Graphics and Displaying Data

Week 7 Presentation 1

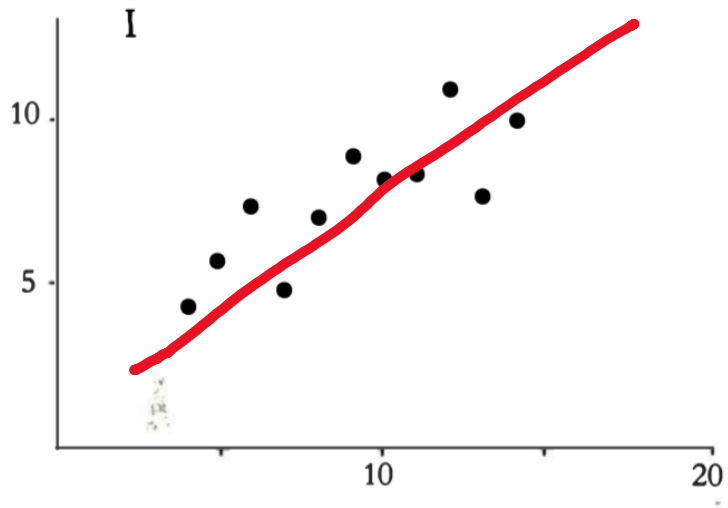
“Graphical excellence consists of the efficient communication of complex quantitative ideas.”

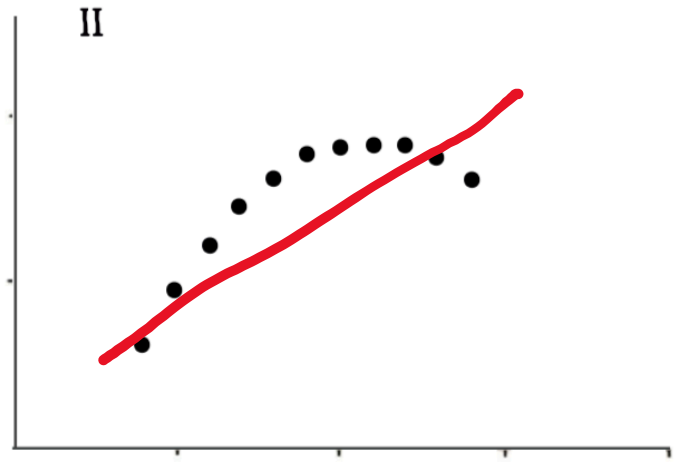
I		II		III		IV	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

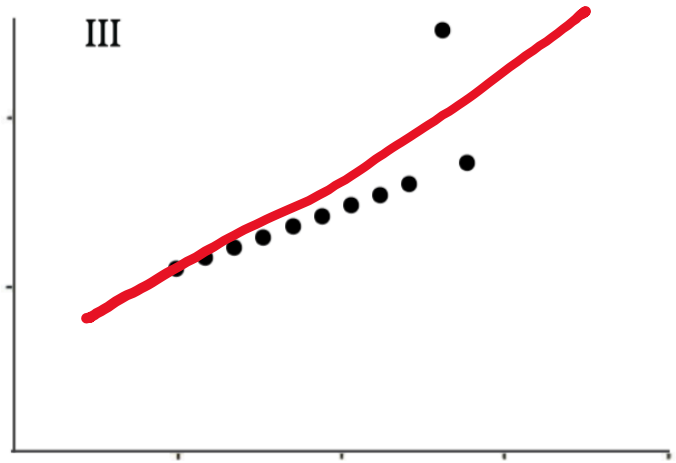
→

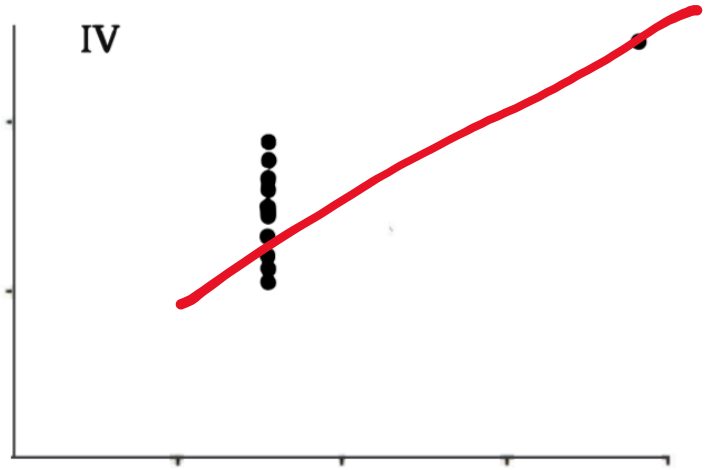
$N = 11$
mean of X's = 9.0
mean of Y's = 7.5
equation of regression line: $Y = 3 + 0.5X$
standard error of estimate of slope = 0.118
$t = 4.24$
sum of squares $X - \overline{X} = 110.0$
regression sum of squares = 27.50
residual sum of squares of $Y = 13.75$
correlation coefficient = .82
$r^2 = .67$

