Final Project Report

VAST Challenge 2014

Mini- Challenge 2

Team Name

ARLM

Team Member

Alex Roeum

Rui Lin

Lauren Doutt

Malik Cameron

Introduction

Everyday people are tasked with solving problems. In this project, our team is not tasked with a real case but mimics one that we may encounter in the real world. We will look to apply the theories and technologies learned from class to solve problems. Visualization tools are important to understanding situations and can allow the viewer to depict patterns.

Background

The Virtual Analytics Sciences and Technology or VAST challenge is about helping the investigation of missing people of a company in a city. To solve this case, we focused on a key part of the investigation and used given data. Three different types of data were given for the various challenges. The information included the company's background information, the employee's daily activities, and some social media pertaining to events. In this project, we created two visualization tools to help us solve Mini Challenge 2. This mini challenge focuses on analyzing the employee's daily activities by representing the information in various visualization tools. The visualization tools helped us better understand the situation and the correlations that exist.

Design Goals

The designs we were originally going to take:

In this task, we were going to use the given debit, credit, loyalty card information, and GPS card tracking records. The two visualization choices for this subtask were an XY scatterplot and a parallel coordinates graph. An advantage of both graphs is they show the correlation relationship between two variables. Multi-dimensional data is very popular nowadays. Right now, parallel coordinates are often used to present multi-dimensional data. However, in data

analysis, people often combine parallel coordinates with other view graphs, such as traditional X-Y scatter plots. We chose the XY scatter plot because it gives a clearer picture of the relationship of two variables. It is user friendly and an easy to read graph. The inspiration behind this graph was Central South University's XY Mini Challenge 2.1 scatterplot.

Our second design was going to be related to Mini Challenge 2.2. In this subtask, we were going to look at employees' movements in regard to time. We chose the interactive Google Map for our second visualization tool. We were going to analyze the movement and location of employees and the location of unusual events. We were going to merge Card ID data onto the map as well for further insight. This design was going to be loosely based off of Central South University map in Mini Challenge 2.2. An interactive google map is very visually appealing while offering loads of information.

The designs we ended up taking:

The visualization designs we ended up making were an interactive histogram and a interactive streamgraph. These two designs were able to be accomplished with our level of programming abilities. The histogram is an interactive graph that plots the prices and names of the employees on the loyalty records. Histograms are a useful visualization tool because they display the relationship between two variables. They are easy to read, easy to understand, and can be of great use when analyzing a situation. By adding in an interactive characteristic to the graph, we can get the user interested and experimenting with the data. The histogram reacts to mouse clicks and re-sorts the data based on where the user clicks.

The other design we ended up doing is a streamgraph. The stream graph we designed plots one sale of the day (the first loyalty sale) for six stores for three consecutive days. The price variable

is on the y-axis and the time variable is on the X-axis. When the user scrolls over the stores on a certain day, the price variables are shown. Stream Graphs are useful for continuous data with time points. The data needs to be repetitive of the same points in order to properly utilize the streamgraph.

Data and Data Pre-processing

Data

The data sets include the tourist map, a geospatial map of Abila, Kronos, and raw data the company was able to obtain. The data is stored in a set of Excel documents. The data includes a document for the employee car assignments; which pairs an employee with a car number and identifies the employee. The data also includes credit card records on purchases employees made with their timestamps, costs, and places. The given data also includes documents of GPS locations with the longitude and latitude of places a car drove to and loyalty points an employee has from purchases.

These data files played a significant role in developing visual diagrams to track employee's daily lives. Implementing the first subtask was challenging. First, we spent time organizing the data into the correct format before we inserted the data into our graphs. We implemented various charts to present the results. Different charts helped us identify patterns and correlations within the data. Visual diagrams show the meaning of the immense amount of data that we are given in this project. As a team we implemented simple functions first and added other features one by one. This gave us a functional foundation in our code that was built upon for specifications.

