Creating Custom Reports Using JMP and JSL

Lucas Beverlin

September 14, 2016

Outline

- Introduction
- Creating Individual Reports
- Tying Them Together
- Editing the Reports

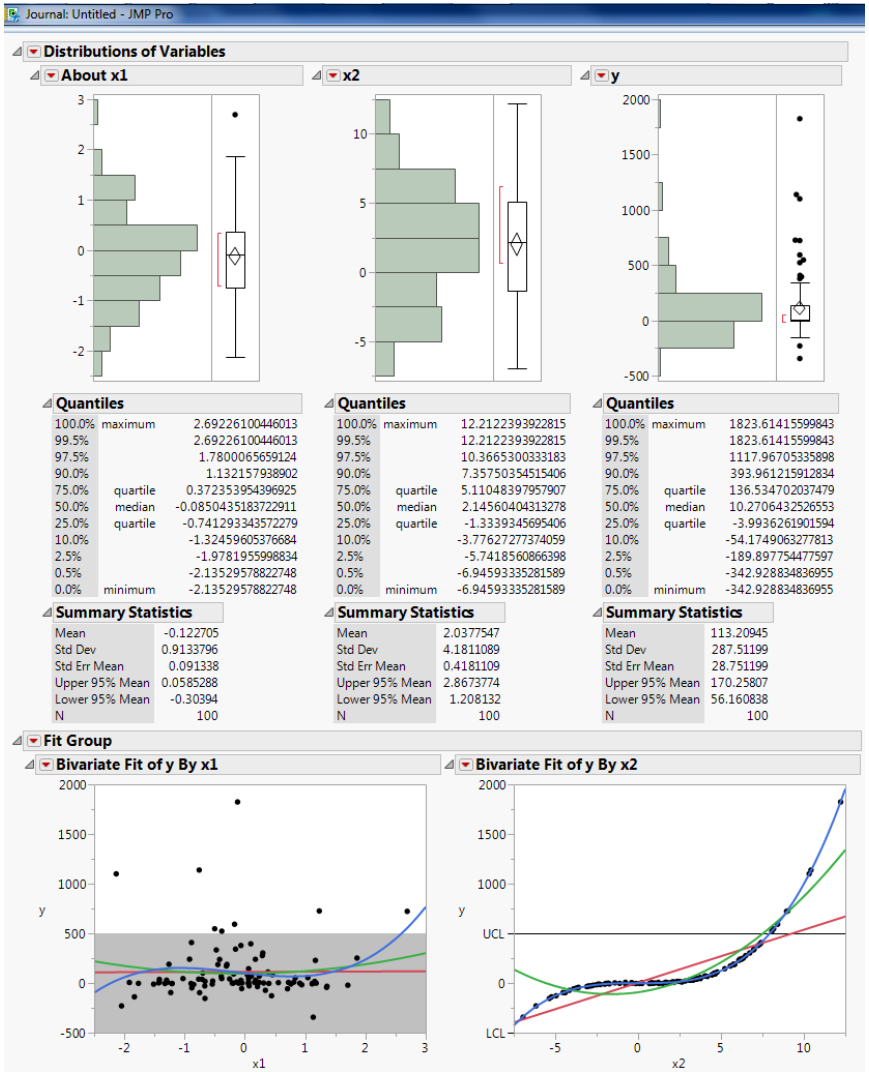
This presentation was done in JMP 12.2.0.

Introduction

- JMP can be used to do many analyses.
- What if you need to explain how you came to a conclusion, when you used different platforms?
- What if you need to repeat an analysis every so often?

JMP Journal

JMP has the ability to record what you have done in a journal.



JSL

- JMP has a scripting language, JSL, that allows up to run many analyses at once, whenever one desires.
- This also allows us to customize the reports.
- The JSL code for platform calls can be extracted from the platform output.

