VTK Charts

2011 Summer Project Week Breakout Session

June 21, 2011

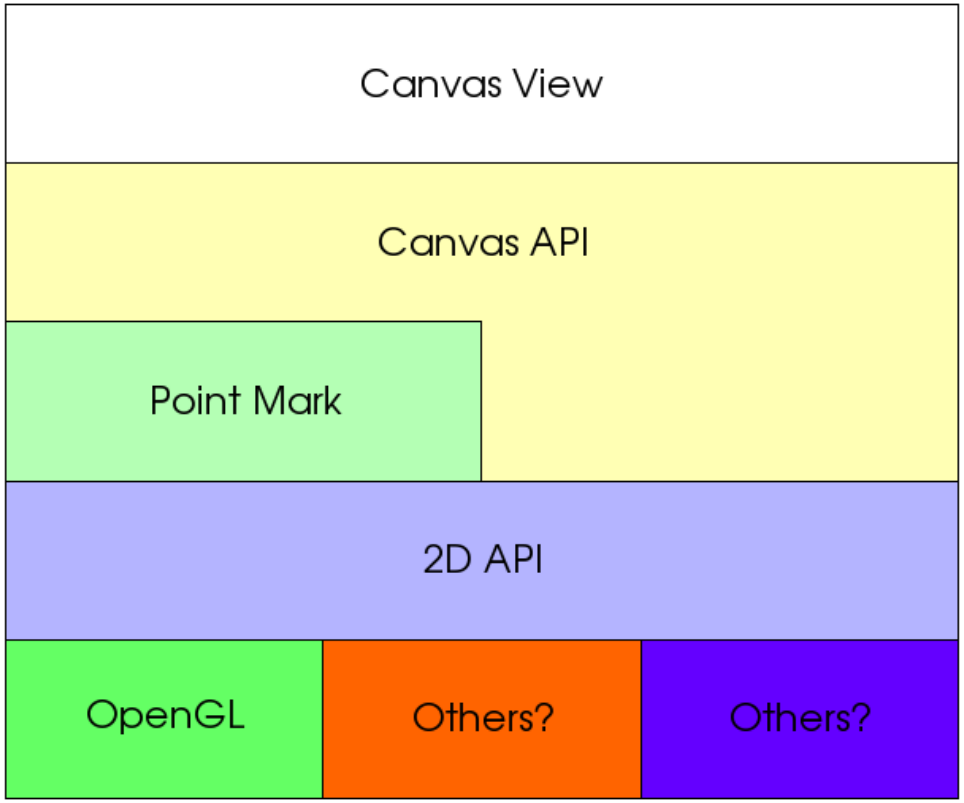
Dr. Marcus D. Hanwell
R&D Engineer
Kitware, Inc.

Basic Chart Features and Types

- Full 2D scene with interaction
  - Multiple charts in one scene
  - Designed with interactivity in mind
- Multiple chart types
  - Line plots
  - Bar graphs
  - Scatter plots
  - Area charts
  - Pie charts
  - Flood plots

Kitware

2D Rendering in VTK



Kitware

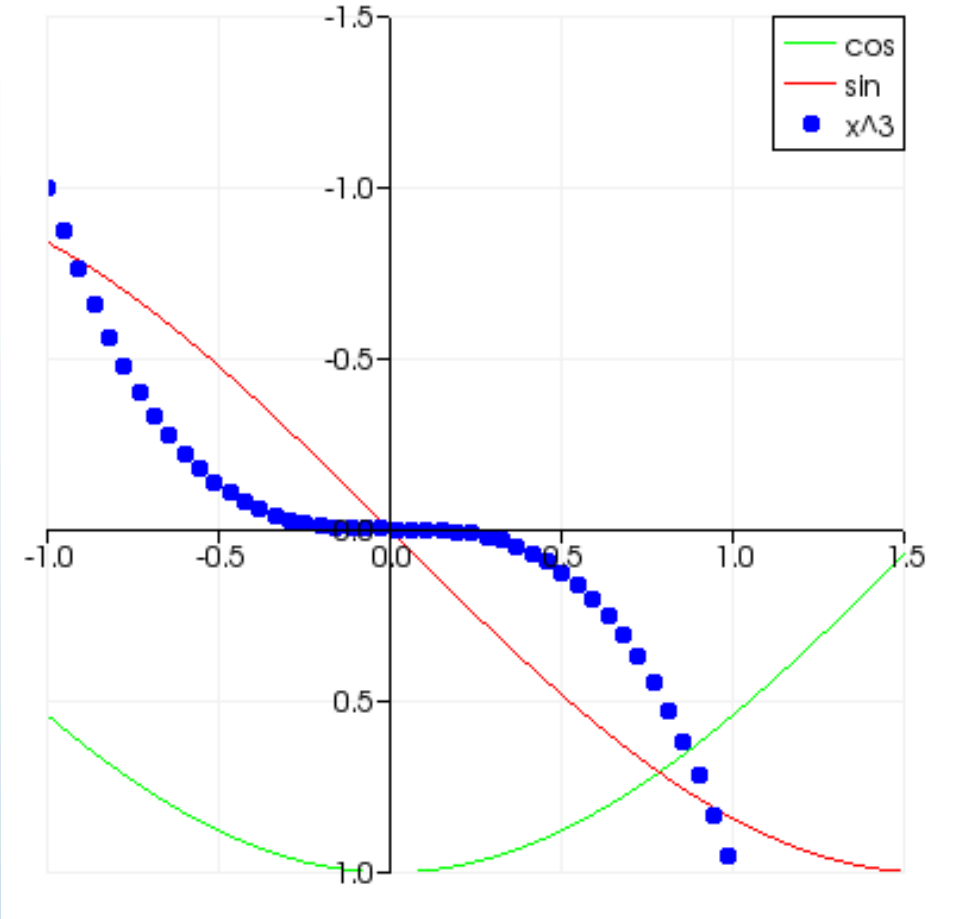
The Chart Scene

- Charts use the 2D scene API
- Hierarchy of objects on a scene
- vtkChart derives from vtkContextItem
  - vtkAxis objects
  - vtkChartLegend
  - vtkContextClip
    - vtkContextTransform
      - vtkPlot derived objects in Cartesian space
- Multiple chart objects can be on a scene
- Propagation of events handled by the scene

Common Features in the Charts

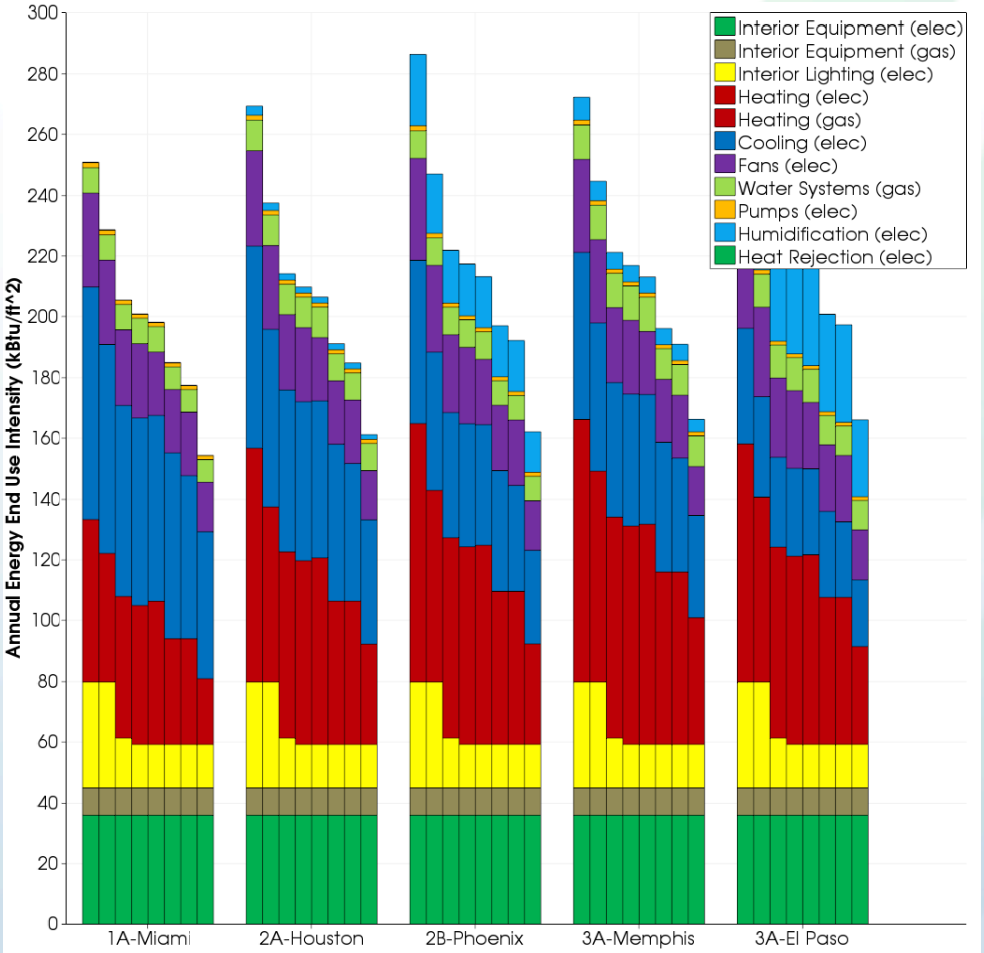
- Points have tooltips
  - Display exact location
  - Add custom names for each point in a plot
- Legends with interactivity
- Normal pan, zoom and selection with mouse
- Already some Qt wrapping
  - STL style API for NREL (Qt + STL)
  - Used in ParaView for all 2D charting
- Rendered using OpenGL, no Qt dependency

