Evaluation of Data Visualization Tools

Abirukmani Venkatachalam
abi@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/creativecomponents

Part of the Business Analytics Commons, and the Other Computer Sciences Commons

Recommended Citation
https://lib.dr.iastate.edu/creativecomponents/272

This Creative Component is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Creative Components by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Evaluation of Data Visualization Tools

by

Abirukmani Venkatachalam

A report submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Christopher J Seeger, Co-major Professor
Shashi Gadia, Co-major Professor
Simanta Mitra

Iowa State University
Ames, Iowa
2019
Table of Contents

LIST OF FIGURES ............................................................................................................................ 3
ACKNOWLEDGMENTS ...................................................................................................................... 4
ABSTRACT ....................................................................................................................................... 5
CHAPTER 1 INTRODUCTION ........................................................................................................... 6
  1.1 Background ................................................................................................................................. 6
  1.2 Report Organization ................................................................................................................... 7
CHAPTER 2 COMPARISON OF METRICS ....................................................................................... 8
  2.1 Getting Started - Setup ............................................................................................................... 8
  2.2 Loading your data ....................................................................................................................... 9
  2.3 Other types of Data Sources .................................................................................................... 13
  2.4 Data Wrangling/ Pre-processing ............................................................................................. 15
  2.5 Advanced Analytics ............................................................................................................... 16
  2.6 Visualizing Data ....................................................................................................................... 18
  2.7 Customizations ......................................................................................................................... 26
  2.8 Interactivity ............................................................................................................................... 28
  2.9 Dashboards ............................................................................................................................... 29
  2.10 Community & Support .......................................................................................................... 29
  2.11 Pricing .................................................................................................................................... 29
  2.12 Programming Expertise ......................................................................................................... 30
CHAPTER 3 SUMMARY OF MY VIEWS ............................................................................................ 31
CHAPTER 4 DASHBOARD SCREENSHOTS ..................................................................................... 32
CHAPTER 5 CONCLUSIONS ............................................................................................................ 38
CHAPTER 6 REFERENCES ............................................................................................................... 38
<table>
<thead>
<tr>
<th>Figure</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tableau – Data Loading</td>
<td>9</td>
</tr>
<tr>
<td>Highcharts – Data Loading</td>
<td>9</td>
</tr>
<tr>
<td>API Access in R</td>
<td>10</td>
</tr>
<tr>
<td>API rate limit handling in R</td>
<td>10</td>
</tr>
<tr>
<td>Web Data Connector – Tableau</td>
<td>11</td>
</tr>
<tr>
<td>Wrapper code for API – Tableau</td>
<td>11</td>
</tr>
<tr>
<td>Wrapper code for API – Tableau</td>
<td>12</td>
</tr>
<tr>
<td>API Access in Python</td>
<td>12</td>
</tr>
<tr>
<td>Types of data connections in R</td>
<td>13</td>
</tr>
<tr>
<td>Types of data connections in Tableau</td>
<td>14</td>
</tr>
<tr>
<td>Date column in R</td>
<td>15</td>
</tr>
<tr>
<td>Summary of data frame in R</td>
<td>16</td>
</tr>
<tr>
<td>Calculated Field in Tableau</td>
<td>16</td>
</tr>
<tr>
<td>Tableau forecast</td>
<td>17</td>
</tr>
<tr>
<td>Tableau actual data</td>
<td>17</td>
</tr>
<tr>
<td>Bar plot in R</td>
<td>18</td>
</tr>
<tr>
<td>Code for bar plot in R</td>
<td>18</td>
</tr>
<tr>
<td>Map in R</td>
<td>19</td>
</tr>
<tr>
<td>Word Cloud in R</td>
<td>19</td>
</tr>
<tr>
<td>Code for word cloud in R</td>
<td>20</td>
</tr>
<tr>
<td>Tiles in R</td>
<td>20</td>
</tr>
<tr>
<td>Bar plot in Tableau</td>
<td>21</td>
</tr>
<tr>
<td>Map in Tableau</td>
<td>21</td>
</tr>
<tr>
<td>Tiles in Tableau</td>
<td>22</td>
</tr>
<tr>
<td>Bar plot in Highcharts</td>
<td>22</td>
</tr>
<tr>
<td>Code for Bar plot in Highcharts</td>
<td>23</td>
</tr>
<tr>
<td>Chicago outline map in Highcharts</td>
<td>24</td>
</tr>
<tr>
<td>Map Error in Highcharts</td>
<td>24</td>
</tr>
<tr>
<td>Projections</td>
<td>24</td>
</tr>
<tr>
<td>Tile in Highcharts</td>
<td>25</td>
</tr>
<tr>
<td>Customization in R</td>
<td>26</td>
</tr>
<tr>
<td>Customization cheat sheet in R</td>
<td>26</td>
</tr>
<tr>
<td>Scale y axis in R</td>
<td>27</td>
</tr>
<tr>
<td>Customizations in Tableau</td>
<td>27</td>
</tr>
<tr>
<td>Dashboard – Crime Data - R</td>
<td>32</td>
</tr>
<tr>
<td>Dashboard – Crime Data - Tableau</td>
<td>33</td>
</tr>
<tr>
<td>Dashboard – Crime Data - Highcharts</td>
<td>34</td>
</tr>
<tr>
<td>Dashboard – Movie Data - R</td>
<td>35</td>
</tr>
<tr>
<td>Dashboard – Movie Data - Tableau</td>
<td>36</td>
</tr>
<tr>
<td>Dashboard – Movie Data - Highcharts</td>
<td>37</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I wish to convey gratitude to my advisor, Professor Christopher J Seeger for his guidance and support throughout the course of this project. I would like to thank my committee members Dr. Shashi Gadia and Professor Simanta Mitra for their help and support.
ABSTRACT

