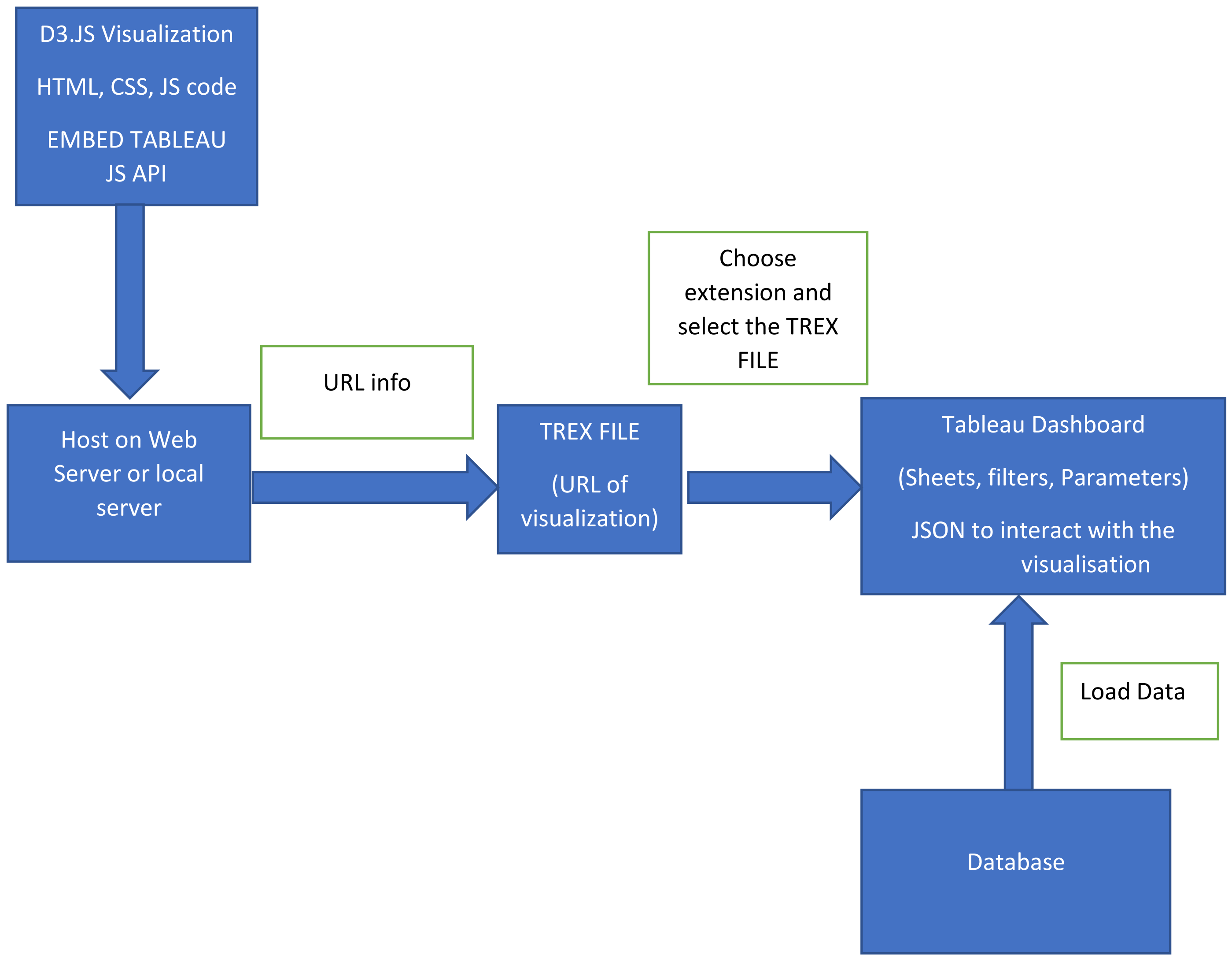
Integrating D3 with Tableau


D3.JS Visualization
HTML, CSS, JS code
EMBED TABLEAU JS API
Host on Web Server or local server
URL info
TREX FILE
(URL of visualization)
Choose extension and select the TREX FILE
Tableau Dashboard
(Sheets, filters, Parameters)
JSON to interact with the visualisation
Load Data
Database

Brief Steps

1. Create a D3.js visualization
2. Create dashboard (filters, Parameters, Sheets) whichever needed
3. Embed Tableau ENTENSION API, TABLEAU JS API to interact the data with the visualization
4. Host the D3 visualization in the server
5. Create TREX file to upload the visualization into tableau
6. Then Manage the dashboard and upload the visualization using extensions (TREX file).

