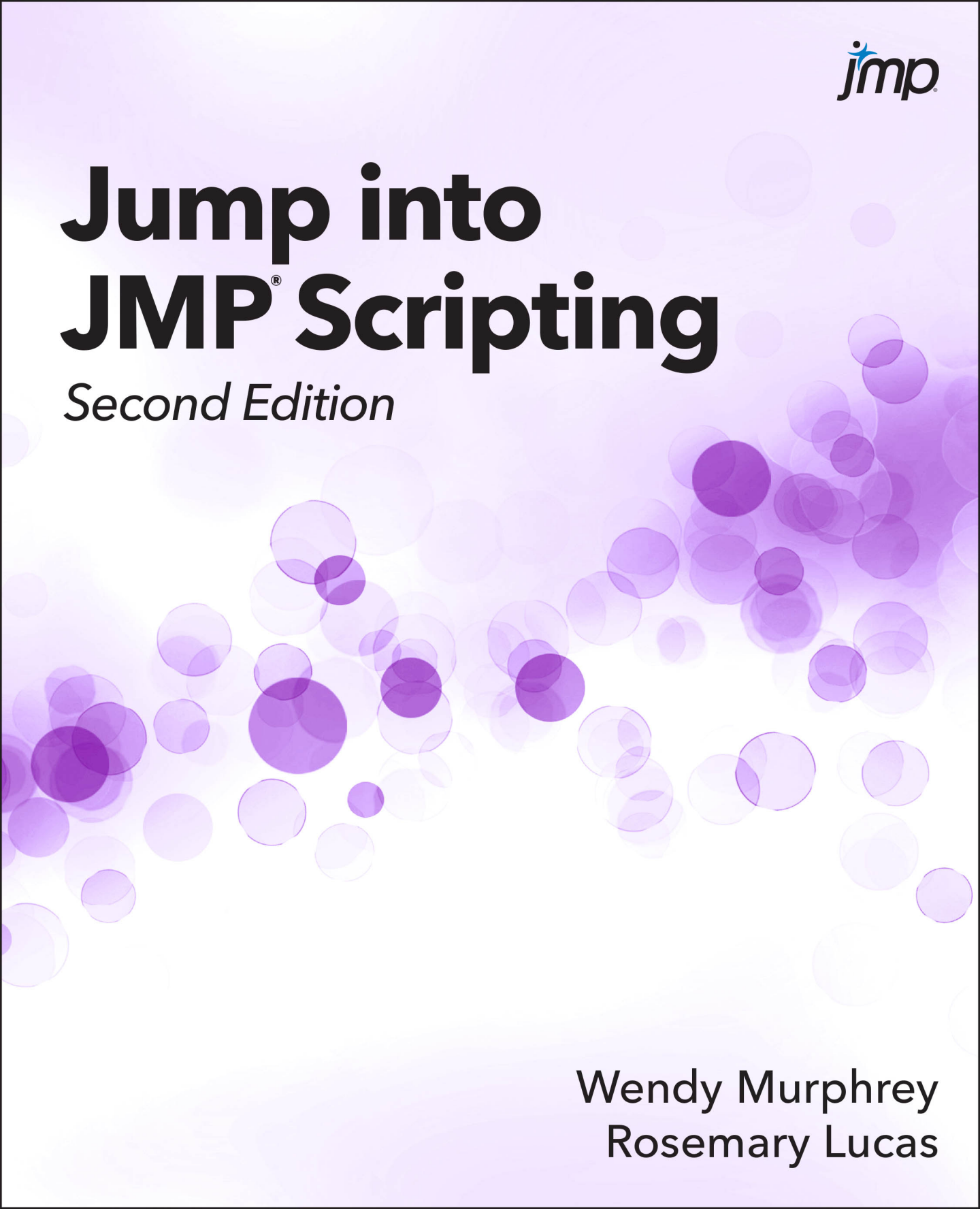

jmp
Jump into
JMP® Scripting
Second Edition
Wendy Murphrey
Rosemary Lucas

The correct bibliographic citation for this manual is as follows: Murphrey, Wendy and Rosemary Lucas. 2018. *Jump into JMP® Scripting, Second Edition*. Cary, NC: SAS Institute Inc.

Jump into JMP® Scripting, Second Edition

978-1-63526-676-4 (Hard copy)
978-1-63526-675-7 (Web PDF)
978-1-63526-673-3 (epub)
978-1-63526-674-0 (mobi)

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

October 2018

Contents

About this Book ..... **vii**
Acknowledgments ..... **xi**

Section 1 ..... **1**
Introduction to JSL 1
Chapter 1: Make JMP Work for You **3**
Overview 3
Capturing Scripts from Your Analyses 3
Capturing Scripts from Imported Data 9
Capturing Scripts from Data Manipulations 10
Creating a Combined Script 15
Summary 23
Chapter 2: Stitching It Together **25**
Overview 25
Stitching Saved Scripts Together 25
Combining Summary Data with an Analysis 25
Combining Data Import, Summary, and Two Analyses 39
Summary 47
Chapter 3: The Double Layer Cake **49**
Overview 49
Folding Back the Layers of the Results 49
The Analysis Layer 49
The Report Layer 53
Summary 71
Chapter 4: JSL Language Foundations **73**
Overview 73
Fundamental Concepts 73
Data Structures 83
Namespaces 88
Name Resolution 92
Conditional Logic 96
Looping 100
Summary 102

Chapter 5: Close Your Eyes and Jump! 103
Overview 103
The Script Editor 103
Resources Available at Your Fingertips 108
What to Do in Times of Trouble 112
Summary 131
Chapter 6: The Dashboard and Application Builders 133
Overview 133
Combine Windows, the Dashboard Builder, and the Application Builder 133
Summary 157
Chapter 7: Advanced Topics 159
Overview 159
Building Custom Dialogs 159
Navigating Reports Programmatically 163
Expression Handling 172
Defensive Programming Strategies 179
Summary 185
Section 2 187
Jump On! 187
Chapter 8: Rows, Columns, and Tables 189
Question 8.1: I can manipulate a data table interactively, such as by selecting and deleting rows, by adding new columns, and so on. How can I get JMP to produce the script for these actions? 190
Question 8.2: How can I select rows where more than one condition is true? How can I select rows where at least one of the specified conditions is true? 192
Question 8.3: There are already rows selected in my data table. How can I select additional rows without deselecting any of the currently selected rows? 194
Question 8.4: How can I select from the currently selected rows? 196
Question 8.5: How can I perform a Select Where on numeric datetime values? 198
Question 8.6: How can I determine the number of selected, excluded, or hidden rows in a data table? 199
Question 8.7: I am attempting to select some rows, and then create a subset data table. When no rows are selected, all the rows are included in the subset data table. How can I create a subset only if there were rows selected? 200
Question 8.8: How can I determine the number of selected columns in a table? How can I get a list of the selected columns? 201
Question 8.9: I want to delete all the character columns in a data table. How can I do this? 202
Question 8.10: Are there any JSL commands for recoding data, as in Cols ▶ Recode? 203
Question 8.11: I have a set of values that I want to use as the control limits for a data table column. How can I use these values to create the Control Limits column property? 206
Question 8.12: How can a character column containing datetime values be converted to a numeric column of JMP date values using a script? 208
Question 8.13: I can replace all the missing numeric values in a table with zeros interactively by performing Edit ▶ Search ▶ Find and Replace. How can I do the same procedure using scripting? 210

Question 8.14: How can I delete the formula from every column in my data table? 211
Question 8.15: How can I use a variable in a new column with a formula? 213
Question 8.16: How can I add multiple columns with formulas in a loop? I want one new column for each continuous column that calculates the mean by a grouping column. 214
Question 8.17: How can I get a list of all open data tables? 217
Question 8.18: How can I open and concatenate multiple files that have the same format? 219
Question 8.19: I have imported several files and performed some data cleanup. How can I concatenate all the open tables without knowing exactly how many tables are open? 221

Chapter 9: Dialog Windows 223

Question 9.1: How can I prompt the user to select columns to be used in an analysis? 224
Question 9.2: How can I stop a script if a user clicks the Cancel button in a dialog window? 226
Question 9.3: How can I prompt the user to select a file in a specific directory? I want to display only those files of a specific type, such as text. 227
Question 9.4: Can a wildcard character be used to open several data tables that contain specific characters in the filename? 228
Question 9.5: How can I can prompt the user for information, and then use that value in the SQL string for extracting data from a database? 230
Question 9.6: How can I prompt the user to select starting and ending dates, as well as a title to be used in a report? 232
Question 9.7: How can I populate a Combo Box based upon what a user selects in another Combo Box, and then generate a subset table that meets the selection criteria? 235
Question 9.8: How can I use Column Switcher to affect multiple graphs in the same window? 237
Question 9.9: Is it possible to show or hide a display box based upon a user's prior selection in the same window? 240
Question 9.10: Why does the Excel Wizard open when I import data from my New Window? 242
Question 9.11: When I try to limit the columns shown in a Col List Box to numeric, no columns appear. Why? 244