Be warned that there are usually multiple ways to code something. For example, for adding a reference line:

```
dtbiv1 = Bivariate(
        Y( :y ),
        X( :x1 ),
        Fit Line( Line Color( 213, 72, 87 ) ),
        Fit Polynomial( 2, Line Color( 57, 177, 67 ) ),
        Fit Polynomial( 3, Line Color( 64, 111, 223 ) ),
        SendToReport(
                Dispatch(
                        ,
                        "2",
                        ScaleBox,
                        Add Ref Line( {-500, 500}, "Solid", "Black", "", 1, 0.25 )
                )
        )
)
```

JSL

```
dtbiv1 = Bivariate(
        Y( :y ),
        X( :x1 ),
        Fit Line( Line Color( 213, 72, 87 ) ),
        Fit Polynomial( 2, Line Color( 57, 177, 67 ) ),
        Fit Polynomial( 3, Line Color( 64, 111, 223 ) ) )
);
report(dtbiv1)[Axis Box(1)]<< Add Ref Line( {-500, 500}, "Solid", "Black", "", 1, 0.25 )
```

Scripting a Journal versus the Report

Be warned that you can script some edits to a journal, but you can do more editing to the report before journaling.

So what messages can you send to the report itself and to the journal itself?

```
showproperties(report(dtbiv1))
```

```
showproperties(Current Journal())
```

Fit Group Properties

```
Profiler(Graphically explore the prediction equation by slicing it one factor at a time. Features for
optimization.)
Contour Profiler(Shows the contours of the response(s) graphically for two factors at a time.)
Surface Profiler
Arrange in Rows [Action](Specify how many reports across the window.)
Order by Goodness of Fit [Action](Reorder the reports by significance of fit)
Script [Subtable]
  Redo Analysis [Action](Rerun this same analysis in a new window. The analysis will be different if
the data has changed.)
  Relaunch Analysis [Action](Return to the launcher for this analysis.)
  Automatic Recalc [Boolean](Redo the analysis automatically for exclude and data changes.)
  Copy Script [Action](Create a script (JSL) to produce this analysis, and put it on the clipboard.)
  Save Script to Data Table [Action](Create a script (JSL) to produce this analysis, and save it as a
table property in the data table.)
  Save Script to Journal [Action](Create a script (JSL) to produce this analysis, and add a Button to
the journal containing this script.)
  Save Script to Script Window [Action](Create a script (JSL) to produce this analysis, and append it
to the current Script text window.)
  Save Script to Report [Action](Create a script (JSL) to produce this analysis, and show it in the
report itself. Useful to preserve a printed record of what was done.)
  Save Script for All Objects [Action](Look at the all the live scriptable objects owned by reports
in this window, and collect all the scripts to reproduce them. Useful when you have BY groups or
other similar multiple reports)
  Save Script for All Objects To Data Table [Action](Look at the all the live scriptable objects
owned by reports in this window, and collect all the scripts to reproduce them. Useful when you have
BY groups or other similar multiple reports)
  Save Script to Project [Action](Create a script (JSL) to produce this analysis, and save it in a
project.)
  Get Script [Action] [Scripting Only]
  Get Script With Data Table [Action] [Scripting Only]
  Get Data Table [Action] [Scripting Only]
  Get Timing [Action] [Scripting Only]
  Data Table Window [Action](Pop the window to the front for the data table used in this analysis)
  Local Data Filter [Action](To filter data to specific groups or ranges, but local to this platform)
  Remove Local Data Filter [Action](If a local data filter has been created, this removes it and
restores the platform to use all the data in the data table directly)
  Column Switcher [Action](Adds a control panel for changing the platform's variables)
  Remove Column Switcher [Action] [Scripting Only] (Remove the Column Switcher(s))
  View Web XML [Action](Return the XML used to create the Interactive HTML report)
  Get Web Support [Action] [Scripting Only] (Return a number indicating the level of Interactive HTML
support for the display object. 1 means some or all elements are supported. 0 means no support.)
  Report [Action] [Scripting Only]
  Top Report [Action] [Scripting Only]
  Title [Action] [Scripting Only]
  SendToReport [Action] [Scripting Only]
  SendToByGroup [Action] [Scripting Only]
  Action [Action] [Scripting Only]
  Ignore Platform Preferences [Boolean] [Scripting Only]
```

