**Data Visualisation and Analytics**

**(F20DV)**

**By Jordan Walker**

**H00222194**

**Date 19th November, 2018**

Table of Contents

[1. Executive summary 3](#_Toc530391454)

[1.1 Goals For The Project 3](#_Toc530391455)

[1.2 Major Findings and Lessons Learnt 3](#_Toc530391456)

[2. Interface Design: Rationale and Critical Review 4](#_Toc530391457)

[3. Interface Design: Layouts and Interactivity 5](#_Toc530391458)

[3.1 List of Layouts 5](#_Toc530391459)

[3.2 Interaction Between Layouts 6](#_Toc530391460)

[3.2.1 Selecting the Unit of Assessment for the User to View 6](#_Toc530391461)

[3.2.2 Appending Topic Weights to The Scatter Graphs 7](#_Toc530391462)

[4. Software Design 8](#_Toc530391463)

[4.1 Design Overview 8](#_Toc530391464)

[4.2 Use of Design Patterns 9](#_Toc530391465)

[5. Original Contribution Made by the Student Contribution 10](#_Toc530391466)

[5.1 Highlights 10](#_Toc530391467)

[5.2 File by File Description 11](#_Toc530391468)

[5.2.1 Controller.js 11](#_Toc530391469)

[5.2.2 Barchart.js 11](#_Toc530391470)

[5.2.3 Piechart.js 11](#_Toc530391471)

[5.2.4 Scatterchart.js 11](#_Toc530391472)

[5.2.5 Sunburst.js 12](#_Toc530391473)

[6 Source Files 13](#_Toc530391474)

# Executive summary

## 1.1 Goals For The Project

The focus of this report is to describe the application that I have created in relation to the specification for F20 Data Visualisation and Analytics. The Specification stated mandatory goals that were expected alongside a section primarily indicating an “originality” factor which encompasses my own personal goals for the project. In Figure 1 a table of goals has been listed alongside

|  |  |  |
| --- | --- | --- |
| Goal | Completed | Not Completed Description |
| Specification Goals | | |
| Create a client side dashboard for a DoR (Director of Research) | 🗸 | N/A |
| Layout and Interactions should be intuitive | 🗸 | N/A |
| Transitions should allow the user to know what is new, changing or exiting data | 🗸 | N/A |
| Must be able to run in Firefox | 🗸 | N/A |
| DoR’s can easily gain insight into how to best write their next “environmental” entry | 🗸 | N/A |
| Only use Version 4 of the D3 library | 🗸 | N/A |
| Make use of Topic Modelling Analysis | 🗸 | N/A |
| Source Code will be modular | 🗸 | N/A |
| At least three layouts are to be used | 🗸 | N/A |
| Personal Goals | | |
| The dashboard will be style appropriately | 🗸 | N/A |
| It is clear what each layout represents | 🗸 | N/A |
| A Novel Layout is used | 🗸 | N/A |

Figure 1 - Goals, Completions and Improvements

## 1.2 Major Findings and Lessons Learnt

The main lesson and finding after completing this coursework specification is that working with D3 is a challenging however rewarding tool for expressing data in a visual format that is not achievable by other languages. It is a language that can be designed in such a modular fashion that every type of visualization i.e. Bar chart, Scatterplot, Pie Chart etc. can be easily reused once they have initially been set up correctly and offer the advantage of easily interacting with previously implemented layouts upon completion.

The main issue that became apparent and my main lesson learnt is that when using D3 to create dashboards, interactive components and visualisations is it can become easy to get intertwined with designing and thus affects the overall deliverable of the project. Unfortunately I feel I succumb to this issue and thus hindered my ability to move onto more advanced interactions and layouts.

Overall what has been learnt is that when one masters D3.js one can create vibrant visualisations, modular applications that can be re-used to easily showcase other pieces of data and reduces the need to focus on CSS creation to achieve the same results.

# Interface Design: Rationale and Critical Review

For the design of my project I wanted to create an interface that was unique and catered to best meet the needs of the client which is the Director of Research. Initially it was my idea to create a visualization where the Director of Research could easily swap between visualizing Institutions and Units of Assessment. This was to be done by click of a “toggle switch” which allowed for the transition of search type. This became an issue with the amount of layouts and re-use of layouts that the implementation of key aspects even with designing in a modularity became confusing, un-aesthetic and mainly negated the main objectives that were set out in the specification

Because of this I had to re-plan my layout to better accustom the needs of the user and thus I finally settled on a design that I believed would be a vast improvement on my initial implementation and comply with the specifications.

My remodelled interface was designed in a way that I believe causes the user to be intuitive. Firstly they are exposed to the Sunburst navigation system which allows the user to follow a navigation through the nested Main Panel either A, B, C or D and from there they can choose the Unit of Assessment that they want to focus on.

Upon selection the user is presented with the main part of the dashboard which shows the following;

* A Pie Chart Showcasing the Wordcount of each University which has that Unit of Assessment.
* A Scatter Graph which shows the amount of FTE (Full Time Employed) staff in each University for the Unit of Assessment.
* A Bar chart showing the Universities of the Unit of Assessment 4\* Environmental Rating.
* A Blank Bar chart that upon selecting a University from any of the previous Layouts will render the topic as 3 words that are most used in the REF submission of the Universities past submission.

By using this design pattern for the interface I believed that being a Director of Research these are the most useful aspects of data to forming an overview of what to include when next writing their next Ref Entry.

Through this design layout for the interface I believed that being a Director of Research these are the most useful aspects of data to forming an overview of what to include when next writing their next Ref Entry.

However, my main critic on my design was not having the full allocation of time to place into the project. I believe with more work that a better interface design could have been achieved with more time dedicated to explore the wide variety of data that may contribute to the benefit of the Director of Researcher. Like many real life projects I have designed the interface to what I believe would be the most relevant layout to get the data I assumed would be helpful for understanding how to write a REF submission based on my own viewing of the data and how I have interacted with it.

If this project was to be made available to an industry standard a lot more research and testing with writers of REF entries would be undertaken to gain information on how that Director of Research best searches for information from the REF Entry’s.

On further critic I decided for all the items in the dashboard to be big for ease of view, this may of not been the best option as it limits the overall view of the data and if I was to improve this layout this is one of the aspects I would change.

# Interface Design: Layouts and Interactivity

## 3.1 List of Layouts

In the Figure 2 an outline of the Layouts that have been used by myself to complete the project are shown, described and how the data shown to the user.

|  |  |  |
| --- | --- | --- |
| Layout | Description | Clarity to the User |
|  | The Data used in this layout is a hierarchal nested format of data. A Nest function was used to have Main Panel => Unit of Assessment. | As this is a hierarchy of already nested data the Sunburst doesn’t get any new data. However, for the user if they are to hover over the different fields they can see what data they are viewing via the breadcrumb trail. If the user also clicks one of the segments of the sunburst the layout will change as it will zoom into that node showcasing any children. If the user clicks in the centre they will be transferred to the parent node and so on. |
|  | The data used for this Layout is the number of FTE staff per topic per University. The Scatter shows the amount from a pre generated X axis which is set to accommodate the highest amounts of staff. | When data is initially rendered in by the Sunburst all the Scatter Plots are the same colour. By either choosing a new Unit of Assessment or by pressing the buttons to sort the data the User can experience the new data appears on the scatter with a different colour value. |
|  | The Pie Chart data is the amount of words used by each University for the searched for Unit of Assessment. | The User can see that when compiling new Unit of Assessment Data that the width of the pie chart arcs either decreases or increases. |
|  | For the Interface two bar charts were used. The first bar chart showcases the University’s with 4\* ratings for that Unit of Assessment.  The Second bar chart initially takes in no data from the Sunburst navigation. However, once either the other bar chart, scatter or pie it renders the Topic Weights for the Unit of Assessment by the clicked on University. | The Bar charts display data like the scatter graphs and the colour scheme for new, updated or exited data remains the same. This is to give a feeling of consistency throughout the interface and not create a sense of confusion with a multitude of colours. |

