11/13/2017

Data Visualization

F21DV

AlMazloum, Amer Eddin

Heriot Watt

**Professor**: Dr. Salih Ismail

**Course Work 1**

Table of Contents

[1. Introduction 3](#_Toc498367933)

[1. D3.js: 3](#_Toc498367934)

[2. Project goals: 3](#_Toc498367935)

[3. Project Motivation: 4](#_Toc498367936)

[1. Graphical User Manual 4](#_Toc498367937)

[1. General: 4](#_Toc498367938)

[2. Early career academics: 5](#_Toc498367939)

[3. University management: 9](#_Toc498367940)

[4. Industrial collaborators: 12](#_Toc498367941)

[2. Application Design 14](#_Toc498367942)

[1. Design Overview: 14](#_Toc498367943)

[2. Reflection Design: 14](#_Toc498367944)

[3. Use of interactions, and transitions: 14](#_Toc498367945)

[4. Description of original features: 14](#_Toc498367946)

[5. Design for three different user types: 14](#_Toc498367947)

[3. Conclusions 15](#_Toc498367948)

[4. Source code file list 16](#_Toc498367949)

[5. Source code listings 17](#_Toc498367950)

1. Introduction
2. D3.js:

**D3.js** is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3’s emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

Download the latest version (4.11.0) here:

[D3.zip](https://github.com/d3/d3/releases/download/v4.11.0/d3.zip)

The [full source and tests](https://github.com/d3/d3) are also available [for download](https://github.com/d3/d3/zipball/master) on GitHub. Show your support for D3’s ongoing development by [buying official stickers](https://www.stickermule.com/user/1070696243/stickers)!

**D3** allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

1. Project goals:

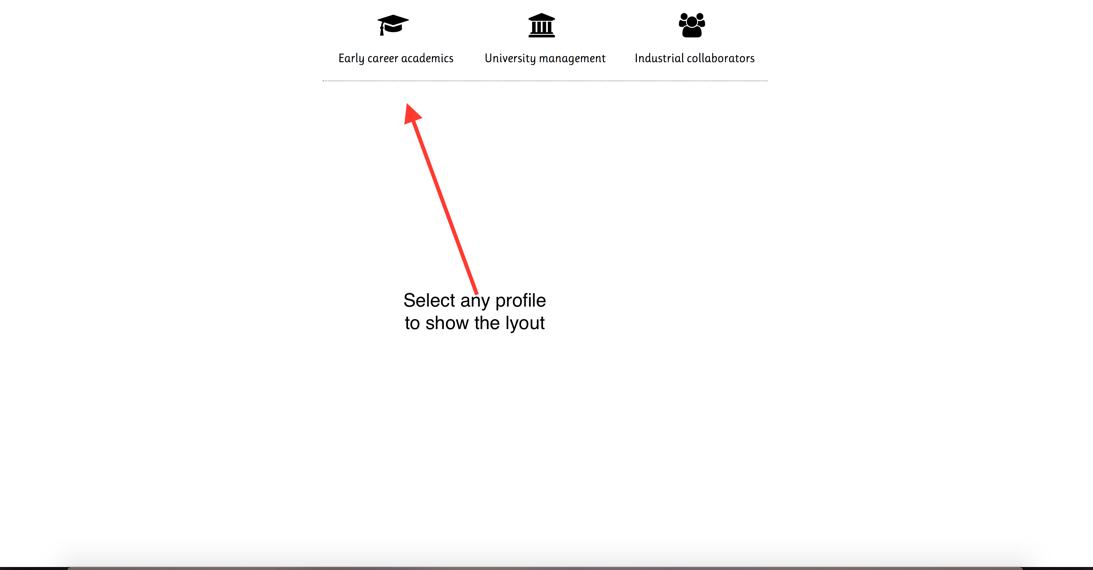
The goal of this project is to produce an interactive internet application for analysis & visualization of REF 2014 data to show at least three different layouts with three different profiles. The application is client side written in D3.js v4 library. The data provided in CSV format it’s about research excellence framework is quality assessment of UK universities’ research comprising:

* 154 UK institutions
* 36 subject-based Unit of assessment (UOAs)
* 4 profiles (outputs, impact, Environments, overall)
* 4 starred quality levels

For more information about this data see this link: <http://www.ref.ac.uk/results/intro/>

1. Project Motivation:
2. Graphical User Manual
3. General:

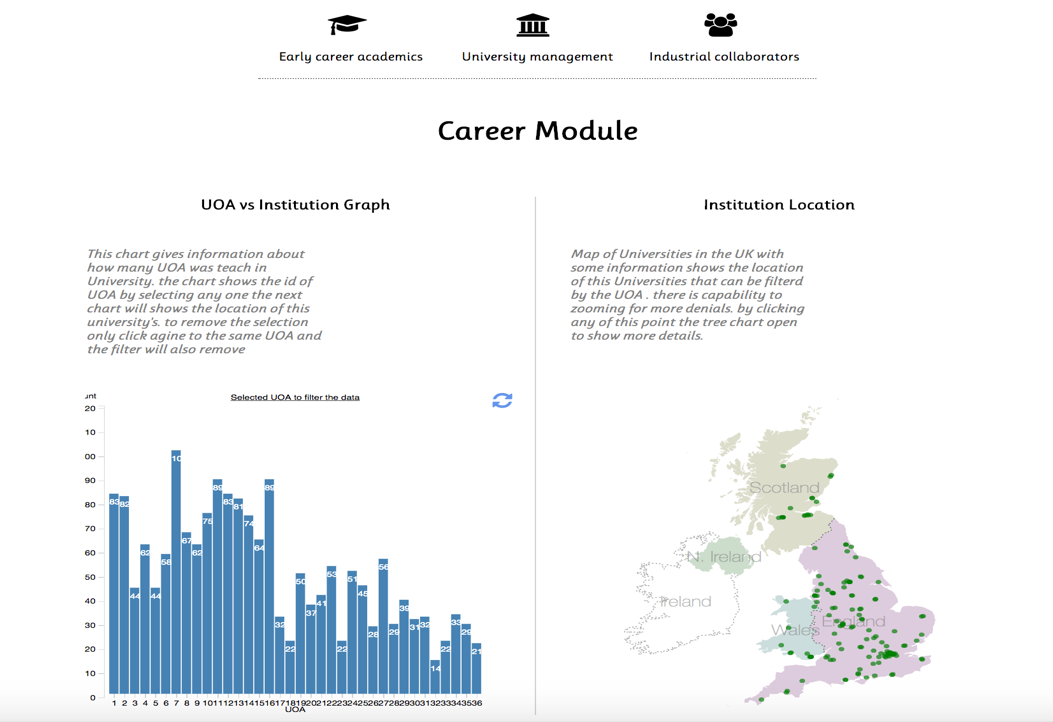
From this page, you can navigate between all the three profiles by clicking on the image.



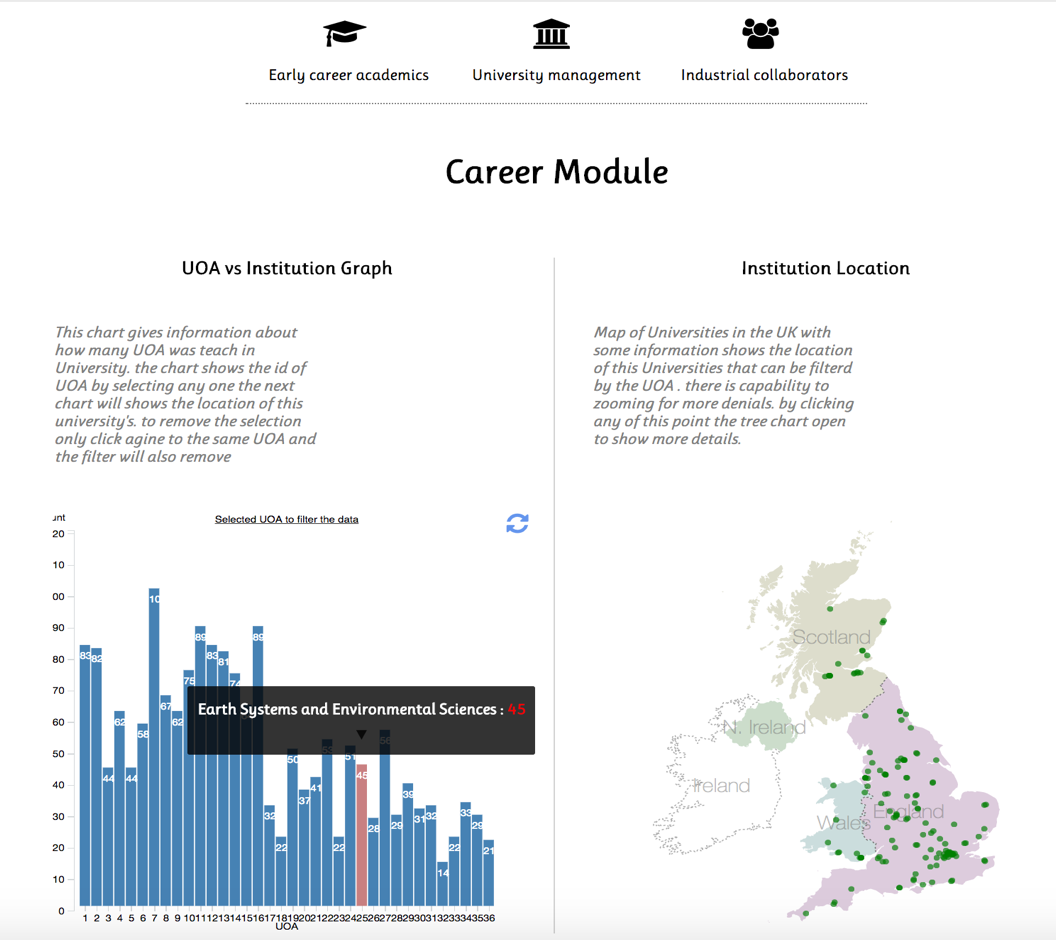
* **Early career academics**: will showing the Early career academics layouts and you can navigate between this layout.
* **University management:** will showing the University management layouts and you can navigate between this layout.
* **Industrial collaborators**: will showing the University management layouts and you can navigate between this layout.

1. Early career academics:

First thing appears once click on Early career is the Figure 1, that showing all the UOA in bar chart and all the Universities in map chart. From this point, you can move the mouse over any bar to show the information of this UOA as shooing in figure 2, by click on the bar you will selected the UOA and you will able to filter and see all the university that teaching this as a green point on the map chart like figure 3. The selected UOA name will appears on the tope of chart as showing in figure 3.

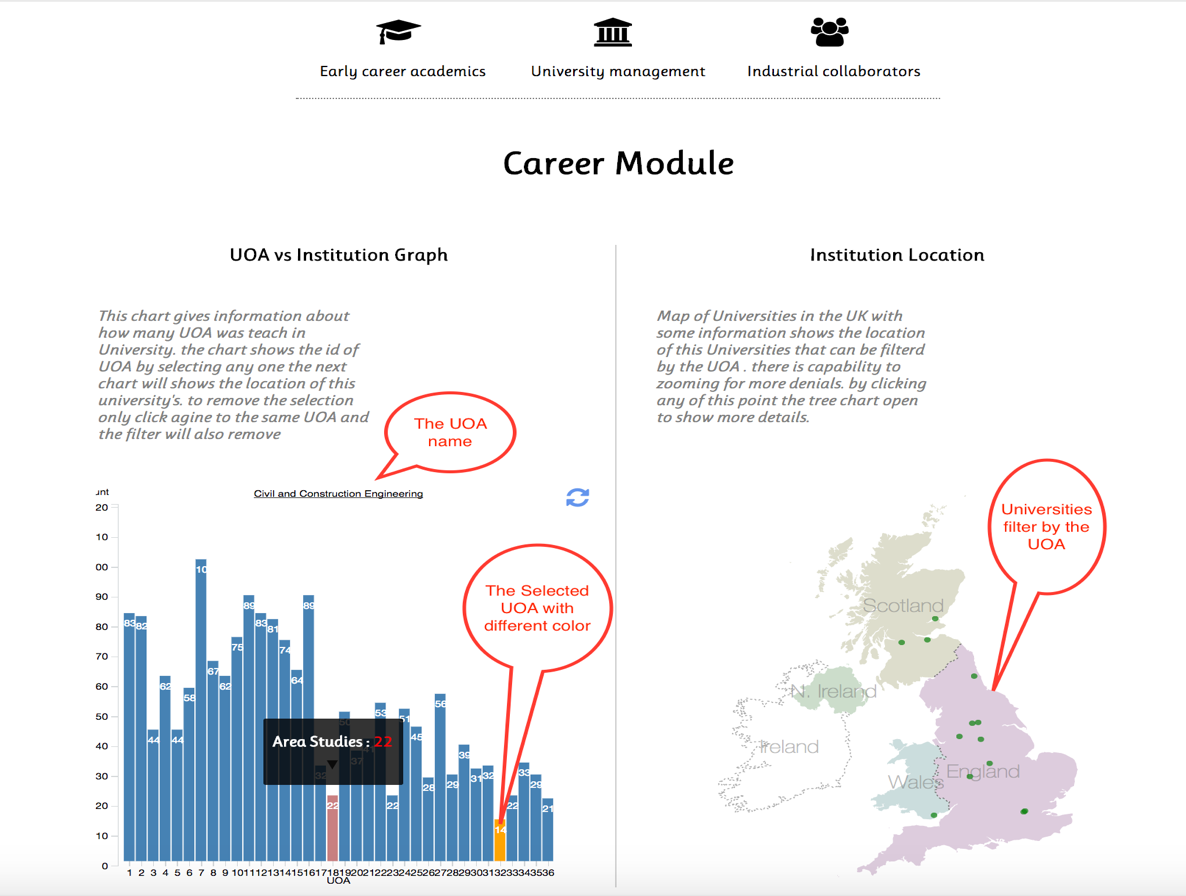


Figure



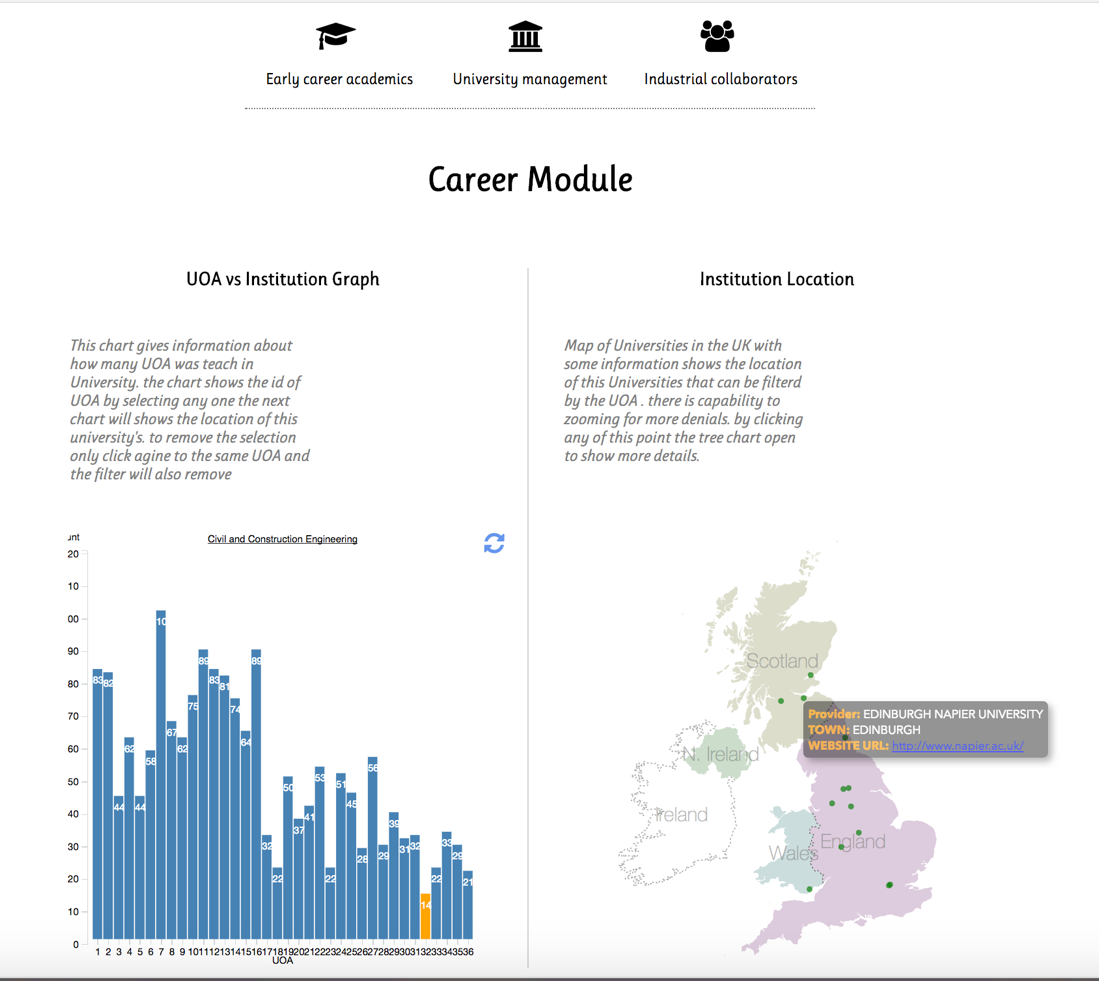
Figure

The Selected name of UOA and the filtered universities will be showing like this chart



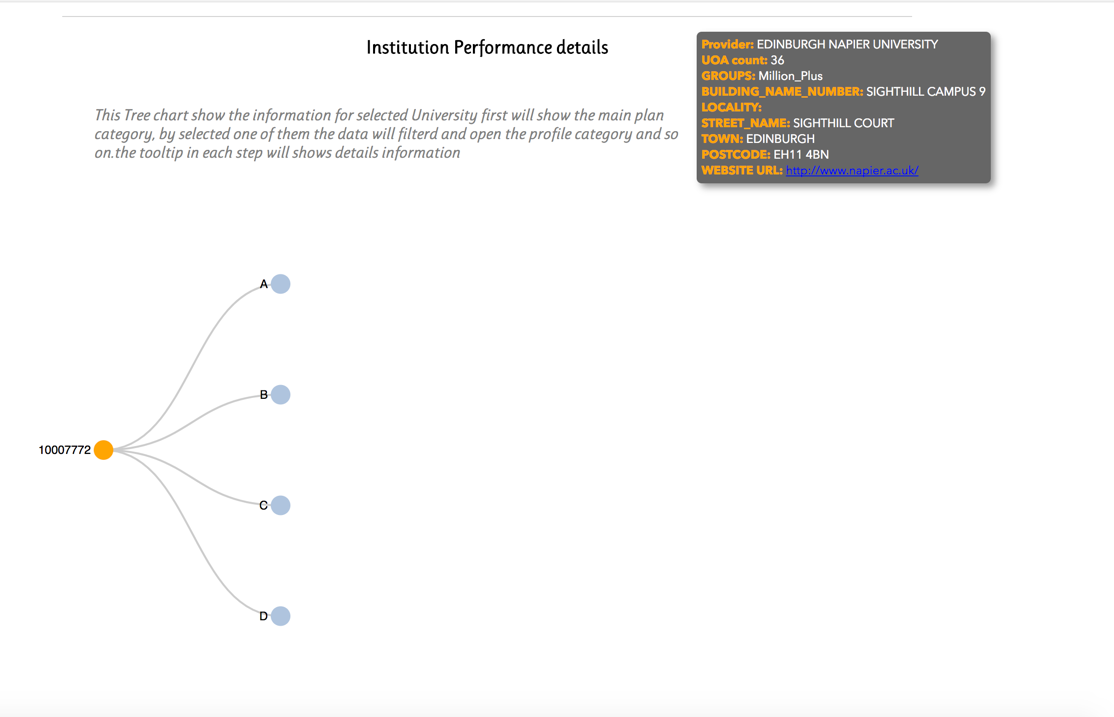
Figure

To show information about any point in the map (here university) just go to this point, when mouse will be over the point the information about this University will showing as tooltips as Figure 4. By click on any point on the map another chart will appear as tree to allow you to see the performance’s UOAs for selected university as Figure 5 or update the old chart to hold new data. There is also capability on the map chart for zooming and drag for more details (will enhance in new virgin) .