Scatter Plot: Scientific Layout



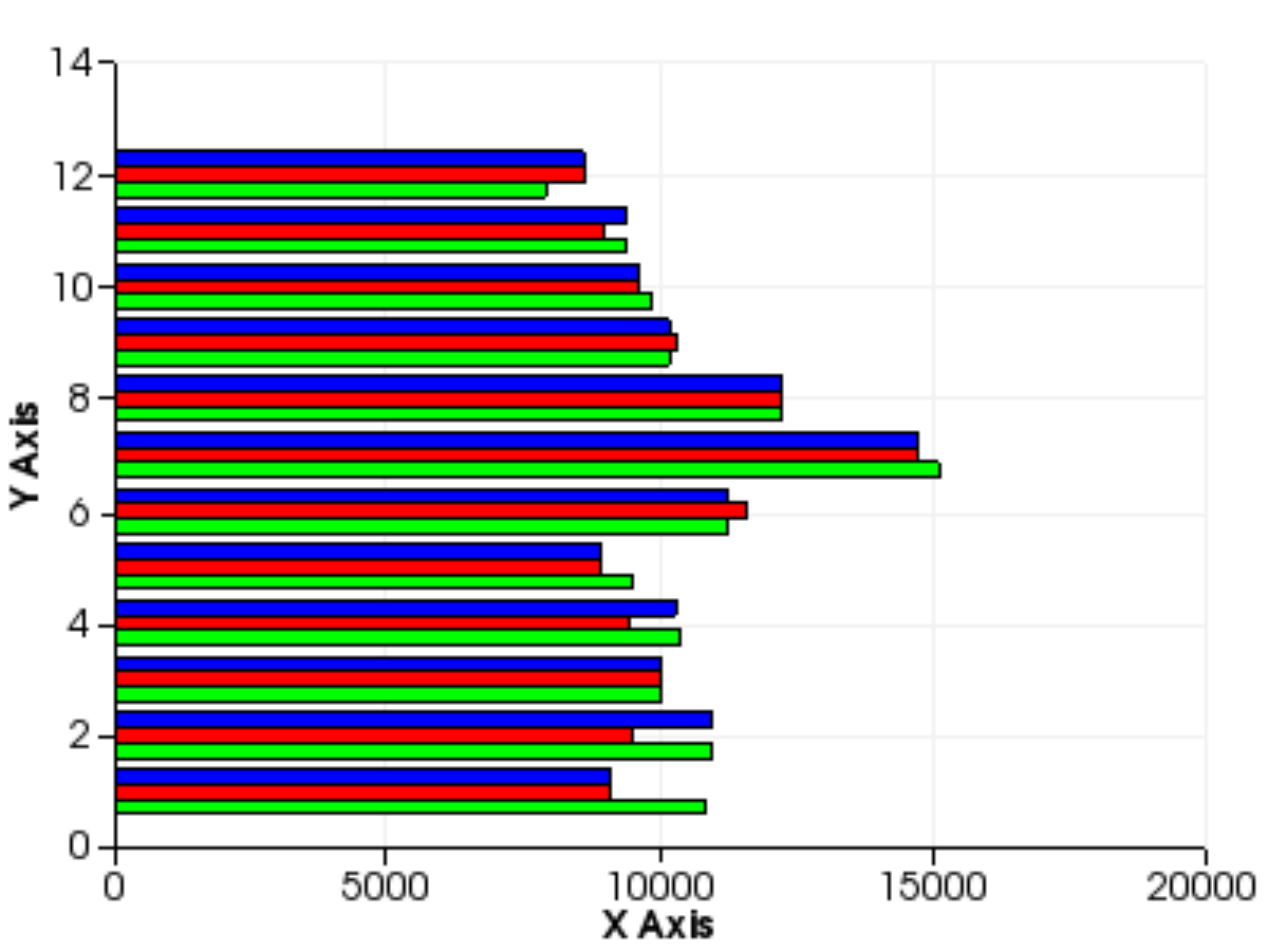
Kitware

Stacked Bar Graphs



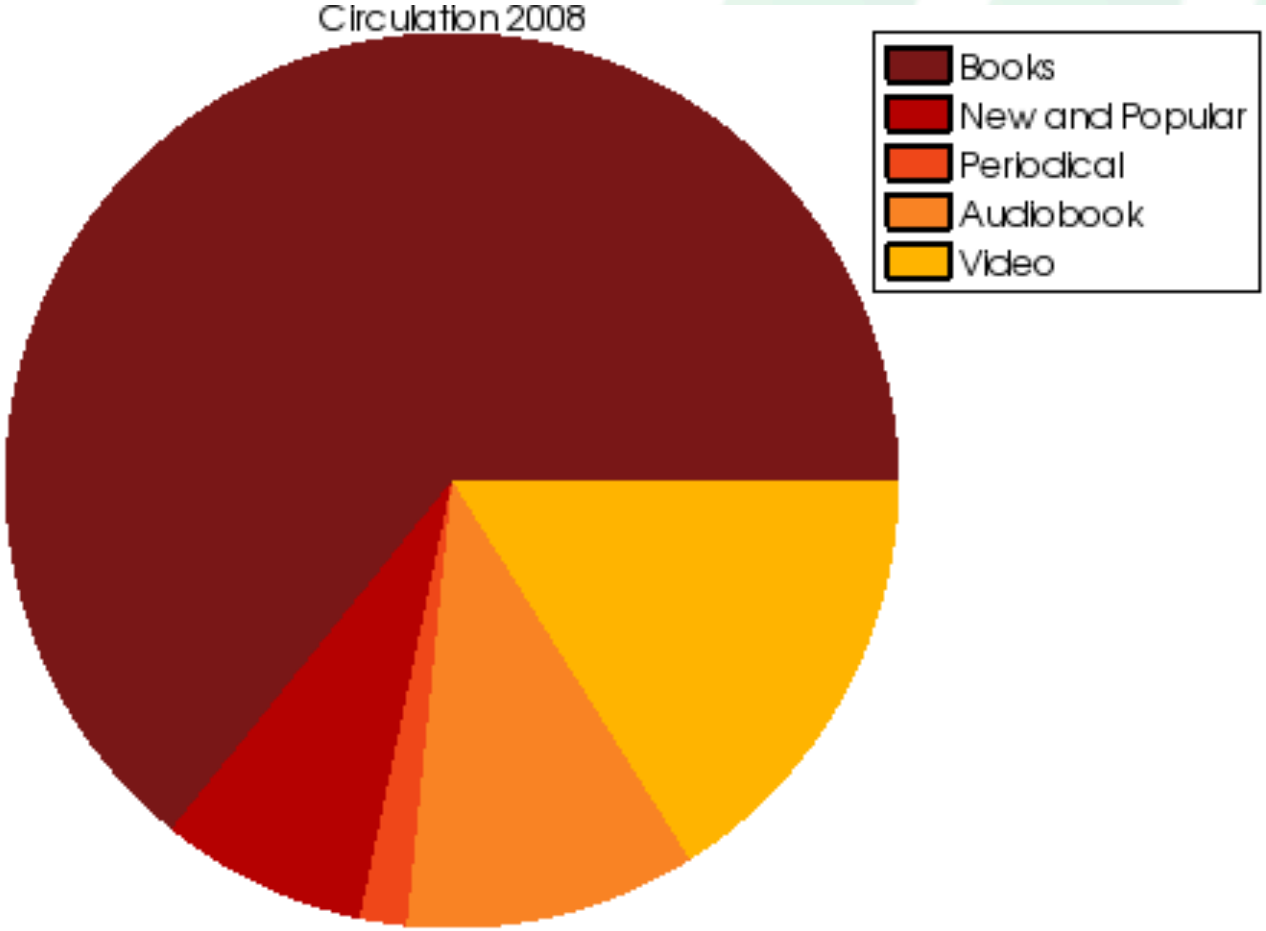
Kitware

Horizontal Bar Graphs: Grouping



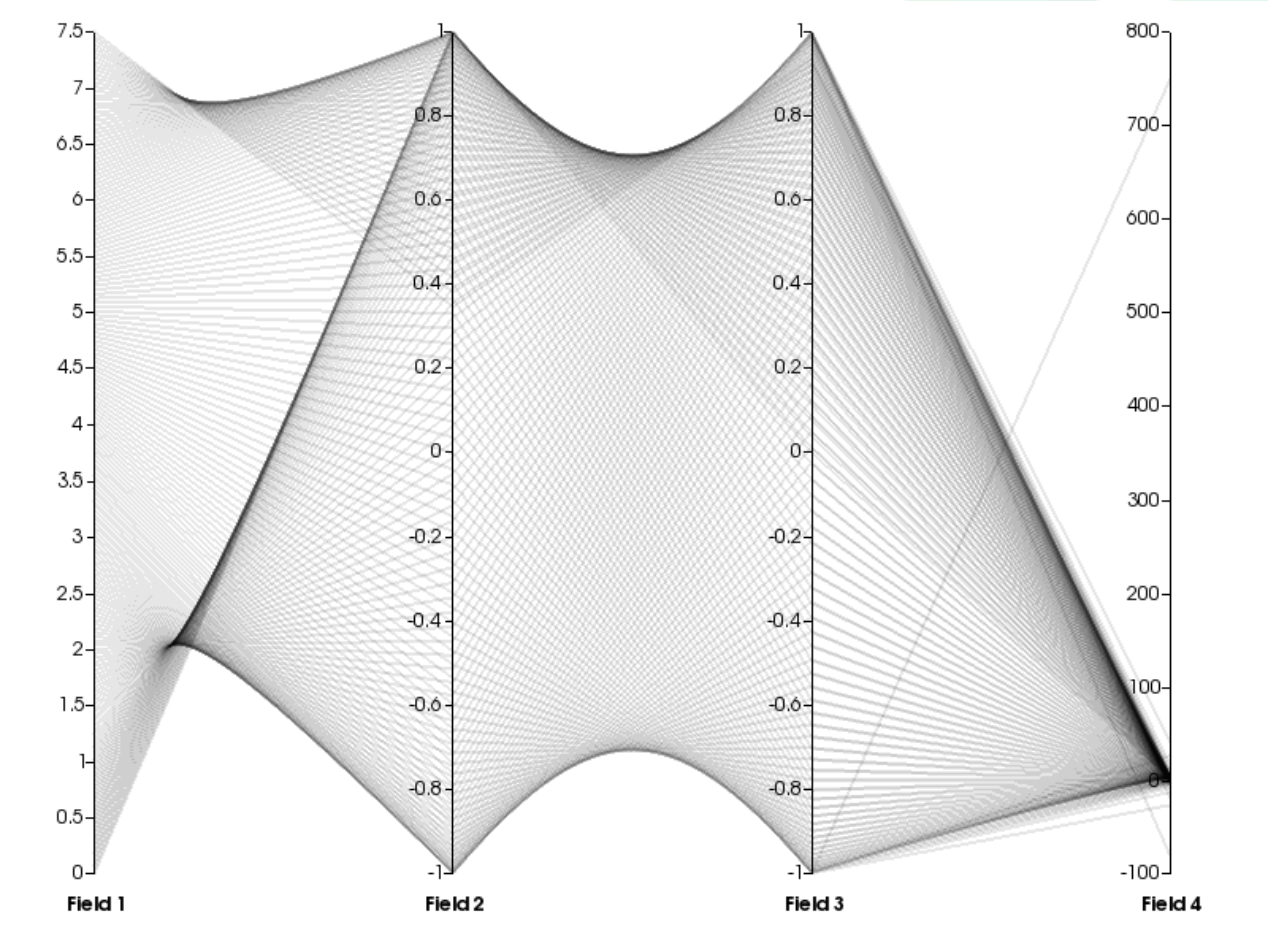
Kitware

Pie Chart



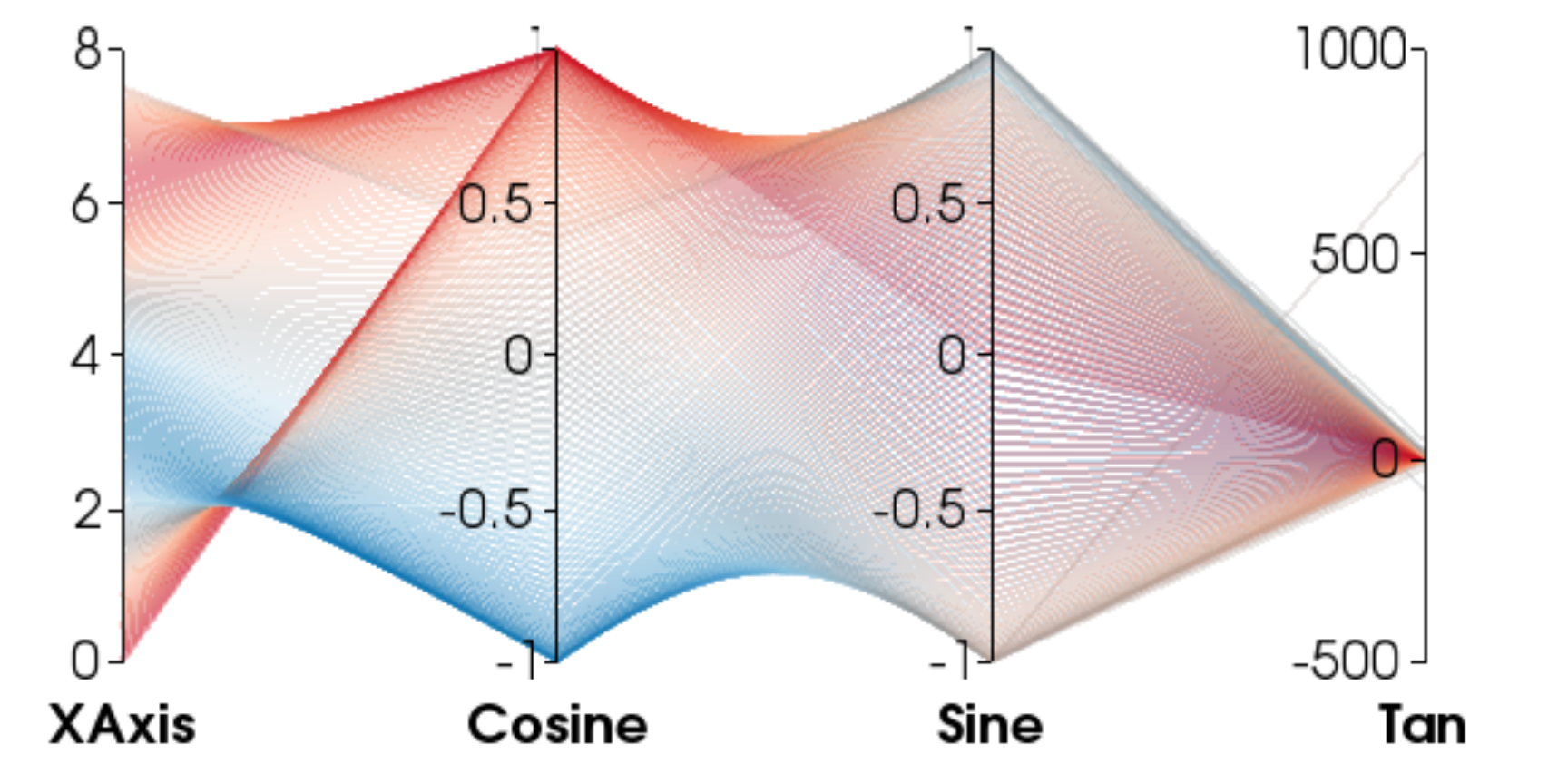
Kitware

Parallel Coordinates



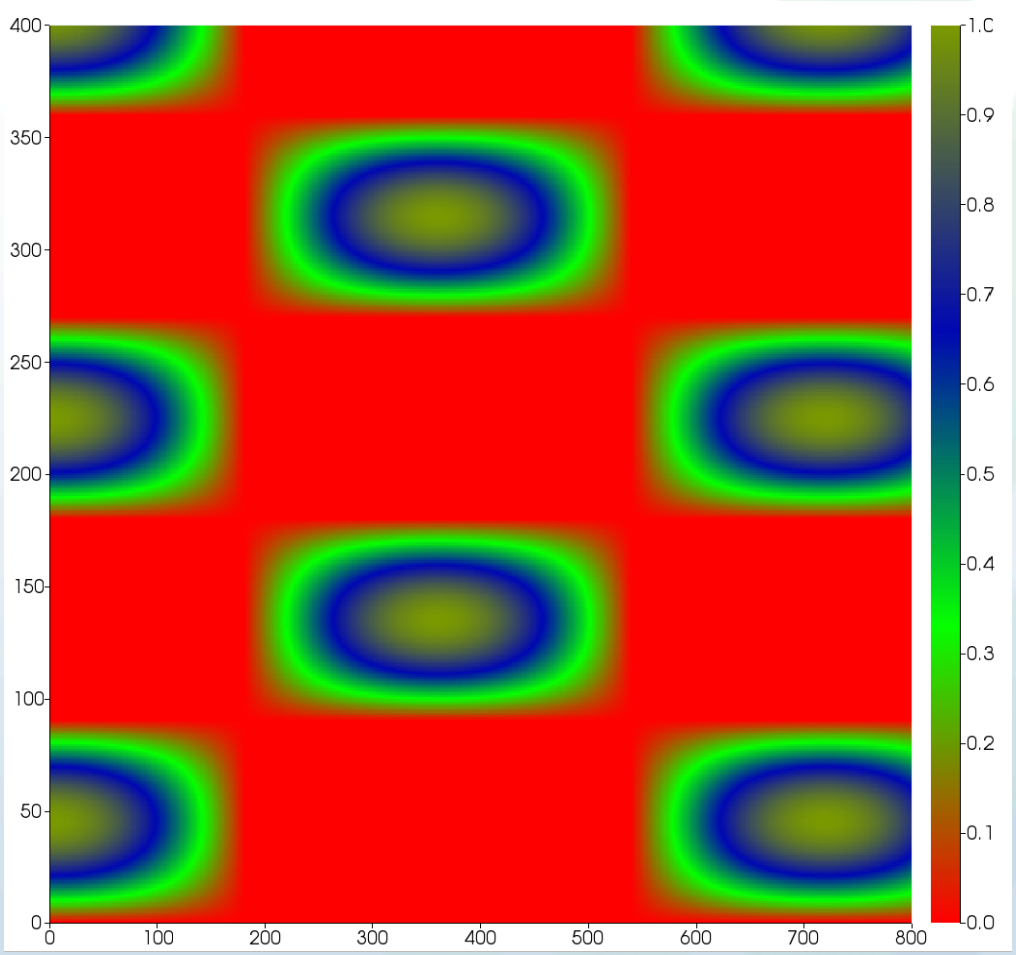
Kitware

Parallel Coordinates: Color



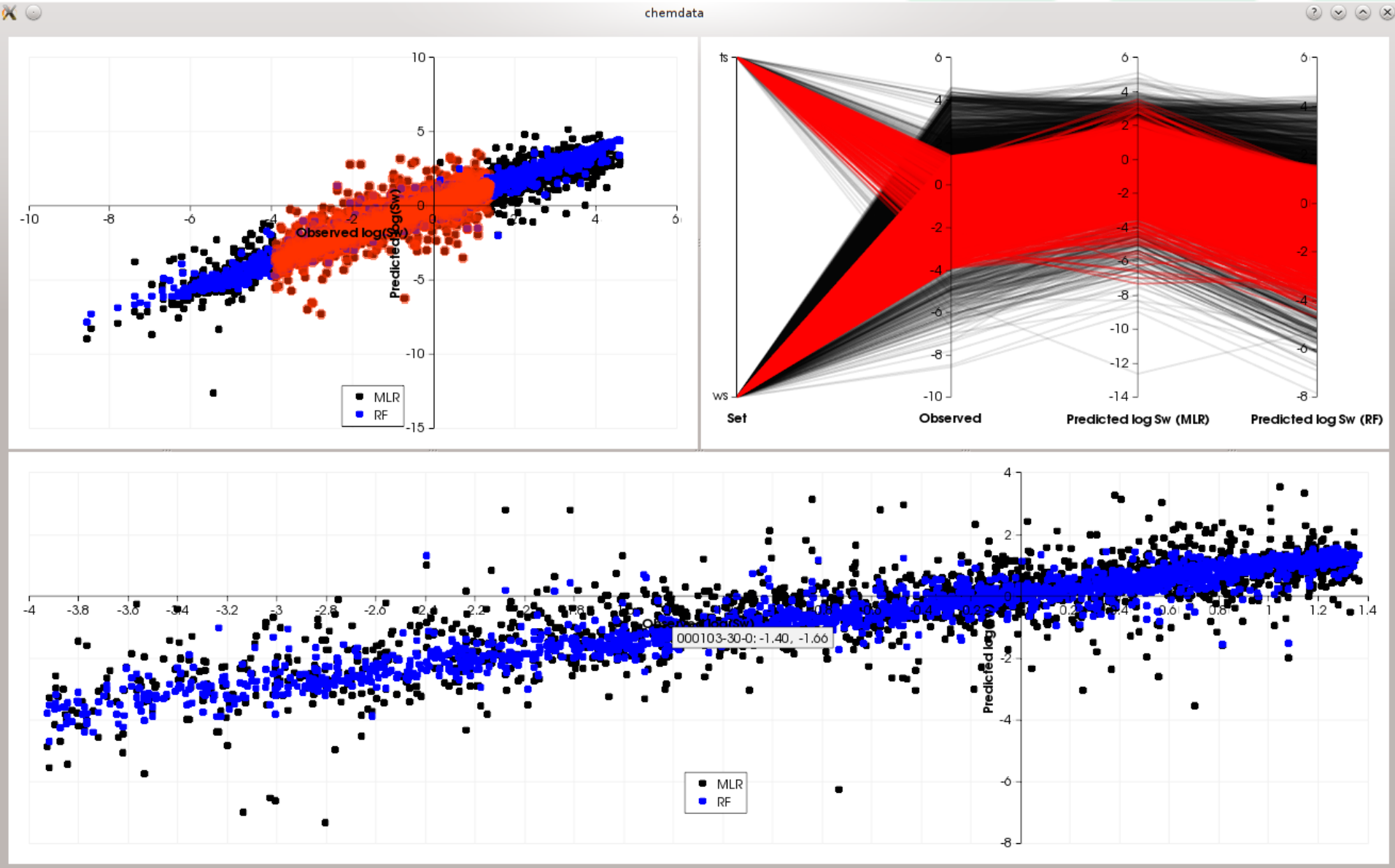
Kitware

2D Histogram/Flood Plot

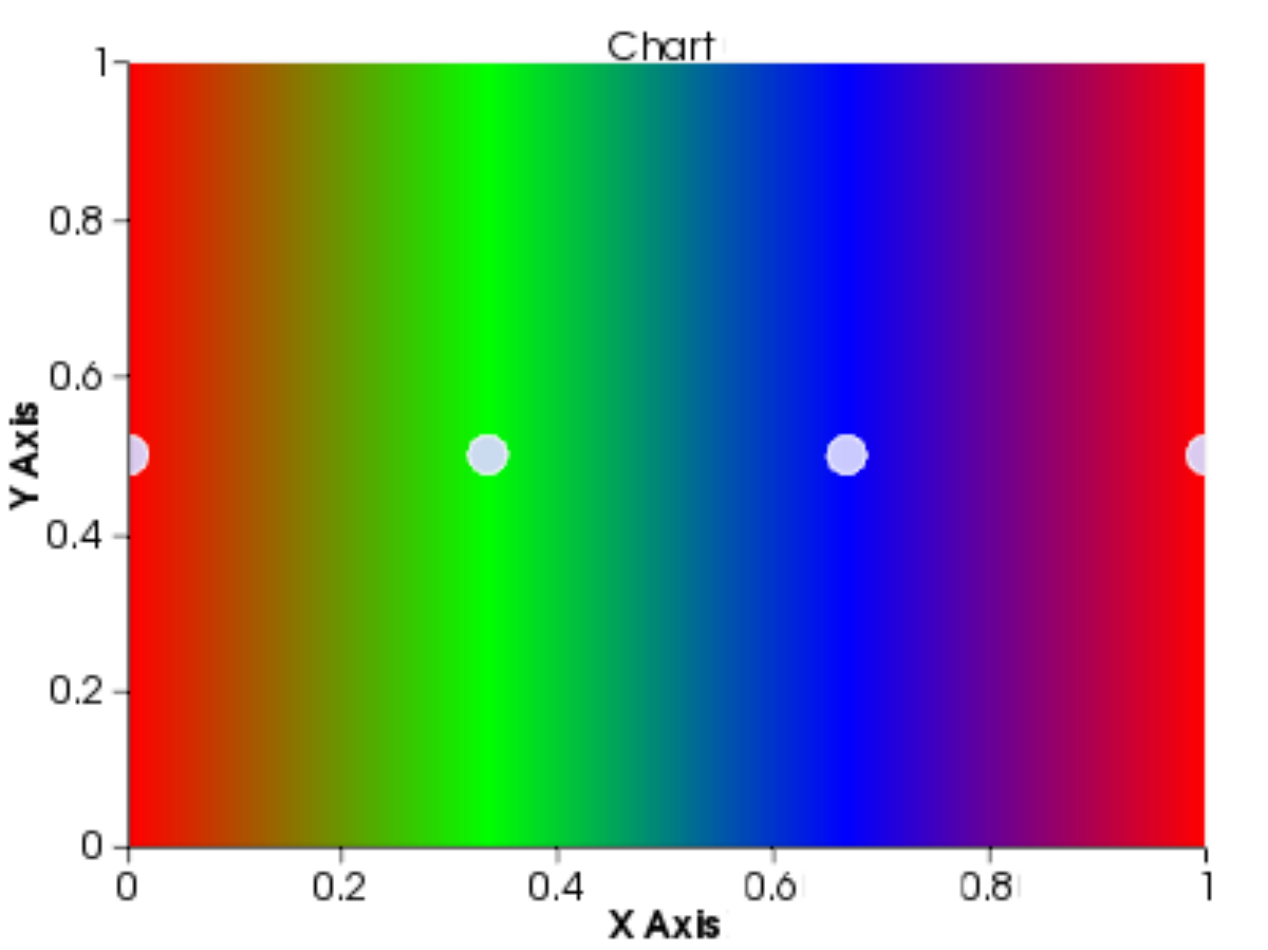


Kitware

Multiple Charts, Linked Selections

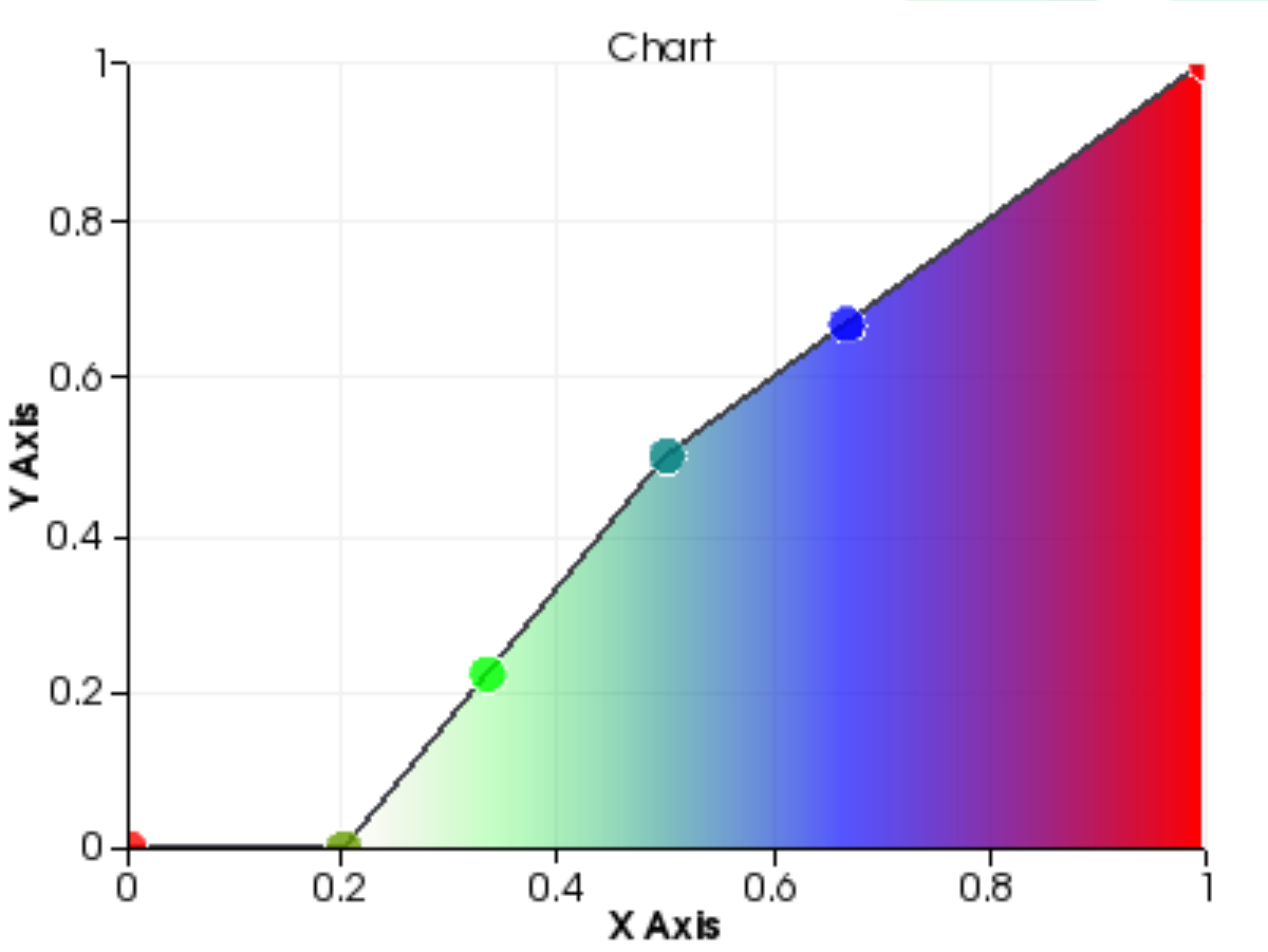


Color Transfer Function



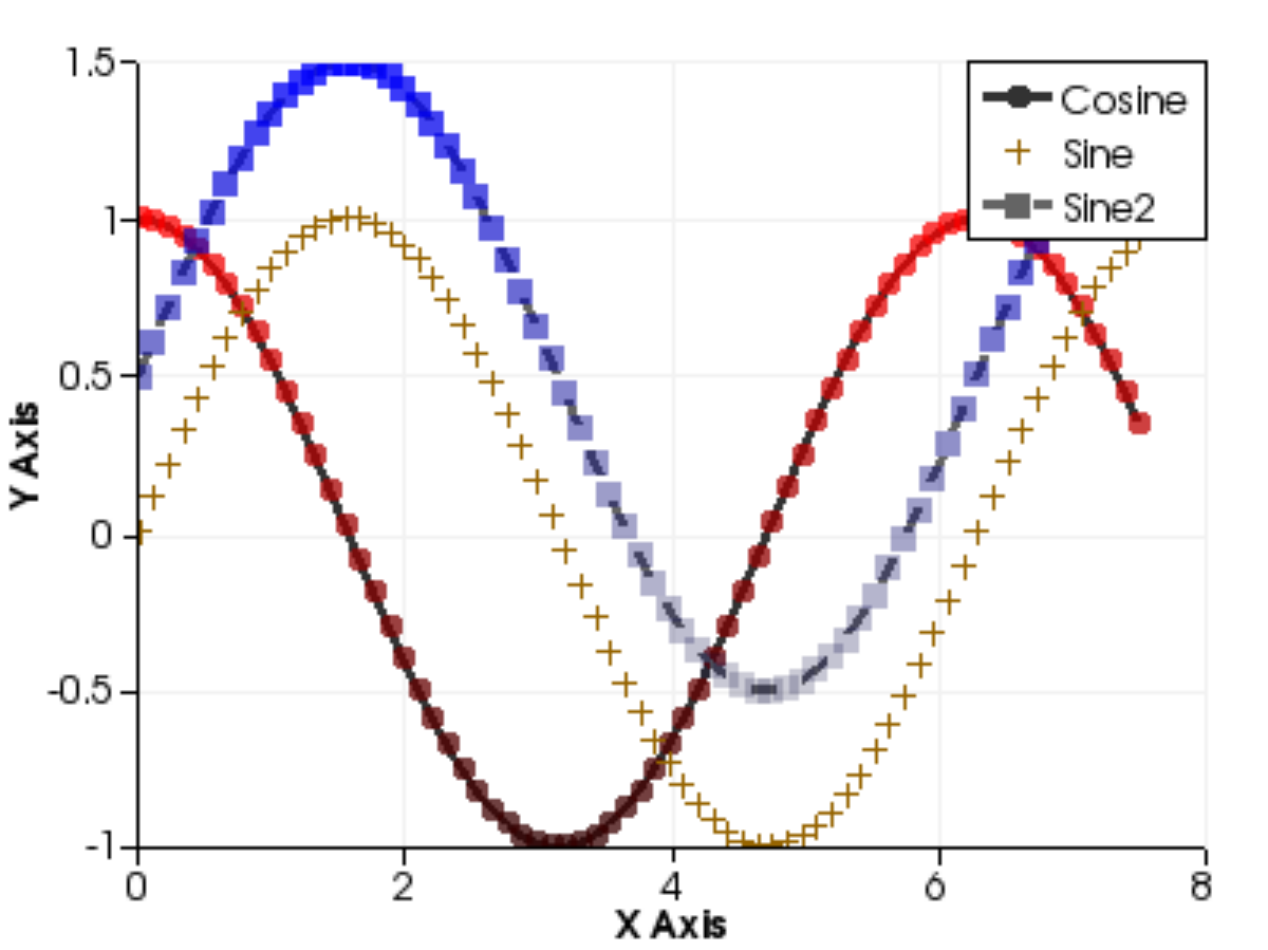
Kitware

Scalars to Colors: Interactive



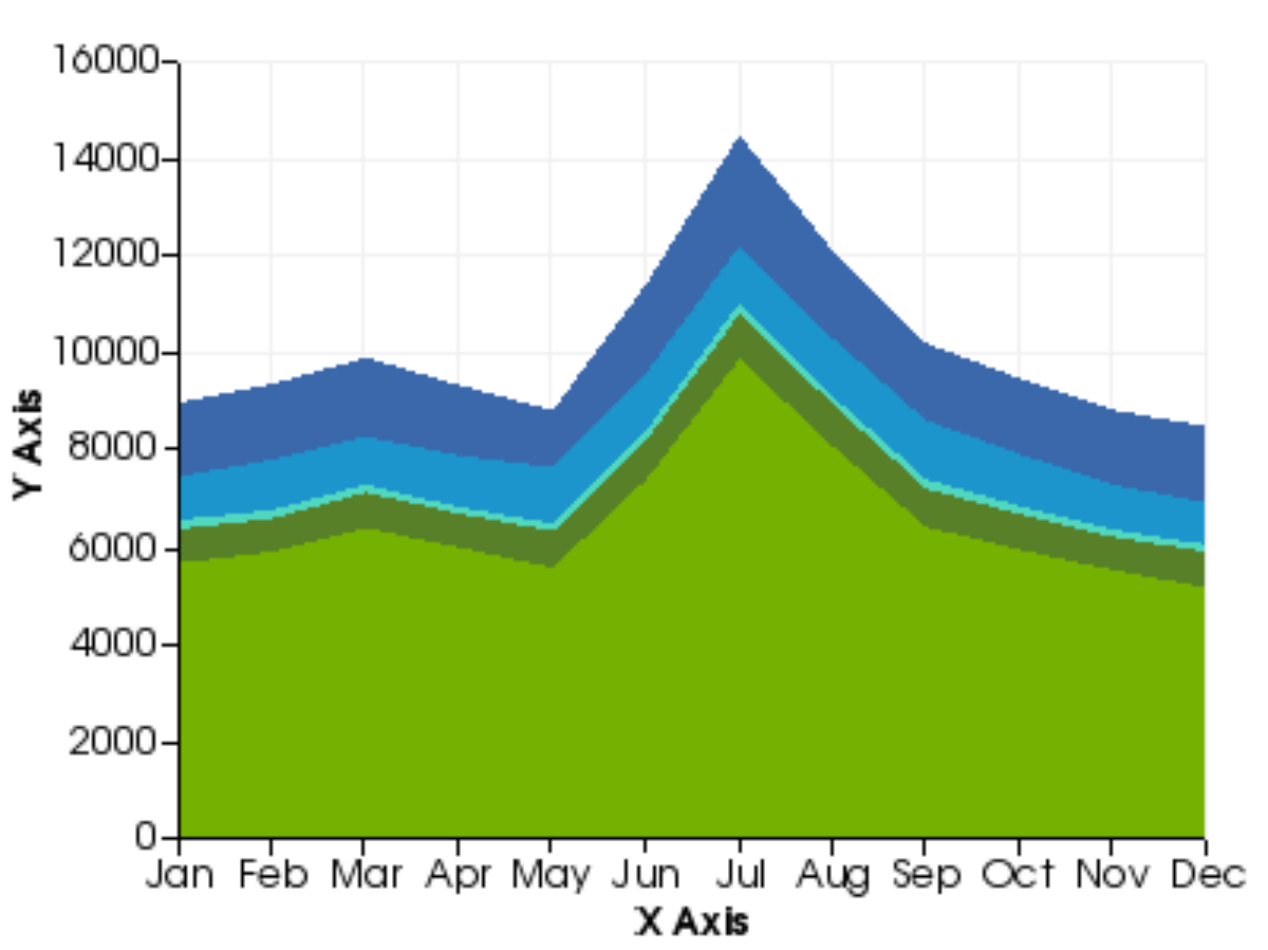
Kitware

Points with Color on 3rd Column



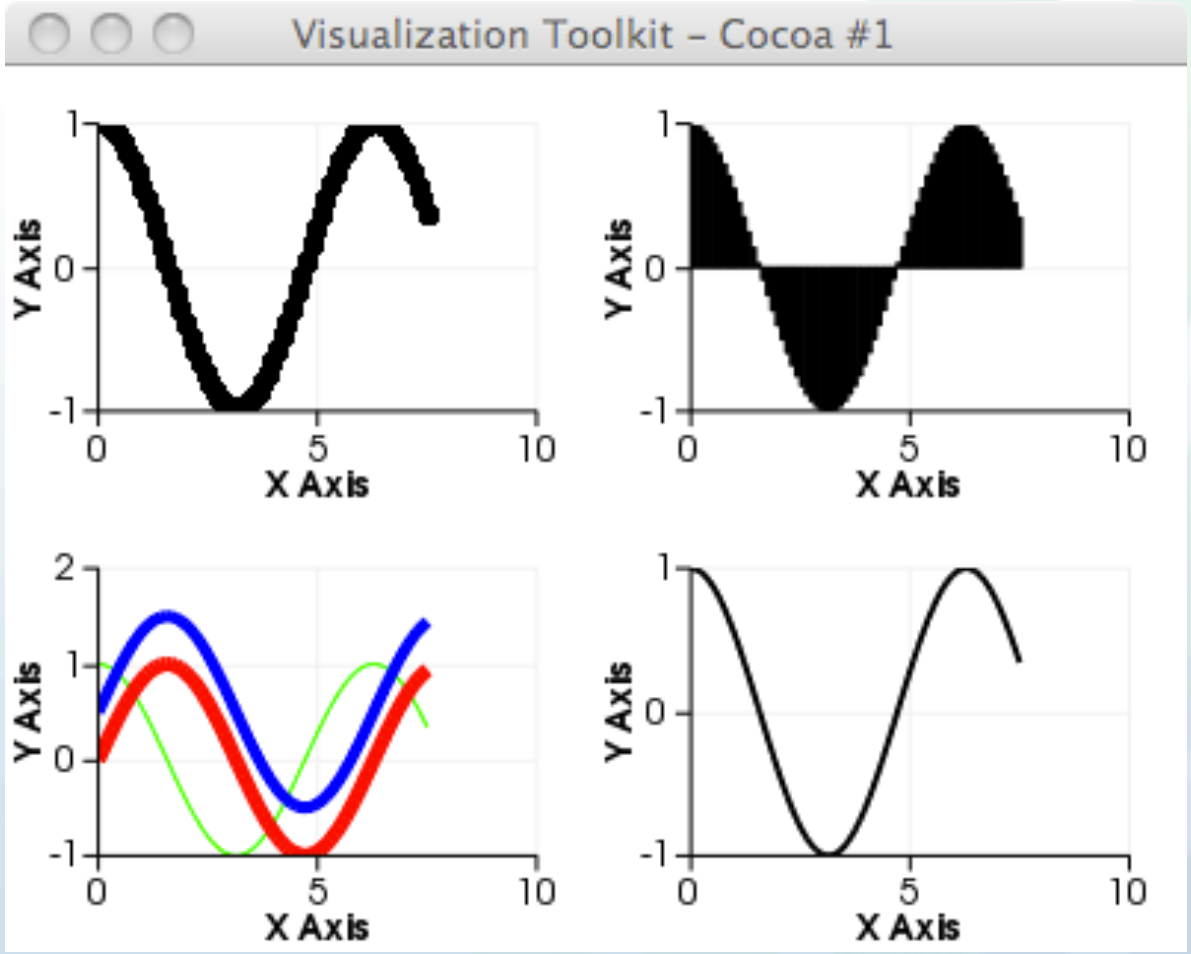
Kitware

Area Chart



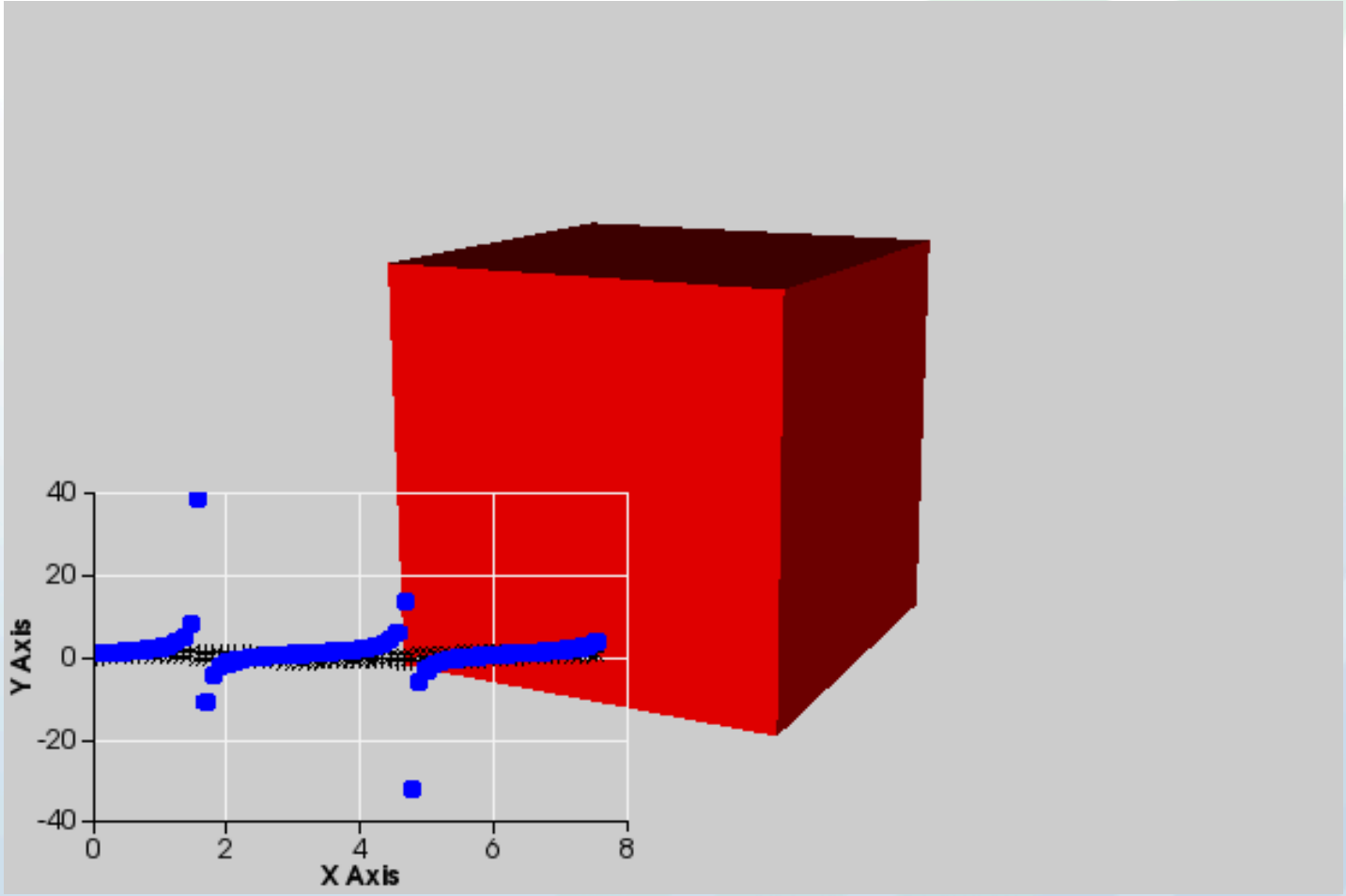
Kitware

Matrix of Plots in One Scene



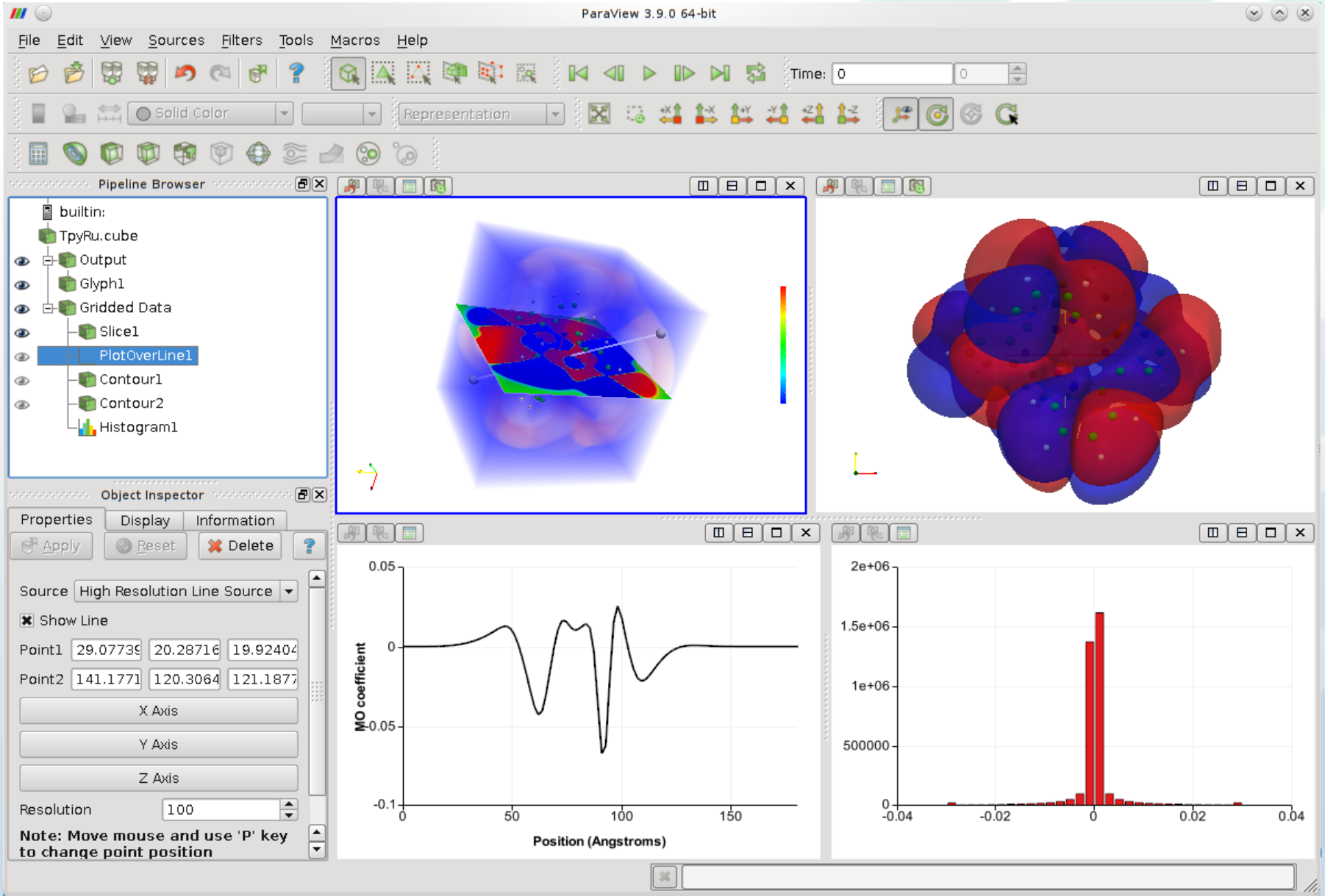
Kitware

Overlay on 3D Scene



Kitware

Charts in ParaView