Chapter 10: Analyses 247

Question 10.1: How can I save the Parameter Estimates table from my Bivariate analysis into a new data table? 248
Question 10.2: How can I save the Parameter Estimates table from my Bivariate analysis into a new data table when I am using a By variable? 250
Question 10.3: I am using the Neural platform and need to add the Prediction Profiler to the Neural report. How can I do this? 252
Question 10.4: How do I use global variables for limit values in a Control Chart script? 254
Question 10.5: For a control chart, how do I request that JMP open and use a table of saved limits rather than those limits calculated at run time? 255
Question 10.6: How do you save limits from the Control Chart analysis into the column property? 257
Question 10.7: How do you save limits from the Control Chart analysis into a new table? 259
Question 10.8: How can I use spec limits that are stored in a separate data table? 261
Question 10.9: I have many variables in my data table and want to perform a Stepwise regression analysis, and then run my new model. How do I script this? 264
Question 10.10: I need to close the Fit Model dialog window. How do I reference it? 265
Question 10.11: How can I use a list in place of the Fit Model effects? 266

Question 10.12: How do I save X number of principal components of my multivariate analysis to the data table? I also want to include the eigenvectors in my report and save them into a new data table. 267
Question 10.13: How can I save the Mahalanobis outlier, Jackknife distances, and T-square values of the outlier analysis to the data table? 268
Chapter 11: Graph Components 271
Question 11.1: How do I replace the default axis label with a different label in Graph Builder? 272
Question 11.2: The font size, type, style, and color of the axis labels need to be changed in my report. How do I accomplish this? 275
Question 11.3: How do I specify a different font for the tick labels on my graph? 278
Question 11.4: The tick labels on an axis need to be rotated and formatted to two decimal places. How is this done? 280
Question 11.5: How can I specify the order of values on an axis? 282
Question 11.6: How do I add a reference line on any graph? 284
Question 11.7: How can I remove a reference line? 285
Question 11.8: How do I add annotations to my graph using a script? 287
Question 11.9: I have a spec limits table with columns holding limits for each process variable. I want to plot these limits on Oneway plots. How can I do this? 288
Question 11.10: How can I get consistent legend colors when using a local data filter with Graph Builder? 290
Chapter 12: Reports and Journals 293
Question 12.1: At the top of my report, there is a text string that begins “Where…”. How can I remove this information from my report and instead show it in the top OutlineBox? 294
Question 12.2: How can I use a variable in a Where statement so that the window title and Where text show the actual variable values rather than the variable name? 296
Question 12.3: How can I arrange my report so that I have two graphs per row? 298
Question 12.4: How can I get a handle to the journal window so that I can save it? 300
Question 12.5: How can I change the name of the journal window? 302
Question 12.6: How do I insert the contents of a data table into a journal, under a specific OutlineBox? 303
Question 12.7: I need to save my report as a .jrp file in my script. How is this done? 305
Question 12.8: How can I use a map as a filter? 307
Question 12.9: How do I create a tabbed report? 309
Question 12.10: My graph has a local data filter. How can I save both components as one interactive HTML file? 311
Index 313

About This Book

What Does This Book Cover?

This book shows how to get started using the JMP scripting language (JSL) and explains the fundamental concepts needed to make the most of JSL. It also provides a rich source of example scripts for those who need to accomplish a specific task with scripting. As a bonus, information about helpful tools and advanced concepts has been added to this, the second edition.

Is This Book for You?

If you want to automate JMP analyses that you perform on a regular basis, want to know more about customizing your JMP reports, or need a good source for example scripts, this book is for you.

What Are the Prerequisites for This Book?

This book assumes that you have experience using JMP interactively and are familiar with the JMP menus and toolbars.

What Is the Scope of This Book?

If you do not know anything about JSL and need a quick start, the first three chapters of this book show you how to capture scripts that JMP generates and pull these scripts together into a cohesive program.

From there, a discussion of the fundamental concepts of JSL is provided to help you understand more about the language. Advanced scripting concepts and some useful tools follow to complete section 1.

If you do know a bit about scripting and seek examples of performing specific tasks, the second section of this book provides diverse example scripts. We hope you find this section of the book to be a good reference resource as you continue learning JSL.

If you are looking for complete and comprehensive treatment of JSL syntax, we recommend that you consult the Scripting Index, *Scripting Guide*, and *JSL Syntax Reference*, which are accessed from the Help menu within JMP software.

What's New in This Edition?

Instructions and examples have been updated, using the new and improved features of JMP software and JSL. Three new chapters about JSL fundamentals, advanced concepts, and helpful tools have been added.

How This Book Is Organized

This book is divided into two main parts. The first section is an introduction to JSL, showing how to use JMP to capture scripts and how to make modifications to combine scripts. This section also contains chapters on advanced scripting concepts, as well as introductions to helpful tools for scripting tasks. The second section of this book contains well over 50 example scripts for common scripting tasks. These scripts can be downloaded from the SAS Press website and modified by users for their specific needs.

Typographical Conventions Used in This Book

The following typographic conventions are used in this book:

regular	is used for most text.
italic	*is used for emphasis and object references within text.*
bold	identifies JMP menu and dialog choices and JSL functions. Choices such as **Help ▶ About JMP** mean you select **Help**, and then select **About JMP**.
//	identifies comments within the code.
/* */	delimits lengthy comments within the code.

Interspersed in the chapters are helpful hints, which are denoted with the shaded gray background and the chair icon. So, when you see the chair icon, hold onto your seat because something important is noted here.

Software Used to Develop the Book's Content

JMP PRO 14.1.0 for Windows was used to develop the scripts and create the screen shots for this book. All scripts were tested using JMP PRO 14.1 for both Windows and Macintosh operating systems.

If you are using a Macintosh operating system, you might find that menu items are different from those described in the book. For example, on Windows you would access the log by selecting the **View** menu. However, on Macintosh you would access the log by selecting the **Window** menu. We recommend reviewing the JMP 14 Menu Card and Quick Reference to find the Macintosh equivalent menu and shortcut items. These PDF files are found by selecting **Help ▶ Books ▶ Menu Card** and **Help ▶ Books ▶ Quick Reference**.

Example Code and Data

You can access the example code and data for this book by linking to the author pages here:
support.sas.com/murphrey
support.sas.com/lucas

References

Morgan, J. 2010. "Expression Handling Functions: Part 1, Unraveling the Expr( ), NameExpr( ), Eval( ),…Conundrum." *JMPer Cable, A Technical Communication for JMP Users* 26 (Winter 2010): 15-19.

SAS Institute Inc. 2018. *JMP® 14 Scripting Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2018. *JMP® 14 JSL Syntax Reference*. Cary, NC: SAS Institute Inc.

Utlaut, T., G. Morgan, and K. Anderson. 2018. *JSL Companion: Applications of the JMP® Scripting Language*, *Second Edition*. Cary, NC: SAS Institute Inc.

The *Scripting Guide* and *JSL Syntax Reference* can be found in the Book section of Help in JMP software installations. They are also available as PDF documents on the JMP website at https://www.jmp.com/en_us/support/jmp-documentation.html.

About the Authors

Wendy Murphrey is a Sr. Principal Technical Support Analyst in the Technical Support Division of SAS Institute, Inc. A JMP user since 2000, she researches and responds to customer questions and problems, writes sample JSL scripts and JMP add-ins, and is an active member of the JMP User Community—all to help customers make the most of their JMP software investment. As team lead, Wendy is a mentor for junior members in the Technical Support Division. Although technical support is her primary focus, Wendy wears many hats at SAS, including serving on the team that developed the JMP Scripting Certification. She received her BS in Business Administration from Meredith College in Raleigh, North Carolina.

Rosemary Lucas is a Database Specialist in the College of Veterinary Medicine and Biomedical Sciences (CVMBS) at Colorado State University, providing database and survey support for the CVMBS College Office. Prior to moving to Colorado, she worked as a Technical Support Analyst in the Technical Support Division, supporting JMP and JMP Genomics, and then later as a JMP Development Tester in the JMP Division at SAS Institute, Inc. A JMP user since 1999, Rosemary received her Master of Microbial Biotechnology degree from North Carolina State University and is a member of the American Statistical Association.

Learn more about these authors by visiting their author pages at support.sas.com/murphrey and support.sas.com/lucas. There you can download free book excerpts, access example code and data, read the latest reviews, get updates, and more.