Data Pre-processing

The data we used was the loyalty data. The loyalty card data for the employees and stores indicates the purchases made with loyalty points. Regarding data processing, we only needed to modify the file once to change it into a tsv file for the histogram. For the histogram graph, we changed the csv file into a tsv file through Excel. For the stream graph, we processed the data as a csv file, but manually modified it to narrow the scope of the data. For the histogram, we used the whole data set in the graph. The whole data set gave a bigger picture of the situation and of the loyalty information. For the stream graph, we selected one loyalty card purchase of the data for six stores for three days to narrow the activity and better understand some of the type of purchases the employees were making.

Visualization Design

Before we implemented the visualization design, we made a few objectives for our user interfaces. We wanted our graphs to be useful, easy to use, and intuitive for the average user. When we started programming, we made it a priority to make the data visually appealing. We made sure our graphs incorporated color and interaction to get the user's attention.

Implementation

Implementation was the most difficult section of the project to complete. After we preprocessed our data, it was time to start coding. We utilized resources from past assignments to help us make our graphs. The interactive histogram was an easier program to build than the streamgraph. To implement the histogram, we created our axises and dimensions first. The attributes from the loyalty.tsv file that were utilized were the first name and price values. The other variables were not graph. All of the points in the data were graphed on the histogram. We

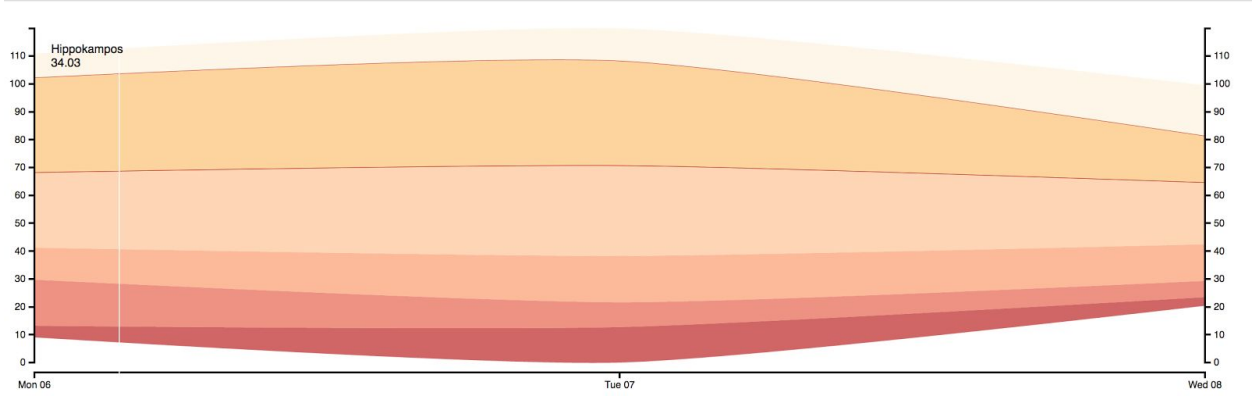
plotted the first names of the employees on the x-axis. We had to change the width of the graph to keep the x-labels from overlapping. We plotted the price values on the y-axis. We also implemented a sorting mouseclicked function. The user can click the x-axis and the y-axis to change the data to decreasing and increasing based on the variable.

The stream graph did not need any pre-data processing so we got to coding right away. The idea is based off of an assignment we have had in the past due to our familiarity with the program. For the stream graph, we selected one loyalty card purchase of the data for six stores for three days to get a better understanding of the loyalty information. By narrowing the scope, it made the graph more readable and understandable to the audience. The stream graph also has a mouseover function that displays the price for the store that is moused over.

Overall, the two visualization tools we implemented were an interactive histogram and an interactive stream graph. By basing the designs off of graphs we were familiar with, it allowed us to create functional, clean graphs. Since all of the members on our team were not widely exposed to d3 and javascript prior to the class, we utilized the knowledge we had to create these graphs.