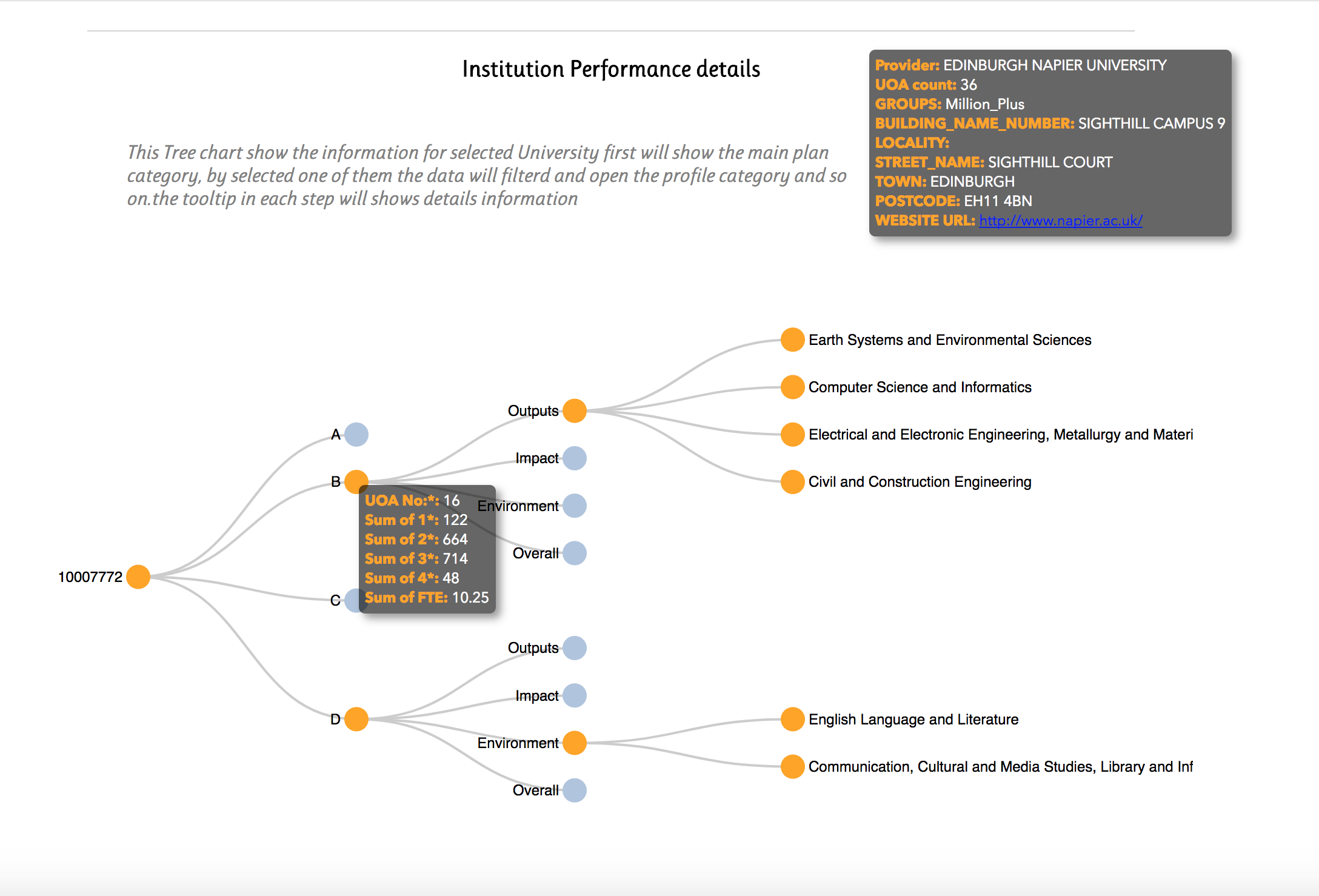
Figure

The tree map chart to show information about selected university.



Figure

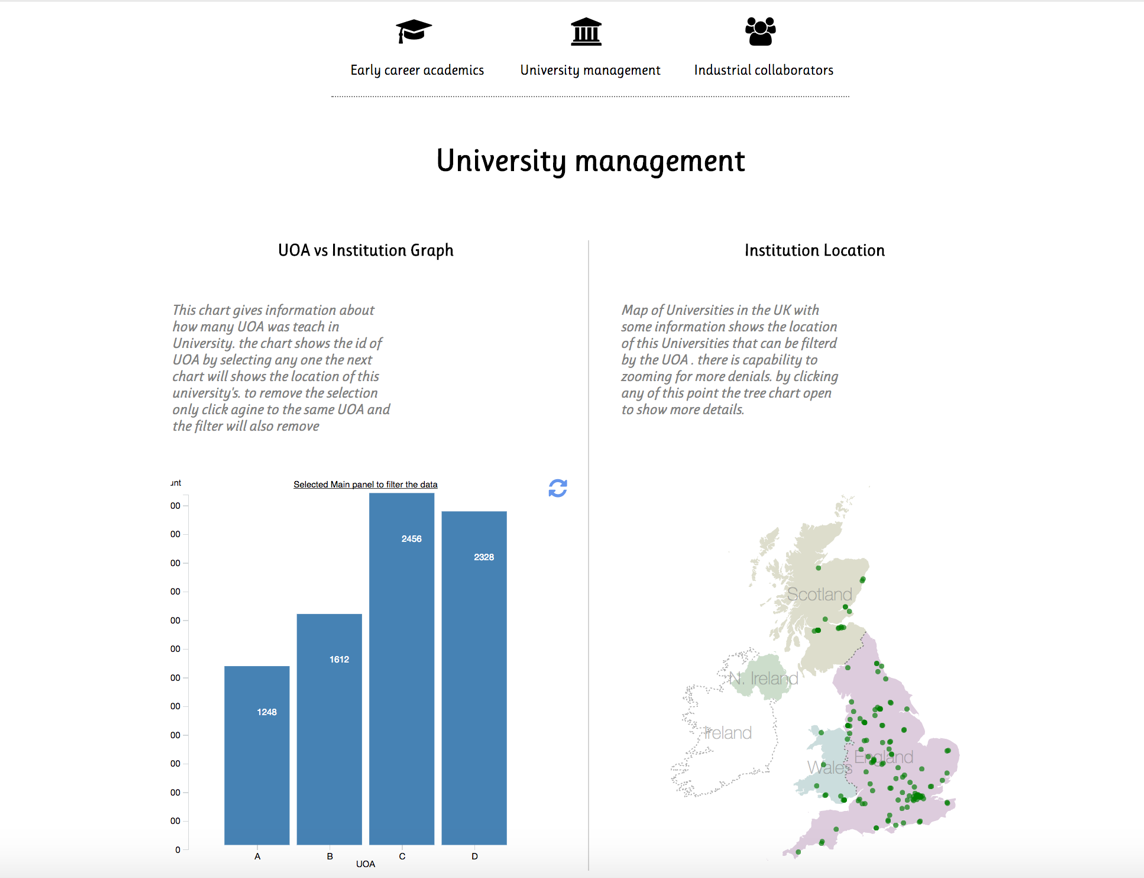
The last layout chart showing information about the institution performance details for each node for example the Main Panel B. as Figure 6 showing there is tooltips has information about the number of UOA under this category and sum of the level start 1\* ,2\* etc. by mouse over on any node the information will showing as tooltips and by click this node if gas any child the tree will grows till there is no more children node.



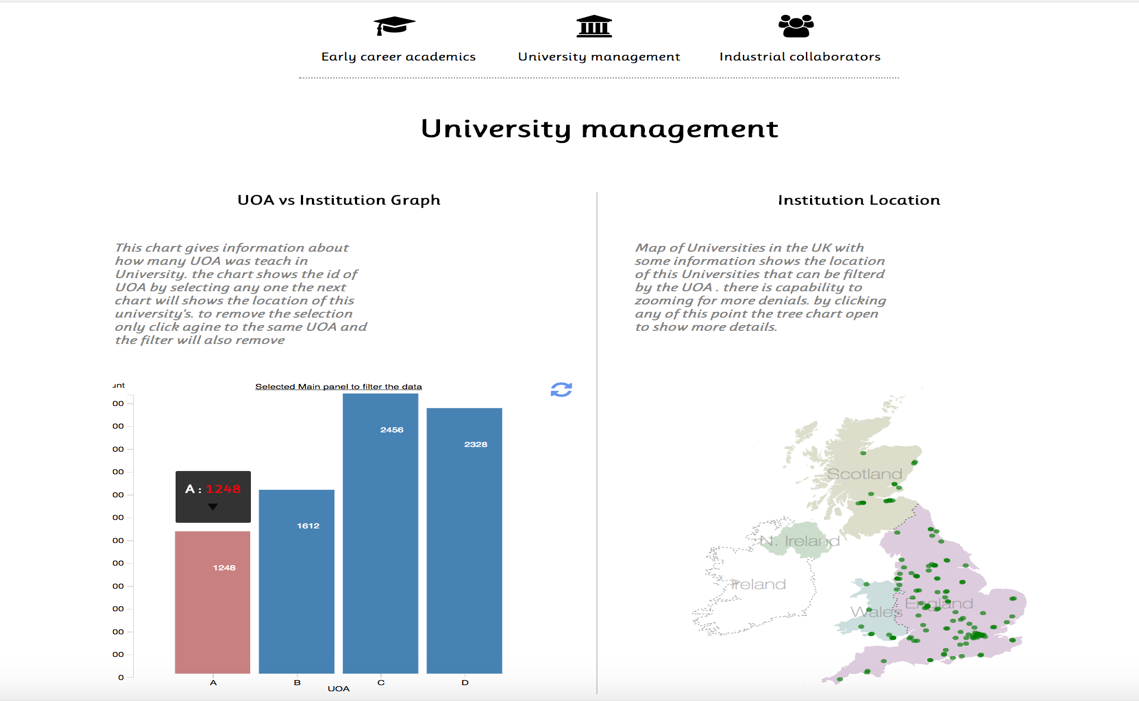
Figure

1. University management:

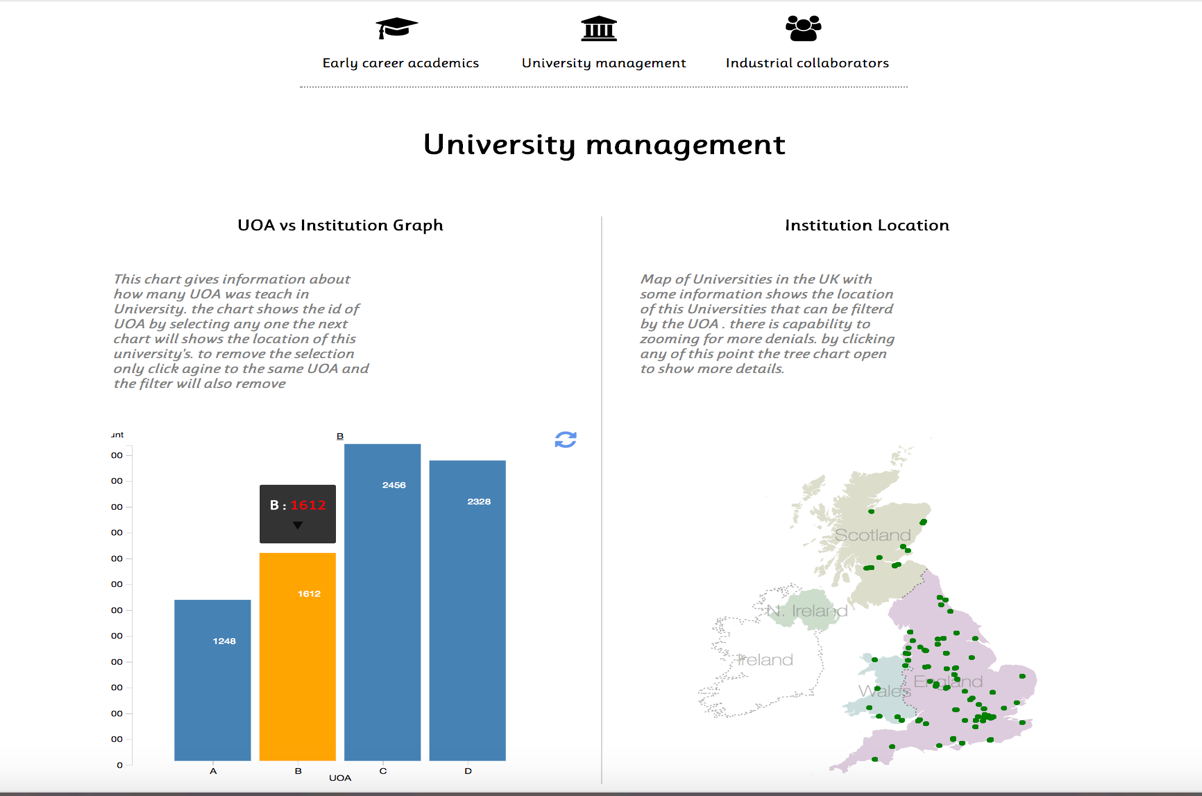
First thing appears once click on Early career is the Figure 7, that showing all the UOA in bar chart and all the Universities in map chart. And all other Figure from 8 to 12 explain more details.