We Want to Hear from You

SAS Press books are written *by* SAS users *for* SAS users. We welcome your participation in their development and your feedback on SAS Press books that you are using. Please visit sas.com/books to do the following:

- sign up to review a book
- recommend a topic
- request information about how to become a SAS Press author
- provide feedback on a book

Do you have questions about a SAS Press book that you are reading? Contact the author through saspress@sas.com or https://support.sas.com/author_feedback.

SAS has many resources to help you find answers and expand your knowledge. If you need additional help, see our list of resources: sas.com/books.

Chapter 1: Make JMP Work for You

Overview 3
Capturing Scripts from Your Analyses 3
Saving Scripts to a Data Table 5
Saving Scripts to a Script Window 8
Saving Scripts to the Clipboard 9
Capturing Scripts from Imported Data 9
Capturing Scripts from Data Manipulations 10
Creating a Combined Script 15
Importing the Text Data 16
Creating a Distribution Analysis 18
Executing a Script 20
Checking the Log 21
Saving a Script 22
Summary 23

Overview

Have you ever worried that you will have to write pages and pages of code, and won't know where to start? There's no need to fret, because JMP, the best scripter of all, can write the scripts for you. We're going to show you how.

In this chapter, you will learn about:

- capturing scripts from your analyses
- using scripts from imported data and data manipulations
- creating a combined script composed of two separate captured scripts

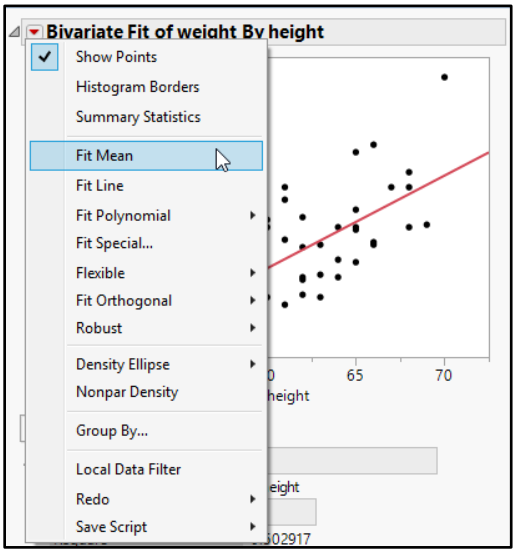
Capturing Scripts from Your Analyses

You just created a report that impresses your manager. She likes it so much that she wants the report weekly. So, what to do?

Do not panic. Instead, save the scripts and execute them next week to create your report.

In your report, you might have noticed the red triangle icons, clicked on a few of the icons, and used some of the options in the drop-down menus to modify or customize your analysis. See Figure 1.1 for an example.

Figure 1.1 Bivariate Menu

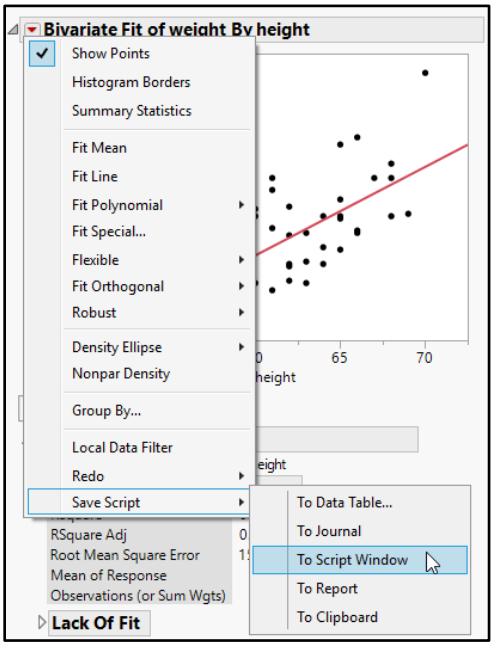


Did you know that you can capture a script of your analysis, plus many of the post-analysis changes you made? While JMP does not record every step you perform interactively, you can reproduce your results with scripting.

In Figure 1.2, notice the last item on the menu is **Save Script**. Selecting it opens a sub-menu that itemizes choices for saving the analysis script to regenerate the report, including most options.

Scripts that are generated by JMP can be captured in a variety of ways using selections in the **Save Script** menu.

Figure 1.2 Save Script Menu



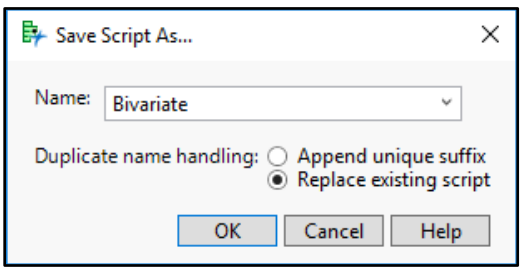
Let's take a look at three of these options.

Saving Scripts to a Data Table

Keeping a script with its associated data is a good idea. The **Save Script ▶ To Data Table** option stores the script as a property of the data table known as a table script. As a table script, it stays with the table until you delete it. You can update or add data to the table, execute the script, and see the results. And when you save the table, you also save the script. So when you give the data table to a colleague, she can open the table, execute the scripts with the data you intended, and view your reports.

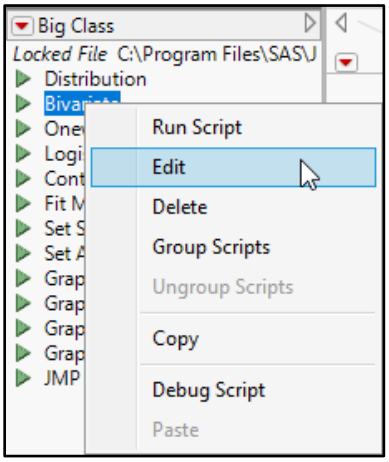
After selecting **Save Script ▶ To Data Table**, you are prompted to provide a name for the table script or accept the default. In this dialog, you can also choose to replace an existing table script that has the same name or simply to append a suffix to ensure a unique name is assigned.

Figure 1.3 Table Script Naming



A green triangle icon, sometimes referred to as a *play* button, appears in the table panel beside the new table script. Clicking the green triangle icon runs the script. Right-clicking the table script name opens a menu with several items: **Run Script**, **Edit**, **Delete**, **Group Scripts/Ungroup Scripts**, **Copy**, **Debug Script**, and **Paste**.

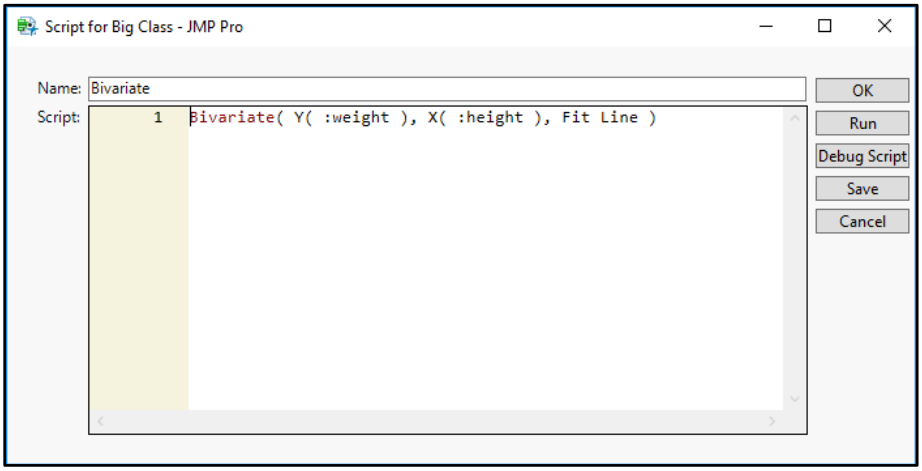
Figure 1.4 Table Script Options



Choosing **Run Script** executes the script. This is the same action as clicking the green triangle icon.

Selecting **Edit** opens a window where you can view and edit the script.

Figure 1.5 Sample Script That Was Saved as a Table Script

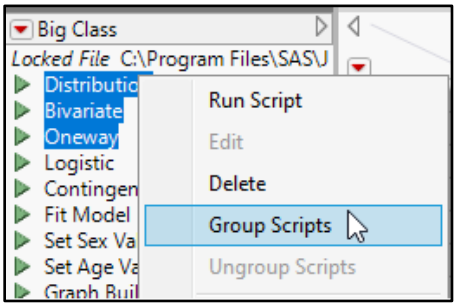


Selecting **Delete** removes all selected table scripts from the data table.

The **Group Scripts** option provides the ability to group selected table scripts. This enables you to organize your table scripts in a manner that makes sense for the data or even your analysis process.