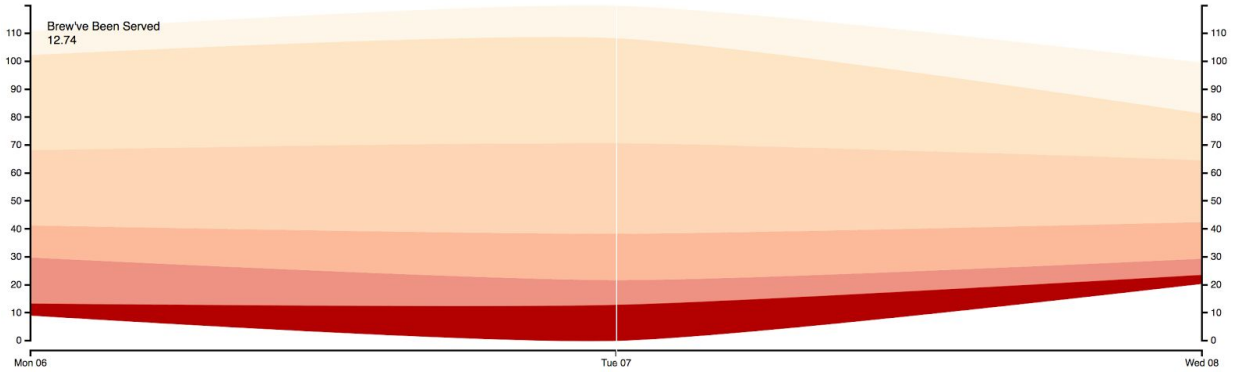
Results

Figure 1



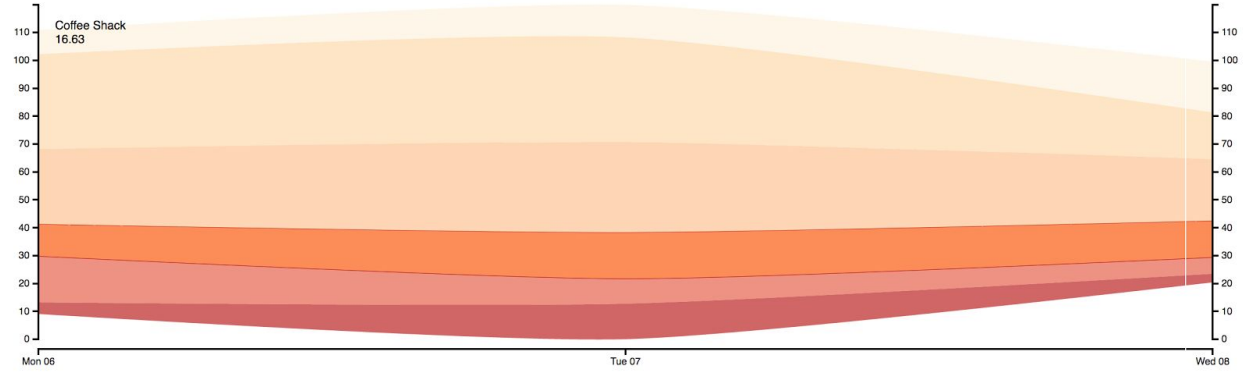
As you can see in Figure 1: Hippokampos' price for the first day is shown at 34.03

Figure 2



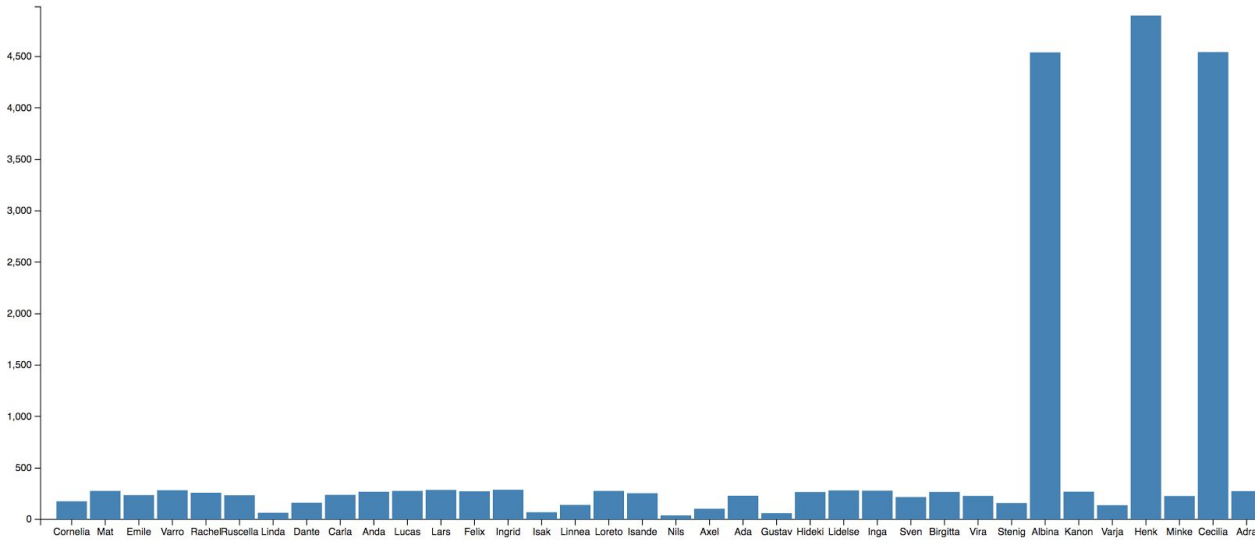
As you can see in Figure 2: Brew've Been Served price for the second day is 12.74

