Lecture 5-Time Series (Cont'd)

By Dr. Shaheera Rashwan

Choosing a Proper Representation (Represent and Refine)

- To represent a time series, we want a simple line with no fill, so we'll use the nofill( ) form of the shape.
- The following method is a variation of drawPoints( ) that draws the data with beginShape( ) and endShape( ), with the alterations highlighted.
- Inside draw( ), comment out the line that reads:

drawDataPoints(currentColumn);

by placing a pair of slashes (//) in front of it. On the line that follows, add:

noFill( );

drawDataLine(currentColumn);

```
void drawDataLine(int col) {
beginShape( );
int rowCount = data.getRowCount( );
for (int row = 0; row < 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);
vertex(x, y);
}
}
endShape( );
}
```

Figure 4-8. Continuously drawn time series using vertices

It’s also easy to mix lines and points in the representation to create a background line

that highlights the individual data points.

Modify the end of draw( ) to read as follows:

```
stroke(#5679C1);
strokeWeight(5);
drawDataPoints(currentColumn);
noFill( );
strokeWeight(0.5);
drawDataLine(currentColumn);
```

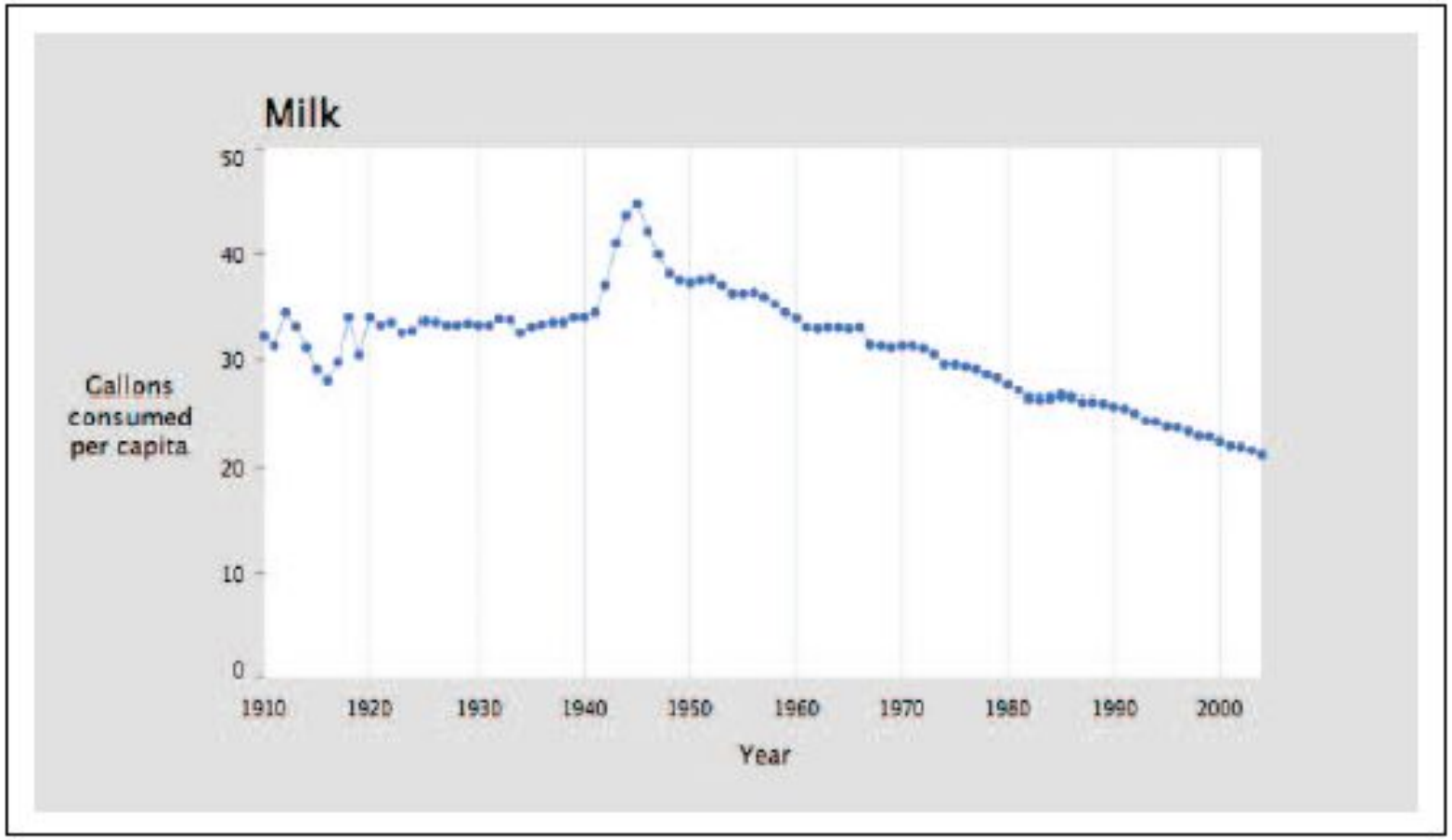

Figure 4-9. Combined dots and continuous line

Using Rollovers to Highlight Points (Interact)