The Visual Display of Quantitative Information – Edward Tufte

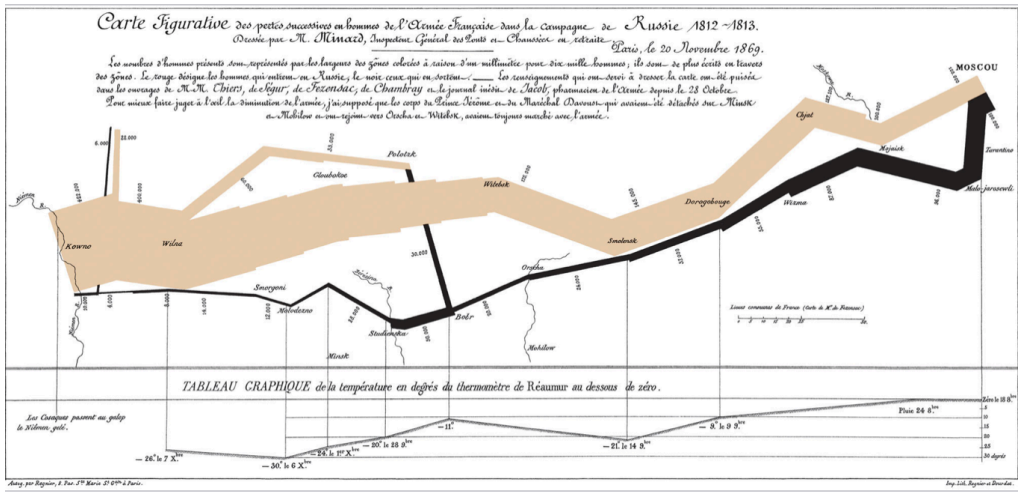




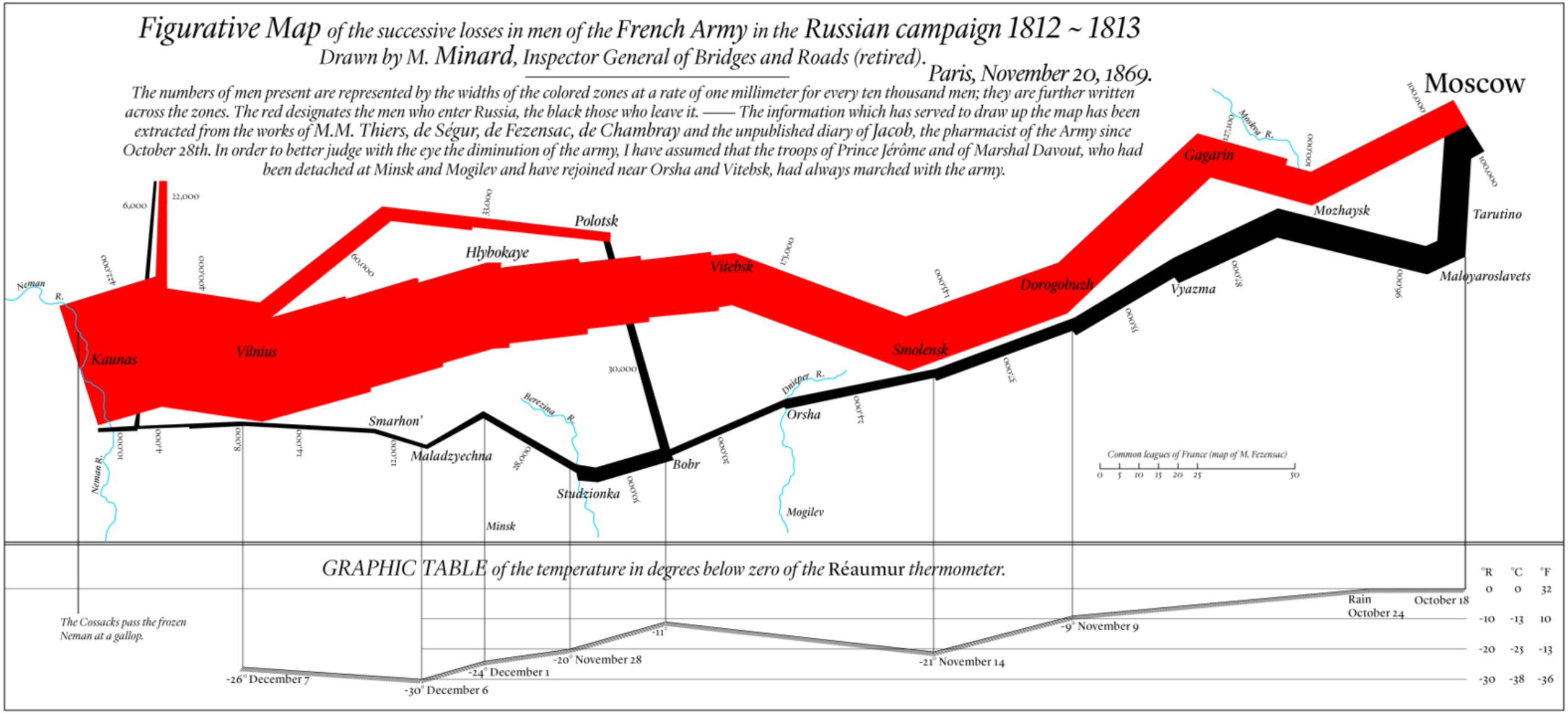




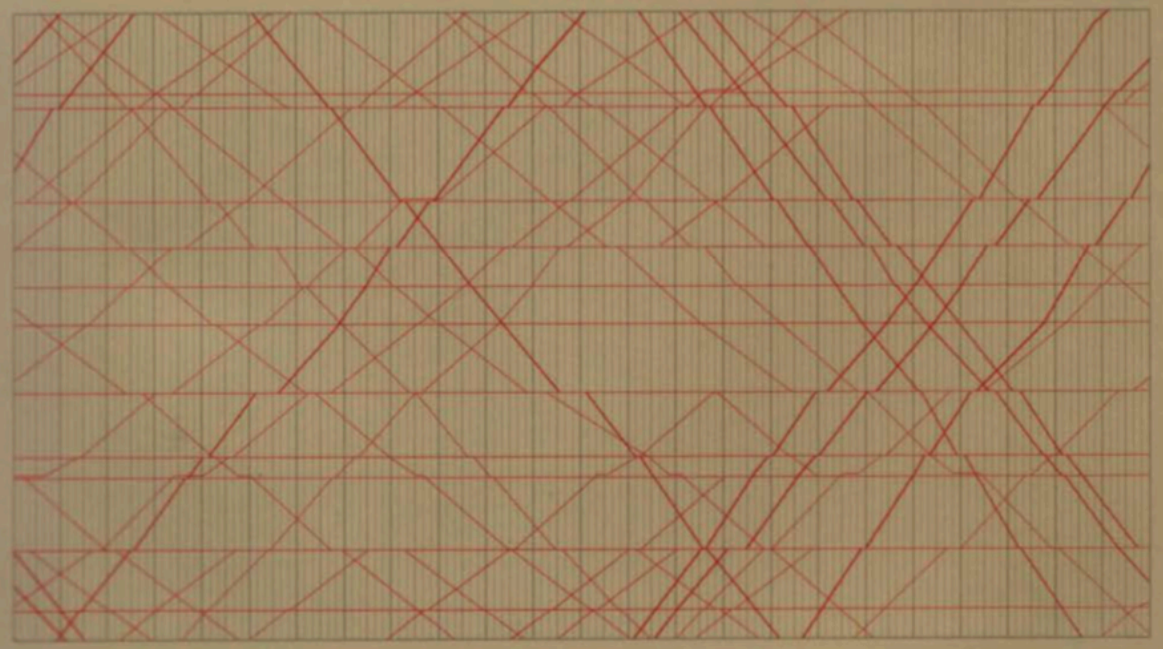
$$Y = 3 + 0.5X$$



“The best statistical graphic ever drawn“, is how statistician Edward **Tufte** described this



(This is a modern rendering in English – the original was in French)



SECOND EDITION

The Visual Display of Quantitative Information

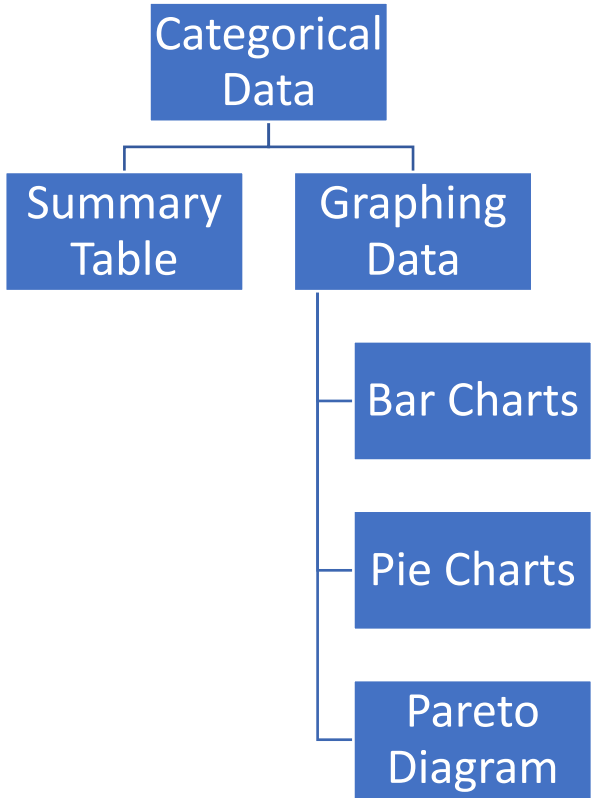
EDWARD R. TUFTE

Principles for Data Presentation

- Communicates complex ideas with clarity, precision and efficiency
- Gives the largest number of ideas in the most efficient manner
- Almost always uses several dimensions
- Requires telling the truth about the data

- ”Data Ink Ratio”