Figure 3:



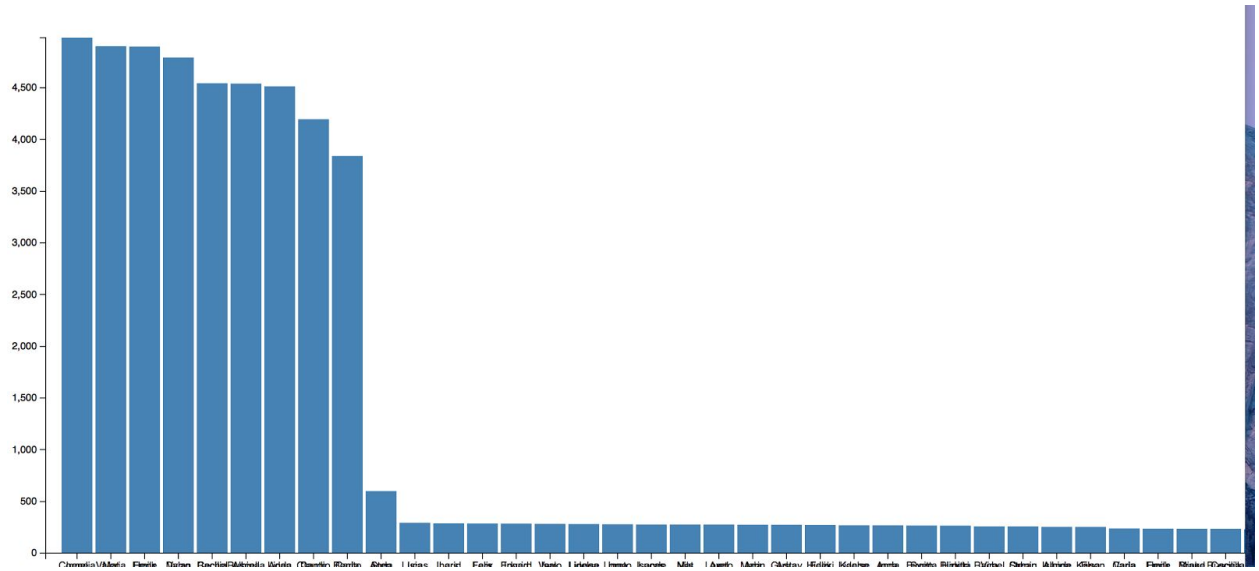
As you can see in Figure 3: Coffee Shack's price for the day is 16.63.

Figure 4:



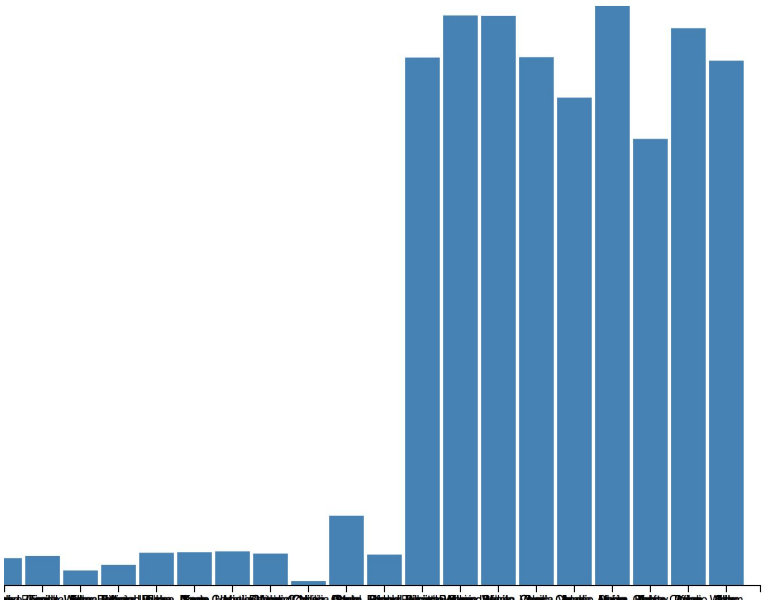
As you can see in Figure 4: A section of the original histogram shows the first name of employees on the x-axis and the prices on the y-axis.

Figure 5:



As you can see in Figure 5: a section of the histogram is descending relative to the y-axis (price).

Figure 6:



As you can see in Figure 6: a section of the histogram is ascending according to the y-axis (price).

Conclusion

What we learned

A few lessons we learned throughout the project include making sure the data is compatible with our designs. We initially were going to make an XY-scatterplot, but were having a lot of trouble with the formatting of the time variable. The d3.time.parse function was not working properly with our format. We learned the importance of having a backup plan ready. Having a backup plan is important when something goes wrong and you run into roadblocks. The histogram and stream graph were not our original visualization design plans, but they ended up working out for our project. Another lesson we learned is to establish descoping plans sooner.

A few more days plotted on our histogram would have been more ideal, but we were manually narrowing the scope on Excel and did not finish more than three days values.

What we would do better if we could do the project again

We would start the programming part of the project sooner. The programming is the hardest part of the project. If we would have started sooner, the end products may have been even more interactive. Our group had different levels of programming skills and it made it difficult at times to merge our styles and skills in the project. Also we would implement another graph to put next to our current graphs to give a more selection of visual displays

Technical aspect: coding and visualization design

Our group learned a lot regarding the technical aspect of the course. Many of us were learning javascript in depth for the first time. We discovered the d3 library is a really useful way to program interactive graphs within javascript and html. Our HTML skills were a bit rusty since some of us had not written in HTML since IST 210. Overall, learning about visualization designs and how the human brain processes images was very interesting. The Monday lectures were very informative and taught us a lot about how humans interact with images in their everyday life.

Team collaboration aspect

Our group learned how empowering working in a group can be. By using four brains instead of one, we were able to get more perspectives and different outlooks on the project. We each have a different background in programming and in technology. Some of our members were very analysis-driven while others were better at programming. Our team balanced each

other out and helped each other succeed. By learning to trust each other and to encourage each other, we found success.

Appendix

URLs to your project:

Histogram:

http://my.up.ist.psu.edu/lmd5596/projhistogram.html

Stream graph:

http://my.up.ist.psu.edu/lmd5596/projectstreamgraph.html

JavaScript codes of your project:

Javascript for stream graph:

```
;<!DOCTYPE html>
<meta charset="utf-8">
<style>

body {
  font: 10px sans-serif;
}
.chart {
  background: #fff;
}
p {
  font: 12px helvetica;
}

.axis path, .axis line {
  fill: none;
  stroke: #000;
  stroke-width: 2px;
  shape-rendering: crispEdges;
}

button {
```

```
  position: absolute;
  right: 50px;
  top: 10px;
}

</style>
<body>
<script src="//d3js.org/d3.v3.min.js"></script>

<div class="chart">
</div>

<script>

chart("please.csv", "orange");

var datearray = [];
var colorrange = [];


function chart(csvpath, color) {

if (color == "blue") {
  colorrange = ["#045A8D", "#2B8CBE", "#74A9CF", "#A6BDDB", "#D0D1E6", "#F1EEF6"];
}
else if (color == "pink") {
  colorrange = ["#980043", "#DD1C77", "#DF65B0", "#C994C7", "#D4B9DA", "#F1EEF6"];
}
else if (color == "orange") {
  colorrange = ["#B30000", "#E34A33", "#FC8D59", "#FDBB84", "#FDD49E", "#FEF0D9"];
}
strokecolor = colorrange[0];

var format = d3.time.format("%m/%d/%y");

var margin = {top: 20, right: 40, bottom: 30, left: 30};
var width = document.body.clientWidth - margin.left - margin.right;
var height = 400 - margin.top - margin.bottom;

var tooltip = d3.select("body")
    .append("div")
    .attr("class", "remove")
    .style("position", "absolute")
    .style("z-index", "20")
    .style("visibility", "hidden")
```

```
    .style("top", "30px")
    .style("left", "55px");

var x = d3.time.scale()
    .range([0, width]);

var y = d3.scale.linear()
    .range([height-10, 0]);

var z = d3.scale.ordinal()
    .range(colorrange);

var xAxis = d3.svg.axis()
    .scale(x)
    .orient("bottom")
    .ticks(d3.time.days, 1)

var yAxis = d3.svg.axis()
    .scale(y);

var yAxisr = d3.svg.axis()
    .scale(y);

var stack = d3.layout.stack()
    .offset("silhouette")
    .values(function(d) { return d.values;})
    .x(function(d) { return d.date; })
    .y(function(d) { return d.value; });

var nest = d3.nest()
    .key(function(d) { return d.key; });


var area = d3.svg.area()
    .interpolate("cardinal")
    .x(function(d) { return x(d.date); })
    .y0(function(d) { return y(d.y0); })
    .y1(function(d) { return y(d.y0 + d.y); });

var svg = d3.select(".chart").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

```
var graph  = d3.csv(csvpath, function(data) {

 data.forEach(function(d) {
   d.date = format.parse(d.date);
   d.value = +d.value;
 });

 var layers = stack(nest.entries(data));

 x.domain(d3.extent(data, function(d) { return d.date; }));
 y.domain([0, d3.max(data, function(d) { return d.y0 + d.y; })]);

 svg.selectAll(".layer")
     .data(layers)
   .enter().append("path")
     .attr("class", "layer")
     .attr("d", function(d) { return area(d.values); })
     .style("fill", function(d, i) { return z(i); });


 svg.append("g")
     .attr("class", "x axis")
     .attr("transform", "translate(0," + height + ")")
     .call(xAxis);

 svg.append("g")
     .attr("class", "y axis")
     .attr("transform", "translate(" + width + ", 0)")
     .call(yAxis.orient("right"));

 svg.append("g")
     .attr("class", "y axis")
     .call(yAxis.orient("left"));

 svg.selectAll(".layer")
   .attr("opacity", 1)
   .on("mouseover", function(d, i) {
     svg.selectAll(".layer").transition()
     .duration(250)
     .attr("opacity", function(d, j) {
       return j != i ? 0.6 : 1;
   })})

   .on("mousemove", function(d, i) {
     mousex = d3.mouse(this);
```

```
    mousex = mousex[0];
    var invertedx = x.invert(mousex);
    invertedx = invertedx.getYear() + invertedx.getDate();
    var selected = (d.values);
    for (var k = 0; k < selected.length; k++) {
      datearray[k] = selected[k].date
      datearray[k] = datearray[k].getYear() + datearray[k].getDate();
    }

    mousedate = datearray.indexOf(invertedx);
    pro = d.values[mousedate].value;

    d3.select(this)
    .classed("hover", true)
    .attr("stroke", strokecolor)
    .attr("stroke-width", "0.5px"),
    tooltip.html( "<p>" + d.key + "<br>" + pro + "</p>" ).style("visibility", "visible");

  })
  .on("mouseout", function(d, i) {
   svg.selectAll(".layer")
    .transition()
    .duration(250)
    .attr("opacity", "1");
    d3.select(this)
    .classed("hover", false)
    .attr("stroke-width", "0px"), tooltip.html( "<p>" + d.key + "<br>" + pro + "</p>" ).style("visibility", "hidden");
})

var vertical = d3.select(".chart")
      .append("div")
      .attr("class", "remove")
      .style("position", "absolute")
      .style("z-index", "19")
      .style("width", "1px")
      .style("height", "380px")
      .style("top", "10px")
      .style("bottom", "30px")
      .style("left", "0px")
      .style("background", "#fff");

d3.select(".chart")
    .on("mousemove", function(){
       mousex = d3.mouse(this);
       mousex = mousex[0] + 5;
```

