

Advice for learning JSL

© 2010 Erin Vang, PMP — Principal Pragmatist, Global Pragmatica LLC — erin.vang@globalpragmatica.com

Look before you leap

try examples in Sample Scripts folder:

Win: c:\Program Files\SAS\JMP\8\Support Files
English\
Mac: /Library/Application Support/JMP/[8]/English/

RTFM (Read the manual)

flip through chapter 1
you have to read and understand all of chapter 2
read the first few pages (at least) of ch3–4
page through the rest of the book just enough to
learn what's there
don't study anything carefully until you care, then
take your time and read slowly, trying examples
if you think the book is wrong, you're probably
right

look at Builtin Scripts

Win: same as above
Mac: .app's Contents/Resources/*.lproj/

Get your feet wet

try writing an easy script

take your time
figure out what the errors mean
get familiar with your mistakes

try a harder script

try making your easy script groovier
write something that will save you some time

take it to uff da!

think of something your organization should do
smarter
list your goals
think about who's going to use it, how, when
sketch what it should look like
get an expert's advice on how to start
start!
struggle with it, but only so much
get help when you're stuck—you're probably
making it too hard

ask questions

[LinkedIn JMP Scripting group](#)
JMP technical support
Global Pragmatica blog
start a buddy system (iChat or Skype are great!)
any interest in a grassroots JSL wikipedia?

To figure out something specific

search the Scripting Guide PDF

check each hit, hope to find something helpful!

check the Scripting Guide indices

there's a syntax reference index *and* a topic index

check Help/Indexes/...

operator, object, display box

learn from JMP's auto-scripting

do the analysis (or whatever) in the GUI first, then
Save Script
try the script
clean out the junk and try it again

make a few changes and try it again

make an object and get its properties

create it in JSL and save it in a global if you can,
e.g. `myDist=Distribution(...)`;
make the object by hand if you have to, then use
use `show commands(scriptable objects)` to
see what it's called
save it in a global, e.g. `myDist=Distribution[1]`
learn about its messages with `show properties`
(`myDist`)

Scripter beware!

basics

Reformat!
everything returns something
scoping is important
order of execution can be surprising
`obj << message << message and ()`
use `Is Something()` or `Type()` operators to make
sure things are what you think they are

Analyze/Graph platforms

don't expect consistency
do it by hand and see how JMP auto-scripts it
weed out the junk
platforms often can't evaluate their arguments
scoping helps a lot
everything returns something

display boxes

pieces need to be contained in list boxes any time
there's more than one
pieces that aren't display boxes can't go there
Show Tree Structure (Control-Shift-Right-click or
Control-Command-Shift-click)

uff da!

Reformat!
look for unmatched `()`s or `()`s that don't match up
the way you thought they would
look for stray commas that should be semi-colons
look for missing semi-colons

uff da! fida! uff da meg!

put `write()` and `show()` statements at strategic
locations
save often and use version control
`wait()` is your friend, but he's not your good
friend

watch out for data tables not keeping up with JSL
when buttons delete themselves, do it last
and on Windows, don't let buttons delete
themselves at all if you can avoid it

Erin's rules of interface design

If you have to explain too much about how to use
it, you should design it better instead.
Explain everything that isn't dead obvious.
Don't be original if you can be boring.
Alpha test it. If more than two people don't get it,
it's not them—it's your design.

Building a fancy JSL widget from the inside out (and the outside in)

© 2010 Erin Vang, PMP — Principal Pragmatist, Global Pragmatica LLC — erin.vang@globalpragmatica.com

Objective

A basic analysis is buried in JMP and requires several fussy, hard-to-remember steps in a row. The data can be set up two different ways. We want to make an all-in-one window that makes the analysis easy.

Why? To illustrate how to build a custom tool—how to put an analysis in a display box and build a nice graphical user interface. When you get an idea for a widget that will make a difference, this is how to start.

