

Powerful Graph Customizations to Effectively Communicate Your Results

CASEY A. VOLINO

volinoca@corning.com

Engineering Associate, Sr. Statistician

Corning Incorporated

JMP offers several ways to customize charts and reports that enhance our ability to present results to others. This paper will discuss some of the techniques used to invoke these customizations. While some of these techniques involve using the JMP Scripting Language (JSL), others do not. All data examples are either part of the JMP example data files, or are non-proprietary data sets that can be made freely available. Readers are encouraged to try these techniques on their own data where possible.

Key Words: reference lines, specification limits, distributions, Weibull, polygons, JMP® Example Data, JSL

Overview

JMP is an excellent software package for exploring data, analyzing data, and creating presentation-quality and interactive graphs to exhibit results. As with all powerful software, there is a learning curve associated with the software. It is a goal of this paper to help guide you through basic to more advanced tasks associated with customizing your graphed results.

This paper is broken up into three main sections. The first section will cover the basics of customizing graphs without using the JMP Scripting Language (JSL). These customizations in JMP can be easily implemented by setting column properties and by using simple point-and-click techniques within graphs. The second section covers some intermediate ways involving simple JSL commands to customize graphs. With these techniques you can easily accomplish tasks such as adding first-principles curves to graphs, adding historical distribution curves to new histograms, and adding shaded regions to graphs to help readers focus on important outcomes. While it's helpful to have some background in JSL to fully leverage graph customizations in this section, it is not necessary. Basic familiarity with adding formulas to columns should be sufficient background. The third section shows a small example of the endless possibilities available when creating JSL-only graphics. Nearly limitless customizations as well as user interactivity are possible. This type of graph creation can be used to make interactive educational and engineering tools among other applications. In addition a section details resources for learning more.

1. Graph Customizations (Without Scripting)

JMP offers many easy graph customizations that help convey a message about data and results to others. This is important, because after we finish collecting data and performing careful analyses, we need to show what we've learned in some effective way to colleagues, management, and customers. Often this is as easy as using the built-in graphics offered by some JMP analysis platform, but it's often necessary to tweak those graphs to clarify the message and help focus our intended reader's attention.

The following example is a randomly generated data set consisting of three Yield values per day for a year. We often encounter this type of data when examining Yields per shift. In order to show several of the powerful point and click graph customizations available, I'll take more steps than I usually would. Let's begin by using one of the workhorse platforms of JMP, "Fit Y by X". After plotting Yield (Y) against Date (X), see Figure 1, it's clear that a step-change is evident in the plotted raw data. Let's suppose we wish to narrow in on an approximate change point in time and show our findings graphically. From this initial graph, we can see that between the beginning of March and the end of June there was an increase in yield.

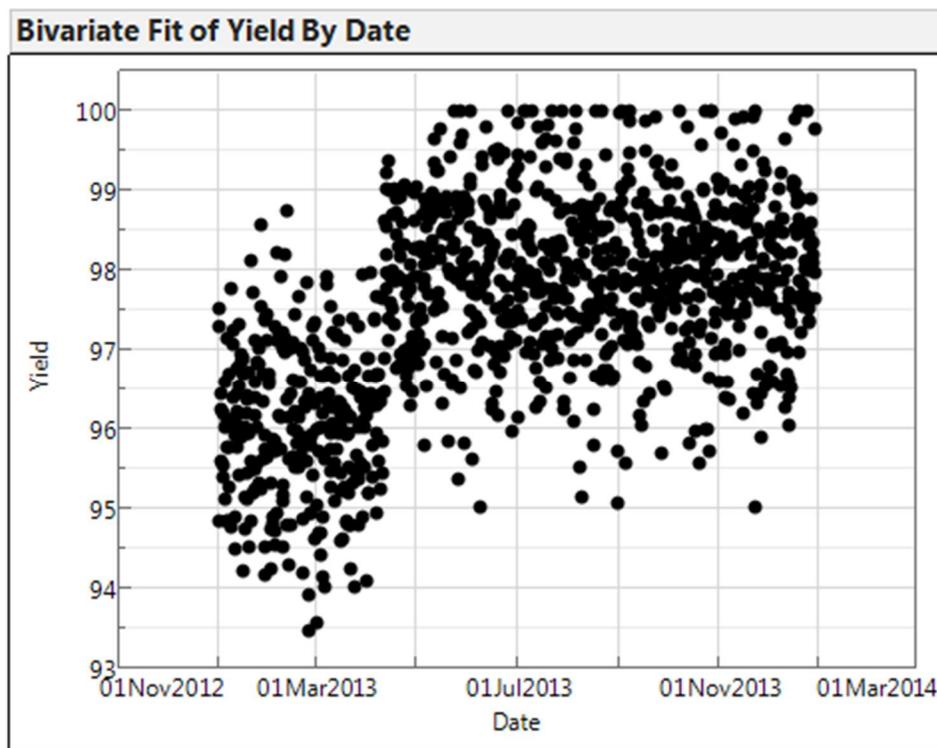


Figure 1 Raw Yield Data over Time

When many data points are plotted it's sometimes difficult to make out the central tendency of the data over time. In JMP we can easily adjust the transparency of the plotted data points which helps to show this central tendency or average. To adjust the transparency of the plotted points, we just right-click the plot area and select 'Transparency' from the pop-up menu.

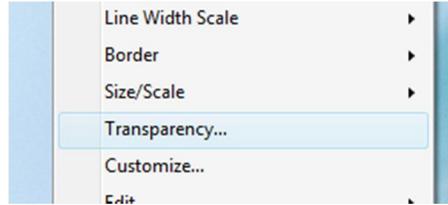


Figure 2 Part of the Pop-up Menu Produced by Right-Clicking the Plot Area

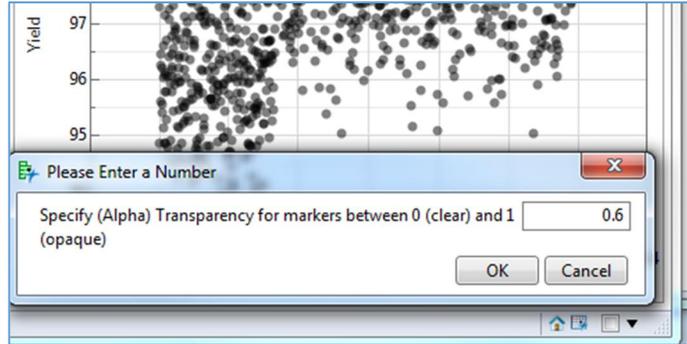


Figure 3 Transparency Settings

This brings up a window that allows you to enter a transparency value between 0 (clear) and 1 (opaque). The default value is 1 so that the points plotted are a solid color and as such they obscure any points plotted behind them. By adjusting the transparency value less than 1, we can now see that Yield data has a central tendency that changes subtly over time.

By adding a new ordinal column called 'Month' and giving it the formula "Month(:Date)", this extracts the month value (from 1 to 12) from the "Date" column. Furthermore we can set the Value Labels property for the 'Month' column to change each of the month numbers from 1 to 12 to an abbreviated month column such as 1=Jan, 2=Feb, and so on.

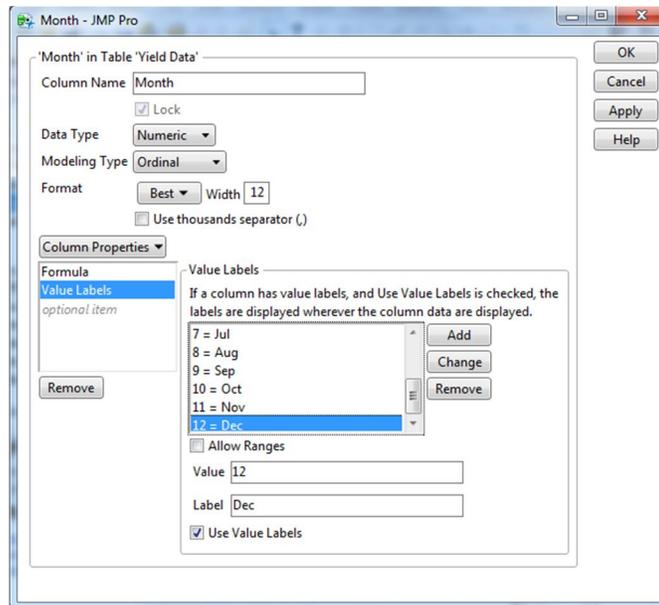


Figure 4 Value Labels Dialogue

Going back to our graph, we can color our points by 'Month' and add a legend by right-clicking the graph again and selecting the 'Row Legend' item on the pop-up menu. By selecting the 'Month' column and clicking OK it appears that the step change in yields occurred about midway through April. The resultant graph is shown in Figure 6. Note that I also selected the 'Apr' legend item along the right-hand side of the graph and changed the markers to "+" to distinguish them from the other months which have solid circles. This last change is helpful if you suspect that any of your readers may view your graphs printed out in black and white.

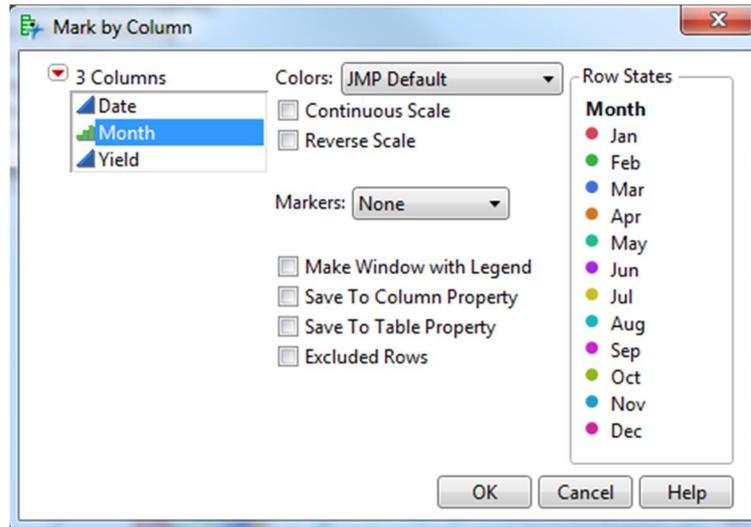


Figure 5 Legend Options Dialogue

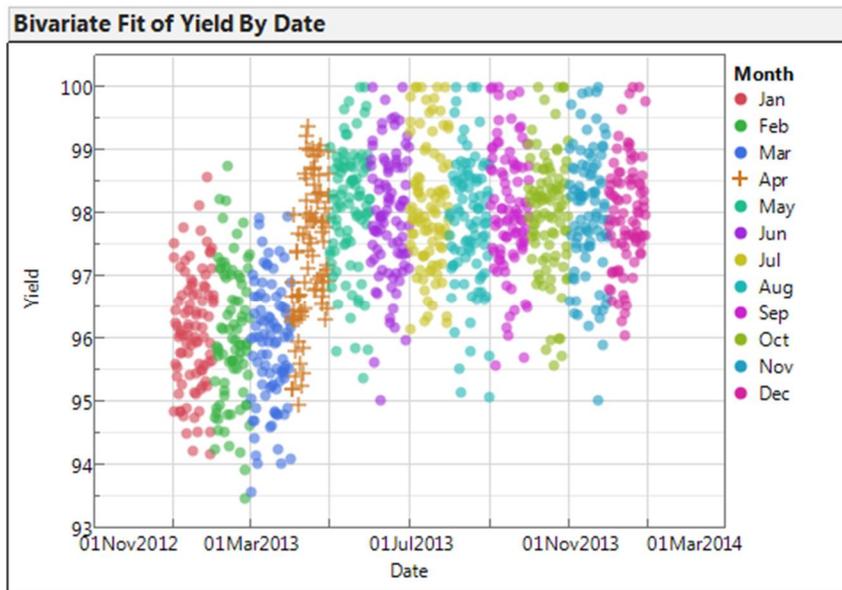


Figure 6 Yield Over Time with Legend and Transparency Settings

Sometimes when plotting data over time it's helpful to change the x-axis values to something more readable. One way to do this is to right-click the x-axis and select "Axis Settings" from the pop-up menu.