Tableau is powerful visualization tool used in Business Intelligence Industry. It offers a lot of visualizations which can be used to transform raw data to a visual format which is much easier to understand. But sometimes the built-in visualizations are not enough when we have some special type of data which require a specific visualization. In that case, tableau has a feature of extensions. Using the extensions API we can create any visualization by utilizing the power of Javascript.

So we are going to use tableau extension to integrate D3 visulization to interact with tableau data,

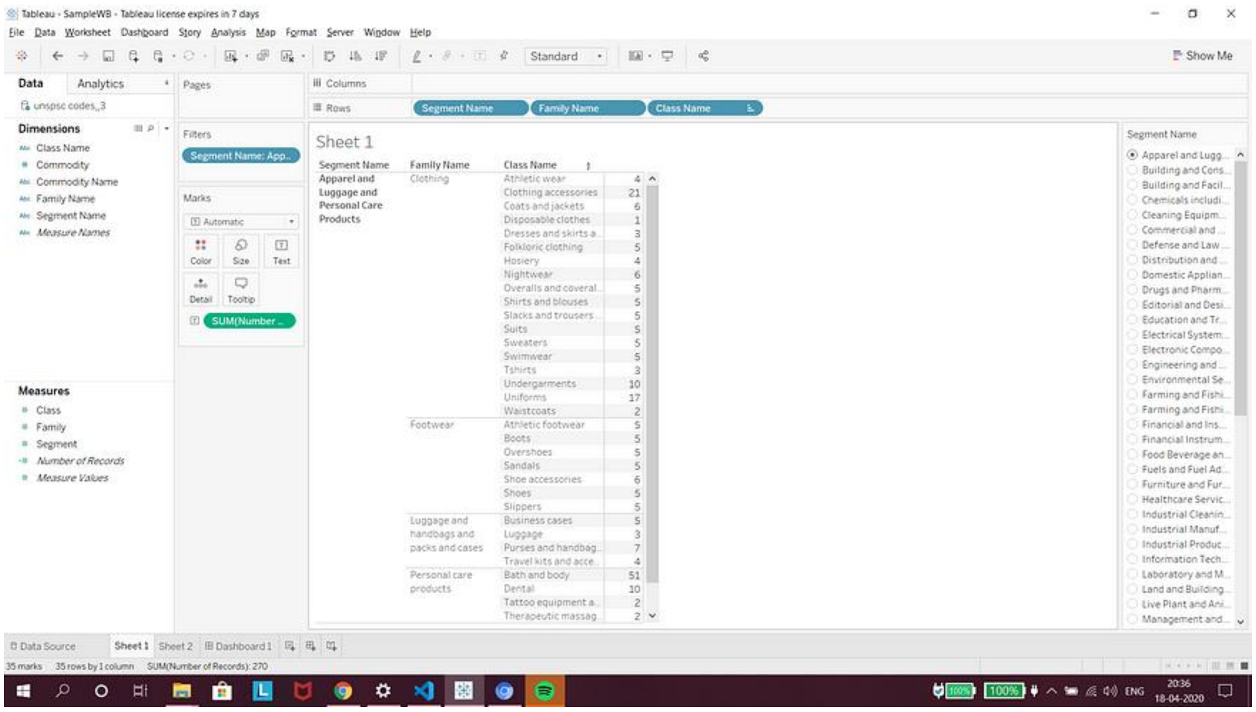
The tableau extensions API basically gives you access to the Tableau Dashboard and its underlying data inside a web page using which you can create whatever you want, be it a fancy D3 chart or a data table with some special features.

Here is sample of sunburst chart made using D3 and integrate with Tableau

1. Create a dashboard in Tableau

We need to create a dashboard in tableau, for which we require a dataset. For the purpose of demonstrating heirarchical data I am using the following dataset

Import the CSV file as new data source inside tableau and create two worksheets as follows :

