

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



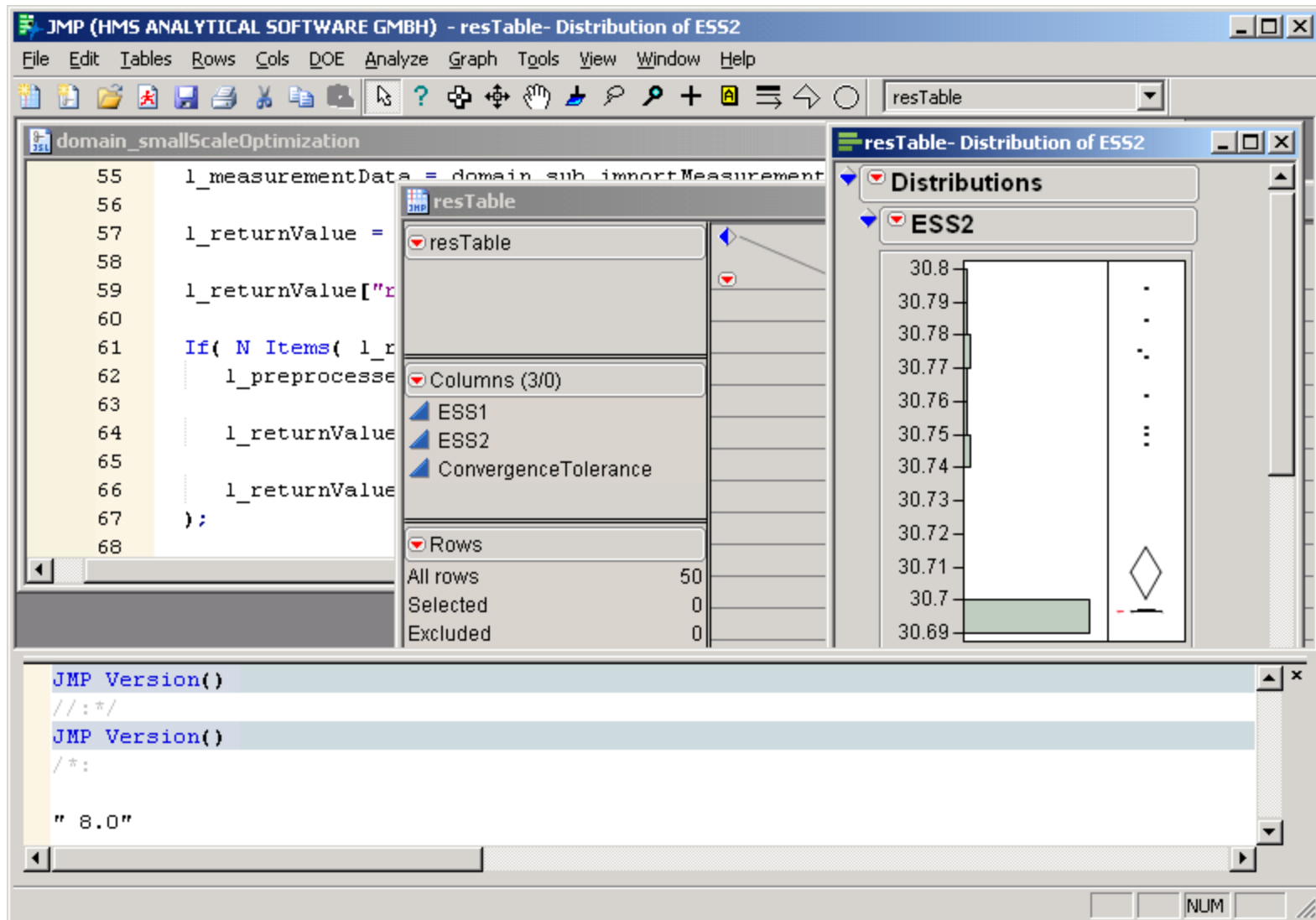
JMP & scripting



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



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

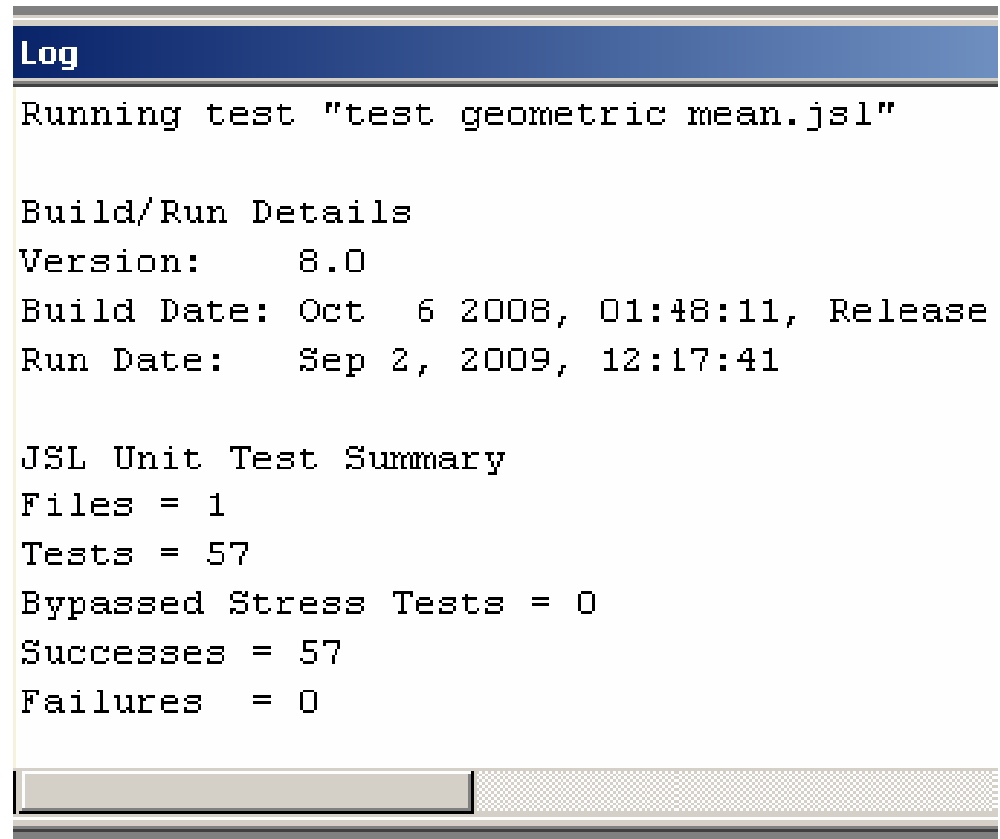