The steps

- Organize data either in one column with binary outcomes on each row, or one column with two outcome rows and a second column with frequencies (counts).
- Make sure the outcome column is nominal.
- Do a Distribution of the outcomes (assign Freq role as needed).
- Find Test Probabilities in the local context menu.
- Fill out a dialog box.

Auto-saved script

```
Distribution(
  Nominal Distribution(
    Column( :Device Success ),
    Test Probabilities( Test(Greater than),
      0.05, 0.95, f )
  )
);
Distribution(
  Freq( :Frequency ),
  Nominal Distribution(
    Column( :Success ),
    Test Probabilities( Test(Greater than),
      0.05, 0.95, f )
  )
);
```

Strategy

Generalize the analysis
Build a custom dialog box
that has error checking
and tool tips
Make it work for either kind of data
different dialog for each
Put it all in one window
with an outline structure
and a Start Over button
Add "About this script" information
and a fancy logo
Use arrays instead of globals
Hide the other globals
Encrypt the script
Give the encrypted script away
Sell the source code (the unencrypted script)

The script (big picture)

```
Expr( get the data );

New Window(
  vlistbox(
    outline box(
      logo, icon, text, links,
    ),
    outline box(
      radio box (raw or summary?),
      dialog box (
        either raw or summary kind,
        get roles, OK, etc.
      )
    )
  )
);

Expr(
  check for errors;
  do the analysis ( either raw or summary);
  put it in another outline box;
  make a Start Over button;
  delete the dialog box's outline box;
);

Expr(
  delete the current analysis;
  get the next data;
  restore the dialog's outline box;
  restore the radio buttons;
  restore the dialog;
);
```

Uff da!

Use an array with a bunch of entries instead of a bunch of globals.
Hide the globals by naming them with a __ prefix.
Encrypt the script (to make it run-only, to prevent changes).
Auto-close the About outline node.

Uff da! Fida!

Add a logo.

Building a fancy JSL widget from the inside out (and the outside in)

© 2010 Erin Vang, PMP — Principal Pragmatist, Global Pragmatica LLC — erin.vang@globalpragmatica.com

The actual script (pre-Uff da version)

```
Try( Dlg_testOneProp<< close window );
Expr_gplogo=Expr(...);
Expr_getData = Expr(
  dt_forOnePropTest= Open(
    Pick File( "Please choose ...:", Get Current Directory(), {"JMP Data|jmp"} )
  );
  List_columns= dt_forOnePropTest<< get column names;
  String_dataType= "Raw data";
);

Expr_getData;
Dlg_testOneProp= New Window( "One Proportion Test",
  V List Box(
    H Center Box( Expr_gplogo ),
    BB_liveArea= Border Box( Left( 10 ), Right( 10 ), top( 10 ), bottom( 10 ),
      V List Box(
        OB_about= Outline Box( "About One Proportion Test",
          H List Box(
            Icon Box( "Distrib" ),
            Text Box( " " ),
            Text Box(
              "One Proportion Test requires either raw data, in which one column records ...)",
              <<set wrap( 300 )
            ),
            Text Box( " " ),
            V List Box(
              Text Box( "Custom JMP script by Erin Vang, PMP, Global Pragmatica LLC" ),
              Button Box( "erin.vang@globalpragmatica.com",
                Web( "mailto:erin.vang@globalpragmatica.com" ),
                underlineStyle( true )
              ),
              ...
            )
          ),
          OB_test= Outline Box( "One Proportion Test" )
        )
      )
    )
  );
);

Expr_showDlg = Expr(
  OB_test<< append(
    PB_assignRoles= V List Box(
      Text Box( "Type of data" ),
      RB_dataType= Radio Box(
        {"Raw data", "Summary data"},
        <<set( 1 ),
        String_dataType= RB_dataType<< get selected;
        Match( String_dataType,
          "Raw data",
          Try(
            BB_freq<< delete;
            CLB_freq<< delete;
          ),
          "Summary data",
          LUB_roles<< append(
            BB_freq= Button Box( "Freq",
              CLB_freq<< Append( LB_allCols<< GetSelected ),
              <<set tip( "One column containing frequencies is required ..." )
            )
          );
          LUB_roles<< append(
            CLB_freq= Col List Box(
              "Frequency column",
```