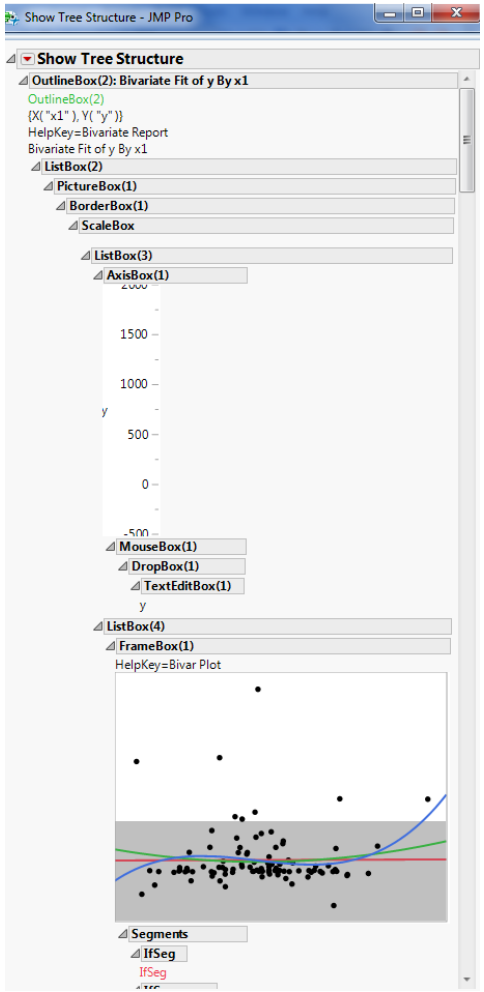
Journal Properties

```
Select [Action]
Deselect [Action]
Reshow [Action] [Scripting Only]
Inval [Action] [Scripting Only]
Update Window [Action] [Scripting Only]
Inval Size [Action] [Scripting Only]
Journal [Action]
Copy Graph [Action]
Copy Picture [Action]
Copy Data [Action]
Page Break [Action]
Set Report Title [Action] [Scripting Only]
Set Window Title [Action] [Scripting Only]
Get Annotation [Action] [Scripting Only]
Add Text Annotation [Action] [Scripting Only]
Add Line Annotation [Action] [Scripting Only]
Add Simple Shape Annotation [Action] [Scripting Only]
Add Polygon Annotation [Action] [Scripting Only]
Add Pin Annotation [Action] [Scripting Only]
Class Name [Action] [Scripting Only]
Window Class Name [Action] [Scripting Only]
Child [Action] [Scripting Only]
Sib [Action] [Scripting Only]
Parent [Action] [Scripting Only]
Next [Action] [Scripting Only]
Top Parent [Action] [Scripting Only]
Append [Action] [Scripting Only]
Sib Append [Action] [Scripting Only]
Sib Prepend [Action] [Scripting Only]
Delete Box [Action] [Scripting Only]
Prepend [Action] [Scripting Only]
Find [Action] [Scripting Only]
Launch [Action] [Scripting Only]
Dispatch [Action] [Scripting Only]
Show Properties [Action]
Show Tree Structure [Action]
Get Text [Action] [Scripting Only]
Get HTML [Action] [Scripting Only]
Get RTF [Action] [Scripting Only]
Get Journal [Action] [Scripting Only]
Get Script [Action] [Scripting Only]
Get Picture [Action] [Scripting Only]
Get XML [Action] [Scripting Only]
XPath [Action] [Scripting Only]
Save Text [Action] [Scripting Only] ((filePath|""))
Save HTML [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save RTF [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save PDF [Action] [Scripting Only] ((filePath|""))
Save Journal [Action] [Scripting Only] ((filePath|""))
Save Picture [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save Capture [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save MSWord [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save Interactive HTML [Action] [Scripting Only] ((filePath|"",PNG|Native|JPEG))
Save Presentation [Action] [Scripting Only]
Clone Box [Action] [Scripting Only]
  Get Window Size [Action] [Scripting Only]
  Set Window Size [Action] [Scripting Only]
```

```
  Set Window Size [Action] [Scripting Only]
  Get Content Size [Action] [Scripting Only]
  Set Content Size [Action] [Scripting Only]
  Zoom Window [Action] [Scripting Only]