Figure 2 List of Layouts, How they are used and also Indication of Transition

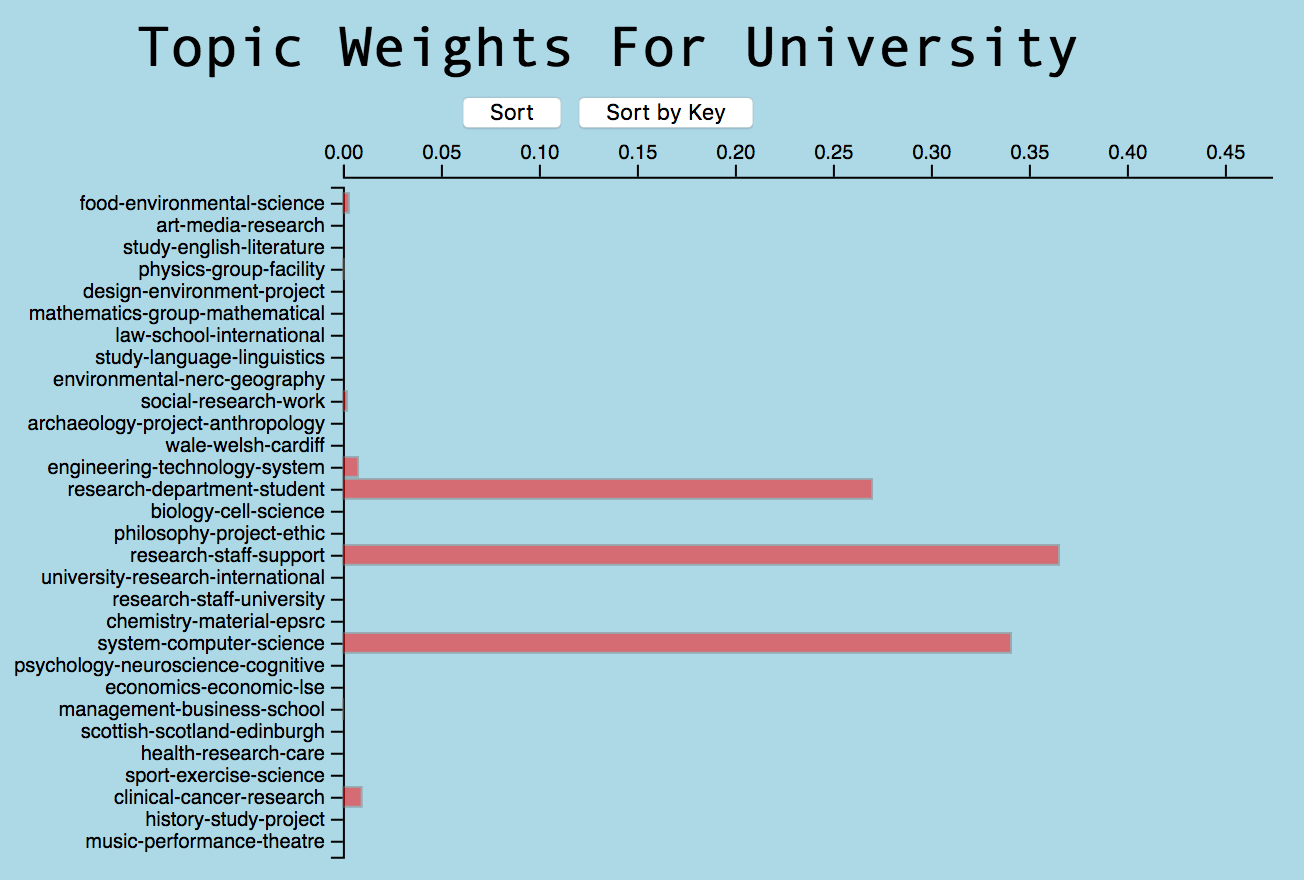
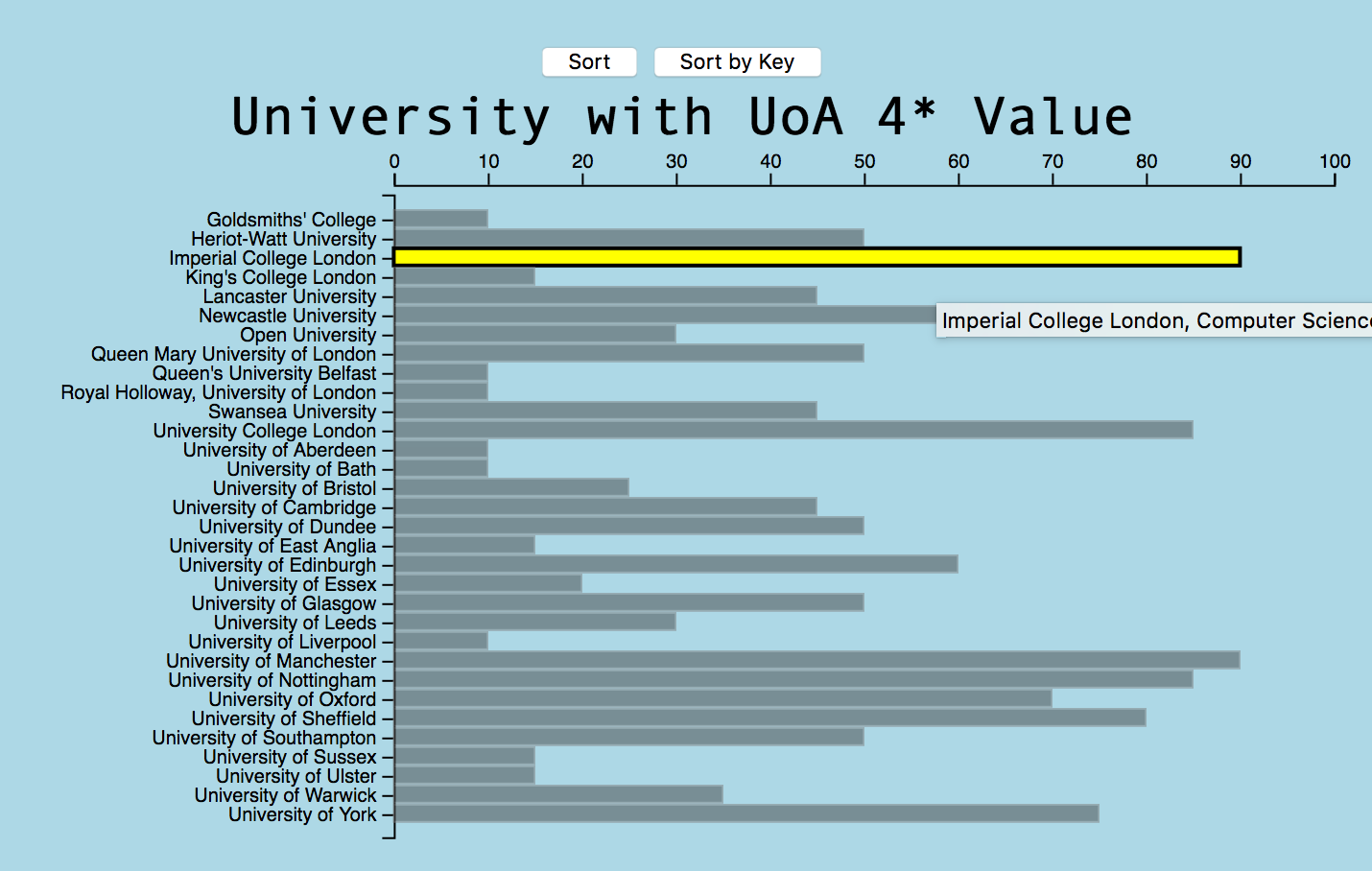
## 3.2 Interaction Between Layouts

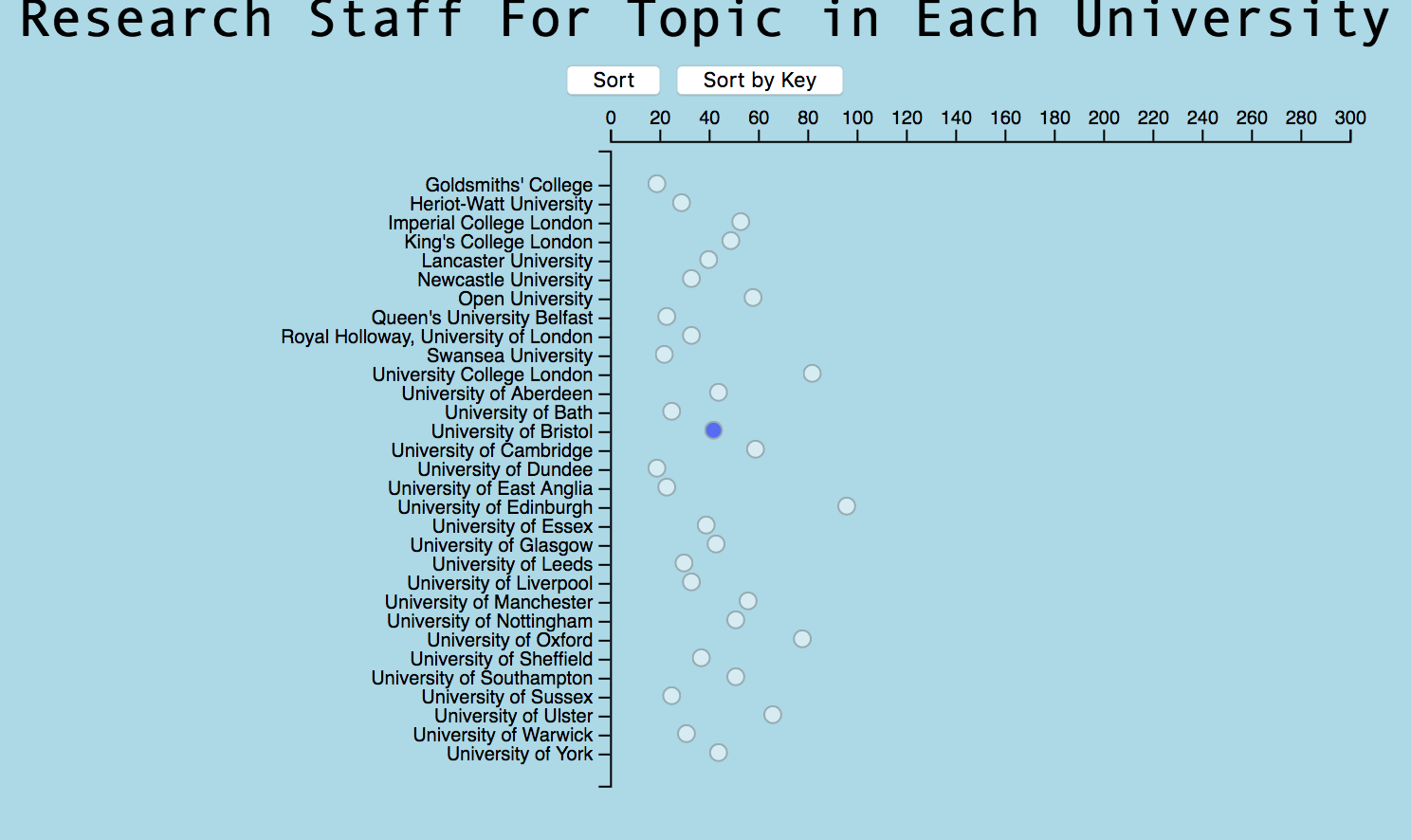
Overall what has been learnt is that when one masters D3.js one can create vibrant visualisations, modular applications that can be re-used to easily showcase other pieces of data and reduces the need to focus on CSS creation to achieve the same results.