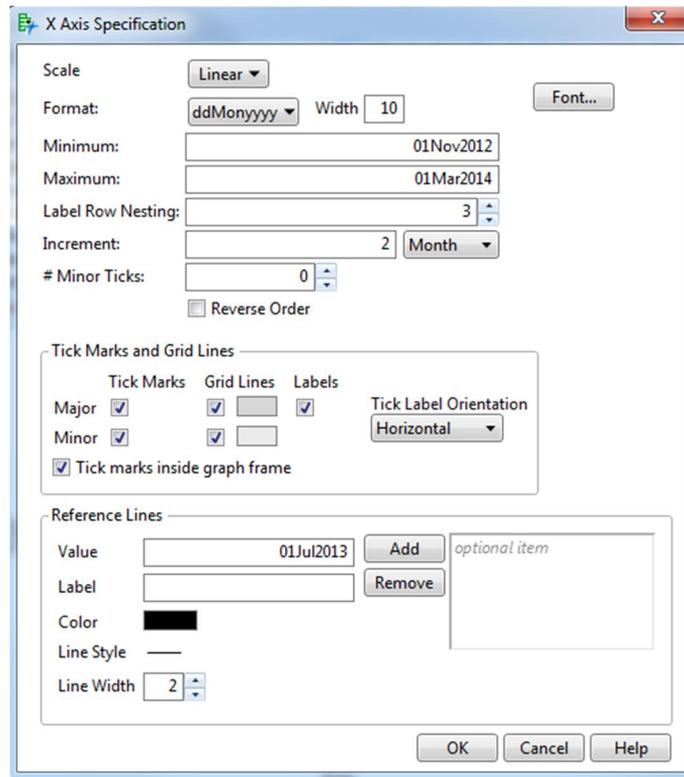


Figure 7 Using Label Row Nesting to split apart dates on axes.

Label Nesting Value	Appearance of Date Axis with ddMonyyyy Format
1	<p>01Mar2013 01May2013 01Jul2013 01Sep2013 01Nov2013</p> <p>Date</p>
2	<p>01Mar 01May 01Jul 01Sep 01Nov</p> <hr/> <p>2013</p> <p>Date</p>
3	<p>01 01 01 01 01</p> <hr/> <p>Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec</p> <hr/> <p>2013</p> <p>Date</p>

Dates formatted as ddMonyyyy have three parts (month, day, year) that can be split apart for easier interpretation when they are used as axis values. By editing the axis settings and then changing the “Label Row Nesting” from a setting of 1 to a setting of 3,

the date values will be plotted as 3 nested labels on the x-axis. Since we have several months of data to show, a cleaner representation of the Date x-axis might be to apply the m/y format with 2 levels of label nesting.

Label Nesting Value	Appearance of Date Axis with m/y Format																
1	11/2012	02/2013	04/2013	06/2013	08/2013	10/2013	12/2013	02/2014									
2	11	12	01	02	03	04	05	06	07	08	09	10	11	12	01	02	03

To show the general trend over time we can also add a spline fit and increase the “stiffness” value until the fitted line appears to be fairly smooth. (See Figure 8)

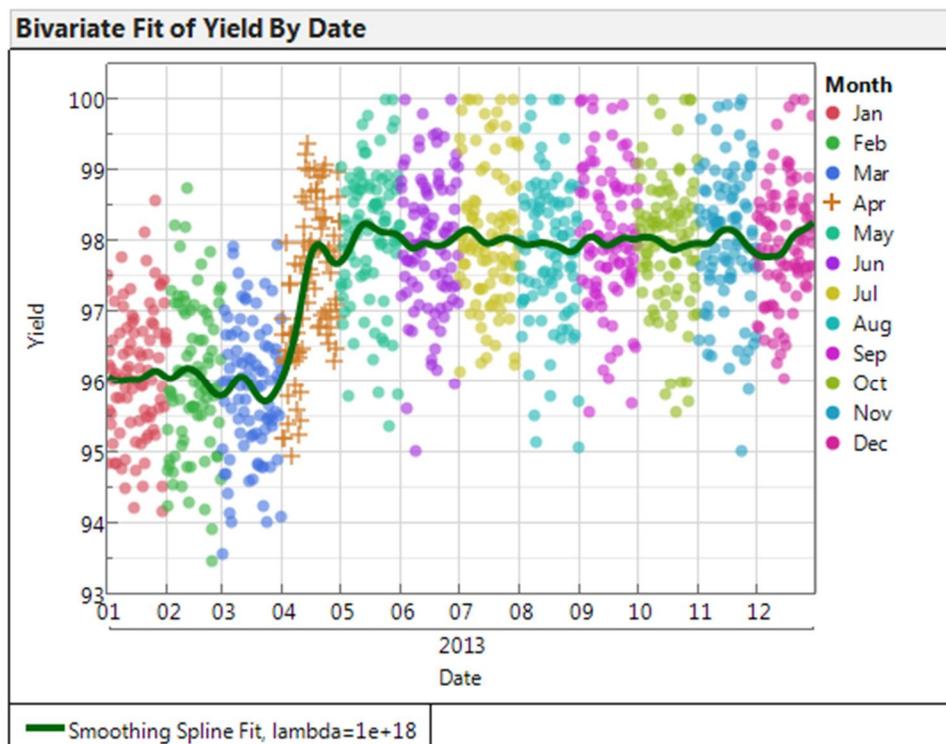
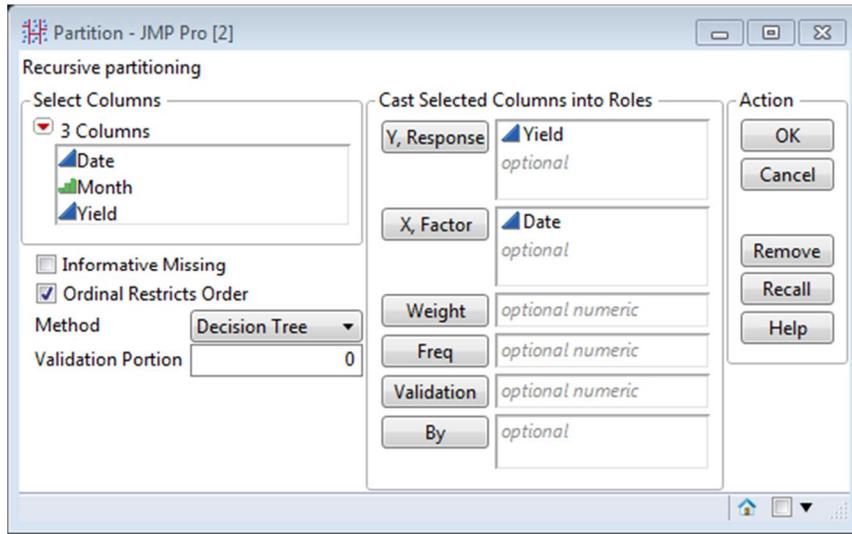


Figure 8 Points colored by month with spline fit.

It would be helpful as we comb through historical data to determine the cause of the shift if we could narrow the window of time in which to look. One technique we could use would be to perform recursive partitioning of the Yield data with Date as the predictor.

We can specify that we want to keep the date values contiguous by checking the box “Ordinal Restricts Order” in the Partition platform dialog.



After a single partition, we have an estimate of 11Apr2013 for the day of the process shift.