  Size Window [Action] [Scripting Only]
  Move Window [Action] [Scripting Only]
  Bring Window To Front [Action] [Scripting Only]
  Get Window Position [Action] [Scripting Only]
  Show Window [Boolean] [Scripting Only]  [Default On]
  Get Show Window [Action] [Scripting Only]
  Minimize Window [Action] [Scripting Only]
  Maximize Window [Action] [Scripting Only]
  Pad Window [Action] [Scripting Only]
  Print Window [Action] [Scripting Only]
  Close Window [Action] [Scripting Only]
  Maximize Display [Action] [Scripting Only]
  Get Window Title [Action] [Scripting Only]
  Set Window Title [Action] [Scripting Only]
  Get Window Icon [Action] [Scripting Only]
  Set Window Icon [Action] [Scripting Only]
  On Close [Action] [Scripting Only]
  Set Main Window [Action] [Scripting Only]
  Set Print Headers [Action] [Scripting Only]
  Set Print Footers [Action] [Scripting Only]
  Set Page Setup [Action] [Scripting Only]
  Get Page Setup [Action] [Scripting Only]
  Get Web Support [Action] [Scripting Only] (Return a number indicating the level of Interactive HTML
support for the display object. 1 means some or all elements are supported. 0 means no support.)
Scroll Window [Action] [Scripting Only]
Journal Window [Action] [Scripting Only]
Get Size [Action] [Scripting Only]
Get Min Size [Action] [Scripting Only]
Set Min Size [Numeric Point] [Scripting Only]
Get Max Size [Action] [Scripting Only]
Set Max Size [Numeric Point] [Scripting Only]
Get Auto Stretching [Action] [Scripting Only]
Set Auto Stretching [Numeric Point] [Scripting Only]
Get Auto Centering [Action] [Scripting Only]
Set Auto Centering [Action] [Scripting Only]
Get Row States [Action] [Scripting Only]
Make RowState Handler [Action] [Scripting Only]
Get Width [Action] [Scripting Only]
Set Width [Action] [Scripting Only]
Get Height [Action] [Scripting Only]
Set Height [Action] [Scripting Only]
Background Color [Color] [Scripting Only]
Text Color [Color] [Scripting Only]
Visibility [ActionChoice] [Scripting Only]  {Visible, Hidden, Collapse}
Padding [Numeric Sides] [Scripting Only]
Border [Numeric Sides] [Scripting Only]
Margin [Numeric Sides] [Scripting Only]
Vertical Alignment [ActionChoice] [Scripting Only]  {Default, Top, Center, Bottom}
Horizontal Alignment [ActionChoice] [Scripting Only]  {Default, Left, Center, Right}
  Set Property [Action] [Scripting Only]
  Get Property [Action] [Scripting Only]
  Get Property List [Action] [Scripting Only]
  Get Properties [Action] [Scripting Only]
```