### 3.2.1 Selecting the Unit of Assessment for the User to View

When the User enters the dashboard they are first introduced to a Sunburst Layout. The User can then interact with the navigation method proposed and upon clicking the Unit of Assessment they want to view the Sunburst interacts with the pie, scatter and first bar chart to render them with the Unit of Assessment that has been desired.

### 3.2.2 Appending Topic Weights to The Scatter Graphs





To interact with the Topic Weights associated to each University by their Unit of Assessment when either the bar chart or scatter chart is clicked and rendered.

# Software Design

## 4.1 Design Overview

This section showcases how I have structured my overall project and how information is used throughout. It also labels all files associated. I chose this way of design because I believed it was best to not have so much JavaScript and html in the same file. I believed that by constructing my model in this fashion that it would allow me to more easily add new layouts, more easily readable and manageable compared to the initial design of having the controller in the html file.

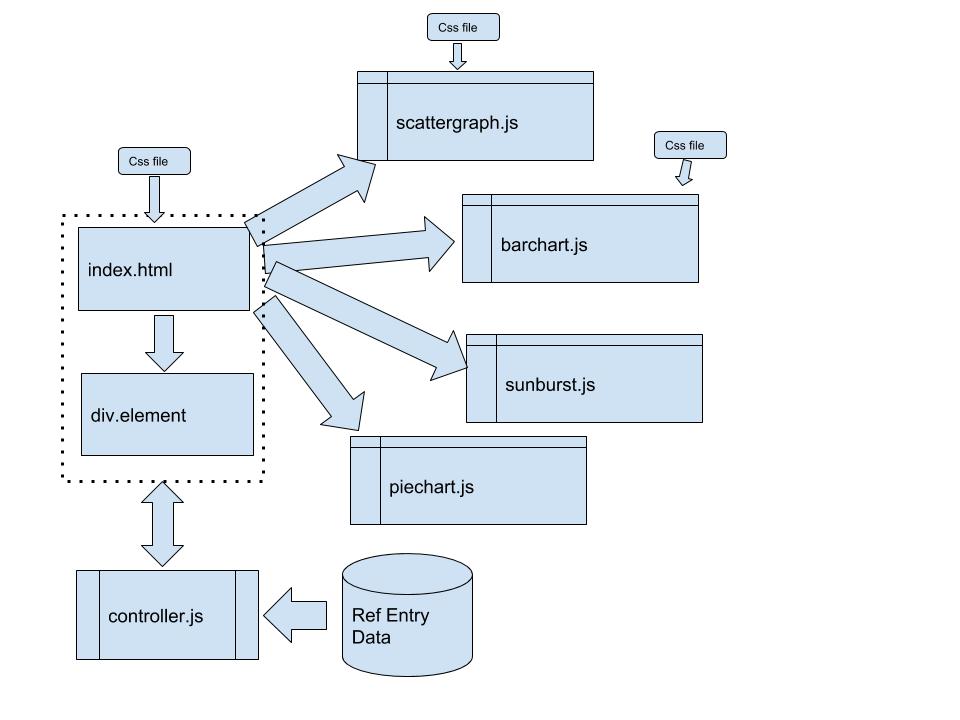


Figure 3 Top Level Design of Application

As Figure 3 shows the controller.js file is the bridge to the entire application working together. Alongside my main reason for implementing the model this way I also believed it would, if I was to continue, add more elements whenever I needed.

Although it follows the same premises as having the controller in the HTML file I believe there was benefits as it was merely a case in the HTML file of declaring a new div element and then in the controller.js file appending a new variable for that div and selecting the data that is needed for the layout from the Data model.

When rendering the data needed for a particular Layout the controller.js file the modules are already available to the developer thus no changes have to be modified in the layout files. When interacting with the Layouts the DOM element is updated as necessary by the controller.js which can handle the runtime interactions needed such as sorting a specific graph.

For styling purpose’s there is a main CSS file that is connected to the index.html file that sets heights, widths and all the necessary items for placement and an overall structured layout. The scatter graph and bar chart have their own CSS file to help with styling these aspects and meaning I can style the plots independently if needed.

## 4.2 Use of Design Patterns

Following from section 4.1 this section is dedicated to explaining the design patterns I have used for my project and also to highlight the advantages and disadvantages of using these patterns.

* + 1. Model View Controller

The model view controller is essentially constructed from three parts aptly a model, a view and a controller and is heavily used in industry for creating user interfaces by utilizing each part that makes up for its entirety.

The model then dictates the logic for the data and prepares the information needed for the user, the view takes the information and presents it in meaningful fashions to the user and finally the command takes in the input of the user and uses that input to change the view or pass new information back to the model.

The model, however, does have its advantages and its disadvantages. Namely the main advantage that I experienced in my project was that it is an implementation that supports asynchronous techniques meaning that the developed application even with all of its on screen modifications of the DOM elements that it still loads and renders fast.

Secondly any modifications that are made do not affect the entire model due to it being separated into its logical parts. This can make for easier debugging, editing and development testing.

However the model does retain certain disadvantages such as when the application scales, poor lack of planning can leave you in a complex situation where the contents of the model don’t make sense anymore. To fully utilise this model thought has to be given initially on how to start and further develop on top of any progress.

One issue that became apparent was that the MVC model created has trouble working with some popular modern browsers such as Chrome. This greatly reduces the “reach” of the finished application as it has to be viewed in Firefox.

4.2.2 Function Factory and Encapsulation

The second technique used was one that is to use the Layouts respective JavaScript file in an Object Orientated Programming way. The problem however, JavaScript doesn’t have classes so instead we use functions to contain all the necessary variables, modifiers etc and by use of method chaining we can see the familiar benefits of polymorphism, encapsulation and inheritance.

The main advantage became clear that by using an encapsulated function for the source files for the layouts we could be sure that the variables contained were kept safe when multiple renders needed to happen at once. This advantage allows for effective reuse of the Layouts previously created and ultimately the data just has to be fed to them.

Opposingly however the main disadvantage that I encountered whilst completing the coursework was that it got confusing very fast. This could be account of my poor function calling but method chaining although advantageous when used right, is complicated when debugging is needed.

# Original Contribution Made by the Student Contribution

## 5.1 Highlights

After completion of the project I was particularly pleased with the aesthetics of the overall design, however, I believe that if I had not tried to over achieve with the initial layout I wanted to follow then I believe I could of made more progress towards a more robust and desirable project.

I am proud nonetheless of the navigation implementation to the Unit of Assessment I have used and adapted however is a novel and unique part of my design and project. The use of a “Sunburst” hierarchy type is what I believe makes my project stand out and what I am most proud of implementing. Adapting this Hierarchal Layout to fit the data was a long process however through doing so has helped with my knowledge of working with the d3.js library.

The Layout as already described in Section 3.1 nests the Ref Entry data by Main Panel and then Unit of Assessment. I am also proud of the interaction the user has with it and given more time I would of explored further possibilities to make it more valuable to the project by either possibly nesting further logical data which for example could be region then by town to show the user which Main Panels and Units of Assessment are exclusive to that particular place. However as a contradiction to this I do also think that it is more appropriate to include all the Universities in the country to gain a comparison against for this particular assignment.

Aside from that I am proud of the data I have chosen to express to the User. It is my understanding from the assignment that does incur some ambiguity as too which data to use in the contribution towards creating a dashboard for a Director of Research to best optimise their next Ref Entry to gain a higher rating. What is meant by this is correlations can be found in many aspects of data which is particularly evident if all the data was used in the creation of the dashboard. It is that then I am proud of using specific quantifiable aspects such as word count, topic-weight, full time employed staff that a more evident correlation appears between these factors and the number of 4\* values associated to the University in that particular Unit of Assessment.

To summarise my highlights I would note that I am overall proud of the completed project I have put together. One of my own goals for this subject and project overall was to implement a novel layout and I have achieved such. In further notes I am also pleased that after hitting a dead end with my initial design choice and effectively “biting off more than I could chew” leaving me with not much time overall that I was able to implement a working dashboard to what I believe a good standard. With more time and the understanding I have gained of the tools and techniques included with d3 I believe a sound deliverable could be achieved.

## 5.2 File by File Description

### 5.2.1 Controller.js

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contribution by student | Contribution from course | Contribution from 3rd party | Source (if 3rd party | Licence (if 3rd party) |
| 40% | **60%** | **N/A** | **N/A** | **N/A** |

The controller.js file is essentially what prepares all my Layouts, Interactions and Data Modelling. It is made up partially of the files given to us in Lab 5 which was used as a starting plot for this project. In the file I have implemented new functions, data nesting, rendering etc. that all assist in the output when the project is loaded in the browser. I wanted to separate the Javascript from the HTML file that was initially given to reduce the mixture of elements and also in regards to the design pattern I have chosen to implement in this product.

### Barchart.js

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contribution by student | Contribution from course | Contribution from 3rd party | Source (if 3rd party | Licence (if 3rd party) |
| 30% | **70%** | **N/A** | **N/A** | **N/A** |

Again this was the initial file barchartWithAxes\_d3v4\_v002.js given to us in Lab 5. I have modified the file however so that it can allow for other data sources to come in namely the “topicAs3words” section of the Ref Model. Changes were also made to certain parts either been removed or edited to allow for interactions.

### Piechart.js

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contribution by student | Contribution from course | Contribution from 3rd party | Source (if 3rd party | Licence (if 3rd party) |
| 50% | **30%** | **20%** | **N/A** | **N/A** |

I have modified a lot of the pie chart file which was originally given for Lab 3. The initial file worked with flat data and thus had to be modified so that it could take in data from the Ref Entry Model. The main contribution was so that the user could interact with the pie chart in a more dynamic way and in result meant a number of new parts had to be included for this desired effect.