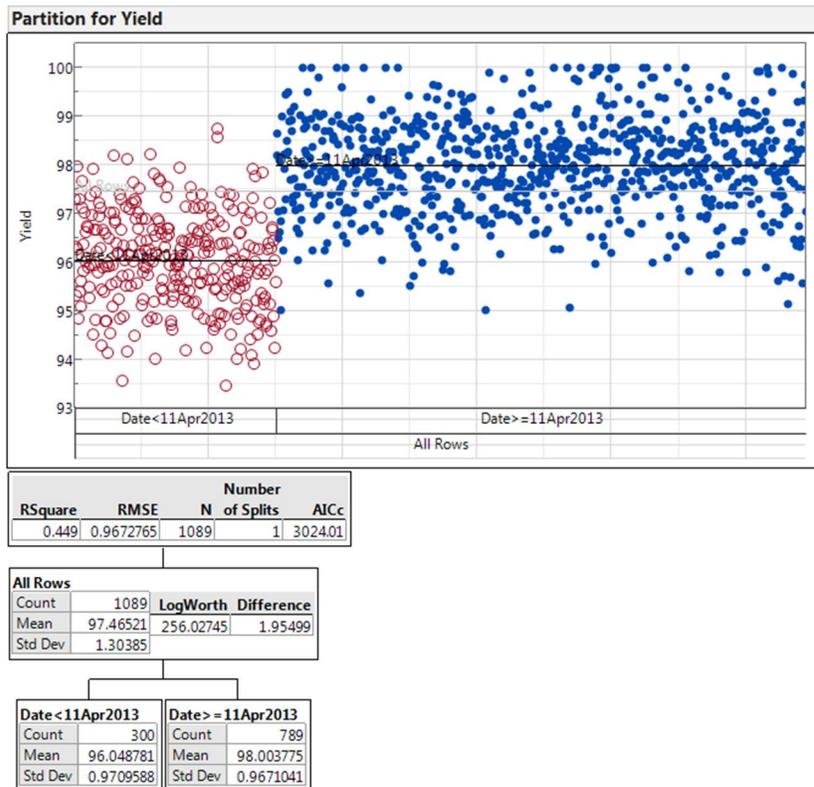
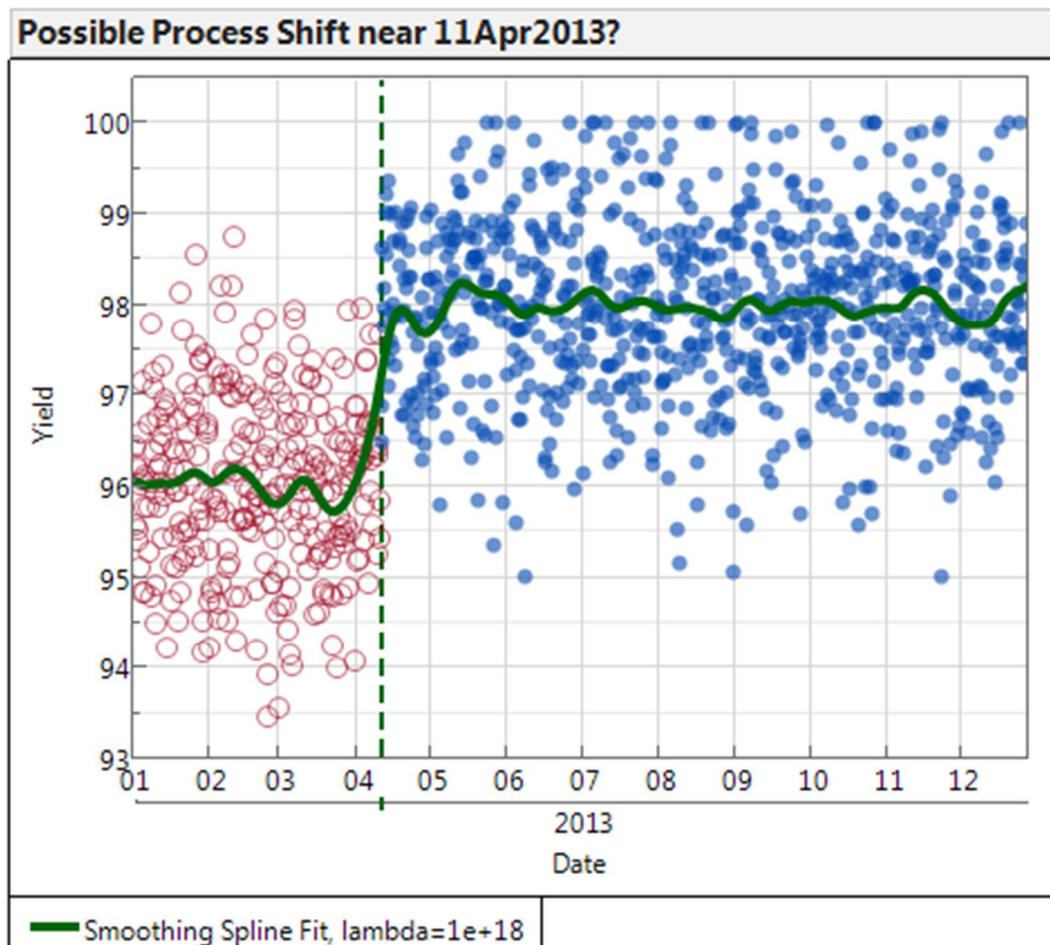
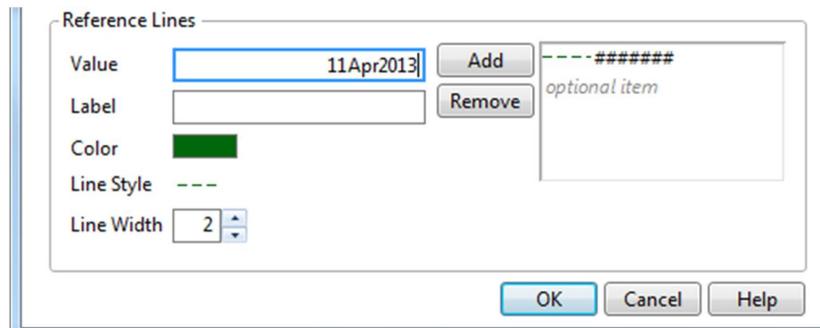


Figure 9 Partition of Yield by Date

With an estimate of 11Apr2013 in hand, we can add a reference line by right-clicking the x-axis, selecting “Axis Settings”, and entering 11Apr2013. This yields the final plot we might show to others.



Sometimes our data consist of ordinal values, such as scores on a Likert scale. By default JMP sorts text values alpha-numerically which can lead to a situation as shown in

Figure 10. There are several ways to make the x-axis scale correctly, but when there are not that many distinct values one good way of handling incorrect sorting is to set the “Value Ordering” property of the data column.

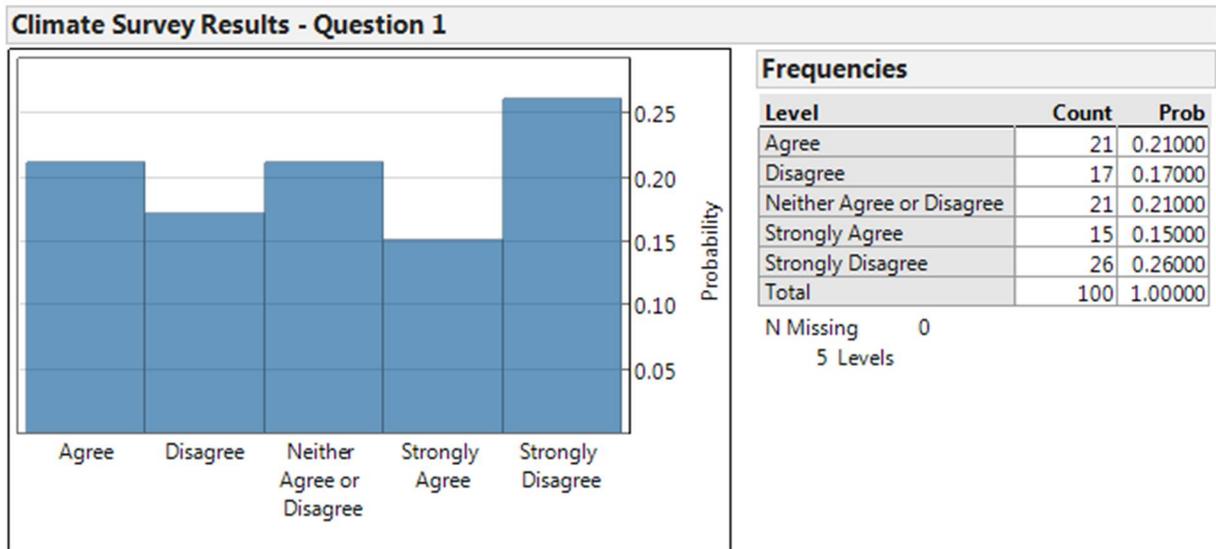


Figure 10 Incorrect sorting of Likert Scale

When we first bring up the Value Ordering property dialog, the values are sorted as shown in Figure 11. We can then select each value and organize the list as we deem appropriate by selecting an item and clicking the Move Up or Move Down buttons. Note that several items in the list can be selected at once by holding down shift as we click on each item.

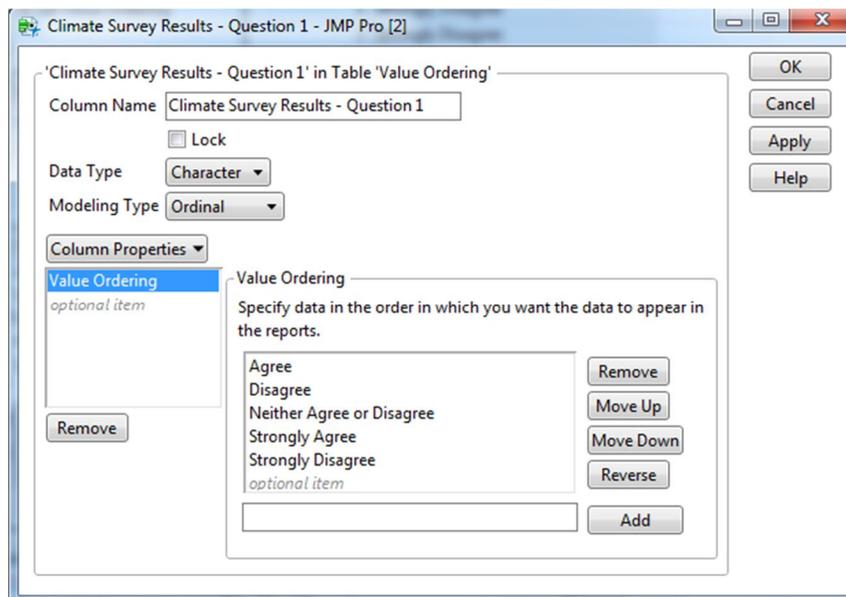


Figure 11 Initial dialog shown for Value Ordering

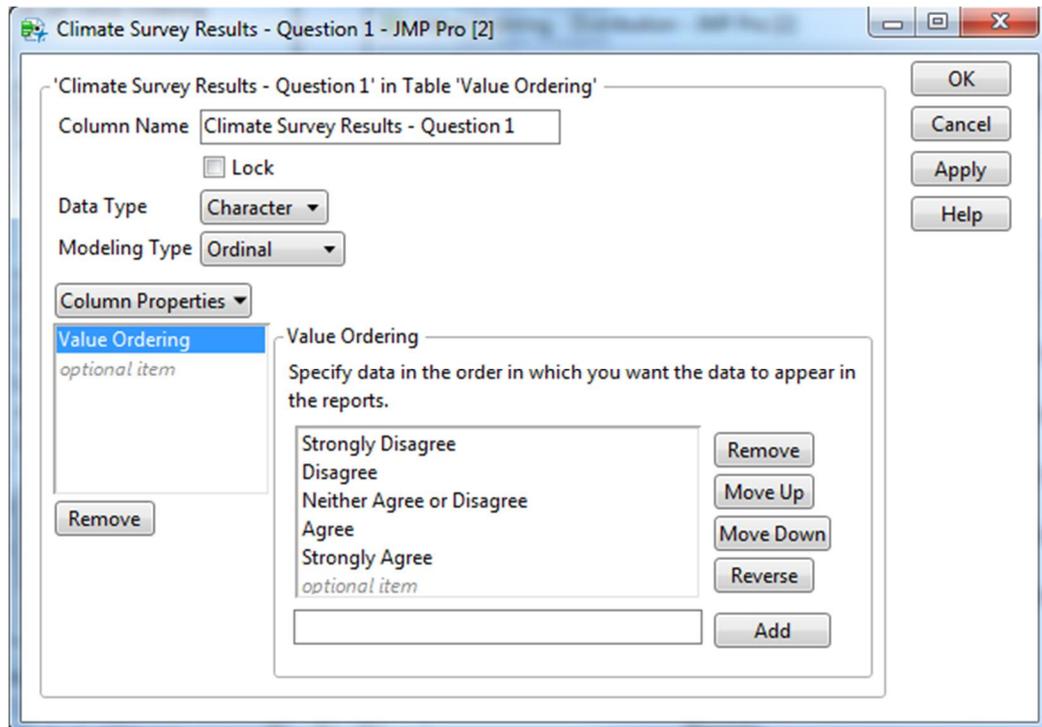


Figure 12 Value ordering dialog after values have been sorted.