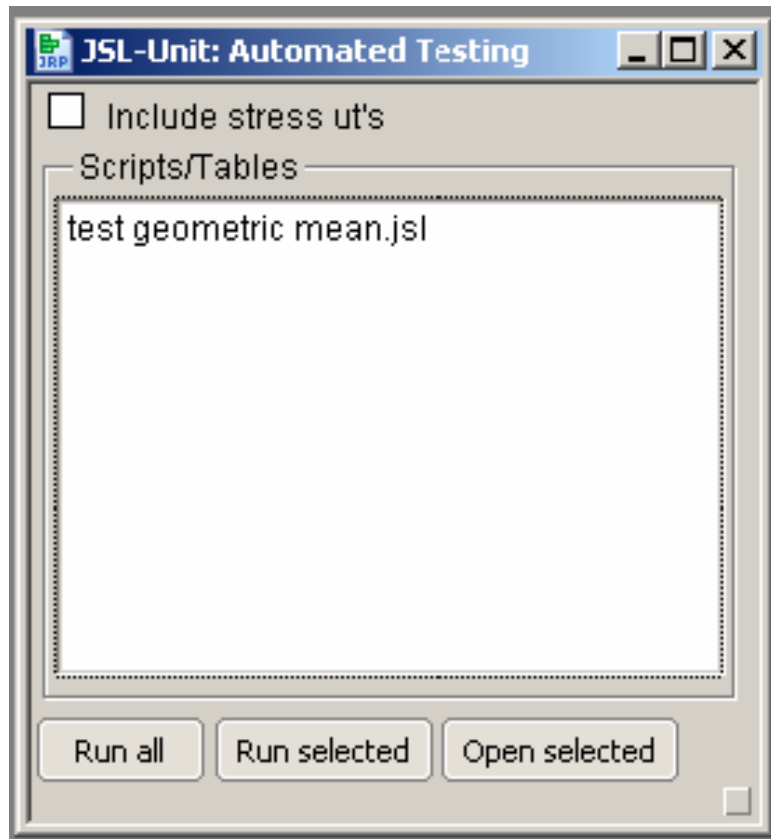
```
<FUNCTION NAME> = FUNCTION( {<ARGUMENTS>},  
    LOCAL( {<LOCAL VARIABLES>},  
    TRY(  
        <FUNCTION BODY SCRIPT CODE>  
    , //CATCH  
        <ERROR HANDLING SCRIPT CODE>  
    );//END TRY  
    );//END LOCAL  
);//END FUNCTION
```

