

# Saving Time and Analysis with JMP Scripting

Byron Wingerd, JMP

June 30, 2017

# Tracking and Trending Manufacturing Metrics

## Part 2

Using JSL to automate repetitive monitoring

Tips and tricks for working with control chart scripts

# Building the Weekly Metrics Report

## Parts List

1. Historical Process Data
2. Current Data
3. Control Limits Table
4. Specification Limits Table
5. Some JMP Reports

**PLUS: About an hour to cobble it all together\***

\*Includes necessary stop for fresh coffee

# This is a Technically Advanced Example

Every effort was made to keep the example simple

- Data manipulation example
- Example of loading control limits to a table
- Example of loading spec limits to a table
- Example of editing an axis box settings
- Example of saving a report as an image

Most people won't use all these parts together, but these are all good things to know how to do.

# JSL Essentials

- Help>Books>Scripting Guide
- Help>Scripting Index
- This one link: <https://community.jmp.com/t5/JMPer-Cable/Do-you-want-to-learn-the-JMP-Scripting-Language/ba-p/37438>
- Script and log windows
- Syntax
  - Only spelling matters, and punctuation
  - Its function based, need to glue between with a semicolon

# Workflow

- Data Preparation
  - Open Historic data and Current data, Concatenate the Current data
  - Add Control Limits
  - Add Specification Limits
- Generate reports
- Make Reporting Media
  - Update PowerPoint slides
  - Save reports as images

# Opening the Data Files

```
dt_old      =open("Historic Metrics Data.jmp");  
dt_new      =open("Current Metrics Data.jmp");  
dt_cl       =open("Control Limits.jmp");  
dt_specs    =open("Specification Limits.jmp");
```

- Note variable name convention: Intentional vs. random or creative
  - Variable names have very few constraints
- File names are missing the path in this example
  - Data files are in the same folder as the script.
  - Q:\User\Data\Project\Subfolder\longlongname\obscurenumber\data\JMPPFILE.jmp would work too
- `dt_update = dt_old<<concatenate(dt_new, Append to first table, create source column);`

# Add New Data to the Historic Data

- Manual: Tables tab, Concatenate
  - Do this once and edit the Source script that is written to the resulting table
- Check the Scripting Index for arguments
- `dt_update = dt_old<<concatenate(dt_new, Append to first table, create source column);`



# Load Control and Specification Limits

- DO NOT!

- Go shopping for obscure ways to do this
- Try to scrape and append column properties
- Spend hours and hours writing complex custom code

**HARD WAY**

- DO

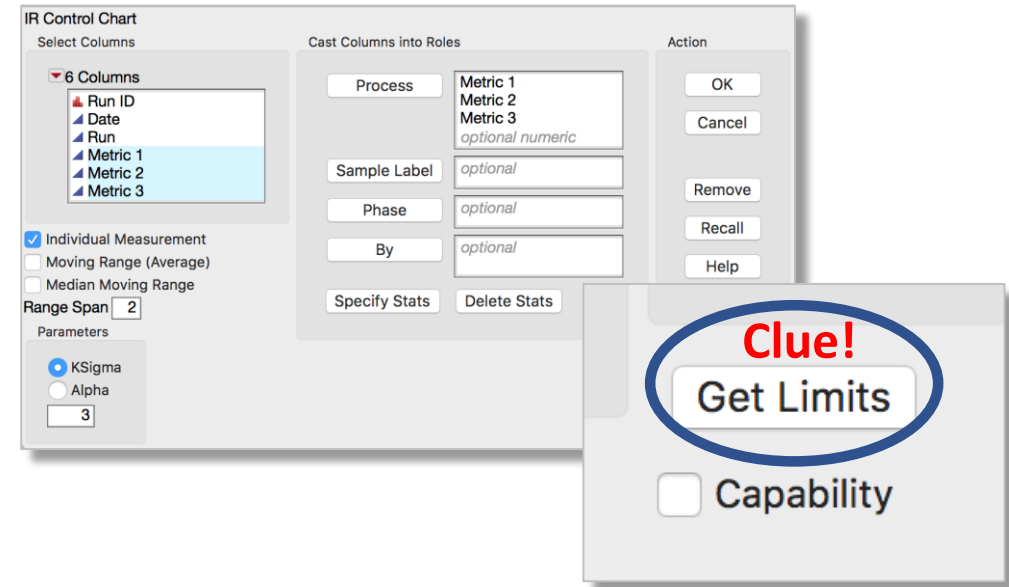
- Script the platforms.
- Check the scripting index
- Look for clues in the platforms