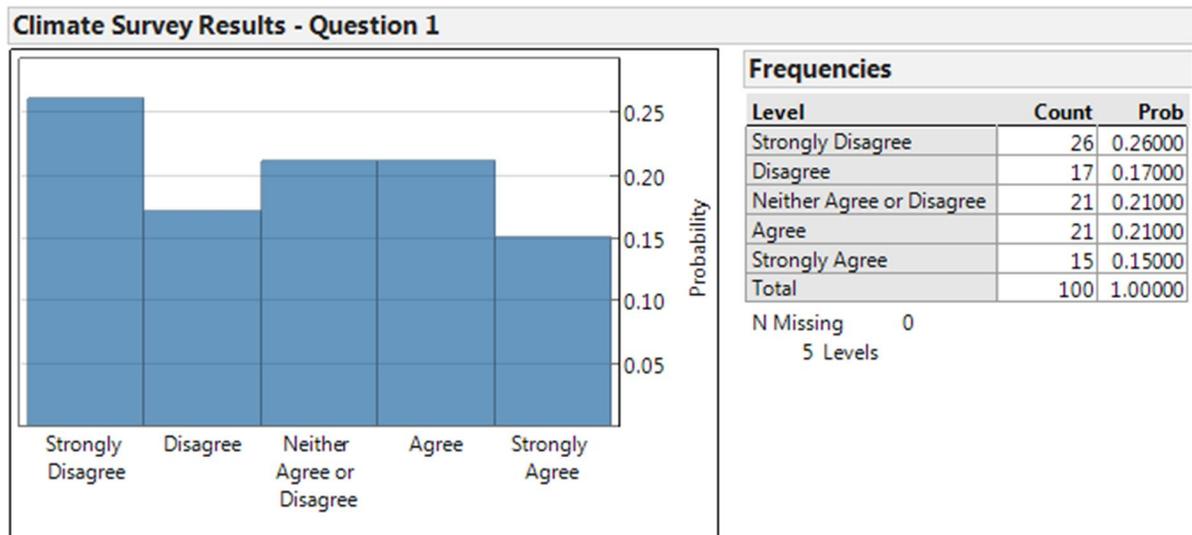


Figure 13 Histogram showing correct sort order of Likert results

In addition to Value Ordering sometimes we want to connect points in a scatterplot in the order they were given in the data table. We might want to do this to show shapes, tool paths, etc. Again there is more than one way to do this in JMP, but one simple way is to start by plotting the points in Graph Builder. With the points plotted hold down the SHIFT key and select the line tool. Initially the points are connected in the order of increasing X as shown in Figure 12. By selecting the "Row Order" checkbox under the "Line" group box, the data points will then be connected as they are sorted in the data table. (See Figure 15).

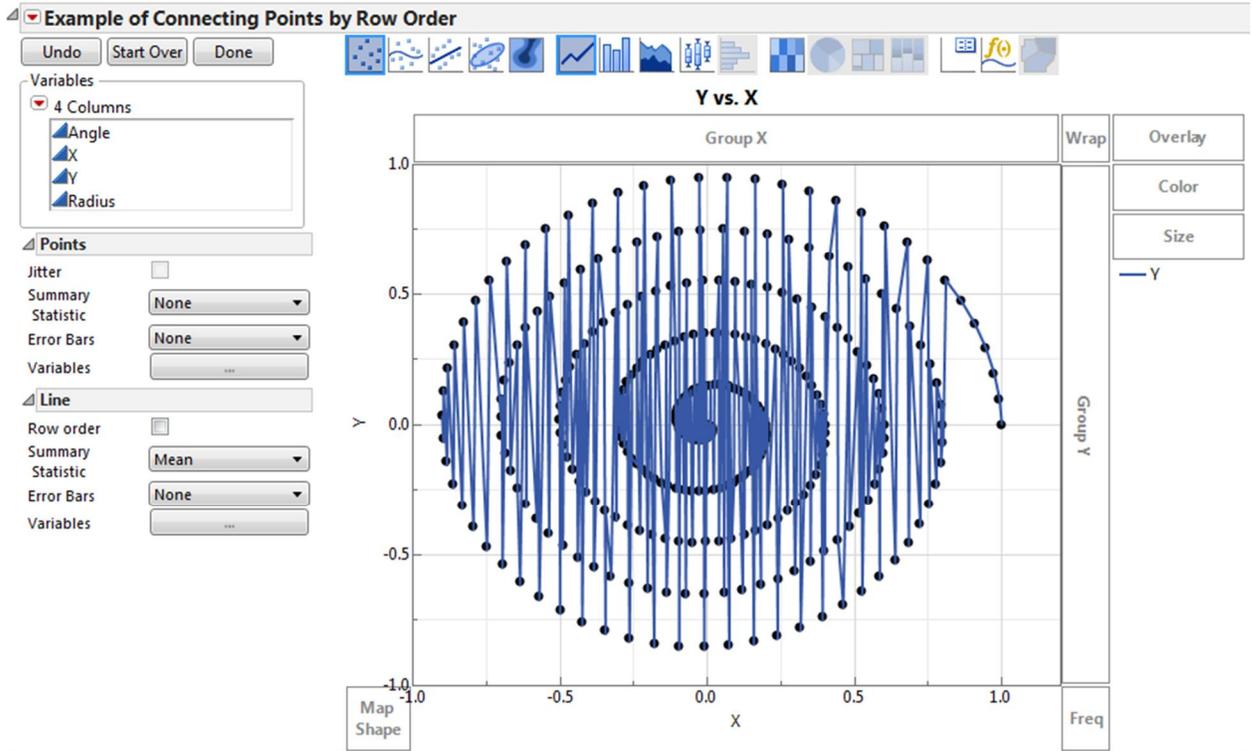


Figure 14 Points connected in increasing X order.

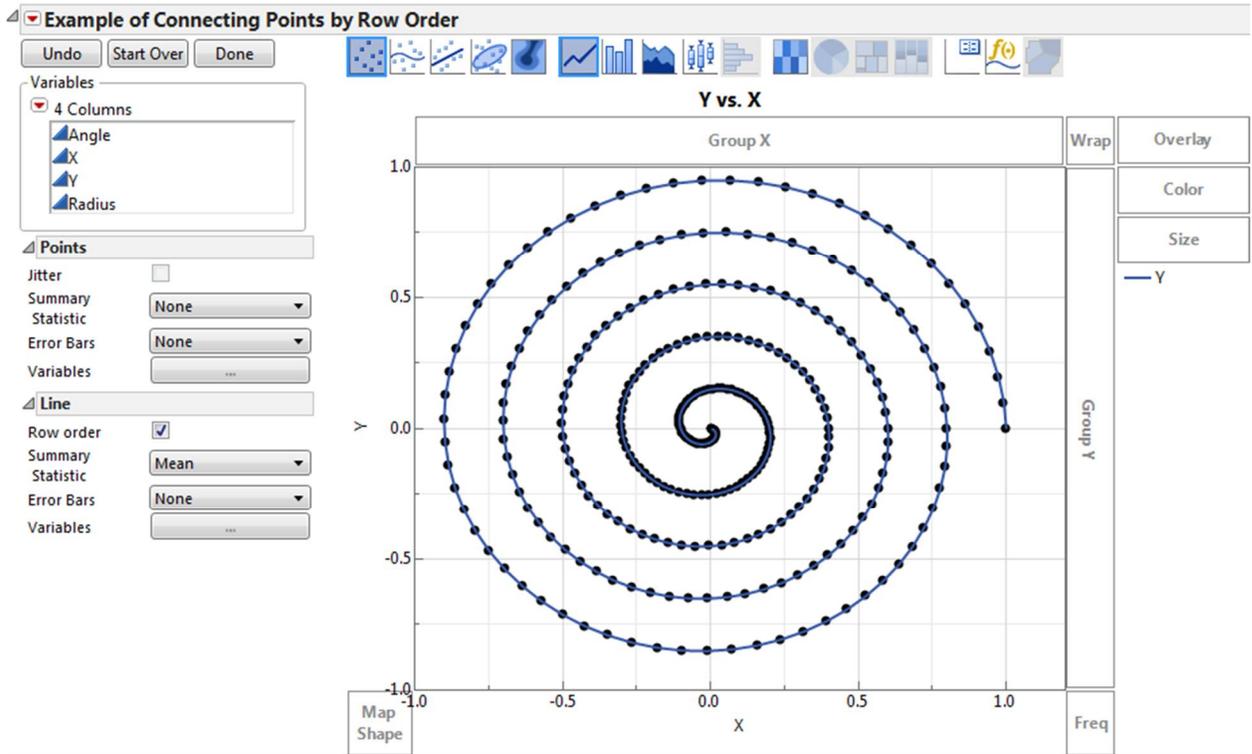


Figure 15 Points connected in Row Order

I'll conclude this section with an example of setting the "Spec Limits" property of a column. If a column in the data table represents an attribute that has associated specification limits, setting the Spec Limits property is a great time-saver. After this property is set, all graphs of that attribute can display spec limits and even some basic analyses, such as counting out of spec values, are automated. For this example I'll use the JMP example data set Diameter.jmp. According to the notes in the example data table, the data originate from a company that manufactures tubing for medical applications. The target outer diameter (OD) of the tubing was 4.4mm, and two prototype runs occurred between May 1 and May 20 (Phase 1), and May 21 and June 9 (Phase 2). While not specifically stated, let's assume the tubing has specifications for Max OD and Min OD of 4.8mm and 4.0mm respectively. The raw data, plotted over time, are shown in Figure 16. Clearly some diameters are larger than 4.8mm and many are smaller than 4.0mm. We can quickly determine how many by setting the Spec Limits property for the DIAMETER column, and plotting the data with the Distribution platform by Phase.

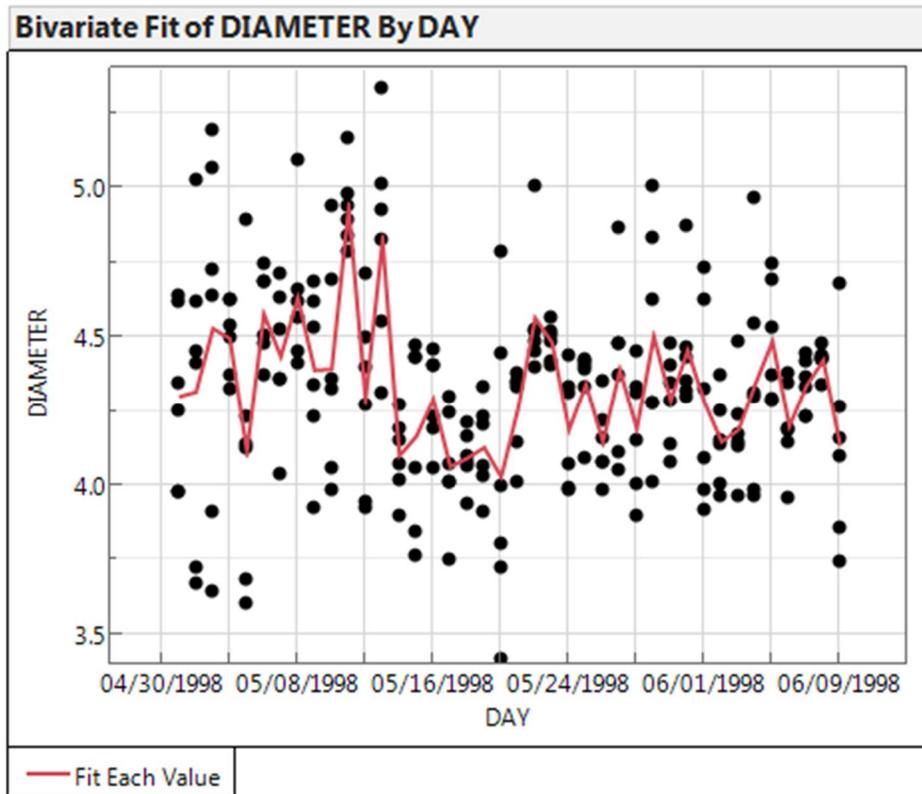


Figure 16 Diameter data plotted over time.