### Scatterchart.js

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contribution by student | Contribution from course | Contribution from 3rd party | Source (if 3rd party | Licence (if 3rd party) |
| 40% | **60%** | **N/A** | **N/A** | **N/A** |

The Scatter Graph was originally the barchartWithAxes\_d3v4\_v002.js file however one of the lab sessions was to modify this file into a scatter plot. I have further modified this to again be more in line with how I designed the final project.

### Sunburst.js

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contribution by student | Contribution from course | Contribution from 3rd party | Source (if 3rd party | Licence (if 3rd party) |
| 50% | 10% | 10%  10%  20% | [5552606564ef37b5de7e47ed2b7dc099](https://gist.github.com/maybelinot/5552606564ef37b5de7e47ed2b7dc099)  [1a3f8e44cdcb3054121dfd991f59fbc2](https://gist.github.com/denjn5/1a3f8e44cdcb3054121dfd991f59fbc2)  [766f8f6d31f645c39f488a0befa1e3c8](https://gist.github.com/kerryrodden/766f8f6d31f645c39f488a0befa1e3c8) | [GNU General Public License, version 3](https://opensource.org/licenses/GPL-3.0).  [GNU General Public License, version 3](https://opensource.org/licenses/GPL-3.0).  [Apache License, Version 2.0](http://www.apache.org/licenses/LICENSE-2.0) |

The novel piece of my project was the Sunburst module. It took a long time to get this to work with the Ref Entry Data and as a result many sources were used in its completion. Initially it started as the tree\_v001.js from Lab 5 but was drastically modelled to transition from the Tree Layout that was initially used into the Sunburst. This included adding arcs, new ways of setting the nodes, creating a breadcrumb for the user to understand what they were seeing etc. It is the file that I have spent the most time editing for a single layout but as mentioned in section 4.1 is the part of the project I am most proud of.

# Source Files

## 6.1 Dashboard.html

1<!DOCTYPE html>

2<html lang="en">

3<head>

4 <title>MainDash</title>

5 <meta name="viewport" content="width=device-width, initial-scale=1">

6 <meta charset="utf-8"/>

7 <link rel="stylesheet" href="css/simplecss.css">

8 <link rel="stylesheet" type="text/css" href="css/barchart-v001.css"/>

9 <link rel="stylesheet" href="css/footer-basic-centered.css">

10 <link rel="stylesheet" href="css/scatter.css">

11 <link rel="stylesheet" href="css/pack\_v001.css">

12 <script type="text/javascript" src="d3/d3.v4.js"></script>

13 <script src="lib/model/ref14model\_v002.js"></script>

14 <script src="lib/controller/controller.js"></script>

15 <script src="lib/views/barchart.js"></script>

16 <script src="lib/views/scatterchart.js"></script>

17 <script src="lib/views/piechart.js"></script>

18 <script src="lib/views/sunburst.js"></script>

19</head>

20

21<body>

22<div class="wrapper" id="wrap">

23 <div id = "newpost">

24 <div class = "topwrap">

25 <div id = "mainItem">

26 <h3>Select Topic:</h3>

27 <h3>University with Topic with 4\*</h3>

28 <div id="sequence"></div>

29 <div id="sunburstDiv2"></div>

30 </div>

31 </div>

32

33 <div class ="middlewrap">

34 <div class="box a">

35 <h3>Word Count of Each Institution</h3>

36 <button type="button" onclick="pie2.reverseSortByDataField()">Sort</button>

37 <button type="button" onclick="pie2.sortByKey()">Sort by Key</button>

38 <div ID=piechartdiv2></div>

39 <div id="explanation" style="visibility: hidden;">

40 </div>

41 </div>

42

43 <div class="box b">

44 <h3>Research Staff For Topic in Each University</h3>

45 <button type="button" onclick="scatter3.reverseSortByDataField()">Sort</button>

46 <button type="button" onclick="scatter3.sortByKey()">Sort by Key</button>

47 <div ID=scatterchart3Div></div>

48 </div>

49 </div>

50 </div>

51

52 <div class ="middlewrap">

53 <div class="box a">

54 <button type="button" onclick="bc3.reverseSortByDataField()">Sort</button>

55 <button type="button" onclick="bc3.sortByKey()">Sort by Key</button>

56 <h3>University with UoA 4\* Value</h3>

57 <div ID=barchart3Div></div>

58 </div>

59

60 <div class="box b">

61 <div id = "mainItem">

62 <h3>Topic Weights For University</h3>

63 <button type="button" onclick="bc4.reverseSortByDataField()">Sort</button>

64 <button type="button" onclick="bc4.sortByKey()">Sort by Key</button>

65 <div ID=barchart4Div></div>

66 </div>

67 </div>

68 </div>

69 </div>

70

71<footer class="footer-basic-centered">

72 <p class="footer-company-motto">The "REF"ermandation</p>

73 <p class="footer-company-name">&copy;Jordan Walker Refermandation Task 2018</p>

74</footer>

75

76</div>

77</body>

78

79</html>

## 6.2 Controller.js

1

2 "use strict"

3

4 var dm1 = modelConstructor();

5 var dataModel;

6 var scatter3;

7 var pie2;

8 var bc3;

9 var bc4;

10 var tr2;

11 //=============== READ DATA FILES ================================

12 d3.queue()

13 .defer(d3.csv, "data/topics/REF2014T30TopicOrder.csv")

14 .defer(d3.csv, "data/290183\_REF\_Contextual\_table\_1314\_0.csv")

15 .defer(d3.csv, "data/learning-providers-plus.csv")

16 .defer(d3.json, "data/topics/REF2014T30Python.json")

17 .await(initialiseApp)

18 //======================== MAIN FUNCTION =================================

19 //Carries out all initialization and setup

20 function initialiseApp(error, ref14data, ref14context , learningProviders, jsonTopicData){

21 //Check data files have loaded

22 if (error) {console.log (" there are errror with loading the data: ", error); return;}

23

24 dm1.loadData(ref14data, ref14context , learningProviders, jsonTopicData)

25 dataModel = dm1.model();

26

27 var nest2 = d3.nest()

28 .key(refEntry => refEntry["Main panel"])

29 .sortKeys(d3.ascending)

30 .key(refEntry => refEntry["UoAString"])

31 .sortKeys(d3.ascending)

32 .rollup(values => values)

33 .entries(dataModel.refEntries);

34

35 tr2 = sunburst("#sunburstDiv2")

36 .appendClickFunction(treeClickFunction2)

37 .overrideTooltipFunction(["hello"])

38 .loadAndRenderNestDataset(nest2, "REF")

39

40 scatter3 = scatterchart("#scatterchart3Div")

41 .overrideDataFieldFunction(e => Number(e.context["scaledFTE"]))

42 .overrideKeyFunction(e => e["Institution name"])

43 .overrideMouseClickFunction(clickScatter)

44 .overrideTooltipFunction(e => {return e["Institution name"] + ", " + e.UoAString + ", 4\* = " + e.context["scaledFTE"];})

45 .maxValueOfDataField(300);

46

47 bc3 = barchart("#barchart3Div")

48 .overrideDataFieldFunction(e => Number(e.environment["4\*"]))

49 .overrideKeyFunction(e => e["Institution name"])

50 .overrideMouseClickFunction(clickScatter)

51 .overrideTooltipFunction(e => {return e["Institution name"] + ", " + e.UoAString + ", 4\* = " + e["UoAValue"];})

52 .maxValueOfDataField(100);

53

54 bc4 = barchart("#barchart4Div")

55 .maxValueOfDataField(0.5);

56

57 pie2 = piechart("#piechartdiv2")

58 .overrideDataFieldFunction(e => Number(e.environment["WordCount"]))

59 }

60

61

62 function renderTopicData(topic){

63

64 var bc3Data = dataModel.refEntries

65 .filter(e => e["UoAString"] == topic)

66 .filter(e => e.environment["4\*"] )

67 .sort(function(a, b){return a["Institution name"] > b["Institution name"]})

68

69 var scatter3Data = dataModel.refEntries

70 .filter(e => e["UoAString"] == topic)

71 .filter(e => e.environment["4\*"] )

72 .filter(e => e.context["scaledFTE"] )

73 .sort(function(a, b){return a["Institution name"] > b["Institution name"]})

74

75 var pie2Data = dataModel.refEntries

76 .filter(e => e["UoAString"] == topic)

77 .filter(e => e.environment["4\*"] )

78 .sort(function(a, b){return a["Institution name"] > b["Institution name"]})

79

80 bc3.loadAndRenderDataset(bc3Data);

81 pie2.loadAndRenderDataset(pie2Data);

82 scatter3.loadAndRenderDataset(scatter3Data);

83 }

84

85 function renderTopicWeights(click){

86 console.log("hello dickhead")

87 var bc4Data = click

88 .filter(e => e["UoAString"] == topic)

89 .filter(e => e.environment["4\*"] )

90 .sort(function(a, b){return a["Institution name"] > b["Institution name"]})

91

92

93 }

94

95 function treeClickFunction2(d){

96 //If leaf node then user has clicked on a University

97 //so render that university's data in a barchart

98 if (d.height == 0) {

99 console.log("tree click, d.height, d = ", d.data.key)

100 var topic = d.data.key;

101 renderTopicData(topic)

102 }

103 }

104

105 function clickScatter(d){

106 var unitOfAssessment = d.UoAString;

107 console.log(unitOfAssessment)

108 var click = d.environment.topicsAsArray

109 console.log(click)

110

111 renderTopicWeights(click)

112

113 }

114

115

116

117 function renderTopicWeights(clickedPoint){

118 console.log("im here")

119 console.log(clickedPoint)

120 bc4.loadAndRenderDataset(clickedPoint)

121 }

## 6.3 sunburst.js

1var hierarchyGraph;

2

3 function sunburst(targetDOMelement) {

4 var sunburstObject = {};

5 sunburstObject.addCSSClassesToDOMelements = function(selectors, cssClassName, trueFalse) {

