Applying D3.js: Stock Price for US IT Companies in the 2000s

Der-Hsuan Tsou, Ngai Tik Chung, Xinyi Ma, *Student, UNCC*

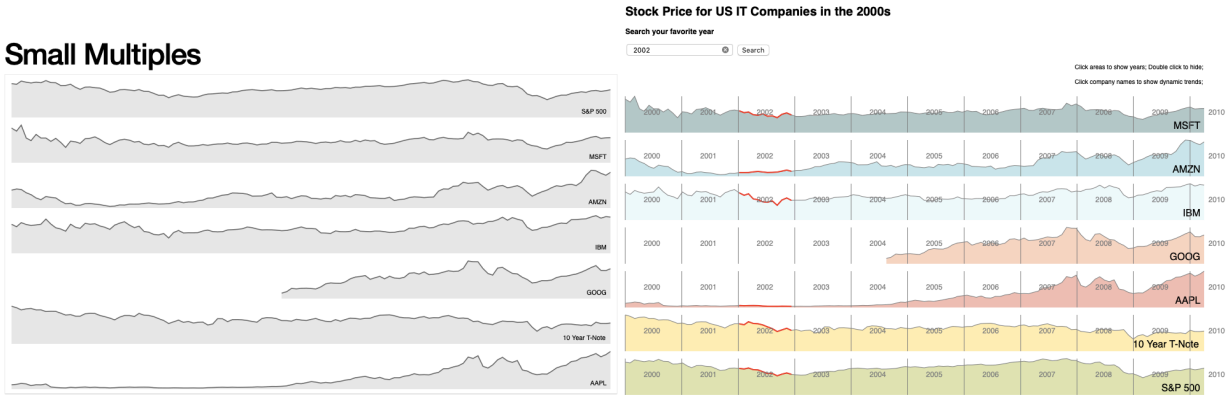

Fig. 1. Original Interface VS. Final Interface of this Project

Abstract—The project intends to improve the original D3 interface, which provides a great learning experience for us in terms of fully delivering our messages with the skills of Visual Analytics. The project has both scientific and creative goals. The scientific goals demonstrate our expertise in utilizing JavaScript, CSS, SVG, and D3 in Visual Studio Code and WebStorm throughout the project, including identifying the defects and attempting to run the script. For creativity goals, the project raises our concerns with the original interface on what changes to better aid the users viewing such visualizations. Changes we have implemented such as changing the colors, converting to dynamic graph, adding axis and appending a search button allow users to view the data specifically.

Index Terms—D3, JavaScript, IT Company, Stock Price, Interactive Visualization, Dynamic Graph

1 Introduction

As the technology advanced, people spend more and more time on the internet. According to the Digital 2019 report, people spend on average 6 hours and 42 minutes online each day. It equates to more than 100 days of online time every year for every Internet user. In other words, we spend more than 27 percent of the year on the internet [1]. While we spend tons of time on the internet, we will also receive lots of information. Thus, it is important to help users understand the idea we want to explain in a limited time while keeping the information in their minds.

The example that our group chose displayed the trend of stock price of 5 US IT companies in the 2000s with two indexes [2]. There are total 806 data in the dataset, which showed three tuples. First, symbols mean the name of the companies, including Apple, Amazon, IBM, Microsoft and Google and two indexes which are S&P500 and 10Year T-Note, respectively. Second, dates from the beginning of 2000 to March, 2010. However, the timeframe for Google started from August, 2004 as it is its IPO date. Finally, the data showed the stock price regarding to the period.

Although it should be an informative visualization that help the user quickly understand the price float and the events happened in the stock market, we can hardly see any information from the example. With only seven grey area charts and the abbreviations on the right, it is difficult to catch the viewer's attention, needless to say remain some impression in the user's mind.

In this article, we are going to modify this graph become an interactive and users-friendly visualization with D3.js.

2 Description of the Existing Interface

The existing interface presented a simple and clean concept. By looking at the original interface, it is displayed in black, grey and white. With company stock names on the right with small font size. Besides, the graph is still and not interactive. However, the original script used D3.js as well.

2.1 Technique I

The dataset that used in the existing interface is stored in CSV file and loaded asynchronously. The D3 function show below will read the file and load the data into an array of objects.

```
d3.csv("stocks.csv", type,
function(error, data) {
```

2.2 Technique II

After loading the data, they are nested in symbols by d3.nest. It allows the elements in the array be grouped into a hierarchical tree structure. As here, it only nested in one level, symbol.