Data has a wide impact in all our lives today. Every little piece of information is being put to use – it is being analyzed, trends are looked for and prediction models are built. A very important step in this process is the Visualization of available data. This provides insight and a better comprehension of data at hand for the audience. It is not just limited to domains that move towards machine learning, but also a lot of corporate companies need the aids of data & visualization to analyze metrics such as the performance of a product, its impact over time, etc.

Said that, how could one analyze and visualize their data? There are plenty of tools & languages in the market to do the task for you. To name a few of the well-known ones – PowerBI, Tableau, Google Data Studio, Alteryx, R, Python, Highcharts, D3. Each of them is best suited for use based on the amount of data, the purpose (Pure Visuals vs Combined Analytics), programming expertise, etc. This report focuses on three of the above, namely – R, Tableau and Highcharts.

These tools have been chosen for analysis because they are quite popular and also each one has a varying level of difficulty and skillset requirements. I did not want to include tools or languages such as D3.js or SVGGraph owing to my prior familiarity with those – which might add bias to the comparisons.
CHAPTER 1. INTRODUCTION

1.1 BACKGROUND

One, a crime dataset that reflects reported incidents of crime that occurred in the City of Chicago from 2001 to present (in a comma separated format) fetched from the data.gov is chosen. For the second, the API from TMDB – a user-based movie database which has information almost dating back to 1883 is used. I chose these two databases to maintain a variety in the type of data sources used and also that they had interesting & historical information to visualize.

From loading data to pre-processing it and until putting together a dashboard of visuals – similar tasks are done for both the datasets across the chosen three tools. This helps in comparing and evaluating the performance & support provided by each of them.