6 selectors.forEach(s => grp.selectAll(s).classed(cssClassName, trueFalse))

7 return sunburstObject;

8 }

9 sunburstObject.loadAndRenderDataset = function(jsonHierarchy) {

10 datasetAsJsonD3Hierarchy = jsonHierarchy;

11 hierarchyGraph = d3.hierarchy(datasetAsJsonD3Hierarchy);

12 addsunburstXYdataAndRender(hierarchyGraph);

13 return sunburstObject;

14 }

15 sunburstObject.loadAndRenderNestDataset = function(nestFormatHierarchy, rootName) {

16 layoutAndRenderHierarchyInNestFormat(nestFormatHierarchy, rootName)

17 return sunburstObject;

18 }

19 sunburstObject.loadAndRenderFlatDataset = function(flatDataset, rootName, keys) {

20 nestFormatHierarchy = createNestFormatHierarchyFromFlatDataset(flatDataset, keys);

21 layoutAndRenderHierarchyInNestFormat(nestFormatHierarchy, rootName)

22 return sunburstObject;

23 }

24 sunburstObject.nodeLabelIfNoKey = function(fn) {

25 nodeLabelIfNoKey = fn;

26 return sunburstObject;

27 }

28 sunburstObject.appendClickFunction = function(fn) {

29 appendClickFunction = fn;

30 return sunburstObject;

31 }

32 sunburstObject.getDatasetAsJsonD3Hierarchy = function() {

33 return datasetAsJsonD3Hierarchy;

34 }

35 sunburstObject.appendedMouseOverFunction = function(callbackFunction) {

36 console.log("appendedMouseOverFunction called", callbackFunction)

37 appendedMouseOverFunction = callbackFunction;

38 return sunburstObject;

39 }

40 sunburstObject.appendedMouseOutFunction = function(callbackFunction) {

41 appendedMouseOutFunction = callbackFunction;

42 return sunburstObject;

43 }

44 sunburstObject.overrideMouseOverFunction = function(callbackFunction) {

45 mouseOverFunction = callbackFunction;

46 render();

47 return sunburstObject;

48 }

49 sunburstObject.overrideMouseOutFunction = function(callbackFunction) {

50 mouseOutFunction = callbackFunction;

51 render(); //Needed to update DOM

52 return sunburstObject;

53 }

54 sunburstObject.overrideTooltipFunction = function(toolTipFunction) {

55 tooltip = toolTipFunction;

56 return sunburstObject;

57 }

58 sunburstObject.render = function(callbackFunction) {

59 render(); //Needed to update DOM

60 return sunburstObject;

61 }

62 var tooltip = function(d) {

63 return d.key + ": " + d.value

64 }

65 var clickFunction = function(clickedNode, i) {

66 clickedNode.xAtEndPreviousGUPrun = clickedNode.x;

67 clickedNode.yAtEndPreviousGUPrun = clickedNode.y;

68 calculateXYpositionsAndRender(hierarchyGraph, clickedNode);

69 appendClickFunction(clickedNode, i);

70 }

71 var appendedMouseOutFunction = function() {};

72 var appendedMouseOverFunction = function() {};

73 var width = 700,

74 height = 700,

75 radius = (Math.min(width, height) / 2) - 10;

76 var formatNumber = d3.format(",d");

77 var x = d3.scaleLinear().range([0, 2 \* Math.PI]);

78 var y = d3.scaleSqrt().range([0, radius]);

79 var color = d3.scaleOrdinal().range(["#f7fcfd", "#e5f5f9", "#ccece6", "#99d8c9", "#66c2a4", "#41ae76", "#238b45", "#006d2c", "#00441b", '#d9d9d9', '#bdbdbd', '#969696']);

80 var partition = d3.partition();

81 var b = {

82 w: 100,

83 h: 70,

84 s: 10,

85 t: 15

86 };

87 var arc = d3.arc().startAngle(function(d) {

88 return Math.max(0, Math.min(2 \* Math.PI, x(d.x0)));

89 }).endAngle(function(d) {

90 return Math.max(0, Math.min(2 \* Math.PI, x(d.x1)));

91 }).innerRadius(function(d) {

92 return Math.max(0, y(d.y0));

93 }).outerRadius(function(d) {

94 return Math.max(0, y(d.y1));

95 });

96 var grp = d3.select(targetDOMelement).append("svg").attr("width", width).attr("height", height).append("g").attr("transform", "translate(" + width / 2 + "," + (height / 2) + ")");

97 var nodesGroup = grp.append("g").classed("nodesGroup", true);

98 var linksGroup = grp.append("g").classed("linksGroup", true);

99 var datasetAsJsonD3Hierarchy;

100 var node;

101 var clickedNode;

102 var listOfNodes;

103 var totalSize = 0;

104 initializeBreadcrumbTrail();

105 //=================== PRIVATE FUNCTIONS ====================================

106 var nodeLabelIfNoKey = function() {

107 return "No name set"

108 };

109 var appendClickFunction = function() {

110 console.log("No click fn appended")

111 };

112 var nodeLabel = function(d) {

113 return d.data.name + "(height:" + d.height + ")";

114 }

115

116 function layoutAndRenderHierarchyInNestFormat(nestFormatHierarchy, rootName) {

117 datasetAsJsonD3Hierarchy = {

118 "key": rootName,

119 "values": nestFormatHierarchy

120 }

121 nodesGroup.append("svg:circle").attr("r", radius).style("opacity", 0)

122

123 function childrenAcessor(d) {

124 return d.values

125 }

126 datasetAsJsonD3Hierarchy = {

127 key: rootName,

128 values: nestFormatHierarchy

129 };

130 hierarchyGraph = d3.hierarchy(datasetAsJsonD3Hierarchy, childrenAcessor).sum(function(d) {

131 return d.value != undefined ? d.value.length : 0

132 });

133 node = hierarchyGraph;

134 nodeLabel = function(d) {

135 if (key) return d.data.key + "(height:" + d.height + ")";

136 else return nodeLabelIfNoKey(d);

137 }

138 calculateXYpositionsAndRender(hierarchyGraph);

139 }

140

141 function arcTweenData(a, i) {

142 var oi = d3.interpolate({

143 x0: a.x0s ? a.x0s : 0,

144 x1: a.x1s ? a.x1s : 0

145 }, a);

146

147 function tween(t) {

148 var b = oi(t);

149 a.x0s = b.x0;

150 a.x1s = b.x1;

151 return arc(b);

152 }

153 if (i == 0) {

154 var xd = d3.interpolate(x.domain(), [node.x0, node.x1]);

155 return function(t) {

156 x.domain(xd(t));

157 return tween(t);

158 };

159 } else {

160 return tween;

161 }

162 }

163

164 function arcTweenZoom(d) {

165 var xd = d3.interpolate(x.domain(), [d.x0, d.x1]),

166 yd = d3.interpolate(y.domain(), [d.y0, 1]), // [d.y0, 1]

167 yr = d3.interpolate(y.range(), [d.y0 ? 40 : 0, radius]);

168 return function(d, i) {

169 return i ? function(t) {

170 return arc(d);

171 } : function(t) {

172 x.domain(xd(t));

173 y.domain(yd(t)).range(yr(t));

174 return arc(d);

175 };

176 };

177 }

178

179 function calculateXYpositionsAndRender(hierarchyGraph) {

180 var mysunburstLayoutGenerator = d3.partition();

181 var hierarchyGraphWithPositions = mysunburstLayoutGenerator(hierarchyGraph);

182 listOfNodes = hierarchyGraphWithPositions.descendants();

183 GUPrenderNodes(listOfNodes);

184 }

185

186 function GUPrenderNodes(listOfNodes) {

187 //DATA BIND

188 var selection = nodesGroup.selectAll("g.cssClassNode").data(listOfNodes, generateUniqueKey);

189 //ENTER

190 var enterSelection = selection.enter().append("path").attr("class", d => {

191 if (d.data.key) return "nest-key--" + d.data.key.replace(/[\W]+/g, "\_");

192 else return "No key";

193 }).attr("fill", d => {

194 while (d.data.height > 1) d = d.parent;

195 return color(d.data.key);

196 }).classed("cssClassNode enterSelection", true).attr("d", arc).on("mouseover", mouseover).on("mouseout", mouseout).style("opacity", 1).on("click", click)

197 nodesGroup.selectAll("path").transition().duration(2000).attrTween("d", arcTweenData)

198 selection.append("title").text(tooltip)

199 }

200

201 function mouseover(d) {

202 var percentage = (100 \* d.value / totalSize).toPrecision(3);

203 var percentageString = percentage + "%";

204 if (percentage < 0.1) {

205 percentageString = "< 0.1%";

206 }

207 d3.select("#percentage").text(percentageString);

208 d3.select("#explanation").style("visibility", "");

209 var sequenceArray = d.ancestors().reverse();

210 sequenceArray.shift(); // remove root node from the array

211 updateBreadcrumbs(sequenceArray);

212 d3.selectAll("path").style("opacity", 0.1);

213 // Then highlight only those that are an ancestor of the current segment.

214 grp.selectAll("path").filter(function(node) {

215 return (sequenceArray.indexOf(node) >= 0);

216 }).style("opacity", 1);

217 }

218

219 function mouseout(d) {

220 console.log("byeJordan");

221 d3.selectAll("path").style("opacity", 1);

222 }

223

224 function click(d) {

225 console.log(d.data.key)

226 if (d.height == 0) clickFunction(d);

227 console.log(d);

228 node = d;

229 nodesGroup.selectAll("path").transition().duration(2000).attrTween("d", arcTweenZoom(d));

230 }

231 var lastKey = 0;

232

233 function generateUniqueKey(d) {

234 if (!d.hasOwnProperty("key")) d.key = ++lastKey;

235 return d.key;

236 }

237

238 function createNestFormatHierarchyFromFlatDataset(flatDataset, keyFunctions) {

239 function applyKey(keyFunction, i) {

240 hierarchy = hierarchy.key(keyFunction);

241 }

242 var hierarchy = d3.nest();

243 keyFunctions.forEach(applyKey);

244 hierarchy = hierarchy.entries(flatDataset);

245 return hierarchy;

246 }

247

248 function initializeBreadcrumbTrail() {

249 // Add the svg area.