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
```

- 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

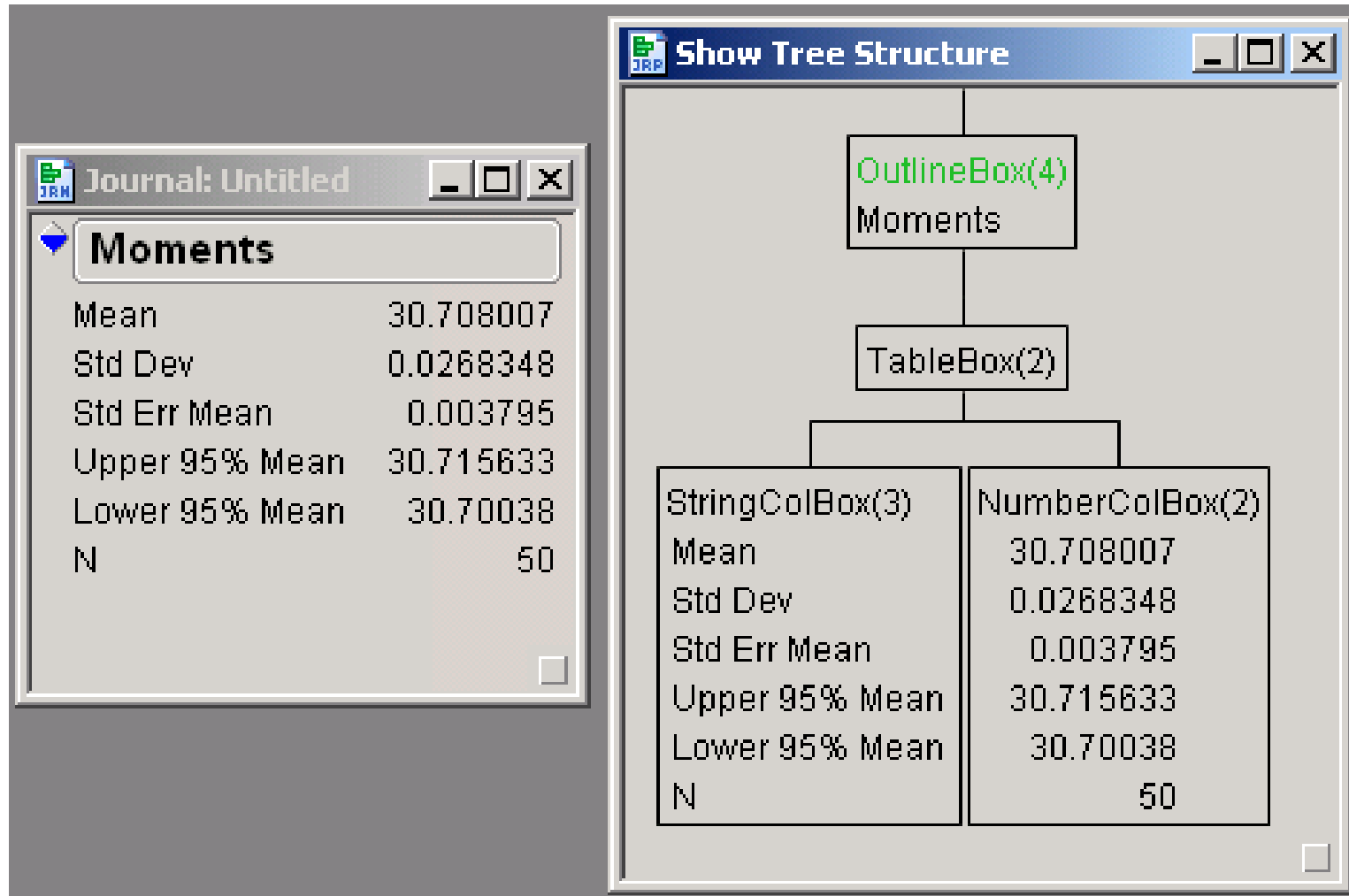


- **Provide JSL code to colleagues or customers:**
 - Use “//!” for automatic execution on opening the script file
 - Hide JSL code with script encryption
 - Extend JMP menues with new scripts or use
 - “deployment containers” for scripts, like JMP journal and project files.
Both of them can contain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



- **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