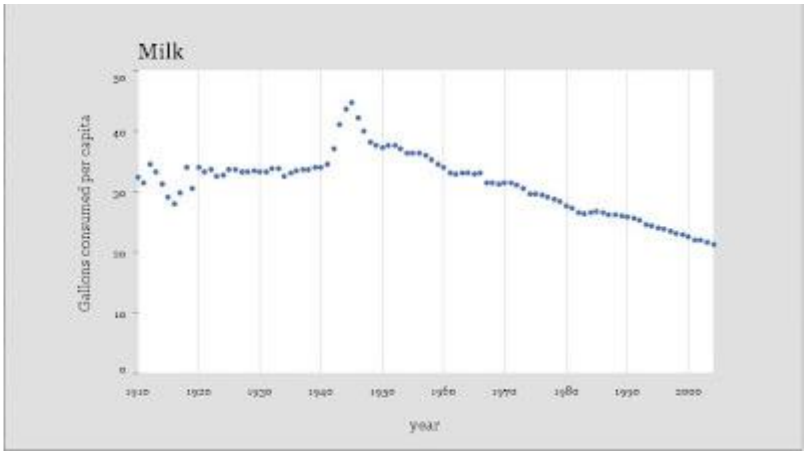
Time Series Exploration

a)

Initially, I changed font labels of x and y axes to "Georgia" font, also I changed text title font to "Verdana", set font size of axes labels to 14 and font size of title to 20. Then I rotated y-axis label by -PI/2.0 degree and removed those "\n" in its label. so the diagram became to :



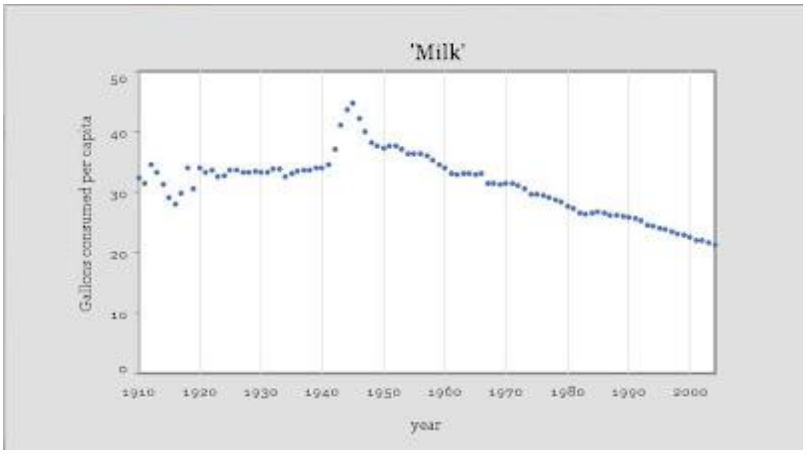
Then I noticed year labels and volume labels are too small to be seen clearly, so I increased their font size to 11. Also I put border for the plot area to be more attractive, for putting border, I removed line noStroke(); in draw function and instead wrote :

```
stroke(115,115,115);

strokeWeight(2);
```

Then I changed the location of chart title from left to center, for this change, I changed text line in drawTitle() function as follow:

```
text("'"+title+"'", (plotX1+plotX2-30)/2, plotY1 -10);
```

So the chart changed to this one:



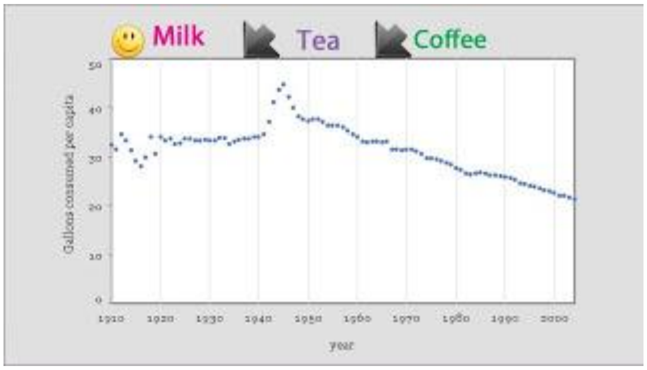
b)
In this part, I enriched keypressed function to allows the user to select favorite representation, also I have changed font of highlight tags (when mouse roll over dots) to "Georgia" font.
So my complete code for this part is :

