F20DV Data Analytics & Visualisation Coursework

Matthew Reilly

19/11/2018

Table of Contents

1. Executive Summary

The goals of this project were to understand and learn who to implement a visualisation tool which is intuitive and easy to use in d3.js to help a Direct of Research (DoR) in deciding the best way to write their next REF 5 submission in order to maximise their REF Environment Assessment. The way this project help the DoR is by letting them examine other REF 5 submissions from other universities in a visual way to help them understand the data more clearly.

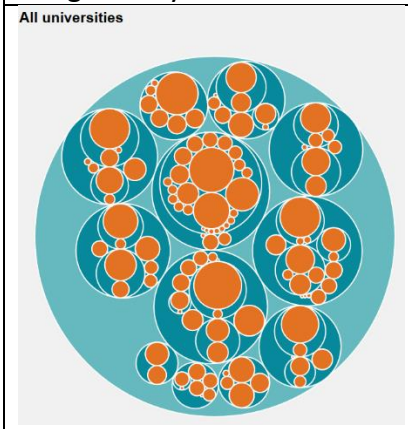
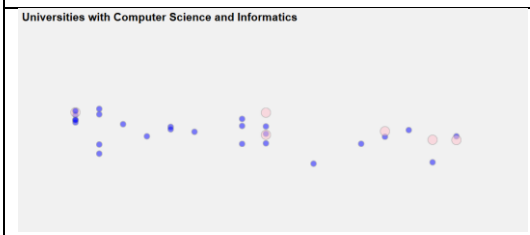
The lessons that I have learned from this project are that data visualization helps immensely in the ability to understand the given data. I have also learned that d3.js is a very difficult language to understand and use when you first start using it and it takes time to get use effectively.

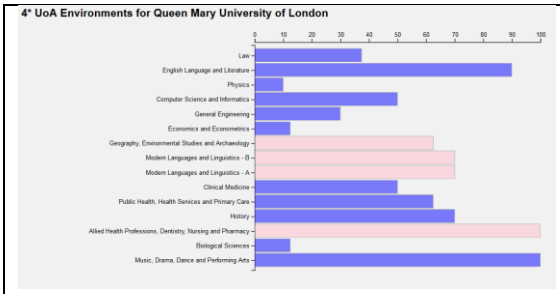
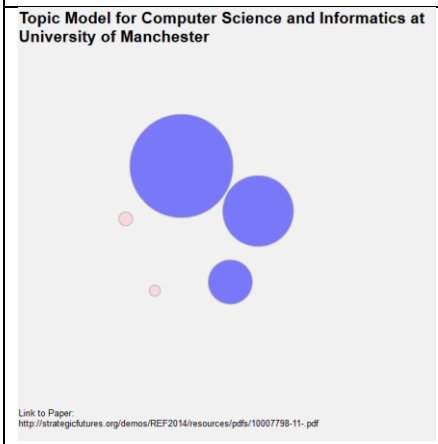
2. Interface Design: Rationale

The rationale behind my design is that on the top half of the screen the DoR will be able to look at a university and find the UoA they are interested in then they can look at the bottom half of the screen which will have all the universities that have that UoA in a scatterplot so the DoR can see what universities have a high percentage and word count more clearly. The DoR can then look at the topic weights of the best universities to find papers which will help them in their next REF 5 submission.

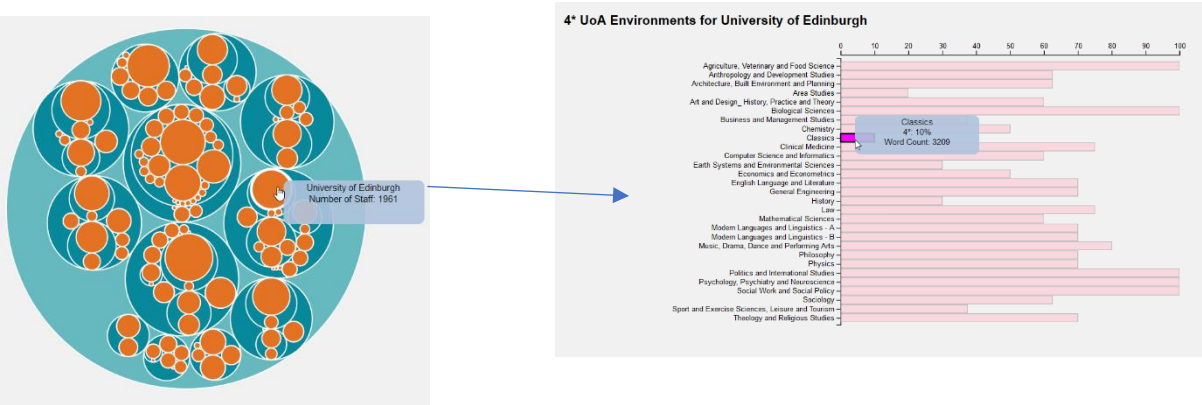
3. Interface Design: Layouts and Interactivity

3.1 List of Layouts

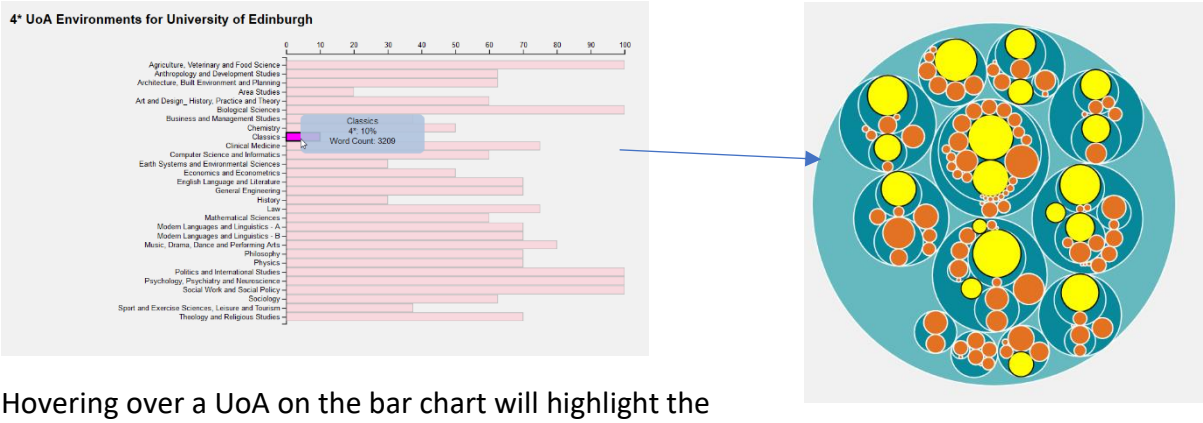
Image of Layout	Description of data used	Change to Data
	The data used in the circle pack is all the universities with at least one 4* unit of assessment environment. The circle is structured by separating the clusters for each region then each town. The size of each circle represents the number of staff at the university	This layout is static, I original had this layout updating with each UoA but I found it wasn't necessary.
	The data used in the scatterplot is the collection of all that universities with a specified UoA showing 4* percentage and word count. Does not show 0% 4*. X axis is 4* percentage and Y axis is word count.	When the data of the scatterplot is changed the user is shown by first the data that is not being used anymore turning grey and fading out, then the data this is staying on the graph changes from pink to blue, reduces in size slightly and moves to its new position. Finally the new data enters the graph pink and goes to the correct position. The title above the scatterplot also changes to let the user know what UoA they are looking at.

