Yaser Nossibeh

Yaser.nossibeh@rci.rogers.com

Overview

This document will serve as a guide to the implementation and usage of the custom Power BI annotated line graph.

Custom Power bi visual

Annotated Line Graph

Contents

[About 2](#_Toc139981723)

[Usage 2](#_Toc139981724)

[Code Explanation 4](#_Toc139981725)

[Data Processing 4](#_Toc139981726)

[Data Rendering 6](#_Toc139981727)

[Interactivity 7](#_Toc139981728)

[Annotation with Dragging Functionality 8](#_Toc139981729)

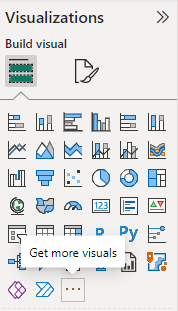
[References 10](#_Toc139981730)

# About:

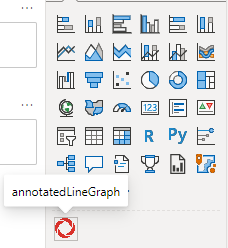
This project was motivated by the current limitations of Microsoft’s Power BI platform. Network planners were seeking the ability to insert text-based annotations onto their visuals, but were unable to do so using the visuals provided by Power BI. Using Power BI’s developer API, TypeScript, HTML, and LESS, we were able to create an interactive line chart that automatically creates annotations from input.

# Usage:

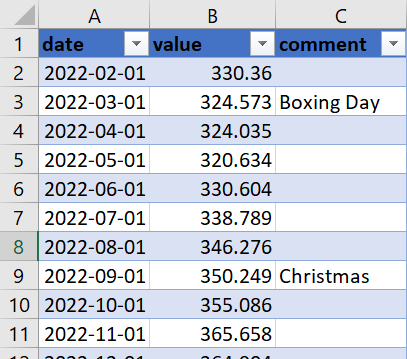
1. Download [this](https://rcirogers.sharepoint.com/:u:/r/sites/NetworkCapitalManagement/Network_Economics/Shared%20Documents/Nwk%20Economics/RawDataAggregation/FORECAST/Documentations/Yaser/annotatedLineGraphF60967A1C89C4EAC9B003DF82B16E019.1.0.0.pbiviz?csf=1&web=1&e=h9m0aZ) file.
2. Open Power BI.
3. Import the downloaded file using the “Get more visuals” button.



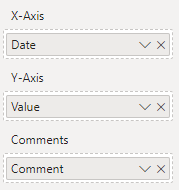
1. Select the new visual and add it to your report.



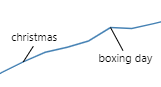
1. Upload your desired dataset. Make sure it is structured as follows:



1. Select columns for the X-Axis, Y-Axis, and Comments data.



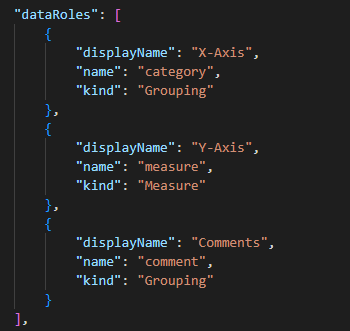
1. Drag the annotations on the graph to the desired location.

# Code Explanation:

## Data Processing

For this visual, the initial data processing/setup is done in the capabilities.json file. In this file, there are two main important sections: DataRoles and DataViewMappings.



DataRoles, as shown above, correspond to the input sections visible to the user in Power BI. We have created 3 data roles: X-Axis, Y-Axis, and Comments. The relationship between these roles is defined in the DataViewMappings.



In the condition section, we have limited each DataRole to accept at most one value. There are different ways you can map data (ie. Single, Categorical, Table). For this visual, we use a “table” DataViewMapping. This allows us to map the x-axis and y-axis variables, while attaching the annotation text to them as well, which is not naturally supported in Power BI. As a result of our mapping, a DataView in the following structure is created:

{ …

table:

…

rows:

0: [330.36, ‘2022-01-01’, ‘New Years Day’]

1: [324.573, ‘2022-02-01’, Null]

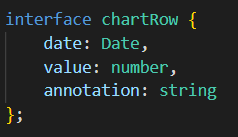
…

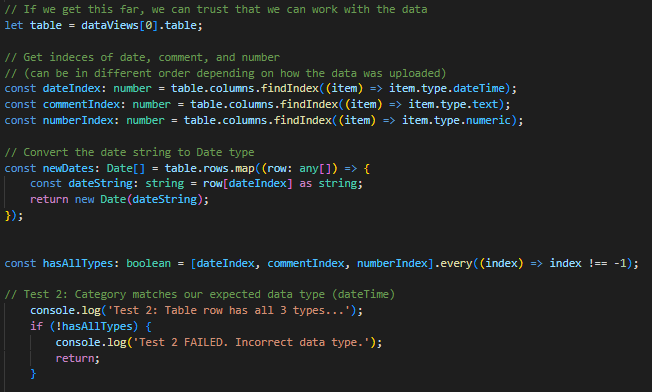
n: [value, date, comment] *\*\*Data can be in different order\*\**

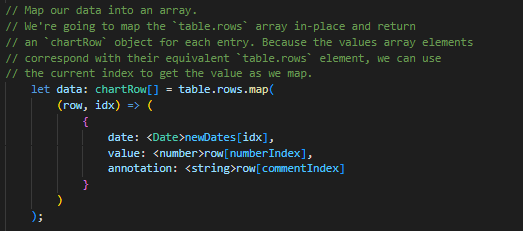
…

… }

We then iterate through the DataView, performing data validation along the way, to create our desired data structure called “Data”, which is an array of chartRow objects.