Major metrics for comparison that are used

- Setup
- Data Loading
- Data Wrangling
- Analytical Support
- Types of Visualization
- Depth ofCustomization Supported
- Interactivity Levels
- Community Support
- Open source
- Programming Expertise
1.2 REPORT ORGANIZATION

Chapter 2 covers the various metrics that have been experimented with each of the tools and how they compare against each other. It includes metrics such as data loading, wrangling, visualizations and customizations. Also features such as programming expertise, pricing and community support are discussed about. These are followed by a summary of my views in Chapter 3, Screenshots in Chapter 4 and finally conclusions.
## 2.1 GETTING STARTED – SETUP

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R</strong></td>
<td><strong>R Studio</strong></td>
</tr>
<tr>
<td></td>
<td>• IDE - Available in both open source &amp; commercial versions</td>
</tr>
<tr>
<td></td>
<td>• Either for Desktop or Server</td>
</tr>
<tr>
<td></td>
<td>• Pre-Requisite: Install R</td>
</tr>
<tr>
<td></td>
<td>• System Requirements:</td>
</tr>
<tr>
<td></td>
<td>More Cores and RAM for larger datasets &amp; better performance</td>
</tr>
<tr>
<td><strong>Shiny R</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Web framework to build web applications using R</td>
</tr>
<tr>
<td><strong>Tableau</strong></td>
<td><strong>Tableau Desktop</strong></td>
</tr>
<tr>
<td></td>
<td><strong>System Requirements</strong></td>
</tr>
<tr>
<td></td>
<td>• Intel Pentium 4 or AMD Opteron processor or faster</td>
</tr>
<tr>
<td></td>
<td>• 2 GB memory</td>
</tr>
<tr>
<td></td>
<td>• 1.5 GB minimum free disk space</td>
</tr>
<tr>
<td></td>
<td>Also available for Tableau Online/ Tableau Server</td>
</tr>
<tr>
<td><strong>Highcharts</strong></td>
<td>Charting Library built using JS</td>
</tr>
<tr>
<td></td>
<td>• Include required JavaScript files in your web page</td>
</tr>
<tr>
<td></td>
<td>• Web Based, not Standalone like Tableau or R</td>
</tr>
</tbody>
</table>
2.2 LOADING YOUR DATA

**Data source 1**

File Type: CSV  
File Size: 1569518 KB/ Approx. 1.6 GB  
Data Source: https://www.data.gov/  

<table>
<thead>
<tr>
<th>R</th>
<th>Took 4.13 minutes to load csv into an R data frame</th>
</tr>
</thead>
</table>

**Tableau**

![Tableau – Data Loading](image)

- Time to load data as live source: 03.02 minutes  
- Time to extract sheet into Tableau data engine: 01:52 minutes

**Highcharts**

- Highcharts provides a data module that allows for pulling data from CSVs, google sheets and html tables into the 'data' object. The limitation being that the data has a scope which is local to each highchart visual being created

```
>>> str(datetime.datetime.now())
'2019-04-07 17:35:43.919586'
>>> crimeData = pd.read_csv("crimes_-2001_to_present.csv")
>>> str(datetime.datetime.now())
'2019-04-07 17:37:22.998728'
```

![Highcharts – Data Loading](image)

- For the experiments, I used a python flask server to host my visuals. Hence used, python's **pandas module** to load & pre-process data

- Pandas took 1 min and 39 seconds to load the csv into a dataframe
**DataSource 2**

The APIs provided by TMDB have different URL formats for each type of request. That is the data for each graph that we intend to build comes from a different API. We cannot thus pull all the available data at once. Every page returned by the API has 20 rows of data. In cases where we need more data that what is being returned by a single page, we will have to do repetitive calls to the API on subsequent pages. The catch being that the API has a rate limit on the request. Hence while making such requests, we will have to ensure we do not hit the threshold and allow for appropriate cooling times.

Below are shown 2 cases of API access

---

**R**

*A simple API access that returns the trending movies of the day*

- The `jsonlite` library provides a built-in function `fromJSON` to fetch data from APIs.

```r
#Movie Database
#Sort either by popularity, vote count or average vote
dailyTrends <- fromJSON("https://api.themoviedb.org/3/trending/all/day?api_key=3d73c979c88b24f1cdab022c99c322bb6")
dailyTrends <- dailyTrends$results %>% drop_n(NULL) %>%
select(title, vote_average, vote_count, popularity)
```