	The data used in this bar chart is the percentage of 4* for a UoA from a specified university that is not 0%	When the data of the bar chart changes the user is shown by first removing the unused data by changing it to grey and fading it out, then the data that is staying on the graph changes to blue and moves to the new y position, at this point the y axis is also updated with the new labels. Finally the new bars are added to the chart in pink. The title above the bar chart able changes to show that university is being displayed
	The data used in this force-directed diagram is the topic weights above 2% similarity for a UoA in a university which is represented by the size of the circles. A URL for the paper is also given.	When the data in the force diagram is changed the unused data is removed by changing to grey and shrinking in size, data that is staying changes to blue and resizes and new data is entered pink. The URL at the bottom updates and the title at the top changes to show the UoA and university.

3.2 Interaction between Layouts

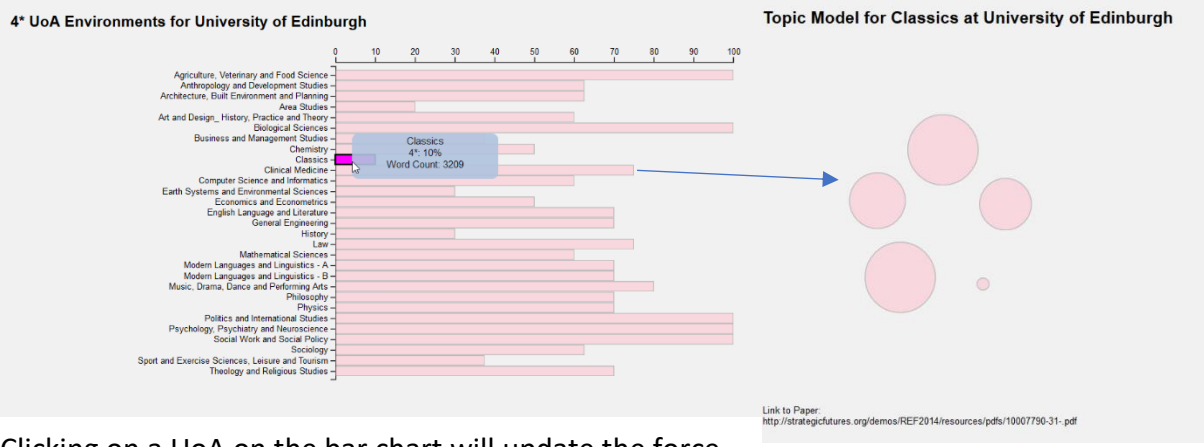


Clicking on a leaf circle on the circle pack will bring up a bar chart with all the units of assessment that the university has showing the percentage of 4*. Hovering over a circle on the circle pack will show the name of the university and the number of staff. Hovering over the bar chart will show the name of the UoA, 4* percentage and the word count of the paper.



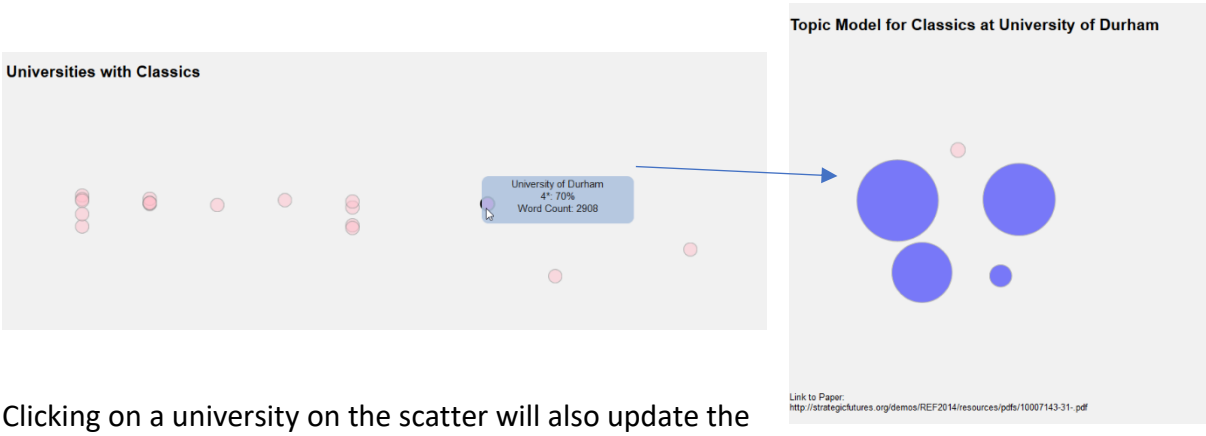
Hovering over a UoA on the bar chart will highlight the other universities on the circle pack which also have that UoA.

Clicking on a UoA on the bar chart will bring up a scatter graph showing all the universities that have that unit of assessment.

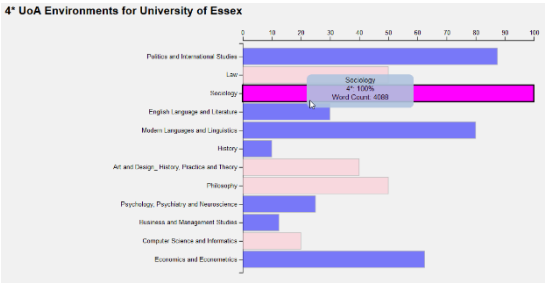


Clicking on a UoA on the bar chart will update the force-directed graph showing the topic weights from that UoA and university.

Clicking on a university on the scatterplot will update the bar chart with the new UoA of that university.