**Easy WAY**

# Loading Control Limits

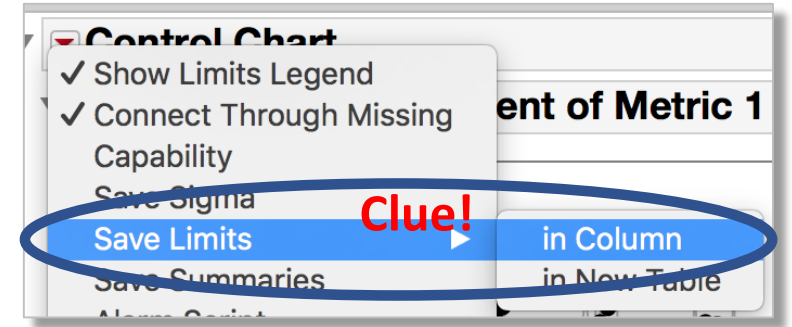
Search Scripting Index for “Get Limits”

Control Chart(  
Chart Col( :Metric Col, Individual Measurement),  
**Get Limits(“Path/Control\_Limits\_file.jmp”)**  
);



# Save Control Limits to Column

Search Scripting Index for “Save Limits”



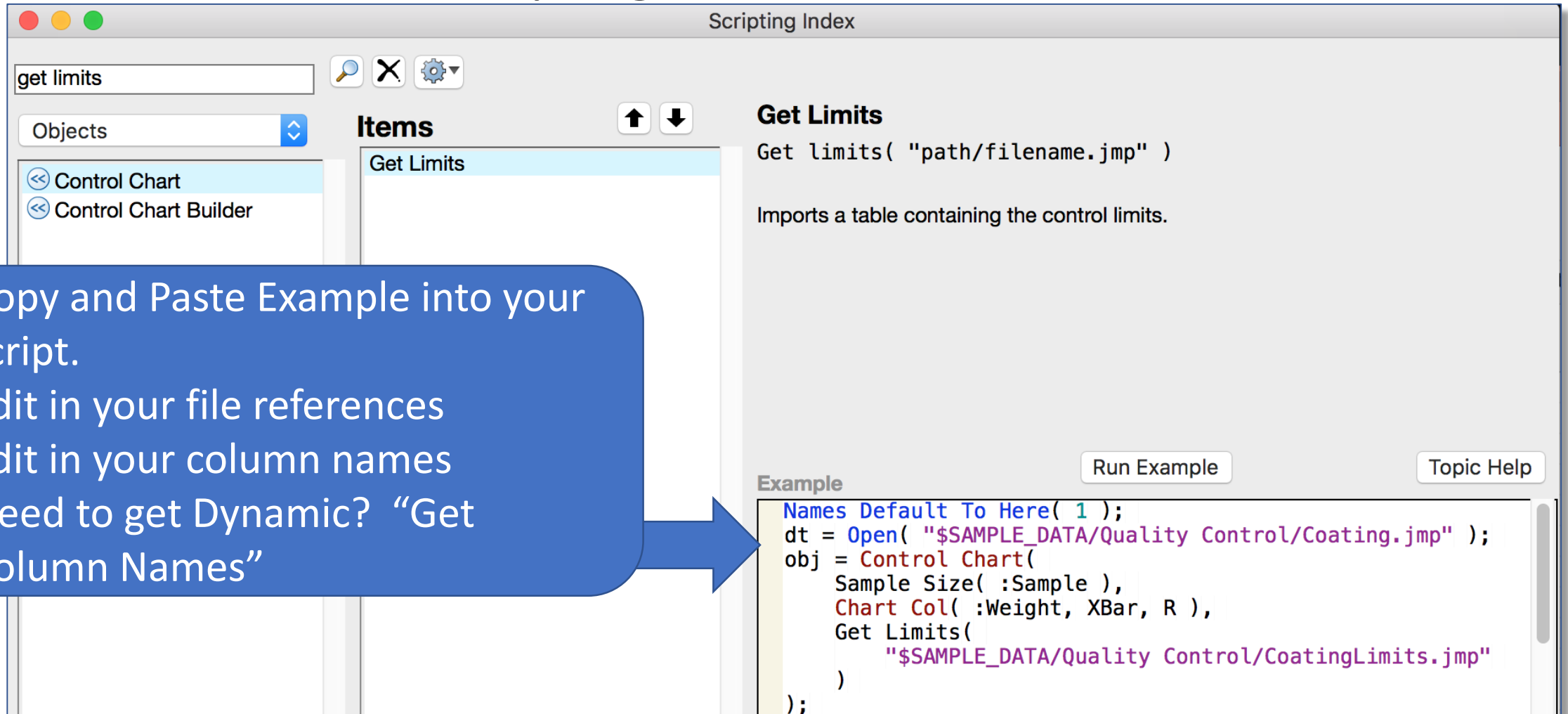
```
Obj=Control Chart(  
Chart Col( :Metric Col, Individual Measurement),  
Get Limits("Path/Control_Limits_file.jmp")  
);
```

```
obj << in Column;
```