highlight individual points when the mouse is nearby.

```
void drawDataHighlight(int col) {
for (int row = 0; row < 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);
if (dist(mouseX, mouseY, x, y) < 3) {
strokeWeight(10);
point(x, y);
fill(0);
textSize(10);
textAlign(CENTER);
text(nf(value, 0, 2) + " (" + years[row] + ")", x, y-8);
}
}
}
}
```

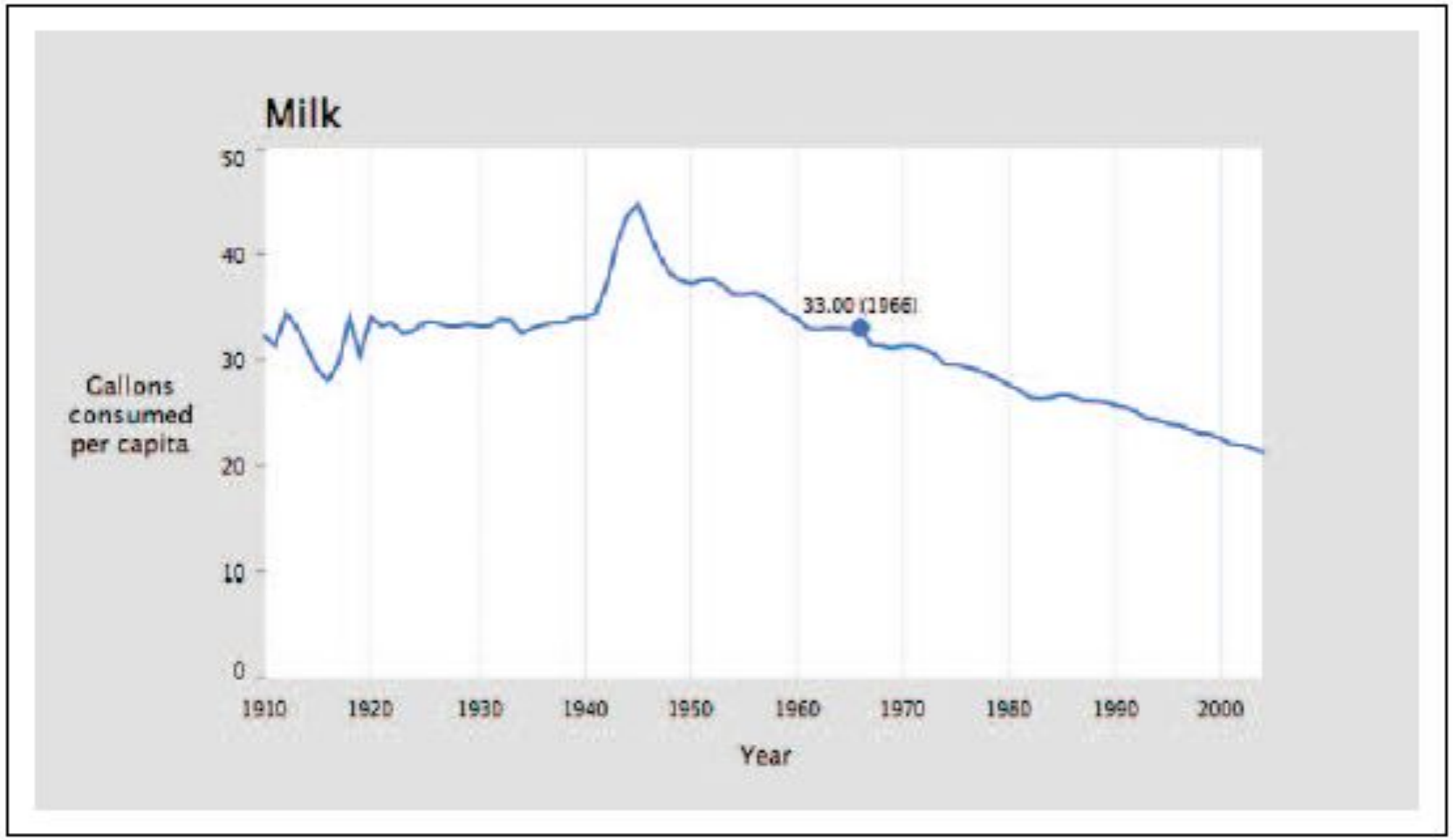

Figure 4-10. Time series with user-selected highlight

Ways to Connect Points (Refine)