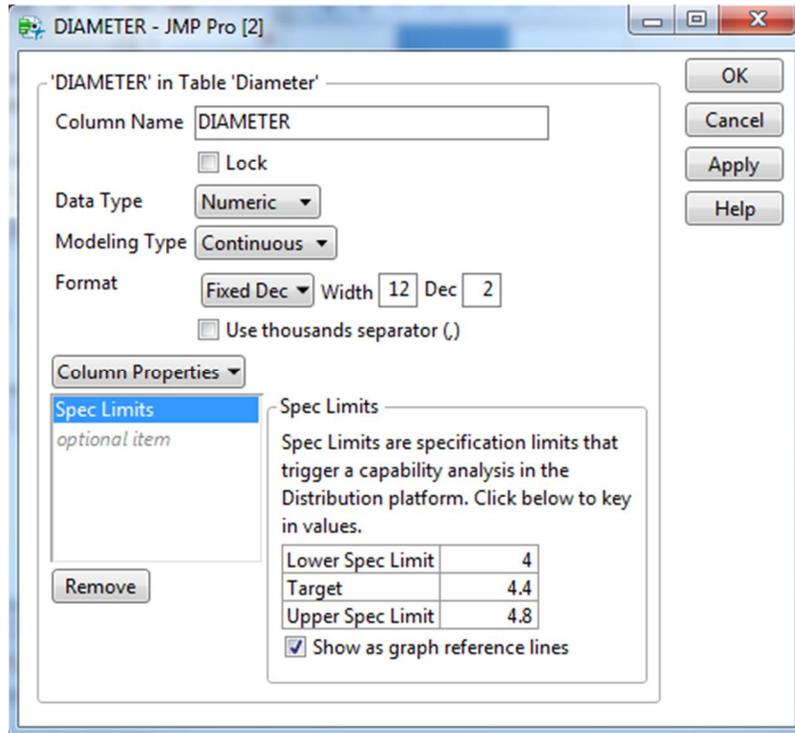


Figure 17 Spec Limits Dialog

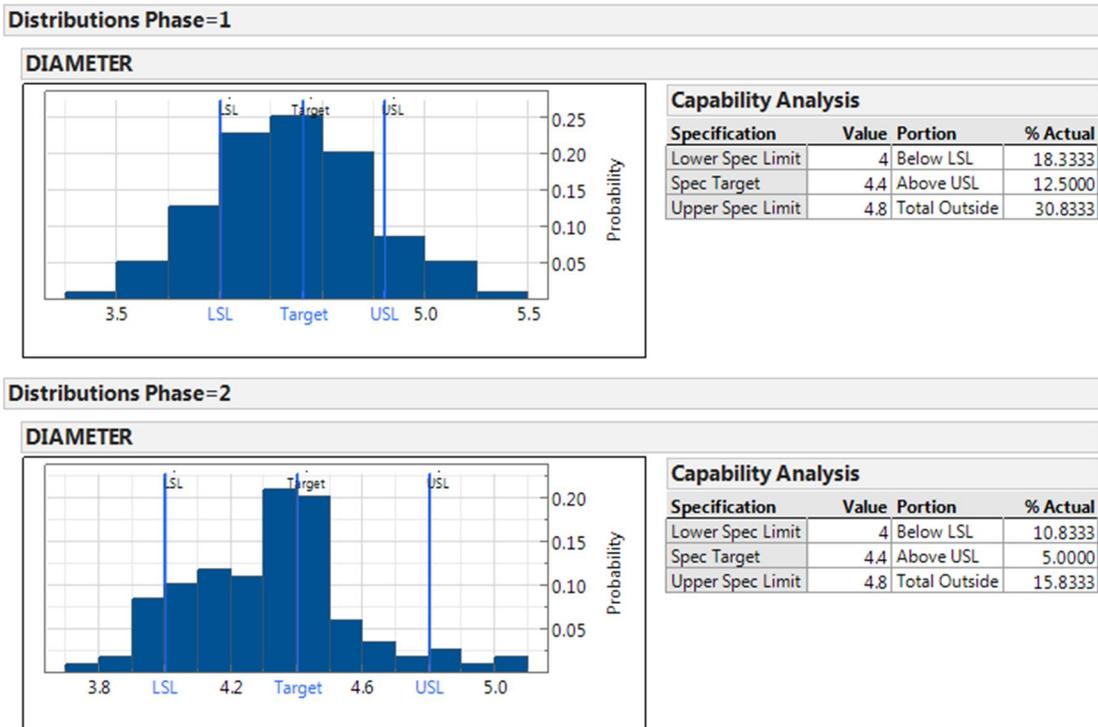


Figure 18 Histograms with Spec Limits and Performance

Now not only are the specification limits included on the plots, but we can see that for Phase I 31% of the diameters were out-of-spec and for Phase II 16% of the diameters were out-of-spec.

Furthermore if we were to fit distributions to these data, which is easily done in the Distributions platform, the fitted distribution would also be automatically compared to the specifications.

2. Graph Customizations (With Simple Scripting)

While point-and-click graph customizations are very powerful, with a little JMP scripting we can take some of our graphs to the next level. Some things done with JMP Scripting Language (JSL) can only be done with JSL. In order to begin doing some simple JSL graph customizations it's easiest to start with an existing graph. The first example will come from the JMP example dataset called "Pendulum". The Pendulum dataset contains the results of an experiment in a physics class comparing the length of a pendulum to its period (the time it takes the pendulum to make one complete swing). The data are shown in Figure 19.

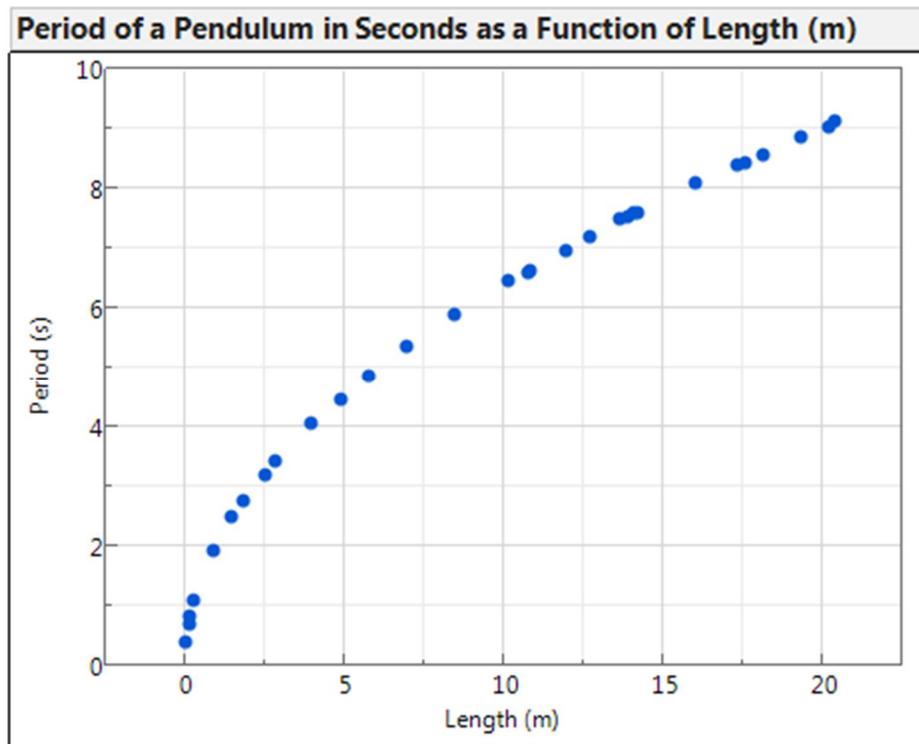


Figure 19 Pendulum Data

Clearly there is a relationship between the period of a pendulum and the length of the pendulum. If you tried to fit a polynomial to these data however, none fit well. As it turns out, there is a known equation that relates the period of a pendulum to its length as shown below:

$$Period = 2\pi \sqrt{\frac{Length}{Gravity}}$$

In order to show this relationship on the graph, first right-click the graph and select the “Customize” menu item.

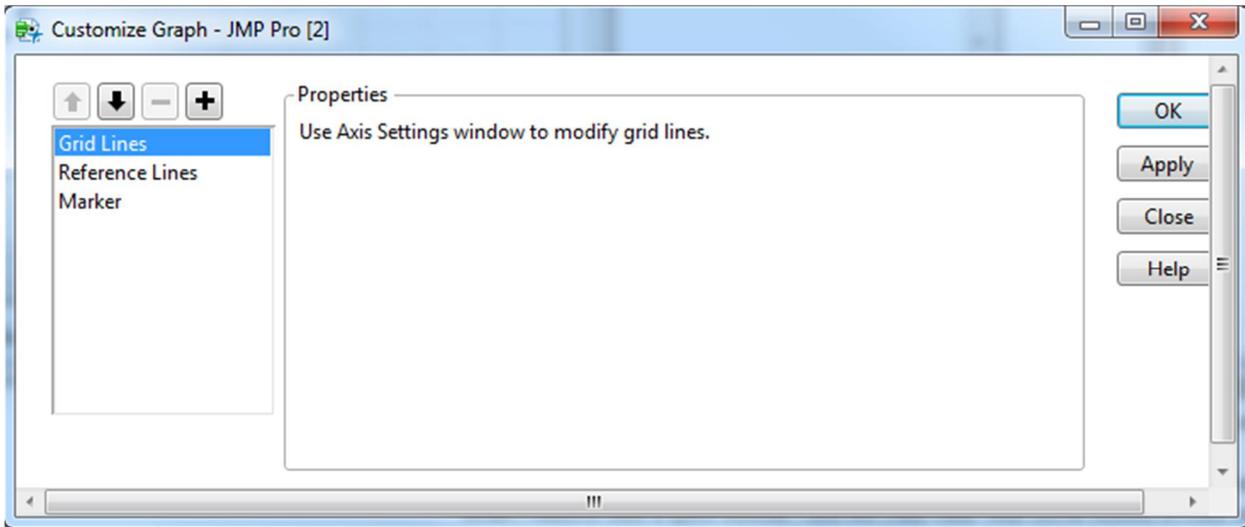


Figure 20 The Customize Dialog

We'll need to add a Script to the graph by clicking the plus sign in the upper left corner. When we do that a script is added to the list on the left, which I've named "Pendulum Equation" and a script window is opened in the Properties box. By clicking the Templates button we see there are several common JSL snippets that can be automatically inserted into the script window for us. I'll select Pen Color, Pen Size, and Y Function for this example.