250 var trail = d3.select("#sequence").append("svg:svg").attr("width", width).attr("height", 50).attr("id", "trail");

251 // Add the label at the end, for the percentage.

252 trail.append("svg:text").attr("id", "endlabel").style("fill", "#000");

253 }

254 // Generate a string that describes the points of a breadcrumb polygon.

255 function breadcrumbPoints(d, i) {

256 var points = [];

257 points.push("0,0");

258 points.push(b.w + ",0");

259 points.push(b.w + b.t + "," + (b.h / 2));

260 points.push(b.w + "," + b.h);

261 points.push("0," + b.h);

262 if (i > 0) { // Leftmost breadcrumb; don't include 6th vertex.

263 points.push(b.t + "," + (b.h / 2));

264 }

265 return points.join(" ");

266 }

267 // Update the breadcrumb trail to show the current sequence and percentage.

268 function updateBreadcrumbs(nodeArray, percentageString) {

269 // Data join; key function combines name and depth (= position in sequence).

270 var trail = d3.select("#trail").selectAll("g").data(nodeArray, function(d) {

271 return d.data.name + d.depth;

272 });

273 // Remove exiting nodes.

274 trail.exit().remove();

275 // Add breadcrumb and label for entering nodes.

276 var entering = trail.enter().append("svg:g");

277 entering.append("svg:polygon").attr("points", breadcrumbPoints).attr("fill", d => {

278 return color(d.data.key);

279 })

280 entering.append("svg:text").attr("x", (b.w)).attr("y", b.h / 2).attr("dy", "0.35em").text(function(d) {

281 return d.data.key;

282 });

283 // Merge enter and update selections; set position for all nodes.

284 entering.merge(trail).attr("transform", function(d, i) {

285 return "translate(" + i \* (b.w + b.s) + ", 0)";

286 });

287 // Now move and update the percentage at the end.

288 d3.select("#trail").select("#endlabel").attr("x", (nodeArray.length + 1) \* (b.w + b.s)).attr("y", b.h / 2).attr("dy", "0.35em").attr("text-anchor", "middle").text(percentageString);

289 // Make the breadcrumb trail visible, if it's hidden.

290 d3.select("#trail").style("visibility", "");

291 }

292 return sunburstObject;

293 }

## 6.4 Scatterchart.js

1"use safe"

2

3 function scatterchart(targetDOMelement) {

4 var scatterchartObject = {};

5 scatterchartObject.appendedMouseOverFunction = function(callbackFunction) {

6 console.log("appendedMouseOverFunction called", callbackFunction)

7 appendedMouseOverFunction = callbackFunction;

8 render();

9 return scatterchartObject;

10 }

11 scatterchartObject.appendedMouseOutFunction = function(callbackFunction) {

12 appendedMouseOutFunction = callbackFunction;

13 render();

14 return scatterchartObject;

15 }

16 scatterchartObject.loadAndRenderDataset = function(data) {

17 dataset = data.map(d => d); //create local copy of references so that we can sort etc.

18 render();

19 return scatterchartObject;

20 }

21 scatterchartObject.overrideDataFieldFunction = function(dataFieldFunction) {

22 dataField = dataFieldFunction;

23 return scatterchartObject;

24 }

25 scatterchartObject.overrideKeyFunction = function(keyFunction) {

26 GUPkeyField = yAxisCategoryFunction = keyFunction;

27 return scatterchartObject;

28 }

29 scatterchartObject.overrideMouseOverFunction = function(callbackFunction) {

30 mouseOverFunction = callbackFunction;

31 render();

32 return scatterchartObject;

33 }

34 scatterchartObject.overrideMouseOutFunction = function(callbackFunction) {

35 mouseOutFunction = callbackFunction;

36 render(); //Needed to update DOM

37 return scatterchartObject;

38 }

39 scatterchartObject.overrideTooltipFunction = function(toolTipFunction) {

40 tooltip = toolTipFunction;

41 return scatterchartObject;

42 }

43 scatterchartObject.overrideMouseClickFunction = function(fn) {

44 mouseClick2Function = fn;

45 render(); //Needed to update DOM if they exist

46 return scatterchartObject;

47 }

48 scatterchartObject.maxValueOfDataField = function(max) {

49 maxValueOfDataset = max;

50 maxValueOfDataField = function() {

51 return maxValueOfDataset

52 };

53 return scatterchartObject;

54 }

55 scatterchartObject.render = function(callbackFunction) {

56 render(); //Needed to update DOM

57 return scatterchartObject;

58 }

59 scatterchartObject.sortByDataField = function() {

60 dataset.sort((a, b) => dataField(a) - dataField(b))

61 render();

62 return scatterchartObject;

63 }

64 scatterchartObject.reverseSortByDataField = function() {

65 dataset.sort((a, b) => dataField(b) - dataField(a))

66 render();

67 return scatterchartObject;

68 }

69 scatterchartObject.sortByKey = function() {

70 //for security we will use D3's descending operator here

71 dataset.sort((a, b) => d3.descending(GUPkeyField(b), GUPkeyField(a)))

72 render();

73 return scatterchartObject;

74 }

75 scatterchartObject.setTransform = function(t) {

76 //Set the transform on the svg

77 svg.attr("transform", t)

78 return scatterchartObject;

79 }

80 scatterchartObject.yAxisIndent = function(indent) {

81 yAxisIndent = indent;

82 return scatterchartObject;

83 }

84 var mouseClick2Function = function(d, i) {

85 console.log("hello------", d)

86 overrideMouseClickFunction(d, i);

87 }

88 //=================== PRIVATE VARIABLES ====================================

89 //Width and height of svg canvas

90 var svgWidth = 700;

91 var svgHeight = 400;

92 var dataset = [];

93 var xScale = d3.scaleLinear();

94 var yScale = d3.scaleBand();

95 var yAxisIndent = 300;

96 var maxValueOfDataset;

97 var svg = d3.select(targetDOMelement).append("svg").attr("width", svgWidth).attr("height", svgHeight).classed("scatterchart", true);

98 var yAxis = svg.append("g").classed("yAxis", true);

99 var xAxis = svg.append("g").classed("xAxis", true);

100 var mouse

101 var dataField = function(d) {

102 return d.datafield

103 }

104 var tooltip = function(d) {

105 return d.key + ": " + d.datafield

106 }

107 var yAxisCategoryFunction = function(d) {

108 return d.key

109 } //Categories for y-axis

110 var GUPkeyField = yAxisCategoryFunction; //For 'keyed' GUP rendering (set to y-axis category)

111 //=================== OTHER PRIVATE FUNCTIONS ====================================

112 var maxValueOfDataField = function() {

113 //Find the maximum value of the data field for the x scaling function using a handy d3 max() method

114 //This will be used to set (normally used )

115 return d3.max(dataset, dataField)

116 };

117 var appendedMouseOutFunction = function() {};

118 var appendedMouseOverFunction = function() {};

119 var mouseOverFunction = function(d, i) {

120 d3.select(this).classed("highlight", true).classed("noHighlight", false);

121 appendedMouseOverFunction(d, i);

122 }

123 var mouseOutFunction = function(d, i) {

124 d3.select(this).classed("highlight", false).classed("noHighlight", true);

125 appendedMouseOutFunction(d, i);

126 }

127 var mouseClick2Function = function(d, i) {

128 console.log("barchart click function = nothing at the moment, d=", d)

129 };

130

131 function render() {

132 updateScalesAndRenderAxes();

133 GUP\_scatter();

134 }

135

136 function updateScalesAndRenderAxes() {

137 //Set scales to reflect any change in svgWidth, svgHeight or the dataset size or max value

138 xScale.domain([0, maxValueOfDataField()]).range([0, svgWidth - (yAxisIndent + 10)]);

139 yScale.domain(dataset.map(yAxisCategoryFunction)) //Load y-axis categories into yScale

140 .rangeRound([25, svgHeight - 40]).padding([.1]);

141 //Now render the y-axis using the new yScale

142 var yAxisGenerator = d3.axisLeft(yScale);

143 svg.select(".yAxis").transition().duration(1000).delay(1000).attr("transform", "translate(" + yAxisIndent + ",0)").call(yAxisGenerator);

144 //Now render the x-axis using the new xScale

145 var xAxisGenerator = d3.axisTop(xScale);

146 svg.select(".xAxis").transition().duration(1000).delay(1000).attr("transform", "translate(" + yAxisIndent + ",20)").call(xAxisGenerator);

147 };

148

149 function GUP\_scatter() {

150 var selection = svg.selectAll("circle").data(dataset, GUPkeyField);

151 var enterSel = selection.enter().append("circle")

152 enterSel.attr("class", d => ("key--" + GUPkeyField(d))).classed("circle enterSelection", true).classed("highlight", d => d.highlight)

153 enterSel.transition().duration(2000).attr("r", 4.5).attr("cx", function(d) {

154 return yAxisIndent + xScale(dataField(d));

155 }).attr("cy", function(d, i) {

156 return yScale(yAxisCategoryFunction(d)) + (yScale.bandwidth() \* 0.5);

157 });

158 enterSel.append("title").text(tooltip)

159 //GUP UPDATE (anything that is already on the page)

160 var updateSel = selection //update CSS classes

161 .classed("noHighlight updateSelection", true).classed("exitSelection", false).classed("highlight", d => d.highlight)

162 updateSel //update bars

163 .transition().duration(2000).attr("cx", function(d) {

164 return yAxisIndent + xScale(dataField(d));