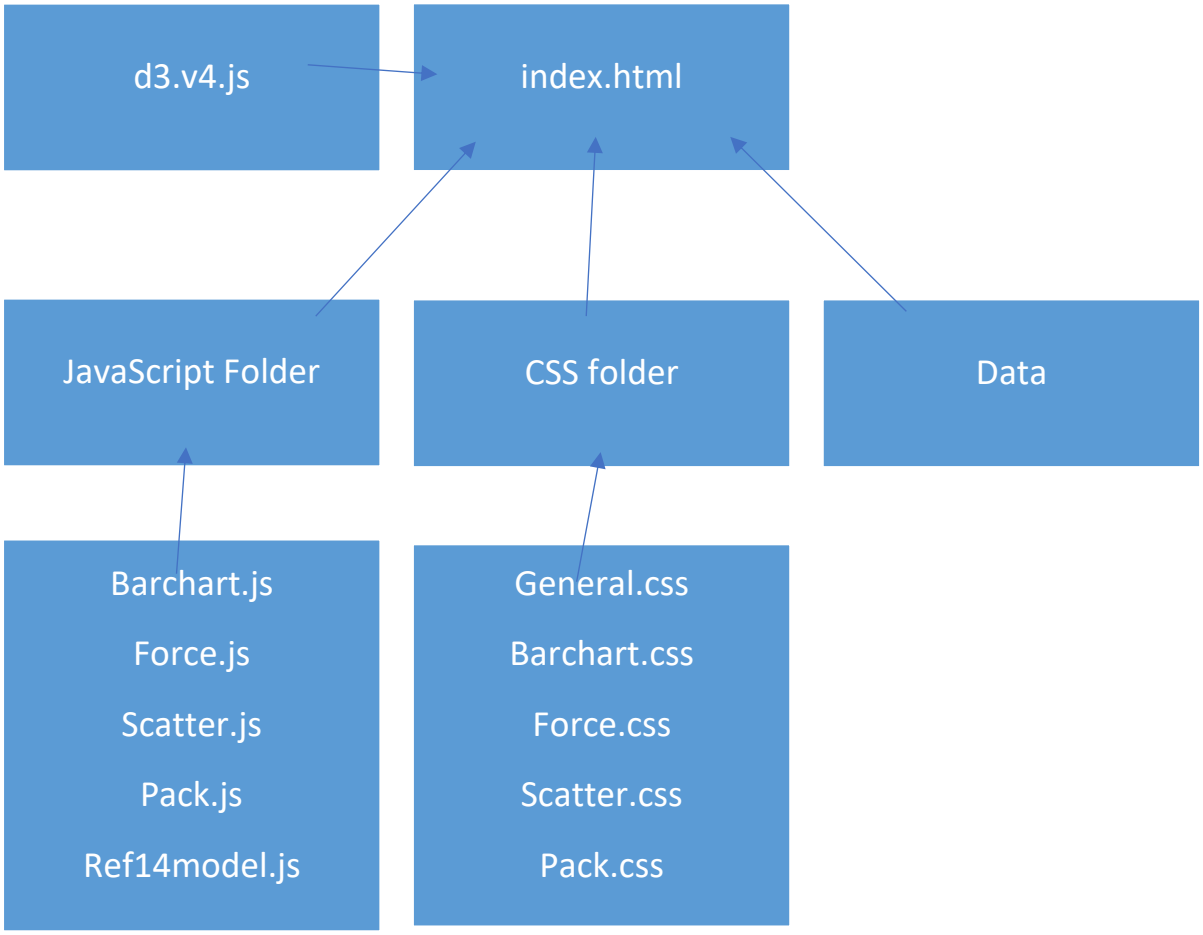
Clicking on a university on the scatter will also update the force-directed diagram with the new universities data.



Hovering over a UoA on the bar will highlight that university on the scatter with the other universities.

4. Software Design

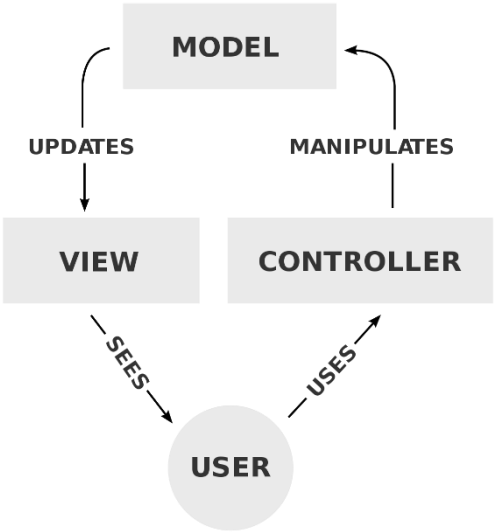
4.1 Design overview



The main.html is what is ran and all the js, css and data file are imported here.

4.2 Use of Design Patterns

My project makes use of the model view controller pattern as if the user performs an action on the dashboard the action will be sent the controller (index.html) when the data will be processed and sent to the relevant model (JavaScript file) where the updates will be applied to the and displayed to the user



5. Student Contribution

5.1 Highlights

- I created a hover tool tip that shows information's about what is on the layout. The tooltip is the same design for every layout on the dashboard.
- I modified a force-directed diagram from Mike Bostock's example.

5.2 File by File description

5.2.1 index.html

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
70%	30%	0%	----	----

Html done, declaration of variables and layouts, scatterClick and barClick functions, changing of the of titles above each layout, tooltip.

5.2.2 scatter.js

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
30%	70%	0%	----	----

Append functions, general update pattern, classes, styling.

5.2.3 force.js

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
50%	0%	50%	https://gist.github.com/mbostock/0adcc447925ffae87975a3a81628a196	`license: gpl-3.0`

Clear zeros function, get size function, simulation declare, get data function, transitions, classes, tooltip.

5.2.4 barchart.js

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
10%	90%	0%	----	----

Classes, click function

5.2.5 pack.js

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
40%	50%	10%	https://bl.ocks.org/mbostock/ca5b03a33affa4160321	GNU Gener

				al Public License, version 3.

General Update Pattern, tooltip

5.2.6 ref14model_v002.js

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
0%	100%	0%	----	----

5.2.7 general.css

Contribution by student	Contribution by course	Contribution by 3rd party	Source (if 3rd party)	Licence (if 3rd party)
100%	0%	0%	----	----

General styling for html and tooltips, other CSS files for each JavaScript file.

6. Source files

6.1 index.html

```
<!------------------------

Authors: Matthew Reilly, Mike Chantler
19/11/2019
What it does:
main html that is ran
all JavaScript, CSS and data file are link to here.

----------------------->
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<title>ref14 data</title>
	<link rel="stylesheet" type="text/css" href="css/pack.css"/>
	<link rel="stylesheet" type="text/css" href="css/barchart.css"/>
	<link rel="stylesheet" type="text/css" href="css/force.css"/>
	<link rel="stylesheet" type="text/css" href="css/general.css"/>
	<link rel="stylesheet" type="text/css" href="css/scatter.css"/>
	<script type="text/javascript" src="d3/d3.v4.js"></script>
	<script src="lib/model/ref14model_v002.js"></script>
	<script src="lib/views/scatter.js"></script>
	<script src="lib/views/barchart.js"></script>
	<script src="lib/views/pack.js"></script>
	<script src="lib/views/force.js"></script>
</head>