*Figure 3 API Access in R*

**Another case, where we had to deal with the rate limiting nature of the API**

- We introduce a sleep time of 10 seconds – the cooling period after we are close to the rate limit threshold

```r
pages <- fromJSON("https://api.themoviedb.org/3/discover/movie?api_key=3d73c979c88b24f1cdab022c99c322bb6&primary_release_year=2010")$total_pages
movies2010 <- fromJSON("https://api.themoviedb.org/3/discover/movie?api_key=3d73c979c88b24f1cdab022c99c322bb6&primary_release_year=2010&pages=", page)

limit <- 28
for (page in 1:pages){
  t <- fromJSON(paste0("https://api.themoviedb.org/3/discover/movie?api_key=3d73c979c88b24f1cdab022c99c322bb6&primary_release_year=2010&pages=", page))
  print(page)
}
```

*Figure 4 API rate limit handling in R*

---

**Tableau**

- Tableau provides web data connectors to fetch data from APIs. However, they need a html-js wrapper to be deployed. Common APIs like LinkedIn/Facebook can be accessed at ease since Tableau has built-in wrappers.
• In the case of the TMDB API, I had to write the wrappers that define the fields to be imported. Below is a sample of one of the wrappers written for the same.

```javascript
(function() {
    // Create the connector object
    var myConnector = tableau.makeConnector();

    // Define the schema
    myConnector.getSchema = function(schemaCallback) {
        var cols = [
            {
                id: "id",
                dataType: tableau.dataTypeEnum.float
            },
            {
                id: "title",
                dataType: tableau.dataTypeEnum.string
            },
            {
                id: "vote_count",
                dataType: tableau.dataTypeEnum.float
            },
            {
                id: "vote_average",
                dataType: tableau.dataTypeEnum.float
            },
            {
                id: "popularity",
                dataType: tableau.dataTypeEnum.float
            }
        ];

        var tableSchema = {
            id: "dailyTrend",
            columns: cols
        };

        schemaCallback(tableSchema);
    }
};
```

Figure 5: Web Data Connector - Tableau

Figure 6: Wrapper code for API - Tableau
Logic like the one in R is used for the rate limiting cases.

**Highcharts (used python for the server part)**

- The requests library of python is used to pull data from APIs.
2.3 OTHER TYPES OF DATA SOURCES

These are not the only two types of data sources we might see in common use. One another commonly faced source type is from a database server like SQL. How do these platforms help connect to our database? What are the other types of data sources one could connect to?

**R**

The connections Pane in R Studio helps make ODBC communications to a variety of db servers.

![Figure 9 Types of data connections in R](image)

Alternatively the same could be done with a few lines of code using packages such as such as:

- `DBI, odbc, keyring and pool`
- `Flat files can be pulled using scan()`
- `gs_read() of googlesheets package reads in data from Sheets`
- `read_excel() of readxl fetches data from excel`
- `As we have seen, jsonlite’s fromJSON helps read into dataframes JSON data.`
- `Data from clipboard too could be read into dataframes!`
Tableau

- In Tableau a lot of things happen with the click of the mouse. Here are the various options of data source one could connect to using the Tableau Desktop

![Figure 10 Types of data connections in Tableau](image)

Highcharts

- The JavaScript based charting library parses CVSs, HTML tables & Google sheets into a virtual table of rows & columns. For this experiment, we did not use the inbuilt data module and had opted to use python.

- Since the data module limits the data availability to each chart being built, in very few cases is the data module used. Mostly developers stick to reading and processing data using JS frameworks or Python in our case.
### 2.4 DATA WRANGLING/ PRE-PROCESSING

#### R

- Though datatype conversions were easy to do, would have been more efficient if R had auto detected certain types like datetime.

  ```r
  > class(d$Date)
  [1] "factor"
  
  Figure 11 Date column in R
  ```

- Not always do we need all rows or columns in a dataset. There is a need to filter out what we really need.

- R packages like **dplyr** and **tidyverse** come in very handy to filter, mutate and perform aggregate operations on our data to arrive at a form of data we need for our visuals. One has to put in a little effort to get familiar with the using the libraries. It is definitely worth the efforts!