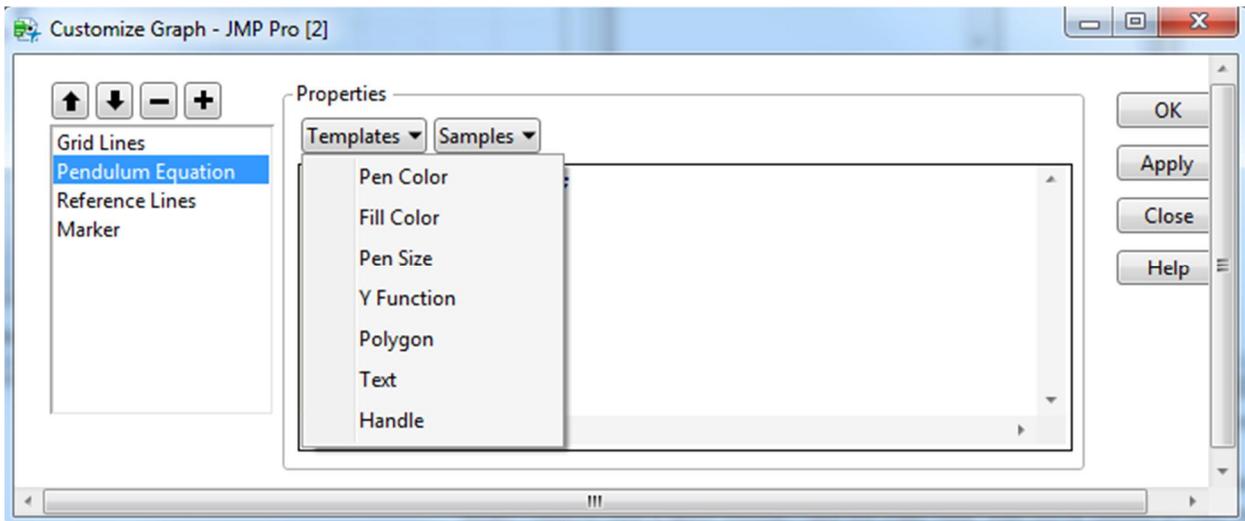


Figure 21 Options shown under Templates in a Script window of the Customize dialog.

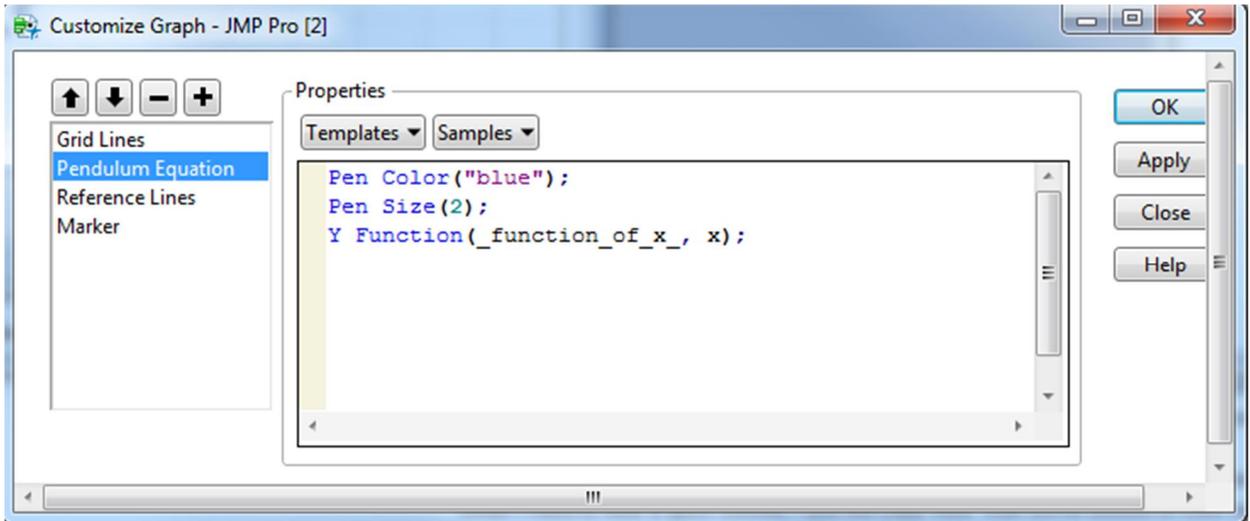


Figure 22 Template items added to script window

Note that each JMP command ends in a semicolon. I'll change the pen color to "green", and change the pen size from 2 to 3. In order to plot the function relating period to pendulum length, we replace the "_function_of_x_" text in the Y Function command with " $2 * \text{Pi}() * \text{Root}(x / 9.8)$ ". After clicking OK, the revised graph is shown below in Figure 23.

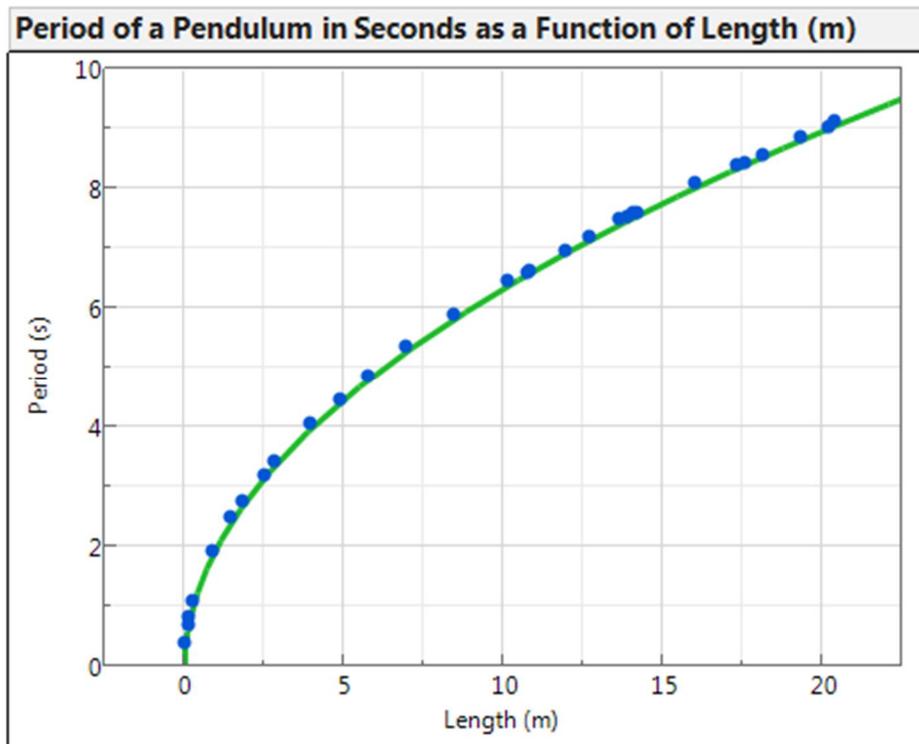


Figure 23 Pendulum Data with first-principles fit

The plotted function fits the data much better than a polynomial could. Note that while it was not shown under the Templates dropdown, there is also an X Function. In fact any JSL command can be used in the custom graph script. Staying with the Y Function for now, another powerful

graph customization is to add custom distribution density curves such as historical distributions to graphs. For this example I'll return to the yield data shown in Figure 1. Clearly a single distribution fit will not work with these data, because there are in fact two distributions. In the distribution platform we can make a histogram of the yield data and fit a Normal 2 Mixture distribution. This distribution is actually a combination of two Normal distributions scaled such that the total area under the combined distribution curve is 1. Figure 24 shows that JMP determined that 27.9% of the data came from a Normal distribution with a mean of 95.96 and a standard deviation of 0.8919, and (100%-27.9% =) 72.1% of the data came from a Normal distribution with an average Yield of 98.0 with a standard deviation of 0.9080.

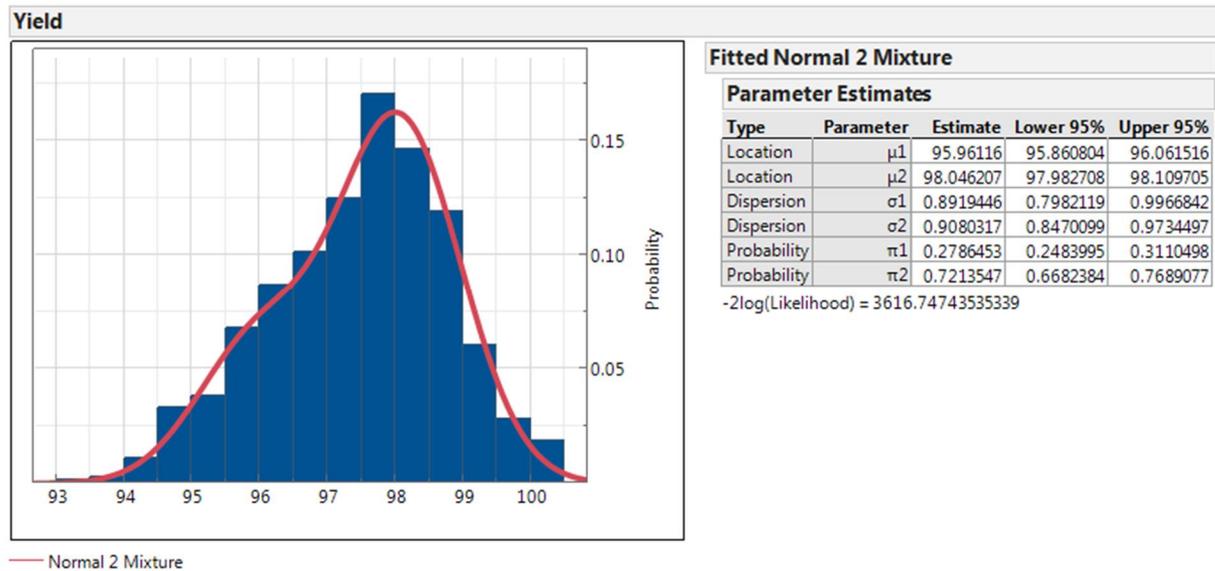


Figure 24 Yield Data with Normal 2 Mixture Fit

To see what these individual distributions look like, and how they add to the Normal 2 Mixture distribution shown, we can again customize the graph and add a script. I'll use two Y Functions in this script and give them each a different color. The resulting script is shown below and the graph is shown in Figure 26.