**Don't write complex scripts.  
Script the platform!**

# Load Control Limits

## Search Scripting Index for “Get Limits”



The screenshot shows the JMP Scripting Index window. The search bar contains "get limits". The "Objects" list on the left includes "Control Chart" and "Control Chart Builder". The "Items" list on the right includes "Get Limits". The main panel displays the "Get Limits" function signature: `Get limits( "path/filename.jmp" )`, followed by the description: "Imports a table containing the control limits." Below this, there is an "Example" section with a "Run Example" button and a "Topic Help" button. The example script is as follows:

```
Names Default To Here( 1 );
dt = Open( "$SAMPLE_DATA/Quality Control/Coating.jmp" );
obj = Control Chart(
    Sample Size( :Sample ),
    Chart Col( :Weight, XBar, R ),
    Get Limits(
        "$SAMPLE_DATA/Quality Control/CoatingLimits.jmp"
    )
);
```

A blue callout box on the left contains the following instructions:

- Copy and Paste Example into your script.
- Edit in your file references
- Edit in your column names
- Need to get Dynamic? “Get Column Names”

An arrow points from the callout box to the example script code.

# Load Control Limits

```
obj=dt_update<<Control Chart(  
    Chart Col( :Metric 1, Individual Measurement ),  
    Chart Col( :Metric 2, Individual Measurement ),  
    Chart Col( :Metric 3, Individual Measurement ),  
    get limits("Control Limits.jmp"));
```

```
obj<<in Column;
```

```
obj<<close window;
```

- Note: “**obj**” is a handle for the report.
  - Send “in Column” to it
  - Send “close” to it ...because I don’t want to use this report, just the platform.

# Loading Specification Limits

- DO NOT!
  - Go shopping for obscure ways to do this
  - Try to scrape and append column properties
  - Spend hours and hours writing complex custom code
- DO
  - Script the platforms.
  - Check the scripting index
  - Look for clues in the platforms

**HARD WAY**

**Easy WAY**

# Load Specification Limits

Search Scripting Index for “Spec Limits”

The screenshot shows the JMP Scripting Index window. The search bar at the top contains the text "spec limits". The left sidebar lists various objects, with "Capability" expanded to show "Spec Limits" selected. The main pane displays the "Import Spec Limits" topic, which is circled in blue. The topic includes a code snippet and a description. Below the main pane, there is an "Example" section with a code snippet and two buttons: "Run Example" and "Topic Help".

Scripting Index

spec limits

Objects

- Capability
  - Spec Limits
- Data Table
  - Column Scripting
- Degradation
- Distribution
  - Continuous Distribution
  - Fit Distribution
- Process Capability
  - Process Capability Analysis
  - Process Capability Analysis
- Process Screening
- Profiler
  - Simulator

Items

Import Spec Limits

**Clue!**  
**Import Spec Limits**  
`obj << Import Spec Limits( "path/filename.jmp" )`  
Imports a table containing the specification limits.

Run Example

Topic Help

Example

```
Names Default To Here( 1 );
dt = Open( "$SAMPLE_DATA/Cities.jmp" );
obj = Capability(
  Y( :OZONE, :CO, :SO2, :NO ),
  Spec Limits(
    Import Spec Limits( "$SAMPLE_DATA/CitySpecLimits.jmp" )
  )
);
```

# Load Specification Limits

Save spec limits back to the table

The screenshot shows the JMP Scripting Index window. The search bar at the top left contains the text "spec limits". Below it, the "Objects" list on the left includes "Process Capability", which is expanded to show "Process Capability Analysis" and "Process Capability Analy". The "Items" list in the center contains "Save Spec Limits as Column Properties", which is circled in blue. To the right of the "Items" list, the "Save Spec Limits as Column Properties" option is highlighted with a blue oval and labeled "Clue!". Below this, the description states: "For each column representing a process, saves the specification limits as a column property within the column of the original data table." At the bottom right, there is an "Example" section with a "Run Example" button and a "Topic Help" button. The example code is as follows:

```
Names Default To Here( 1 );
dt = Open( "$SAMPLE_DATA/Cities.jmp" );
obj = Process Capability(
    Process Variables( :OZONE, :CO, :SO2, :NO ),
    Spec Limits( Import Spec Limits( "$SAMPLE_DATA/CitySpecLimits.jm
);
obj << Save Spec Limits as Column Properties;
```