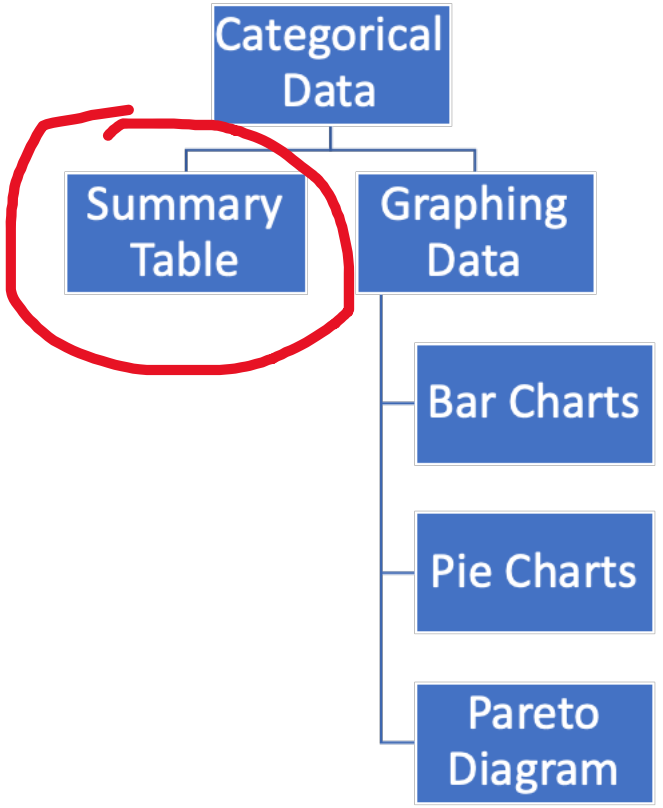
Tabulating and Graphing Categorical Data



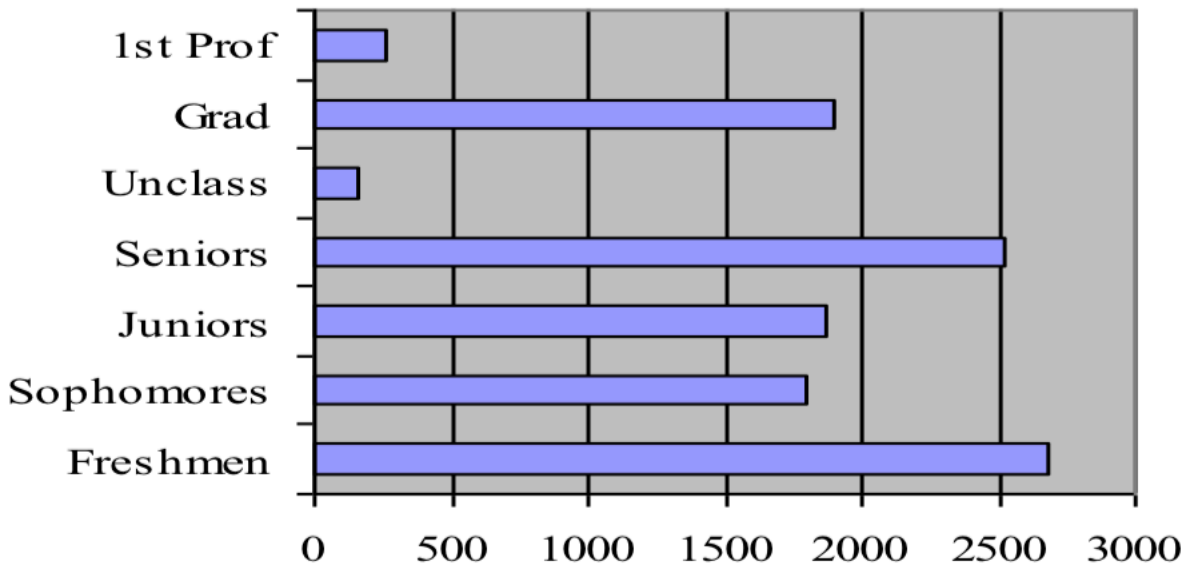
https://pdfs.semanticscholar.org/1d72/1668009bc65c899a6a631686427d987806d4.pdf?_ga=2.50376045.557616602.1585118213-775591735.1585118213