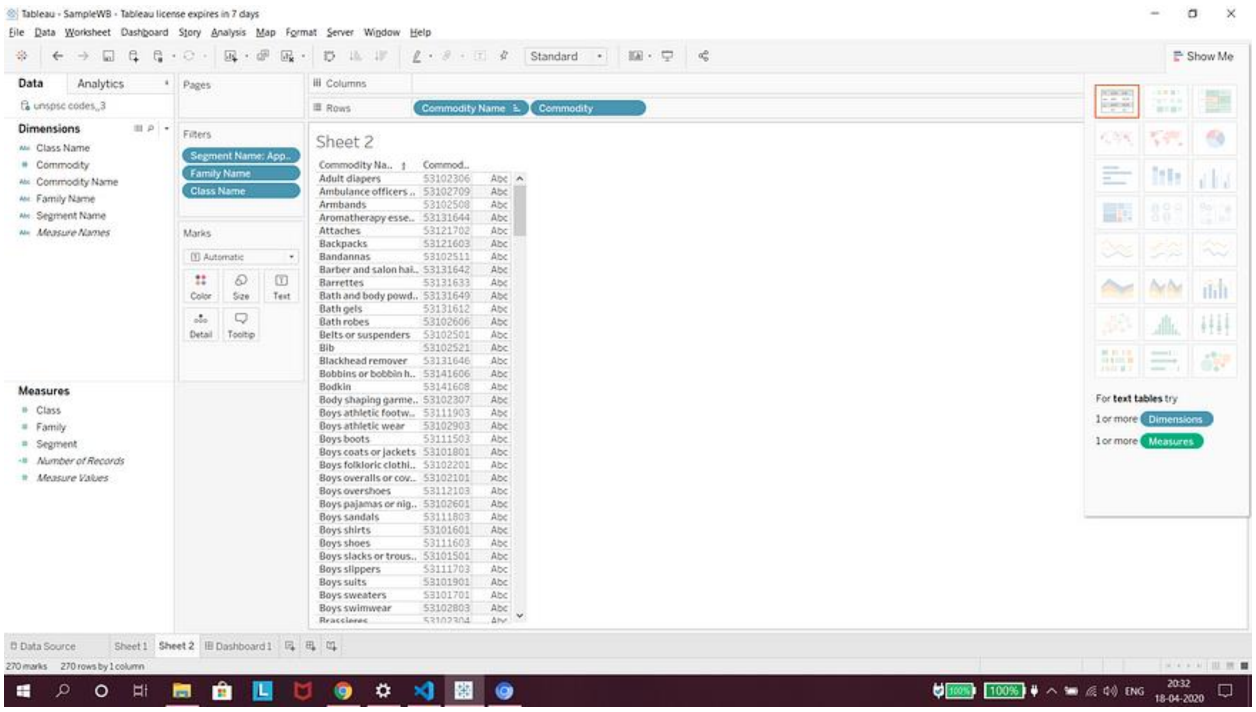


Sheet 1

Segment Name	Family Name	Class Name	
Apparel and Luggage and Personal Care Products	Clothing	Athletic wear	4
		Clothing accessories	21
		Coats and jackets	6
		Disposable clothes	1
		Dresses and skirts a...	3
		Folkloric clothing	5
		Hosiery	4
		Nightwear	6
		Overalls and coveral...	5
		Shirts and blouses	5
		Slacks and trousers ...	5
		Suits	5
		Sweaters	5
		Swimwear	5
		Tshirts	3
		Undergarments	10
		Uniforms	17
		Waistcoats	2
	Footwear	Athletic footwear	5
		Boots	5
		Overshoes	5
		Sandals	5
		Shoe accessories	6
		Shoes	5
		Slippers	5
	Luggage and handbags and packs and cases	Business cases	5
		Luggage	3
		Purses and handbag...	7
		Travel kits and acce...	4
	Personal care products	Bath and body	51
		Dental	10
		Tattoo equipment a...	2
		Therapeutic massag...	2

Worksheet 1

The first worksheet shows the data as a table in a heirarchical form. It also contains a “Segment Name” filter which applies accross all worksheets. Make sure to create the sheet exacty like shown in the screenshot as the data we get from the worksheet inside the Extension depends on the way it is arranged inside the worksheet. The above format gives us the data in a heirarchcal format which is required to create a Sunburst chart.



Sheet 2

Commodity Na..	Commod..	
Adult diapers	53102306	Abc
Ambulance officers ..	53102709	Abc
Armbands	53102508	Abc
Aromatherapy esse..	53131644	Abc
Attaches	53121702	Abc
Backpacks	53121603	Abc
Bandannas	53102511	Abc
Barber and salon hai..	53131642	Abc
Barrettes	53131633	Abc
Bath and body powd..	53131649	Abc
Bath gels	53131612	Abc
Bath robes	53102606	Abc
Belts or suspenders	53102501	Abc
Bib	53102521	Abc
Blackhead remover	53131646	Abc
Bobbins or bobbin h..	53141606	Abc
Bodkin	53141608	Abc
Body shaping garme..	53102307	Abc
Boys athletic footw..	53111903	Abc
Boys athletic wear	53102903	Abc
Boys boots	53111503	Abc
Boys coats or jackets	53101801	Abc
Boys folkloric clothi..	53102201	Abc
Boys overalls or cov..	53102101	Abc
Boys overshoes	53112103	Abc
Boys pajamas or nig..	53102601	Abc
Boys sandals	53111803	Abc
Boys shirts	53101601	Abc
Boys shoes	53111603	Abc
Boys slacks or trous..	53101501	Abc
Boys slippers	53111703	Abc
Boys suits	53101901	Abc
Boys sweaters	53101701	Abc
Boys swimwear	53102803	Abc
Brassieres	53102304	Abc

Worksheet 2

The second worksheet contains a simple table to show individual rows. This would be used to filter out the rows based on the filters applied from the Sunburst chart.