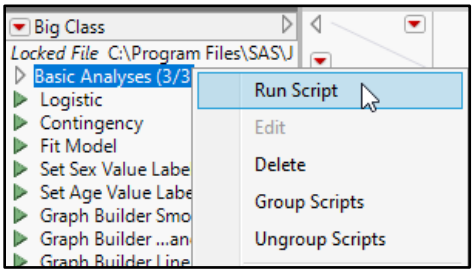
For example, suppose you wanted to arrange the first three table scripts in the Big Class data table as a group called *Basic Analyses*. You would begin by selecting the three table scripts, right-clicking any one of them, and selecting **Group Scripts**.

Figure 1.6 Group Scripts



The default name is based upon the name of the first table script and also provides an indication of the number of table scripts included in the group. Simply click the group name once to rename it to something more meaningful. Right-clicking the group and selecting **Run Script** runs each table script in the order in which they appear in the group.

Figure 1.7 Renamed and Run Group Script



The **Copy** option seems obvious but can be a bit deceiving. In fact, it does more than store the selected table script to your clipboard. The script that is copied includes the entire data table message used to add a new table script, which makes for easy pasting of the table script into a different data table.

To see what we mean, suppose you saved a script in your data table named *Distribution of height*. Selecting **Copy** would place the following code on your clipboard:

```
Add Properties to Table(
    {New Script(
           "Distribution of height",
           Distribution( Continuous Distribution( Column( :height ) ) )
    )}
)
```

If you only wanted the contents of the table script, you have the following options:

1. Paste the preceding code into your main script and delete the unwanted options.
2. Select **Edit,** and then select and copy the desired section of the script.

Selecting the **Debug Script** option opens the JSL Debugger. The details and usage of the JSL Debugger are discussed in chapter 5.

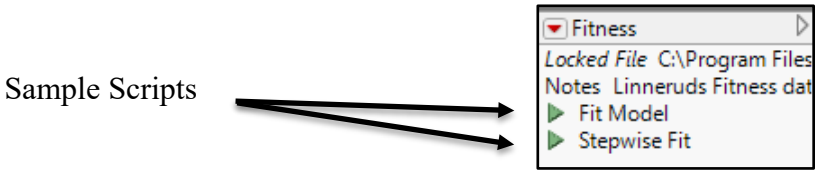
Finally, the **Paste** option is another one that seems straight forward but has a little twist. When you select the **Paste** option, JMP actually just runs the script that is on your clipboard.

For example, if you had the entire **Add Properties to Table()** script, as shown above, stored on your clipboard, the script would be executed and the new table script would appear in the table panel.

On the other hand, if you only had the **Distribution()** code on your clipboard, then the Distribution report would appear and no new table script would be added to the table panel.

Did you know that many of the sample data tables in your JMP installation include scripts that have already been saved? Right-click the green triangle icon to the left of a script that interests you, and then select **Edit** to display a ready-made script.

Figure 1.8 Table Panel with Saved Scripts

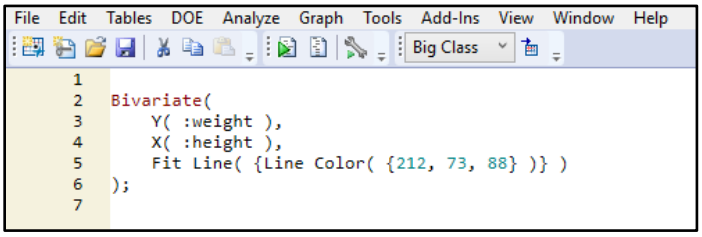


Saving Scripts to a Script Window

The **Save Script ▶ To Script Window** option places your script into a window named Script Window. This window is a script editor where you can edit your saved script. If this window is kept open, you can accumulate scripts from additional analyses by issuing the **Save Script ▶ To Script Window** command from those reports.

It is easy to save your script as a script file from this window. Select **File ▶ Save As**, and then select **Save As Type, JMP Scripts (*.jsl)**.

Figure 1.9 Script Window with Bivariate Script



Saving Scripts to the Clipboard

This is an easy one. The **To Clipboard** option places the script on the clipboard, and then you can paste the script wherever you need it.

Capturing Scripts from Imported Data

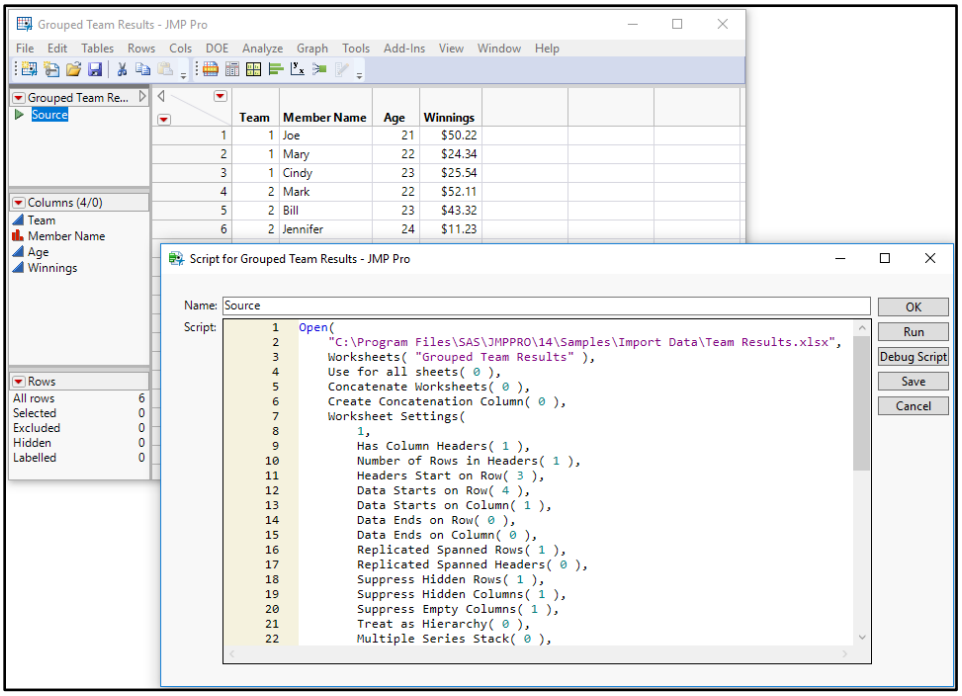
Importing data from various sources can be a tedious, arduous task. For the data types used most often, you can extract the script from JMP after importing the data using interactive methods.

The resulting JMP table automatically includes a table script named **Source**. This table script contains the JSL needed to import the data again using the same settings.

Figure 1.10 Text Import Table

To see the code that JMP generated to reproduce the import of the data, right-click the green triangle icon next to **Source**, and select **Edit**. Notice in the following figure that the code consists of an **Open** statement with all possible settings needed to import the Excel file.

Figure 1.11 Source Script for Excel Import



Using **File ▶ Open,** you can choose to open a variety of file types. Of the file formats supported by default, **Source** table scripts are included when any of the following data files are imported:

- Excel files
- text files
- SAS data sets

A new feature for JMP 14 is the ability to import multiple text data files. By accessing this feature from the **File ▶ Import Multiple Files** menu, you can choose to import text files stored in a variety of formats from a single directory. This feature allows you the flexibility to specify all the files in the directory or to narrow the list by specifying the file types, characters in the name, range of file sizes, or a range of file modified dates. For unstructured file types, there are options for how the data should be imported. There are also options to stack files that have a similar format into a single data table or to open each file in separate data tables. You can also add columns to the resulting data table that identify the name, size, and modified date of the source file.

This handy new feature is already popular with users. Just like other imported data, the resulting JMP data tables will have a **Source** table script containing the code to perform the import with the same settings again. A second table script called **Files** contains a list of files that were imported.

In the **File ▶ Database** menu, there are two options to import data from a database. The **Query Builder** enables you to build complex SQL queries without having to write the SQL statements. Upon running the query, the resulting JMP data table contains a **Source** table script. Passwords are masked by the code %_PWD_%.

The **Open Table** option is the traditional method for connecting to an ODBC data source, which is frequently used to import data using less complicated SQL statements. With default preference settings, the resulting table contains a **Source** table script with the password clearly visible. If you do not see any table scripts in the resulting table, select **File ▶ Preferences** and deselect the check box beside **ODBC Hide Connection String** found in the Tables preference group. With this preference turned off, the **Source** table script will appear the next time you use the **Open Table** method to import data from an ODBC database.

Using **File ▶ SAS**, you can connect to SAS installed on your local machine or SAS that is running on a server. The **Browse Data** option enables you to navigate your libraries to select the data to be imported. The SAS Query Builder is similar to the SQL Query Builder in appearance and functionality, except it is specifically used for importing SAS data. With either of these methods a **Source** table script is included in the resulting JMP data table.