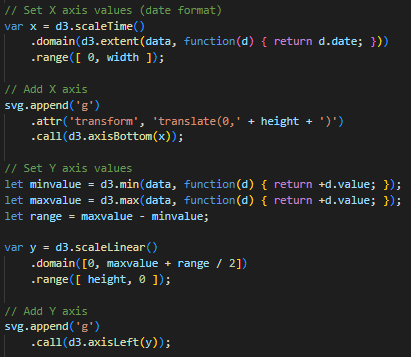




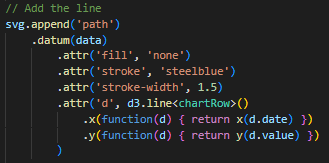
That is it for our data processing portion. We now have all the data we need in the “Data” array, which we will use to render the data in the following section.

## Data Rendering

To render the data visualization, we used the D3.js library. Based on the data, appropriate x-axis and y-axis scaled are determined and added to the graph.



Lastly, we add the line to the SVG, along with the data points from the “Data” array:

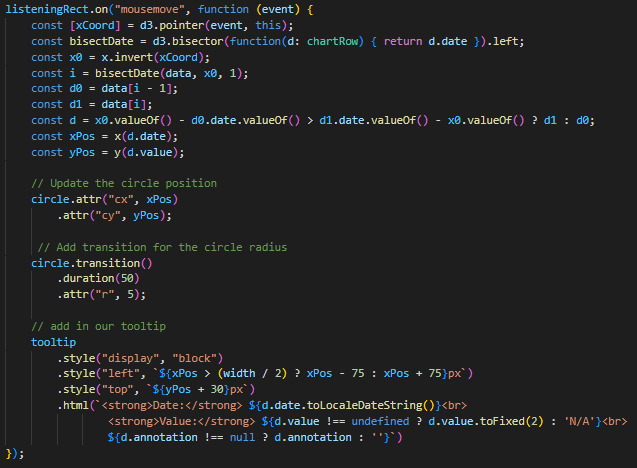
****

That concludes the data rendering portion, we now have an SVG element that displays a simple line graph with data from Power BI.

## Interactivity

In addition to the basic line chart, we have decided to offer interactive features to enhance the readability of the graph. The first feature we implemented is the tooltip, which appears when you hover over the graph.

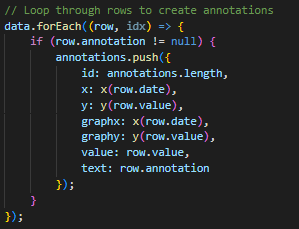
We created a listeningRect, which listens for mouse movement and runs the following function whenever the mouse hovers over the graph:



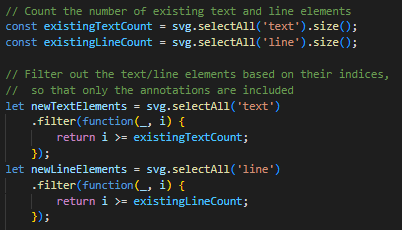
The function above essentially calculates the closest point on the graph to the current mouse position using the D3 bisector method. After determining the closest point, we move a small circle to that data point on the line, and display the tooltip featuring the date, value, and comment (if available).

## Annotation with Dragging Functionality

The main feature of this graph is the ability to add text annotations to charts, which is not a natural feature in Power BI. To create the annotations, we loop through the “Data” array and check if the rows have a comment or not. For the rows that do have comments, we add them to an array called “annotations”. This array stores objects that hold information for each annotation.



We then get the existing number of text and line elements in the SVG, in order to not alter the pre-existing elements when we add the new annotations and arrows.



We will use the newTextElements and newLineElements to initialize the annotations, and also whenever we want to update them after an edit has been made. We have the following update() method which does exactly that. It sets up the annotations (text and line elements) if they do not yet exist, and updates them if they do already exist.



In the section above we briefly mention the dragging functionality, but we have not yet introduced it. The dragging functionality allows the text portion of annotations to be dragged anywhere on the graph using the mouse. To implement the ability to drag items, we first call d3.drag(), and create 3 functions alongside it, which are called when the drag event starts, updates, and ends, respectively.



Now that we have explained how the drag functionality works, we can see that the update() method from above is called whenever an annotation is grabbed and moved to a new location. It essentially changes a text’s location every time an event occurs, while also changing the endpoint of the corresponding line to follow as well.

# References:

[Power BI visuals documentation - Power BI | Microsoft Learn](https://learn.microsoft.com/en-us/power-bi/developer/visuals/)

[Lifting and Shifting a Simple d3.js Line Chart into A Power BI Custom Visual (Part 1) | coacervo](https://coacervo.co/examples/d3-line-chart-1)

[Using the `table` DataViewMapping in Power BI Custom Visuals | coacervo](https://coacervo.co/examples/table-dv-simple#:~:text=The%20Power%20BI%20custom%20visuals%20SDK%20allows%20you,when%20they%20are%20exposed%20to%20your%20custom%20visual.)

[Picking, Dragging and Brushing with D3 (d3indepth.com)](https://www.d3indepth.com/interaction/)

[Getting started | D3 by Observable (d3js.org)](https://d3js.org/getting-started)