Initializing the 2D Scene

- Need a render window, empty scene

```
vtkNew<vtkContextView> view;
```

- You should then add something to the scene

```
vtkNew<vtkChartXY> chart;
chart->SetRenderEmpty(true);
view->GetScene()->AddItem(chart.GetPointer());
```

- The scene will now have something to render
- Start interaction and show the window

```
view->GetInteractor()->Initialize();
view->GetInteractor()->Start();
```

Kitware

Using the 2D API With Qt

- Use the QVTKWidget to get a QWidget
- Need to get interactor/window right
- Can then use as any other widget in application

```
QVTKWidget *widget = new QVTKWidget;
vtkContextView *view = vtkContextView::New();

// Now set up interactor and render window
view->SetInteractor(widget->GetInteractor());
widget->SetRenderWindow(view->GetRenderWindow());
```

Kitware

Chart Input – vtkTable

- The charts use vtkTable as input
  - A collection of vtk*Array
  - Named columns, used as default labels

```
// Create a table with two named arrays
vtkNew<vtkTable> table;
vtkNew<vtkFloatArray> arrX;
arrX->SetName("X Axis");
table->AddColumn(arrX.GetPointer());
vtkNew<vtkFloatArray> arrC;
arrC->SetName("Cosine");
table->AddColumn(arrC.GetPointer());
```

Kitware

vtkTable – Adding Data

- All columns should be the same length
- Populated from readers, or programmatically

```
// Create a table with two named arrays
int numPoints = 69;
float inc = 7.5 / (numPoints-1);
table->SetNumberOfRows(numPoints);
for (int i = 0; i < numPoints; ++i)
  {
  table->SetValue(i, 0, i * inc);
  table->SetValue(i, 1, cos(i * inc));
  }
```

Kitware

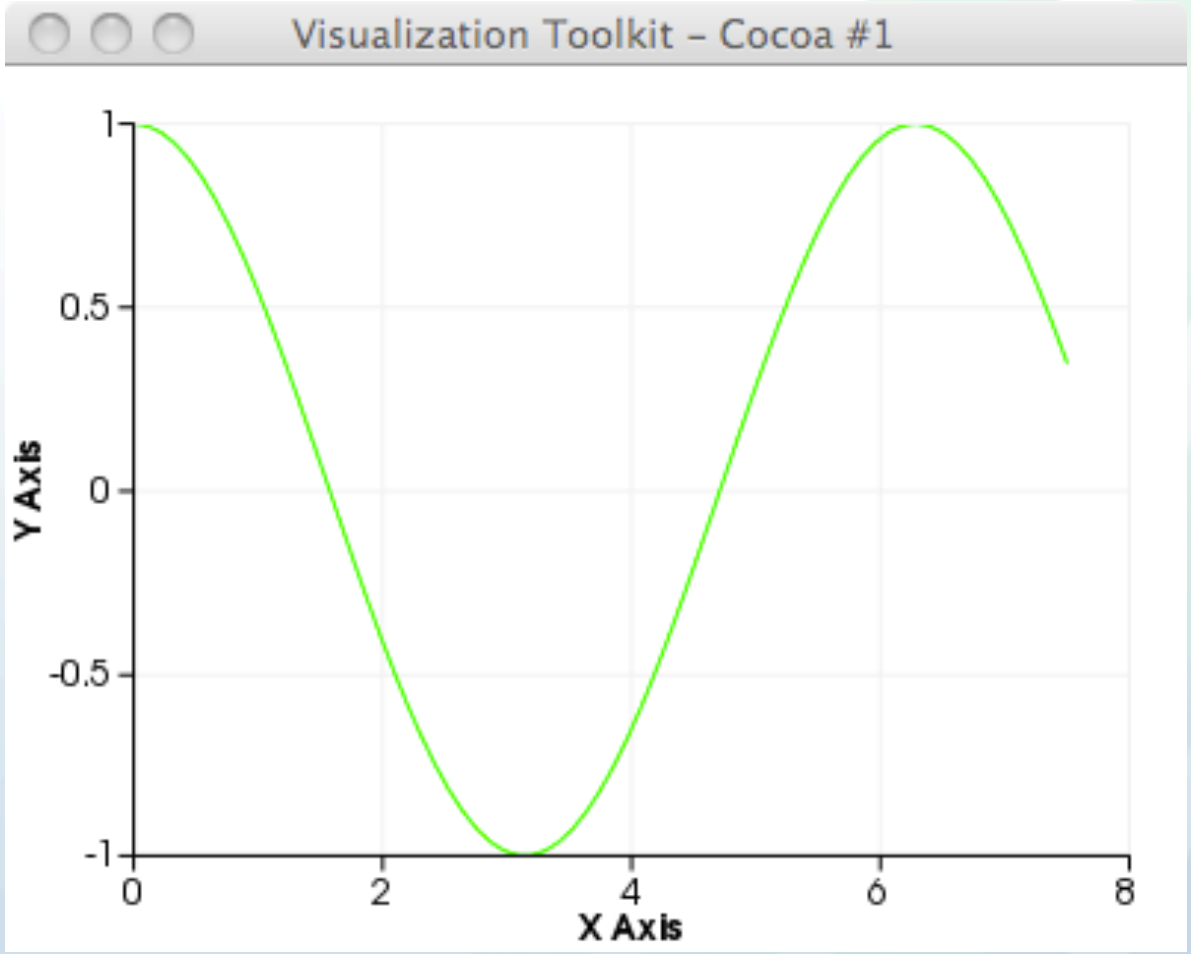
vtkChartXY – Adding Plots

- Simple API to add plots to a chart
- Several types of plot
  - Plots have many properties that can be set
  - Each plot can have a different vtkTable as input

```
// Add a line plot to the chart
vtkPlot *line = chart->AddPlot(vtkChart::LINE);
// Use columns 0 and 1 for x and y
line->SetInput(table.GetPointer(), 0, 1);
// Make the plot green, with a width of 1.0 pixels
line->SetColor(0, 255, 0, 255);
line->SetWidth(1.0);
```