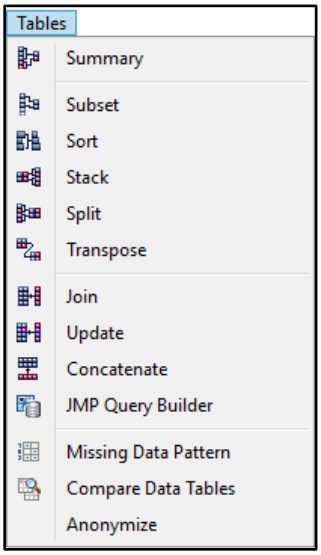
Finally, the **File ▶ Internet Open** menu enables you to import data from a URL. When you use the **Open as Data** option, the data selected from the URL is imported into a new JMP data table. The resulting table contains a **Source** table script.

Capturing Scripts from Data Manipulations

Just as you captured scripts from importing data, you can perform some data manipulation tasks through the interactive menus, and JMP will save the script for you.

Almost every option on the **Tables** menu saves a **Source** table script.

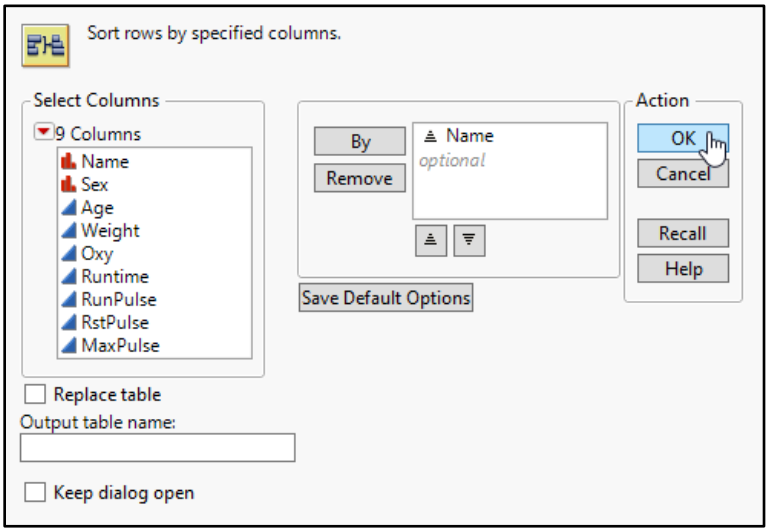
Figure 1.12 Tables Menu



The exceptions are **Anonymize** and any of the other options that enable you to replace or update the existing table.

For example, suppose you wanted to sort the Fitness sample data table by name. You would select **Tables** ▶ **Sort** and cast Name as the By column.

Figure 1.13 Sort by Name



If you click **OK** at this point, a new data table is generated with a **Source** table script that provides you with the code to sort the original data table by the Name column. However, if you select the **Replace table** option, then the data is sorted in place, and no **Source** table script is added to the original table. If you do

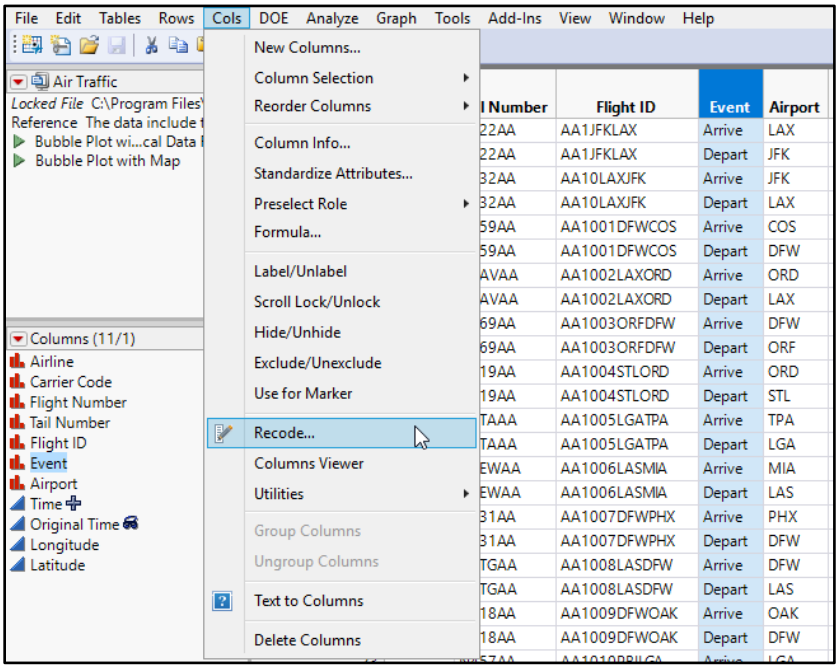
want the table replaced, copy the **Source** script and add a comma plus the option **Replace Table,** as shown below:

```
Data Table( "Fitness" ) << Sort(
    By( :Name ),
    Order( Ascending ),
    Replace Table
)
```

Compare Data Tables is special in that it does not generate a new data table unless requested. In some ways, it is more like an analysis. There is a **Save Script** red triangle option in the results window. If you select **Save Differences Summary** from the red triangle, a new data table appears with a **Source** table script that contains the JSL script to re-create the Compare Data Tables report.

Another type of data manipulation where you can extract the script is **Cols ▶ Recode**.

Figure 1.14 Recode Menu with Column Selected



The key is to extract the script using the red triangle before you dismiss the window. In Figure 1.15, the script was saved for recoding the Event column in place by selecting **Script ▶ Save to Script Window** from the red triangle. You can also save the script for recoding to a new column or a new formula column.

Figure 1.15 Save Recode Script to Script Window

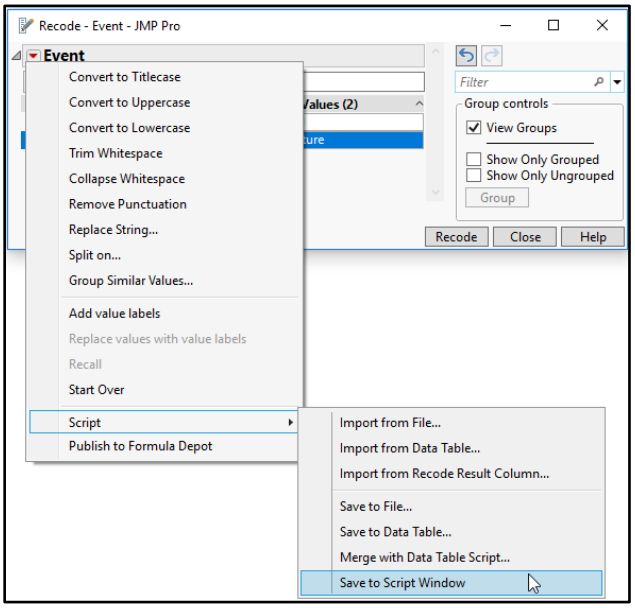
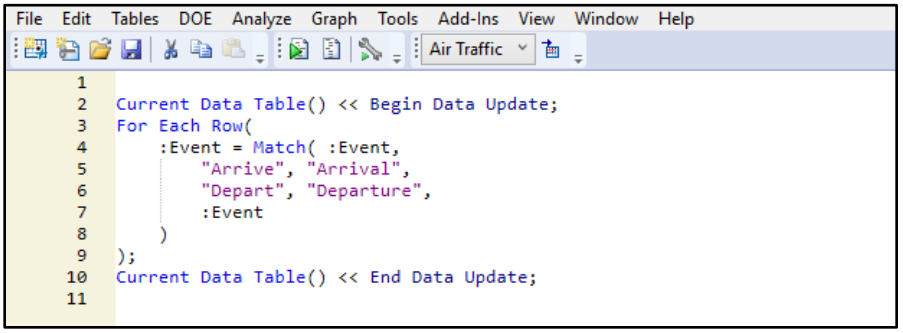
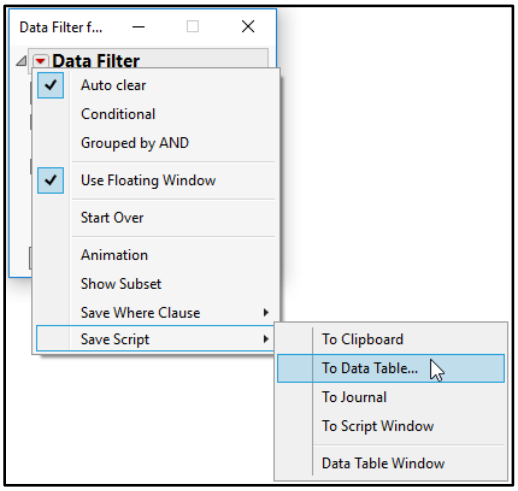


Figure 1.16 Saved Script for Recode



The data filter for the data table, **Rows ▶ Data Filter**, gives users the ability to create complex data subsets, as well as options to hide and exclude portions of data from plots and analyses. After you have done all that hard work to make the perfect filtering setup for your table, you can easily save a script of the data filter for the next time. Click the red triangle icon next to **Data Filter** and select **Save Script ▶ To Data Table,** where it will be conveniently stored with the table. Or you can select one of the other options from the **Save Script** submenu, such as the **To Script Window**, to incorporate the data filter script into a custom script of your design.