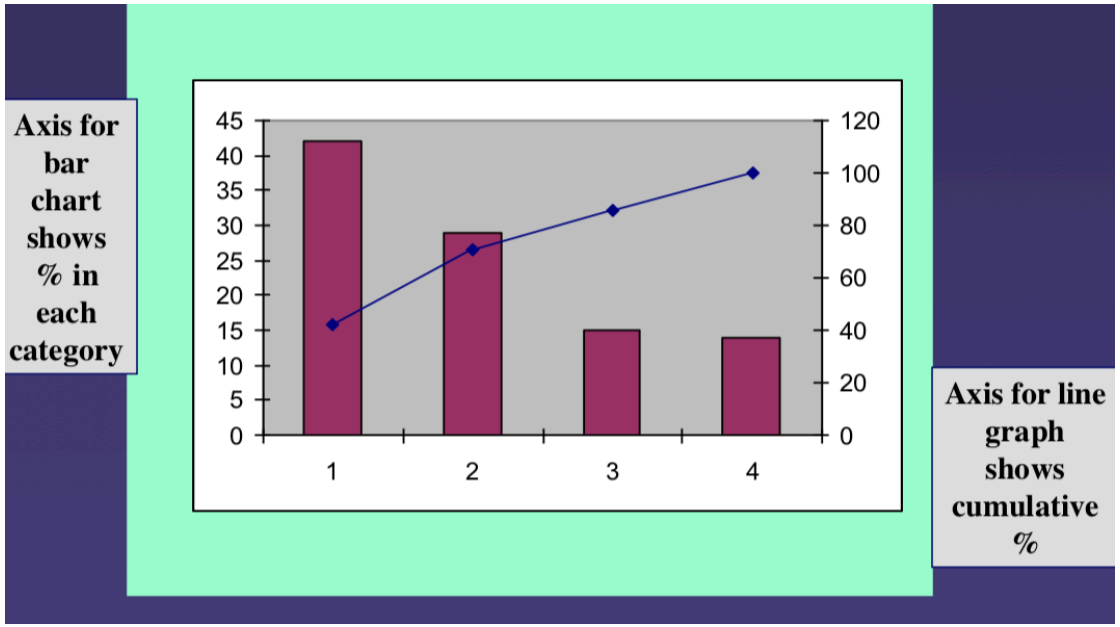
Summary Table of University Revenues

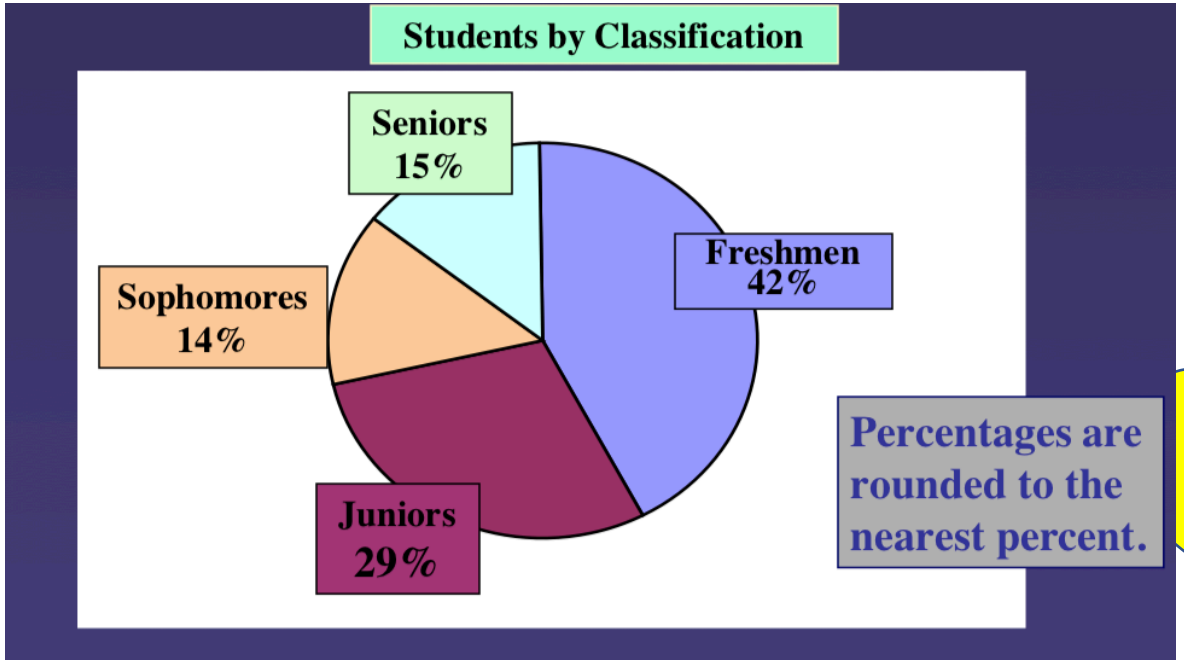
Revenue Category	Amount (in thousands $)	Percentage
Patient Services	46.5	42.27
Tution fees	32	29.09
Appropriations	15.5	14.09
Grants/Contracts	16	14.55
Total	**110**	**100**



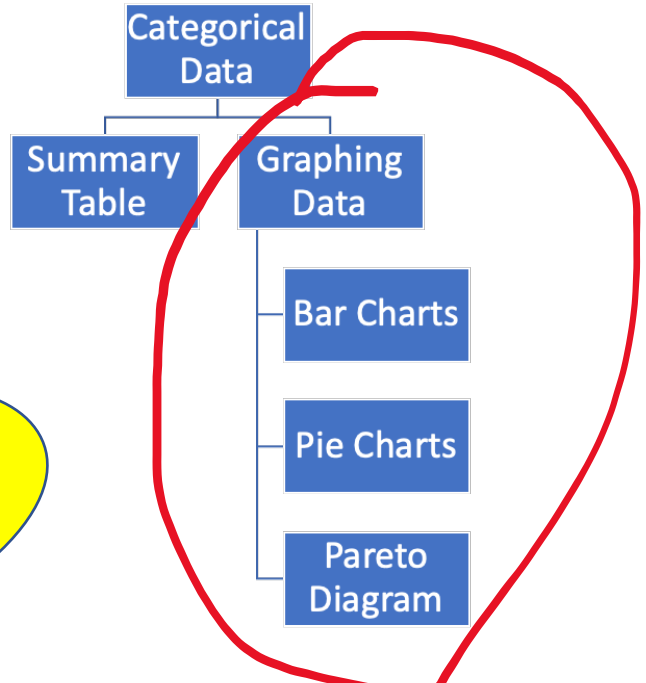
Variables are categorical

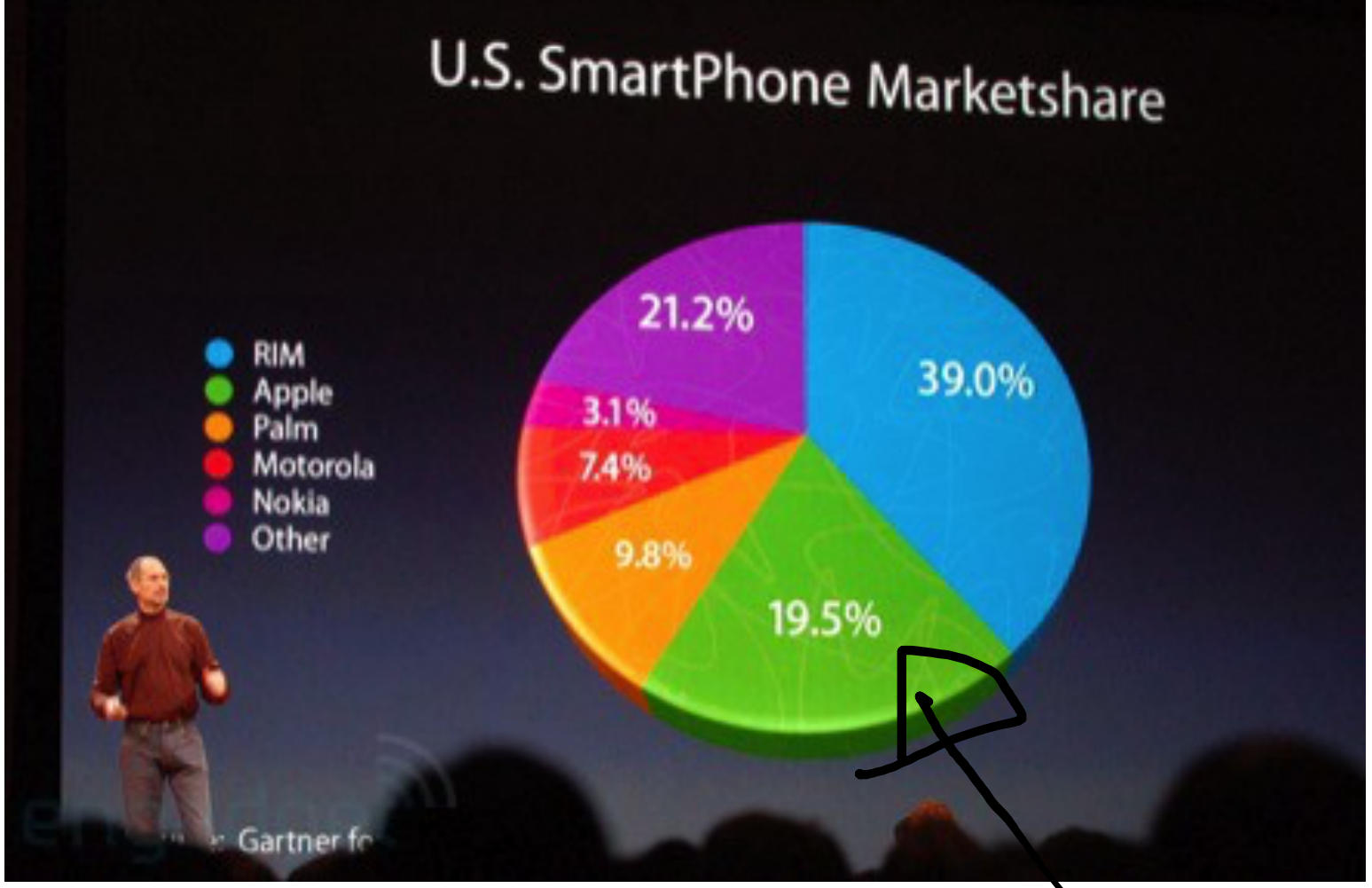






Pie charts are not liked by Tufte





Looks bigger because of perspective

Common Charting Errors

- **Using ‘chart junk’**
- **No relative basis**
  In comparing data
- **Batches**
- **Compressing the Vertical axis**
- **No zero point on the Vertical axis**