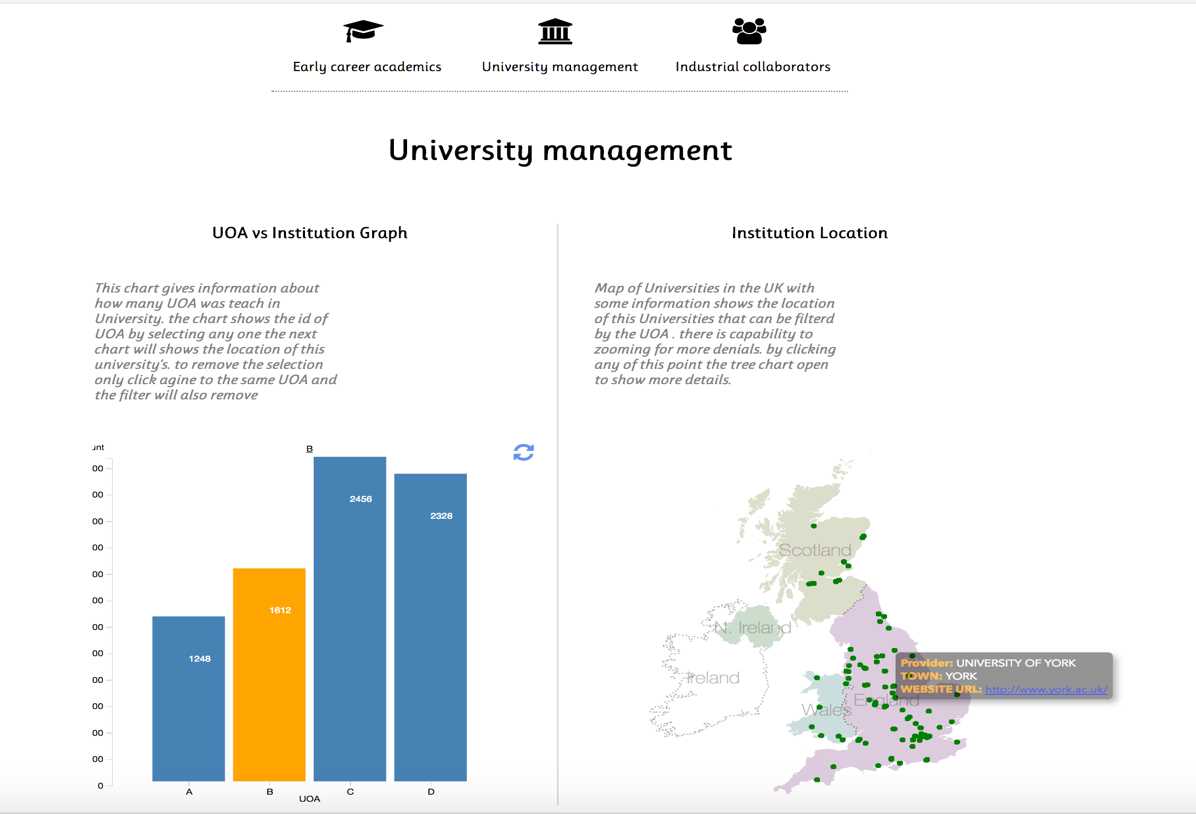
Figure



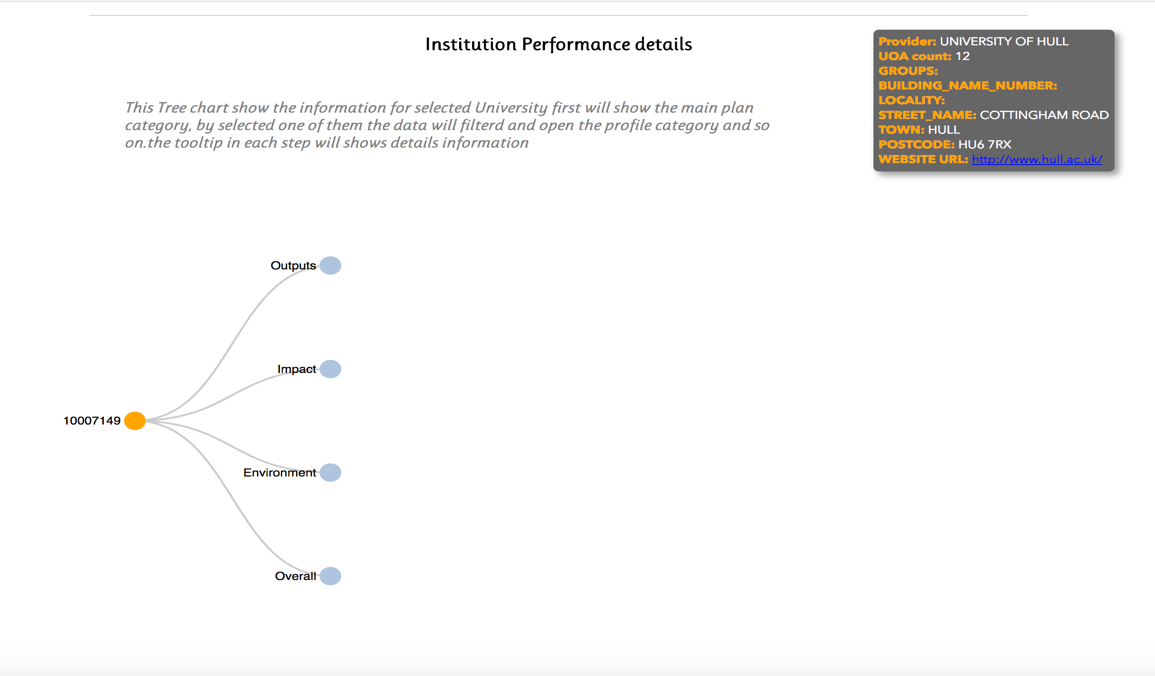
Figure



Figure



Figure



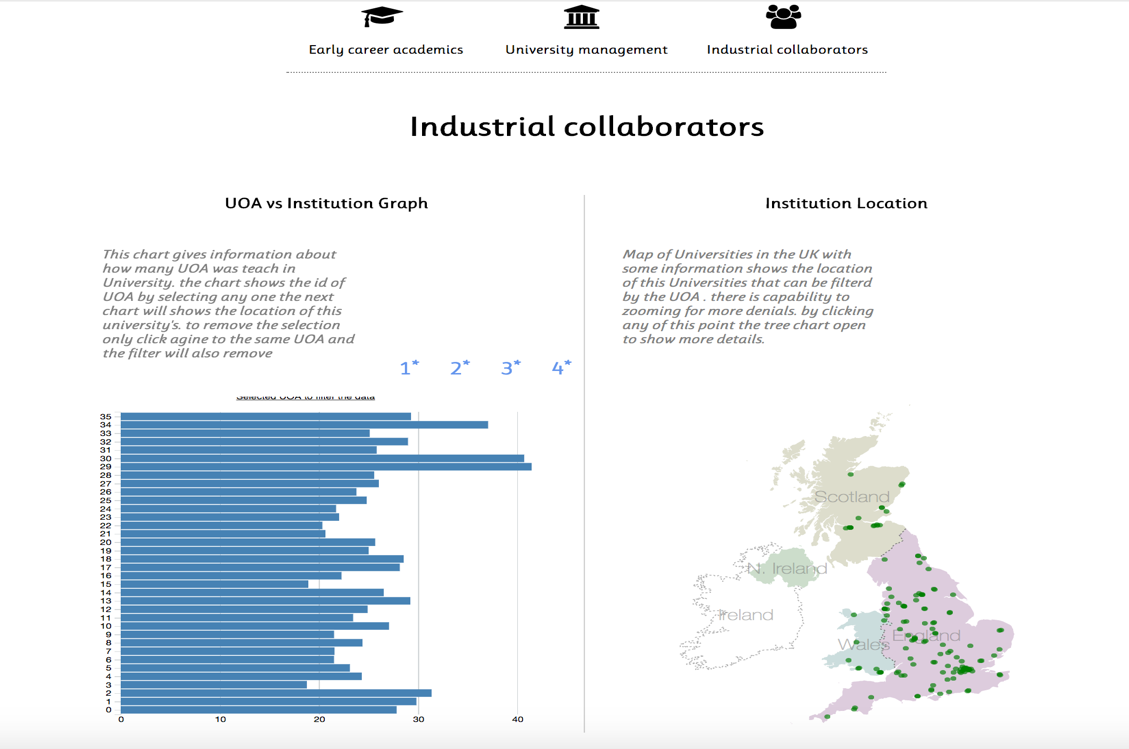
Figure



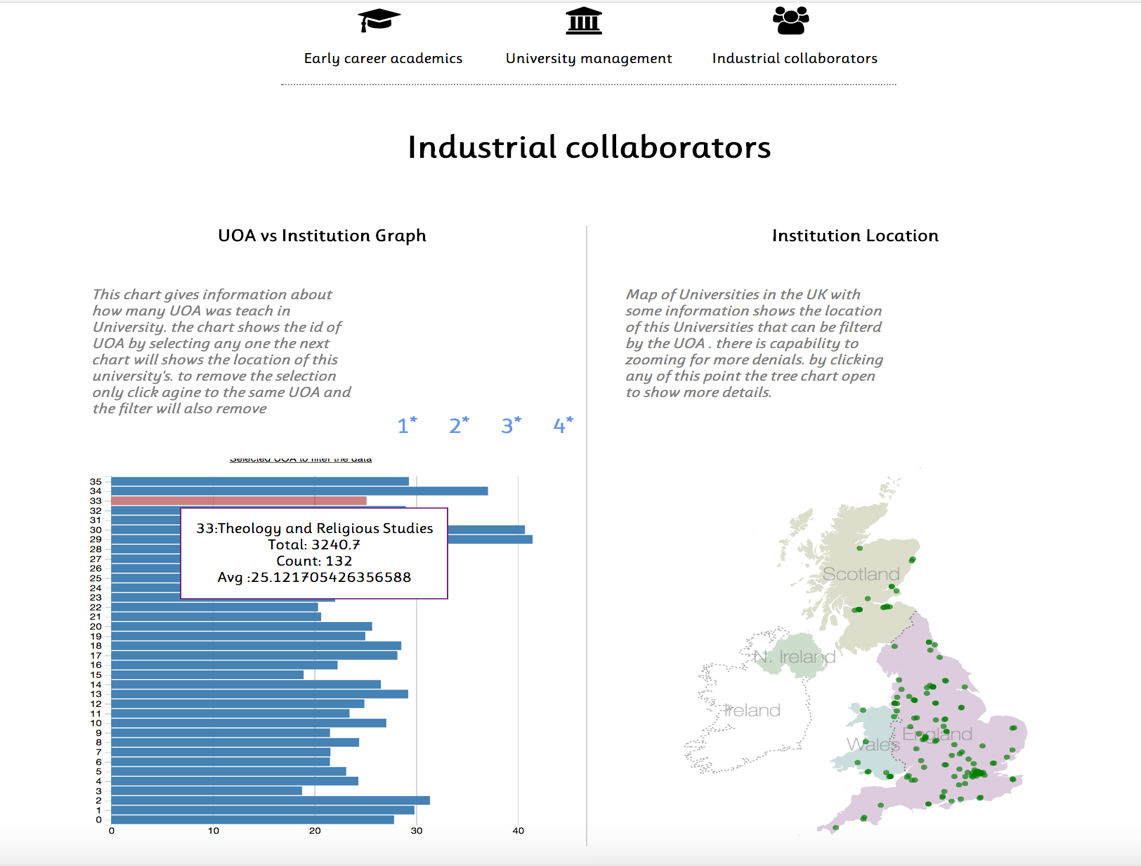
Figure

1. Industrial collaborators:

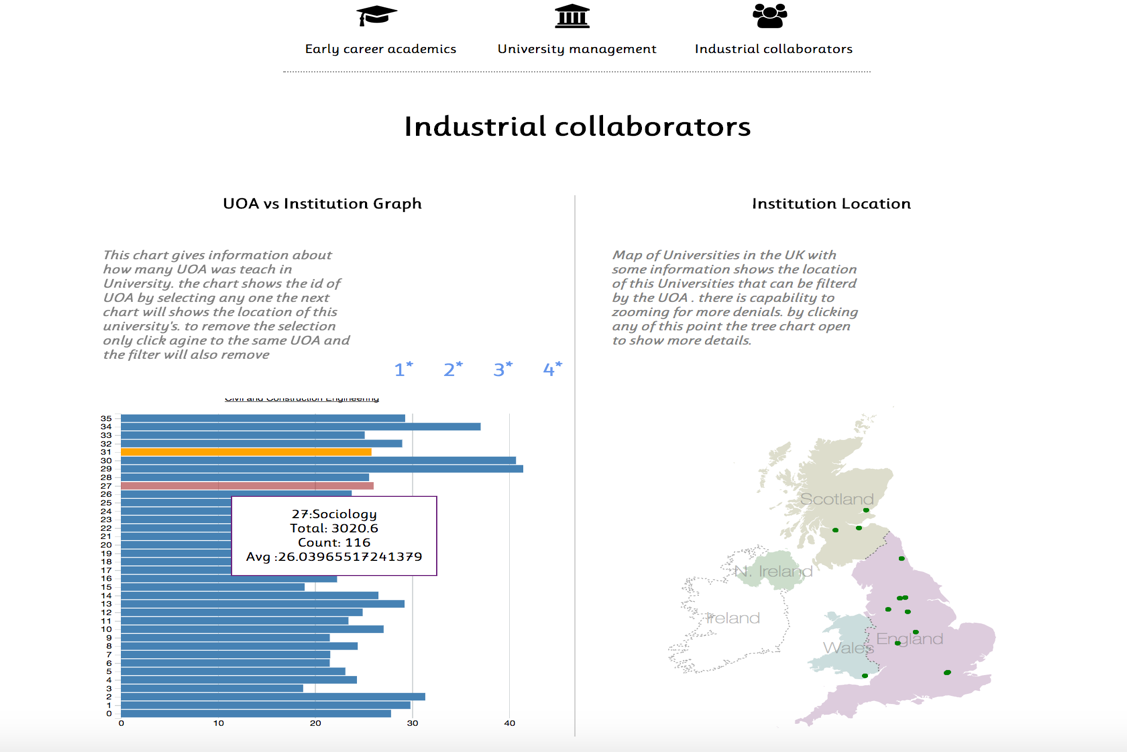
First thing appears once click on Early career is the Figure 13, that showing all the UOA in bar chart and all the Universities in map chart. And all other Figure from 14 to 17 explain more details.



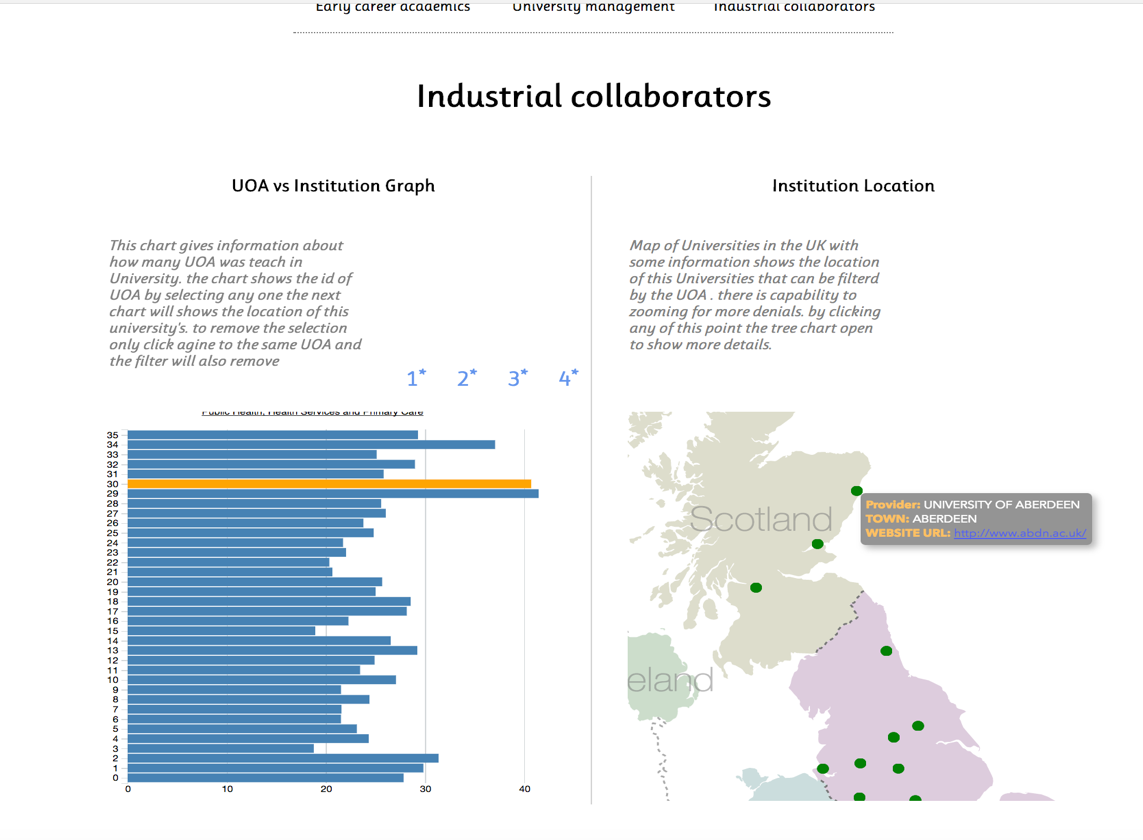
Figure



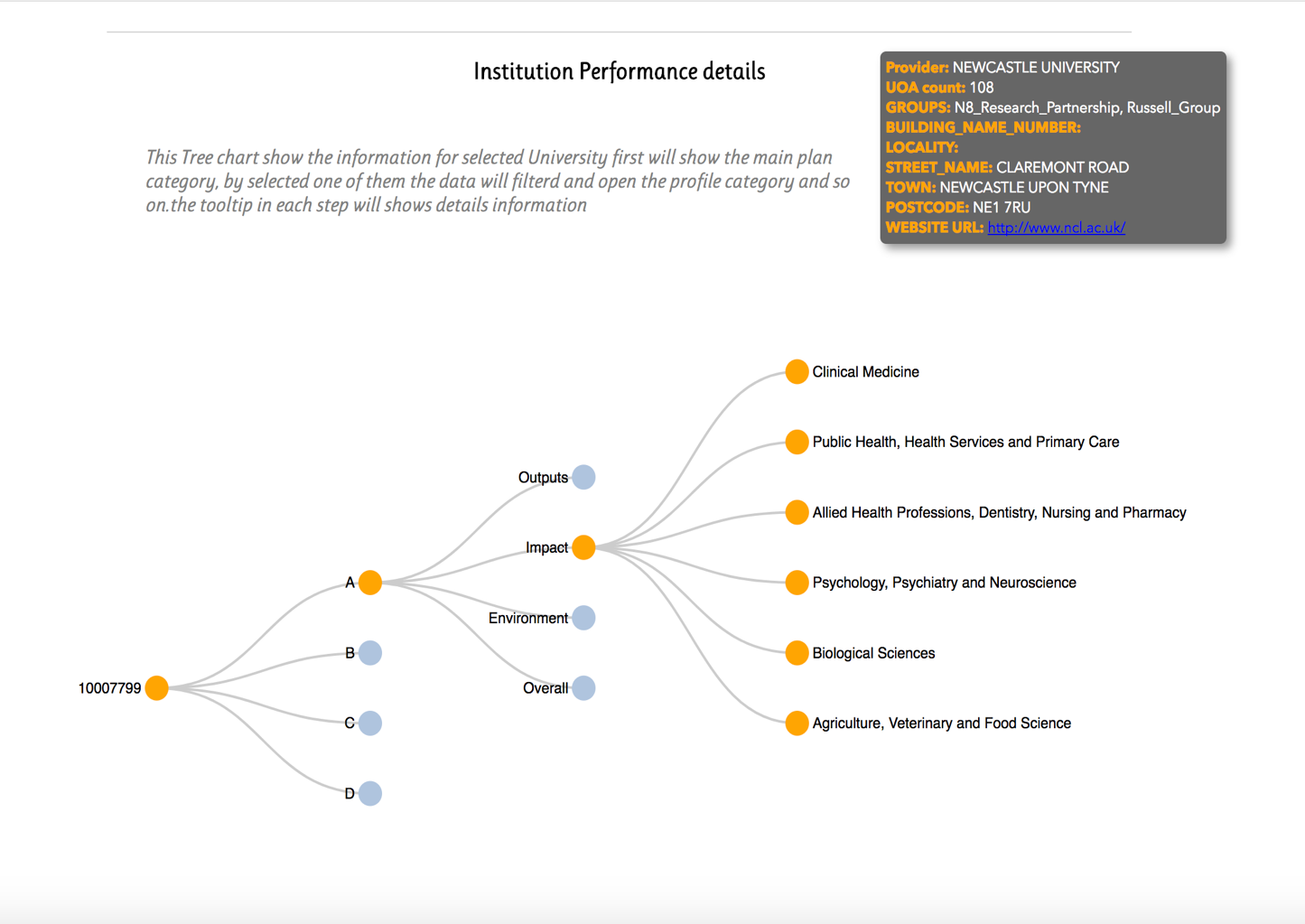
Figure



Figure



Figure



Figure

1. Application Design
   1. Design Overview:
   2. Reflection Design:
   3. Use of interactions, and transitions:
   4. Description of original features:
   5. Design for three different user types:
2. Conclusions:

The feature of the project that I am most proud of

The most interesting aspect of Data Visualisation that I have learned from the project

Reflection on any changes to your approach that you would make for the next project

1. Source code file list:

This is a table, which has one row for each of your source code files.

|  |  |  |
| --- | --- | --- |
| File name | By Me | by course or d3 modules |
| Indx.html | 90% | 10% course example |
| main.js | 70% | 30% course example and other |
| topojson.v1.min.js | 0% | Other source |
| d3.tip.js | 0% | 100 % d3 modules |
| d3.min.js | 0% | 100% d3 modules |

1. Source code listings:

main.js:

*<!--Author: Amer Eddin Al MAzloum-->  
<!--Date:13-Nov-2017-->  
<!--References (if any)-->  
  
// for the map charts:  
  
// https://bost.ocks.org/mike/map/  
// https://mono.software/2017/08/10/d3-js-map-of-croatia/  
// https://bl.ocks.org/d3noob/bf44061b1d443f455b3f857f82721372  
// https://bl.ocks.org/mbhall88/126b3c2c54215b4d1ffbd2d778ce3973  
// http://bl.ocks.org/d3noob/8375092  
  
// For tree chart  
  
  
// https://gist.github.com/d3noob/43a860bc0024792f8803bba8ca0d5ecd  
// http://bl.ocks.org/d3noob/8375092  
  
// Gerneral information  
  
// https://bl.ocks.org/alandunning/7008d0332cc28a826b37b3cf6e7bd998  
  
  
<!--Overall function of the code-->  
<!--Usage (a brief description of public methods, properties, and events)-->  
<!--The percentage of code written by the Me 70%-->  
<!--The percentage of code taken from course examples or other source 30%-->*

*<!--Author: Amer Eddin Al MAzloum-->  
<!--Date:13-Nov-2017-->  
<!--References (if any)-->  
  
// for the map charts:  
  
// https://bost.ocks.org/mike/map/  
// https://mono.software/2017/08/10/d3-js-map-of-croatia/  
// https://bl.ocks.org/d3noob/bf44061b1d443f455b3f857f82721372  
// https://bl.ocks.org/mbhall88/126b3c2c54215b4d1ffbd2d778ce3973  
// http://bl.ocks.org/d3noob/8375092  
  
// For tree chart  
  
  
// https://gist.github.com/d3noob/43a860bc0024792f8803bba8ca0d5ecd  
// http://bl.ocks.org/d3noob/8375092  
  
// Gerneral information  
  
// https://bl.ocks.org/alandunning/7008d0332cc28a826b37b3cf6e7bd998  
  
  
<!--Overall function of the code-->  
<!--Usage (a brief description of public methods, properties, and events)-->  
<!--The percentage of code written by the Me 70%-->  
<!--The percentage of code taken from course examples or other source 30%-->  
  
  
//=================== Career Module Charts =========================***function** *barchart\_cm*(targetDOMelement) {  
  
 *//Delare the main object that will be returned to caller* **var** barchartObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* barchartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.render = **function** (callbackFunction) {  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.loadAndRenderDataset = **function** (data, pdata) {  
 dataset = data;  
 InstitutionPointData = pdata;  
 *// console.log("dataset ",dataset);* barData = **d3**.*nest*()  
 *// .key(function(d) { return d.Profile; })* .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
 .**key**(**function** (d) {  
 **return** d[**"Institution code (UKPRN)"**];  
 })  
 .rollup(**function** (v) {  
 **return** v.**length**;  
 })  
 .entries(dataset);  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.width = **function** (w) {  
 svgWidth = w;  
 grp.attr(**"transform"**, **"translate("** + [w / 2, svgHeight / 2] + **")"**)  
  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 *//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas  
  
 // var tooltipclass = "toolTip"* **var** dataset = [];  
 **var** barData = [];  
 **var** InstitutionPointData = [];  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
  
 **var** x = **d3**.*scaleBand*()  
 .range([0, svgWidth], .1);  
  
 **var** y = **d3**.*scaleLinear*()  
 .range([svgHeight, 0]);  
  
 **var** xAxis = **d3**.*axisBottom*(x);  
  
 **var** yAxis = **d3**.*axisLeft*(y)  
 *//Declare and append tooltip that we will use to show tooltip barchart within the svg* **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
 svg.append(**"text"**)  
 .attr(**"x"**, (svgWidth / 2))  
 .attr(**"y"**, 2 - (margintop / 2))  
 .attr(**'class'**, **'bartitleselected'**)  
 .**text**(**"Selected UOA to filter the data"**);  
  
  
  
 **var** tip = **d3**.*tip*();  
  
  
 *//Declare and append group that we will use in the barchart within the svg  
  
  
 //=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
 *//Taken and addapted from https://github.com/d3/d3-shape/blob/master/README.md#pie* tip = **d3**.*tip*()  
 .attr(**'class'**, **'d3-tip'**)  
 .offset([-10, 0])  
 .**html**(**function** (d) {  
 **return "<strong>"** + d.**key** + **" :</strong> <span style='color:red'>"** + d.values.**length** + **"</span>"**;  
 });  
 svg.attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
  
 x.domain(barData.map(**function** (d) {  
 **return** barData.indexOf(d) + 1;  
 }))  
 .paddingInner(0.1)  
 .paddingOuter(0.5);  
  
 y.domain([0, **d3**.**max**(barData, **function** (d) {  
 **return** (d.values.**length** + margintop);  
 })]);  
  
  
 svg.append(**"g"**)  
 .attr(**"class"**, **"x axis"**)  
 .attr(**"transform"**, **"translate(0,"** + (svgHeight - 5) + **")"**)  
 .call(xAxis)  
 .call(tip);  
  
 svg.append(**"g"**)  
 .attr(**"class"**, **"y axis"**)  
 .call(yAxis);  
  
 svg.append(**"text"**)  
 .attr(**"transform"**,  
 **"translate("** + (svgWidth / 2) + **" ,"** + (svgHeight + marginbottom) + **")"**)  
 .*style*(**"text-anchor"**, **"middle"**)  
 .**text**(**"UOA"**);  
  
 svg.append(**"text"**)  
 .attr(**"class"**, **"label"**)  
 .attr(**"y"**, -(marginleft / 2))  
 .attr(**"x"**, -(margintop / 2) + 2)  
 .*style*(**"text-anchor"**, **"end"**)  
 .**text**(**"Count"**);  
  
 svg.*selectAll*(**".bar"**)  
 .data(barData)  
 .*enter*()  
 .append(**"rect"**)  
 .attr(**"class"**, **"bar"**)  
 *// .style("fill", function(d) {return color(barData.indexOf(d)+1);})* .attr(**"x"**, **function** (d) {  
 **return** x(barData.indexOf(d) + 1);  
 })  
 .attr(**"width"**, x.bandwidth())  
 .attr(**"y"**, **function** (d) {  
 **return** y(d.values.**length**) - 5;  
 })  
 .attr(**"height"**, **function** (d) {  
 **return** svgHeight - y(d.values.**length**);  
 })  
 .on(**'mouseover'**, tip.show)  
 .on(**'mouseout'**, tip.hide)  
 .on(**'click'**, **function** (d) {  
 *// console.log("this", this.classList.contains("barselected"));* **if** (**this**.**classList**.contains(**"barselected"**)) {  
  
 **d3**.*selectAll*(**".barselected"**).attr(**"class"**, **"bar"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(**"Selected UOA to filter the data"**);  
  
  
 *// careerModulel2.Update(InstitutionPointData);* } **else** {  
  
 **d3**.*selectAll*(**".barselected"**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"bar"**);  
  
 **d3**.select(**this**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"barselected"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(d.**key**);  
  
 *updateotherchart*(d);  
 }  
  
 });  
 svg.*selectAll*(**".barlabel"**)  
 .data(barData)  
 .*enter*()  
 .append(**"text"**)  
 .attr(**"class"**, **"barlabel"**)  
 .attr(**"x"**, (**function** (d) {  
 **return** x(barData.indexOf(d) + 1);  
 } ))  
 .attr(**"y"**, **function** (d) {  
 **return** y(d.values.**length**) + 10;  
 })  
 .attr(**"dy"**, **"0.001em"**)  
 .**text**(**function** (d) {  
 **return** d.values.**length**;  
 });  
  
  
 }  
  
 **function** *updateotherchart*(parm) {  
  
 **var** selected = [];  
 parm.values.forEach(*processUniversity*);  
  
 **function** *processUniversity*(dr) {  
 InstitutionPointData.forEach(**function** (rec) {  
 **if** (rec.UKPRN == dr.**key**) {  
 *// console.log("rec.UKPRN: ",rec.UKPRN,"row.key : ",dr.key);* **var** str = **'{ "UKPRN":'** + rec.UKPRN + **', "PROVIDER\_NAME": "'** + rec.PROVIDER\_NAME + **'", "LATITUDE":"'** + rec.LATITUDE + **'", "LONGITUDE":"'** + rec.LONGITUDE + **'", "TOWN":"'** + rec.TOWN + **'" , "WEBSITE\_URL":"'** + rec.WEBSITE\_URL + **'"}'**;  
 **var** result = ***JSON***.parse(str);  
 selected.push(result);  
 }  
 })  
 }  
  
 ***cm2***.Update(selected);  
 ***cm3***.clear();  
 }  
  
 **function** *type*(d) {  
 **return** d;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** barchartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *ukMap\_cm*(targetDOMelement, jsonMapData) {  
 *//Delare the main object that will be returned to caller* **var** mapObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* mapObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.render = **function** (callbackFunction) {  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.loadAndRenderDataset = **function** (jsonPointData) {  
 InstitutionPointData = jsonPointData;  
 *layoutAndRender*();  
  
 **return** mapObject;  
 }  
 mapObject.Update = **function** (Data) {  
 InstitutionPointData = Data;  
 *// console.log("InstitutionPointData",InstitutionPointData)  
 // layoutAndRender();  
 GUP\_towns*(svg, InstitutionPointData);  
 *// var circle= d3.selectAll("circle");  
 // repeat();  
 // function repeat() {  
 //  
 // circle  
 // .attr("opacity", .7)  
 // .transition() // apply a transition  
 // .duration(4000) // apply it over 2000 milliseconds  
 // .attr("fill", "red") // move the circle to 920 on the x axis  
 // .attr("r", "8px")  
 // .transition() // apply a transition  
 // .duration(4000) // apply it over 2000 milliseconds  
 // .attr("fill", "green") // return the circle to 40 on the x axis  
 // .attr("r", "4px")  
 // .on("end", repeat); // when the transition finishes start again  
 // };* **return** mapObject;  
 }  
 mapObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.width = **function** (w) {  
 svgWidth = w;  
 projection.translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.scale = **function** (scale) {  
 projectionscale = scale;  
 projection.scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
*//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas  
  
 // var tooltipclass = "toolTip"* **var** InstitutionPointData = [];  
  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
 **var** projectionscale = 2500, pointRadius = 2, active = **d3**.select(**null**);  
  
 **var** projection = **d3**.*geoAlbers*()  
 .center([0, 55.4])  
 .rotate([4.4, 0])  
 .parallels([50, 60])  
 .scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 3]);  
  
 *//Define path generator (takes projected 2D geometry and formats for SVG)* **var** path = **d3**.*geoPath*()  
 .*projection*(projection)  
 .pointRadius(pointRadius);  
  
 *//Create SVG* **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop \* 3 + **")"**);  
  
 svg.append(**"rect"**)  
 .attr(**"width"**, svgWidth)  
 .attr(**"height"**, svgHeight)  
 .*style*(**"fill"**, **"none"**)  
 .*style*(**"pointer-events"**, **"all"**)  
 .call(**d3**.*zoom*()  
 .scaleExtent([1 / 2, 4])  
 .on(**"zoom"**, *zoomed*));  
  
 **function** *zoomed*() {  
 svg.attr(**"transform"**, **d3**.**event**.**transform**);  
 }  
  
  
  
 **var** tooltip = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"tooltip"**)  
 .*style*(**"opacity"**, 0)  
 .*style*(**"width"**, 600);  
  
 *//=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
  
  
 *//Read in JSON file of UK map and do all the D3 stuff:* **d3**.**json**(jsonMapData, **function** (error, uk) {  
 **if** (error) **return console**.error(error);  
 **console**.log(uk);  
  
  
 *// Displaying Polygons* **var** subunits = **topojson**.*feature*(uk, uk.objects.subunits).features;  
 *GUP\_subunits*(svg, subunits, uk);  
 *GUP\_towns*(svg, InstitutionPointData);  
  
  
 });  
  
  
 *//* }  
  
 **function** *GUP\_subunits*(svg, subunits, uk) {  
  
 svg.append(**"path"**)  
 .*datum*(subunits)  
 .attr(**"d"**, path);  
 **console**.log(**"svg"**, svg);  
  
 *// Styling Polygons* svg.*selectAll*(**".subunit"**)  
 .data(subunits)  
 .*enter*()  
 .append(**"path"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit "** + d.**id**;  
 })  
 .attr(**"d"**, path);  
  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 **return** a !== b && a.**id** !== **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary"**);  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 *// console.log("a id",a.id);* **return** a === b && a.**id** === **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary IRL"**);  
  
 *// Country Labels* svg.*selectAll*(**".subunit-label"**)  
 .data(subunits)  
 .*enter*().append(**"text"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit-label "** + d.**id**;  
 })  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + path.centroid(d) + **")"**;  
 })  
 .attr(**"dy"**, **".35em"**)  
 .**text**(**function** (d) {  
 **return** d.**properties**.**name**;  
 });  
  
  
 }  
  
 **function** *GUP\_towns*(svg, places) {  
  
 **var** circle = svg.*selectAll*(**"circle"**);  
  
  
 circle  
 .remove().*exit*()  
 .data(InstitutionPointData)  
 .*enter*()  
 .append(**"circle"**)  
 .attr(**"cx"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[0];  
 })  
 .attr(**"cy"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[1];  
 })  
 .attr(**"r"**, **"3px"**)  
 .attr(**"fill"**, **"green"**)  
 .attr(**"opacity"**, .7)  
 .on(**"mouseover"**, **function** (d) {  
 *// var tip = "<h3>" + d.PROVIDER\_NAME + "</h3>";* **var** t = **"<strong>Provider: </strong><span class='details'>"** + d.PROVIDER\_NAME + **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + d.TOWN + **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + d.WEBSITE\_URL + **"\" >"** + d.WEBSITE\_URL + **"</a></span>"**;  
  
 tooltip  
 .**html**(t)  
 .*style*(**"left"**, (**d3**.**event**.**pageX**) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**)  
 .*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, .7);  
  
  
 })  
 .on(**"mouseout"**, **function** (d) {  
 tooltip.*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, 0);  
 })  
 .on(**"click"**, **function** (d) {  
 ***cm3***.loadAndRenderDataset(*preperTreeData*(d));  
  
  
 });  
  
  
 }  
  
  
  
 **function** *preperTreeData*(selectedpoint) {  
  
 **console**.log(**"selectedpoint"**, selectedpoint);  
 **DataFiltered** = **ref14data**.filter(**function** (d) {  
  
 **if** (d[**"Institution code (UKPRN)"**] == selectedpoint.UKPRN) {  
 **return** d  
 }  
 });  
 **console**.log(**"DataFiltered"**, **DataFiltered**);  
  
 **var** treeData = **d3**.*nest*()  
 .**key**(**function** (d) {  
 **return** d[**"Main panel"**];  
 })  
 .**key**(**function** (d) {  
 **return** d[**"Profile"**];  
 })  
 .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
  
 .rollup(**function** (v) {  
 **return** v  
 })  
 .entries(**DataFiltered**);  
  
 *// console.log("treeData",treeData);* **var** refJSON = ***JSON***.stringify(treeData);  
 refJSON = refJSON.replace(**new** RegExp(**'values'**, **'g'**), **'children'**);  
 refJSON = refJSON.replace(**new** RegExp(**'key'**, **'g'**), **'name'**);  
  
 **var** jsonObj = ***JSON***.parse(refJSON);  
  
  
 **var** augRefJSON = {**"name"**: selectedpoint.UKPRN, **"children"**: jsonObj};  
  
  
 **return** augRefJSON;  
 }  
  
 **function** *reset*() {  
 active.*classed*(**"active"**, **false**);  
 active = **d3**.select(**null**);  
  
 svg.*transition*()  
 .**duration**(750)  
 .call(*zoom*.transform, **d3**.zoomIdentity); *// updated for d3 v4* }  
  
  
 *//================== IMPORTANT do not delete ==================================* **return** mapObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *treechart\_cm*(targetDOMelement) {  
*//Delare the main object that will be returned to caller* **var** treechartObject = {};  
 *//=================== PRIVATE VARIABLES ====================================  
// Set the dimensions and margins of the diagram* **var** margin = {**top**: 20, **right**: 90, **bottom**: 30, **left**: 90},  
 width = 960 - margin.**left** - margin.**right**,  
 height = 500 - margin.**top** - margin.**bottom**;  
 **var** i = 0,  
 duration = 750,  
 root;  
 **var** treeData = [];  
 **var** nodes;  
  
 **var** div = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"Treetooltip"**)  
 .*style*(**"opacity"**, 1e-6);  
  
 **var** aside = **d3**.select(targetDOMelement).append(**"aside"**)  
 .attr(**"class"**, **"selecteinfo"**);  
  
*// declares a tree layout and assigns the size* **var** treemap = **d3**.*tree*().size([height, width]);  
  
  
 **var** svg = **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** + **")"**);  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* treechartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.clear = **function** () {  
  
  
 svg.*style*(**"display"**, **"none"**);  
 aside.*style*(**"display"**, **"none"**);  
  