Now create a new dashboard and add the first sheet to it and then the second one. The order in which we add the worksheets to the dashboard is important as we are going to fetch the data from these sheets inside the extension in the same order.

2. Create the Extension

Now, let’s create the extension which we will later add to the tableau dashboard.

A Tableau extension consists of a manifest file (`.trex`), a web page that uses a Tableau-provided JavaScript library, and the JavaScript file (or files) that contain your extension logic.

Lets create a new folder and create a file named Sunburst.trex inside it. Paste the following content inside the trex file.

Sunburst.trex

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest manifest-version="0.1" xmlns="http://www.tableau.com/xml/extension_manifest">
    <dashboard-extension id="com.example.extensions.name" extension-version="0.1.0">
      <default-locale>en_US</default-locale>
      <name resource-id="name"/>
      <description>Extension Description</description>
      <author name="USERNAME" email="USER@example.com" organization="MyCo"
website="https://www.example.com"/>
      <min-api-version>1.0</min-api-version>
      <source-location>
        <url>http://localhost:8000/index.html</url>
      </source-location>
      <icon>

iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAABHNCSVQICAgIfAhkiAAA
AAlwSFlzAAALEwAACxMBAJqcGAAAAlhJREFUOI2Nkt9vy1EYh5/3bbsvRSySCZbIxI+ZCKsN2
TKtSFyIrV2WuRCJuBiJWxfuxCVXbvwFgiEtposgLFJElnbU1SxIZIIRJDKTrdu+53Uhra4mce7O
e57Pcz7JOULFisViwZ+29LAzOSjQYDgz1ZcCvWuXV11MJpN+OS/lm6179teqH0yDqxPTCyKS
A8DcDsyOmOprnCaeP7459pdgy969i0LTC3IO/RQMyoHcQN+3cnljW3dNIFC47qDaK3g7Bwd
TkwBaBELT4ZPOUVWgKl4ZBnjxJPUlMDnTDrp0pmr6RHFeEjjcUUXPDGeSEwDN0Xg8sivx
MhJNjGzbHd8PkM3eHRfkrBM5NkcQaY2vUnTlrDIA0NoaX+KLXFFlowr14tvVpqb2MICzmQc
Kqxvbumv+NAhZGCCIPwEw6QWXKYRL/VUXO0+rAUJiPwAk5MIlgVfwPjjHLCL1APmHN94
ZdqeYN+NW/mn6I4BvwQYchcLnwFhJMDiYmlRxAzjpKWZkYkUCcZ2I61wi37tLbYyjiN0fHk5Oz
3nGSLSzBbNHCF35R7f6K1/hN9PRhek11FrymfQQQKB4+Gl05P2qNRtmETlXW7e+b2z01dfyc
GNbfFMAbqNyKp9Jp4rzOT8RYFs0njJkc2iqsCObvTsOsDWWqA5C1uFy+Uz/oXJeKwVT4h0Rm
PUXhi79vuC0Ku6yOffTK3g9lfxfDQAisY516sg5kfOCiJk7HoLt2cf9b/9LANAc7dznm98PagG1fU
OZ9IP5uMB8Q4CPoyNvausapkTt3rNMuvdf3C/o6+czhtdwmwAAAABJRU5ErkJggg==
      </icon>
    </dashboard-extension>
  <resources>
    <resource id="name">
      <text locale="en_US">Sunburst</text>
      <text locale="fr_BE">Sunburst</text>
      <text locale="de_DE">Sunburst</text>
    </resource>
  </resources>
</manifest>
```

The manifest file (`EXTENSION-NAME.trex`) is an XML file that describes the extension and provides information to register the extension with Tableau.

The url in the above file is the path to the webpage where the extension is hosted. Here we are going to host it on the localhost and port 8000.

Now create a folder inside it named lib and put the Extensions API JavaScript library (`tableau.extensions.1.latest.js`) in it, which is available inside lib folder in the Extensions API sample provided by tableau here. You can also find this file in my repository mentioned at the end of this post.

Create a new file named `index.html` inside previous folder which looks like following :

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <style>
    path {
      stroke: #fff;
      fill-rule: evenodd;
    }
    text {
      font-family: Arial, sans-serif;
      font-size: 12px;
      pointer-events: none;
    }

    .filters {
      margin: 20px auto;
      display: flex;
      justify-content: center;
      flex-direction: column;
    }

    div.tooltip {
      padding: 5px;
      position: absolute;
      text-align: center;
      min-width: 100px;
      min-height: 30px;
      font: 14px sans-serif;
      background: #fff;
      box-shadow: 1px 2px 6px 1px rgba(48, 48, 48, 1);
      border: 0px;
      border-radius: 4px;
      pointer-events: none;
    }
  </style>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <title>Sunburst</title>

  <script src="https://code.jquery.com/jquery-3.3.1.js" integrity="sha256-
2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
    crossorigin="anonymous">
    </script>
  <script src="https://cdn.jsdelivr.net/npm/lodash@4.17.11/lodash.min.js"></script>

  <!-- Include the Tableau Extensions library so we can communicate with Tableau -->
  <script src="./lib/tableau.extensions.1.latest.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.17/d3.js"></script>

  <!-- Include our own style sheets and javascript code -->
  <script src="./script.js"></script>
```

```
</head>

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

This file is the entry point of the extension. It includes the Extensions API javascript library, D3, and lodash which we need to create the custom visualization. It also contains some basic styles for our extension.

Next, create the `script.js` file which contains all the logic of our extension. It fetches data from the dashboard, converts it into appropriate json format and then plots the sunburst chart using d3. It also adds the event listeners for filters both to and from the extension.
script.js