- Connecting the points with a curve is often a better option because it prevents the spikiness of the plot from overwhelming the data itself.
- The curveVertex( ) function is similar to the vertex( ) function, except that it connects successive points by fitting them to a curve.

```
void drawDataCurve(int col) {
beginShape( );
for (int row = 0; row < 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);
curveVertex(x, y);
// Double the curve points for the start and stop
if ((row == 0) || (row == rowCount-1)) {
curveVertex(x, y);
}
}
}
endShape( );
}
```

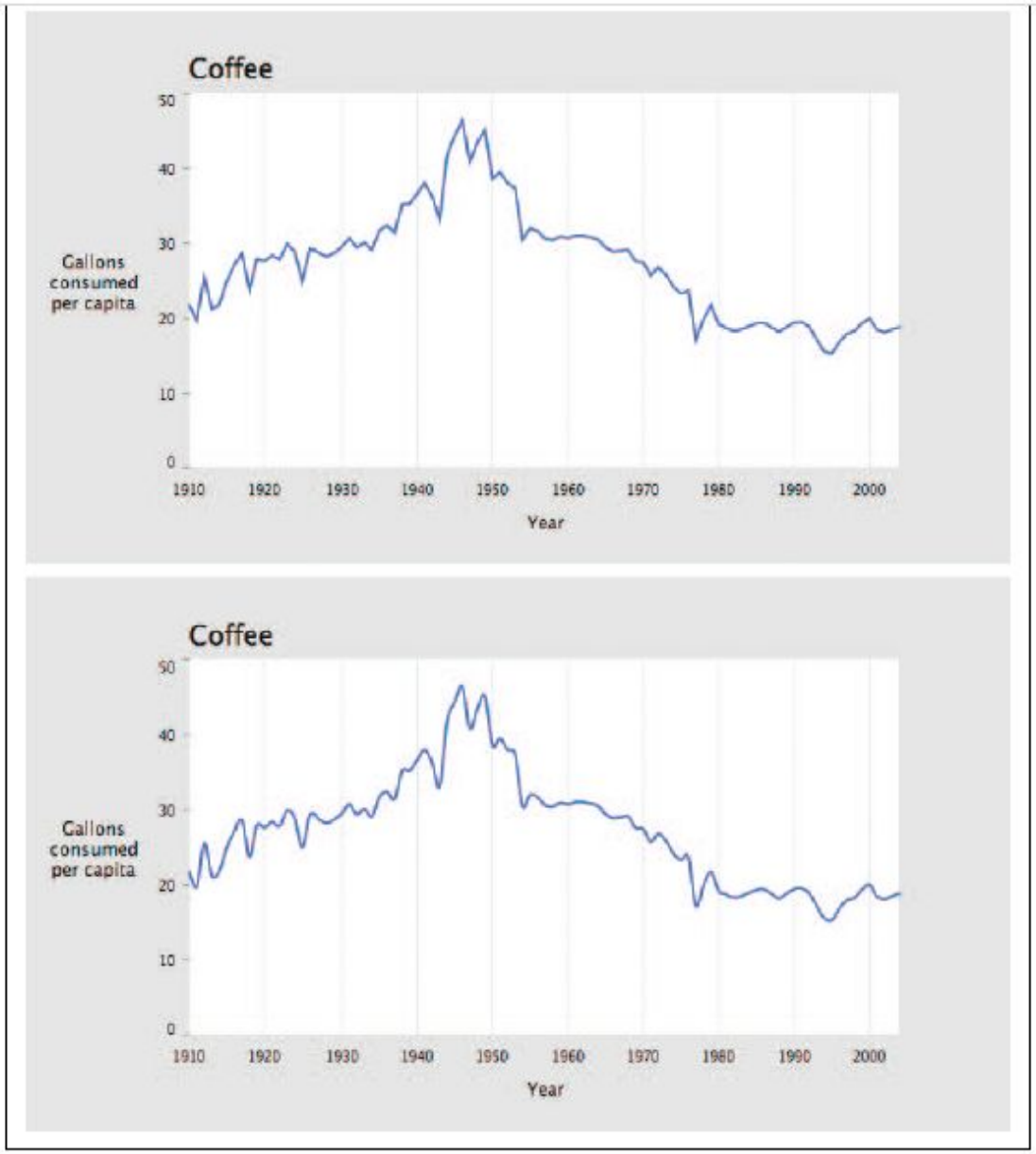

Figure 4-11. Comparison of the use of vertices (top) and curve vertices (bottom)

Showing Data As an Area

Another variation of drawDataLine( ) draws the values as a filled area.