 **return** treechartObject;  
 }  
 treechartObject.loadAndRenderDataset = **function** (data) {  
  
 treeData = data;  
 *// Assigns parent, children, height, depth* root = **d3**.*hierarchy*(treeData, **function** (d) {  
 **return** d.**children**;  
 });  
 root.**x0** = height / 2;  
 root.**y0** = 0;  
 *// Collapse after the second level* root.**children**.forEach(*collapse*);  
 *update*(root);  
 *// Collapse the node and all it's children* **function** *collapse*(d) {  
 **if** (d.**children**) {  
 d.**\_children** = d.**children** d.**\_children**.forEach(*collapse*)  
 d.**children** = **null** }  
 }  
  
 **return** treechartObject;  
 }  
  
 **function** *update*(source) {  
 *// Assigns the x and y position for the nodes* **var** treeData = treemap(root);  
 *addInfoData*(treeData);  
 svg.*style*(**"display"**, **"block"**);  
 aside.*style*(**"display"**, **"block"**);  
 *// Compute the new tree layout.* nodes = treeData.descendants(),  
 **links** = treeData.descendants().slice(1);  
 *// Normalize for fixed-depth.* nodes.forEach(**function** (d) {  
 d.**y** = d.**depth** \* 180  
 });  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Nodes section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the nodes...* **var** node = svg.*selectAll*(**'g.node'**)  
 .data(nodes, **function** (d) {  
 **return** d.**id** || (d.**id** = ++i);  
 });  
 *// Enter any new modes at the parent's previous position.* **var** nodeEnter = node.*enter*().append(**'g'**)  
 .attr(**'class'**, **'node'**)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y0** + **","** + source.**x0** + **")"**;  
 })  
 .on(**'click'**, *click*);  
 *// Add Circle for the nodes* nodeEnter.append(**'circle'**)  
 .on(**"mouseover"**, *mouseover*)  
 .on(**"mousemove"**, **function** (d) {  
 *mousemove*(d);  
 })  
 .on(**"mouseout"**, *mouseout*)  
 .attr(**'class'**, **'node'**)  
 .attr(**'r'**, 1e-6)  
 .*style*(**"fill"**, **function** (d) {  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#fff"**;  
 });  
 *// Add labels for the nodes* nodeEnter.append(**'text'**)  
 .attr(**"dy"**, **".35em"**)  
 .attr(**"x"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? -13 : 13;  
 })  
 .attr(**"text-anchor"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? **"end"** : **"start"**;  
 })  
 .**text**(**function** (d) {  
 **return** d.**data**.**name**;  
 });  
 *// UPDATE* **var** nodeUpdate = nodeEnter.*merge*(node);  
 *// Transition to the proper position for the node* nodeUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + d.**y** + **","** + d.**x** + **")"**;  
 });  
 *// Update the node attributes and style* nodeUpdate.select(**'circle.node'**)  
 .attr(**'r'**, 10)  
 .*style*(**"fill"**, **function** (d) {  
  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#ffa500"**;  
 })  
 .attr(**'cursor'**, **'pointer'**);  
 *// Remove any exiting nodes* **var** nodeExit = node.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y** + **","** + source.**x** + **")"**;  
 })  
 .remove();  
 *// On exit reduce the node circles size to 0* nodeExit.select(**'circle'**)  
 .attr(**'r'**, 1e-6);  
 *// On exit reduce the opacity of text labels* nodeExit.select(**'text'**)  
 .*style*(**'fill-opacity'**, 1e-6);  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* links section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the links...* **var** link = svg.*selectAll*(**'path.link'**)  
 .data(**links**, **function** (d) {  
 **return** d.**id**;  
 });  
 *// Enter any new links at the parent's previous position.* **var** linkEnter = link.*enter*().*insert*(**'path'**, **"g"**)  
 .attr(**"class"**, **"link"**)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x0**, **y**: source.**y0**}  
 **return** *diagonal*(o, o)  
 });  
 *// UPDATE* **var** linkUpdate = linkEnter.*merge*(link);  
 *// Transition back to the parent element position* linkUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **return** *diagonal*(d, d.**parent**)  
 });  
 *// Remove any exiting links* **var** linkExit = link.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x**, **y**: source.**y**}  
 **return** *diagonal*(o, o)  
 })  
 .remove();  
 *// Store the old positions for transition.* nodes.forEach(**function** (d) {  
 d.**x0** = d.**x**;  
 d.**y0** = d.**y**;  
 });  
  
 *// Creates a curved (diagonal) path from parent to the child nodes* **function** *diagonal*(s, d) {  
 *path* = **`M** ${s.**y**} ${s.**x**} **C** ${(s.**y** + d.**y**) / 2} ${s.**x**}**,** ${(s.**y** + d.**y**) / 2} ${d.**x**}**,** ${d.**y**} ${d.**x**}**`  
 return path** }  
  
 **function** *mouseover*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1);  
 }  
  
 **function** *mousemove*(d) {  
 *// console.log("d data",d);* **var** toolttiptext;  
 *// calcUniversityperformance(treeData);* **if** (d.**depth** == 0) {  
 toolttiptext = *getUniversity*(d, **"html"**);  
 } **else** {  
 **var** perdata = *getPerformance*(d);  
 **if** (perdata.**CountofUOA** > 1) {  
 toolttiptext = **"<strong>UOA No:\*: </strong><span class='details'>"** + perdata.**CountofUOA** +  
 **"<br/></span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 } **else** {  
 toolttiptext = **"</span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 }  
  
  
 }  
  
 div.**html**(toolttiptext)  
 .*style*(**"left"**, (**d3**.**event**.**pageX** ) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**);  
 }  
  
 **function** *mouseout*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1e-6);  
 }  
  
 *// Toggle children on click.* **function** *click*(d) {  
  
 **if** (d.**children**) {  
 d.**\_children** = d.**children**;  
 d.**children** = **null**;  
  
 *// div.transition()  
 // .duration(500)  
 // .style("opacity", 0);  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", 0);* } **else** {  
  
 *// ref14data.forEach(findlastnode);  
 // function findlastnode(dr){  
 //  
 // if (dr["Unit of assessment name"]==d.data.name){  
 // console.log("d.data.name",dr["Unit of assessment name"]);  
 //  
 // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span>";  
 // // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \"" + d.WEBSITE\_URL + "\" >" + d.WEBSITE\_URL + "</a></span>";  
 //  
 // tooltip.html(t)  
 // .style("left", (d3.event.pageX) + "px")  
 // .style("top", (d3.event.pageY) + "px");  
 //  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", .7);  
 //  
 // }}* d.**children** = d.**\_children**;  
 d.**\_children** = **null**;  
 }  
 *update*(d);  
 }  
  
  
 }  
  
 **function** *addInfoData*(objdata) {  
 **console**.log(**"treeData.data"**, objdata);  
 **var** textinfo = *getUniversity*(objdata, **"html"**);  
 aside.**html**(textinfo);  
  
 }  
  
 **function** *getUniversity*(un, output) {  
 **var** result;  
 **console**.log(**"d.data"**, un);  
 **learningProviders**.forEach(**function** (rec) {  
  
 **if** (rec.UKPRN == un.**data**.**name**) {  
 **if** (output == **"html"**) {  
 result = **"<strong>Provider: </strong><span class='details'>"** + rec.PROVIDER\_NAME +  
 **"<br/></span> <strong>UOA count: </strong> <span class='details'>"** + *getPerformance*(un).**CountofUOA** +  
 **"<br/></span> <strong>GROUPS: </strong> <span class='details'>"** + rec.GROUPS +  
 **"<br/></span> <strong>BUILDING\_NAME\_NUMBER: </strong> <span class='details'>"** + rec.BUILDING\_NAME\_NUMBER +  
 **"<br/></span> <strong>LOCALITY: </strong> <span class='details'>"** + rec.LOCALITY +  
 **"<br/></span> <strong>STREET\_NAME: </strong> <span class='details'>"** + rec.STREET\_NAME +  
 **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + rec.TOWN +  
 **"<br/></span> <strong>POSTCODE: </strong> <span class='details'>"** + rec.POSTCODE +  
 **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + rec.WEBSITE\_URL + **"\" >"** + rec.WEBSITE\_URL +  
 **"</a></span>"**;  
 } **else** {  
 result = [rec.PROVIDER\_NAME, calcUniversityperformance(), rec.GROUPS, rec.BUILDING\_NAME\_NUMBER, rec.LOCALITY, rec.STREET\_NAME, rec.TOWN, rec.POSTCODE, rec.WEBSITE\_URL]  
 }  
  
  
 }  
 **return** result;  
 });  
  
 **return** result;  
 }  
  
 **function** *getPerformance*(obj) {  
 **var** Performance = {**CountofUOA**: 0, **sumOfOne**: 0, **sumOftwo**: 0, **sumOftree**: 0, **sumOffour**: 0, **sumFTB**: 0};  
  
 **function** *getLeafNodes*(leafNodes, obj) {  
  
 **if** (obj.**children**) {  
  
 obj.**children**.forEach(**function** (child) {  
 *getLeafNodes*(leafNodes, child)  
  
 });  
 } **else** {  
  
 leafNodes.push(obj);  
 }  
 }  
  
 **var** leafNodes = [];  
 *getLeafNodes*(leafNodes, obj.**data**);  
 Performance.**CountofUOA** = leafNodes.**length**;  
 leafNodes.forEach(**function** (child) {  
 **var** childobject = child.**value**[0];  
 *// console.log("childobject[\"1\*\"]",childobject["1\*"])* Performance.**sumOfOne** += ***Math***.round(parseFloat(childobject[**"1\*"**]));  
 Performance.**sumOftwo** += ***Math***.round(parseFloat(childobject[**"2\*"**]));  
 Performance.**sumOftree** += ***Math***.round(parseFloat(childobject[**"3\*"**]));  
 Performance.**sumOffour** += ***Math***.round(parseFloat(childobject[**"4\*"**]));  
 Performance.**sumFTB** += ***Math***.round(parseFloat(childobject[**"FTE Category A staff submitted"**])) / Performance.**CountofUOA**;  
 });  
  
 **return** Performance;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** treechartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
*//=================== University Management Charts =========================***function** *barchart\_um*(targetDOMelement) {  
  
 *//Delare the main object that will be returned to caller* **var** barchartObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* barchartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.render = **function** (callbackFunction) {  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.loadAndRenderDataset = **function** (data, pdata) {  
 dataset = data;  
 InstitutionPointData = pdata;  
 *// console.log("dataset ",dataset);* barData = **d3**.*nest*()  
 *// .key(function(d) { return d.Profile; })* .**key**(**function** (d) {  
 **return** d[**"Main panel"**];  
 })  
 .rollup(**function** (v) {  
 **console**.log(**"v"**, v);  
 **return** v.**length**;  
 })  
 .entries(dataset);  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.width = **function** (w) {  
 svgWidth = w;  
 grp.attr(**"transform"**, **"translate("** + [w / 2, svgHeight / 2] + **")"**)  
  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 *//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas* **var** dataset = [];  
 **var** barData = [];  
 **var** InstitutionPointData = [];  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
  
 **var** x = **d3**.*scaleBand*()  
 .range([0, svgWidth], .1);  
  
 **var** y = **d3**.*scaleLinear*()  
 .range([svgHeight, 0]);  
  
 **var** xAxis = **d3**.*axisBottom*(x);  
  
 **var** yAxis = **d3**.*axisLeft*(y)  
 *//Declare and append tooltip that we will use to show tooltip barchart within the svg* **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
 svg.append(**"text"**)  
 .attr(**"x"**, (svgWidth / 2))  
 .attr(**"y"**, 2 - (margintop / 2))  
 .attr(**'class'**, **'bartitleselected'**)  
 .**text**(**"Selected Main panel to filter the data"**);  
  
 **var** tip = **d3**.*tip*();  
  
  
 *//Declare and append group that we will use in the barchart within the svg  
  
  
 //=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
 **console**.log(**"barData "**, barData);  
 *//Taken and addapted from https://github.com/d3/d3-shape/blob/master/README.md#pie* tip = **d3**.*tip*()  
 .attr(**'class'**, **'d3-tip'**)  
 .offset([-10, 0])  
 .**html**(**function** (d) {  
 **return "<strong>"** + d.**key** + **" :</strong> <span style='color:red'>"** + d.**value** + **"</span>"**;  
 });  
 svg.attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
  
 x.domain(barData.map(**function** (d) {  
 **return** d.**key**;  
 }))  
 .paddingInner(0.1)  
 .paddingOuter(0.5);  
 **console**.log(**"d.value "**, barData[3].**value**);  
 y.domain([0, **d3**.**max**(barData, **function** (d) {  
 **return** (d.**value** + margintop);  
 })]);  
  
  
 svg.append(**"g"**)  
 .attr(**"class"**, **"x axis"**)  
 .attr(**"transform"**, **"translate(0,"** + (svgHeight - 5) + **")"**)  
 .call(xAxis)  
 .call(tip);  
  
 svg.append(**"g"**)  
 .attr(**"class"**, **"y axis"**)  
 .call(yAxis);  
  
 svg.append(**"text"**)  
 .attr(**"transform"**,  
 **"translate("** + (svgWidth / 2) + **" ,"** + (svgHeight + marginbottom) + **")"**)  
 .*style*(**"text-anchor"**, **"middle"**)  
 .**text**(**"UOA"**);  
  
 svg.append(**"text"**)  
 .attr(**"class"**, **"label"**)  
 .attr(**"y"**, -(marginleft / 2))  
 .attr(**"x"**, -(margintop / 2) + 2)  
 .*style*(**"text-anchor"**, **"end"**)  
 .**text**(**"Count"**);  
  
 svg.*selectAll*(**".bar"**)  
 .data(barData)  
 .*enter*()  
 .append(**"rect"**)  
 .attr(**"class"**, **"bar"**)  
 *// .style("fill", function(d) {return color(barData.indexOf(d)+1);})* .attr(**"x"**, **function** (d) {  
 **return** x(d.**key**);  
 })  
 .attr(**"width"**, x.bandwidth())  
 .attr(**"y"**, **function** (d) {  
 **return** y(d.**value**) - 5;  
 })  
 .attr(**"height"**, **function** (d) {  
 **return** svgHeight - y(d.**value**);  
 })  
 .on(**'mouseover'**, tip.show)  
 .on(**'mouseout'**, tip.hide)  
 .on(**'click'**, **function** (d) {  
 *// console.log("this", this.classList.contains("barselected"));* **if** (**this**.**classList**.contains(**"barselected"**)) {  
  
 **d3**.*selectAll*(**".barselected"**).attr(**"class"**, **"bar"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(**"Selected UOA to filter the data"**);  
  
 *// careerModulel2.Update(InstitutionPointData);* } **else** {  
  
 **d3**.*selectAll*(**".barselected"**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"bar"**);  
  
 **d3**.select(**this**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"barselected"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(d.**key**);  
  
 *updateotherchart*(d);  
 }  
  
 });  
 svg.*selectAll*(**".barlabel"**)  
 .data(barData)  
 .*enter*()  
 .append(**"text"**)  
 .attr(**"class"**, **"barlabel"**)  
 .attr(**"x"**, (**function** (d) {  
 **return** x(d.**key**) + x.bandwidth() / 2;  
 } ))  
 .attr(**"y"**, **function** (d) {  
 **return** y(d.**value**) + 50;  
 })  
 .attr(**"dy"**, **"0.001em"**)  
 .**text**(**function** (d) {  
 **return** d.**value**;  
 });  
  
  
 }  
  
 **function** *updateotherchart*(parm) {  
 **console**.log(**"parm"**, parm);  
 *updatemapchart*(parm);  
  
 **function** *updatemapchart*(parm) {  
  
 **var** selected = [];  
  
 **var** filtereddata = dataset.filter(**function** (d) {  
 **return** d[**"Main panel"**] == parm.**key**;  
 });  
 **console**.log(**"filtereddata"**, filtereddata);  
  
 filtereddata.forEach(*processUniversity*);  
  
 **function** *processUniversity*(dr) {  
 InstitutionPointData.forEach(**function** (rec) {  
  
 **if** (rec.UKPRN == dr[**"Institution code (UKPRN)"**]) {  
 *// console.log("rec.UKPRN: ",rec.UKPRN,"row.key : ",dr.key);* **var** str = **'{ "UKPRN":'** + rec.UKPRN + **', "PROVIDER\_NAME": "'** + rec.PROVIDER\_NAME + **'", "Main\_panel": "'** + dr[**"Main panel"**] + **'", "LATITUDE":"'** + rec.LATITUDE + **'", "LONGITUDE":"'** + rec.LONGITUDE + **'", "TOWN":"'** + rec.TOWN + **'" , "WEBSITE\_URL":"'** + rec.WEBSITE\_URL + **'"}'**;  
  
 **var** result = ***JSON***.parse(str);  
 selected.push(result);  
 }  
 })  
 }  
  
 *// svg.selectAll(".d3-tip").style("fill","red");* **console**.log(**"selected Data"**, selected);  
 ***um2***.Update(selected);  
 }  
  
 ***um3***.clear();  
  
  
 }  
  
 **function** *type*(d) {  
 *// d.values.length = +d.values.length;* **return** d;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** barchartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *ukMap\_um*(targetDOMelement, jsonMapData) {  
 *//Delare the main object that will be returned to caller* **var** mapObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* mapObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.render = **function** (callbackFunction) {  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.loadAndRenderDataset = **function** (jsonPointData) {  
 InstitutionPointData = jsonPointData;  
 *layoutAndRender*();  
  
 **return** mapObject;  
 }  
 mapObject.Update = **function** (Data) {  
 InstitutionPointData = Data;  
 *// layoutAndRender();  
 GUP\_towns*(svg, InstitutionPointData);  
  
 **return** mapObject;  
 }  
 mapObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.width = **function** (w) {  
 svgWidth = w;  
 projection.translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.scale = **function** (scale) {  
 projectionscale = scale;  
 projection.scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
*//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas  
  
 // var tooltipclass = "toolTip"* **var** InstitutionPointData = [];  
  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
 **var** projectionscale = 2500, pointRadius = 2, active = **d3**.select(**null**);  
  
 **var** projection = **d3**.*geoAlbers*()  
 .center([0, 55.4])  
 .rotate([4.4, 0])  
 .parallels([50, 60])  
 .scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 3]);  
  
 *//Define path generator (takes projected 2D geometry and formats for SVG)* **var** path = **d3**.*geoPath*()  
 .*projection*(projection)  
 .pointRadius(pointRadius);  
  