```
'use strict';

// Wrap everything in an anonymous function to avoid poluting the global namespace
(function () {
  // Event handlers for filter change
  let unregisterHandlerFunctions = [];

  let worksheet1, worksheet2;
  // Use the jQuery document ready signal to know when everything has been initialized
  $(document).ready(function () {
    // Initialize tableau extension
    tableau.extensions.initializeAsync().then(function () {
      // Get worksheets from tableau dashboard
      worksheet1 = tableau.extensions.dashboardContent.dashboard.worksheets[0];
      worksheet2 = tableau.extensions.dashboardContent.dashboard.worksheets[1];


      function getDataAndPlotChart() {
        // load data from worksheet
        let dataArr = [];
        worksheet1.getSummaryDataAsync().then(data => {
          let dataJson;
          data.data.map(d => {
            dataJson = {};
            dataJson[data.columns[0].fieldName] = d[0].value; //1st column
            dataJson[data.columns[1].fieldName] = d[1].value; //2nd column
            dataJson[data.columns[2].fieldName] = d[2].value; //3rd column
            dataJson[data.columns[3].fieldName] = d[3].value; //4th column
            dataArr.push(dataJson);
          });

          // converting data to heirarchical json
          let formattedJson = _(dataArr)
            .groupBy(x => x["Segment Name"])
            .map((value1, key) => ({
              name: key, count: sum(value1), children: _(value1)
                .groupBy(x => x["Family Name"])
                .map((value2, key) => ({
                  name: key, count: sum(value2), children: _(value2)
                    .groupBy(x => x["Class Name"])
                    .map((value3, key) => ({ name: key, count: sum(value3), children: [] }))
                    .value()
                }))
                .value()
            }))
            .value();

          plotChart(formattedJson);
        });
      }

      getDataAndPlotChart();

      // event listener for filters
      let unregisterHandlerFunction =
worksheet1.addEventListener(tableau.TableauEventType.FilterChanged,
filterChangedHandler);
      unregisterHandlerFunctions.push(unregisterHandlerFunction);

      function filterChangedHandler(event) {
        // for filter change
        // Add fieldName with (||) for other filters
        if (event.fieldName === "Segment Name") {
          // reload summary data
          getDataAndPlotChart();
        }
      }
    });
```

```
});

function sum(arr) {
  let count = 0;
  arr.forEach(element => {
    count += parseInt(element["SUM(Number of Records)"]);
  });
  return count;
}

// ========================== D3 CHART ====================== //
function plotChart(data) {

  var div = d3.select("body").append("div")
    .attr("class", "tooltip")
    .style("opacity", 0);

  var width = 600,
    height = 600,
    radius = height / 2;

  var x = d3.scale.linear()
    .range([0, 2 * Math.PI]);

  var y = d3.scale.linear()
    .range([0, radius]);

  var color = d3.scale.category20c();
  var arc;
  function graph() {

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

    var partition = d3.layout.partition()
      .value(function (d) { return d.count; });

    arc = d3.svg.arc()
      .startAngle(function (d) {
        return Math.PI / 2 + Math.max(0, Math.min(2 * Math.PI, x(d.x)));
      })
      .endAngle(function (d) {
```