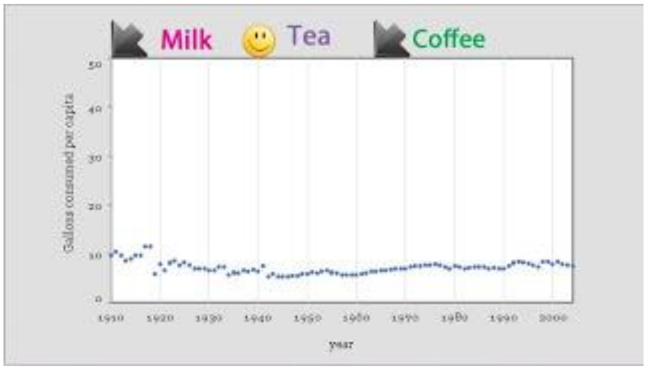
```
FloatTable data;
float dataMin, dataMax;

float plotX1, plotY1;
float plotX2, plotY2;
float labelX, labelY;

int currentColumn = 0;
int columnCount, rowCount;

int yearMin, yearMax;
int[] years;

int yearInterval = 10;
int volumeInterval = 10;
int volumeIntervalMinor = 5;

PFont plotFont;

void setup(){
  size(720, 405);

  data = new FloatTable("milk-tea-coffee.tsv");
  rowCount = data.getRowCount();
  columnCount = data.getColumnCount();

  years = int(data.getRowNames());
  yearMin = years[0];
  yearMax = years[years.length-1];
  dataMin = 0;
  dataMax = ceil(data.getTableMax() / volumeInterval) * volumeInterval;

  plotX1 = 120; //50
  plotX2 = width - 80; //-plotX1
  labelX = 50;
  plotY1 = 60;
  plotY2 = height - 70; //- plotY1
  labelY = height - 25;

//  plotFont = createFont("SansSerif", 20);
//  println(PFont.list());
  plotFont = createFont("Verdana", 20);
  textFont(plotFont);

  smooth();
}

void draw(){
  background(224);
  fill(255);
  rectMode(CORNERS);
//  noStroke();
  stroke(115,115,115);
  strokeWeight(2);
  rect(plotX1, plotY1, plotX2, plotY2);

  //draw the title of the current plot
  drawTitle();
  drawAxisLabels();

  drawYearLabels();
  drawVolumeLabels();

  stroke(#5679c1);
  strokeWeight(5);
  drawDataPoints(currentColumn);
}

void drawTitle(){
  fill(0);
  textSize(20);
  textAlign(LEFT);
  String title = data.getColumnName(currentColumn);
  text("'"+title+"'", (plotX1+plotX2-30)/2, plotY1 -10);
}

void drawAxisLabels(){
  fill(0);
  textSize(14);
  textLeading(15);
  textAlign(CENTER, CENTER);
  plotFont = createFont("Georgia", 14);
  textFont(plotFont);
  pushMatrix();
//  translate(10,10);
  rotate(-PI/2.0);
```