 *//Create SVG* **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop \* 3 + **")"**);  
  
 svg.append(**"rect"**)  
 .attr(**"width"**, svgWidth)  
 .attr(**"height"**, svgHeight)  
 .*style*(**"fill"**, **"none"**)  
 .*style*(**"pointer-events"**, **"all"**)  
 .call(**d3**.*zoom*()  
 .scaleExtent([1 / 2, 4])  
 .on(**"zoom"**, *zoomed*));  
  
 **function** *zoomed*() {  
 svg.attr(**"transform"**, **d3**.**event**.**transform**);  
 }  
  
  
 **var** tooltip = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"tooltip"**)  
 .*style*(**"opacity"**, 0)  
 .*style*(**"width"**, 600);  
  
 *//=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
  
  
 *//Read in JSON file of UK map and do all the D3 stuff:* **d3**.**json**(jsonMapData, **function** (error, uk) {  
 **if** (error) **return console**.error(error);  
 **console**.log(uk);  
  
  
 *// Displaying Polygons* **var** subunits = **topojson**.*feature*(uk, uk.objects.subunits).features;  
 *// var places = topojson.feature(uk, uk.objects.places).features;  
 GUP\_subunits*(svg, subunits, uk);  
 *GUP\_towns*(svg, InstitutionPointData);  
  
  
 });  
  
  
 }  
  
 **function** *GUP\_subunits*(svg, subunits, uk) {  
  
 svg.append(**"path"**)  
 .*datum*(subunits)  
 .attr(**"d"**, path);  
 **console**.log(**"svg"**, svg);  
  
 *// Styling Polygons* svg.*selectAll*(**".subunit"**)  
 .data(subunits)  
 .*enter*()  
 .append(**"path"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit "** + d.**id**;  
 })  
 .attr(**"d"**, path);  
  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 **return** a !== b && a.**id** !== **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary"**);  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 *// console.log("a id",a.id);* **return** a === b && a.**id** === **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary IRL"**);  
  
 *// Country Labels* svg.*selectAll*(**".subunit-label"**)  
 .data(subunits)  
 .*enter*().append(**"text"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit-label "** + d.**id**;  
 })  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + path.centroid(d) + **")"**;  
 })  
 .attr(**"dy"**, **".35em"**)  
 .**text**(**function** (d) {  
 **return** d.**properties**.**name**;  
 });  
  
  
 }  
  
 **function** *GUP\_towns*(svg, places) {  
  
 **var** circle = svg.*selectAll*(**"circle"**);  
  
 circle  
 .remove().*exit*()  
 .data(InstitutionPointData)  
 .*enter*()  
 .append(**"circle"**)  
 .attr(**"cx"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[0];  
 })  
 .attr(**"cy"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[1];  
 })  
 .attr(**"r"**, **"3px"**)  
 .attr(**"fill"**, **"green"**)  
 .attr(**"opacity"**, .7)  
 .on(**"mouseover"**, **function** (d) {  
 *// var tip = "<h3>" + d.PROVIDER\_NAME + "</h3>";* **var** t = **"<strong>Provider: </strong><span class='details'>"** + d.PROVIDER\_NAME + **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + d.TOWN + **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + d.WEBSITE\_URL + **"\" >"** + d.WEBSITE\_URL + **"</a></span>"**;  
  
 tooltip  
 .**html**(t)  
 .*style*(**"left"**, (**d3**.**event**.**pageX**) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**)  
 .*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, .7);  
  
  
 })  
 .on(**"mouseout"**, **function** (d) {  
 tooltip.*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, 0);  
 })  
 .on(**"click"**, **function** (d) {  
 ***um3***.loadAndRenderDataset(*preperTreeData*(d));  
  
 });  
  
 }  
  
  
 **function** *preperTreeData*(selectedpoint) {  
 **console**.log(**"selectedpoint"**, selectedpoint);  
 **DataFiltered** = **ref14data**.filter(**function** (d) {  
  
 **if** (d[**"Institution code (UKPRN)"**] == selectedpoint.UKPRN && d[**"Main panel"**] == selectedpoint.Main\_panel) {  
 **return** d  
 }  
 });  
 **console**.log(**"DataFiltered"**, **DataFiltered**);  
  
 **var** treeData = **d3**.*nest*()  
 *// .key(function(d) { return d["Main panel"]; })* .**key**(**function** (d) {  
 **return** d[**"Profile"**];  
 })  
 *// .key(function(d) { return d["FTE Category A staff submitted"]; })* .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
  
 .rollup(**function** (v) {  
 **return** v  
 })  
 .entries(**DataFiltered**);  
  
 *// console.log("treeData",treeData);* **var** refJSON = ***JSON***.stringify(treeData);  
 refJSON = refJSON.replace(**new** RegExp(**'values'**, **'g'**), **'children'**);  
 refJSON = refJSON.replace(**new** RegExp(**'key'**, **'g'**), **'name'**);  
  
 **var** jsonObj = ***JSON***.parse(refJSON);  
  
 *// console.log("jsonObj",jsonObj);* **var** augRefJSON = {**"name"**: selectedpoint.UKPRN, **"children"**: jsonObj};  
  
 *// console.log("augRefJSON",augRefJSON);* **return** augRefJSON;  
 }  
  
 **function** *reset*() {  
 active.*classed*(**"active"**, **false**);  
 active = **d3**.select(**null**);  
  
 svg.*transition*()  
 .**duration**(750)  
 *// .call( zoom.transform, d3.zoomIdentity.translate(0, 0).scale(1) ); // not in d3 v4* .call(*zoom*.transform, **d3**.zoomIdentity); *// updated for d3 v4* }  
  
  
 *//================== IMPORTANT do not delete ==================================* **return** mapObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *treechart\_um*(targetDOMelement) {  
*//Delare the main object that will be returned to caller* **var** treechartObject = {};  
 *//=================== PRIVATE VARIABLES ====================================  
// Set the dimensions and margins of the diagram* **var** margin = {**top**: 20, **right**: 90, **bottom**: 30, **left**: 90},  
 width = 960 - margin.**left** - margin.**right**,  
 height = 500 - margin.**top** - margin.**bottom**;  
 **var** i = 0,  
 duration = 750,  
 root;  
 **var** treeData = [];  
 **var** nodes;  
 *// var div = d3.select(targetDOMelement).append("div")  
 // .attr("class", "tooltip")  
 // .style("opacity", 0);  
  
 // var tooltip= d3.select("body").append("div")  
 // .attr("class", "Treetooltip")  
 // .style("opacity", 0)  
 // .style("width", 600);* **var** div = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"Treetooltip"**)  
 .*style*(**"opacity"**, 1e-6);  
  
 **var** aside = **d3**.select(targetDOMelement).append(**"aside"**)  
 .attr(**"class"**, **"selecteinfo"**);  
  
*// declares a tree layout and assigns the size* **var** treemap = **d3**.*tree*().size([height, width]);  
  
*// append the svg object to the body of the page  
// appends a 'group' element to 'svg'  
// moves the 'group' element to the top left margin* **var** svg = **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** + **")"**);  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* treechartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.clear = **function** () {  
  
  
 svg.*style*(**"display"**, **"none"**);  
 aside.*style*(**"display"**, **"none"**);  
  
 **return** treechartObject;  
 }  
 treechartObject.loadAndRenderDataset = **function** (data) {  
 *// console.log("treeData inside",treeData);* svg.*style*(**"display"**, **"block"**);  
 aside.*style*(**"display"**, **"block"**);  
 treeData = data;  
 *// Assigns parent, children, height, depth* root = **d3**.*hierarchy*(treeData, **function** (d) {  
 **return** d.**children**;  
 });  
 root.**x0** = height / 2;  
 root.**y0** = 0;  
 *// Collapse after the second level* root.**children**.forEach(*collapse*);  
 *update*(root);  
 *// addInfoData();  
 // Collapse the node and all it's children* **function** *collapse*(d) {  
 **if** (d.**children**) {  
 d.**\_children** = d.**children** d.**\_children**.forEach(*collapse*)  
 d.**children** = **null** }  
 }  
  
 **return** treechartObject;  
 }  
  
 **function** *update*(source) {  
 *// Assigns the x and y position for the nodes* **var** treeData = treemap(root);  
 *addInfoData*(treeData);  
 *// Compute the new tree layout.* nodes = treeData.descendants(),  
 **links** = treeData.descendants().slice(1);  
 *// Normalize for fixed-depth.* nodes.forEach(**function** (d) {  
 d.**y** = d.**depth** \* 180  
 });  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Nodes section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the nodes...* **var** node = svg.*selectAll*(**'g.node'**)  
 .data(nodes, **function** (d) {  
 **return** d.**id** || (d.**id** = ++i);  
 });  
 *// Enter any new modes at the parent's previous position.* **var** nodeEnter = node.*enter*().append(**'g'**)  
 .attr(**'class'**, **'node'**)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y0** + **","** + source.**x0** + **")"**;  
 })  
 .on(**'click'**, *click*);  
 *// Add Circle for the nodes* nodeEnter.append(**'circle'**)  
 .on(**"mouseover"**, *mouseover*)  
 .on(**"mousemove"**, **function** (d) {  
 *mousemove*(d);  
 })  
 .on(**"mouseout"**, *mouseout*)  
 .attr(**'class'**, **'node'**)  
 .attr(**'r'**, 1e-6)  
 .*style*(**"fill"**, **function** (d) {  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#fff"**;  
 });  
 *// Add labels for the nodes* nodeEnter.append(**'text'**)  
 .attr(**"dy"**, **".35em"**)  
 .attr(**"x"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? -13 : 13;  
 })  
 .attr(**"text-anchor"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? **"end"** : **"start"**;  
 })  
 .**text**(**function** (d) {  
 **return** d.**data**.**name**;  
 });  
 *// UPDATE* **var** nodeUpdate = nodeEnter.*merge*(node);  
 *// Transition to the proper position for the node* nodeUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + d.**y** + **","** + d.**x** + **")"**;  
 });  
 *// Update the node attributes and style* nodeUpdate.select(**'circle.node'**)  
 .attr(**'r'**, 10)  
 .*style*(**"fill"**, **function** (d) {  
  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#ffa500"**;  
 })  
 .attr(**'cursor'**, **'pointer'**);  
 *// Remove any exiting nodes* **var** nodeExit = node.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y** + **","** + source.**x** + **")"**;  
 })  
 .remove();  
 *// On exit reduce the node circles size to 0* nodeExit.select(**'circle'**)  
 .attr(**'r'**, 1e-6);  
 *// On exit reduce the opacity of text labels* nodeExit.select(**'text'**)  
 .*style*(**'fill-opacity'**, 1e-6);  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* links section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the links...* **var** link = svg.*selectAll*(**'path.link'**)  
 .data(**links**, **function** (d) {  
 **return** d.**id**;  
 });  
 *// Enter any new links at the parent's previous position.* **var** linkEnter = link.*enter*().*insert*(**'path'**, **"g"**)  
 .attr(**"class"**, **"link"**)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x0**, **y**: source.**y0**}  
 **return** *diagonal*(o, o)  
 });  
 *// UPDATE* **var** linkUpdate = linkEnter.*merge*(link);  
 *// Transition back to the parent element position* linkUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **return** *diagonal*(d, d.**parent**)  
 });  
 *// Remove any exiting links* **var** linkExit = link.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x**, **y**: source.**y**}  
 **return** *diagonal*(o, o)  
 })  
 .remove();  
 *// Store the old positions for transition.* nodes.forEach(**function** (d) {  
 d.**x0** = d.**x**;  
 d.**y0** = d.**y**;  
 });  
  
 *// Creates a curved (diagonal) path from parent to the child nodes* **function** *diagonal*(s, d) {  
 *path* = **`M** ${s.**y**} ${s.**x**} **C** ${(s.**y** + d.**y**) / 2} ${s.**x**}**,** ${(s.**y** + d.**y**) / 2} ${d.**x**}**,** ${d.**y**} ${d.**x**}**`  
 return path** }  
  
 **function** *mouseover*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1);  
 }  
  
 **function** *mousemove*(d) {  
 *// console.log("d data",d);* **var** toolttiptext;  
 *// calcUniversityperformance(treeData);* **if** (d.**depth** == 0) {  
 toolttiptext = *getUniversity*(d, **"html"**);  
 } **else** {  
 **var** perdata = *getPerformance*(d);  
 **if** (perdata.**CountofUOA** > 1) {  
 toolttiptext = **"<strong>UOA No:\*: </strong><span class='details'>"** + perdata.**CountofUOA** +  
 **"<br/></span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 } **else** {  
 toolttiptext = **"</span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 }  
  
  
 }  
  
 div.**html**(toolttiptext)  
 .*style*(**"left"**, (**d3**.**event**.**pageX** ) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**);  
 }  
  
 **function** *mouseout*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1e-6);  
 }  
  
 *// Toggle children on click.* **function** *click*(d) {  
  
 **if** (d.**children**) {  
 d.**\_children** = d.**children**;  
 d.**children** = **null**;  
  
 *// div.transition()  
 // .duration(500)  
 // .style("opacity", 0);  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", 0);* } **else** {  
  
 *// ref14data.forEach(findlastnode);  
 // function findlastnode(dr){  
 //  
 // if (dr["Unit of assessment name"]==d.data.name){  
 // console.log("d.data.name",dr["Unit of assessment name"]);  
 //  
 // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span>";  
 // // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \"" + d.WEBSITE\_URL + "\" >" + d.WEBSITE\_URL + "</a></span>";  
 //  
 // tooltip.html(t)  
 // .style("left", (d3.event.pageX) + "px")  
 // .style("top", (d3.event.pageY) + "px");  
 //  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", .7);  
 //  
 // }}* d.**children** = d.**\_children**;  
 d.**\_children** = **null**;  
 }  
 *update*(d);  
 }  
  
  
 }  
  
 **function** *addInfoData*(objdata) {  
 **console**.log(**"treeData.data"**, objdata);  
 **var** textinfo = *getUniversity*(objdata, **"html"**);  
 aside.**html**(textinfo);  
  
 }  
  
 **function** *getUniversity*(un, output) {  
 **var** result;  
 **console**.log(**"d.data"**, un);  
 **learningProviders**.forEach(**function** (rec) {  
  
 **if** (rec.UKPRN == un.**data**.**name**) {  
 **if** (output == **"html"**) {  
 result = **"<strong>Provider: </strong><span class='details'>"** + rec.PROVIDER\_NAME +  
 **"<br/></span> <strong>UOA count: </strong> <span class='details'>"** + *getPerformance*(un).**CountofUOA** +  
 **"<br/></span> <strong>GROUPS: </strong> <span class='details'>"** + rec.GROUPS +  
 **"<br/></span> <strong>BUILDING\_NAME\_NUMBER: </strong> <span class='details'>"** + rec.BUILDING\_NAME\_NUMBER +  
 **"<br/></span> <strong>LOCALITY: </strong> <span class='details'>"** + rec.LOCALITY +  
 **"<br/></span> <strong>STREET\_NAME: </strong> <span class='details'>"** + rec.STREET\_NAME +  
 **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + rec.TOWN +  
 **"<br/></span> <strong>POSTCODE: </strong> <span class='details'>"** + rec.POSTCODE +  
 **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + rec.WEBSITE\_URL + **"\" >"** + rec.WEBSITE\_URL +  
 **"</a></span>"**;  
 } **else** {  
 result = [rec.PROVIDER\_NAME, calcUniversityperformance(), rec.GROUPS, rec.BUILDING\_NAME\_NUMBER, rec.LOCALITY, rec.STREET\_NAME, rec.TOWN, rec.POSTCODE, rec.WEBSITE\_URL]  
 }  
  
  
 }  
 **return** result;  
 });  
 *// console.log("result data",result);* **return** result;  
 }  
  
 **function** *getPerformance*(obj) {  
 **var** Performance = {**CountofUOA**: 0, **sumOfOne**: 0, **sumOftwo**: 0, **sumOftree**: 0, **sumOffour**: 0, **sumFTB**: 0};  
  
 **function** *getLeafNodes*(leafNodes, obj) {  
  
 **if** (obj.**children**) {  
  
 obj.**children**.forEach(**function** (child) {  
 *getLeafNodes*(leafNodes, child)  
  
 });  
 } **else** {  
  
 leafNodes.push(obj);  
 }  
 }  
  
 **var** leafNodes = [];  
 *getLeafNodes*(leafNodes, obj.**data**);  
 Performance.**CountofUOA** = leafNodes.**length**;  
 leafNodes.forEach(**function** (child) {  
 **var** childobject = child.**value**[0];  
 *// console.log("childobject[\"1\*\"]",childobject["1\*"])* Performance.**sumOfOne** += ***Math***.round(parseFloat(childobject[**"1\*"**]));  
 Performance.**sumOftwo** += ***Math***.round(parseFloat(childobject[**"2\*"**]));  
 Performance.**sumOftree** += ***Math***.round(parseFloat(childobject[**"3\*"**]));  
 Performance.**sumOffour** += ***Math***.round(parseFloat(childobject[**"4\*"**]));  
 Performance.**sumFTB** += ***Math***.round(parseFloat(childobject[**"FTE Category A staff submitted"**])) / Performance.**CountofUOA**;  
 });  
  
 **return** Performance;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** treechartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
*//=================== industrial Module Charts =========================***function** *barchart\_ic*(targetDOMelement) {  
  
 *//Delare the main object that will be returned to caller* **var** barchartObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* barchartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.render = **function** (callbackFunction) {  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 barchartObject.loadAndRenderDataset = **function** (data, pdata) {  
 dataset = data;  
 InstitutionPointData = pdata;  
 *// console.log("dataset ",dataset);* barData = **d3**.*nest*()  
 .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
  
 .rollup(**function** (v) {  
 **return** {  
 **count**: v.**length**,  
 **total**: **d3**.*sum*(v, **function** (d) {  
 **return** parseFloat(d[**"4\*"**]);  
 }),  
 **max**: **d3**.**max**(v, **function** (d) {  
 **return** parseFloat(d[**"4\*"**]);  
 }),  
 **min**: **d3**.**min**(v, **function** (d) {  
 **return** parseFloat(d[**"4\*"**]);  
 }),  
 **avg**: **d3**.*mean*(v, **function** (d) {  
 **return** parseFloat(d[**"4\*"**])  
 })  
 };  
  