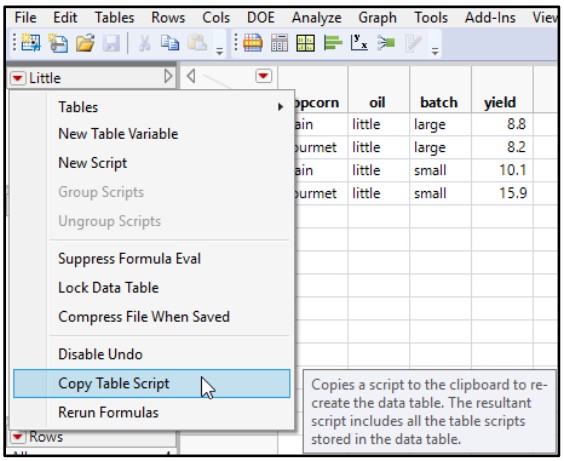
Figure 1.17 Data Filter Save Script Menu



The **Data Filter** option also enables you to capture the selection criteria by selecting **Save Where Clause** from the red triangle menu. The result is a **Select Where()** expression that you could use in a main script. However, unlike other scripts that we have shown you, this code cannot be run as it is. We will demonstrate **Select Where()** in further detail in chapter 7. For now, we just want to point out that you can save the selection criteria.

Another useful feature is the **Copy Table Script** option, located in the red triangle menu to the left of the table name:

Figure 1.18 Copy Table Script Menu Item

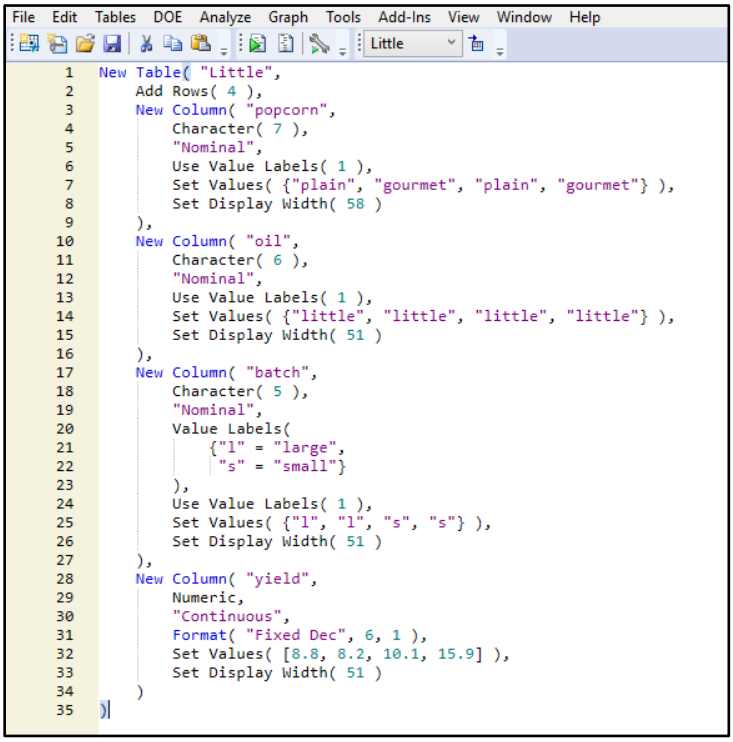


By selecting this option, you can create a script that makes an exact duplicate of your data table. Keep in mind that the JMP application does not record all of your changes and actions made to the data table. Rather, the **Copy Table Script** option captures the current state of the data table in a script when selected.

Why use this option? Capturing a script of your table is a great method for showing you how features are scripted in your table. Want an easy way to capture a script of those formulas or column properties you painstakingly created interactively? Use **Copy Table Script** and paste the captured script into a Script window, where you can locate and extract those portions of the script for use elsewhere.

Keep in mind that **Copy Table Script** places all of the values in each row and column on your clipboard. If your table is rather large, copying and pasting the script will be slower than for smaller data tables.

Figure 1.19 Table Script




```
New Table( "Little",
    Add Rows( 4 ),
    New Column( "popcorn",
        Character( 7 ),
        "Nominal",
        Use Value Labels( 1 ),
        Set Values( {"plain", "gourmet", "plain", "gourmet"} ),
        Set Display Width( 58 )
    ),
    New Column( "oil",
        Character( 6 ),
        "Nominal",
        Use Value Labels( 1 ),
        Set Values( {"little", "little", "little", "little"} ),
        Set Display Width( 51 )
    ),
    New Column( "batch",
        Character( 5 ),
        "Nominal",
        Value Labels(
            {"l" = "large",
             "s" = "small"}
        ),
        Use Value Labels( 1 ),
        Set Values( {"l", "l", "s", "s"} ),
        Set Display Width( 51 )
    ),
    New Column( "yield",
        Numeric,
        "Continuous",
        Format( "Fixed Dec", 6, 1 ),
        Set Values( [8.8, 8.2, 10.1, 15.9] ),
        Set Display Width( 51 )
    )
);
```

Creating a Combined Script

Now that you've seen how easy it is to capture scripts, we are going to show you something a little more interesting. Remember that your manager wants the report generated each week? We will show you how easy it is to create a script that imports the data *and* makes the report.

In this sample, we put together a script that does the following:

- imports some text data
- creates a Distribution analysis

Importing the Text Data

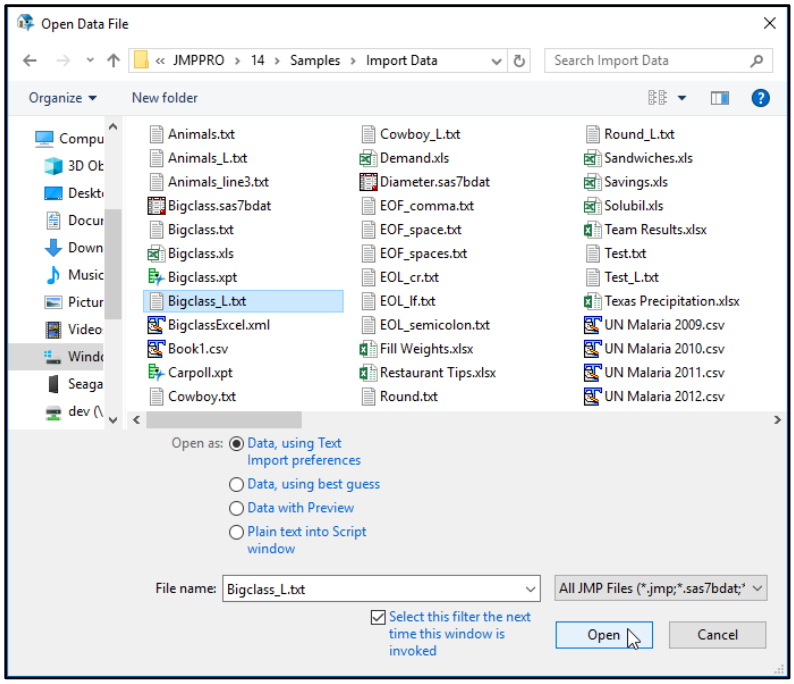
Let's begin by importing the Bigclass_L.txt file.

1. Select **File ▶ Open**. You can find this file in the Sample Import Data folder. For a typical JMP or JMP Pro 14 Windows installation, you would find the file in this directory:

   C:\Program Files\SAS\JMP\14\Samples\Import Data\
   C:\Program Files\SAS\JMPPRO\14\Samples\Import Data\

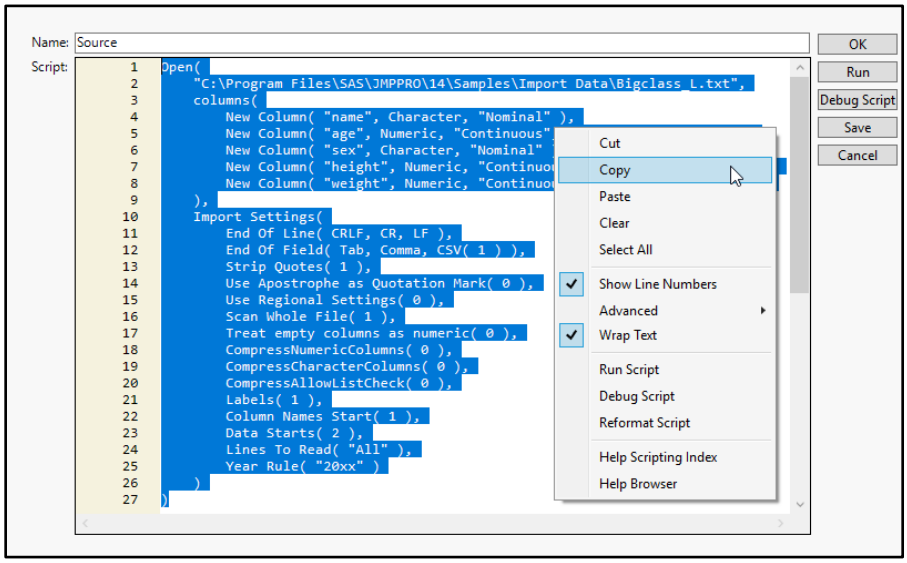
   Bigclass_L.txt is a tab-delimited file. If you use the **Data, using Text Import preferences** option as shown in Figure 1.20, tab must be specified as an End of Field option in your text import preferences (this is a default setting).