Chart Junk

Bad Presentation

Minimum Wage


1960: $1.00

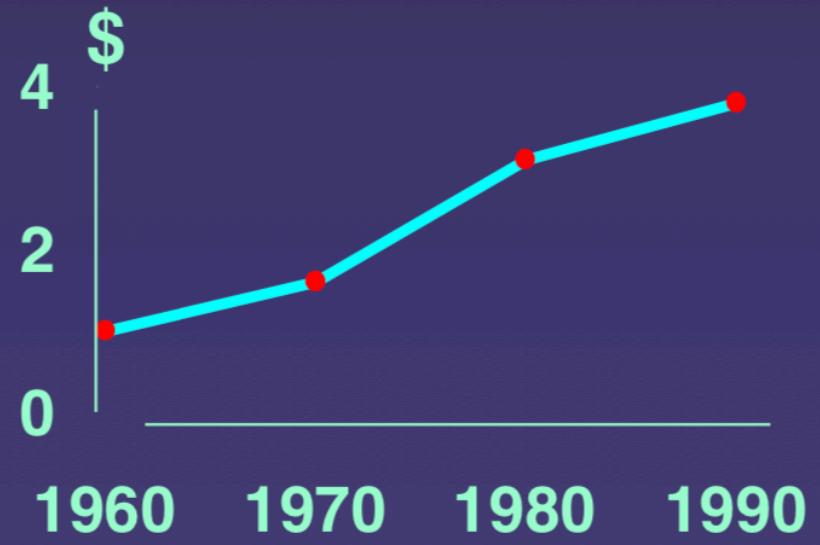

1970: $1.60

1980: $3.10

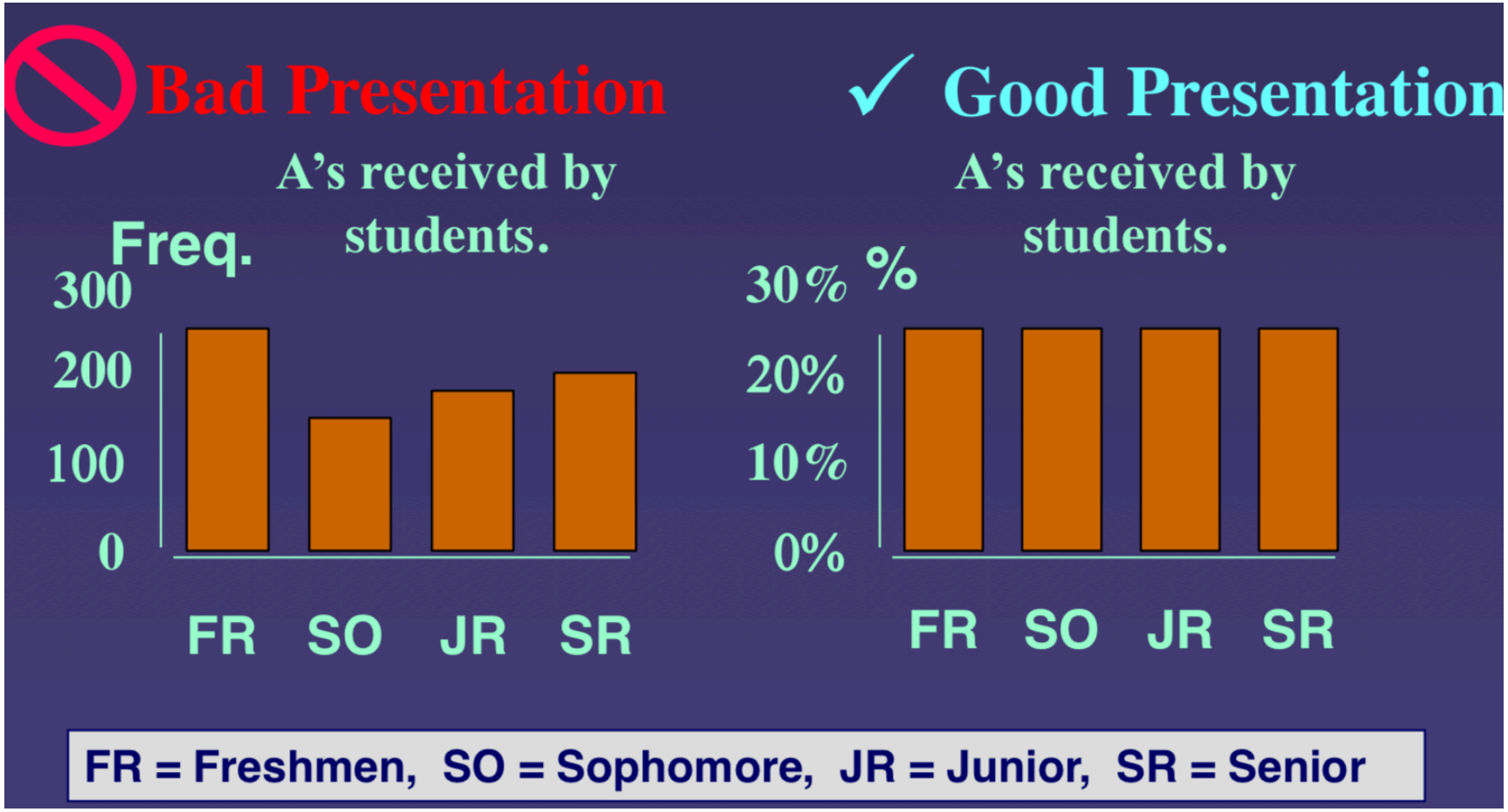
1990: $3.80

Good Presentation

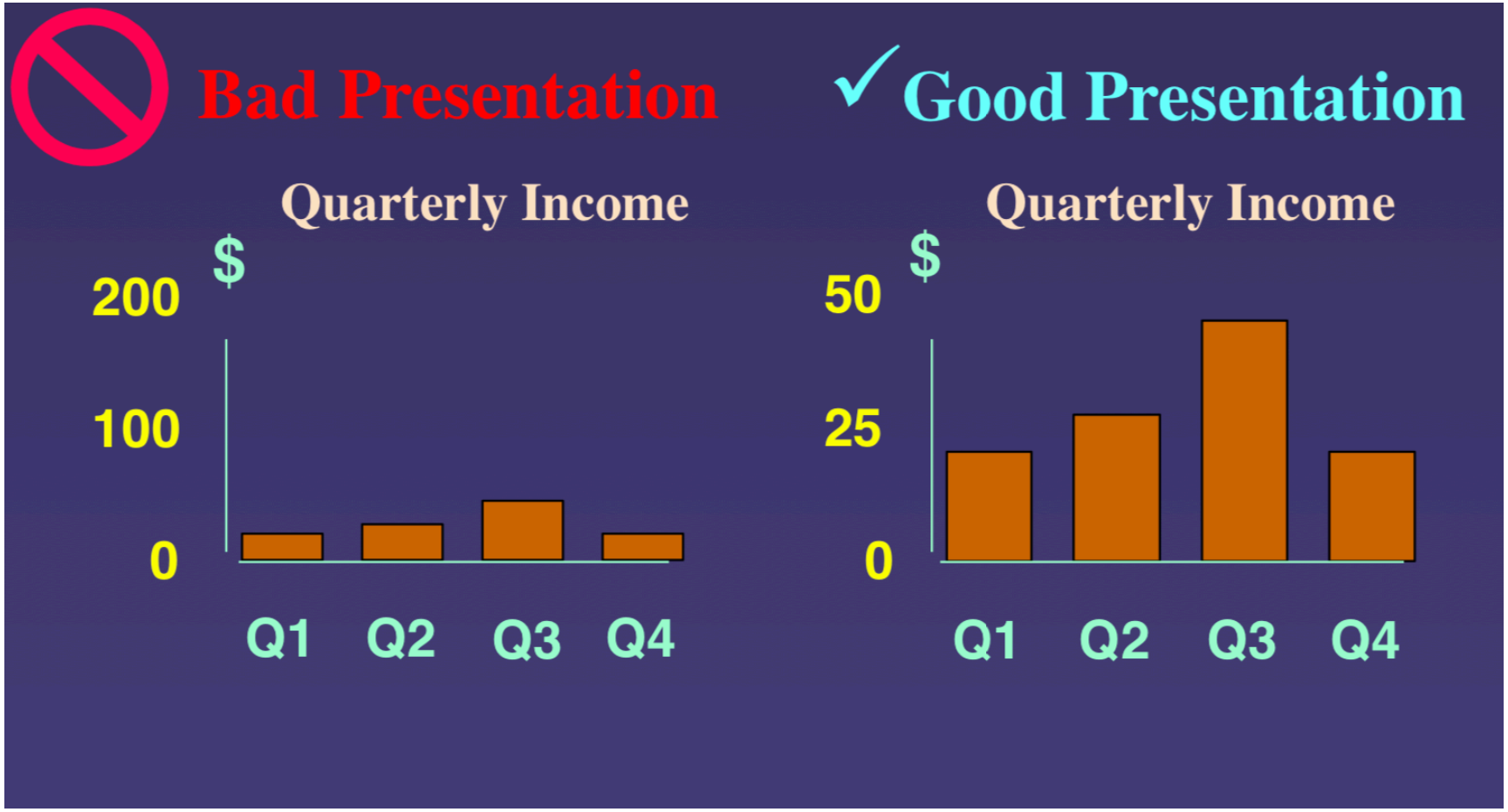
Minimum Wage

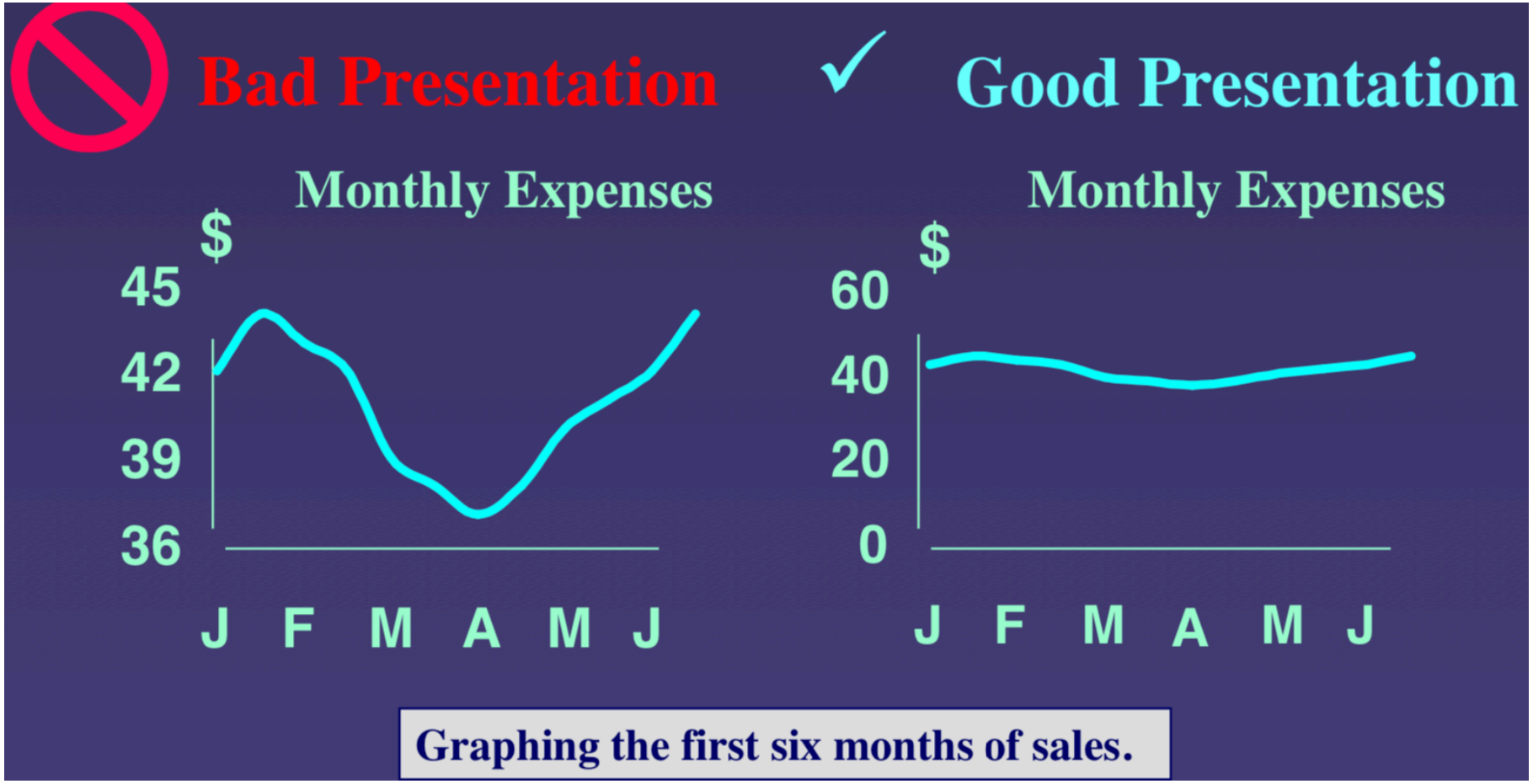


No relative basis

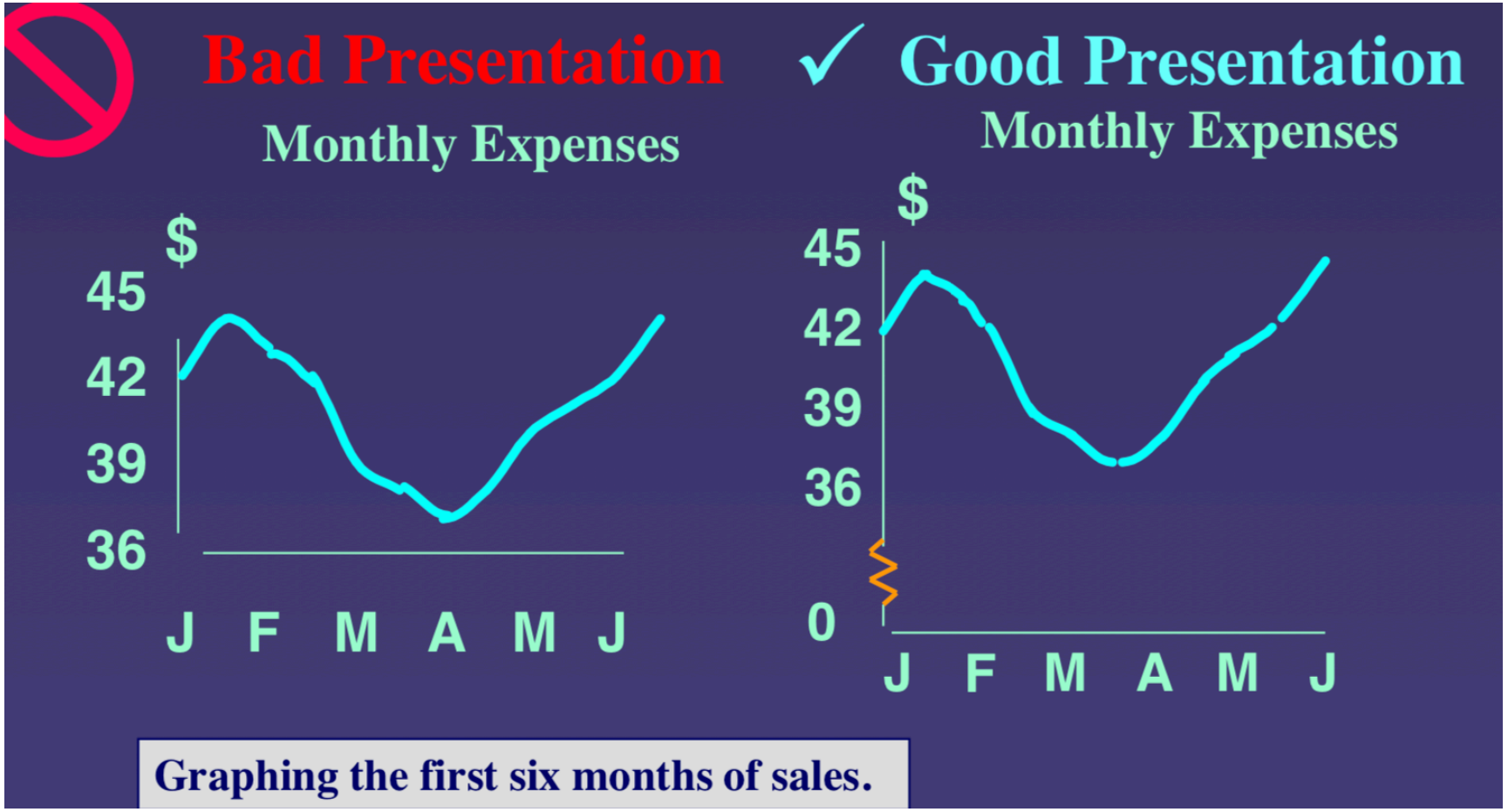


Compressing vertical axis





No zero point on vertical axis



No zero point on vertical axis

Online Tutorials on JavaFX:

JavaFX教程 @ 易百教程
https://www.yiibai.com/javafx

JavaFX – Charts @ oracle.com
https://docs.oracle.com/javafx/2/api/javafx/scene/chart/package-summary.html

JavaFX – Charts @ tutorialspoint.com
http://www.tutorialspoint.com/javafx/javafx_charts.htm

Visualizing Data with Plots

Data visualization is an important and exciting component in Scientific and Engineering Domains, especially in data science.

Data visualization should always take into consideration the audience, there are roughly three kinds of consumers of a visualization:
yourself, industry expert, everybody else

Visualizing Data for Yourself

The first is yourself, the all-knowing expert who is most likely iterating quickly on an analysis or algorithm development.

Your requirements are to see the data as plainly and quickly as possible. Things such as setting plot titles, axis labels, smoothing, legends, or date formatting might not be important, because you are intimately aware of what you are looking at.

In essence, we often plot data to get a quick overview of the data landscape, without concerning ourselves with how others will view it.

