharts in different formats.
  - ❏ Adding compatibility for more chart types in daru-view.

[08 June - 05 July]

- I aim to complete the google charts related work in this period. This includes:
  - ❏ Implementation of the ChartWrapper and ChartEditor class.
  - ❏ Exporting charts in different formats.
  - ❏ Importing data from google spreadsheets.
  - ❏ Adding more methods of Google DataTables in Daru-view.
  - ❏ Add a feature to show multiple charts.

[06 July - 20 July]

- In this period I would be completing the DataTables related goals and will work on adding examples. This includes:
  - ❏ I will complete the export_html_file and generate_html method in datatables.rb (link) to create the html file. Further, I will work on creating the data array from dataframe and vector (link).
  - ❏ Loading large set of data piece by piece.

    - ❏ I will add examples for web frameworks to demonstrate various features that have been added and various capabilities of daru-view.

[21 July - 5 August]

- Complete the remaining work (if something is left).
- I will work on the implementation of some advanced HighCharts like the circular gauge chart which will include some more feature addition.
- Complete the basic structure for adding plugin, that is, to create the rake task.
- If behind on stuff, then catch up. If not, then continue with plan, with an intent to get some optional ideas started in the future.
- I will try to complete the TODO tasks left in the code which includes:
    - ❏ Improve code quality by optimizing various methods (link) and removing rubocop comments.
- Also, I would be working on reducing the bunch of lines at the source html file because of js.

[6 August - 14 August]

- Pencils down time. Work on final submission and make sure that everything is okay.

After Summer of Code

- I will continue working on daru-view with the target to implement the plugin command to add new adapter. I would be further adding more features to it.
- I will remain active in sciruby community.

Apart from all this, I will regularly write blog posts on my code, advances, problems faced, and solutions explored. I also believe in writing well tested code, so I will write tests for each piece of code I write. I will actively write and improve the documentation of the software.

FURTHER QUESTIONS

- **Long Term Vision for Scientific software:**

Scientific software add their value by providing a closer look at innovation. Scientific software supports research and development in industry as well as academia. It provides value by enabling researchers to widen the scope of their research, get new insights that can lead to novel products, reduce efforts and costs associated with the synthesis of new products, make more efficient use of experimentation and improve the interpretation of results.
One thing that signifies the need of scientific softwares is the large increase in data. Today much larger amounts of data are generated and archived than five and ten years ago may be interpreted as an indirect support for this hypothesis. In the long term we need scientific softwares that can handle large and complex amount of data efficiently.

- **Little description about yourself:**
  I have been bestowed with talents like reading, writing, dancing, sports. I like to spent some time of my day dancing. Also, being the captain and the coordinator of college dance group, I have represented my college on various intercollege fest and dance competition in the past. I am fun loving person who enjoys sense of responsibility. I am a proactive volunteer of GLUG(Gnu/linux user group) of our college. Apart from all this, I have been involved in the designing of annual college magazine.
  I like to live a balanced life giving to everything that matters, whether it's work or my hobby. As my life goal, I always wanted to be a nobel laureate and always believed that education is the key to change the world. Education must be free, unbounded and equal for everyone.

- **One question and its responses that always intrigued me is how programming fits the world of sciences? Is it just a tool that simplifies the process or it's a separate domain of science like any other domain like science of chemicals or organisms.**
  I have been to different responses of people. As quoted by some, programming is more of a method to solve complex problems. On a wider base, it's just a strategy implemented to let computers understand what we want to do.
  As of me, I stick to the point that it's a whole new domain of science that emerged into the light in the recent past. With the development of rigorous AI techniques we are on the verge of developing new brains. Thus misinterpreting programming as just a mere tool is not at all justified.

- **How to get women and people from underrated groups more involved?**
  Getting into programming must be something that should be inherited from the very early age. Girls should make to get along with coding from the school days. Computer gaming could be one such thing that can attract girls towards computer. Sounds a bit awkward but everybody like games. I remember in my early days I was so attracted towards our desktop because of the minesweeper game installed in it. Once you get into a PC you automatically start exploring stuff. It's the start that is most important. Once they get along with stuff and learn a programming language, that's the current time to let them know the importance of open source software and

development. They should know what importance it holds and why their participation matters.

Other than this, little things like a woman getting more allowance than the normal pay scale or offering other additional financial and social benefits to women can significantly increase their participation in this field.

Talking about my personal experience, I worked as a volunteer at GLUG-NITH, the community of open-source enthusiasts and Linux users where I helped organise Software Freedom Day and motivated young women to participate actively in open-source.