Figure 1.20 Open Text File for Windows



2. After the data file is imported into JMP, right-click the green triangle icon beside **Source** and select **Edit**.
3. In the resulting window, select the entire script and copy it by right-clicking the selected text and selecting **Copy**.

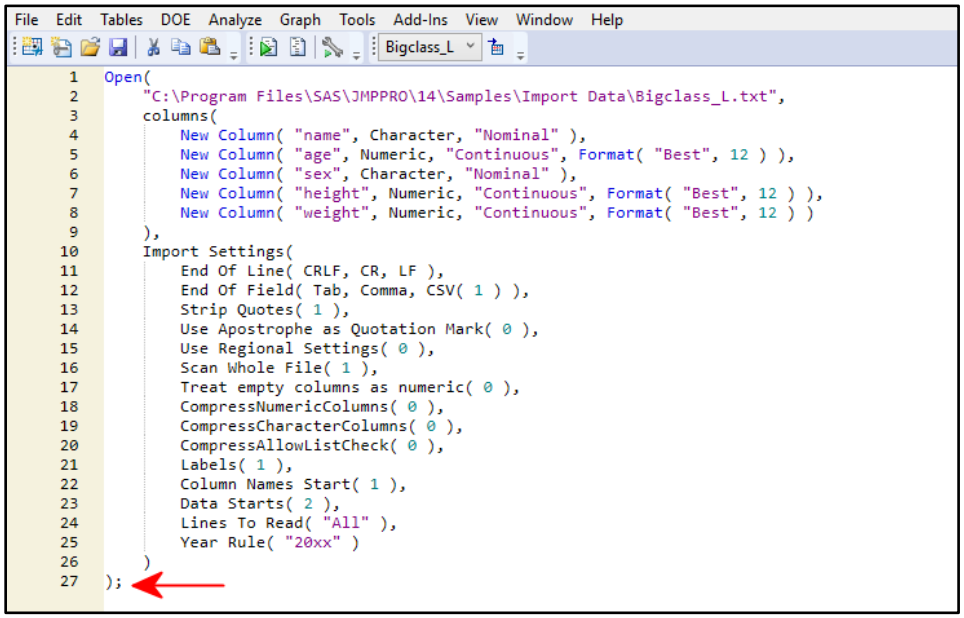
Figure 1.21 Copy of Source Script



4. Click **OK** to dismiss the Source Table Script window.
5. From the **File** menu, select **New ▶ Script**.
6. Paste the script by clicking the **Edit** menu and selecting **Paste**.
7. Add a semicolon after the last closing parenthesis, because you will be adding more code.

The semicolon, known as the **Glue** operator, is necessary in this case because it tells JMP to expect further JSL statements. Additional details about the **Glue** operator can be found in chapter 4.

Figure 1.22 Paste of Import Code



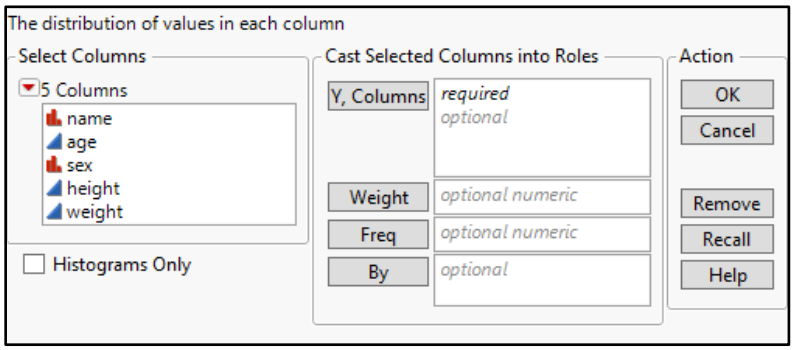
8. Press the **Enter** key a couple of times to move the cursor down a few lines.

Creating a Distribution Analysis

So far, we have pasted the script to import a text file into a Script window. Now, we will create a Distribution analysis and save its script.

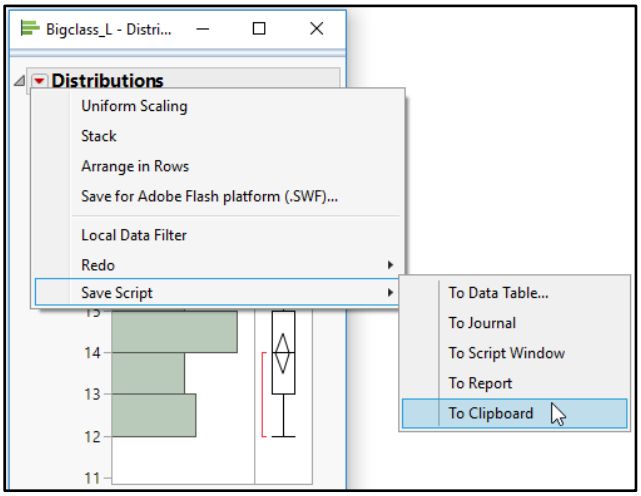
1. From the **Analyze** menu, select **Distribution**.
2. Cast **age** in the **Y, Columns** role and click **OK**.

Figure 1.23 Distribution Launch Window



3. In the Distribution analysis window, capture the script by clicking the uppermost red triangle and selecting **Save Script ▶ To Clipboard.**

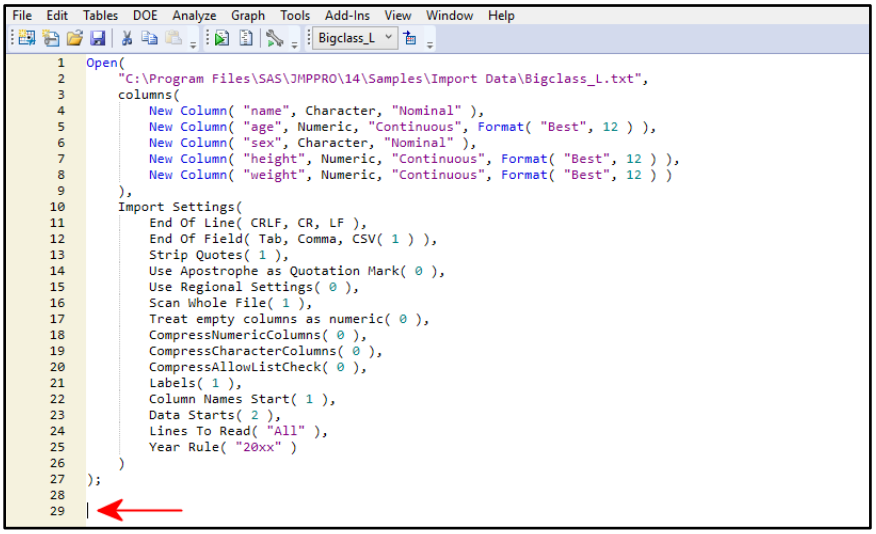
Figure 1.24 Copy Script to Clipboard



This action saves the **Distribution** script onto the clipboard.

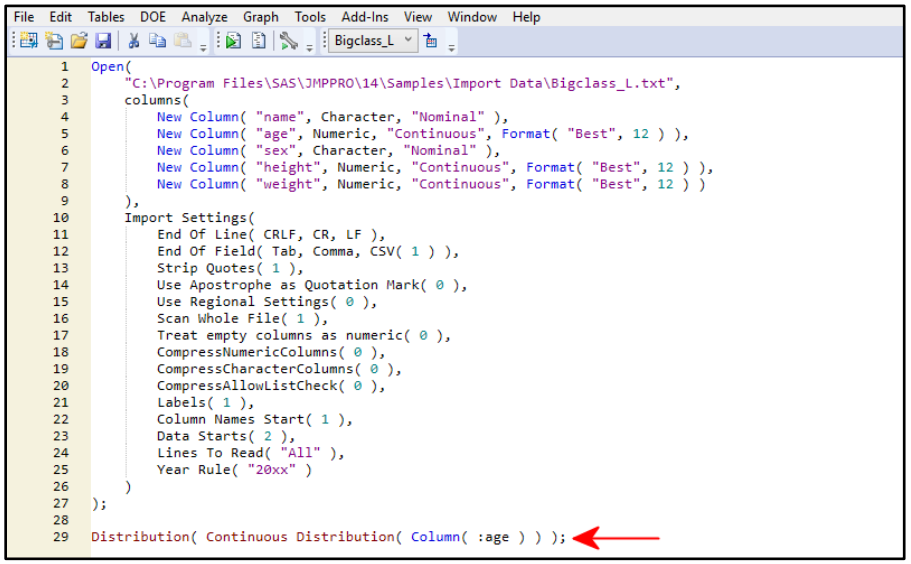
Now return to the Script window that contains the text import script, and place the cursor in the space below the semicolon, near the bottom of the window.