Visualizing Data for Industry Expert

The second consumer of data visualizations is the industry expert. After you have solved a data science problem and you think it's ready to share, it's essential to fully label the axis, put a meaningful, descriptive title on it, make sure any series of data are described by a legend, and ensure that the graphic you have created can mostly tell a story on its own.

Even if it's not visually stunning, your colleagues and peers will probably not be concerned with eye candy, but rather the message you are trying to convey.

In fact, it will be much easier to make a scientific evaluation on the merits of the work if the visualization is clear of graphical widgets and effects. Of course, this format is also essential for archiving your data. One month later, you will not remember what those axes are if you don't label them now!

Visualizing Data for Everybody

The third category of visualization consumer is everybody else. This is the time to get creative and artistic, because a careful choice of colors and styles can make good data seem great.

Be cautious, however, of the tremendous amount of time and effort you will spend preparing graphics at this level of consumer.

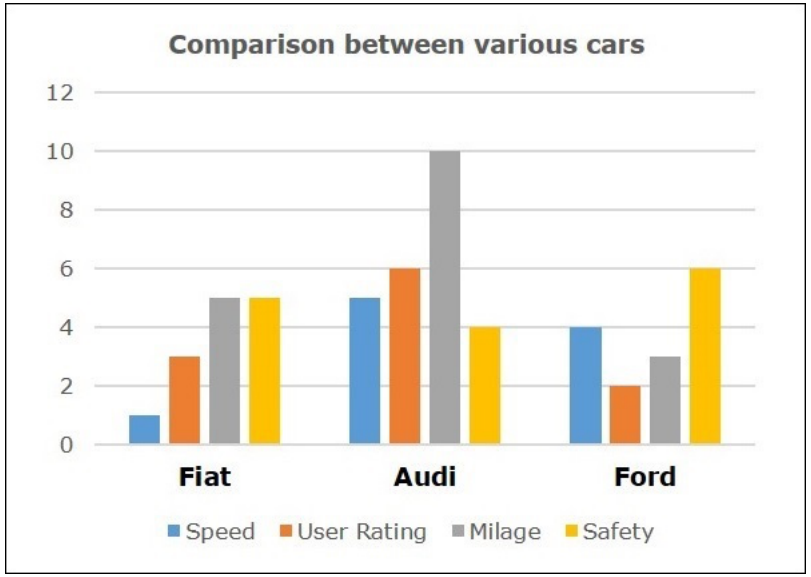
An added advantage of using JavaFX is the interactivity allowed via mouse options. This enables you to build a graphical application similar to many of the web-based dashboards you are accustomed to.

```
import javafx.scene.chart.*;
```

class PieChart

class XYChart
x can be
numeric or
a category
name

- AreaChart
- BarChart
- BubbleChart