Editing

There are many edits we can make to either the journal or the report. Consider the tree structure of a report.



Aim Carefully

We can use the name of a box or the number of a particular type of box to call it. From that we can send messages to that particular box.

For example, suppose we simply want to make sure we've selected the correct box.

```
report(dtbiv1)[Axis Box(1)]<<Select;
wait(1);
report(dtbiv1)[Axis Box(1)]<<Deselect;

report(dtbiv1)["Bivariate Fit of y By x1"][Axis Box(1)]<<Select;
wait(1);
report(dtbiv1)["Bivariate Fit of y By x1"][Axis Box(1)]<<Deselect
```

By Statements

Be careful when doing an analysis by some variable and scripting edits.

```
dtdis2 = dt<<Distribution(
        Continuous Distribution( Column( :x1 ) ),
        Continuous Distribution( Column( :x1 ) ),
        Continuous Distribution( Column( :y ) ),
        By( :Group )
);
```

Note that a call to dtdis2 returns a list of Distribution platforms!

```
//:*/
dtdis2
/*:

{Distribution[], Distribution[]}
```

By Statements

While the list for Distribution had 2 entries, not all platforms do it this way.

```
dtfitbiv = dt<<Bivariate(
        Y( :y ),
        X( :x1, :x2 ),
        By( :Group )
);
```

This time the list has 4 entries!

```
//:*/
dtfitbiv
/*:

{Bivariate[], Bivariate[], Bivariate[], Bivariate[]}
```

How to get to a display box?

It appears that there are two ways that JMP formats its reports[2]:

- Each X (or the platform's rough equivalent) and By group combination is its own report.
  - Bivariate, Oneway, Logistic, Fit Y by X, Contingency, Variability Chart, Contour Plot
- Each X is nested within a By group's report.
  - Distribution, Fit Model, Control Chart

[2] Thanks to Utlaut, Morgan, and Anderson for this!

By Statements

To make an edit to a display box, you'll need to know which format you're in:

```
report(dtdis2[1])["x1"][Axis Box(1)] << Add Ref Line( -1, "Solid", "Black" )

report(dtfitbiv[1])[Axis Box(1)] << Add Ref Line( -1, "Solid", "Black" )
```

When each X and By group combination is its own report, be sure to check that you have the right report!

Report vs Report Message

Also notice the following difference between the report function and the report message for when each X is nested within a By group:

```
//:*/
rptdis = report(dtdis2);
/*:


.
//:*/
rptdis2 = dtdis2<<Report;
/*:

{DisplayBox[], DisplayBox[]}
//:*/
dis_lst = rptdis(Outline Box())<<Get Title;
/*:
dis_lst = rptdis( Outline Box() ) << Get Title/*###*/

{}
 ERROR: Send Expects Scriptable Object{100} in access or evaluation of 'Send' , rptdis( Outline Box() )
//:*/
dis_lst2 = rptdis2(Outline Box())<<Get Title;
/*:

{"Distributions Group=1", "Distributions Group=2"}
```

Where Statements

An advantage with scripting an analysis is that with the Where statement, we can analyze any subset of the data that we choose.

```
dtdis3 = dt<<Distribution(
        Continuous Distribution( Column( :x1 ) ),
        Continuous Distribution( Column( :x2 ) ),
        Continuous Distribution( Column( :y ) ),
        Where(Row() >= 30)
);
```

Wildcarding

In some instances, you will know part of the title of an Outline Box, but not all of it. If it only occurs once, then you can use a wildcard to find it.

```
report(dtdis)[Outline Box("x?")]<< Select;
wait(1);
report(dtdis)[Outline Box("x?")]<< Deselect;
```

XPath

Let's suppose you want to create a table with just the slopes for each model fit with Fit Y by X.

First, you'd need to find the outline boxes that hold those values, then specify where they are within.

For this, we can use XPath.

What does XPath do?