```
    return Math.PI / 2 + Math.max(0, Math.min(2 * Math.PI, x(d.x + d.dx)));
  })
  .innerRadius(function (d) { return Math.max(0, y(d.y)); })
  .outerRadius(function (d) { return Math.max(0, y(d.y + d.dy)); });

var root = data[0];

var g = svg.selectAll("g")
  .data(partition.nodes(root))
  .enter().append("g")
  .on("mouseover", function (d) {
    div.transition()
      .duration(200)
      .style("opacity", .9);
    div.html(d.name + "<br/> Count : " + d.count)
      .style("left", (d3.event.pageX + 10) + "px")
      .style("top", (d3.event.pageY - 28) + "px");
  })
  .on("mouseout", function (d) {
    div.transition()
      .duration(200)
      .style("opacity", 0);
  });

var path = g.append("path")
  .attr("d", arc)
  .style("fill", function (d) { return color((d.children ? d : d.parent).name); })
  .on("click", (d) => click(d));
var text = g.append("text")
  .attr("transform", function (d) {
    return "translate(" + arc.centroid(d) + ")rotate(" + computeTextRotation(d) + ")";
    // return "rotate(" + computeTextRotation(d) + ")";
  })
  .attr("text-anchor", "middle")
  .attr("dx", "0") // margin
  .attr("dy", ".35em") // vertical-align
  .text(function (d) {
    return d.dx * height > 7 ? `${d.name.substring(0, 10)}...` : "...";
  })

function click(d) {
  // apply filters from d3 chart to worksheet2 to populate respective data
  let segment = "Segment Name", family = "Family Name", className = "Class Name";
  switch (d.depth) {
    case 0: {
      worksheet2.clearFilterAsync(family).then(
              worksheet2.clearFilterAsync(className).then(
                worksheet2.applyFilterAsync(segment, [d.name], tableau.FilterUpdateType.Replace)
              )
            )
            break;
          }
          case 1: {
            worksheet2.clearFilterAsync(className).then(
              worksheet2.applyFilterAsync(segment, [d.parent.name],
tableau.FilterUpdateType.Replace).then(
                worksheet2.applyFilterAsync(family, [d.name], tableau.FilterUpdateType.Replace)
              )
            )
            break;
          }
          case 2: {
            worksheet2.applyFilterAsync(segment, [d.parent.parent.name],
tableau.FilterUpdateType.Replace).then(
              worksheet2.applyFilterAsync(family, [d.parent.name],
tableau.FilterUpdateType.Replace).then(
                worksheet2.applyFilterAsync(className, [d.name],
tableau.FilterUpdateType.Replace)
              )
            )
            break;
          }
          default:
        }

        text.transition().attr("opacity", 0);

        path.transition()
          .duration(750)
          .attrTween("d", arcTween(d))
          .each("end", function (e, i) {
            // check if the animated element's data e lies within the visible angle span given in d
            if (e.x >= d.x && e.x < (d.x + d.dx)) {
              let startAngle = Math.PI / 2 + Math.max(0, Math.min(2 * Math.PI, x(e.x)));
              let endAngle = Math.PI / 2 + Math.max(0, Math.min(2 * Math.PI, x(e.x + e.dx)));
              // get a selection of the associated text element
              var arcText = d3.select(this.parentNode).select("text");
              // fade in the text element and recalculate positions
              arcText.transition().duration(750)
                .attr("opacity", 1)
                .text(d => {
                  return (endAngle - startAngle) > 0.080 ? `${d.name.substring(0, 10)}...` : "...";
                })
                .attr("transform", function () {
                  return "translate(" + arc.centroid(e) + ")rotate(" + computeTextRotation(e) + ")";
                })
                .attr("text-anchor", "middle")
            }
          });
        }
      }
      graph();