Kitware

Plot of Cosine



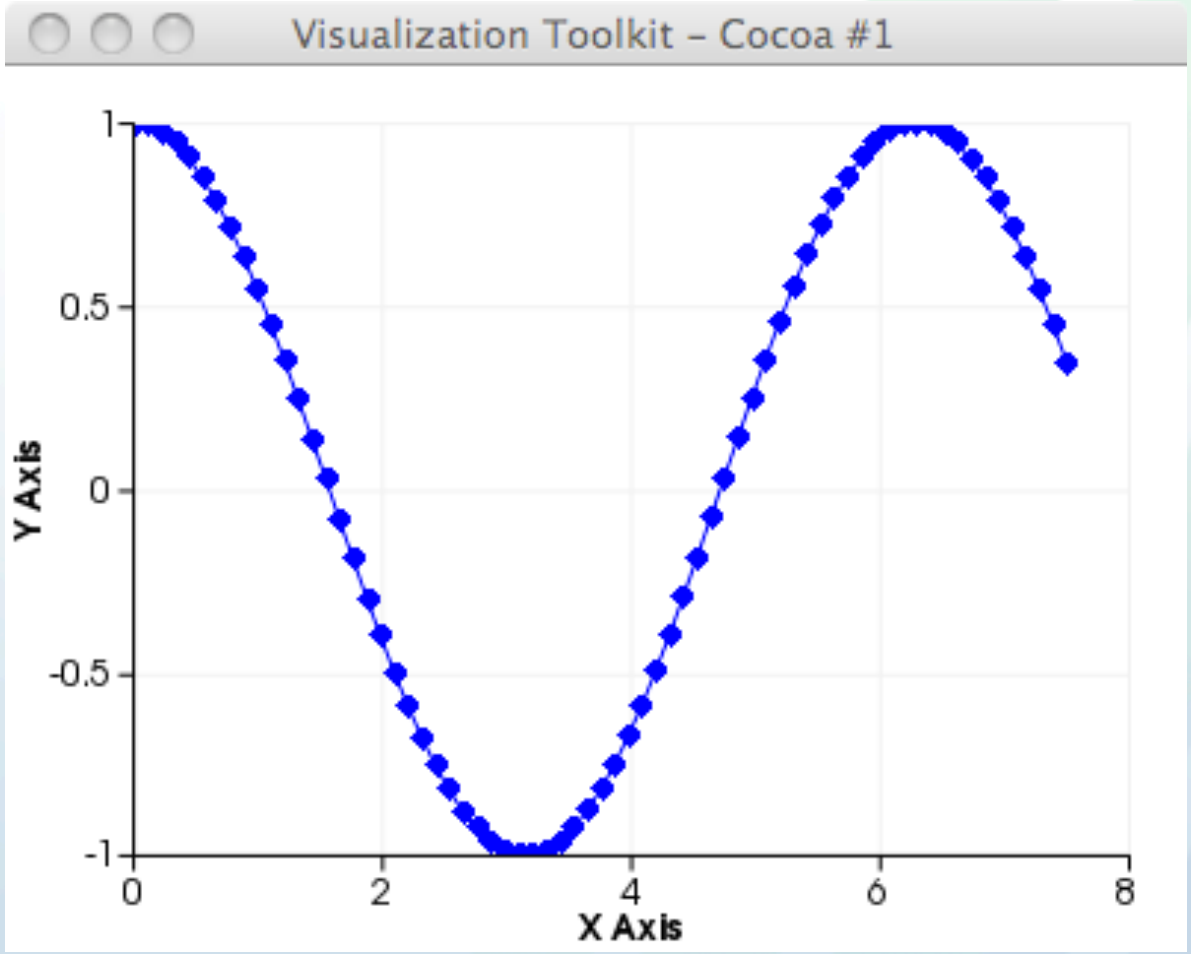
Kitware

Setting Plot Properties

- Using a plot object, can modify symbol, color…

```
// Use a diamond symbol for the points in the plot
vtkPlotPoints::SafeDownCast(line)
  ->SetMarkerStyle(vtkPlotPoints::DIAMOND);
line->SetColor(0, 0, 255, 255);
```

Line Plot with Symbols



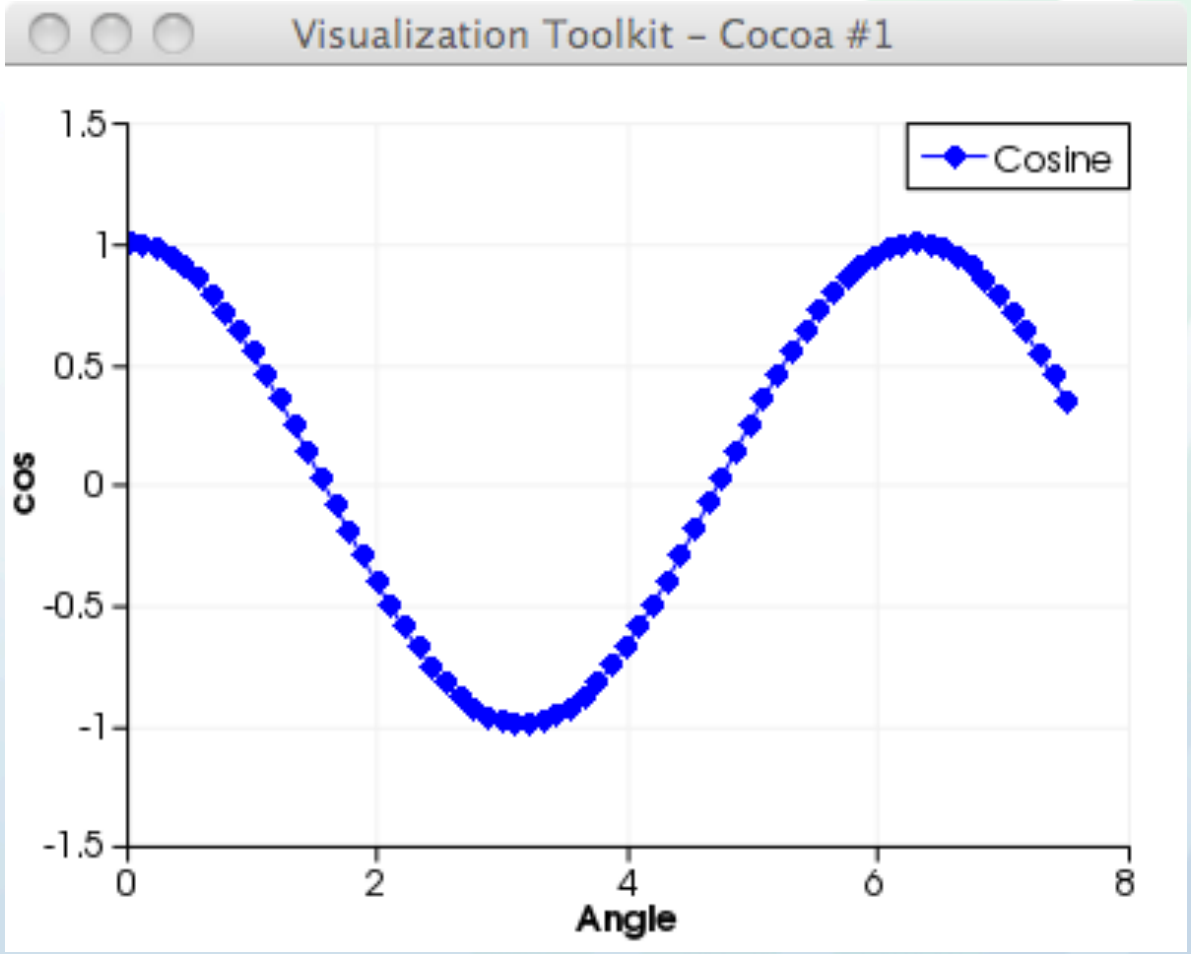
Kitware

Setting Chart Properties

- The charts have axes, legends, plots
- Usually get the object, then modify it

```
chart->GetAxis(vtkAxis::LEFT)->SetTitle(”cos");
chart->GetAxis(vtkAxis::LEFT)->SetRange(-1.5, 1.5);
// Prevent rescaling the axis
chart->GetAxis(vtkAxis::LEFT)
  ->SetBehavior(vtkAxis::FIXED);
chart->GetAxis(vtkAxis::BOTTOM)->SetTitle(”Angle");
chart->SetShowLegend(true);
```

Chart Showing Legend + Axis Labels



Kitware

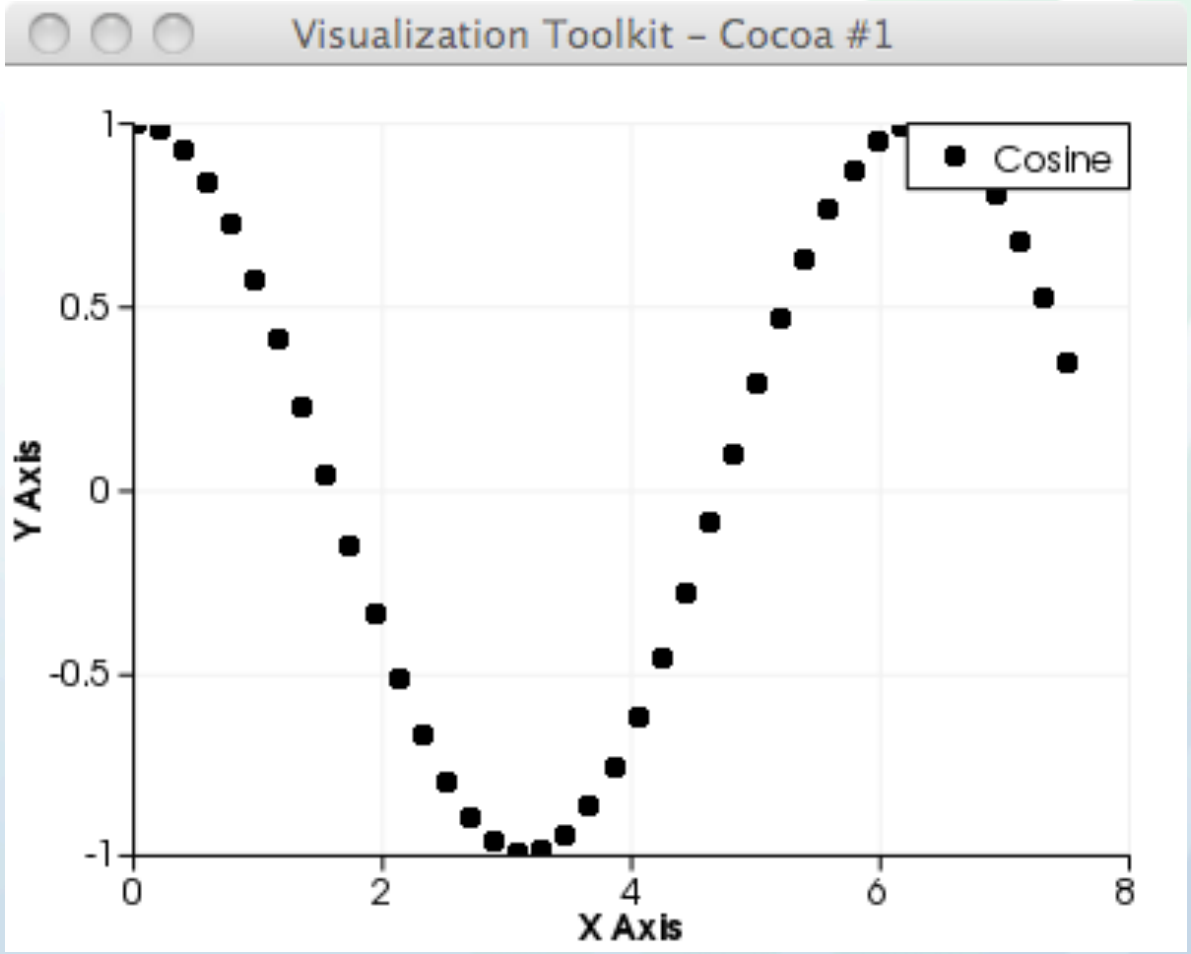
Scatter Plots

- Scatter plots are very similar…
- Defaults are different
- Line plot derived from scatter plot

```
vtkPlot *points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 1);
```

Kitware

Simple Scatter Plot



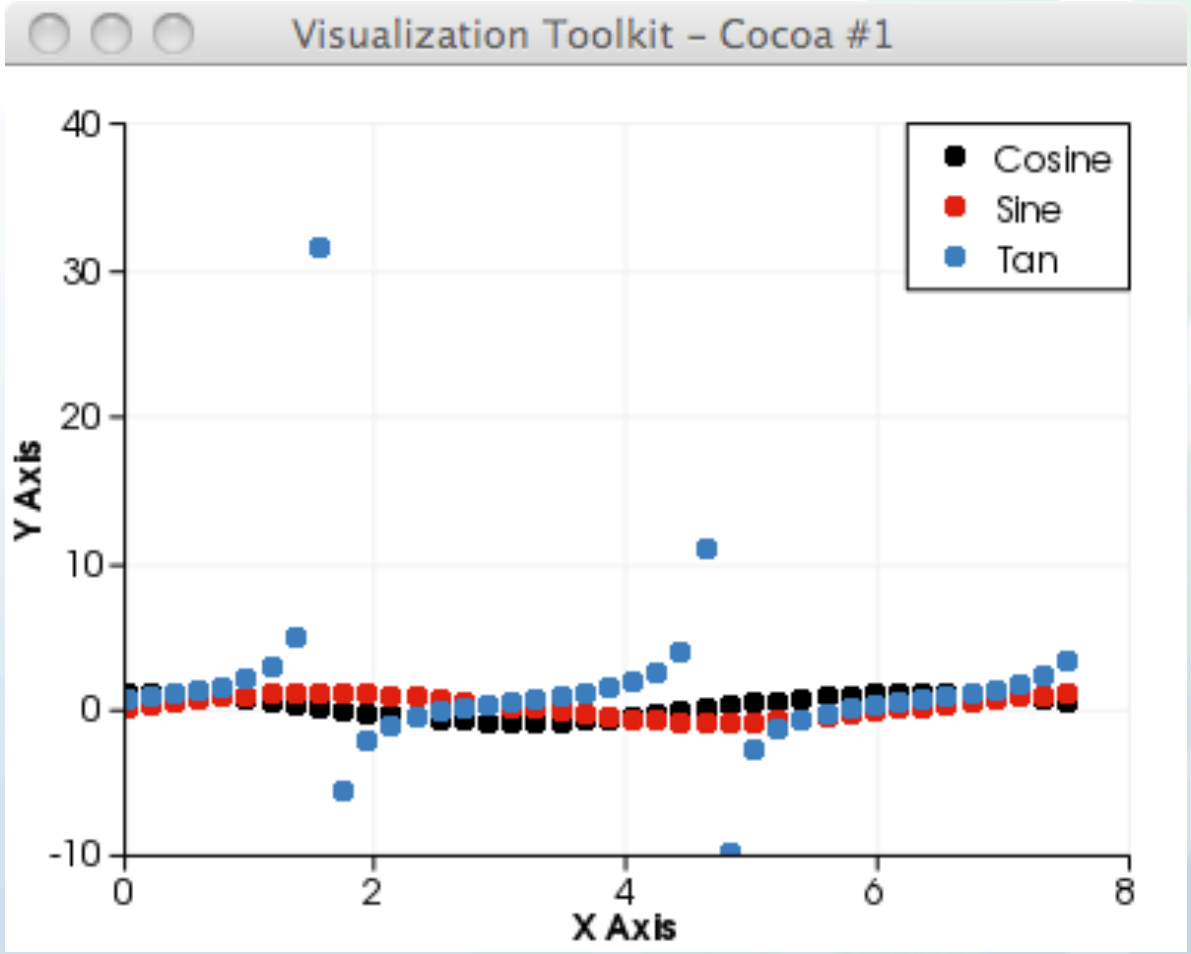
Kitware

Scatter Plots – Multiple Series

- Just add more plots of type vtkChart::POINTS

```
points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 2);
points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 3);
```