Building a fancy JSL widget from the inside out (and the outside in)

© 2010 Erin Vang, PMP — Principal Pragmatist, Global Pragmatica LLC — erin.vang@globalpragmatica.com

```
        width( 125 ),
        nlines( 1 ),
        mincol( 1 ),
        maxcol( 1 ),
        numeric
    )
);
);
<<set tip(
    "One Proportion Test requires either raw data, in which one column records...."
)
),
Text Box( " " ),
Panel Box( "Select the existing design factors and (optional) specify how many ...",
    H List Box(
        PB_colList= Panel Box( "Columns",
            LB_allCols= List Box(
                List_columns,
                width( 125 ),
                nlines( Max( N Items( List_columns) / 2, 15 ) )
            )
        ),
        PB_roles= Panel Box( "Assign roles",
            LUB_roles= Lineup Box( N Col( 2 ),
                Button Box( "Outcome",
                    CLB_outcome<< Append(
                        LB_allCols<< GetSelected
                    ),
                    <<set tip( "One outcome column is required for One Proportion Test." )
                ),
                CLB_outcome= Col List Box(
                    "Outcome",
                    width( 125 ),
                    nlines( 1 ),
                    mincol( 1 ),
                    maxcol( 1 )
                )
            )
        ),
        PB_actions= Panel Box( "Actions",
            Lineup Box( N Col( 1 ),
                Button Box( "OK", Expr_doAnalysis ),
                Button Box( "Remove",
                    Try( CLB_outcome<< Remove Selected );
                    Try( CLB_freq<< remove selected );
                    Try( Col_outcome=. );
                    Try( Col_freq=. );
                ),
                Button Box( "Cancel", Dlg_testOneProp<< close window )
            )
        )
    )
);

Expr_doAnalysis = Expr(
    Match( String_dataType,
        "Raw data",
        Try( Col_outcome= (CLB_outcome<< get items)[1] );
        If( Is Missing(Col_outcome),
            Throw(
                "No outcome column:\N One outcome column is required for One Proportion Test
with raw data. Please try again or click Cancel."
            ),
            OB_test<< append(
                VLB_results= V List Box(
```

Building a fancy JSL widget from the inside out (and the outside in)

© 2010 Erin Vang, PMP — Principal Pragmatist, Global Pragmatica LLC — erin.vang@globalpragmatica.com

```
        Button Box( "Start Over", Expr_startOver ),
        Distribution(
            Nominal Distribution( Column( Column( Col_outcome) ), Test Probabilities() )
        )
    );
    PB_assignRoles<< delete;
);,
"Summary data",
Try( Col_outcome= (CLB_outcome<< get items)[1] );
Try( Col_freq= (CLB_freq<< get items)[1] );
If( Is Missing(Col_outcome),
    Throw(
        "No outcome column:\!N    One outcome column is required for One Proportion Test..."
    ),
    If( Is Missing(Col_freq),
        Throw(
            "No frequency column:\!N    One frequency column is required for One Proportion..."
        ),
        OB_test<< append(
            VLB_results= V List Box(
                Button Box( "Start Over", Expr_startOver ),
                Distribution(
                    Freq( Column( Col_freq) ),
                    Nominal Distribution(
                        Column( Column( Col_outcome) ),
                        Test Probabilities()
                    )
                )
            )
        );
        PB_assignRoles<< delete;
    )
);
);

Expr_startOver = Expr(
    Try( Col_outcome=. );
    Try( Col_freq=. );
    Expr_getData;
    Wait( 0 );
    Expr_showDlg;
    String_dataType= "Raw data";
    Wait( .1 );
    Dlg_testOneProp<< bring window to front;
    Current Data Table( dt_forOnePropTest);
    VLB_results<< delete;
);

Expr_showDlg;
Dlg_testOneProp<< bring window to front;
```