```

```
<body>
<section>
  <nav>
                <h3>All Universities</h3>
                <div id="packDiv"></div>
                <h3><div id="forceTitle"></div></h3>
                <div id="forceDiv"></div>
                <div id ="forcetext1"></div>
<div id="forceHyperlink"></div>
        </nav>
        <article>
                <h3><div id="barTitle"></div></h3>
                <div id="barchart1Div"></div>
        </article>
        <article>
    <h3><div id="scatterTitle"></div></h3>
                <div id="myscatterplotDiv1"></div>
        </article>
        </section>


<script type="text/javascript">
"use strict"

var dm1 = modelConstructor(); //Create datamodel object (gives access to methods in ref14model.js
etc )
var dataModel; //shorthand for dm1.model() and declared as nasty outer block variable for easy
access from console.
var bc1;
var pc1;
var f1;
var sp1;
```

```
var cdata;
var sdata;
var fdata;
var div = d3.select("#myscatterplotDiv1").append("div")
                    .attr("class", "tooltip")
                    .style("opacity", 0);
//=============== READ DATA FILES ================================


d3.queue()
        .defer(d3.csv, "data/topics/REF2014T30TopicOrder.csv")
        .defer(d3.csv, "data/290183_REF_Contextual_table_1314_0.csv")
        .defer(d3.csv, "data/learning-providers-plus.csv")
        .defer(d3.json, "data/topics/REF2014T30Python.json")
        .await(initialiseApp);

//======================== MAIN FUNCTION ================================
//Carries out all initialization and setup
function initialiseApp(error, ref14data, ref14context , learningProviders, jsonTopicData){
        //Check data files have loaded
        if (error) {console . log (" there are errror with loading the data: ", error); return;}

        //Create data model
        dm1.loadData(ref14data, ref14context , learningProviders, jsonTopicData);
        dataModel = dm1.model();
        dataModel = dataModel.refEntries.filter(e=>e.environment["4*"] > 0);

        //Layout and render flat data as pack
        var nest = d3.nest()
                    .key(refEntry => refEntry.context.regionProvider)
                            .sortKeys(d3.ascending) //sort a-z
```

```
                .key(refEntry => refEntry.lp.TOWN)
                        .sortKeys(d3.ascending)
                .key(refEntry => refEntry["Institution name"])
                        .sortKeys(d3.ascending)
                .rollup(function (e) {return d3.sum(e,e=>e.context.scaledFTE);}) //add rollup to
campact leaves and store refEntry info
                .entries(dataModel);


        pc1 = pack("#packDiv")
                .appendClickFunction(packClickFunction)
                .loadAndRenderNestDataset(nest, "REF2014");


                //Create and load scatterplot1
                sp1 = scatterplot("#myscatterplotDiv1")
                        .overrideKeyFunction(e => e["Institution name"])
                .appendedMouseOverFunction(highligtpackNodesOfUniversitiesWithThisUni)
                .appendedMouseOutFunction(removeHighlightingUni)
                .appendedClickFunction(scatterClick)


        //Create barchart
        bc1 = barchart("#barchart1Div")
        .overrideDataFieldFunction(e => Number(e.environment["4*"])) //Use the 4* assessment as
the bar size
        .overrideKeyFunction(e => e["UoAString with Multiple submission letter appended"])  //GUP
key and y-axis category  //        .overrideTooltipFunction(e => {return e["Institution name"] + ", " +
e.UoAString + ", 4* = " + e["4*"];})
                .appendedMouseOverFunction(highligtpackNodesOfUniversitiesWithThisUoA)
                .appendedMouseOutFunction(removeHighlightingUoA)
                        .overrideMouseClickFunction(barClick)
```

```
                .maxValueOfDataField(100);


                f1 = force("#forceDiv")

                var institutionClassesToHighlight; //Remember what we have highlighted so that we can remove the highlighting
                var uoAtoHighlight;
//bar chart clicked, update force and scatter
        function barClick(d){
                var refEntriesWithThisUoA = dataModel
                        .filter(e=>e.UoAString == d.UoAString);
                console.log(d)

                        sp1.loadAndRenderDataset(refEntriesWithThisUoA)


                var     data = dataModel.filter(e=>e.DocumentID == d.DocumentID)
                data = d3.nest().entries(data)
                f1.loadAndRenderDataset(data[0].environment.topicWeights)

                d3.selectAll("#forceTitle")
                        .text("Topic Model for "+d.UoAString+" at "+data[0]["Institution name"]);
                d3.selectAll("#forceHyperlink")
                        .text(d.environment.URL);

                d3.selectAll("#forcetext1")
                        .text("Link to Paper: ");
                d3.selectAll("#scatterTitle")
                        .text("Universities with "+ d.UoAString);
        }
```

```
        //scatter clicked, update force and barchart
        function scatterClick(d){
                var refEntriesWithThisUoA = dataModel
                        .filter(e=>e["Institution name"] == d["Institution name"]);
                console.log(d)


                                bc1.loadAndRenderDataset(refEntriesWithThisUoA)


                var     data = dataModel.filter(e=>e.DocumentID == d.DocumentID)
                data = d3.nest().entries(data)
                f1.loadAndRenderDataset(data[0].environment.topicWeights)
                d3.selectAll("#barTitle")
                                .text("4* UoA Environments for "+d["Institution name"]);
                d3.selectAll("#forceTitle")
                                .text("Topic Model for "+d.UoAString+" at "+data[0]["Institution
name"]);
                d3.selectAll("#forceHyperlink")
                                .text(d.environment.URL);

                d3.selectAll("#forcetext1")
                                .text("Link to Paper: ");
        }

//hightlight pack and scatter
        function highligtpackNodesOfUniversitiesWithThisUoA(d){
                //Get UoA name of clicked bar
                var clickedUoA = d.UoAString;

                //Get list of REF entries with this UoA
```

```
186                     var refEntriesWithThisUoA = dataModel
187                             .filter(e=>e.UoAString == clickedUoA);
188                     //Extract list of cleaned university (institution) class names
189                     institutionClassesToHighlight = refEntriesWithThisUoA
190                             .map(function(e){
191                                     return ".nest-key--"+e["Institution name"].replace(/[\W]+/g,"_")
192                             })
193
194                             uoAtoHighlight=".nest-key--"+d["Institution name"].replace(/[\W]+/g,"_");
195
196                     institutionClassesToHighlight.forEach(function(institutionClass){
197                             d3.selectAll(institutionClass).classed ("highlight", true)
198                     })
199
200
201                     d3.selectAll(".key--"+d["Institution name"].replace(/[\W]+/g,"_")) //select all DOM
202 elements with class "key--<d.keyField>"
203                                     .classed("highlight", true)
204
205                     div.transition()
206                     .duration(200)
207                     .style("opacity", 0.9);
208                     div.html(d.context["Unit of assessment name"]+ "</br>"+"4*:
209 "+d.environment["4*"]+"%" + "<br/>"+ "Word Count: " + d.environment.WordCount)
210                     .style("left", (d3.event.pageX) + "px")
211                     .style("top", (d3.event.pageY - 28) + "px");
212             }
213 //remove hightlighting on pack and scatter
214             function removeHighlightingUoA (d){
215                     institutionClassesToHighlight.forEach(function(institutionClass){
216                             d3.selectAll(institutionClass).classed ("highlight", false)
217                     })
```

```
                d3.selectAll(".key--"+d["Institution name"].replace(/[\W]+/g,"_")) //select all DOM elements with class "key--<d.keyField>"
                        .classed("highlight", false)
                div.transition()
                .duration(500)
                .style("opacity", 0);
        }
//highlight pack and barchart
        function highligtpackNodesOfUniversitiesWithThisUni(d){
                //Get UoA name of clicked bar
                var clickedUni = d["Institution name"];

                //Get list of REF entries with this UoA
                var refEntriesWithThisUni = dataModel
                        .filter(e=>e["Institution name"] == clickedUni);
                //Extract list of cleaned university (institution) class names
                institutionClassesToHighlight = refEntriesWithThisUni
                        .map(function(e){
                                return ".nest-key--"+e["Institution name"].replace(/[\W]+/g,"_")
                        })

                        uoAtoHighlight=".nest-key--"+d.UoAString.replace(/[\W]+/g,"_");

                institutionClassesToHighlight.forEach(function(institutionClass){
                        d3.selectAll(institutionClass).classed ("highlight", true)
                })


                d3.selectAll(".key--"+d.UoAString.replace(/[\W]+/g,"_")) //select all DOM elements with class "key--<d.keyField>"
                        .classed("highlight", true)
```

```
250
251                     div.transition()
252                     .duration(200)
253                     .style("opacity", 0.9);
254                     div.html(d["Institution name"]+ "</br>"+"4*: "+d.environment["4*"]+"%" + "<br/>"+
255 "Word Count: " + d.environment.WordCount)
256                     .style("left", (d3.event.pageX) + "px")
257                     .style("top", (d3.event.pageY - 28) + "px");
258             }
259
260 //remove highlighting from pack and barchart
261             function removeHighlightingUni (d){
262                     institutionClassesToHighlight.forEach(function(institutionClass){
263                             d3.selectAll(institutionClass).classed ("highlight", false)
264                     })
265
266                     d3.selectAll(".key--"+d.UoAString.replace(/[\W]+/g,"_")) //select all DOM elements
267 with class "key--<d.keyField>"
268                                     .classed("highlight", false)
269                     div.transition()
270                     .duration(500)
271                     .style("opacity", 0);
272             }
273
274 }
275
276 //==================== HELPER FUNCTIONS ========================================
277 function packClickFunction(d){
278             //If leaf node then user has clicked on a University
279             //so render that university's data in a barchart
280                     cdata =d;
281                     var refEntriesWithThisUoA = dataModel
```

```
282             if (d.height == 0) {
283
284                     console.log("pack click, d.height, d = ", d.data.key)
285                     var uni = d.data.key;
286                     renderUniversityDataAsBarchart(uni)
287             }
288
289     }
290
291     function renderUniversityDataAsBarchart(university){
292             //Generate set of sorted REF entries for this university
293             var bc1Data = dataModel
294                     .filter(e => e["Institution name"] == university)
295                             .filter(e => e.environment["4*"] >0)
296                     .sort((a, b) => d3.ascending(
297                             a["UoAString with Multiple submission letter appended"],
298                             b["UoAString with Multiple submission letter appended"]
299                     ))
300                     d3.selectAll("#barTitle")
301                                     .text("4* UoA Environments for "+university);
302             //Render the barchart
303
304
305     sdata=bc1Data
306             bc1.loadAndRenderDataset(sdata)
307
308
309
310     }
311
312     </script>
```

```
</body>
</html>
```