Technically, XPath searches an XML represenation of the JMP report. To see it, send the report the Get XML message.


```
"<OutlineBox width=\!"823\!" height=\!"1453\!" helpKey=\!"FitGroup\!" isOpen=\!"true\!">Fit
Group<!--end of text-->
  <LineUpBox leftOffset=\!"12\!" topOffset=\!"23\!" width=\!"810\!" height=\!"1429\!">
    <OutlineBox width=\!"405\!" height=\!"1429\!" helpKey=\!"Bivariate Report\!"
isOpen=\!"true\!">Bivariate Fit of y By x1<!--end of text-->
      <ListBox leftOffset=\!"12\!" topOffset=\!"23\!" width=\!"392\!" height=\!"349\!">
        <PictureBox width=\!"392\!" height=\!"293\!">
          <BorderBox width=\!"392\!" height=\!"293\!">
            <ScaleBox leftOffset=\!"5\!" topOffset=\!"8\!" width=\!"373\!" height=\!"277\!"
charID=\!"1\!">
              <ScaleBox width=\!"373\!" height=\!"277\!" charID=\!"2\!">
                <ListBox width=\!"373\!" height=\!"277\!">
                  <AxisBox charID=\!"2\!">
                    <MouseBox leftOffset=\!"0\!" topOffset=\!"112\!" width=\!"7\!" height=\!"16\!">
                      <DropBox width=\!"7\!" height=\!"16\!">
                        <TextEditBox width=\!"7\!" height=\!"16\!">y</TextEditBox>
                      </DropBox>
                    </MouseBox>
                  </AxisBox>
                  <ListBox leftOffset=\!"53\!" topOffset=\!"0\!" width=\!"320\!" height=\!"277\!">
                    <FrameBox width=\!"320\!" height=\!"240\!" helpKey=\!"Bivar Plot\!">
                      <IfSeg description=\!"Individual Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <IfSeg description=\!"Fit Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <IfSeg description=\!"Individual Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <IfSeg description=\!"Fit Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <IfSeg description=\!"Individual Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <IfSeg description=\!"Fit Confidence Region\!" isTrue=\!"false\!">
                        <PolySeg/>
                      </IfSeg>
                      <SegPlaceholder description=\!"Grid Lines\!"/>
                      <SegPlaceholder description=\!"Reference Lines\!"/>
                      <IfSeg isTrue=\!"true\!">
                        <MarkerSeg/>
                      </IfSeg>
                      <IfSeg description=\!"Linear Fit\!" isTrue=\!"true\!">
                        <LineSeg/>
                      </IfSeg>
                      <IfSeg description=\!"Fit Confidence Interval\!" isTrue=\!"false\!">
                        <LineSeg/>
                        <LineSeg/>
                      </IfSeg>
                      <IfSeg description=\!"Individual Confidence Interval\!" isTrue=\!"false\!">
                        <LineSeg/>
                        <LineSeg/>
```

Harnessing the Power of XPath

Expressions that can be used in XPath[1]

Expression	Output
/	Selects the root node
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes
*	Wildcard
//StringColBox	Selects all StringColBoxes no matter where they are
//OutlineBox/text()	Returns the titles of all outline boxes
/OutlineBox/FrameBox	Selects all frame boxes that are children of the root element outline box

[1] Thanks Brown and Monsoor!

XPath

Now, to get the outline boxes that contains the slopes, we can do this:

```
outlineboxes = (report(dtfityx))<< XPath("//OutlineBox[text() = 'Parameter Estimates']");
```

XPath

Then we can iterate over the outline boxes:

```
nboxes = NItems(outlineboxes);
slopes = J(6,1,0);
for(i=1,i<=nboxes,i++,
        slopes[i] = num((reportpars[i]<< XPath("//NumberColBox[NumberColBoxHeader/text() =
                'Estimate']/NumberColBoxItem[2]/text()"))[1])
);
```

Moving Things Around

We have the flexibility to move things around, if we are not immediately happy with where things were drawn.

With a JMP journal, we can use the Selection tool to drag boxes around. With JSL, we have to use code to put boxes where we wish.

First, we must call for a New Window. Its first argument is a string that will be the name of the window.

Various Boxes

There can be as many calls to boxes and other things as needed.

- H List Box - Stack boxes horizontally
- V List Box - Stack boxes vertically
- Lineup Box - With the ncol argument, stack ncol of them horizontally, then start a new row and do it again!