The new drawDataArea( ) function is:

```
void drawDataArea(int col) {
beginShape( );
for (int row = 0; row < 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);
vertex(x, y);
}
}
// Draw the lower-right and lower-left corners.
vertex(plotX2, plotY2);
vertex(plotX1, plotY2);
endShape(CLOSE);
}
```

- Next, modify the end of the draw( ) method to replace the stroke(#5679C1) line with fill(#5679C1), and change noFill( ) to noStroke( ); drawing an outline around an already filled shape is unnecessary:

```
noStroke( );
fill(#5679C1);
drawDataArea(currentColumn);
```

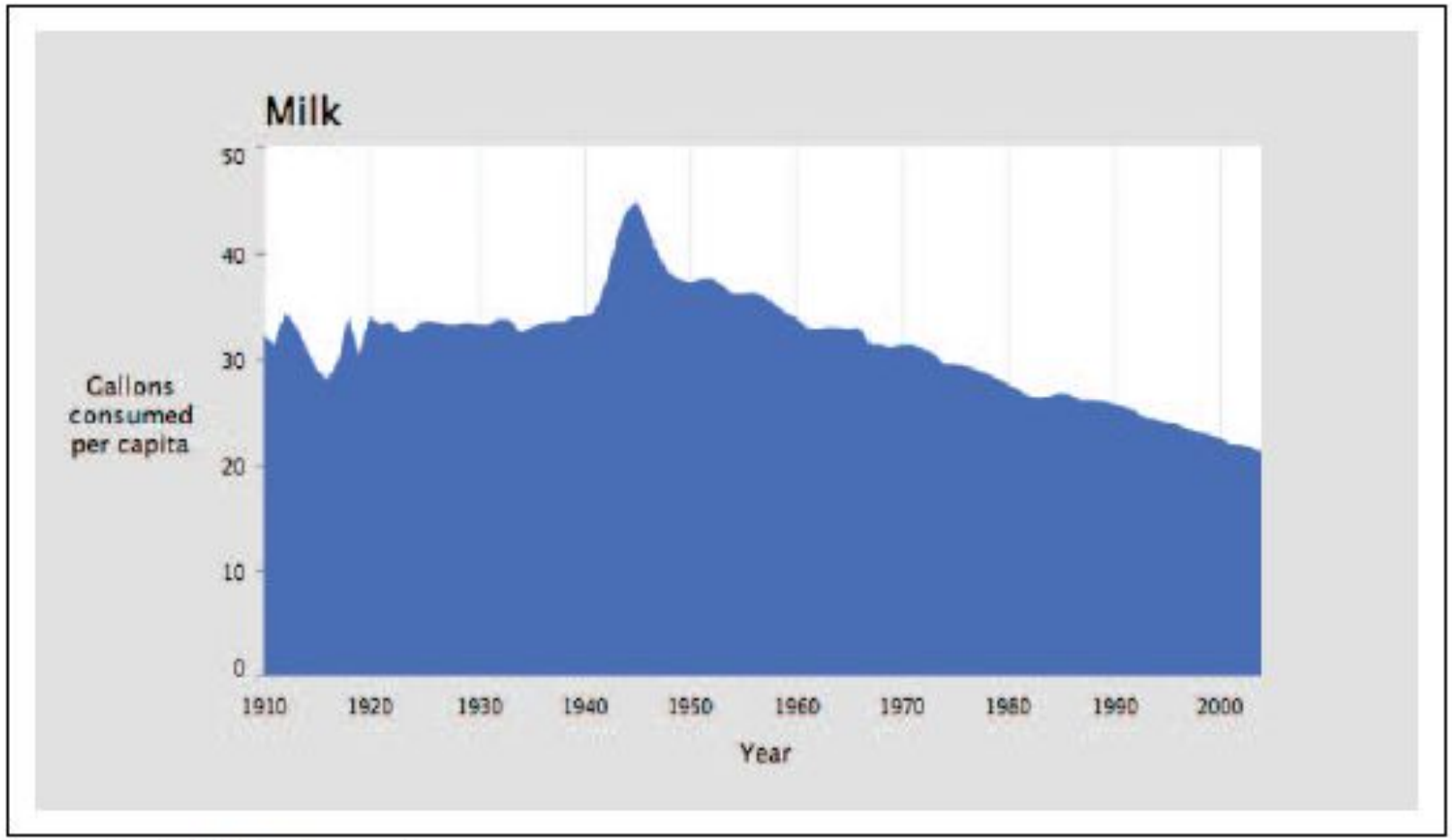

Figure 4-12. Filled time series

References

- Book of the course-Chapter 4

End of Lecture