      // Interpolate the scales!
      function arcTween(d) {
        var xd = d3.interpolate(x.domain(), [d.x, d.x + d.dx]),
        yd = d3.interpolate(y.domain(), [d.y, 1]),
        yr = d3.interpolate(y.range(), [d.y ? 20 : 0, radius]);
        return function (d, i) {
        return i
            ? function (t) { return arc(d); }
          : function (t) { x.domain(xd(t)); y.domain(yd(t)).range(yr(t)); return arc(d); };
        };
      }

      function computeTextRotation(d) {
        var ang = (Math.PI / 2 + x(d.x + d.dx / 2) - Math.PI / 2) / Math.PI * 180;
        return (ang > 270 || ang < 90) ? ang : 180 + ang;
      }
    }
})();
```

WHOA! That's a lot of code! Let's try to understand what's actually going on here. First of all we are executing a function when the document is ready, which ensures that all our dependencies are loaded. Then, we have declared some variables which we will need later.

Next, `tableau.extensions.initializeAsync()` is the function provided by tableau extensions API which initializes the extension. It returns a promise so we attached `.then()` to it which is also a function and accepts another function as an argument, that we want to run when the promise is resolved successfully.

`tableau.extensions.dashboardContent.dashboard` is the object which gives access to different parts of the dashboard. We have fetched the worksheets included in the dashboard using `tableau.extensions.dashboardContent.dashboard.worksheets` which gives us an array of worksheets, so we have assigned both of our worksheets to the variables respectively.

Inside the `getDataAndPlotChart` function we are using `getSummaryDataAsync` function provided by the extension API to get the underlying data of a worksheet. After which, we have formatted the data to get an array of objects (group of key-value pairs), where each object represents a single row like below:

```
{"Segment Name" : "Defense and Law Enforcement and Security and
Safety Equipment and Supplies","Family Name" : "Conventional war
weapons","Class Name" : "Naval weapons","SUM(Number of Records)" :
1,}
```

Then, we have converted the data into hierarchical JSON required for plotting sunburst chart by d3, using `groupBy` function of lodash. After which we have called the `plotChart` function which contains the d3 code to plot the sunburst chart and also adds various filters to worksheet2 on click of each section depending upon the depth of section in sunburst.

In line no. 60, we have added an EventListener to the worksheet1, which listens for filterChange events and executes our `filterChangedHandler` function whenever a filter is changed in the worksheet1. The `filterChangedHandler` function checks if the filter applied is on “Segment Name” and refreshes the data and the d3 chart if so.

3. Host the Extension

Now, we have to start a web server to host the extension. For this, you can either use python simplehttpserver if you have python installed by running the following command in a terminal inside the folder which contains the html file :

```
python -m SimpleHTTPServer 8000
```

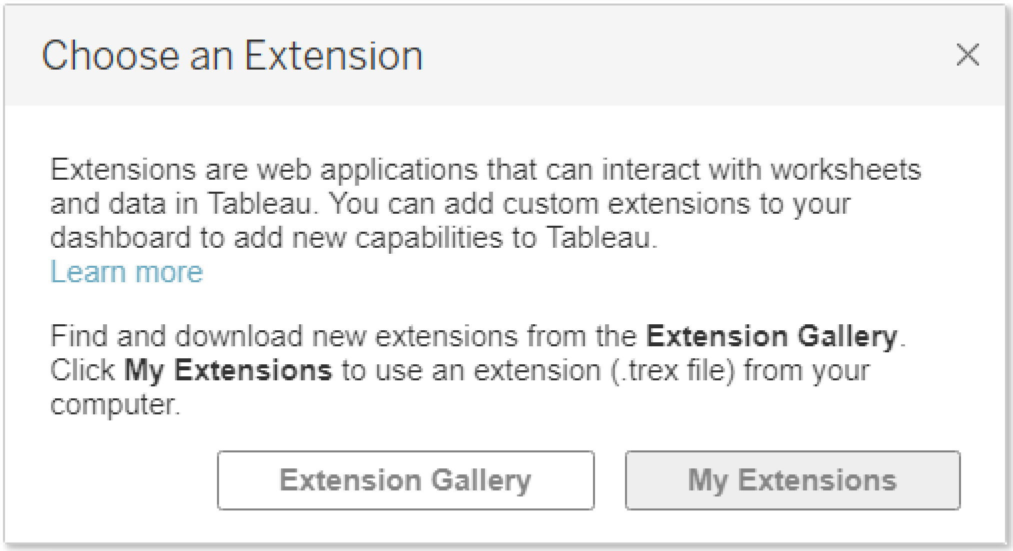
Or you can use the http-server npm package for which you have to have node installed. For this, run the following commands in a terminal inside the same folder :

