Effective Scripting in JMP

HMS Analytical Software GmbH - Dr. P. Warnat

PhUSE 2009
Background: What is JMP?

- application for statistical analysis
- interactive and easy-to-handle user interface

Comparison JMP and SAS

JMP

SAS

JMP & scripting
Overview

• Introduction JSL scripting
• How to work effectively with JSL?
  Recommendations regarding the topics:
  – STRUCTURING AND REUSE
  – UNIT-TESTS
  – DEPLOYMENT
  – CUSTOM REPORTS
  – EXTERNAL TOOLS

• Conclusion
Introduction JSL scripting

```
55 56 57 58 59 60
l_returnValue =
  l_returnValue""

61 62 63
If( N Items( l_result ) > 0 )
  l_result = 'Preprocess'
64 65 66 67 68 69
l_result =

l_returnValue =
```

```
// **
JMP Version()
/*
JMP Version()
" 8.0"
```
Introduction JSL scripting

- **Jmp Scripting Language (JSL):**
  - automation of a sequence of tasks within JMP
  - implementation of your own algorithms within JMP
  - syntax: “functional” programming approach
  - JMP environment components represented as objects with attributes and methods
  - JSL is well documented in the JMP scripting guide
How to work effectively with JSL?

The following slides render a personal list of useful tips resulting out of lessons learned in several mid-size projects utilizing JSL scripting.
STRUCTURING AND REUSE

- Define own functions

- Use local variables

- Use local error handling in every function

- Structure your functions in "modules" with naming conventions

- Use include to use function definitions out of other files
Example – function template

<FUNCTION NAME> = FUNCTION( {<ARGUMENTS>},
LOCAL( {<LOCAL VARIABLES>},
TRY(
    <FUNCTION BODY SCRIPT CODE>
, //CATCH
    <ERROR HANDLING SCRIPT CODE>
);//END TRY
); //END LOCAL
); //END FUNCTION
Example – real function

util_getSumOfListElements = Function( {i_List},
  Local( {l_index, l_return},
    l_return = 0;
    Try(
      For( l_index = 1, l_index <= N Items( i_List ), l_index++,
        l_return = l_return + i_List[l_index]
      );
    ,
    Throw( "Unexpected ERROR in util_getSumOfListElements: " ||
      Char( exception_msg )
    );
    ); //End Try
    l_return;
  ); // End Local
); // End Function
UNIT-TESTS

• To get JSL code that runs reliable -> run tests!

• For JSL, a unit-test framework named JSL-Unit is available (JMP website)

• Implement your tests as JSL code

Advantages:
  – Executable documentation of what you have tested
  – enables you to repeat execution of your tests easily (to check whether later modifications on your functions have unintended side effects)
JSL Unit Example

Running test "test geometric mean.jsl"

Build/Run Details
Version: 8.0
Build Date: Oct 6 2008, 01:48:11, Release
Run Date: Sep 2, 2009, 12:17:41

JSL Unit Test Summary
Files = 1
Tests = 57
Bypassed Stress Tests = 0
Successes = 57
Failures = 0
DEPLOYMENT

• Provide JSL code to colleagues or customers:
  – Use “//!” for automatic execution on opening the script file
  – Hide JSL code with script encryption
  – Extend JMP menues with new scripts or use
    “deployment containers” for scripts, like JMP journal and project files.
    Both of them can contain scripts or links to script files
CUSTOM REPORTS

- Reuse components of reports generated by build-in JMP methods

- The display tree visualization of an existing report helps a lot

- To get the display tree for a given report window hold down the Control and Shift keys while right-clicking on a blue triangle in the report; select Edit > Show Tree Structure
Example – Display Tree

```
Moments

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>30.708007</td>
</tr>
<tr>
<td>Std Dev</td>
<td>0.0268348</td>
</tr>
<tr>
<td>Std Err Mean</td>
<td>0.003795</td>
</tr>
<tr>
<td>Upper 95% Mean</td>
<td>30.715633</td>
</tr>
<tr>
<td>Lower 95% Mean</td>
<td>30.70038</td>
</tr>
<tr>
<td>N</td>
<td>50</td>
</tr>
</tbody>
</table>
```

```plaintext
OutlineBox(4)
Moments

TableBox(2)

StringColBox(3)
Mean
Std Dev
Std Err Mean
Upper 95% Mean
Lower 95% Mean
N

NumberColBox(2)
30.708007
0.0268348
0.003795
30.715633
30.70038
50
```
EXTERNAL TOOLS

• Use external software tools to support your JSL code development
  – a version control system
    (e.g. Subversion; http://subversion.tigris.org/)
    → recover old versions of your code whenever you want or need
  – a source code documentation system
    (e.g. Doxygen; http://www.doxygen.org)
    → generate API documentation out of source code documentation
  – a build tool
    (e.g. Apache Ant; http://ant.apache.org/)
    → automate some tasks
      (e.g. copy actual version of your source code files and archive them into a zip-file)
Conclusion

- **JMP**: nice and easy-to-handle user interface

- **JSL scripting**: a mighty scripting language for automation or enhancing of JMP functionality.

- The recommendations presented here will help you implementing even large projects effectively with JSL.

- Give it a try!
Thank you for your attention

Dr. Patrick René Warnat

HMS Analytical Software GmbH
Rohrbacher Str. 26
69115 Heidelberg
Germany

www.analytical-software.de