- *Der-Hsuan Tsou is with DSBA Master's Program of UNCC. E-mail: dtsou@uncc.edu.*
- *Ngai Tik Chung is with DSBA Master's Program of UNCC. E-mail: nchung2@uncc.edu.*
- *Xinyi Ma is with DSBA Master's Program of UNCC. E-mail: xma10@uncc.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/.

```
var symbols = d3.nest()
    .key(function(d) { return
d.symbol; })
    .entries(data);
```

2.3 Technique III

The third D3.js technique used in the original example is to draw the x-axis. The x-scale is set to be between the range of maximum and the minimum of the date. However, it did not mark the label on the x-axis, which will lead confusion to the users.

```
x.domain([
    d3.min(symbols, function(s)
{ return s.values[0].date; }),
    d3.max(symbols, function(s)
{ return s.values[s.values.length -
1].date; })
]);
```

2.4 Technique IV

Then, it set up multiple y-axes so that every nested symbol has its independent y-scale. The y-scale represents the price of the stock price. Also, there is no y-axis label, which is not user-friendly.

```
svg.append("path")
    .attr("class", "area")
    .attr("d", function(d)
{ y.domain([0, d.maxPrice]); return
area(d.values); });

svg.append("path")
    .attr("class", "line")
    .attr("d", function(d)
{ y.domain([0, d.maxPrice]); return
line(d.values); });
```

2.5 Technique V

To better display the floating trend of the graph, the area and line shapes are used to fill and stroke separately. However, it is still difficult to tell the performance of the price in a specific year without adding highlights and proper notes.

```
.liline {
    fill: none;
    stroke: #666;
    stroke-width: 1.5px;
}
.area {
    fill: #e7e7e7;
}
```

3 Modifications by the Project Team

In this project, totally we have made four major modifications with the original interface.

3.1 Modification I

First of all, the original interface is monochromatic. All the companies on the canvas were assigned same colors. As a result, the whole visualization is so boring that can hardly catch the attention of its audience. Rost emphasized the importance of color in data visualization. She believes that color can be used to guide the viewer's eye and draw attention to particular features [3]. Therefore, our first modification is to use different colors to differentiate companies (or indexes). Seven colors are represented using Hex codes and defined in one array variable:

```
var selectColors =
['#004C4F','#5EB1BF','#CDEDF6','#EF7
B45','#D84727','#FFCE21','#97B200'];
```

Then, when creating the area path element, we add a function to return the color code sequentially and use the returned value to fill the area:

```
.attr("fill", function(d, i) {return
selectColors[i]; })
```

In addition, in order to have more interactions with users, we add two mouse events. We set the original opacity of all the areas as "0.3". When the mouse is on one area, the opacity of that area will change to "1" to catch the attention of the user. When the mouse is out of that area, the opacity will change back to "0.3":

```
.on('dblclick', function(){
    svg.selectAll('.line1')
        .remove();
    svg.selectAll('.yearT')
        .remove();});
```

3.2 Modification II

The second deficiency we find with the original interface is that, it does not contain axis. Therefore, it is hard to identify year for each part of the area. To solve this problem, we make the second modification including adding the year division lines and year texts. To realize this modification, we first calculate the locations of each lines. The distance between the first value to the last value in the chart is 940 px, and in between there are 123 months. For 2000 to 2009, we have the data for the whole year while for the year 2010, we only have data in 3 months. After calculation, one array variable yearDIV containing the horizontal axis of each line was defined:

```
var yearDIV = [0, 940*12/123,
940*24/123, 940*36/123, 940*48/123,
940*60/123, 940*72/123, 940*84/123,
940*96/123, 940*108/123,
940*120/123];
```

Based on "yearDIV", a series of lines and texts are added:

```
svg.selectAll('.line1')
    .data(yearDIV)
    .enter().append("line")
    .attr('class', 'line1')
    .attr("x1", function(d,i)
{return yearDIV[i];})
    .attr("y2", 0)
    .attr("x2", function(d,i)
{return yearDIV[i];})
    .attr("y2", height)
    .attr("stroke", "grey")
    .attr("stroke-width", "1px");

svg.selectAll('.yearT')
    .data(yearDIV)
    .enter().append('text')
    .attr('class', 'yearT')
    .attr('x', function(d,i) {return
yearDIV[i]+30;})
    .attr('y', 30)
    .text(function(d,i){return
i+2000})
    .attr('font-size', 12)
    .attr('fill','grey')
```