```
npm install http-server -g
http-server -p 8000
```

Note that the port here is the same that we had entered in the url inside the trex file.

4. Add the extension to the Dashboard

To add the extension to the dashboard, click and drag Extension from Objects pane (at the lower left corner) into the dashboard. Select “My extensions” from the dialogue box which opens.



Choose an Extension

Extensions are web applications that can interact with worksheets and data in Tableau. You can add custom extensions to your dashboard to add new capabilities to Tableau.
Learn more

Find and download new extensions from the **Extension Gallery**. Click **My Extensions** to use an extension (.trex file) from your computer.

Browse and select the trex file that we just created. Select “OK” from the next dialogue box and wait for a few seconds.

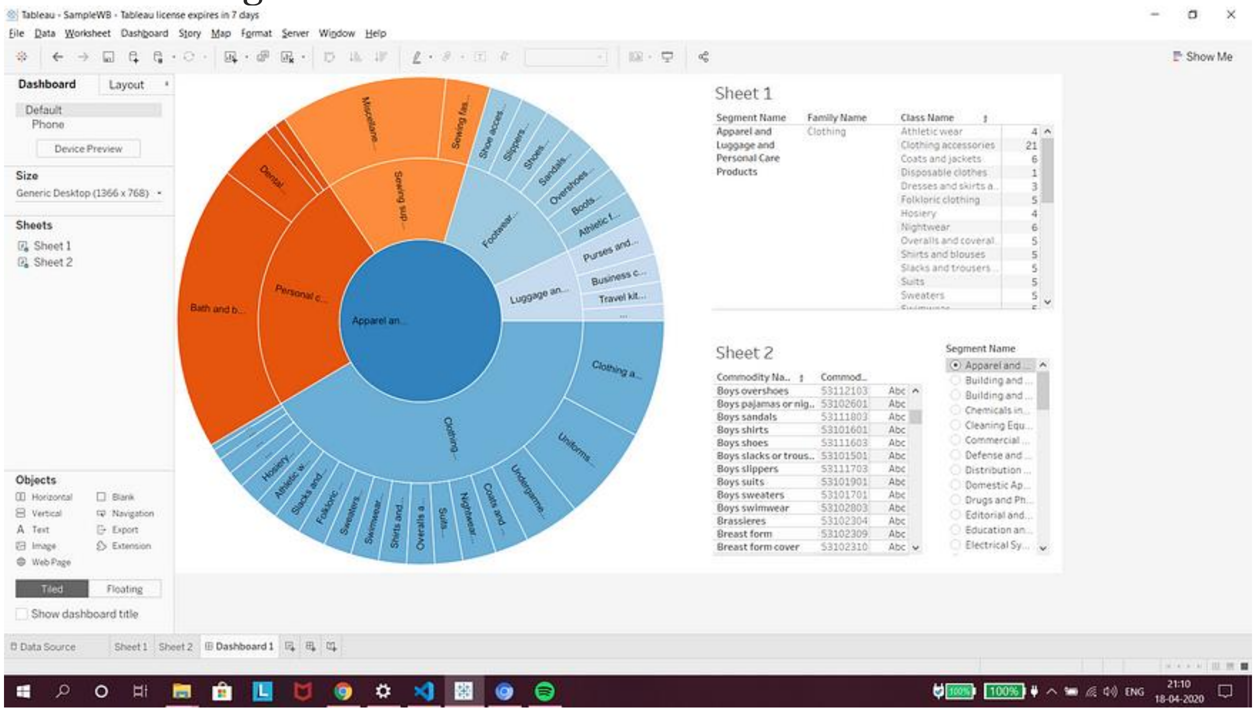


Tableau - SampleWB - Tableau license expires in 7 days

Sheet 1

Segment Name	Family Name	Class Name	
Apparel and Luggage and Personal Care Products	Clothing	Athletic wear	4
		Clothing accessories	21
		Coats and jackets	6
		Disposable clothes	1
		Dresses and skirts a..	3
		Folkloric clothing	5
		Hosiery	4
		Nightwear	6
		Overalls and coveral..	5
		Shirts and blouses	5
		Slacks and trousers ..	5
		Suits	5
		Sweaters	5

Sheet 2

Commodity Na..	Commod..	
Boys overshoes	53112103	Abc
Boys pajamas or nig..	53102601	Abc
Boys sandals	53111803	Abc
Boys shirts	53101601	Abc
Boys shoes	53111603	Abc
Boys slacks or trous..	53101501	Abc
Boys slippers	53111703	Abc
Boys suits	53101901	Abc
Boys sweaters	53101701	Abc
Boys swimwear	53102803	Abc
Brassieres	53102304	Abc
Breast form	53102309	Abc
Breast form cover	53102310	Abc

Segment Name
- [✓ Apparel and ...]
- Building and ...
- Building and ...
- Chemicals in...
- Cleaning Equ...
- Commercial ...
- Defense and ...
- Distribution ...
- Domestic Ap...
- Drugs and Ph...
- Editorial and...
- Education an...
- Electrical Sy...

https://www.theinformationlab.co.uk/2018/10/15/build-your-first-tableau-dashboard-extension/