Scatter Plot with Multiple Series



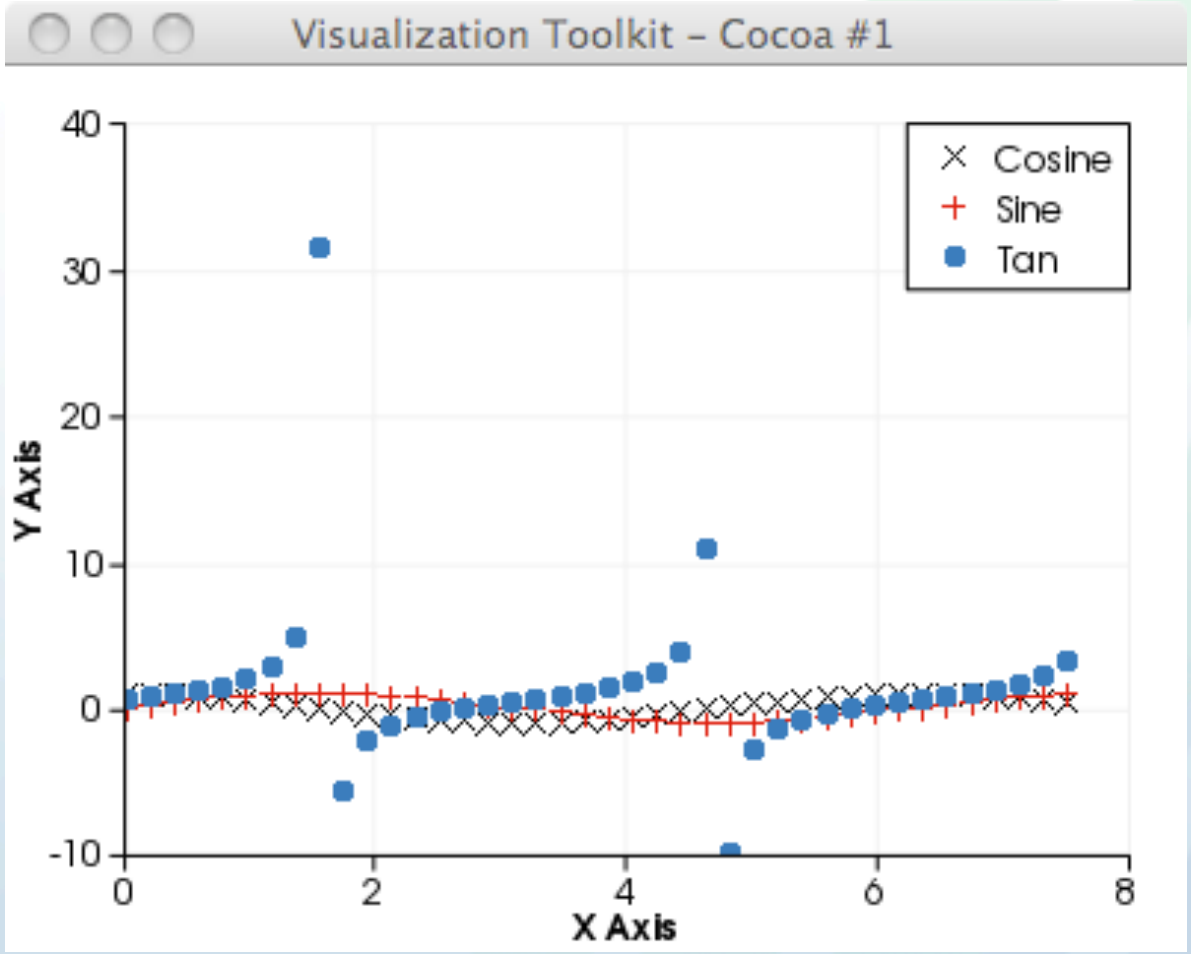
Kitware

Scatter Plots – Marker Styles

- Instead of the previous code – different styles

```
vtkPlot *points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 1);
vtkPlotPoints::SafeDownCast(points)
  ->SetMarkerStyle(vtkPlotPoints::CROSS);
points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 2);
vtkPlotPoints::SafeDownCast(points)
  ->SetMarkerStyle(vtkPlotPoints::PLUS);
points = chart->AddPlot(vtkChart::POINTS);
points->SetInput(table.GetPointer(), 0, 3);
```

Scatter Plot Symbols



Kitware

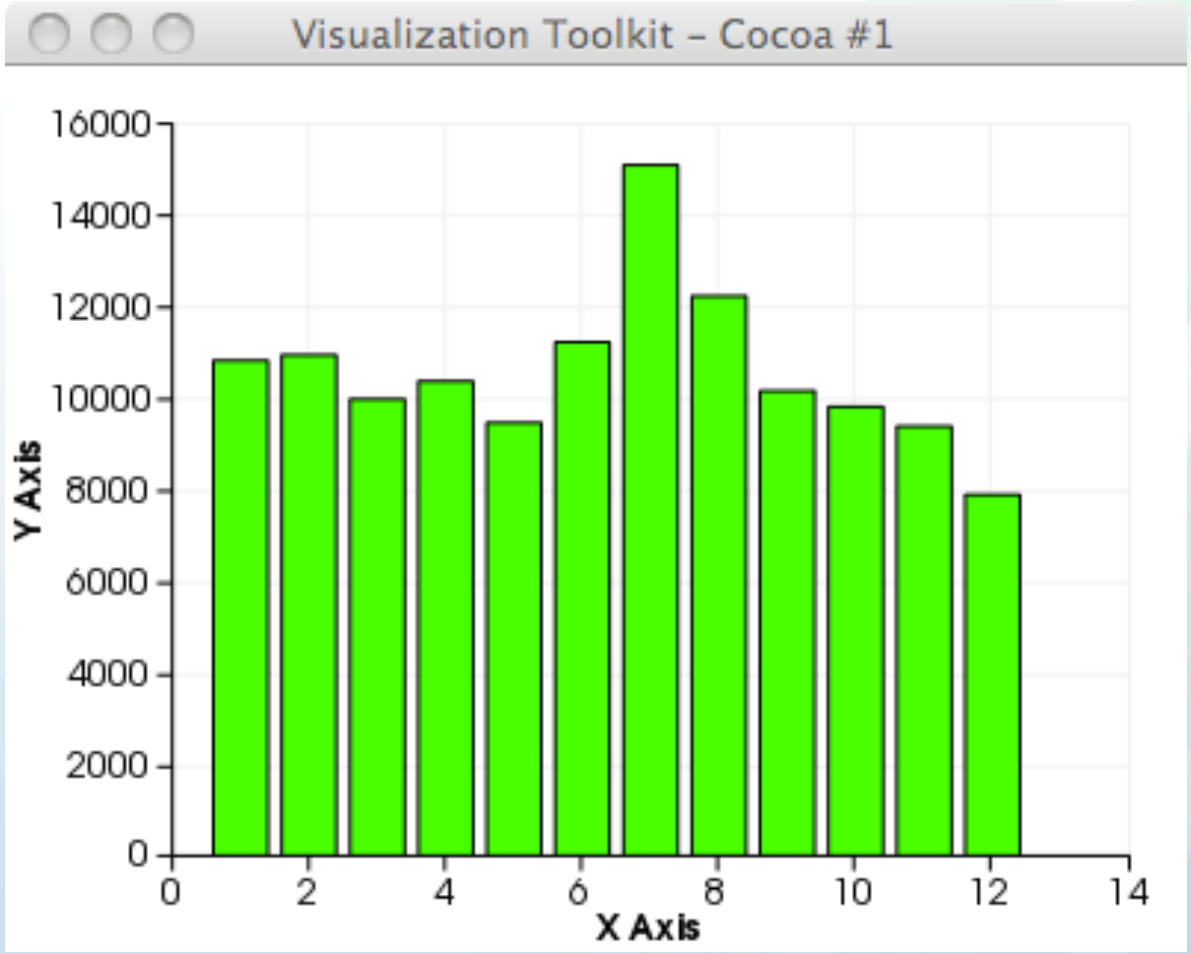
Bar Graphs

- Bar graphs are very similar again…
- If no x source is supplied assume 0, 1, 2

```
vtkPlot *bar = chart->AddPlot(vtkChart::BAR);
plot->SetInput(table.GetPointer(), 0, 1);
plot->SetColor(0, 255, 0, 255);
```

Kitware

Simple Bar Graph



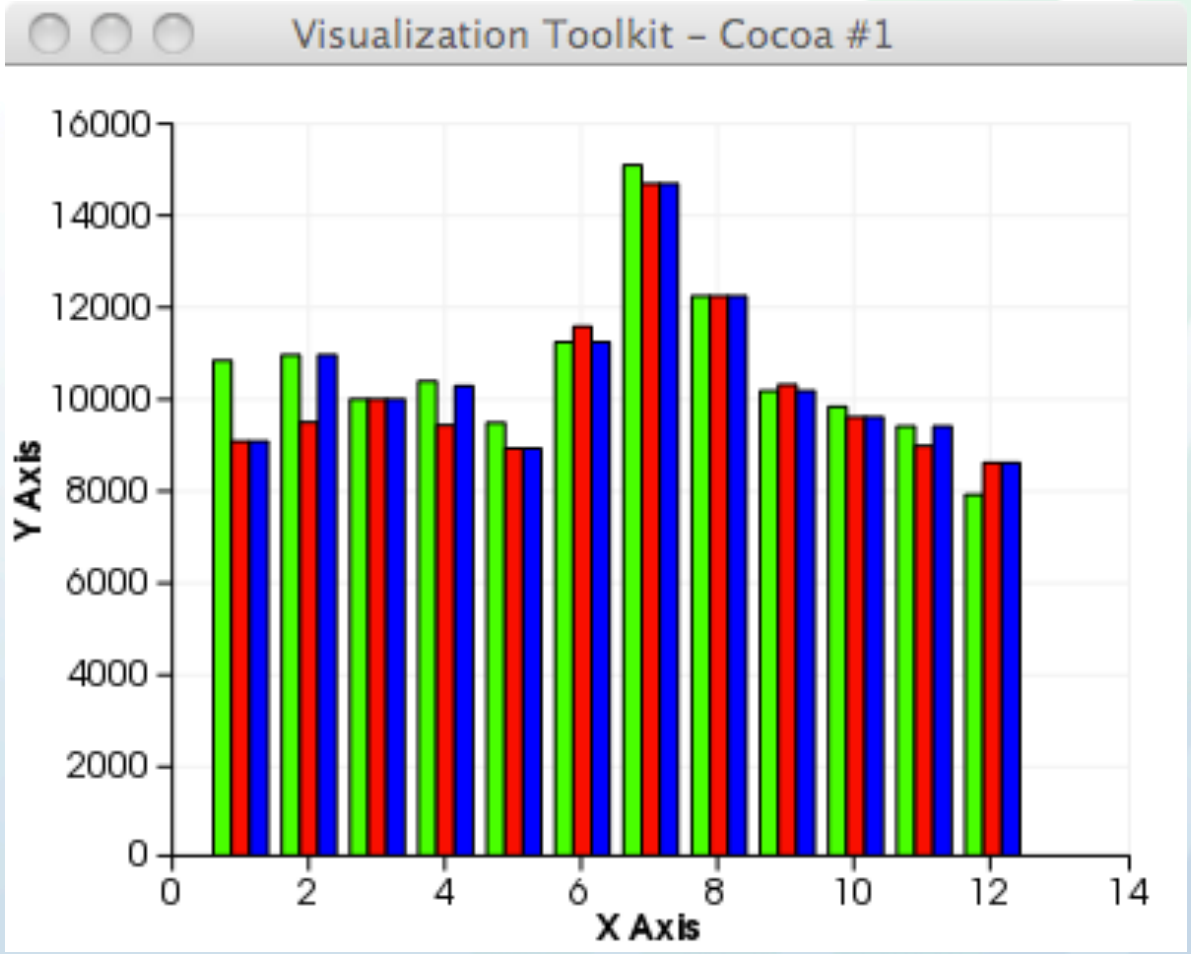
Kitware

Grouped Bar Graphs

- Add multiple bar graphs to a chart to group

```
bar = chart->AddPlot(vtkChart::BAR);
bar->SetInput(table.GetPointer(), 0, 2);
bar->SetColor(255, 0, 0, 255);
bar = chart->AddPlot(vtkChart::BAR);
bar->SetInput(table.GetPointer(), 0, 3);
bar->SetColor(0, 0, 255, 255);
```