Although the year division lines are useful in analysing data, they are too dense and may be annoying for someone if remaining on the screen. So we add another mouse event so that the lines and texts can appear when the user click the area and disappear if the user double click the area.

```
.on('dblclick', function(){
    svg.selectAll('.line1')
        .remove();
    svg.selectAll('.yearT')
        .remove();});
```

3.3 Modification III

In the third modification, we want to add a dynamic transformation of the area to show how the stock price of these companies changes in 11 years. The way we realize it is to first create shadows that cover the whole canvas and then move them out of the canvas slowly using transition. In user's eyes, the whole area will slowly appear on the left. We also add mouse event to activate the dynamic change only when the user click the company's name.

```
.on('click', function(){
    let shadow = svg.append('rect')
        .attr('width', width+50)
        .attr('height', height)
        .style('fill', 'white')
        .attr('x', 0);
     shadow.transition()
        .delay(500)
        .duration(10000)
        .attr('x', width+50)
        .attr('width', 0);});
```

3.4 Modification IV

In the last modification, we add a red line on the graph for the selected year chosen by the user inside the search button. Creating a search bar allows the user to view the data based on their interests. In this case, the graph is filtered by the year to see the stock price within that perspective year.

The first step is to add a search button:

```
<div id="search">
    <input type="search" id="year-
search" name="search"
placeholder="Search for year.."  >
    <input type="button" id="my-
button" value="Search">
</div>
```

Then, we add new class name with the desired style:

```
.highlights {
        fill: none;
        stroke: red;
        stroke-width: 2px;
    }
```

Next, we add an action listener to the button:

```
document.getElementById("my-
button").addEventListener("click",
function(e) {})
    }
```

In other words, this is simply saying from this document searching by the ID to return function (e) on click.

Finally, we create a variable for the selectedYear chosen by the user and assign inside the action listener:

```
var selectedYear =
Number(String(document.getElementByI
d('year-search').value).trim());
if (!!selectedYear) {
svg.select(".highlights").remove()
svg.append("path")
        .attr("class", "highlights")
        .attr("d", function (d) {
                y.domain([0,
        d.maxPrice]);
                return
        line(d.values.filter(e =>
        new
        Date(e.date).getFullYear()
        === selectedYear));
        });
} else {
        console.log('do nothing');
}
```

This can be viewed as a mathematical equation by eliminating parentheses by parentheses [4][5]. We have first located the year from what the user input, then turn it to a string number. Next, we use trim to remove any spacing before or after. If there is nothing or white space for the input, it will turn out to be doing nothing. If there is the year input, then the function will act upon the input. The previous highlights will be removed and a new highlight will be added based on function (d).

4 Experience Learning D3.js

D3.js provides us a flexible way in creating data visualizations. In this project, we find that it is especially powerful in modifying an existing script. During the process of modification, we can add or delete any elements we want (or do not want) in (or from) the original script without changing the whole.

In addition, because d3.js is a standardized library for data visualization, different users can realize different functions with the same standards. So when we try to realize certain functions in html, for instance, to add a search bar on the chart, we can learn from other people's examples and adapt into our own script.

All in all, D3.js can be very powerful, as the stock price example which we found online was monochromatic and non-interactive, after we modified it, it is now colourful and can react to the user. Also the script is intuitive and compatible, which we can easily apply with other JavaScript libraries.

Team Member Contributions

There are three members in our group. Der-Hsuan Tsou was in charge of understanding the original script and adding comments. Xinyi Ma made the first three modifications and Ngai Tik Chung made the last modification. The story telling part was performed by all the members based on group discussion.

References

[1] Matthew Hughes, "Study shows we're spending an insane amount of time online," *The Next Web*, https://thenextweb.com/tech/2019/01/31/study-shows-were-spending-an-insane-amount-of-time-online/. 2019.

[2] Mike Bostock, "Small Multiples," *Popular Blocks*, https://bl.ocks.org/mbostock/1157787/. 2016.

[3] L.C. Rost, "Your Friendly Guide to Colors in Data Visualization," *Datawapper*, https://blog.datawrapper.de/colorguide. 2018.

[4] Mike Bostock, "Color via Clipping," *Line chart with drawings*, https://bl.ocks.org/mbostock/4062844. May 2019.

[5] Mike Bostock, "Gradient Encoding," *Line chart with drawings*, https://observablehq.com/@d3/gradient-encoding. July 2019.