```
        vertical.style("left", mousex + "px" )})
      .on("mouseover", function(){
        mousex = d3.mouse(this);
        mousex = mousex[0] + 5;
        vertical.style("left", mousex + "px")});
});
}
</script>
```

Javascript for histogram:

```
;//The margins, width, and height of the chart are set
var margin = {top: 50, right: 30, bottom: 30, left: 40},
   width = 1900 - margin.left - margin.right,
   height = 600 - margin.top - margin.bottom;

//Here the chart is made
var chart = d3.select(".chart")
   .attr("width", width + margin.left + margin.right)
   .attr("height", height + margin.top + margin.bottom)
   .append("g")
   .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

//This scales the x-axis
var x = d3.scale.ordinal()
   .rangeRoundBands([0, width], .1);

//This scales the y-axis
var y = d3.scale.linear()
   .range([height, 0]);

//Here the axis are defined and scaled
var xAxis = d3.svg.axis()
   .scale(x)
   .orient("bottom");

var yAxis = d3.svg.axis()
   .scale(y)
   .orient("left");

//here is the array the data from loyalty_data.tsv is stored in
var bardata = [ ];

//this variable sets the current state of the graph to ascending, and will help fluctuate the graph's movement
var currstate= 1;
//The data from loyalty_data.svg is read and the chart is made
d3.tsv("loyalty_data.tsv", type, function(error, data) {
```

```
bardata= data;
checkbardata= data;

  //X and Y's dimensions are set
  x.domain(data.map(function(d) { return d.FirstName; }));
  y.domain([0, d3.max(data, function(d) { return d.price; })]);

  //The x-axis is made and put in the bottom
  chart.append("g")
          .attr("class", "axis")
          .attr("transform", "translate(0," + height + ")")
          .call(xAxis);

  //The y-axis is made
  chart.append("g")
          .attr("class", "axis")
          .call(yAxis);

  //The bars are made
  chart.selectAll(".bar")
          .data(data)
          .enter().append("rect")
          .attr("class", "bar")
          .attr("x", function(d) { return x(d.FirstName); }) //this is the x-position
          .attr("y", function(d) { return y(d.price); })//this is the y-position
          .attr("height", function(d) { return height - y(d.price); })//Sets the sixe of the rectangle
          .attr("width", x.rangeBand()); //The width is set

});

//the mouse down function responds when the user clicks the mouse
d3.select("svg")
.on("mousedown", function(){


  d3.select(".xAxis").remove();

        //the variables of where the user clicks
        var x=d3.mouse(this)[0]
        var y =d3.mouse(this)[1]
        //tests to see if the user clicks the x axis or below

        if(x>=40 || y<=0){

        //console.log("x"+ d3.mouse(this)[0] +"y"+d3.mouse(this)[1]);
```

```
//checks to see if the current state is ascending
if (currstate == 1){

//change the bar data to descending in regards to the FirstName data
bardata.sort(function(a,b) {return d3.descending(a.FirstName, b.FirstName)});
//redraw the graph descending in regards to X
redrawXDescending();
//set the current state to 0
currstate=0;
}

//checks to see if the current state is 0 (descending)
else if(currstate == 0){

        //set the bardata to ascending in regards to the FirstName data
        bardata.sort(function(a,b) {return d3.ascending(a.FirstName, b.FirstName)});
        //redraw the graph ascending in regards to X
        redrawXAscending();
        //set the current state to 1 (so the program will descend next time)
        currstate=1;
}

}

//test to see if the user clicked the y axis or over
else if (x<=40 || y>=0){

        //tests to see if the currstate is 1
        if(currstate == 1){

        //sets the bar data into descending order in regards to price
        bardata.sort(function(a,b) {return d3.descending(a.price, b.price)});
        //redraws the graph descending in regards to y
        redrawYDescending();
        //sets the current state to 0 so the program ascends the data next time
        currstate=0;
        }
        //tests to see if the current state is 0
        else if(currstate==0){