6.2 force.js

```
/****************************************
Authors: Matthew Reilly, Mike Bostock
19/11/2018
What it does:
renders force-directed diagram

****************************************/


function force(targetDOMelement) {

var forceObject = {};
var target = targetDOMelement;

//clear all topic with below 2%.
clearZero = function(obj){
  var size = 0,key;
  for(key in obj){
      if (obj[key] <= 0.02) delete obj[key];
  }
  return obj;
};
//get size of object
Object.size = function(obj) {
      var size = 0, key;
      for (key in obj) {
        console.log(key);
          if (obj.hasOwnProperty(key)) size++;
      }
      return size;
```

```
  };

forceObject.loadAndRenderDataset = function (data) {

    layoutAndRenderData(data);
    return forceObject;
  };


  var width = 400,
      height = 400,
      maxRadius = 150;
var   enternode,updatenode,exitnode
var nodes = []
    color = d3.scaleOrdinal(d3.schemeCategory10);
    // Get the size of an object


    var simulation = d3.forceSimulation(nodes)
        .force("charge", d3.forceManyBody().strength(-300))
        .force("forceX", d3.forceX().strength(.1))
        .force("forceY", d3.forceY().strength(.1))
        .force("center", d3.forceCenter())
        .alphaTarget(1)
        .on("tick", ticked);

    var svg = d3.select(targetDOMelement).append("svg").attr("width", width).attr("height",
height).classed("force",true)
        g = svg.append("g").attr("transform", "translate(" + width / 2 + "," + height / 2 + ")"),
        node = g.append("g").attr("stroke", "#fff").attr("stroke-width", 1.5).selectAll(".node");

```

```


  function layoutAndRenderData(data) {

 getData(data)
    // transition
    var t = d3.transition()
        .duration(750);

    // Apply the general update pattern to the nodes.
    node = node.data(nodes, function(d) { return d.name;});
    //exit
    node.exit()
    .classed("updateSelection enterSelection", false)
    .classed("exitSelection", true)
    .transition(t)
        .attr("r", 0)
        .remove();
//update
    node
    .classed("updateSelection", true)
    .classed("enterSelection exitSelection", false)
        .transition(t).delay(750)
         .attr("r", function(d){ return d.radius; });
//enter
    node = node.enter().append("circle")
        .classed("force enterSelection", true)
        .attr("r", function(d){ return d.radius })
        .on("mouseover", function(d) {
                div.transition()
                  .duration(200)
```

```
                    .style("opacity", 0.9);
                  div.html(d.name + "<br/>" + "Similarity: "+((d.radius/maxRadius)*100)+"%")
                    .style("left", (d3.event.pageX) + "px")
                    .style("top", (d3.event.pageY - 28) + "px");
                  })
                .on("mouseout", function(d) {
                  div.transition()
                    .duration(500)
                    .style("opacity", 0);
                  })
          .merge(node);

      // Update and restart the simulation.
      simulation.nodes(nodes)
        .force("collide", d3.forceCollide().strength(1).radius(function(d){ return d.radius+1;
}).iterations(1));

    }
//every tick
    function ticked() {
      node.attr("cx", function(d) { return d.x; })
          .attr("cy", function(d) { return d.y; })

    }




//convert the data into nodes.
function getData(data){
  data = clearZero(data);
```

```
  var size = Object.size(data);
  keys = Object.keys(data);
  var c =0;
  var name = new Array(size);
  nodes = d3.range(size).map(function() {
    var i = keys[c],
        r = data[keys[c]] * maxRadius,
        d = {name: i, radius: r};
        c++;
    return d;
  });
  console.log(nodes)
}


return forceObject;
}
```

6.3 pack.js

```
/****************************************
Authors: Matthew Reilly,Mike Chantler, Mike Bostock
19/11/2018
What it does:
render circle pack

****************************************/

var hierarchyGraph; //The graph of objects used to represent the hierarchy

function pack(targetDOMelement) {
	//Here we use a function declaration to imitate a 'class' definition
	//
	//Invoking the function will return an object (packObject)
	//	e.g. pack_instance = pack(target)
	//	This also has the 'side effect' of appending an svg to the target element
	//
	//The returned object has attached public and private methods (functions in JavaScript)
	//For instance calling method 'updateAndRenderData()' on the returned object
	//(e.g. pack_instance) will render a pack to the svg




	//Delare the main object that will be returned to caller
	var packObject = {};



```

```
        //=================== PUBLIC FUNCTIONS ========================
        //


        packObject.loadAndRenderNestDataset = function (nestFormatHierarchy, rootName) {
                //Loads and renders (format 2) hierarchy in "nest" or "key-values" format.
                layoutAndRenderHierarchyInNestFormat(nestFormatHierarchy, rootName)
                return packObject; //for method chaining
        }


        packObject.nodeLabelIfNoKey = function (fn) {
                //Leaf nodes from d3.nest typically have no 'key' property
                //By default the d3.nest 'key' property is used as the node text label
                //If this does not exist the nodeLabelIfNoKey() function will be called to
                // provide the label
                nodeLabelIfNoKey = fn;
                return packObject; //for method chaining
        }
  packObject.appendClickFunction = function (fn) {
    //Instead of overriding the internal click function
    //this will append the invocation of 'fn' to the end of it


    appendClickFunction = fn;
    return packObject; //for method chaining
  }


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