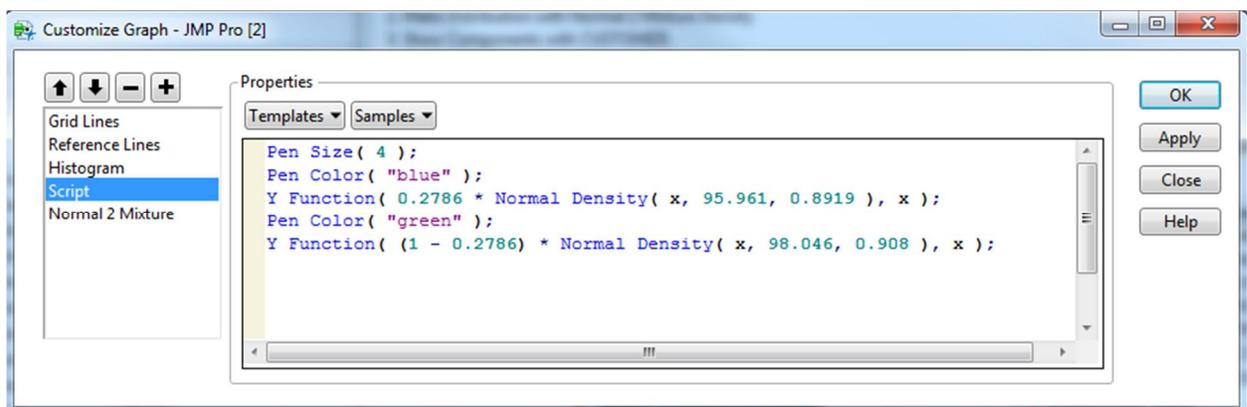


Figure 25 Script to show scaled individual Normal curves

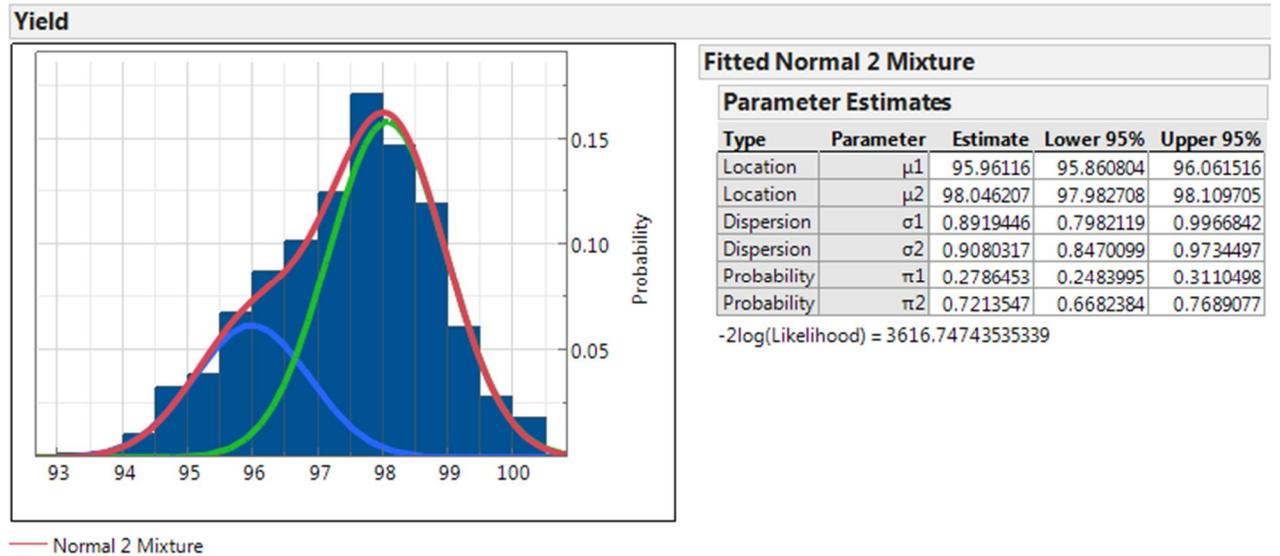


Figure 26 Histogram of Yield data with Overall Normal 2 Mixture Fit and its components

Another useful graph customization is the use of polygons. Returning to the JMP example dataset DIAMETER, recall that we said the lower specification limit (LSL) for diameter was 4.0mm and the upper specification limit (USL) for diameter was 4.8mm. Let's further assume that for any particular tube measured there is also an out-of-round (OOR) specification of 0.2mm. OOR is typically defined as the max diameter minus the min diameter. For a perfect circle the OOR = 0, and for ovals the OOR > 0.

If we start with a plot of max diameter versus min diameter, we can again right-click the graph and select the Customize menu option. After adding a script, we'll choose the Polygon function from the Templates dropdown. The Polygon function has this form:

Polygon([_x0_, _x1_, ...], [_y0_, _y1_, ...])

As such we need to supply a list of x-coordinates and a list of y-coordinates. The polygons I want to add to the graph of max diameter versus min diameter will highlight areas where one or more specifications are not met, and the area where all specifications are met. When related dimensions such as diameter and OOR have their own specifications, it's important to understand how these specifications relate to each other as well. Figure 27 shows the completed graph customization script, and the customized graph is shown in Figure 28. Note that I have also manually added text boxes, an oval, and some arrows to the graph.

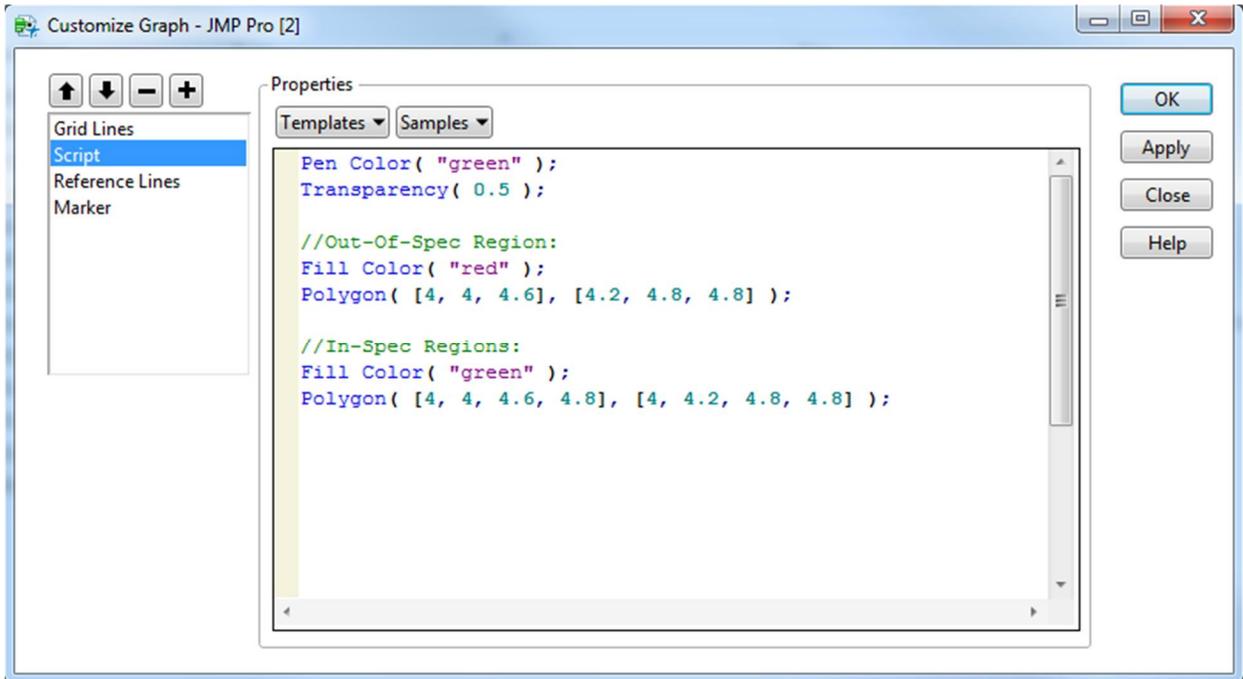


Figure 27 Graph customization script to add polygons

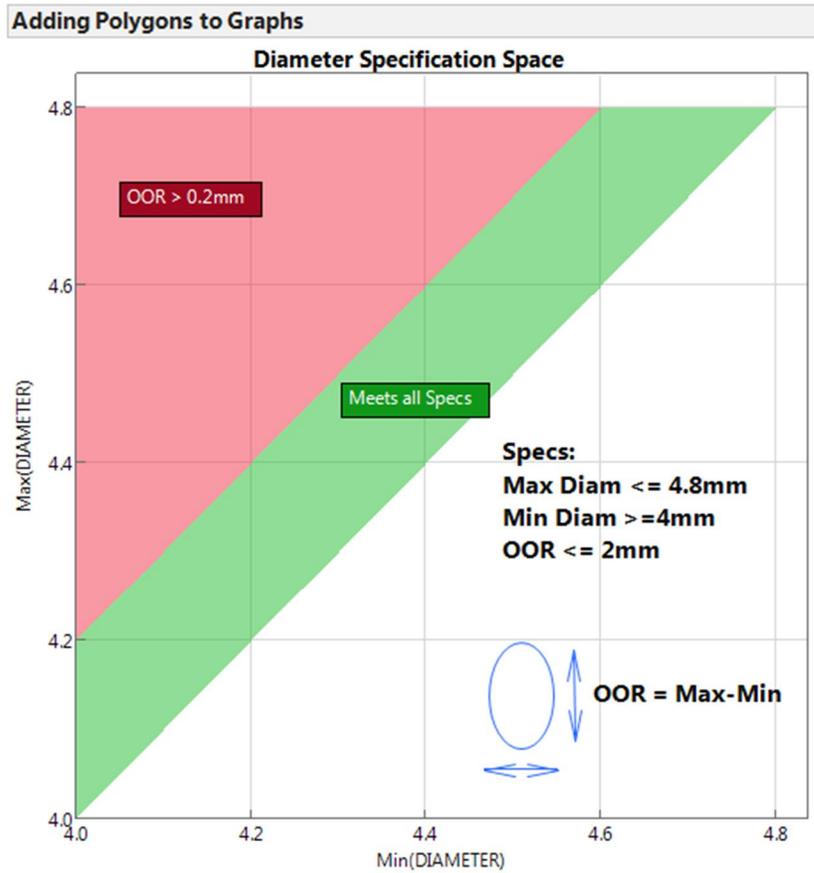
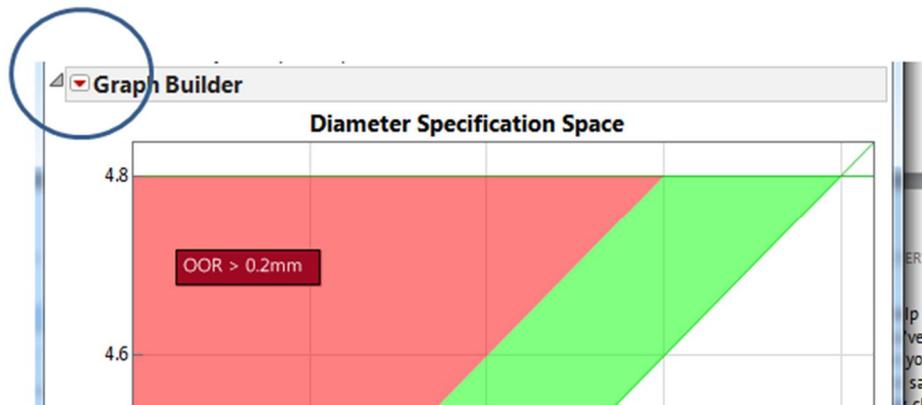


Figure 28 Customized Graph with specification regions shown.