#### Tableau

- Tableau did a great job of identifying the datatypes in each of the columns – like date fields etc. than when compared to R or Python.

- Filtering, sorting and mutating all happen with certain drags and drops of field names into certain locations in the application. These make the process quicker. The downside being that you are limited by the functionalities provided by the tool and when you want to go ahead customizing your own pre-processing function, you will have to step out of your ‘too comfortable tableau’ mindset.

#### Python/Pandas

- Though datatype conversions were easy to do, would have been more efficient if pandas had auto detected certain types like datetime. Also, it is to be noted that for a large dataset, the conversion into datetime took a little longer than one would expect.

- What better language than python to process your data. There is abundance of built in functions to wrangle data. Also, it is to be observed that pandas and R have a lot of similarities.
2.5 ADVANCED ANALYTICS

R

- R is built for statistics and graphics. It provides a range of techniques like linear and nonlinear modelling, statistical tests, time-series analysis, clustering and classification.
- In plain terms, a simple `summary(dataframe)` provides insight into the distribution of data.

![Figure 12 Summary of dataframe in R](image)

Tableau

- Tableau is more of a Business Intelligence tool – focusing on fetching, transforming and reporting data. It does come with a few models for analysis, however it cannot be a standalone tool for building a statistical model. Empowered by R or python, it could be a better fit for the job.
- One could also add simple calculations, include new columns based on existing ones etc. using the 'Calculated Field' option.

![Figure 13 Calculated Field in Tableau](image)
An interesting option that I noticed is the ‘FORECAST’ option in tableau charts. With a click, it tries to forecast what the next value in the series could be. And comparing the forecasted value with the original data does not reveal huge disparities.

**Figure 14 Tableau forecast**

**Figure 15 Tableau actual data**

**Highcharts**

- Highcharts and the family of libraries it provides us with are for the sole purpose of visualizing and charting data. They do not help analyze and build statistical models. It is the core language/tool such as Python which runs behind it that has to take care of this aspect.
2.6 VISUALIZING DATA

R

- The ggplot package in R is used to create the visuals. Supply ggplot with the data, the aesthetics for the graph and other primitive parameters – it puts together a comprehensible visual.
- Once one gets familiar with the stricture of ggplot options, it is easy to create & render the charts.
- For quick references, there is plenty of support available online and a self-explanatory documentation

Below are some charts generated using ggplot

*Figure 16 Bar plot in R*

```
output$topCrimesGraph <- renderPlotly(
  print(
    ggplot(topTopYear~Year(), aes(x=Primary_Type,y=n,fill=Primary_Type)) +
    geom_bar(stat="identity") + theme(axes.text.x=element_blank()) + ylab("Incidents") + xlab("Crime Type")
  )
)
```

*Figure 17 Code for bar plot in R*
• For creating the map with locations of crime displayed on it – the ggmap library is used. One needs to register with Google Maps to obtain the key & use the same while generating a map.

• The word cloud is generated by a few steps of pre-processing the movie titles. Stop words are removed and term frequencies need to be computed.
The above tile of movies that you see in R, has been created by bringing together a number of Shiny Elements called ‘infoBox’es.

**Tableau**

- It required the least effort to create visuals using Tableau – visuals were charted by dragging necessary columns into the workspace and Tableau suggested possible ways of representing the available data. It is about choosing the correct filters, marks, rows & columns.
In the other tools explored, displaying crime locations on the map of Chicago was a slightly tricky task. However, in Tableau it was more like creating another bar graph. However, it does not display street names etc., like in the case of a Google map. It only shows, the proximity of crimes and overview of the locations.

The other visual is the word cloud of movie titles, I did not have to calculate the word frequencies of the corpus like I had to in R. Using the titles as the measure, filter and size determining factors takes care of the word cloud.
- The movie tiles in Tableau are comprised of a table of texts that on hover display the details.