```

```
//Declare and append SVG element
var margin = {top: 20, right: 20, bottom: 20, left: 20},
width = 600 - margin.right - margin.left,
height = 500 - margin.top - margin.bottom;

//Set up SVG and append group to act as container for pack graph
var grp = d3.select(targetDOMelement).append("svg")
        .attr("width", width + margin.right + margin.left)
        .attr("height", height + margin.top + margin.bottom)
        .append("g")
        .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

//Add group for the nodes, just for clarity when 'inspecting' the html & svg
var nodesGroup = grp
        .append("g")
        .classed("nodesGroup", true);

//Add group for the links, just for clarity when 'inspecting' the html & svg
var linksGroup = grp
        .append("g")
        .classed("linksGroup", true);



//=================== PRIVATE FUNCTIONS ====================================
var nodeLabelIfNoKey = function(){return "No name set"};
var appendClickFunction = function(){console.log ("No click fn appended")};
var clickFunction = function (d,i){console.log("node clicked, d = ",d)
packClickFunction(d)
d3.select(this)
.style('fill', 'orange');
```

```
        }
        var nodeLabel = function(d) {return d.data.name + "(height:"+d.height+")";}


        function layoutAndRenderHierarchyInNestFormat (nestFormatHierarchy, rootName){
        //Lays out and renders (format 2) hierarchy in "nest" ("key-values" format).
        console.log(nestFormatHierarchy)
                //Move the 'nest' array into a root node:
                var datasetAsJsonD3Hierarchy = {"key":rootName, "values": nestFormatHierarchy}

                //Now create hierarchy structure
                //Note that we need to add the "children" accessor "d=>d.values" in order
                //to tell d3.hierarchy to use nest's 'values' as children
                hierarchyGraph = d3
                        .hierarchy(datasetAsJsonD3Hierarchy, d=>d.values) //
                        .sum(d=>d.value) //usually not required for pack (this adds the sum of all descendants' sizes and stores in node.value)
                        .sort(function(a, b) { return b.value - a.value; });

                //And we'll use the nest 'keys' as the node labels
                nodeLabel = function(d) {
                        if (d.data.key) return d.data.key + "(value:"+ d.value+")";
                        else return nodeLabelIfNoKey(d);
                }

                //Can now calculate XY data and render
                calculateXYpositionsAndRender(hierarchyGraph);
        }
```

```
        function calculateXYpositionsAndRender(hierarchyGraph){

                //get and setup the pack layout generator
                var mypackLayoutGenerator = d3.pack().size([height, height]);

                //Add x and y properties to each node in the hierarchy graph.
                var hierarchyGraphWithPositions = mypackLayoutGenerator(hierarchyGraph);

                //Get lists of nodes
                var listOfNodes = hierarchyGraphWithPositions.descendants();
console.log("list of nodes = ", listOfNodes);
                //Render links and nodes
                GUPrenderNodes(listOfNodes);
        }


        function GUPrenderNodes(listOfNodes){

                //DATA BIND

                var selectionGroup = nodesGroup
                        .selectAll("g.cssClassNode") //select groups with class = "cssClassNode"
                        .data(listOfNodes);

                //ENTER  SELECTION PROCESSING

                //Create groups
                var enterSelectionGroup = selectionGroup
                        .enter()
                        .append("g")
```

```
                        .attr("class", d=>{if(d.data.key) return "nest-key--
"+d.data.key.replace(/[\W]+/g,"_"); else return "No key";})
                        .classed("cssClassNode enterSelection", true)

                enterSelectionGroup
                .append("circle")
                .attr("r", function(d) {console.log("d=",d);return d.r} );

                enterSelectionGroup
                        .on("click", clickFunction)
                        .on("mouseover", function(d) {
            div.transition()
              .duration(200)
              .style("opacity", 0.9);
            div.html(d.data.key + "<br/>" + "Number of Staff: "+d.value)
              .style("left", (d3.event.pageX) + "px")
              .style("top", (d3.event.pageY - 28) + "px");
            })
           .on("mouseout", function(d) {
            div.transition()
              .duration(500)
              .style("opacity", 0);
            });
                        enterSelectionGroup
                                //set appropriate classes for the group
                                .classed("leafNode", d => d.height == 0)
                                .classed("rootNode", d => d.depth == 0)
                                .classed("intermediateNode", d => (d.height != 0 && d.depth != 0));

                        enterSelectionGroup
                                .attr("transform", function(d) {
```

```
                                        return "translate(" + d.y + "," + d.x + ")";
                                    });


        //update
            var updateSelection = selectionGroup

            updateSelection.select("circle")
            .attr("r", function(d) {console.log("d=",d);return d.r} );


            updateSelection
                .attr("transform", function(d) {
                    return "translate(" + d.y + "," + d.x + ")";
                });

            updateSelection
                //set appropriate classes for the group
                .classed("leafNode", d => d.height == 0)
                .classed("rootNode", d => d.depth == 0)
                .classed("intermediateNode", d => (d.height != 0 && d.depth != 0));

            //Create Merged ENTER + UPDATE selections for the text element in the group

            // EXIT
            var exitSel =selectionGroup
                .exit()
                .classed("enterSelection updateSelection", false)
                .classed("exitSelection", true)
                .remove();
        }
```

```
        //================== IMPORTANT do not delete
==================================
        return packObject; // return the main object to the caller to create an instance of the 'class'

} //End of pack() declaration
```

6.4 scatter.js

```
/****************************************
Authors: Matthew Reilly, Mike Chantler
19/11/2018
What it does:
rneders scatterplot

****************************************/
"use safe"

function scatterplot(targetDOMelement) {
        //Here we use a function declaration to imitate a 'class' definition
        //
        //Invoking the function will return an object (scatterplotObject)
        //    e.g. scatterplot_instance = scatterplot(target)
        //    This also has the 'side effect' of appending an svg to the target element
        //
        //The returned object has attached public and private methods (functions in JavaScript)
        //For instance calling method 'updateAndRenderData()' on the returned object
        //(e.g. scatterplot_instance) will render a scatterplot to the svg


        //Delare the main object that will be returned to caller
        var scatterplotObject = {};

        //=================== PUBLIC FUNCTIONS ========================
        //



 scatterplotObject.appendedClickFunction = function (callbackFunction) {
```

```
  appendClickFunction = callbackFunction;
  return scatterplotObject;
 }


 scatterplotObject.appendedMouseOverFunction = function (callbackFunction) {
                console.log("appendedMouseOverFunction called", callbackFunction)
                appendedMouseOverFunction = callbackFunction;
                return scatterplotObject;
        }

        scatterplotObject.appendedMouseOutFunction = function (callbackFunction) {
                appendedMouseOutFunction = callbackFunction;
                return scatterplotObject;
        }

        scatterplotObject.loadAndRenderDataset = function (data) {
                dataset=data.map(d=>d)
                GUP_bars();
                return scatterplotObject;
        }

        scatterplotObject.overrideMouseoverFunction = function (callbackFunction) {
                mouseoverCallback = callbackFunction;
                return scatterplotObject;
        }

        scatterplotObject.overrideMouseoutFunction = function (callbackFunction) {
                mouseoutCallback = callbackFunction;
                return scatterplotObject;
        }
```

```

        scatterplotObject.overrideDataFieldFunction = function (dataFieldFunction) {
                dataField = dataFieldFunction;
                return scatterplotObject;
        }

  scatterplotObject.overrideKeyFunction = function (keyFunction) {
    //The key function is used to obtain keys for GUP rendering and
    //to provide the categories for the y-axis
    //These valuse should be unique
    GUPkeyField = yAxisCategoryFunction = keyFunction;
    return scatterplotObject;
  }

        //=================== PRIVATE VARIABLES ====================================
        //Width and height of svg canvas
        var svgWidth = 900;
        var svgHeight = 500;
        var dataset = [];


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

        //Declare and append SVG element
        var svg = d3
                .select(targetDOMelement)
                .append("svg")
                .attr("width", svgWidth)
                .attr("height", svgHeight)
                .classed("scatterplot",true);