- LineChart
- ScatterChart
- StackedAreaChart
- StackedBarChart



A bar chart

https://docs.oracle.com/javafx/2/api/javafx/scene/chart/package-summary.html

Package javafx.scene.chart

The JavaFX User Interface provides a set of chart components that are a very convenient way for data visualization.

See: Description

Class Summary

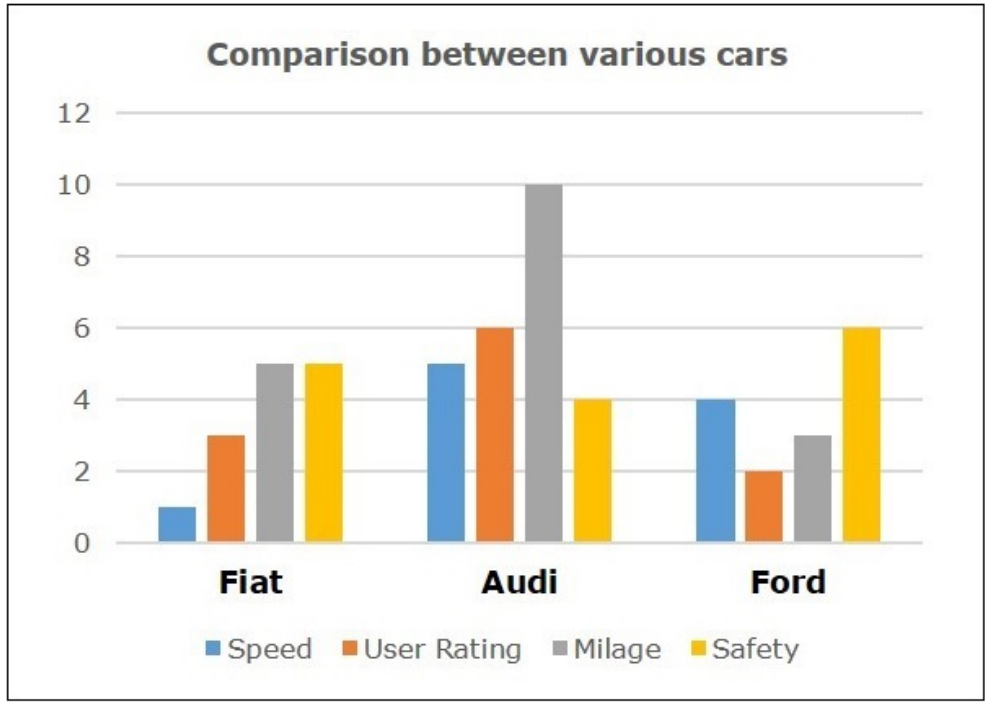
Class	Description
AreaChart<X,Y>	AreaChart - Plots the area between the line that connects the data points and the 0 line on the Y axis.
AreaChartBuilder<X,Y,B extends **AreaChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.AreaChart
Axis<T>	Base class for all axes in JavaFX that represents an axis drawn on a chart area.
Axis.TickMark<T>	TickMark represents the label text, its associated properties for each tick along the Axis.
AxisBuilder<T,B extends **AxisBuilder**<T,B>>	Builder class for javafx.scene.chart.Axis
BarChart<X,Y>	A chart that plots bars indicating data values for a category.
BarChartBuilder<X,Y,B extends **BarChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.BarChart
BubbleChart<X,Y>	Chart type that plots bubbles for the data points in a series.
BubbleChartBuilder<X,Y,B extends **BubbleChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.BubbleChart
CategoryAxis	A axis implementation that will works on string categories where each value as a unique category(tick mark) along the axis.
CategoryAxisBuilder	Builder class for javafx.scene.chart.CategoryAxis
Chart	Base class for all charts.
ChartBuilder<B extends **ChartBuilder**<B>>	Builder class for javafx.scene.chart.Chart