IMPORTANT FINAL NOTE!!

After spending time customizing a graph to help clarify your results, it is vital that you save those customizations for future use. Whether you've created the perfect point-and-click graph or modified a graph with JSL, it is important to save your work. As with many things in JMP there is more than one way to do this. My preference is to save graphs made from a particular data table with that same data table. This can be accomplished by clicking the red arrow next to the top outline box in the graph window.



This produces a menu and at or near the bottom will be the Script menu item. When you hover over the Script menu item another menu appears which allows you to save the script in various ways. If you select "Save Script to Data Table" then a new script will be added to the data table that you can run at a later time. All your graph customizations will be saved in that script, and will reside with the data table when you save it. You can run them again at any time.

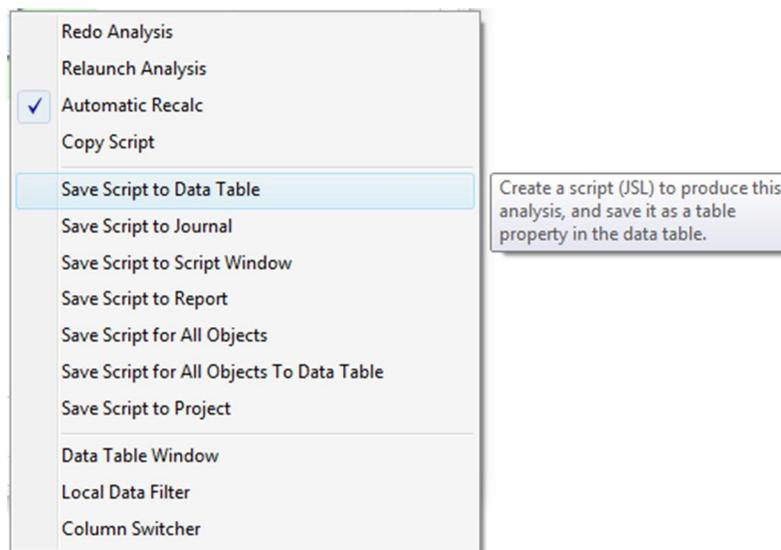


Figure 29 Menu to save your script to the data table

3. Making Interactive Graphs with JSL

A comprehensive account of making custom interactive JSL graphs is not a small undertaking. My own JSL skills are still under development, so it would be hard to do the topic justice. I have however created several custom interactive graphs as teaching and engineering tools. One such application is shown below:

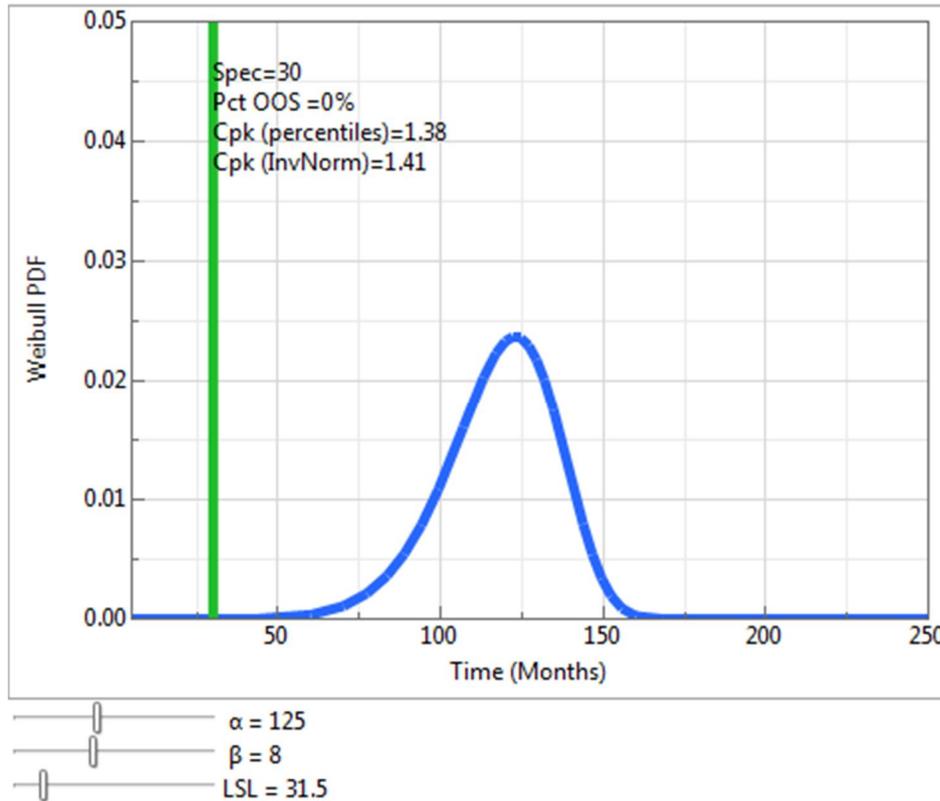


Figure 30 Tunable Weibull Distribution JSL App

This application allows the user to move the sliders that change the scale and shape parameters of a two-parameter Weibull distribution as well as the lower specification limit. In addition to seeing how the distribution changes form with different parameters, I calculate the percent of the distribution falling below the lower specification limit as well as two methods of estimating the Cpk process capability metric. I've included the JSL code in the appendix.

4. Conclusion

It is my sincere hope that the reader was able to glean something useful from this presentation. I have benefitted greatly by learning JMP from others and by my own research. If any of these topics were new and interesting to you, I highly recommend that you try the techniques you found to be interesting on your own data.

5. Where to Learn More

In addition to learning from a few of my JMP-savvy colleagues, I have personally benefitted from the following resources.

- All of the pdf JMP manuals to be found under the help menu in JMP in the “Books” section.
- Example JMP data sets
- Murphrey, Wendy, and Rosemary Lucas. *Jump into JMP® Scripting*. Cary, NC: SAS Institute Inc.
- Utlaut, Theresa L., Georgia Z. Morgan, and Kevin C. Anderson. 2011. *JSL Companion: Applications of the JMP® Scripting Language*. Cary, NC: SAS Institute Inc.
- [Jmp.com/haveyoutried](http://jmp.com/haveyoutried)
- The JMP Blog: <http://blogs.sas.com/content/jmp/>
- JMP live and on-demand webcasts

6. Appendix – Code for Tunable Weibull Distribution

```

/*-----
Program Name: exploringweibull.jsl
Version: 1.0
Date: 02JUL2013
By: Casey A. Volino, Corning Incorporated
Contact: volinoca@corning.com
JMP Version: JMP 11 Pro
OS Platform: Windows 7
-----

```

Problem to Solve:

None. Nice visual aid to learn about Weibull parameters

Description:

This script creates a chart displaying a Weibull density curve, and allows the user to dynamically change spec, shape, and scale.

Version History:

1.0 - initial build

Future Improvement Ideas:

None

```

-----*/
Names Default To Here( 1 );

```

```

//-----
// Initialize Variables
//-----

```

```

dblalpha = 125;
dblbeta = 8;
Wbl_LSL = 36;
Ymax = 0.05;
YStep = YMax/10;
XMax = 1;

```

```

//-----
// Main Program Starts Here
//-----

```

```

New Window( "Tunable Weibull Distribution",
  g1 = Graph Box(
    Y Scale( 0, .05 ),
    YName( "Weibull PDF" ),
    X Scale( 5, 250 ),
    XName( "Time (Months)" ),
    Pen Color( "blue" );
    Pen Size( 5 );
    Y Function( Weibull Density( x, dblbeta, dblalpha ), x );
    Pen Color( "green" );
    Line( {Wbl_LSL, 0.00}, {Wbl_LSL, .99} );
    Pen Color( "red" );
    Line( {0, 0.1}, {5*Wbl_LSL, 0.1} );
    showOutput;
  ),

```

```

  V List Box(
    H List Box(

```

```

Slider Box( 1, 300, dblalpha,
            dblalpha = round(dblalpha,0);
            t1 << Set Text(" \!U03B1 = " || char(dblalpha));
            Wait(0);
            g1 << reshow),
H List Box(
Slider Box( 0.1, 20, dblbeta,
            dblbeta = round(dblbeta, 1);
            t2 << set Text(" \!U03B2 = " || char(dblbeta));
            Wait(0);
            g1 << reshow ),
t2 = Text Box( " \!U03B2 = " || char(dblbeta) ) ),
H List Box( Slider Box( 0.1, 200, Wbl_LSL, Wbl_LSL =
round(Wbl_LSL,1); t3 << Set Text("LSL = " || char(Wbl_LSL)); Wait(0); g1 <<
reshow ), t3 = Text Box( "LSL = " || char(Wbl_LSL) ) )
),
);
//-----
//      Macros and Functions
//-----

///// Macro to show the numeric results:
showOutput = Expr(
//Note that "\!UU03B1" is the Greek alpha symbol
//Note that "\!UU03B2" is the Greek beta symbol

//0.0013rd pctile;
WblLow = Weibull Quantile( 0.0013, dblbeta, dblalpha );

//median;
Wblmedian = Weibull Quantile( 0.5, dblbeta, dblalpha );
WblCpk = (Wblmedian - Wbl_LSL) / (Wblmedian - WblLow);
myProb = Weibull Distribution(Wbl_LSL, dblbeta,dblalpha );
q = -Normal Quantile(myProb);
normCpk = q/3;
//Text( {Wbl_LSL, 0.0225}, "\!U03B1=", Round( dblalpha, 2 ) );
//Text( {Wbl_LSL, 0.02}, " \!U03B2=", Round( dblbeta, 2 ) );
YMax = g1[axisbox(1)]<< get max;
Text({Wbl_LSL, 0.9*YMax}, "Spec=", Round(Wbl_LSL,1));
Text({Wbl_LSL, 0.85*YMax}, "Pct OOS =", Round(myProb*100,2), "%");
Text( {Wbl_LSL, 0.8*YMax}, "Cpk (percentiles)=", Round( wblcpk, 2 ) );
Text( {Wbl_LSL, 0.75*YMax}, "Cpk (InvNorm)=", Round( normCpk, 2 ) );
);

```