 })  
 .entries(dataset);  
 **console**.log(**"barData ic "**, barData);  
 *// console.log("l1Data values: ",barData.values.length);  
 layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.update = **function** (val) {  
 **console**.log(**"dataset "**, dataset);  
 **console**.log(**"barData "**, barData);  
 barData = **d3**.*nest*()  
 .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
 *// .key(function(d) { return d["Institution code (UKPRN)"]; })  
 // .rollup(function(v) {  
 // console.log("v",v);  
 // return v.length; })* .rollup(**function** (v) {  
 **return** {  
 **count**: v.**length**,  
 **total**: **d3**.*sum*(v, **function** (d) {  
 **return** parseFloat(d[**"3\*"**]);  
 }),  
 **max**: **d3**.**max**(v, **function** (d) {  
 **return** parseFloat(d[**"3\*"**]);  
 }),  
 **min**: **d3**.**min**(v, **function** (d) {  
 **return** parseFloat(d[**"3\*"**]);  
 }),  
 **avg**: **d3**.*mean*(v, **function** (d) {  
 **return** parseFloat(d[**"3\*"**])  
 })  
 };  
  
 })  
 .entries(dataset);  
 **console**.log(**"barData ic val"**, barData);  
 *// console.log("l1Data values: ",barData.values.length);  
 layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
 barchartObject.width = **function** (w) {  
 svgWidth = w;  
 grp.attr(**"transform"**, **"translate("** + [w / 2, svgHeight / 2] + **")"**)  
  
 *layoutAndRender*();  
 **return** barchartObject;  
 }  
  
 *//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas  
  
 // var tooltipclass = "toolTip"* **var** dataset = [];  
 **var** barData = [];  
 **var** InstitutionPointData = [];  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
  
 **var** x;  
 **var** y;  
  
 **var** xAxis;  
 **var** yAxis;  
  
  
 **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom);  
  
 **var** g = svg.append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
 svg.append(**"text"**)  
 .attr(**"x"**, (svgWidth / 2))  
 .attr(**"y"**, (margintop / 6))  
 .attr(**'class'**, **'bartitleselected'**)  
 .**text**(**"Selected UOA to filter the data"**);  
  
  
 *// var tip = d3.tip();* **var** tooltip = **d3**.select(**"body"**).append(**"div"**).attr(**"class"**, **"uitoolTip"**);  
  
  
 *//Declare and append group that we will use in the barchart within the svg  
  
  
 //=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
 **console**.log(**"barData "**, barData);  
 *//Taken and addapted from https://github.com/d3/d3-shape/blob/master/README.md#pie* x = **d3**.*scaleLinear*().range([0, svgWidth]);  
 y = **d3**.*scaleBand*().range([svgHeight, 0]);  
  
 xAxis = **d3**.*axisBottom*(x).ticks(5).tickFormat(**function** (d) {  
 **return** parseInt(d / 1);  
 }).tickSizeInner([-svgHeight]);  
 yAxis = **d3**.*axisLeft*(y);  
  
 barData.sort(**function** (a) {  
 **return** a.**key**;  
 });  
  
 svg.attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom);  
 g.attr(**"transform"**, **"translate("** + marginleft + **","** + margintop + **")"**);  
  
 x.domain([0, **d3**.**max**(barData, **function** (d) {  
 **return** d.**value**.**avg**;  
 })]);  
  
 y.domain(barData.map(**function** (d) {  
 **return** barData.indexOf(d);  
 }))  
 .**padding**(0.1);  
  
 g.append(**"g"**)  
 .attr(**"class"**, **"x axis"**)  
 .attr(**"transform"**, **"translate(0,"** + svgHeight + **")"**)  
 .call(xAxis);  
  
 g.append(**"g"**).attr(**"class"**, **"y axis"**)  
 .call(yAxis);  
  
 g.*selectAll*(**".bar"**)  
 .data(barData)  
 .*enter*()  
 .append(**"rect"**)  
 .attr(**"class"**, **"bar"**)  
 .attr(**"x"**, 0)  
 .attr(**"height"**, y.bandwidth())  
 .attr(**"y"**, **function** (d) {  
 **return** y(barData.indexOf(d));  
 })  
 .attr(**"width"**, **function** (d) {  
 **return** x(d.**value**.**avg**);  
 })  
 .on(**"click"**, **function** (d) {  
  
 **console**.log(**"this"**, **this**.**classList**.contains(**"barselected"**));  
 **if** (**this**.**classList**.contains(**"barselected"**)) {  
  
 **d3**.*selectAll*(**".barselected"**).attr(**"class"**, **"bar"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(**"Selected UOA to filter the data"**);  
  
 *// careerModulel2.Update(InstitutionPointData);* } **else** {  
  
 **d3**.*selectAll*(**".barselected"**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"bar"**);  
  
 **d3**.select(**this**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"barselected"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(d.**key**);  
  
 *// updateotherchart(d);* }  
  
 })  
 .on(**"mouseover"**, **function** (d) {  
 tooltip  
 .*style*(**"left"**, **d3**.**event**.**pageX** - 50 + **"px"**)  
 .*style*(**"top"**, **d3**.**event**.**pageY** + 10 + **"px"**)  
 .*style*(**"display"**, **"inline-block"**)  
 .**html**(barData.indexOf(d) + **":"** + (d.**key**)  
 + **"</br>"** + **"Total: "** + (d.**value**.**total**)  
 + **"</br>"** + **"Count: "** + (d.**value**.**count**)  
 *// + "<br>" + "Max : " + (d.value.max)  
 // + "<br>" + "Min : " + (d.value.min)* + **"</br>"** + **"Avg :"** + (d.**value**.**avg**));  
 })  
 .on(**"mouseout"**, **function** (d) {  
 tooltip.*style*(**"display"**, **"none"**);  
 })  
 .on(**'click'**, **function** (d) {  
 *// console.log("this", this.classList.contains("barselected"));* **if** (**this**.**classList**.contains(**"barselected"**)) {  
  
 **d3**.*selectAll*(**".barselected"**).attr(**"class"**, **"bar"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(**"Selected UOA to filter the data"**);  
  
 *// careerModulel2.Update(InstitutionPointData);* } **else** {  
  
 **d3**.*selectAll*(**".barselected"**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"bar"**);  
  
 **d3**.select(**this**)  
 .*transition*()  
 .**duration**(200)  
 .attr(**"class"**, **"barselected"**);  
 **d3**.*selectAll*(**".bartitleselected"**).**text**(d.**key**);  
  
 *updateotherchart*(d);  
 }  
  
 });  
  
  
 *// svg.append("g")  
 // .attr("class", "x axis")  
 // .attr("transform", "translate(0," + (svgHeight-5) + ")")  
 // .call(xAxis)  
 // .call(tip);  
 //  
 // svg.append("g")  
 // .attr("class", "y axis")  
 // .call(yAxis);  
 //  
 // svg.append("text")  
 // .attr("transform",  
 // "translate(" + (svgWidth/2) + " ," + (svgHeight+ marginbottom) + ")")  
 // .style("text-anchor", "middle")  
 // .text("UOA");  
 //  
 // svg.append("text")  
 // .attr("class", "label")  
 // .attr("y", -(marginleft/2))  
 // .attr("x", -(margintop/2)+2 )  
 // .style("text-anchor", "end")  
 // .text("Count");  
 //  
 // svg.selectAll(".bar")  
 // .data(barData)  
 // .enter()  
 // .append("rect")  
 // .attr("class", "bar")  
 // // .style("fill", function(d) {return color(barData.indexOf(d)+1);})  
 // .attr("x", function(d) { return x(d.key); })  
 // .attr("width", x.bandwidth())  
 // .attr("y", function(d) { return y(d.value)-5; })  
 // .attr("height", function(d) { return svgHeight - y(d.value); })  
 // .on('mouseover', tip.show)  
 // .on('mouseout', tip.hide)  
 // .on('click', function (d) {  
 // // console.log("this", this.classList.contains("barselected"));  
 // if(this.classList.contains("barselected")){  
 //  
 // d3.selectAll(".barselected").attr("class", "bar");  
 // d3.selectAll(".bartitleselected").text("Selected UOA to filter the data");  
 //  
 // // careerModulel2.Update(InstitutionPointData);  
 //  
 //  
 // }else{  
 //  
 // d3.selectAll(".barselected")  
 // .transition()  
 // .duration(200)  
 // .attr("class", "bar");  
 //  
 // d3.select(this)  
 // .transition()  
 // .duration(200)  
 // .attr("class","barselected");  
 // d3.selectAll(".bartitleselected").text( d.key);  
 //  
 // updateotherchart(d);  
 // }  
 //  
 // } );  
 // svg.selectAll(".barlabel")  
 // .data(barData)  
 // .enter()  
 // .append("text")  
 // .attr("class","barlabel")  
 // .attr("x", (function(d) { return x(d.key)+ x.bandwidth()/2; } ))  
 // .attr("y", function(d) { return y(d.value)+50; })  
 // .attr("dy", "0.001em")  
 // .text(function(d) { return d.value; });* }  
  
 **function** *updateotherchart*(parm) {  
 **console**.log(**"parm"**, parm);  
 *updatemapchart*(parm);  
 ***ic3***.clear();  
  
 **function** *updatemapchart*(parm) {  
  
 **var** selected = [];  
  
 **var** filtereddata = dataset.filter(**function** (d) {  
 **return** d[**"Unit of assessment name"**] == parm.**key**;  
 });  
 **console**.log(**"filtereddata"**, filtereddata);  
  
 filtereddata.forEach(*processUniversity*);  
  
 **function** *processUniversity*(dr) {  
 InstitutionPointData.forEach(**function** (rec) {  
  
 **if** (rec.UKPRN == dr[**"Institution code (UKPRN)"**]) {  
 *// console.log("rec.UKPRN: ",rec.UKPRN,"row.key : ",dr.key);* **var** str = **'{ "UKPRN":'** + rec.UKPRN + **', "PROVIDER\_NAME": "'** + rec.PROVIDER\_NAME + **'", "Main\_panel": "'** + dr[**"Main panel"**] + **'", "LATITUDE":"'** + rec.LATITUDE + **'", "LONGITUDE":"'** + rec.LONGITUDE + **'", "TOWN":"'** + rec.TOWN + **'" , "WEBSITE\_URL":"'** + rec.WEBSITE\_URL + **'"}'**;  
  
 **var** result = ***JSON***.parse(str);  
 selected.push(result);  
 }  
 })  
 }  
  
 *// svg.selectAll(".d3-tip").style("fill","red");* **console**.log(**"selected Data"**, selected);  
 ***ic2***.Update(selected);  
  
 }  
  
  
  
 }  
  
 **function** *type*(d) {  
 *// d.values.length = +d.values.length;* **return** d;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** barchartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *ukMap\_ic*(targetDOMelement, jsonMapData) {  
 *//Delare the main object that will be returned to caller* **var** mapObject = {};  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* mapObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.render = **function** (callbackFunction) {  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
 mapObject.loadAndRenderDataset = **function** (jsonPointData) {  
 InstitutionPointData = jsonPointData;  
 *layoutAndRender*();  
  
 **return** mapObject;  
 }  
 mapObject.Update = **function** (Data) {  
 InstitutionPointData = Data;  
 *// console.log("InstitutionPointData",InstitutionPointData)  
 // layoutAndRender();  
 GUP\_towns*(svg, InstitutionPointData);  
 *// var circle= d3.selectAll("circle");  
 // repeat();  
 // function repeat() {  
 //  
 // circle  
 // .attr("opacity", .7)  
 // .transition() // apply a transition  
 // .duration(4000) // apply it over 2000 milliseconds  
 // .attr("fill", "red") // move the circle to 920 on the x axis  
 // .attr("r", "8px")  
 // .transition() // apply a transition  
 // .duration(4000) // apply it over 2000 milliseconds  
 // .attr("fill", "green") // return the circle to 40 on the x axis  
 // .attr("r", "4px")  
 // .on("end", repeat); // when the transition finishes start again  
 // };* **return** mapObject;  
 }  
 mapObject.height = **function** (h) {  
 svgHeight = h;  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.width = **function** (w) {  
 svgWidth = w;  
 projection.translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
 mapObject.scale = **function** (scale) {  
 projectionscale = scale;  
 projection.scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 2]);  
  
 *layoutAndRender*();  
 **return** mapObject;  
 }  
  
*//=================== PRIVATE VARIABLES ====================================  
 //Width and height of svg canvas  
  
 // var tooltipclass = "toolTip"* **var** InstitutionPointData = [];  
  
 **var** svgWidth = 400, svgHeight = 400, margintop = 20, marginright = 20, marginbottom = 20, marginleft = 20;  
 **var** projectionscale = 2500, pointRadius = 2, active = **d3**.select(**null**);  
  
 **var** projection = **d3**.*geoAlbers*()  
 .center([0, 55.4])  
 .rotate([4.4, 0])  
 .parallels([50, 60])  
 .scale(projectionscale)  
 .translate([svgWidth / 2, svgHeight / 3]);  
  
 *//Define path generator (takes projected 2D geometry and formats for SVG)* **var** path = **d3**.*geoPath*()  
 .*projection*(projection)  
 .pointRadius(pointRadius);  
  
 *//Create SVG* **var** svg = **d3**.select(targetDOMelement)  
 .append(**"svg"**)  
 .attr(**"width"**, svgWidth + marginleft + marginright)  
 .attr(**"height"**, svgHeight + margintop + marginbottom)  
 .append(**"g"**)  
 .attr(**"transform"**, **"translate("** + marginleft + **","** + margintop \* 3 + **")"**);  
  
 svg.append(**"rect"**)  
 .attr(**"width"**, svgWidth)  
 .attr(**"height"**, svgHeight)  
 .*style*(**"fill"**, **"none"**)  
 .*style*(**"pointer-events"**, **"all"**)  
 .call(**d3**.*zoom*()  
 .scaleExtent([1 / 2, 4])  
 .on(**"zoom"**, *zoomed*));  
  
 **function** *zoomed*() {  
 svg.attr(**"transform"**, **d3**.**event**.**transform**);  
 }  
  
 *// .call(d3.zoom().scaleExtent([1, 8]).on("zoom", function () {  
 // svg.style("stroke-width", 1.5 / d3.event.transform.k + "px");  
 // svg.attr("transform", d3.event.transform)  
 // }))  
 // .on("click", function(){  
 // if (active.node() === this) return reset();  
 // active.classed("active", false);  
 // active = d3.select(this).classed("active", true);  
 //  
 // // var bounds = path.bounds(d),  
 // // dx = bounds[1][0] - bounds[0][0],  
 // // dy = bounds[1][1] - bounds[0][1],  
 // // x = (bounds[0][0] + bounds[1][0]) / 2,  
 // // y = (bounds[0][1] + bounds[1][1]) / 2,  
 // // scale = Math.max(1, Math.min(8, 0.9 / Math.max(dx / svgWidth, dy / svgHeight))),  
 // // translate = [svgWidth / 2 - scale \* x, svgHeight / 2 - scale \* y];  
 // //  
 // // svg.transition()  
 // // .duration(750)  
 // // .call( zoom.transform, d3.zoomIdentity.translate(translate[0],translate[1]).scale(scale) ); // updated for d3 v4  
 //  
 // });  
  
 /\*svg.append("text")  
 .attr("x", (svgWidth / 2))  
 .attr("y", 0 - (margintop / 2))  
 .attr("text-anchor", "middle")  
 .style("font-size", "16px")  
 .style("text-decoration", "underline")  
 .text("Institution Location ");\*/* **var** tooltip = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"tooltip"**)  
 .*style*(**"opacity"**, 0)  
 .*style*(**"width"**, 600);  
  
 *//=================== PRIVATE FUNCTIONS ====================================* **function** *layoutAndRender*() {  
  
  
 *//Read in JSON file of UK map and do all the D3 stuff:* **d3**.**json**(jsonMapData, **function** (error, uk) {  
 **if** (error) **return console**.error(error);  
 **console**.log(uk);  
  
  
 *// Displaying Polygons* **var** subunits = **topojson**.*feature*(uk, uk.objects.subunits).features;  
 *// var places = topojson.feature(uk, uk.objects.places).features;  
 GUP\_subunits*(svg, subunits, uk);  
 *GUP\_towns*(svg, InstitutionPointData);  
  
  
 });  
  
  
 *//* }  
  
 **function** *GUP\_subunits*(svg, subunits, uk) {  
  
 svg.append(**"path"**)  
 .*datum*(subunits)  
 .attr(**"d"**, path);  
 **console**.log(**"svg"**, svg);  
  
 *// Styling Polygons* svg.*selectAll*(**".subunit"**)  
 .data(subunits)  
 .*enter*()  
 .append(**"path"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit "** + d.**id**;  
 })  
 .attr(**"d"**, path);  
  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 **return** a !== b && a.**id** !== **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary"**);  
  
 *// Displaying Boundaries* svg.append(**"path"**)  
 .*datum*(**topojson**.*mesh*(uk, uk.objects.subunits, **function** (a, b) {  
 *// console.log("a id",a.id);* **return** a === b && a.**id** === **"IRL"**;  
 }))  
 .attr(**"d"**, path)  
 .attr(**"class"**, **"subunit-boundary IRL"**);  
  
 *// Country Labels* svg.*selectAll*(**".subunit-label"**)  
 .data(subunits)  
 .*enter*().append(**"text"**)  
 .attr(**"class"**, **function** (d) {  
 **return "subunit-label "** + d.**id**;  
 })  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + path.centroid(d) + **")"**;  
 })  
 .attr(**"dy"**, **".35em"**)  
 .**text**(**function** (d) {  
 **return** d.**properties**.**name**;  
 });  
  
