Writing Programs for JMP® Software

Al M. Best, Virginia Commonwealth University, Richmond VA

ABSTRACT

Repetitive tasks can be challenging to automate with a GUI. Such programs are controlled by various mouse events and keyboard entries that can be more difficult to work with. However, version 3 of JMP on the Macintosh supports AppleScript, a language for controlling applications (programs). In the present implementation of JMP, it is possible to record analyses into a script and replay it later. This paper briefly reviews the AppleScript object-oriented language as it applies to JMP and demonstrates a tool-box of utilities that increase the functionality of scripts for JMP.

One of the benefits of AppleScript is that it enables end users to automate tasks—either routine or highly complex tasks. It can also be used to customize or extend applications. Two examples are shown.

BACKGROUND

Commands and objects

AppleScript is an object-oriented script language (Apple, 1993). At the core of object-oriented systems is the ability of objects to send messages, called commands, to other objects, even in a different application. AppleScript commands include verbs such as open, close, or print. Objects are things in applications such as windows, menus, tables, and graphs as well as applications themselves. Each object understands only certain commands but objects that share similar characteristics, for instance the commands they understand, can belong to the same object class.

Apple events

Apple events are the medium through which AppleScript directs applications to perform tasks. There are system level events that every application should support but developers extend this language by organizing their own events and objects into event suites. Developers register these suites with Apple to insure a standard syntax. As such, AppleScript is an interapplication messaging system that is open. It is extendible by developers but also maintains the consistency of the graphical user interface.

JMP support of Apple events

There are three ways that an application can support Apple events: An application can be scriptable, recordable, and/or attachable. JMP 3.0 does respond to apple events so it is scriptable. However, JMP is about 75% scriptable (SAS Institute, 1994) in that there are many events that JMP responds to through the user interface that can not be duplicated through Apple events. Recordable applications post their internal actions to the operating system so that script recorders can record user actions. Thus users can generate a script without typing anything. Most events in the Analyze and Graph menus and their resulting windows can be recorded. JMP does not record events to the data table or menus, except those specified by the standard Apple events suite. If JMP scripts were attachable, it would be possible to directly extend JMP by hooking a script into JMP itself so that an event generated in JMP could activate a script. Apple mistakenly reports that JMP is attachable (Apple Computer, Inc. (1994)).

Software for Scripting

Required: Macintosh system 7.0.1 or later, 68020 or PowerPC, 4MB of RAM, hard disk, AppleScript 1.1, AppleScript 1.1 includes the ScriptEditor software v1.1 (Apple Computer, 1993a) and FaceSpan software v 1.01 (Apple Computer, 1993b). A newer version of FaceSpan will soon be available from the developer (Software Designs Unlimited, Inc). To run the scripts in this paper, you must have AppleScript, FaceSpan, and PreFab Player.

EXAMPLE 1 – A JMP TOOLBAR

The Windows version of JMP 3.1 includes a toolbar with buttons for some entries in the File menu, all of the Analyze and Graph menu entries, and the Tools menu. Although Macintosh users may find this undesirable, I thought it would be interesting to program one for the Macintosh version.

It is relatively straightforward to create a document window that has no title bar, and is not closable, zoomable, or growable directly below the menu bar. The default FaceSpan script includes code to handle the Apple menu About... dialog, and the Quit command in the File menu. I then used a screen grabber program to capture the buttons from the Windows version of JMP and turn them into color icons using ResEdit. In FaceSpan, color icons can be attached to objects that act like push buttons. The next task was to write scripts for the buttons so that they would do what you would expect toolbar buttons to do. The first button is equivalent to choosing Distribution of Y from the Analyze menu of JMP. Pressing the button should result in a dialog if necessary. Otherwise, the Analyze or Graph result window should appear.

The first problem I ran into was that JMP does not support the scripting of menu events. The JMP Statistics Suite launches Analyses or Graphs with commands like analyze distribution for "Age" from table "Big Class".

This suite thus does not support launching an Analysis or Graph with a dialog. There are various ways around this. The one I choose was the Player software from PreFab. This software supports, among other things, the generation of menu events while within JMP such as the following:

tell application "PreFab Player™" to do menu menu item "Copy" of menu "Edit".

The following handler verifies that JMP is running, activates it and then executes a menu item within a JMP menu. Handlers within objects containing other objects can be particularly useful. The handler below resides in

1188
the toolbar window. The window contains buttons which generate the DoaMenu event. The message is sent to the window. Thus, if a different menu event program were used, the code need only change in one place.

```perl
on DoaMenu(MenuName, Menultem)
tell application "JMP" to activate
tell application "PreFab Player™" to do menu menu item Menultem of menu MenuName
end DoaMenu
```

Each of the buttons has a name. The first button is called "Distribution of Y". Thus, the following script within each of the first seven buttons results in the appropriate Analyze menu command.

```perl
on hiilted theObj
set Menultem to name of theObj
tell window "JMP toolbar" to DoaMenu("Analyze", Menultem)
end hiilted
```

Only one word need be changed in the scripts for the Graph menu buttons.

Future improvements of this toolbar could include buttons to execute command from any of the menus, although I have not discovered how to address the entries in the Tools menu. Another look-and-feel issue is that since this tool-bar is a separate application with its own window, while in JMP you must press a button twice. The first click activates the tool-bar application and the second click presses the button. I presume there is some way to make the scripts work with a single click.

**EXAMPLE 2—DOING ROC CURVES IN JMP**

**Background**

In a situation where you have a test with continuous values predicting a binary outcome, in JMP you perform a logistic regression with the test as the explanatory X column and the outcome as the Y response. If the relationship is nearly a step function, the next step is often to determine the cut-off point that will optimally discriminate between the two outcomes.