```

```
        //===================== PRIVATE FUNCTIONS
========================================

        var dataField = function(d){return d.dataField} //The length of the bars
var yAxisCategoryFunction = function(d){return d.UoAString} //Categories for y-axis
var GUPkeyField = yAxisCategoryFunction;
        var mouseoverCallback = function(d){
                d.highlight = true;
                GUP_bars();
        }




        var mouseoutCallback = function(d){
                d.highlight =false;
                GUP_bars();
        }


var appendedMouseOutFunction = function(){};

var appendedMouseOverFunction = function(d){};

var mouseOverFunction = function (d,i){
    d3.select(this).classed("highlight", true).classed("noHighlight", false);
  appendedMouseOverFunction(d,i);
}

var mouseOutFunction = function (d,i){
    d3.select(this).classed("highlight", false).classed("noHighlight", true);
```

```
  appendedMouseOutFunction(d,i);
 }
 var appendedClickFunction = function (d,i){
     console.log("scatter click function = nothing at the moment, d=",d)
         };


         var GUP_bars = function(){
                 //GUP = General Update Pattern to render bars


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


  //GUP: ENTER SELECTION
 var enterSel = selection //Create DOM rectangles, positioned @ x=yAxisIndent
  .enter()
  .append("circle")


 enterSel //Add CSS classes
 .attr("class", (d=>"key--"+d["Institution name"].replace(/[\W]+/g,"_")))
  .classed("scatter enterSelection", true)
  .classed("highlight", d=>d.highlight)

 enterSel
  .transition()
  .duration(1000)
  .delay(2000)
```

```
    .attr("r", 8)
    .attr("cx", function(d) { return (10+d.environment["4*"]*8); })
    .attr("cy", function(d) { return (10+d.environment.WordCount/25); })

    enterSel
                    .on("mouseover", mouseOverFunction)
                    .on("mouseout", mouseOutFunction)
                    .on("click", appendClickFunction)


  //GUP UPDATE (anything that is already on the page)
  var updateSel = selection //update CSS classes
    .classed("noHighlight updateSelection", true)
    .classed("highlight enterSelection exitSelection", false)
    .classed("highlight", d=>d.highlight)

  updateSel       //update bars
    .transition()
    .duration(1000)
    .delay(1000)
    .attr("r", 5)
    .attr("cx", function(d) { return (10+d.environment["4*"]*8); })
    .attr("cy", function(d) { return (10+d.environment.WordCount/25); })

    updateSel
                    .on("mouseover", mouseOverFunction)
                    .on("mouseout", mouseOutFunction)
                    .on("click", appendClickFunction)
```

```

  //GUP EXIT selection
  var exitSel = selection.exit()
    .classed("highlight updateSelection enterSelection", false)
    .classed("exitSelection", true)

      .attr("r",0)
      .remove()
};



            //================== IMPORTANT do not delete
==================================
            return scatterplotObject; // return the main object to the caller to create an instance of the
'class'

} //End of scatterplot() declaration
```

6.5 barhcart.js

```
/****************************************
Authors: Matthew Reilly, Mike Chantler
19/11/2018
What it does:
renders barchart
****************************************/
"use safe"

function barchart(targetDOMelement) {

	var barchartObject = {};

	//=================== PUBLIC FUNCTIONS =========================

	barchartObject.appendedMouseOverFunction = function (callbackFunction) {
		console.log("appendedMouseOverFunction called", callbackFunction)
		appendedMouseOverFunction = callbackFunction;
		render();
		return barchartObject;
	}

	barchartObject.appendedMouseOutFunction = function (callbackFunction) {
		appendedMouseOutFunction = callbackFunction;
		render();
		return barchartObject;
	}

	barchartObject.loadAndRenderDataset = function (data) {
		dataset=data.map(d=>d); //create local copy of references so that we can sort etc.
		render();
```

```
            return barchartObject;
        }

        barchartObject.overrideDataFieldFunction = function (dataFieldFunction) {
            dataField = dataFieldFunction;
            return barchartObject;
        }

        barchartObject.overrideKeyFunction = function (keyFunction) {
            //The key function is used to obtain keys for GUP rendering and
            //to provide the categories for the y-axis
            //These valuse should be unique
            GUPkeyField = yAxisCategoryFunction = keyFunction;
            return barchartObject;
        }

        barchartObject.overrideMouseOverFunction = function (callbackFunction) {
            mouseOverFunction = callbackFunction;
            render();
            return barchartObject;
        }

        barchartObject.overrideMouseOutFunction = function (callbackFunction) {
            mouseOutFunction = callbackFunction;
            render(); //Needed to update DOM
            return barchartObject;
        }

        barchartObject.overrideTooltipFunction = function (toolTipFunction) {
            tooltip = toolTipFunction;
            return barchartObject;
```

```
        }

        barchartObject.overrideMouseClickFunction = function (fn) {
                mouseClick2Function = fn;
                render(); //Needed to update DOM if they exist
                return barchartObject;
        }

        barchartObject.maxValueOfDataField = function (max) {
                maxValueOfDataset = max;
                maxValueOfDataField=function(){return maxValueOfDataset};
                return barchartObject;
        }

        barchartObject.render = function (callbackFunction) {
                render(); //Needed to update DOM
                return barchartObject;
        }

        barchartObject.sortByDataField = function () {
                dataset.sort((a,b)=>dataField(a)-dataField(b))
                render();
                return barchartObject;
        }

        barchartObject.reverseSortByDataField = function () {
                dataset.sort((a,b)=>dataField(b)-dataField(a))
                render();
                return barchartObject;
        }