        //sorts the bar data into ascending order in regards to price
        bardata.sort(function(a,b) {return d3.ascending(a.price, b.price)});
        //redraws the graph ascending in regards to Y
        redrawYAscending();
```

```
                        //sets the current state to 1
                        currstate=1;
                        }



            }
}

);
//this function redraws the graph in descending order in regards to X
function redrawXDescending(){
            //sets the domain for x and y
             x.domain(bardata.map(function(d) { return d.FirstName; }));
   y.domain([0, d3.max(bardata, function(d) { return d.price; })]);
   chart.append("g")
               .attr("class", "axis")
               .attr("transform", "translate(0," + height + ")")
               .call(xAxis);


   //the y-axis is made
   chart.append("g")
               .attr("class", "axis")
               .call(yAxis);

               //the bars are made
chart.selectAll("rect")
                        .data(bardata)
                        .transition()
                        .duration(1000)
                        .attr("y", function (d) { return y(d.price); })//this is the y-position
                        .attr("x", function (d) {return x(d.FirstName); })//this is the x-posiion
                        .attr("height", function(d) { return height - y(d.price); })//sets the size of the rectangle
            .attr("width", x.rangeBand());//the width is set
}

//this function redraws the graph in ascending order in regards to X
function redrawXAscending(){
            x.domain(bardata.map(function(d) { return d.FirstName; }));
   y.domain([0, d3.max(bardata, function(d) { return d.price; })]);

             //the x-axis is made and put at the bottom
   chart.append("g")
               .attr("class", "axis")
```

```
                .attr("transform", "translate(0," + height + ")")
                .call(xAxis);

    //the y-axis is made
    chart.append("g")
                .attr("class", "axis")
                .call(yAxis);

                //the bars are made
                chart.selectAll("rect")
                            .data(bardata)
                            .transition()
                            .duration(1000)
                            .attr("y", function (d) { return y(d.price); })//this is the y-position
                            .attr("x", function (d) {return x(d.FirstName); })//this is the x-position
                            .attr("height", function(d) { return height - y(d.price); })//Sets the size of the rectangle
                .attr("width", x.rangeBand());// the width is set


}

//this function redraws the graph in descending order in regards to Y
function redrawYDescending(){
                x.domain(bardata.map(function(d) { return d.FirstName; }));
    y.domain([0, d3.max(bardata, function(d) { return d.price; })]);

                //makes the x-axis and puts it at the bottom
    chart.append("g")
                .attr("class", "axis")
                .attr("transform", "translate(0," + height + ")")
                .call(xAxis);

    //makes the y-axis
    chart.append("g")
                .attr("class", "axis")
                .call(yAxis);

                //makes the bars
chart.selectAll("rect")
                            .data(bardata)
                            .transition()
                            .duration(1000)
                            .attr("y", function (d) { return y(d.price); }) // this is the y-position
                            .attr("x", function (d) {return x(d.FirstName); })// this is the x-position
                            .attr("height", function(d) { return height - y(d.price); })//sets the size of the rectangle
```

```
                .attr("width", x.rangeBand());//sets the width
}

//this function redraws the graph in ascending order in regards to Y
function redrawYAscending(){
                        x.domain(bardata.map(function(d) { return d.FirstName; }));
   y.domain([0, d3.max(bardata, function(d) { return d.price; })]);

              //makes the x-axis and puts it at the bottom
   chart.append("g")
                .attr("class", "axis")
                .attr("transform", "translate(0," + height + ")")
                .call(xAxis);

   //makes the y-axis
   chart.append("g")
                .attr("class", "axis")
                .call(yAxis);
//makes the bars
chart.selectAll("rect")
                        .data(bardata)
                        .transition()
                        .duration(1000)
                        .attr("y", function (d) { return y(d.price); })//this is the y-position
                        .attr("x", function (d) {return x(d.FirstName); })//this is the x-position
                        .attr("height", function(d) { return height - y(d.price); })//sets the size of the rectangle
                .attr("width", x.rangeBand());//sets the width



}

function type(d) {
   d.price = +d.price; // Coerce to Number
   return d;
}
```