**Highcharts**

- The library provides templates of code for each of the chart types. Plugging in appropriate data renders our charts.
- Highmaps is a part of Highcharts, that assists in representing geospatial data. It has a set of custom base maps – such as countries and states. Since the data to be plot is for the city of Chicago, I created a base map using Custom GeoJson.
However, it was not feasible to add latitudes & longitudes of our crime locations on the generated map. Trying to do so, gave the error that

**Map does not support latitude/longitude**

The loaded map does not support latitude/longitude functionality. This is only supported with maps from the official Highcharts map collection from version 1.1.0 onwards.

This issue is due to the fact that the projections of our geo-json for Chicago is different from the kind of projections Highcharts uses.
Highmaps had a chart type called the tile map, that seemed to be best suited for the idea of representing movie details as Tiles.

Another issue that I faced while working on Highcharts was that when I imported both the Highmaps & Highcharts library into code, there was a namespace collision issue. To resolve the same, I had to rename the first imported 'Highcharts
2.7 CUSTOMIZATION

R

Customizing factors such as color, axis labels and titles, legend names, position of legends etc. can be accomplished by changing a parameter or by adding an extra line of code.

A few examples to give an idea:

*Removes values displayed in the ticks of x axis and defines axes titles*

```
theme(axis.text.x = element_blank()) + ylab("Incidents") + xlab("Crime Type")
```

*Sets angle of x axis labels, assigns color to y axis text, a size for the points in the graph*

```
ggplot(data = crimeMonthsYearSpecific(), aes(x = month, y = n, color = month)) +
  geom_point(size = 3) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        axis.title.y = element_text(colour = "red")) +
  ylab("Incidents")
```

*Figure 31 Customization in R*

![Customization in R](image_url)

*Figure 32 Customization cheat sheet in R*

*Image Courtesy*

In the graph that represents the popularity, vote count and vote average of today’s trending movies – initially since the three series were of varying distribution - the columns were not well scaled. To fix this, the y axis had to be made logarithmic. This is done by merely adding `scale_y_log10()` to the ggplot graph.

Tableau

Customizations are easy to make, and one does not need a cheat sheet to know how to do those. Tableau is user friendly when it comes to making those changes that you would want.
Highcharts
Each aspect like the title, the axis, pointers, colors etc. are all defined like configurations for the chart and can be modified to how we need them. There is no limit on the kind of customizations you could add and there is enough help online for first timers to find the right field to edit to make a particular change.

In fact, everything but the data field in the highcharts template is some sort of a customization. All of the charts across all three tools could be made to look alike using the power of customizations.

2.8 INTERACTIVITY

R
- In certain cases where we need to get user input to filter and process data accordingly, Shiny has a range of input widgets like sliders, search boxes, radio buttons and the likes. We can add them as sub elements with a few lines of code to the dashboard structure and use the server-UI interaction to process data based on inputs.

- To do a custom ‘Sort by’ operation we will have to use input widgets and code for the same

- Plotly is used to wrap the ggplot graphs – this automatically adds in interactivity like displaying a tool tip on hover and options to zoom/compare or screenshot the chart.

Tableau
- When the charts are rendered on the dashboard, a filter icon appears on the top right. It houses numerous options from where you could decide which field you want to provide the user with the ‘filter’ option of. Also, it lets you choose if the options will have to be a drop-down or a slider etc.,

- For a sort by, you create a new dimension/field called sort by and use it as part of your graphs.

- Interactive tool tips are by default a part of the charts’ tableau produces.

Highcharts
- Filters will have to be constructed using html widgets and linked to the charts using JS functions
- Tool tips are defaulted in highcharts and can be customized
2.9 DASHBOARDS

- All of these individual graphs have been put together into a dashboard/web page.
- In case of R, the R Shiny package has been used. Few rendering crashes and low speed in Shiny were observed.
- In Tableau – sheets can be grouped together to form dashboards.
- And in case of Highcharts, each of the graphs is like a div in a html page and will have to be put together and aligned using html tags. With Html divs, positioning could be a bit of a hassle at time.