LineChart<X,Y>	Line Chart plots a line connecting the data points in a series.
LineChartBuilder<X,Y,B extends **LineChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.LineChart
NumberAxis	A axis class that plots a range of numbers with major tick marks every "tickUnit".
NumberAxis.DefaultFormatter	Default number formatter for NumberAxis, this stays in sync with auto-ranging and formats values appropriately.
NumberAxisBuilder	Builder class for javafx.scene.chart.NumberAxis
PieChart	Displays a PieChart.
PieChart.Data	PieChart Data Item, represents one slice in the PieChart
PieChartBuilder<B extends **PieChartBuilder**<B>>	Builder class for javafx.scene.chart.PieChart
ScatterChart<X,Y>	Chart type that plots symbols for the data points in a series.
ScatterChartBuilder<X,Y,B extends **ScatterChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.ScatterChart
StackedAreaChart<X,Y>	StackedAreaChart is a variation of `AreaChart` that displays trends of the contribution of each value.
StackedAreaChartBuilder<X,Y,B extends **StackedAreaChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.StackedAreaChart
StackedBarChart<X,Y>	StackedBarChart is a variation of `BarChart` that plots bars indicating data values for a category.
StackedBarChartBuilder<X,Y,B extends **StackedBarChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.StackedBarChart
ValueAxis<T extends java.lang.Number>	A axis who's data is defined as Numbers.
ValueAxisBuilder<T extends java.lang.Number,B extends **ValueAxisBuilder**<T,B>>	Builder class for javafx.scene.chart.ValueAxis
XYChart<X,Y>	Chart base class for all 2 axis charts.
XYChart.Data<X,Y>	A single data item with data for 2 axis charts
XYChart.Series<X,Y>	A named series of data items
XYChartBuilder<X,Y,B extends **XYChartBuilder**<X,Y,B>>	Builder class for javafx.scene.chart.XYChart

JavaFX - Bar Chart

A bar chart is used to represent grouped data using rectangular bars. The length of these bars depicts the values. The bars in the bar chart can be plotted vertically or horizontally.

Following is a bar chart, comparing various car brands.



In JavaFX, a Bar chart is represented by a class named **BarChart**. This class belongs to the package **javafx.scene.chart**. By instantiating this class, you can create an BarChart node in JavaFX.

Example

The following example depicts various car statistics with the help of a bar chart. Following is a list of car brands along with their different characteristics, which we will show using a bar chart –

Car	Speed	User Rating	Millage	Safety
Fiat	1.0	3.0	5.0	5.0
Audi	5.0	6.0	10.0	4.0
Ford	4.0	2.0	3.0	6.0

Following is a Java program which generates a bar chart, depicting the above data using JavaFX.

Save this code in a file with the name **BarChartExample.java**.

```java
//BarChartExample

import java.util.Arrays;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.stage.Stage;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;

public class BarChartExample extends Application {
    @Override
    public void start (Stage stage) {
        //Defining the axes
        CategoryAxis xAxis = new CategoryAxis();
        xAxis.setCategories(FXCollections.<String>observableArrayList(
            Arrays.asList("Speed", "User rating", "Milage", "Safety")));
        xAxis.setLabel("category");

        NumberAxis yAxis = new NumberAxis();
        yAxis.setLabel("score");
```

Define the X and Y axis of the bar chart and set labels to them. In our example, X axis represents the category of comparison and the y axis represents the score.

The constructor of this class, pass the objects representing the X and Y axis.

Instantiate the **XYChart.Series** class and add the data (a series of, x and y coordinates) to the Observable list of this class.

```
26      //Creating the Bar chart
27      BarChart<String, Number> barChart = new BarChart<>(xAxis, yAxis);
28      barChart.setTitle("Comparison between various cars");
29
30      //Prepare XYChart.Series objects by setting data
31      XYChart.Series<String, Number> series1 = new XYChart.Series<>();
32      series1.setName("Fiat");
33      series1.getData().add(new XYChart.Data<>("Speed", 1.0));
34      series1.getData().add(new XYChart.Data<>("User rating", 3.0));
35      series1.getData().add(new XYChart.Data<>("Milage", 5.0));
36      series1.getData().add(new XYChart.Data<>("Safety", 5.0));
37
38      XYChart.Series<String, Number> series2 = new XYChart.Series<>();
39      series2.setName("Audi");
40      series2.getData().add(new XYChart.Data<>("Speed", 5.0));
41      series2.getData().add(new XYChart.Data<>("User rating", 6.0));
42      series2.getData().add(new XYChart.Data<>("Milage", 10.0));
43      series2.getData().add(new XYChart.Data<>("Safety", 4.0));
44
45      XYChart.Series<String, Number> series3 = new XYChart.Series<>();
46      series3.setName("Ford");
47      series3.getData().add(new XYChart.Data<>("Speed", 4.0));
48      series3.getData().add(new XYChart.Data<>("User rating", 2.0));
49      series3.getData().add(new XYChart.Data<>("Milage", 3.0));
50      series3.getData().add(new XYChart.Data<>("Safety", 6.0));
51
```

```
52          //Setting the data to bar chart
53          barChart.getData().addAll(series1, series2, series3);
54
55          //Creating a Group object
56          Group root = new Group(barChart);
57
58          //Creating a scene object
59          Scene scene = new Scene(root, 600, 400);
60
61          //Setting title to the Stage
62          stage.setTitle("Bar Chart");
63
64          //Adding scene to the stage
65          stage.setScene(scene);
66
67          //Displaying the contents of the stage
68          stage.show();
69      }
70
71      public static void main (String args[]) {
72          launch(args);
73      }
74  }
```

Add the data series prepared to the bar chart.

Create group object (extends **javafx.scene)** Pass the BarChart (node) object as a parameter to the constructor of the Group class.

Create a Scene with width and height

You can set the title to the stage using the **setTitle()** method of the **Stage** class.

You can add a Scene object to the stage using **setScene()** and then show the contents

Finally, aunch the JavaFX application by calling the static method **launch()** of the **Application** class from the main method.

$ javac BarChartExample.java
$ java BarChartExample