```

```
        barchartObject.sortByKey = function () {
                //for security we will use D3's descending operator here
                dataset.sort((a,b)=>d3.descending(GUPkeyField(b),GUPkeyField(a)))
                render();
                return barchartObject;
        }

        barchartObject.setTransform = function (t) {
                //Set the transform on the svg
                svg.attr("transform", t)
                return barchartObject;
        }

        barchartObject.yAxisIndent = function (indent) {
                yAxisIndent=indent;
                return barchartObject;
        }



        //=================== PRIVATE VARIABLES ====================================
        //Width and height of svg canvas
        var svgWidth = 900;
        var svgHeight = 450;
        var dataset = [];
        var xScale = d3.scaleLinear();
        var yScale = d3.scaleBand(); //This is an ordinal (categorical) scale
        var yAxisIndent = 400; //Space for labels
        var maxValueOfDataset; //For manual setting of bar length scaling (only used if
.maxValueOfDataset() public method called)

```

```
	//=================== INITIALISATION CODE ===================================

	//Declare and append SVG element
	var svg = d3
		.select(targetDOMelement)
		.append("svg")
		.attr("width", svgWidth)
		.attr("height", svgHeight)
		.classed("barchart",true);

	//Declare and add group for y axis
	var yAxis = svg
		.append("g")
		.classed("yAxis", true);

	//Declare and add group for x axis
	var xAxis = svg
		.append("g")
		.classed("xAxis", true);




	//===================== ACCESSOR FUNCTIONS ========================================

	var dataField = function(d){return d.datafield} //The length of the bars
	var tooltip = function(d){return  d.UoAString + ": "+  Number(d.environment["4*"])} //tooltip text for bars
	var yAxisCategoryFunction = function(d){return d.key} //Categories for y-axis
	var GUPkeyField = yAxisCategoryFunction; //For 'keyed' GUP rendering (set to y-axis category)
```

```
	//=================== OTHER PRIVATE FUNCTIONS ===================================
	var maxValueOfDataField = function(){
		//Find the maximum value of the data field for the x scaling function using a handy d3 max() method
		//This will be used to set (normally used )
		return d3.max(dataset, dataField)
	};

	var appendedMouseOutFunction = function(){};

	var appendedMouseOverFunction = function(){};

	var mouseOverFunction = function (d,i){
	d3.select(this).classed("highlight", true).classed("noHighlight", false);
		appendedMouseOverFunction(d,i);
	}

	var mouseOutFunction = function (d,i){
	d3.select(this).classed("highlight", false).classed("noHighlight", true);
		appendedMouseOutFunction(d,i);
	}

	var mouseClick2Function = function (d,i){
	console.log("barchart click function = nothing at the moment, d=",d)
	};

	function render () {
		updateScalesAndRenderAxes();
		GUP_bars();
```

```
        }

        function updateScalesAndRenderAxes(){
                //Set scales to reflect any change in svgWidth, svgHeight or the dataset size or max value
                xScale
                        .domain([0, maxValueOfDataField()])
                        .range([0, svgWidth-(yAxisIndent+10)]);
                yScale
                        .domain(dataset.map(yAxisCategoryFunction)) //Load y-axis categories into yScale
                        .rangeRound([25, svgHeight-40])
                        .padding([.1]);

                //Now render the y-axis using the new yScale
                var yAxisGenerator = d3.axisLeft(yScale);
                svg.select(".yAxis")
                        .transition().duration(1000).delay(1000)
                        .attr("transform", "translate(" + yAxisIndent + ",0)")
                        .call(yAxisGenerator);

                //Now render the x-axis using the new xScale
                var xAxisGenerator = d3.axisTop(xScale);
                svg.select(".xAxis")
                        .transition().duration(1000).delay(1000)
                        .attr("transform", "translate(" + yAxisIndent + ",20)")
                        .call(xAxisGenerator);
        };

        function GUP_bars(){
                //GUP = General Update Pattern to render bars

```

```
221                 //GUP: BIND DATA to DOM placeholders
222                 var selection = svg
223                         .selectAll(".bars")
224                         .data(dataset, GUPkeyField);
225
226
227             //GUP: ENTER SELECTION
228                 var enterSel = selection //Create DOM rectangles, positioned @ x=yAxisIndent
229                         .enter()
230                         .append("rect")
231                         .attr("x", yAxisIndent)
232
233
234                 enterSel //Add CSS classes
235                  .attr("class", (d=>"key--"+d.UoAString.replace(/[\W]+/g,"_")))
236
237                         .classed("bars enterSelection", true)
238                         .classed("highlight", d=>d.highlight)
239
240                 enterSel //Size the bars
241                         .transition()
242                         .duration(1000)
243                         .delay(2000)
244                                 .attr("width", function(d) {return xScale(dataField(d));})
245                                 .attr("y", function(d, i) {return yScale(yAxisCategoryFunction(d));})
246                                 .attr("height", function(){return yScale.bandwidth()});
247
248
249
250
251                 //GUP UPDATE (anything that is already on the page)
```

```
252                 var updateSel = selection //update CSS classes
253                         .classed("noHighlight updateSelection", true)
254                         .classed("highlight enterSelection exitSelection", false)
255                         .classed("highlight", d=>d.highlight)
256
257                 updateSel       //update bars
258                         .transition()
259                         .duration(1000)
260                         .delay(1000)
261                                 .attr("width", function(d) {return xScale(dataField(d));})
262                                 .attr("y", function(d, i) {return yScale(yAxisCategoryFunction(d));})
263                                 .attr("height", function(){return yScale.bandwidth()});
264
265                 updateSel //update tool tip
266                         .select("title") //Note that we already created a <title></title> in the Enter
267 selection
268                                 .text(tooltip)
269
270
271                 //GUP: Merged Enter & Update selections (so we don't write these twice)
272                 var mergedSel = enterSel.merge(selection)
273                         .on("mouseover", mouseOverFunction)
274                         .on("mouseout", mouseOutFunction)
275                         .on("click", mouseClick2Function)
276
277
278                 //GUP EXIT selection
279                 var exitSel = selection.exit()
280                         .classed("highlight updateSelection enterSelection", false)
281                         .classed("exitSelection", true)
282                         .transition()
```

```
                                .duration(1000)
                                        .attr("width",0)
                                        .remove()
            };


            //================== IMPORTANT do not delete
==================================
            return barchartObject; // return the main object to the caller to create an instance of the
'class'

} //End of barchart() declaration
```

6.6 general.css

```
* {
    box-sizing: border-box;
}

body {
    font-family: Arial, Helvetica, sans-serif;
      background:  #f1f1f1;
}
#forceHyperlink{
  font-size: 10px
}

# forcetext1{
  font-size: 10px
}
/* Create two columns/boxes that floats next to each other */
nav {
    float: left;
    width: 35%;

    background:  #f1f1f1;
    padding: 0px;
}



article {
    float: left;
```

```
    padding: 10px;
    width: 65%;
    background-color: #f1f1f1;

}

/* Clear floats after the columns */
section:after {
    content: "";
    display: table;
    clear: both;
}

exit {
  fill:#b26745;
}
update {
  fill:#3a403d;
}
enter {
  fill: #45b29d;
}

div.tooltip {
  position: absolute;
  text-align: center;
  width: 180px;
  height: 56px;
  padding: 2px;
  font: 12px sans-serif;
  background: lightsteelblue;
```

```
  border: 0px;
  border-radius: 8px;
  pointer-events: none;
}
/* Responsive layout - makes the two columns/boxes stack on top of each other instead of next to
each other, on small screens */
@media (max-width: 600px) {
    nav, article {
        width: 100%;
        height: auto;
    }
}
```