Grouped Bar Graphs



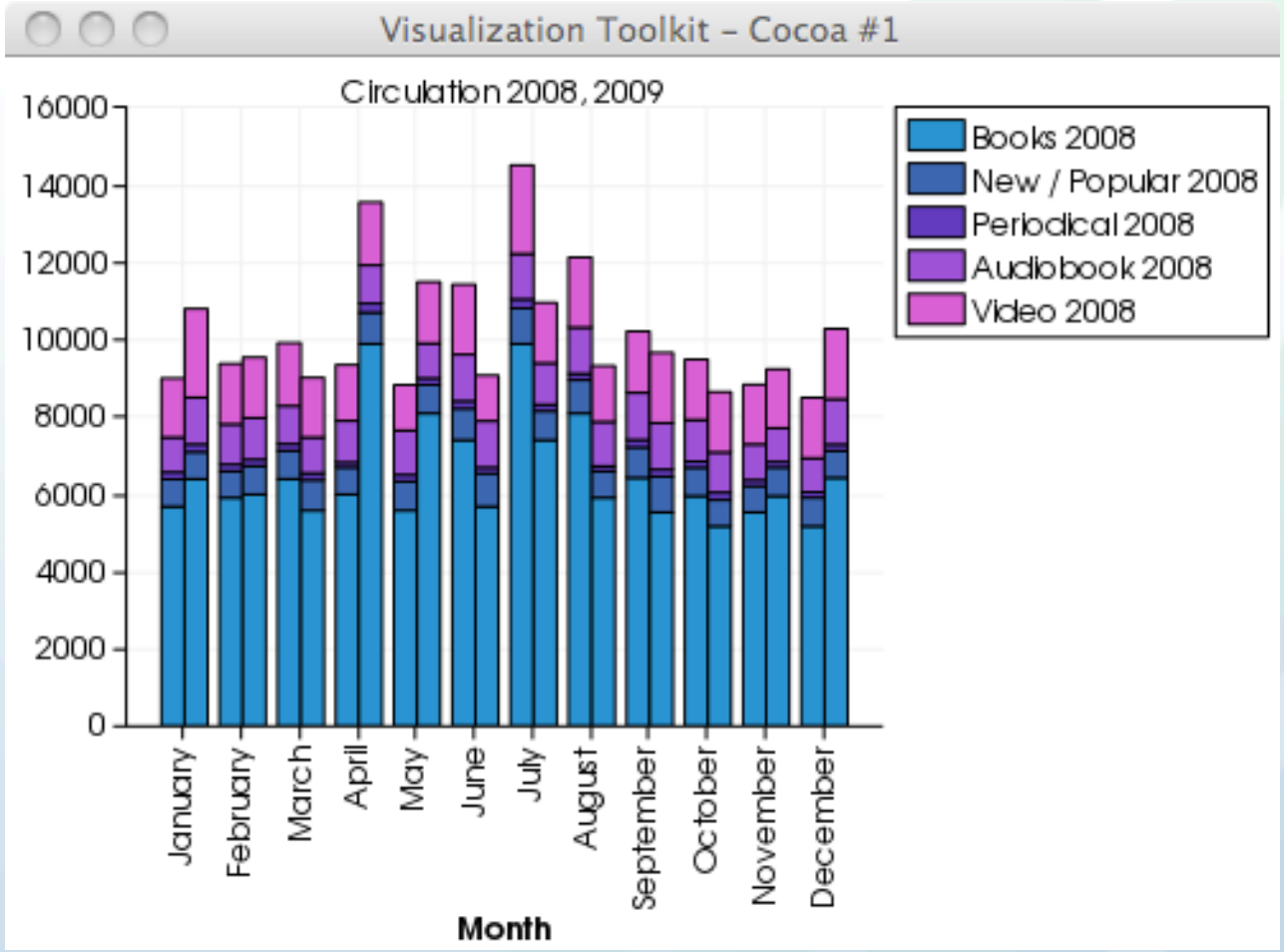
Kitware

Stacked Bar Charts

- Alternatively, you can stack bar charts

```
vtkPlotBar *bar;
bar = vtkPlotBar::SafeDownCast(
  chart->AddPlot(vtkChart::BAR));
bar->SetColorSeries(colorSeries1);
bar->SetInput(table, "Month", "Books 2008");
bar->SetInputArray(2,"New / Popular 2008");
bar->SetInputArray(3,"Periodical 2008");
bar->SetInputArray(4,"Audiobook 2008");
bar->SetInputArray(5,"Video 2008");
```

Stacked, Grouped Bars



Kitware

Pie Charts – Data

```
vtkNew<vtkTable> table;
vtkNew<vtkIntArray> arrData;
vtkNew<vtkStringArray> labelArray;
arrData->SetName("2008 Circulation");
for (int i = 0; i < numberOfItems; i++)
  {
  arrData->InsertNextValue(data[i]);
  labelArray->InsertNextValue(labels[i]);
  }
table->AddColumn(arrData.GetPointer());
// Create a color series to use with our stacks.
vtkNew<vtkColorSeries> colorSeries;
colorSeries->SetColorScheme(vtkColorSeries::WARM);
```

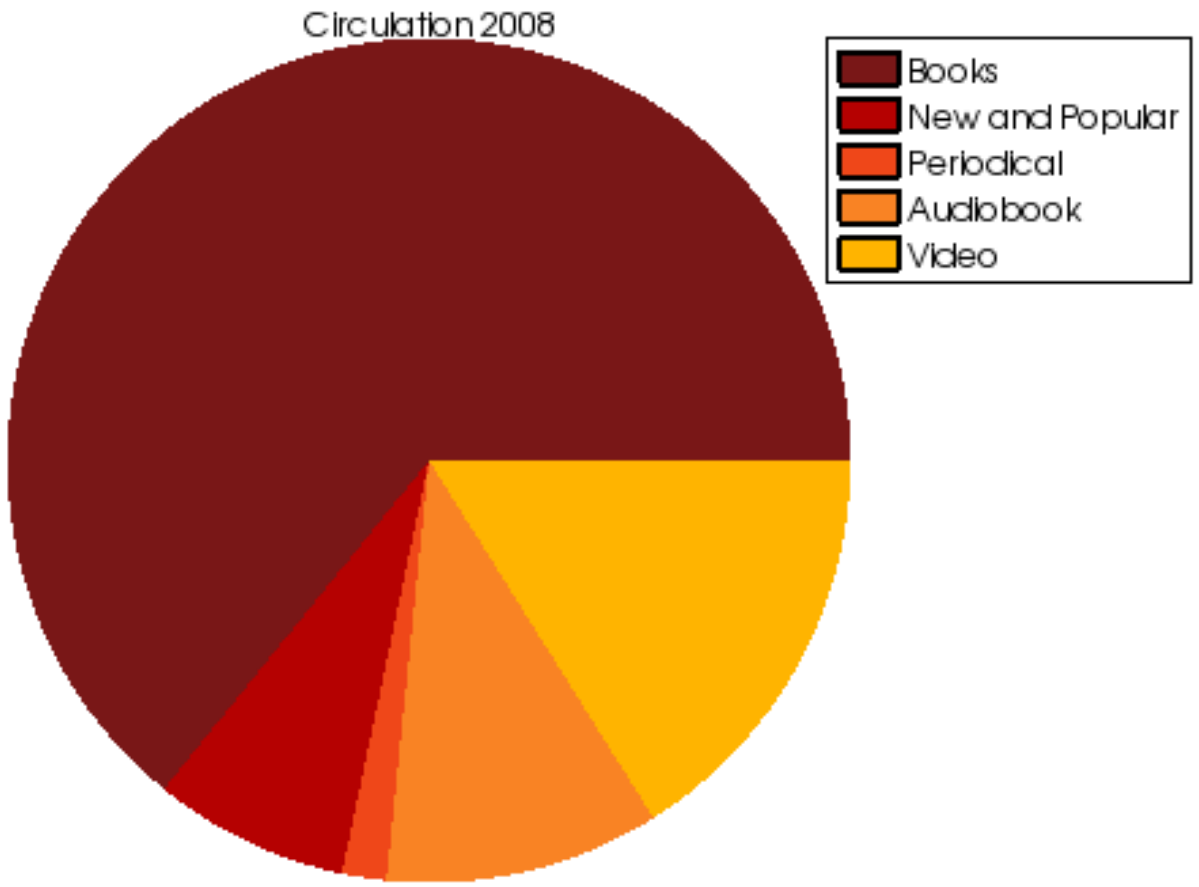
Kitware

Pie Charts – Data

```
// Create a pie chart object
vtkNew<vtkChartPie> chart;
view->GetScene()->AddItem(chart.GetPointer());
// Add the pie chart
vtkPlotPie *pie =
  vtkPlotPie::SafeDownCast(chart->AddPlot(0));
pie->SetColorSeries(colorSeries.GetPointer());
pie->SetInput(table.GetPointer());
pie->SetInputArray(0, "2008 Circulation");
pie->SetLabels(labelArray.GetPointer());
chart->SetShowLegend(true);
chart->SetTitle("Circulation 2008");
```

Kitware

Pie Chart



Kitware

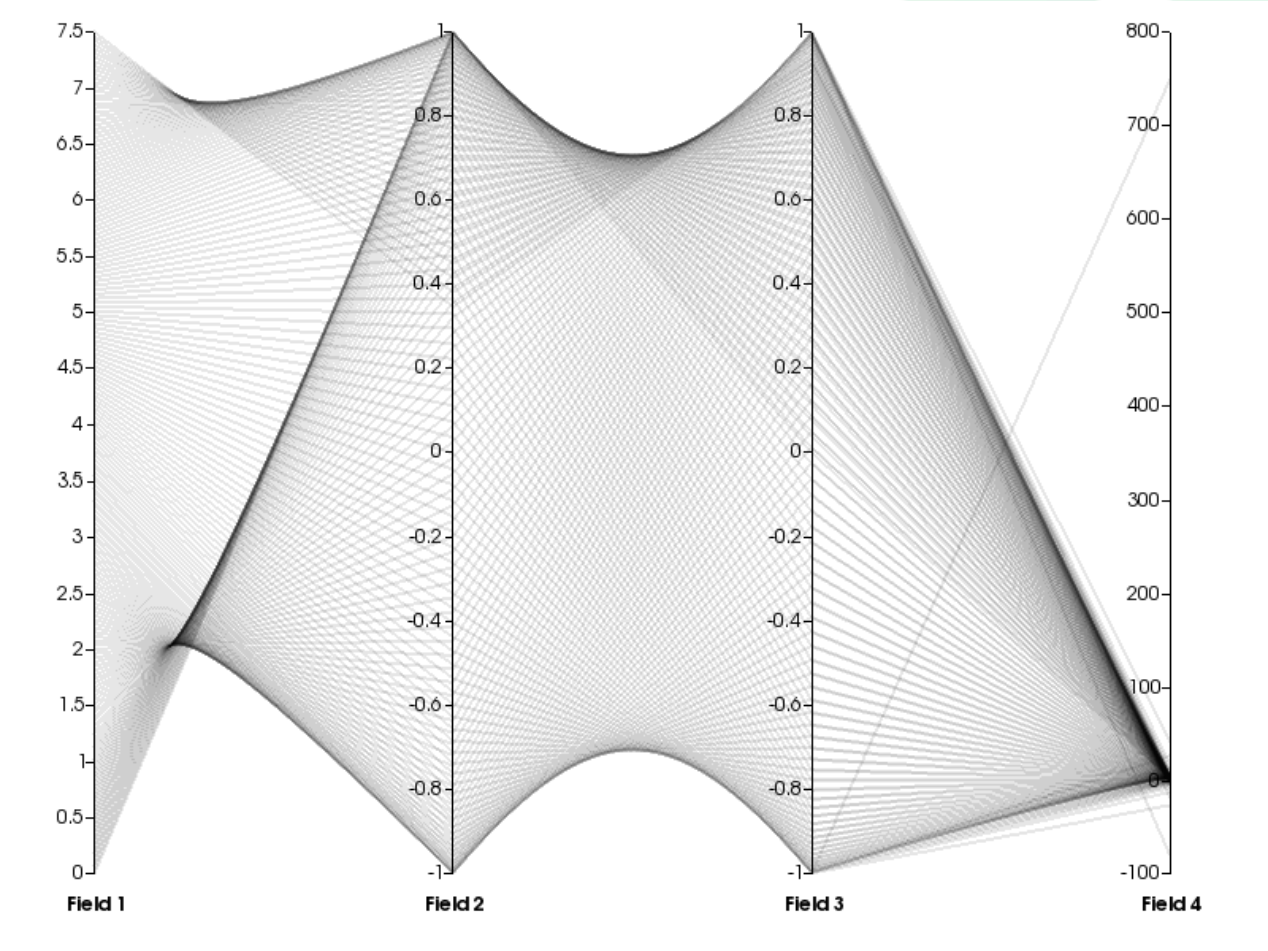
Parallel Coordinates

- Very good for multidimensional data
- Set visibility of columns – become axes
- Given a table with 4 columns

```
vtkNew<vtkChartParallelCoordinates> chart;
view->GetScene()->AddItem(chart.GetPointer());
// Set the input - only one plot per chart for PC.
// Column names used as axis titles.
chart->GetPlot(0)->SetInput(table.GetPointer());
```

Kitware

Parallel Coordinates



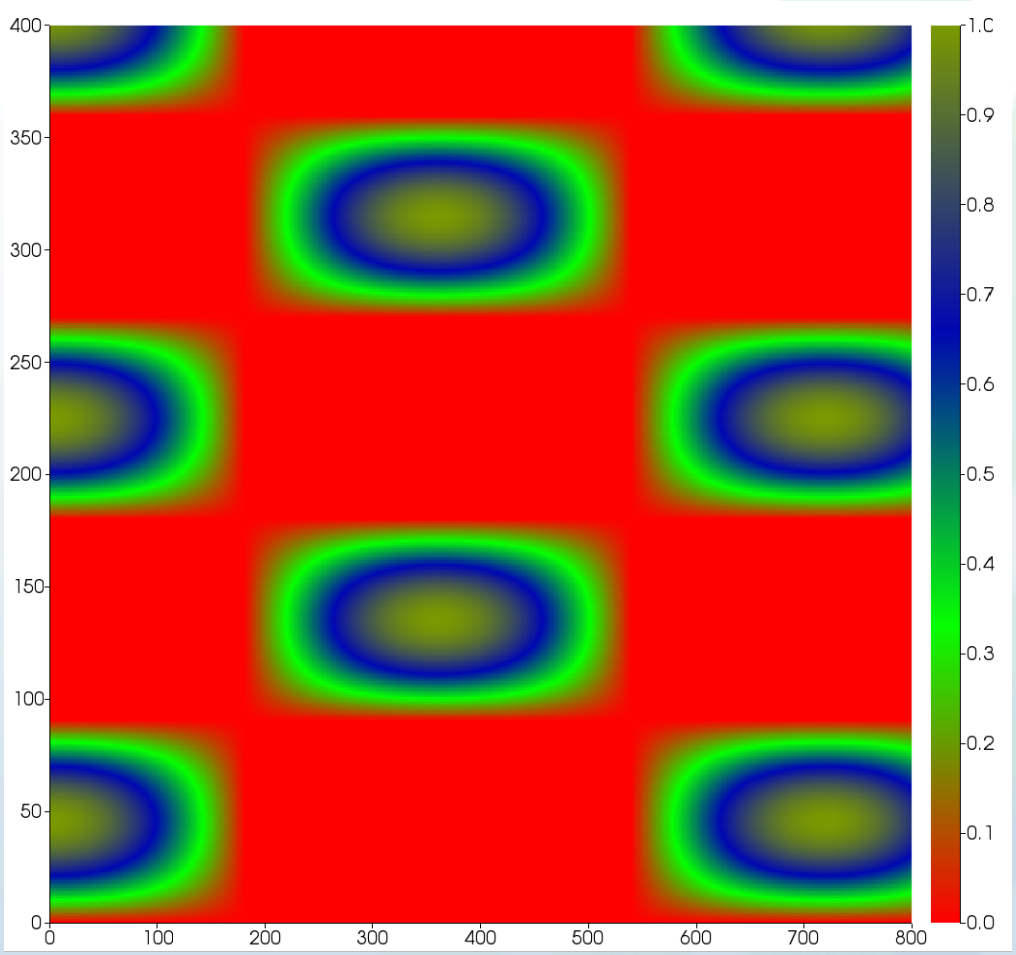
Kitware

2D Histogram

- Uses a vtkImageData as input
- Maps scalars to colors using transfer function

```
vtkNew<vtkChartHistogram2D> chart;
view->GetScene()->AddItem(chart.GetPointer());
vtkNew<vtkImageData> data;
// Set as input to the chart
chart->SetInput(data.GetPointer());
vtkNew<vtkColorTransferFunction> tf;
chart->SetTransferFunction(tf.GetPointer());
```

2D Histogram/Flood Plot



Kitware

Mixing Bars and Scatter Plots

- A vtkChartXY can have many plot types on it

```
vtkPlot *plot = 0;
plot = chart->AddPlot(vtkChart::BAR);
plot->SetInput(table.GetPointer(), 0, 1);
plot->SetColor(0, 255, 0, 255);
plot = chart->AddPlot(vtkChart::POINTS);
plot->SetInput(table.GetPointer(), 0, 2);
plot->SetColor(255, 0, 0, 255);
chart->SetPlotCorner(plot, 1); // Use bottom-right
vtkPlotPoints::SafeDownCast(plot)
  ->SetErrorArray(deviation.GetPointer());
```