# Load Specification Limits

```
obj = dt_update<<Capability(  
    Y( :Metric 1, :Metric 2, :Metric 3 ),  
    Spec Limits(Import Spec Limits( "Specification Limits.jmp" )));  
obj << Save Spec Limits as Column Properties;  
obj<<close window;
```

- Note: “**obj**” is a handle for the report.
  - Send “in Column” to it
  - Send “close” to it ...because I don’t want to use this report, just the platform.

Also, This step might take a few seconds with a long list of column names

# Progress Check: Look at the data table

The screenshot shows a software interface with a left sidebar and a main data table. The sidebar has a 'Columns (7/1)' section with a list of columns: 'Source Table', 'Run ID', 'Date', 'Run', 'Metric 1 \*', 'Metric 2 \*', and 'Metric 3 \*'. A context menu is open for 'Metric 3 \*', showing options: 'Control Limits', 'Sigma', and 'Spec Limits'. The main data table is titled 'Source Table' and contains rows of data. A blue arrow points from a text box on the right to the 'Source Table' header, and another blue arrow points from a text box at the bottom to the 'Control Limits' option in the context menu.

	Source Table
169	Historic Metrics Data.jmp
170	Historic Metrics Data.jmp
171	Historic Metrics Data.jmp
172	Historic Metrics Data.jmp
173	Historic Metrics Data.jmp
174	Historic Metrics Data.jmp
175	Current Metrics Data.jmp
176	Current Metrics Data.jmp
177	Current Metrics Data.jmp
178	Current Metrics Data.jmp

Old and New data in the same table with a column to identify the source

Control Limits and Spec Limits loaded to the Column Properties

# Generate Reports

- Manually run the platform to get your report/graph
  - Manually format the graph to make it look “right”
- Get the script
- Edit the script
- String scripts together

# Working with Control Chart Scripts

**fig1 = dt\_update <<**

Control Chart Builder(

Size( 533, 449 ), Show Control Panel( 0 ),

Variables( Subgroup( :Date ), Y( :Metric 1 ) ),

Chart( Position( 1 ) ), Chart( Position( 2 ) ),

SendToReport(

Dispatch({}, "Date", ScaleBox, {Min( Col Maximum( :Date ) - In Days( 30 ) ), Max( Col Maximum( :Date ) ), Interval( "Week" ), Inc( 1 ), Minor Ticks( 6 ), Label Row( Label Orientation( "Angled" ) )}))

);

**Edit 1: Be specific, send the control chart to the right data table**

# Working with Control Chart Scripts

```
fig1 = dt_update <<
```

```
Control Chart Builder(
```

```
  Size( 533, 449 ), Show Control Panel( 0 ),
```

```
  Variables( Subgroup( :Date ), Y( :Metric 1 ) ),
```

```
  Chart( Position( 1 ) ), Chart( Position( 2 ) ),
```

```
  SendToReport(
```

```
    Dispatch({}, "Date", ScaleBox, {Min( Col Maximum( :Date ) - In Days( 30 ) ), Max( Col  
    Maximum( :Date ) ), Interval( "Week" ), Inc( 1 ), Minor Ticks( 6 ), Label Row( Label  
    Orientation( "Angled" ) )}))
```

```
);
```

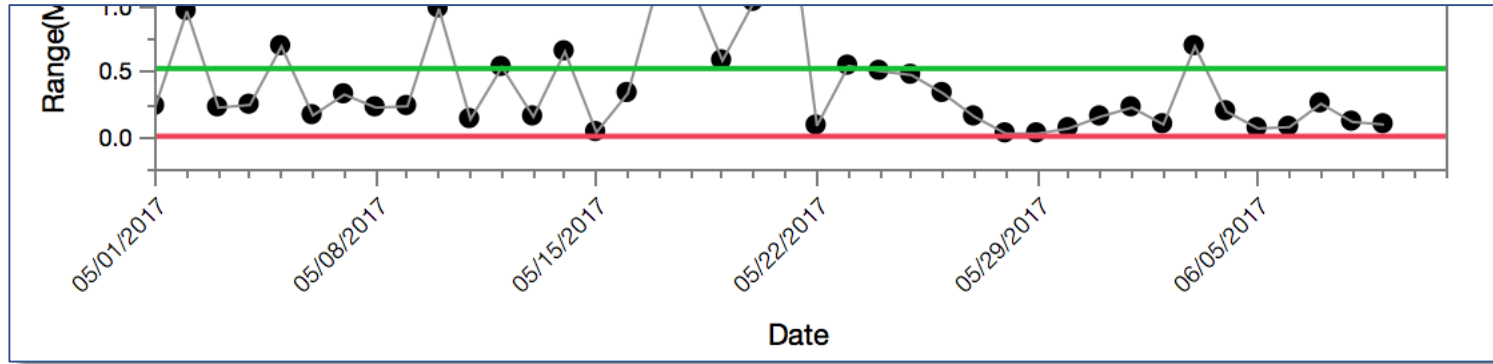
- Edit 2: Unpack and edit the “SendToReport”