2.10 COMMUNITY SUPPORT

<table>
<thead>
<tr>
<th>R</th>
<th>The official site provides numerous resources to learn R from – ranging from webinars, cheat sheets and guides. The RStudio Community is active and easy to find answers to commonly faces issues. Stack Overflow is always there to rescue.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tableau</td>
<td>They provide a set of training videos online, however support is not available for the free community.</td>
</tr>
</tbody>
</table>
| Highcharts | The site hosts demos for all the template types that they provide with links to jsfiddle/codepen that lets one to tweak and experiment with the charts                                                                 |}

2.11 PRICING

| R     | Open source & Commercial  
      | Desktop Commercial comes with priority support  
      | Server Commercial – Admin tools, Enhanced Security, Resource Management |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tableau | 14-day free trial  
          | $70USD/user/month  
          | Tableau Desktop – connects only to local data |
| Highcharts | Open Source  
            | Free as well as Commercial Licensing  
            | Commercial Pricing varies depending on libraries & number of developers |
### 2.12 PROGRAMMING EXPERTISE

<table>
<thead>
<tr>
<th>Tool</th>
<th>Skill Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R</strong></td>
<td>Coding needed. There is a moderate learning curve involved towards using R and Shiny.</td>
</tr>
<tr>
<td><strong>Tableau</strong></td>
<td>No coding skill is required. Except one place where you might have to code being the wrappers for custom web data connectors. That too, is not very hard given the guides and support on the internet.</td>
</tr>
<tr>
<td><strong>Highcharts</strong></td>
<td>It is JavaScript based – you will need to have knowledge of basic HTML, CSS and JS to render charts and plugin data. And since these only provide the templates for the actual charts, the server logic will have to be done using JavaScript frameworks or Python (like in our case – a Flask server)</td>
</tr>
</tbody>
</table>
### 3. SUMMARY OF MY VIEWS ON THE THREE TOOLS

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>Tableau</th>
<th>Highcharts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup Ease</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Data loading &amp; handling (Speeds)</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Data loading &amp; handling ease</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Building Charts</td>
<td>✔</td>
<td>✔️</td>
<td>✔</td>
</tr>
<tr>
<td>Analytics &amp; Statistics Support</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Customizations</td>
<td>✔</td>
<td>✔️</td>
<td>✔</td>
</tr>
<tr>
<td>Interactivity</td>
<td>✔</td>
<td>✔️</td>
<td>✔</td>
</tr>
<tr>
<td>Rendering Dashboard</td>
<td>✔️</td>
<td>✔️</td>
<td>✔</td>
</tr>
<tr>
<td>Programming Expertise</td>
<td>Moderate</td>
<td>None</td>
<td>Moderate</td>
</tr>
<tr>
<td>Community &amp; Support</td>
<td>✔</td>
<td>✔️</td>
<td>✔</td>
</tr>
</tbody>
</table>
4. DASHBOARD SCREENSHOTS

4.1 MOVIES DASHBOARD

4.1.1 Screenshot of complete dashboard from R for Movie dataset
4.1.2 Screenshot of complete dashboard from Tableau for Movie dataset
4.1.3 Screenshot of complete dashboard from Highcharts for Movie dataset
4.2 CRIME DATA

4.2.1 Screenshot of complete dashboard from R for Crime dataset
4.2.2 Screenshot of complete dashboard from Tableau for Crime dataset
4.2.3 Screenshot of complete dashboard from Highcharts for Crime dataset
5 CONCLUSIONS

The three tools under experiments did perform equally well, one taking over the other in different aspects. R was great handling data from APIs, Tableau provided great ease in use, Highcharts did a great job at no cost – no installation involved. If you had to decide which of these tools to use, the answer is it depends on the use case and the type of data at hand.

For large datasets – R scales well provided your system supports it with the needed RAM and performance. If you are looking for more interactive ways to create the visuals rather than code, you must pick Tableau. The comfort comes at a price though. If charting is the only task to be done, Highcharts is a definite yes. They have a variety of templates – also it is lighter – no installations required.

If I had to pick a tool for re-doing the Crime Dataset and the Movie Database – my choices would be Tableau for Crime data and for the API based Movie dataset – I would work on either R or Highcharts.

6 REFERENCES