One can also use the Append message to include it in various places.

Other Programs Can Be Called from JMP

JMP also allows one to perform operations and draw graphics in other programs. In JMP 11 and 12, MATLAB and R can be used.

- R >= 2.13.0 in Windows
- MATLAB >= 7.14.0 (version R2012a)

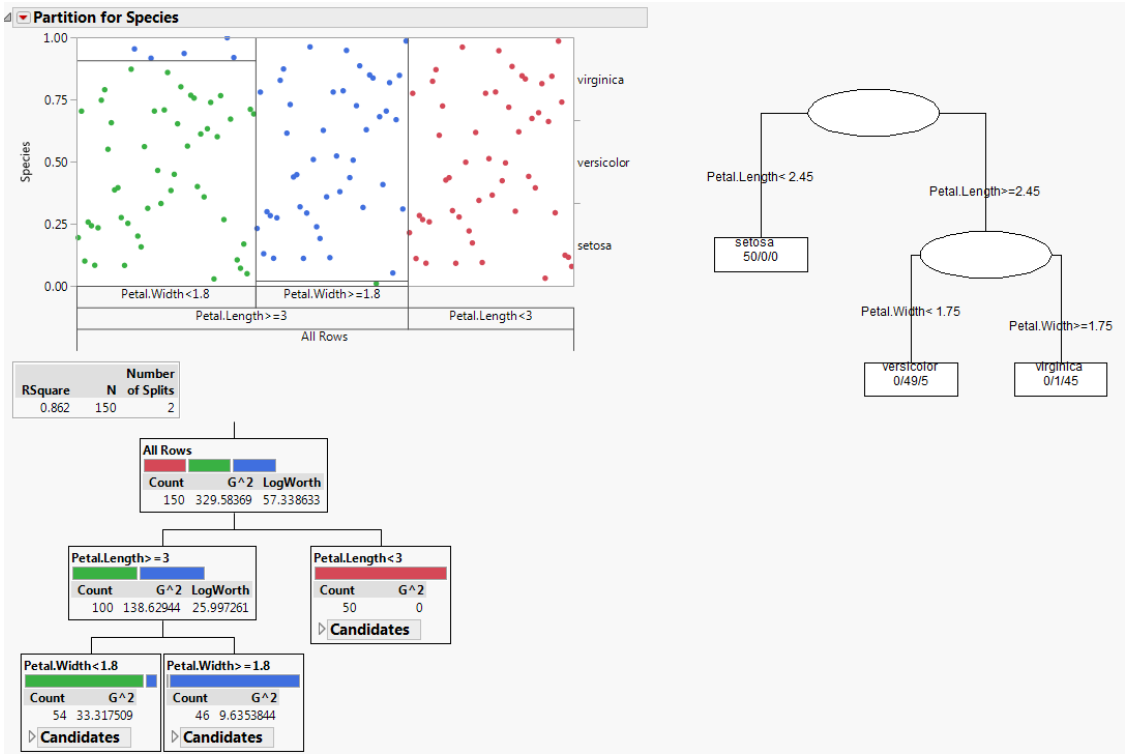
How to Call R from JMP

This is an example from the JMP Scripting Index:

```
R Init( );
R Submit(  "\[plot(function(x) dnorm(x), -5, 5, main = "Normal(0,1) Density")]\"  );
picture = R Get Graphics( "png" );
New Window( "Picture", picture );
Wait( 10 );
R Term( );
```

Using R Output in a Report

One example of R output being used in a JMP report is as follows:



Bibliography

1. Brown, Philip, and Farhan Mansoor. *Mining JMP Reports using XPath.* 2015 Discovery Summit presentation, `https://community.jmp.com/docs/DOC-8116`.
2. Utlaut, Theresa L., Georgia Z. Morgan, and Kevin C. Anderson. *JSL Companion: Applications of the JMP Scripting Language* SAS Press, 2011.