  
 }  
  
 **function** *GUP\_towns*(svg, places) {  
  
 **var** circle = svg.*selectAll*(**"circle"**);  
 *// console.log("InstitutionPointData",InstitutionPointData);  
 // tip = d3.tip()  
 // .attr('class', 'd3-tip')  
 // .offset([-10, 0])  
 // .html(function(d) {  
 // return "<strong>" + d.PROVIDER\_NAME + " :</strong> >";  
 // });* circle  
 .remove().*exit*()  
 .data(InstitutionPointData)  
 .*enter*()  
 .append(**"circle"**)  
 .attr(**"cx"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[0];  
 })  
 .attr(**"cy"**, **function** (d) {  
 **return** projection([d.LONGITUDE, d.LATITUDE])[1];  
 })  
 .attr(**"r"**, **"3px"**)  
 .attr(**"fill"**, **"green"**)  
 .attr(**"opacity"**, .7)  
 .on(**"mouseover"**, **function** (d) {  
 *// var tip = "<h3>" + d.PROVIDER\_NAME + "</h3>";* **var** t = **"<strong>Provider: </strong><span class='details'>"** + d.PROVIDER\_NAME + **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + d.TOWN + **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + d.WEBSITE\_URL + **"\" >"** + d.WEBSITE\_URL + **"</a></span>"**;  
  
 tooltip  
 .**html**(t)  
 .*style*(**"left"**, (**d3**.**event**.**pageX**) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**)  
 .*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, .7);  
  
  
 })  
 .on(**"mouseout"**, **function** (d) {  
 tooltip.*transition*()  
 .**duration**(500)  
 .*style*(**"opacity"**, 0);  
 })  
 .on(**"click"**, **function** (d) {  
 *// console.log("careerModulel3",careerModulel3);* ***ic3***.loadAndRenderDataset(*preperTreeData*(d));  
 *// careerModulel3.data(preperTreeData(d));  
 // d3.select('#careerModulel3')  
 // .call(careerModulel3);  
 // careerModulel3.exit().remove();* });  
  
  
 *// console.log("InstitutionPointData",InstitutionPointData[0].LATITUDE);  
  
 // svg.selectAll(".place-label")  
 // .data(InstitutionPointData)  
 // .enter().append("text")  
 // .attr("class", "place-label")  
 // .attr("transform", function(d) { return "translate(" + projection([d.LONGITUDE,d.LATITUDE]) + ")"; })  
 // .attr("x", function(d) { return [d.LONGITUDE,d.LATITUDE][0] > -1 ? 6 : -6; })  
 // .attr("dy", ".35em")  
 // .style("text-anchor", function(d) { return [d.LONGITUDE,d.LATITUDE][0] > -1 ? "start" : "end"; })  
 // .text(function(d) { return InstitutionPointData.indexOf(d)+1; });* }  
  
  
 **function** *preperTreeData*(selectedpoint) {  
  
 **console**.log(**"selectedpoint"**, selectedpoint);  
 **DataFiltered** = **ref14data**.filter(**function** (d) {  
  
 **if** (d[**"Institution code (UKPRN)"**] == selectedpoint.UKPRN) {  
 *// console.log("ref14data d val",d["Institution code (UKPRN)"]);  
 // console.log("selectedpoint",selectedpoint.UKPRN);* **return** d  
 }  
 });  
 **console**.log(**"DataFiltered"**, **DataFiltered**);  
  
 **var** treeData = **d3**.*nest*()  
 .**key**(**function** (d) {  
 **return** d[**"Main panel"**];  
 })  
 .**key**(**function** (d) {  
 **return** d[**"Profile"**];  
 })  
 *// .key(function(d) { return d["FTE Category A staff submitted"]; })* .**key**(**function** (d) {  
 **return** d[**"Unit of assessment name"**];  
 })  
  
 .rollup(**function** (v) {  
 **return** v  
 })  
 .entries(**DataFiltered**);  
  
 *// console.log("treeData",treeData);* **var** refJSON = ***JSON***.stringify(treeData);  
 refJSON = refJSON.replace(**new** RegExp(**'values'**, **'g'**), **'children'**);  
 refJSON = refJSON.replace(**new** RegExp(**'key'**, **'g'**), **'name'**);  
  
 **var** jsonObj = ***JSON***.parse(refJSON);  
  
 *// console.log("jsonObj",jsonObj);* **var** augRefJSON = {**"name"**: selectedpoint.UKPRN, **"children"**: jsonObj};  
  
 *// console.log("augRefJSON",augRefJSON);* **return** augRefJSON;  
 }  
  
 **function** *reset*() {  
 active.*classed*(**"active"**, **false**);  
 active = **d3**.select(**null**);  
  
 svg.*transition*()  
 .**duration**(750)  
 *// .call( zoom.transform, d3.zoomIdentity.translate(0, 0).scale(1) ); // not in d3 v4* .call(*zoom*.transform, **d3**.zoomIdentity); *// updated for d3 v4* }  
  
  
 *//================== IMPORTANT do not delete ==================================* **return** mapObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
**function** *treechart\_ic*(targetDOMelement) {  
*//Delare the main object that will be returned to caller* **var** treechartObject = {};  
 *//=================== PRIVATE VARIABLES ====================================  
// Set the dimensions and margins of the diagram* **var** margin = {**top**: 20, **right**: 90, **bottom**: 30, **left**: 90},  
 width = 960 - margin.**left** - margin.**right**,  
 height = 500 - margin.**top** - margin.**bottom**;  
 **var** i = 0,  
 duration = 750,  
 root;  
 **var** treeData = [];  
 **var** nodes;  
 *// var div = d3.select(targetDOMelement).append("div")  
 // .attr("class", "tooltip")  
 // .style("opacity", 0);  
  
 // var tooltip= d3.select("body").append("div")  
 // .attr("class", "Treetooltip")  
 // .style("opacity", 0)  
 // .style("width", 600);* **var** div = **d3**.select(**"body"**).append(**"div"**)  
 .attr(**"class"**, **"Treetooltip"**)  
 .*style*(**"opacity"**, 1e-6);  
  
 **var** aside = **d3**.select(targetDOMelement).append(**"aside"**)  
 .attr(**"class"**, **"selecteinfo"**);  
  
*// declares a tree layout and assigns the size* **var** treemap = **d3**.*tree*().size([height, width]);  
  
*// append the svg object to the body of the page  
// appends a 'group' element to 'svg'  
// moves the 'group' element to the top left margin* **var** svg = **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** + **")"**);  
  
 *//=================== PUBLIC FUNCTIONS =========================  
 //* treechartObject.overrideMouseOverFunction = **function** (callbackFunction) {  
 **mouseOverFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.overrideMouseOutFunction = **function** (callbackFunction) {  
 **mouseOutFunction** = callbackFunction;  
 *// layoutAndRender();* **return** treechartObject;  
 }  
  
 treechartObject.clear = **function** () {  
  
  
 svg.*style*(**"display"**, **"none"**);  
 aside.*style*(**"display"**, **"none"**);  
  
 **return** treechartObject;  
 }  
 treechartObject.loadAndRenderDataset = **function** (data) {  
 *// console.log("treeData inside",treeData);* svg.*style*(**"display"**, **"block"**);  
 aside.*style*(**"display"**, **"block"**);  
  
 treeData = data;  
 *// Assigns parent, children, height, depth* root = **d3**.*hierarchy*(treeData, **function** (d) {  
 **return** d.**children**;  
 });  
 root.**x0** = height / 2;  
 root.**y0** = 0;  
 *// Collapse after the second level* root.**children**.forEach(*collapse*);  
 *update*(root);  
 *addInfoData*();  
  
 *// Collapse the node and all it's children* **function** *collapse*(d) {  
 **if** (d.**children**) {  
 d.**\_children** = d.**children** d.**\_children**.forEach(*collapse*)  
 d.**children** = **null** }  
 }  
  
 **return** treechartObject;  
 }  
  
 **function** *update*(source) {  
 *// Assigns the x and y position for the nodes* **var** treeData = treemap(root);  
 *addInfoData*(treeData);  
 *// Compute the new tree layout.* nodes = treeData.descendants(),  
 **links** = treeData.descendants().slice(1);  
 *// Normalize for fixed-depth.* nodes.forEach(**function** (d) {  
 d.**y** = d.**depth** \* 180  
 });  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Nodes section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the nodes...* **var** node = svg.*selectAll*(**'g.node'**)  
 .data(nodes, **function** (d) {  
 **return** d.**id** || (d.**id** = ++i);  
 });  
 *// Enter any new modes at the parent's previous position.* **var** nodeEnter = node.*enter*().append(**'g'**)  
 .attr(**'class'**, **'node'**)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y0** + **","** + source.**x0** + **")"**;  
 })  
 .on(**'click'**, *click*);  
 *// Add Circle for the nodes* nodeEnter.append(**'circle'**)  
 .on(**"mouseover"**, *mouseover*)  
 .on(**"mousemove"**, **function** (d) {  
 *mousemove*(d);  
 })  
 .on(**"mouseout"**, *mouseout*)  
 .attr(**'class'**, **'node'**)  
 .attr(**'r'**, 1e-6)  
 .*style*(**"fill"**, **function** (d) {  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#fff"**;  
 });  
 *// Add labels for the nodes* nodeEnter.append(**'text'**)  
 .attr(**"dy"**, **".35em"**)  
 .attr(**"x"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? -13 : 13;  
 })  
 .attr(**"text-anchor"**, **function** (d) {  
 **return** d.**children** || d.**\_children** ? **"end"** : **"start"**;  
 })  
 .**text**(**function** (d) {  
 **return** d.**data**.**name**;  
 });  
 *// UPDATE* **var** nodeUpdate = nodeEnter.*merge*(node);  
 *// Transition to the proper position for the node* nodeUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + d.**y** + **","** + d.**x** + **")"**;  
 });  
 *// Update the node attributes and style* nodeUpdate.select(**'circle.node'**)  
 .attr(**'r'**, 10)  
 .*style*(**"fill"**, **function** (d) {  
  
 **return** d.**\_children** ? **"lightsteelblue"** : **"#ffa500"**;  
 })  
 .attr(**'cursor'**, **'pointer'**);  
 *// Remove any exiting nodes* **var** nodeExit = node.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**"transform"**, **function** (d) {  
 **return "translate("** + source.**y** + **","** + source.**x** + **")"**;  
 })  
 .remove();  
 *// On exit reduce the node circles size to 0* nodeExit.select(**'circle'**)  
 .attr(**'r'**, 1e-6);  
 *// On exit reduce the opacity of text labels* nodeExit.select(**'text'**)  
 .*style*(**'fill-opacity'**, 1e-6);  
 *// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* links section \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Update the links...* **var** link = svg.*selectAll*(**'path.link'**)  
 .data(**links**, **function** (d) {  
 **return** d.**id**;  
 });  
 *// Enter any new links at the parent's previous position.* **var** linkEnter = link.*enter*().*insert*(**'path'**, **"g"**)  
 .attr(**"class"**, **"link"**)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x0**, **y**: source.**y0**}  
 **return** *diagonal*(o, o)  
 });  
 *// UPDATE* **var** linkUpdate = linkEnter.*merge*(link);  
 *// Transition back to the parent element position* linkUpdate.*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **return** *diagonal*(d, d.**parent**)  
 });  
 *// Remove any exiting links* **var** linkExit = link.*exit*().*transition*()  
 .**duration**(duration)  
 .attr(**'d'**, **function** (d) {  
 **var** o = {**x**: source.**x**, **y**: source.**y**}  
 **return** *diagonal*(o, o)  
 })  
 .remove();  
 *// Store the old positions for transition.* nodes.forEach(**function** (d) {  
 d.**x0** = d.**x**;  
 d.**y0** = d.**y**;  
 });  
  
 *// Creates a curved (diagonal) path from parent to the child nodes* **function** *diagonal*(s, d) {  
 *path* = **`M** ${s.**y**} ${s.**x**} **C** ${(s.**y** + d.**y**) / 2} ${s.**x**}**,** ${(s.**y** + d.**y**) / 2} ${d.**x**}**,** ${d.**y**} ${d.**x**}**`  
 return path** }  
  
 **function** *mouseover*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1);  
 }  
  
 **function** *mousemove*(d) {  
 *// console.log("d data",d);* **var** toolttiptext;  
 *// calcUniversityperformance(treeData);* **if** (d.**depth** == 0) {  
 toolttiptext = *getUniversity*(d, **"html"**);  
 } **else** {  
 **var** perdata = *getPerformance*(d);  
 **if** (perdata.**CountofUOA** > 1) {  
 toolttiptext = **"<strong>UOA No:\*: </strong><span class='details'>"** + perdata.**CountofUOA** +  
 **"<br/></span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 } **else** {  
 toolttiptext = **"</span> <strong>Sum of 1\*: </strong><span class='details'>"** + perdata.**sumOfOne** +  
 **"<br/></span> <strong>Sum of 2\*: </strong> <span class='details'>"** + perdata.**sumOftwo** +  
 **"<br/></span> <strong>Sum of 3\*: </strong> <span class='details'>"** + perdata.**sumOftree** +  
 **"<br/></span> <strong>Sum of 4\*: </strong> <span class='details'>"** + perdata.**sumOffour** +  
 **"<br/></span> <strong>Sum of FTE: </strong> <span class='details'>"** + perdata.**sumFTB** + **"</span>"**;  
 }  
  
  
 }  
  
 div.**html**(toolttiptext)  
 .*style*(**"left"**, (**d3**.**event**.**pageX** ) + **"px"**)  
 .*style*(**"top"**, (**d3**.**event**.**pageY**) + **"px"**);  
 }  
  
 **function** *mouseout*() {  
 div.*transition*()  
 .**duration**(300)  
 .*style*(**"opacity"**, 1e-6);  
 }  
  
 *// Toggle children on click.* **function** *click*(d) {  
  
 **if** (d.**children**) {  
 d.**\_children** = d.**children**;  
 d.**children** = **null**;  
  
 *// div.transition()  
 // .duration(500)  
 // .style("opacity", 0);  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", 0);* } **else** {  
  
 *// ref14data.forEach(findlastnode);  
 // function findlastnode(dr){  
 //  
 // if (dr["Unit of assessment name"]==d.data.name){  
 // console.log("d.data.name",dr["Unit of assessment name"]);  
 //  
 // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span>";  
 // // var t= "<strong>UOA: </strong><span class='details'>" + dr["Unit of assessment name"] + "<br/></span> <strong>FTE: </strong> <span class='details'>" + dr["FTE Category A staff submitted"] +"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \"" + d.WEBSITE\_URL + "\" >" + d.WEBSITE\_URL + "</a></span>";  
 //  
 // tooltip.html(t)  
 // .style("left", (d3.event.pageX) + "px")  
 // .style("top", (d3.event.pageY) + "px");  
 //  
 // tooltip.transition()  
 // .duration(500)  
 // .style("opacity", .7);  
 //  
 // }}* d.**children** = d.**\_children**;  
 d.**\_children** = **null**;  
 }  
 *update*(d);  
 }  
  
  
 }  
  
 **function** *addInfoData*(objdata) {  
 **console**.log(**"treeData.data"**, objdata);  
 **var** textinfo = *getUniversity*(objdata, **"html"**);  
 aside.**html**(textinfo);  
  
 }  
  
 **function** *getUniversity*(un, output) {  
 **var** result;  
 **console**.log(**"d.data"**, un);  
 **learningProviders**.forEach(**function** (rec) {  
  
 **if** (rec.UKPRN == un.**data**.**name**) {  
 **if** (output == **"html"**) {  
 result = **"<strong>Provider: </strong><span class='details'>"** + rec.PROVIDER\_NAME +  
 **"<br/></span> <strong>UOA count: </strong> <span class='details'>"** + *getPerformance*(un).**CountofUOA** +  
 **"<br/></span> <strong>GROUPS: </strong> <span class='details'>"** + rec.GROUPS +  
 **"<br/></span> <strong>BUILDING\_NAME\_NUMBER: </strong> <span class='details'>"** + rec.BUILDING\_NAME\_NUMBER +  
 **"<br/></span> <strong>LOCALITY: </strong> <span class='details'>"** + rec.LOCALITY +  
 **"<br/></span> <strong>STREET\_NAME: </strong> <span class='details'>"** + rec.STREET\_NAME +  
 **"<br/></span> <strong>TOWN: </strong> <span class='details'>"** + rec.TOWN +  
 **"<br/></span> <strong>POSTCODE: </strong> <span class='details'>"** + rec.POSTCODE +  
 **"<br/></span> <strong>WEBSITE URL: </strong> <span class='details'> <a href= \""** + rec.WEBSITE\_URL + **"\" >"** + rec.WEBSITE\_URL +  
 **"</a></span>"**;  
 } **else** {  
 result = [rec.PROVIDER\_NAME, calcUniversityperformance(), rec.GROUPS, rec.BUILDING\_NAME\_NUMBER, rec.LOCALITY, rec.STREET\_NAME, rec.TOWN, rec.POSTCODE, rec.WEBSITE\_URL]  
 }  
  
  
 }  
 **return** result;  
 });  
 *// console.log("result data",result);* **return** result;  
 }  
  
 **function** *getPerformance*(obj) {  
 **var** Performance = {**CountofUOA**: 0, **sumOfOne**: 0, **sumOftwo**: 0, **sumOftree**: 0, **sumOffour**: 0, **sumFTB**: 0};  
  
 **function** *getLeafNodes*(leafNodes, obj) {  
  
 **if** (obj.**children**) {  
  
 obj.**children**.forEach(**function** (child) {  
 *getLeafNodes*(leafNodes, child)  
  
 });  
 } **else** {  
  
 leafNodes.push(obj);  
 }  
 }  
  
 **var** leafNodes = [];  
 *getLeafNodes*(leafNodes, obj.**data**);  
 Performance.**CountofUOA** = leafNodes.**length**;  
 leafNodes.forEach(**function** (child) {  
 **var** childobject = child.**value**[0];  
 *// console.log("childobject[\"1\*\"]",childobject["1\*"])* Performance.**sumOfOne** += ***Math***.round(parseFloat(childobject[**"1\*"**]));  
 Performance.**sumOftwo** += ***Math***.round(parseFloat(childobject[**"2\*"**]));  
 Performance.**sumOftree** += ***Math***.round(parseFloat(childobject[**"3\*"**]));  
 Performance.**sumOffour** += ***Math***.round(parseFloat(childobject[**"4\*"**]));  
 Performance.**sumFTB** += ***Math***.round(parseFloat(childobject[**"FTE Category A staff submitted"**])) / Performance.**CountofUOA**;  
 });  
  
 **return** Performance;  
 }  
  
 *//================== IMPORTANT do not delete ==================================* **return** treechartObject; *// return the main object to the caller to create an instance of the 'class'*}  
  
  
*//=================== End Charts =========================*