165 }).attr("cy", function(d, i) {

166 return yScale(yAxisCategoryFunction(d)) + (yScale.bandwidth() \* 0.5);

167 });

168 updateSel //update tool tip

169 .select("title") //Note that we already created a <title></title> in the Enter selection

170 .text(tooltip)

171 //GUP: Merged Enter & Update selections (so we don't write these twice)

172 var mergedSel = enterSel.merge(selection)

173 mergedSel.on("mouseover", mouseOverFunction).on("mouseout", mouseOutFunction).on("click", mouseClick2Function)

174 //GUP EXIT selection

175 var exitSel = selection.exit().classed("highlight updateSelection enterSelection", false).classed("exitSelection", true).remove()

176 };

177 //================== IMPORTANT do not delete ==================================

178 return scatterchartObject; // return the main object to the caller to create an instance of the 'class'

179 } //End of barchart() declaration

## 6.5 piechart.js

1function piechart(targetDOMelement) {

2 var piechartObject = {};

3 piechartObject.overrideDataFieldFunction = function(dataFieldFunction) {

4 dataField = dataFieldFunction;

5 return piechartObject;

6 }

7 piechartObject.overrideMouseOverFunction = function(callbackFunction) {

8 mouseOverFunction = callbackFunction;

9 layoutAndRender();

10 return piechartObject;

11 }

12 piechartObject.overrideMouseOutFunction = function(callbackFunction) {

13 mouseOutFunction = callbackFunction;

14 layoutAndRender();

15 return piechartObject;

16 }

17 piechartObject.render = function(callbackFunction) {

18 layoutAndRender();

19 return piechartObject;

20 }

21 piechartObject.loadAndRenderDataset = function(data) {

22 dataset = data;

23 console.log(dataset);

24 console.log('here');

25 layoutAndRender();

26 return piechartObject;

27 }

28 piechartObject.overrideTooltipFunction = function(toolTipFunction) {

29 tooltip = toolTipFunction;

30 return piechartObject;

31 }

32 piechartObject.overrideMouseClickFunction = function(fn) {

33 mouseClick2Function = fn;

34 render(); //Needed to update DOM if they exist

35 return piechartObject;

36 }

37 piechartObject.reverseSortByDataField = function() {

38 dataset.sort((a, b) => dataField(b) - dataField(a))

39 render();

40 return piechartObject;

41 }

42 //=================== PRIVATE VARIABLES ====================================

43 //Width and height of svg canvas

44 var svgWidth = 700;

45 var svgHeight = 600;

46 var color = d3.scaleOrdinal().range(["#f7fcfd", "#e5f5f9", "#ccece6", "#99d8c9", "#66c2a4", "#41ae76", "#238b45", "#006d2c", "#00441b", '#d9d9d9', '#bdbdbd', '#969696', '#737373', '#525252', '#252525', '#000000', '#ef3b2c', '#cb181d', '#a50f15', '#67000d', '#54278f', '#3f007d']);

47 var dataset = [];

48 var text = "";

49 //=================== INITIALISATION CODE ====================================

50 //Declare and append SVG element

51 var svg = d3.select(targetDOMelement).append("svg").attr("width", svgWidth).attr("height", svgHeight).classed("piechart", true).attr("text-anchor", "middle").style("font", "12px sans-serif");

52 //Declare and append group that we will use tp center the piechart within the svg

53 var grp = svg.append("g");

54 //=================== PRIVATE FUNCTIONS ====================================

55 var dataField = function(d) {

56 return d.dataField

57 }

58 var tooltip = function(d) {

59 return d.UoAString + ": " + GUPkeyField

60 }

61 var key = function(d) {

62 return d.data["UoAString"];

63 };

64 var mouseClick2Function = function(d, i) {

65 console.log("key is==", d.key)

66 console.log("piechart click function = nothing at the moment, d=", d.data["UoAString"])

67 };

68 var appendedMouseOutFunction = function() {};

69 var appendedMouseOverFunction = function() {};

70 //Set up shape generator

71 var arcShapeGenerator = d3.arc().outerRadius(svgHeight / 3).innerRadius(svgHeight / 6).padAngle(0.01).cornerRadius(8);

72 var arcOver = d3.arc().outerRadius(svgHeight / 2).innerRadius(svgHeight / 6).padAngle(0.01).cornerRadius(8);

73 var labelArc = d3.arc().outerRadius(svgHeight / 2).innerRadius(svgHeight / 6);

74

75 function layoutAndRender() {

76 var arcsLayout = d3.pie().value(dataField).sort(null)

77 (dataset);

78 grp.attr("transform", "translate(" + [svgWidth / 2, svgHeight / 2] + ")")

79 GUP\_pies(arcsLayout, arcShapeGenerator);

80 }

81

82 function GUP\_pies(arcsLayout, arcShapeGenerator) {

83 var selection = grp.selectAll("path").data(arcsLayout, key);

84 var selection2 = grp.selectAll("text").data(arcsLayout, key);

85 var enterSel = selection.enter().append("path").style("fill", function(d, i) {

86 return color(i);

87 }).classed("noHighlight", true).classed("highlight", false).on("click", mouseClick2Function).on("mouseover", function(d) {

88 d3.select(this).transition().duration(2000).attr("d", arcOver);

89 }).on("mouseout", function(d) {

90 d3.select(this).transition().duration(2000).attr("d", arcShapeGenerator);

91 }).each(function(d) {

92 this.dPrevious = d;

93 });

94 //GUP ENTER AND UPDATE selection

95 var mergedSel = enterSel.merge(selection)

96 mergedSel.on("click", mouseClick2Function).classed("highlight", true).classed("noHighlight", false);

97 mergedSel.on("click", mouseClick2Function).classed("highlight", false).classed("noHighlight", true);

98 mergedSel.attr("fill", function(d) {

99 return d.data.colour

100 }).on("click", mouseClick2Function);

101 mergedSel.transition().duration(750).attrTween("d", arcTween); //Use custom tween to draw arcs

102 //GUP EXIT selection

103 selection.exit().remove()

104 };

105 //Ignore this function unless you really want to know how interpolators work

106 function arcTween(dNew) {

107 //Create the linear interpolator function

108 //this provides a linear interpolation of the start and end angles

109 //stored 'd' (starting at the previous values in 'd' and ending at the new values in 'd')

110 var interpolateAngles = d3.interpolate(this.dPrevious, dNew);

111 //Now store new d for next interpoloation

112 this.dPrevious = dNew;

113 //Return shape (path for the arc) for time t (t goes from 0 ... 1)

114 return function(t) {

115 return arcShapeGenerator(interpolateAngles(t))

116 };

117 }

118 //================== IMPORTANT do not delete ==================================

119 return piechartObject; // return the main object to the caller to create an instance of the 'class'

120 } //End of piechart() declaration

## 6.6 barchart.js

1 function barchart(targetDOMelement) {

2 var barchartObject = {};

3 barchartObject.appendedMouseOverFunction = function(callbackFunction) {

4 console.log("appendedMouseOverFunction called", callbackFunction)

5 appendedMouseOverFunction = callbackFunction;

6 render();

7 return barchartObject;

8 }

9 barchartObject.appendedMouseOutFunction = function(callbackFunction) {

10 appendedMouseOutFunction = callbackFunction;

11 render();

12 return barchartObject;

13 }

14 barchartObject.loadAndRenderDataset = function(data) {

15 dataset = data; //create local copy of references so that we can sort etc.

16 render();

17 return barchartObject;

18 }

19 barchartObject.overrideDataFieldFunction = function(dataFieldFunction) {

20 dataField = dataFieldFunction;

21 return barchartObject;

22 }

23 barchartObject.overrideKeyFunction = function(keyFunction) {

24 GUPkeyField = yAxisCategoryFunction = keyFunction;

25 return barchartObject;

26 }

27 barchartObject.overrideMouseOverFunction = function(callbackFunction) {

28 mouseOverFunction = callbackFunction;

29 render();

30 return barchartObject;

31 }

32 barchartObject.overrideMouseOutFunction = function(callbackFunction) {

33 mouseOutFunction = callbackFunction;

34 render(); //Needed to update DOM

35 return barchartObject;

36 }

37 barchartObject.overrideTooltipFunction = function(toolTipFunction) {

38 tooltip = toolTipFunction;

39 return barchartObject;

40 }

41 barchartObject.overrideMouseClickFunction = function(fn) {

42 mouseClick2Function = fn;

43 render(); //Needed to update DOM if they exist

44 return barchartObject;

45 }

46 barchartObject.maxValueOfDataField = function(max) {

47 maxValueOfDataset = max;

48 maxValueOfDataField = function() {

49 return maxValueOfDataset

50 };

51 return barchartObject;

52 }

53 barchartObject.render = function(callbackFunction) {

54 render(); //Needed to update DOM

55 return barchartObject;

56 }

57 barchartObject.sortByDataField = function() {

58 dataset.sort((a, b) => dataField(a) - dataField(b))

59 render();

60 return barchartObject;

61 }

62 barchartObject.reverseSortByDataField = function() {

63 dataset.sort((a, b) => dataField(b) - dataField(a))

64 render();

65 return barchartObject;

66 }

67 barchartObject.sortByKey = function() {

68 //for security we will use D3's descending operator here

69 dataset.sort((a, b) => d3.descending(GUPkeyField(b), GUPkeyField(a)))

70 render();

71 return barchartObject;

72 }

73 barchartObject.setTransform = function(t) {

74 //Set the transform on the svg

75 svg.attr("transform", t)

76 return barchartObject;

77 }

78 barchartObject.yAxisIndent = function(indent) {

79 yAxisIndent = indent;

80 return barchartObject;

81 }

82 //=================== PRIVATE VARIABLES ====================================

83 //Width and height of svg canvas

84 var svgWidth = 700;

85 var svgHeight = 400;