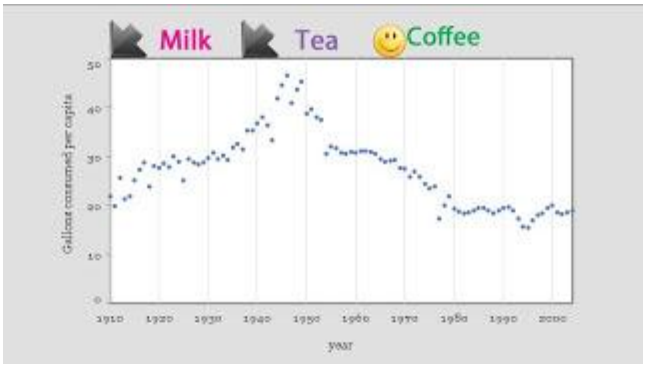
```
//  text("Gallons\nconsumed\nper capita", labelX, (plotY1+plotY2)/2);
  text("Gallons consumed per capita", -190, 70);
  popMatrix();
  text("year", labelY, (plotX1+plotX2)/2);
}

void drawYearLabels(){
  fill(0);
  textSize(11);
  textAlign(CENTER, TOP);

  //use thin gray lines to draw the grid
  stroke(224);
  strokeWeight(1);

  for (int row = 0; row &lt; rowCount; row++){
      if (years[row] % yearInterval == 0){
        float x = map(years[row], yearMin, yearMax, plotX1, plotX2);
        text(years[row], x , plotY2+10);
        line(x, plotY1, x, plotY2);
      }
   }
}

void drawVolumeLabels(){
  fill(0);
  textSize(11);
  stroke(128);
  strokeWeight(1);

  for (float v = dataMin; v &lt;= dataMax; v+= volumeInterval){
    if (v % volumeIntervalMinor == 0){
      float y = map(v, dataMin, dataMax, plotY2, plotY1);
      if (v % volumeInterval == 0){
        if (v == dataMin){
          textAlign(RIGHT);
        }else if (v == dataMax){
          textAlign(RIGHT, TOP);
        }else{
          textAlign(RIGHT, CENTER);
        }
        text(floor(v), plotX1-10, y);
        line(plotX1-4, y , plotX1, y);
      }else{
        //line(plotX1-2, y , plotX1, y);
      }
    }
  }
}


//draw data as a series of points
void drawDataPoints(int col){
  rowCount = data.getRowCount();
  for (int row = 0; row &lt; rowCount; row++){
    if (data.isValid(row,col)){
      float value = data.getFloat(row, col);
      float x = map(years[row], yearMin, yearMax, plotX1, plotX2);
      float y = map(value, dataMin, dataMax, plotY2, plotY1);
      point(x,y);
    }
  }
}

void keyPressed(){
  if (key == '['){
    currentColumn --;
    if (currentColumn &lt; 0){
      currentColumn = columnCount - 1;
    }
  }else if (key == ']'){
      currentColumn ++;
      if (currentColumn == columnCount){
        currentColumn = 0;
      }
    }
}
```

I think the clearest way to examine a regular time series is with a line chart. Drawing data as separate points makes it difficult to be traced by eyes, because data points might be too dispersed along vertical axis, so in this case for tracing them, user's eyes should jump from one point to another... and it is not user friendly at all. Bar chart and area chart are to clumsy, fill most of the screen's plot, while we need attention only on rise and falls of data. Furthermore, bar chart has limited applicability, they can be used only in discrete cases.
Among all these visualization, I prefer line chart, they are concise, showing rise and falls of data, can be applied in both discrete and continuous cases, allows user to trace change of data by his eyes easily. Also by using line chart we can show multiple time series data in one plot, and compare them easily with each other. (In case of using bar chart and area chart, It is almost impossible to show more than one diagram in one plot)

c)
When I implemented code of this part, my diagrams changed to ones below:





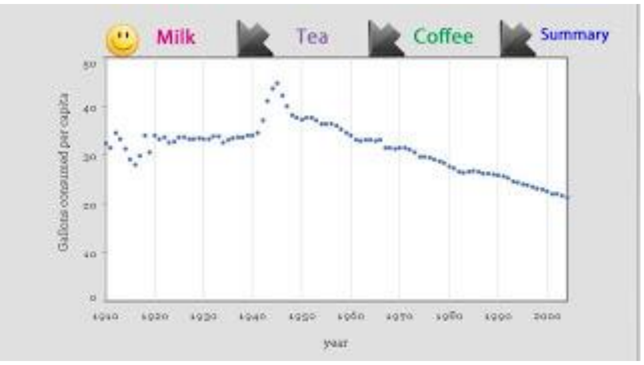


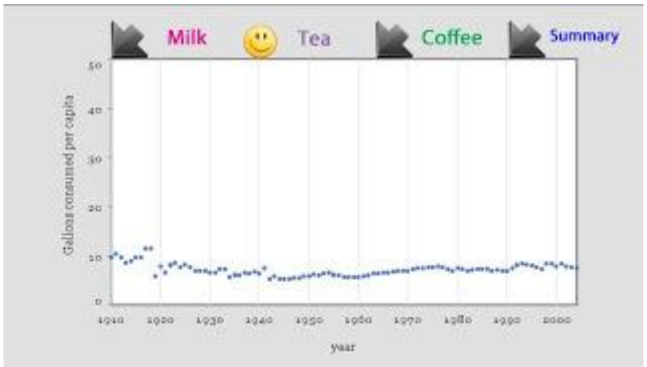
As we can see, there are three image tabs, each selected tab is shown by an yellow face, unselected ones are in gray arrow shape.

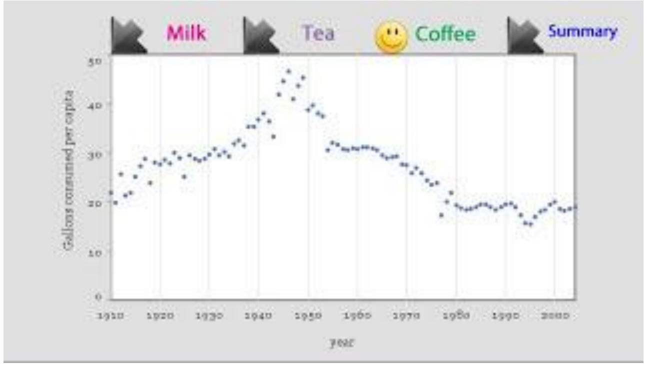
Although Integration provide an effective way of demonstrating data, I think it is not well equipped for dealing with time series data. I expect an efficient time series tool to have some data analysis functionality such as having an auto-correlation function, performing fourrier transform to investigate series in frequency domain, use of a filter to remove unwanted noise, etc.
Furthermore, I think processing can not deal with large data sets, we see that there is some delay in loading milk-tea-coffee data set (which is an small one), this delay increased when we used integrator class. so that this tool is not much fast that can be employed for large time series.

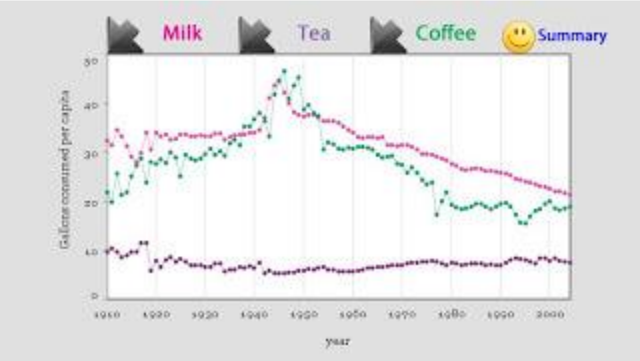
Last point is that processing is not user friendly. If I wanted to change some thing in it, I would make it more user friendly, adding some option for analyzing data.
In processing you have to implement a code for every thing you want to achieve, every diagram or analysis. so in case of working with a big data set, you must to implement a lot of code, this not only is sophisticated but also is tedious. At least programming environment of processing should be enriched via some auto complete, suggestions for using functions, etc.

d)
After implementing codes of this part, my diagrams changed to :









As you can see in summary tab, all 3 diagrams are shown, each in its tab color. via this tab, we can have a single view of 3 time series and therefore we are able to compare them easily.
in summary tab, all diagrams are shown in dot forms plus they are connected via a narrow line. as I said before, using line chart provides us best possibility for comparison, we can not compare them when they are drawn in bar chart or area chart.