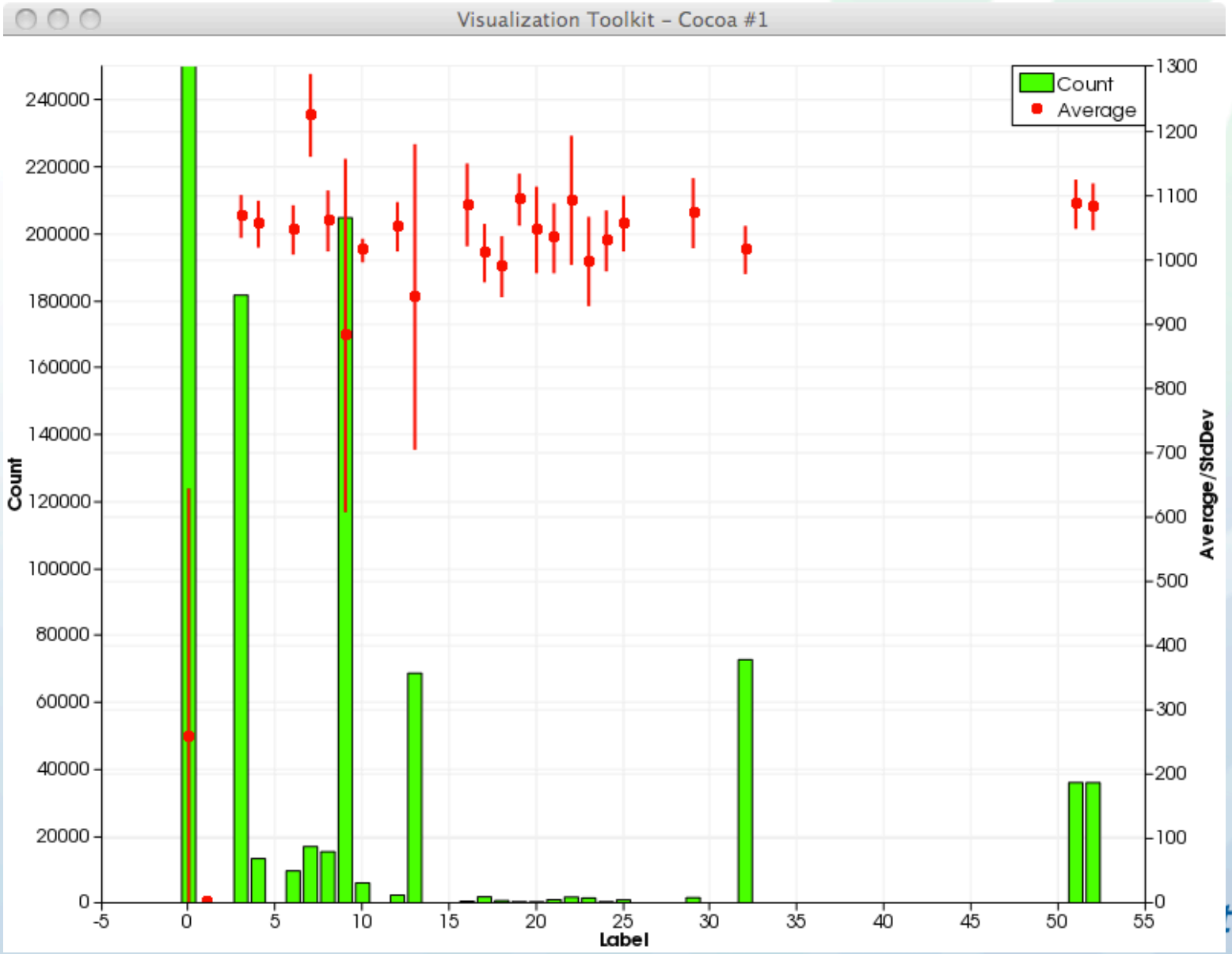
Kitware

Mixing Bars and Scatter Plots

- Some specialization of the chart axes

```
chart->GetAxis(vtkAxis::LEFT)->SetTitle("Count");
chart->GetAxis(vtkAxis::LEFT)->SetRange(0, 250000);
chart->GetAxis(vtkAxis::LEFT)
  ->SetBehavior(vtkAxis::FIXED);
chart->GetAxis(vtkAxis::BOTTOM)->SetTitle("Label");
chart->GetAxis(vtkAxis::RIGHT)
  ->SetTitle("Average/StdDev");
// Show the legend too
chart->SetShowLegend(true);
```

Mixing Bars, Scatter and Errors



Controlling Interaction

- API available to change behaviors

```
// Use the right mouse button for panning
chart->SetActionToButton(vtkChart::PAN,
  vtkContextMouseEvent::RIGHT_BUTTON);

// Notify on left clicks on points in a scatter plot
// This will generate an event, supplying index of
// the clicked point
chart->SetClickActionToButton(vtkChart::NOTIFY,
  vtkContextMouseEvent::LEFT_BUTTON);
```

Kitware

Linked Selections

- Use the vtkAnnotationLink class
- Multiple charts can share a selection
  - Selections are row based
  - Events sent when selection changes

```
vtkNew<vtkAnnotationLink> link;
chart->SetAnnotationLink(link.GetPointer());
parallel->SetAnnotationLink(link.GetPointer());
```

Kitware

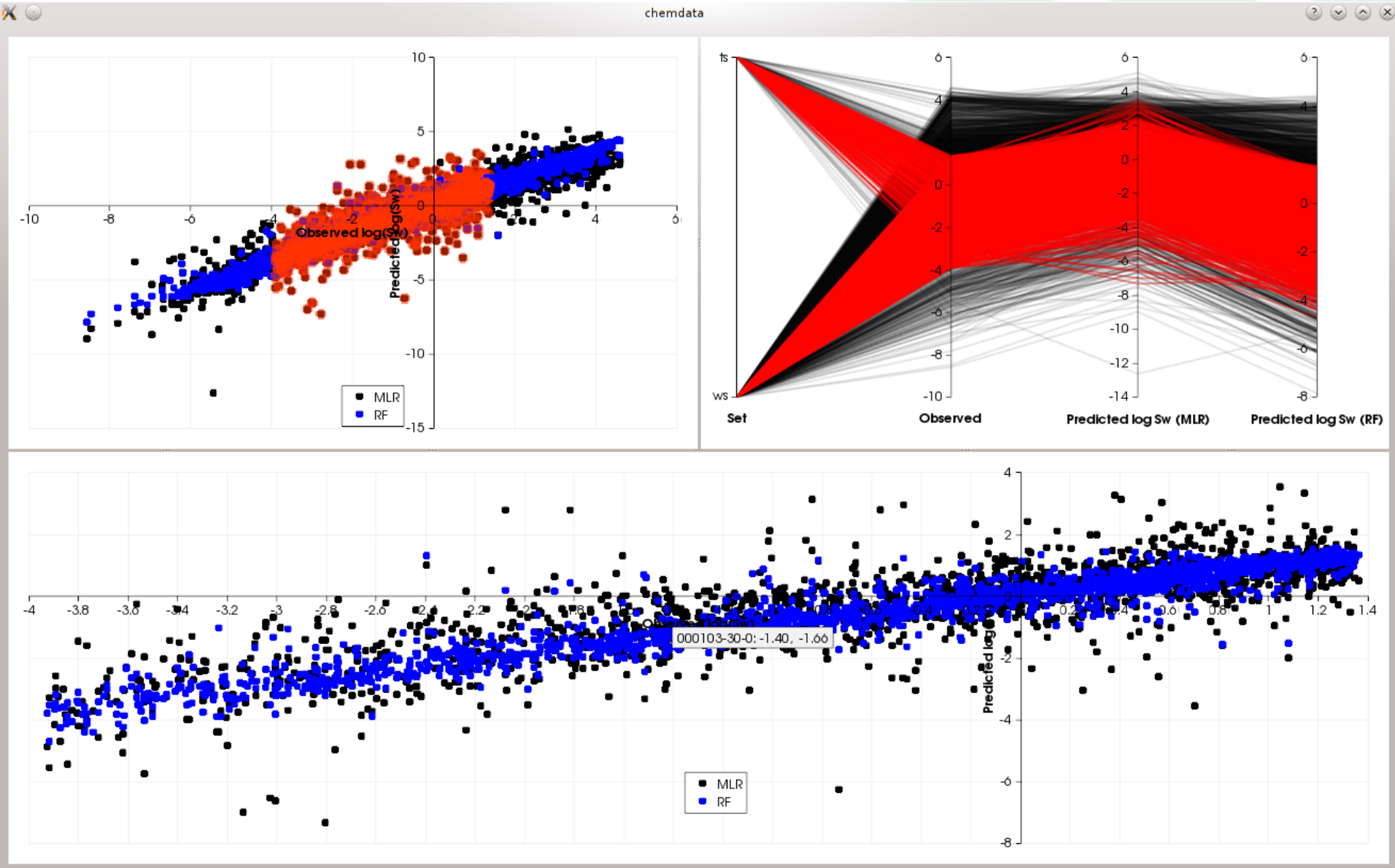
Plotting Selected Subsets of Data

- Sometimes one chart affects another
  - Select relevant data in one chart
  - Plot selected points in another

```
m_extract->SetInputConnection(0,
  table->GetProducerPort());
m_extract->SetInputConnection(1,
  vtkSelection::SafeDownCast(
  link->GetOutputDataObject(2))->GetProducerPort());
m_extract->Update();
vtkTable *selection = m_extract->GetOutput();
scatter = chart2->AddPlot(vtkChart::POINTS);
scatter->SetInput(selection, 2, 3);
```

Kitware

Multiple Charts, Linked Selections



Custom Point Labels

- Can set a column with indexed labels
- Custom tooltip string with substitution

```
// Set a vtkStringArray with labels
points->SetIndexedLabels(labels.GetPointer());
// Standard substitutions in API documentation
// '%i' The IndexedLabels entry for the plot element
// '%l' The value of the plot's GetLabel()
points
  ->SetTooltipLabelFormat("%i from %l (%x, %y)");
```

Kitware

Charts from VTK Python

- Python, TCL and Java wrapped
- Heavily used by several VTK Python users

```
view = vtk.vtkContextView()
view.GetRenderWindow().SetSize(400,300)
chart = vtk.vtkChartXY()
view.GetScene().AddItem(chart)
# Create a table with some points in it
table = vtk.vtkTable()
arrX = vtk.vtkFloatArray()
arrX.SetName("X Axis”)
arrC = vtk.vtkFloatArray()
arrC.SetName("Cosine")
```

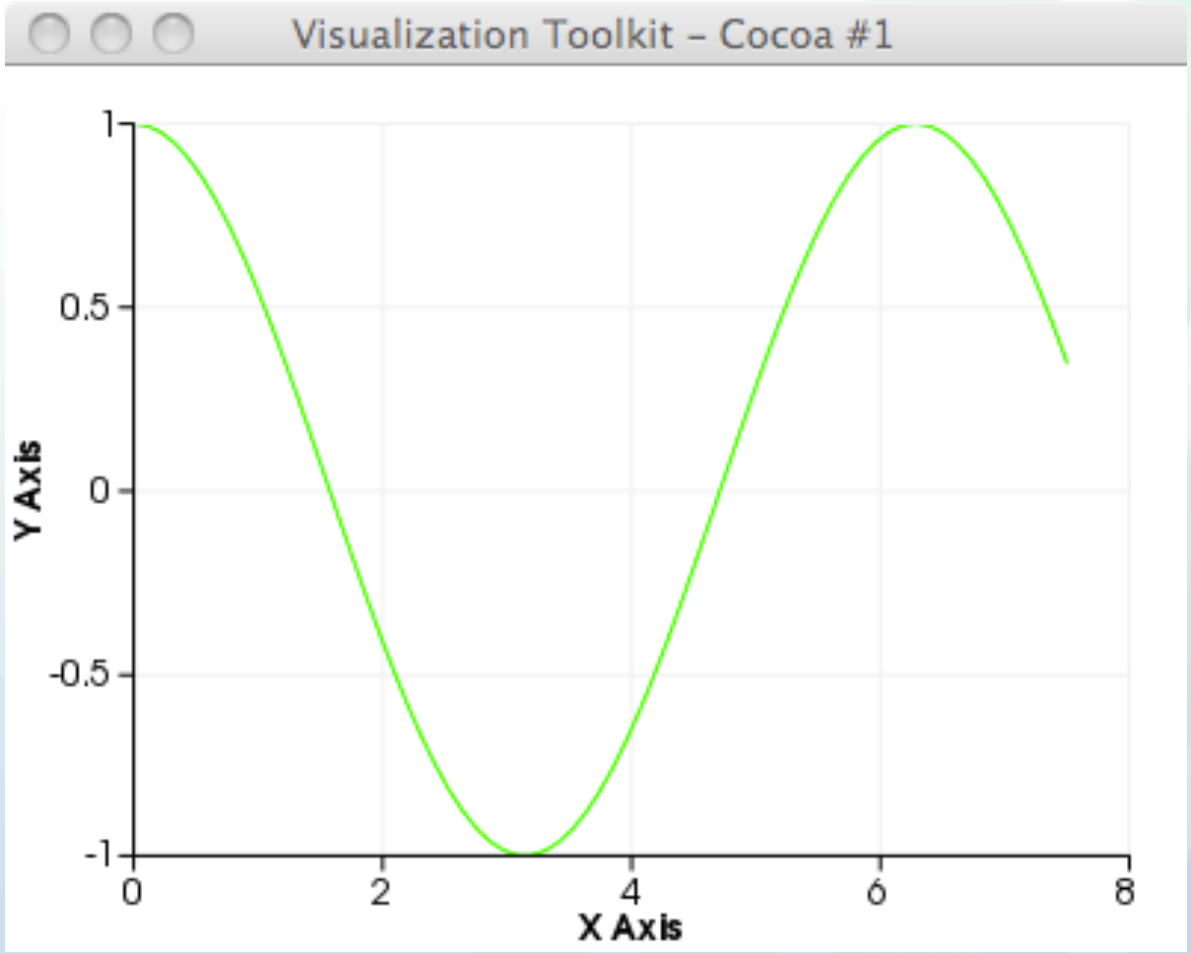
Kitware

Line Plot from Python

```
numPoints = 69
inc = 7.5 / (numPoints - 1)
for i in range(0, numPoints):
  arrX.InsertNextValue(i*inc)
  arrC.InsertNextValue(math.cos(i * inc) + 0.0)
table.AddColumn(arrX)
table.AddColumn(arrC)
# Now add the line plots
line = chart.AddPlot(0)
line.SetInput(table,0,1)
line.SetColor(0,255,0,255)
line.SetWidth(1.0)
```

Kitware

Plot of Cosine using Python



Kitware

NREL VTK Charts Library

- Developed for NREL
- C++ STL style API on top of charts
- Integration with Qt, proxies emit signals
- Supports non-Qt build – no X server required
- Wrapped in Ruby using SWIG
- Thin wrapper around VTK charts

NREL VTK Chart API Example

```
std::vector<float> x_floats(numPoints);
std::vector<float> sine(numPoints);
vtkCharts::Chart chart();
vtkCharts::Plot sinePlot =
  chart.addPlot(x_floats, sine, "Sine");
sinePlot.setLineStyle(vtkCharts::Plot::DOTTED);
sinePlot.setMarkerStyle(vtkCharts::Plot::CROSS);
sinePlot.setMarkerSize(20);
```

Kitware

NREL VTK Chart API Example

```
std::vector<vtkCharts::Color3ub> colors;
colors.push_back(vtkCharts::Color3ub(255,0,0));
colors.push_back(vtkCharts::Color3ub(0,255,0));
colors.push_back(vtkCharts::Color3ub(0,0,255));
chart.setColors(colors);
sinePlot
  .setColor(vtkCharts::Color4ub(200, 10, 255));
chart.axis(vtkCharts::Axis::LEFT)
  .setTitle("My axis title!");
chart.axis(vtkCharts::Axis::BOTTOM)
  .setTitle("My other axis title...");
chart.setShowLegend(true);
```

Kitware

Future Work in VTK Charts

- Overlaid charts on 3D not interactive…yet
- Further acceleration using more OpenGL 2 (optionally – test at runtime)
- Rendering abstracted from the start
  - High quality SVG backend for publication export
- New chart types
- Enhanced interaction framework
- Select on both columns and rows
- VTK modularization – depend on a smaller VTK!
- Mobile platforms – OpenGL ES 2: iOS, Android

Kitware