86 var dataset = [];

87 var xScale = d3.scaleLinear();

88 var yScale = d3.scaleBand();

89 var yAxisIndent = 200;

90 var maxValueOfDataset;

91 var svg = d3.select(targetDOMelement).append("svg").attr("width", svgWidth).attr("height", svgHeight).classed("barchart", true);

92 var yAxis = svg.append("g").classed("yAxis", true);

93 var xAxis = svg.append("g").classed("xAxis", true);

94 var dataField = function(d) {

95 return d.weight

96 }

97 var tooltip = function(d) {

98 return d.topicAs3words + ": " + d.weight

99 }

100 var yAxisCategoryFunction = function(d) {

101 return d.topicAs3words

102 } //Categories for y-axis

103 var GUPkeyField = yAxisCategoryFunction; //For 'keyed' GUP rendering (set to y-axis category)

104 //=================== OTHER PRIVATE FUNCTIONS ====================================

105 var maxValueOfDataField = function() {

106 //Find the maximum value of the data field for the x scaling function using a handy d3 max() method

107 //This will be used to set (normally used )

108 return d3.max(dataset, dataField)

109 };

110 var appendedMouseOutFunction = function() {};

111 var appendedMouseOverFunction = function() {};

112 var mouseOverFunction = function(d, i) {

113 d3.select(this).classed("highlight", true).classed("noHighlight", false);

114 appendedMouseOverFunction(d, i);

115 }

116 var mouseOutFunction = function(d, i) {

117 d3.select(this).classed("highlight", false).classed("noHighlight", true);

118 appendedMouseOutFunction(d, i);

119 }

120 var mouseClick2Function = function(d, i) {

121 console.log("barchart click function = nothing at the moment, d=", d)

122 };

123

124 function render() {

125 updateScalesAndRenderAxes();

126 GUP\_bars();

127 }

128

129 function updateScalesAndRenderAxes() {

130 //Set scales to reflect any change in svgWidth, svgHeight or the dataset size or max value

131 xScale.domain([0, maxValueOfDataField()]).range([0, svgWidth - (yAxisIndent + 10)]);

132 yScale.domain(dataset.map(yAxisCategoryFunction)) //Load y-axis categories into yScale

133 .rangeRound([25, svgHeight - 40]).padding([.1]);

134 //Now render the y-axis using the new yScale

135 var yAxisGenerator = d3.axisLeft(yScale);

136 svg.select(".yAxis").transition().duration(1000).delay(1000).attr("transform", "translate(" + yAxisIndent + ",0)").call(yAxisGenerator);

137 //Now render the x-axis using the new xScale

138 var xAxisGenerator = d3.axisTop(xScale);

139 svg.select(".xAxis").transition().duration(1000).delay(1000).attr("transform", "translate(" + yAxisIndent + ",20)").call(xAxisGenerator);

140 };

141

142 function GUP\_bars() {

143 //GUP = General Update Pattern to render bars

144 //GUP: BIND DATA to DOM placeholders

145 var selection = svg.selectAll(".bars").data(dataset, GUPkeyField);

146 //GUP: ENTER SELECTION

147 var enterSel = selection //Create DOM rectangles, positioned @ x=yAxisIndent

148 .enter().append("rect").attr("x", yAxisIndent)

149 enterSel //Add CSS classes

150 .attr("class", d => ("key--" + GUPkeyField(d))).classed("bars enterSelection", true).classed("highlight", d => d.highlight)

151 enterSel //Size the bars

152 .transition().duration(1000).delay(2000).attr("width", function(d) {

153 return xScale(dataField(d));

154 }).attr("y", function(d, i) {

155 return yScale(yAxisCategoryFunction(d));

156 }).attr("height", function() {

157 return yScale.bandwidth()

158 });

159 enterSel //Add tooltip

160 .append("title").text(tooltip)

161 //GUP UPDATE (anything that is already on the page)

162 var updateSel = selection //update CSS classes

163 .classed("noHighlight updateSelection", true).classed("highlight enterSelection exitSelection", false).classed("highlight", d => d.highlight)

164 updateSel //update bars

165 .transition().duration(1000).delay(1000).attr("width", function(d) {

166 return xScale(dataField(d));

167 }).attr("y", function(d, i) {

168 return yScale(yAxisCategoryFunction(d));

169 }).attr("height", function() {

170 return yScale.bandwidth()

171 });

172 updateSel //update tool tip

173 .select("title") //Note that we already created a <title></title> in the Enter selection

174 .text(tooltip)

175 //GUP: Merged Enter & Update selections (so we don't write these twice)

176 var mergedSel = enterSel.merge(selection).on("mouseover", mouseOverFunction).on("mouseout", mouseOutFunction).on("click", mouseClick2Function)

177 //GUP EXIT selection

178 var exitSel = selection.exit().classed("highlight updateSelection enterSelection", false).classed("exitSelection", true).transition().duration(1000).attr("width", 0).remove()

179 };

180 //================== IMPORTANT do not delete ==================================

181 return barchartObject; // return the main object to the caller to create an instance of the 'class'

182 } //End of barchart() declaration

## 6.7 simple.css

1 body {

2 background-color: #f6f6f6;

3 }

4 h3 {

5 margin: auto;

6 color: black;

7 text-align: center;

8 font-family: Andale Mono, monospace;

9 }

10 .newpost {

11 display: block;

12 }

13 .newpost2 {

14 display: none;

15 }

16 .header {

17 text-align: center;

18 font-family: Andale Mono, monospace;

19 }

20 .topwrap {

21 margin-top: 40px;

22 text-align: center;

23 }

24 #mainItem {

25 display: block;

26 overflow: auto;

27 width: 95%;

28 background-color: lightblue;

29 margin: auto;

30 }

31 .middlewrap {

32 text-align: center;

33 display: grid;

34 width: 95%;

35 margin-left: auto;

36 grid-template-columns: 30% 65%;

37 grid-gap: 10px;

38 color: #444;

39 padding-top: 10px;

40 overflow: auto;

41 }

42 .bottomwrap {

43 text-align: center;

44 width: 1000px;

45 margin-left: 300px;

46 background-color: lightblue;

47 color: #444;

48 padding-top: 10px;

49 }

50 .box {

51 text-align: center;

52 height: 450px;

53 background-color: lightblue;

54 color: #fff;

55 border-radius: 5px;

56 padding: 20px;

57 font-size: 150%;

58 margin-left: auto;

59 overflow: auto;

60 }

61 footer, p {

62 text-align: center;

63 }

64 canvas {

65 display: block;

66 margin: 0 auto 0 auto;

67 }

68 a {

69 color: #000;

70 }

71 #newpost2 {

72 display: none;

73 }

74 #newpost {

75 display: block;

76 }

77 #searchTag {

78 margin-left: 60px;

79 }

80 #sequence {

81 margin-left: 40%;

82 width: 600px;

83 height: 70px;

84 }

85 #legend {

86 padding: 10px 0 0 3px;

87 }

88 #sequence text, #legend text {

89 font-weight: 600;

90 fill: #000;

91 }

92 #explanation {

93 position: absolute;

94 top: 260px;

95 left: 305px;

96 width: 140px;

97 text-align: center;

98 color: #666;

99 z-index: -1;

100 }

101 #percentage {

102 font-size: 2.5em;

103 }

## 6.8 scatter.css

1

2.scatterchart .circle.enterSelection { /\* Select decendants of class 'barchart' elements where the descendants are both of class 'bars' AND class 'enterSelection' \*/

3 fill: white;

4 stroke: gray;

5 stroke-width: 1;

6 opacity: 0.5;

7}

8

9.scatterchart .circle.updateSelection { /\* Select decendants of class 'barchart' elements where the descendants are both of class 'bars' AND class 'enterSelection' \*/

10 fill: red;

11 stroke: gray;

12 stroke-width: 1;

13 opacity: 0.5;

14}

15

16.scatterchart .circle:hover { /\* Select decendants of class 'barchart' elements where the descendants are both of class 'bars' AND class 'enterSelection' \*/

17 fill: blue;

18 stroke: gray;

19 stroke-width: 1;

20 opacity: 0.5;

21}

22

23/\* .scatterchart .scatters:hover{

24 stroke: black;

25 stroke-width: 2;

26 opacity: 1.0;

27}

28\*/

29

## 6.8 footer-basic-centered.css

1.footer-basic-centered{

2 background-color: #292c2f;

3 box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.12);

4 box-sizing: border-box;

5 width: 100%;

6 text-align: center;

7 font: normal 18px sans-serif;

8

9 padding: 45px;

10 margin-top: 80px;

11}

12

13.footer-basic-centered .footer-company-motto{

14 color: #8d9093;

15 font-size: 24px;

16 margin: 0;

17}

18

19.footer-basic-centered .footer-company-name{

20 color: #8f9296;

21 font-size: 14px;

22 margin: 0;

23}

24

25.footer-basic-centered .footer-links{

26 list-style: none;

27 font-weight: bold;

28 color: #ffffff;

29 padding: 35px 0 23px;

30 margin: 0;

31}

32

33.footer-basic-centered .footer-links a{

34 display:inline-block;

35 text-decoration: none;

36 color: inherit;

37}

38

## 6.9 barchart-v001

1

2.barchart .bars.enterSelection {

3 fill: #444;

4 stroke: gray;

5 stroke-width: 1;

6 opacity: 0.5;

7}

8

9.barchart .bars.updateSelection {

10 fill: red;

11 stroke: gray;

12 stroke-width: 1;

13 opacity: 0.5;

14}

15

16.barchart .bars.exitSelection {

17 fill: gray;

18 stroke: gray;

19 stroke-width: 1;

20 opacity: 0.5;

21}

22

23.barchart .bars.highlight {

24 fill: yellow;

25 stroke: black;

26 stroke-width: 2;

27 opacity: 1.0;

28}

29

30

31