# Note: If your are following along, this is missing in the example

```
SendToReport(  
    Dispatch(  
        {},  
        "Metric 1",  
        ScaleBox,  
        {Add Ref Line( 13, "Solid", "Blue", "LSL", 1 ),  
        Add Ref Line( 17, "Solid", "Blue", "USL", 1 )}  
    )  
)
```

**Tip: Just delete this part, it happens automatically anyway**

# Formatting the Axis Box



## Byron's Preference for Axis Boxes in Control Charts (today's version)

1. I like the date angled
2. I like the dates at one week intervals
3. I like 6 tick marks between the weekly intervals
4. I really only want to see the current month or so of data

Do it manually and then copy the script

# Formatting the Axis Box

- Argument sent to the text edit the Axis box

```
Dispatch(  
    {},  
    "Date",  
    ScaleBox,  
    {Min( 3576441600 ), Max( 3579984000 ), Interval( "Week" ), Inc( 1 ),  
     Minor Ticks( 6 ), Label Row( Label Orientation( "Angled" ) )}  
),
```

Edit: I need relative Date References not specific ones

Replace “Min(n)” and “Max(n)” with this:

- Min( Col Maximum( :Date ) - In Days( 30 ) )
- Max( Col Maximum( :Date )

Now the last 30 days are displayed on the chart (days not rows)



# Formatting the Axis Box (alternate approach)

- Argument the edit the Axis box

```
Dispatch(  
  {},  
  "Date",  
  ScaleBox,  
  {Min( 3576441600 ), Max( 3579984000 ), Interval( "Week" ), Inc( 1 ),  
   Minor Ticks( 6 ), Label Row( Label Orientation( "Angled" ) )}  
)
```

In this case, the label column is Numeric, Continuous, Date Format. So the Min and Max are in units of the column, which happen to be seconds.

If there is no label, or a nominal label, this will work to show the last 50

```
{min(nrows()-50, max{nrows()})
```

nrows() refers to the number or rows in the table.

# Making Reporting Media

```
fig1 = dt_update << Control Chart Builder(.....
```

1. Be specific, send the report script to the right data table
2. Get a **handle**
3. Do stuff with the **handle**

- **fig1** << save picture( "figure1.jpg" ,jpg);     Make a jpg image
- **fig1** << save picture( "figure1.emf" ,emf);     Make an enhance metafile
- **fig1** << save picture( "figure1.pdf" ,pdf);     Make a PDF

In this example I need graphics for the next step

# Making Reporting Media

## In PowerPoint

1. Insert Picture from File
2. Insert and Link Option

- Save and Close PowerPoint File
- Run JSL to make figures and save new images.
- Open PowerPoint File



# Making Reporting Media Notes

- The simple script to create the control chart automatically produces the capability report because we already saved the spec limits to the column properties.
- JMP does a lot of complex things automatically, take advantage of them



# Weekly Metrics Update Example

- Make a separate PowerPoint file
- Use the insert, and link function in PowerPoint
- Six Figures
  - Three Control Chart Slides with Capability Analysis
    - One figure per page
  - One slide with some other graphs
    - Two figures on one page

# Notes on Reproducing this Presentation

First, run this script: **Data Script.jsl**  
This script will run: **Reports Script.jsl**

The script generates some figures that will end up in the same directory as the Data Script.jsl file.

- Ideally when you run the PowerPoint deck, you will be able to click on the bold blue triangle to launch a second PowerPoint deck. Depending on your version of MS and administrative control this might be broken.
- Ideally the second PowerPoint file would have figures that automatically update. Unfortunately the links break when I package the file.
- If you want to see the auto update feature work, you have to re-insert and link the images to the appropriate pages.