Figure 1.25 Placing the Cursor



4. To paste the **Distribution** script from the clipboard into the Script window, select **Edit ▶Paste**.
5. Add a semicolon after the last closing parenthesis in case you later decide to add code and because this is a good programming practice.

Figure 1.26 Script Window with Distribution Script Pasted



6. Close the Distribution analysis window that we created interactively, and then close the data table, Bigclass_L, leaving only the Script window open.
7. Finally, let's look at the different methods we can use to initiate script execution and then run the script to verify that it produces the expected results.

Executing a Script

There are several ways to execute a JSL script:

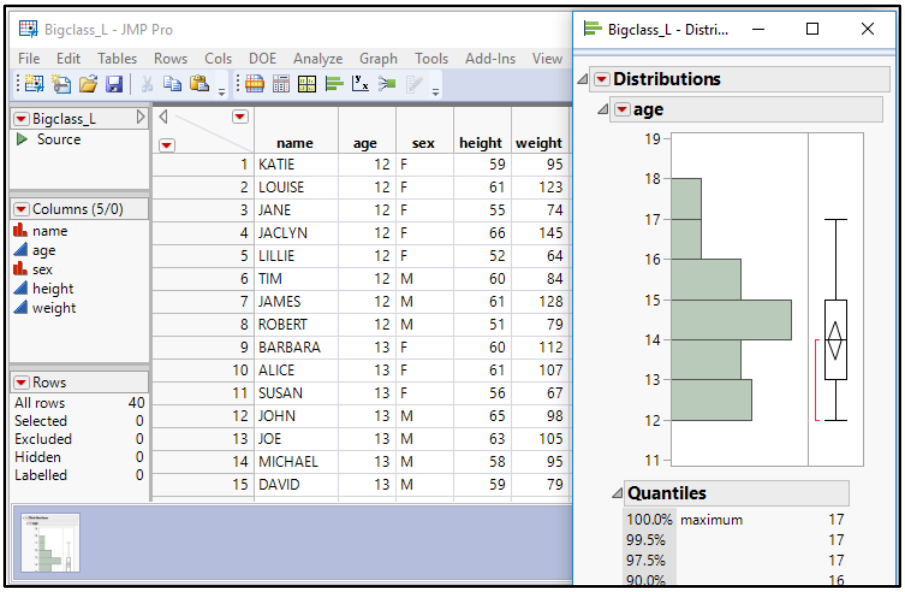
- From the **Edit** menu, select **Run Script**.
- Click the **Run Script** button on the toolbar. 
- Right-click anywhere in the Script Editor window, and select **Run Script** from the pop-up menu.
- Use the keyboard shortcut for this same action: **CTRL+R**.
- Double-click a JSL file from a file browser.

To *execute* or *run* a script means the same thing, and we might use the terms interchangeably throughout this book.

For this case, we will use the first method. From the **Edit** menu, select **Run Script**.

Now you will see that the text was imported into a data table, and the Distribution analysis was created from that data.

Figure 1.27 Results of Executing Combined Captured Scripts



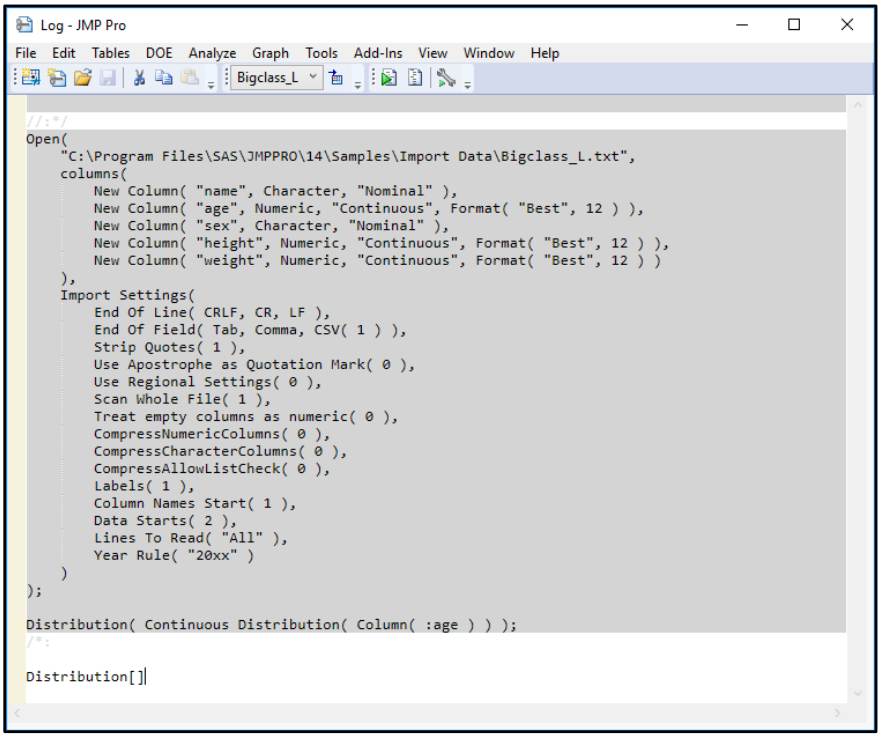
Checking the Log

The Log window is where the code that you executed is displayed, along with any messages JMP has returned. If you have not already done so, display the Log window by clicking the **View** menu, and then selecting **Log**.

You can either leave the Log window as a separate window, or you can embed it into your Script window. To embed the Log into the Script window, right-click anywhere inside the script, and then select **Show Embedded Log**.

The Log window is also a Script Editor, which means that you can select code and execute it directly from within the log.

Figure 1.28 Log Window Showing Code



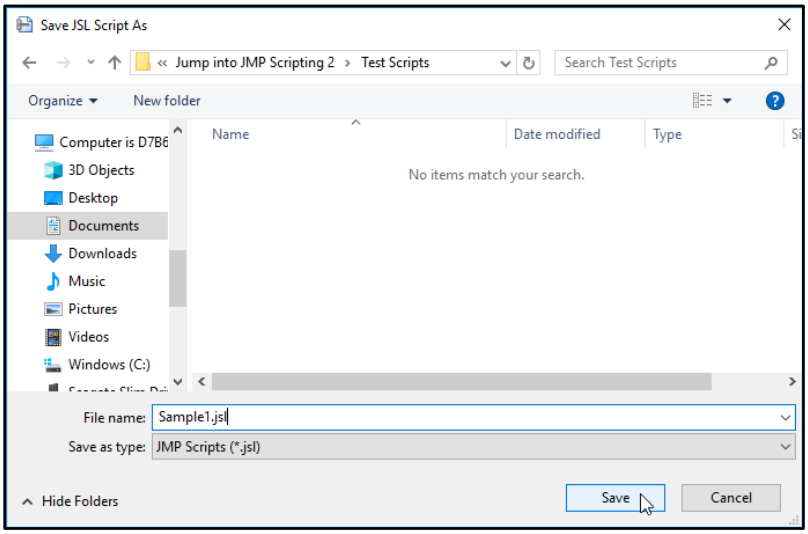
The **Distribution[ ]** shown at the bottom of the log represents the Distribution object that is returned by the Distribution platform. This is simply the return value from running the **Distribution()** script.

Saving a Script

Let's save the script, because we will be coming back to it later.

1. Bring the script that we created to the forefront of the JMP application by selecting **Window ▶ Script**.
2. On the **File** menu, select **Save As**.
3. Navigate to a convenient, yet memorable, location, and name the script **Sample1.JSL.**
4. Click the **Save** button.

Figure 1.29 Save Sample1.JSL



You are finished! You have put together an entire script that will do the following:

- import text data
- create a Distribution report

Summary

Now you've learned how JMP can work for you by extracting scripts of your reports, your imported data, and some data manipulations.

What's next? It's time to roll up your sleeves, because in the next chapter we will show you how to stitch together multiple scripts using the Script Editor.