Receiver operating characteristic curve (ROC) analysis (Metz, 1978) calculates all possible sensitivities and specificities and finds the value of X that maximizes accuracy. Accuracy is typically defined in terms of sensitivity, specificity, and prevalence but may also include other more subjective considerations. In any event, determining all possible cut-offs in JMP and then calculating each sensitivity and specificity is time consuming (and boring). A task ripe for automation.

The core calculations are performed with the JMP calculator. A template was created that takes the two columns, X and Y, and determines all of the necessary quantities. Presently, the calculations assume no missing values. The formulas go through all of the possible cut-off values to determine all of the possible statistics. Thus, it is convenient to sort the data table on the X column so that, as the cut-off value increments, the specificity increases and the sensitivity decreases.

### X change

The statistics are thus only calculated when the X value changes. These rows are flagged using the formula below.

\[
\begin{cases} 
0, & \text{if } X=X_{i+1} \\
1, & \text{otherwise}
\end{cases}
\]

For convenience in identifying these rows, a row-state column is also calculated so that the rows with non-missing statistics are selected.

\[
\begin{cases} 
\text{match } X \text{ change:} \\
\left( \sum_{j=1}^{i} (\text{not } Y_j) \right), & \text{when } 1 \\
\text{otherwise}
\end{cases}
\]

\[
n-\sum_{j=1}^{n} Y_j
\]

**Specificity**

Specificity is:

**Number of true negative decisions**

**Number of actually negative cases**

In the JMP formula, the false positive fraction (1 - Specificity) is calculated. The formula for sensitivity is shown below. The denominator is the number of rows minus the number of actually positive cases; thus, it is the number of negative cases. The numerator is the number of negative cases using this cut-off or above.

**Y(Cases) By X (Scale)**

![Graph](image)
Sensitivity

Sensitivity is:

Number of true positive decisions
Number of actually positive cases

In the JMP formula for the true positive fraction is shown below. The denominator is the sum of all the 0/1 Y variables; This is the number of actually positive cases, the prevalence. The denominator does not change for the different cut-offs. The numerator is the number of positive cases below this row’s cut-off.

\[
\text{match } X \text{ change:} \begin{cases} 
  \sum_{j=1}^{n} Y_j, & \text{when } 1 \\
  0, & \text{otherwise}
\end{cases}
\]

Accuracy

In Metz, accuracy is defined as:

\[\text{Sensitivity} \times \text{Prevalence} + \text{Specificity} \times (1-\text{Prevalence}).\]

This is accomplished with the JMP formula below.

\[TPF(Sen) \cdot Y + Spec \cdot (1-Y)\]

Optimal Value

This calculation looks for the maximum accuracy, the 100%tile. The row with this value have the value 1.

\[
\begin{cases}
  1, & \text{if } \text{Accuracy} = \text{quantile}_1 \text{ Accuracy} \\
  0, & \text{otherwise}
\end{cases}
\]

For convenience, the row with the optimal value is labeled with the value of the optimal X cut-off value.

\[
\begin{cases}
  \text{if Optimal Value} \\
  \text{otherwise}
\end{cases}
\]

Cut-off

The continuous X column is dichotomized using a cut-off value. The formula must use a different logic depending upon whether high X values are indications of 1’s or 0’s in Y. Any reasonable cut-off score could be used. The value in entered in the second assignment in the JMP formula.

\[
slope = X_1 - X_n \\
\text{cut off} = \begin{cases} 
  \cdot, & \text{if } X=\cdot \\
  0, & \text{if slope} < 0 \text{ and } X \leq \text{cut off} \\
  0, & \text{if slope} > 0 \text{ and } X \geq \text{cut off} \\
  1, & \text{otherwise}
\end{cases}
\]

The Steps Used

The calculation of a ROC curve and related statistics was broken into three phases, each with its own script. The phases each have a button.

As in the case of the JMP toolbar, the interface was built using FaceSpan. The Sort Table button’s script takes the first two columns of a the first open JMP table and sorts it into the appropriate order. The script is shown below.

Script for Sort Table button:

```
--hilited theObj
-- tell application "JMP"
-- This assumes that the data table you wish to
-- analyze is frontmost in JMP,
-- that no other windows are open,
-- that the (numeric continuous) X variable is the
-- first column
-- that the (numeric 0/1 nominal) Y variable is the
-- second column
activate
set ActiveTable to name of window 1
set Xvar to name of column 1 of table ActiveTable
set Yvar to name of column 2 of table ActiveTable

-- Question: are these right?
display dialog ("Table:" & ActiveTable &", X:" & Xvar & " Y:" & Yvar) buttons{"OK","Cancel"} default button 1

-- Do a logistic regression
analyze fit from table ActiveTable for Yvar by Xvar

set pValue to row 2 of column "Prob>ChiSq" of reportTable 1 of report "Parameter Estimates"

-- Do a logistic regression
analyze fit from table ActiveTable for Yvar by Xvar

set pValue to row 2 of column "Prob>ChiSq" of reportTable 1 of report "Parameter Estimates"

display dialog "p-value" default answer pValue buttons{"OK","Cancel"} default button 1
```

1190
- determine the slope of the curve

set Slope to row 2 of column "Estimate" of reportTable 1 of report "Parameter Estimates"
set SortOrder to "descending sort order"
if Slope < 0 then set SortOrder to "ascending sort order"
display dialog SortOrder buttons ("OK")
close window 2

- now sort the existing data table in the right order for the next step
if Slope < 0 then
  sort table ActiveTable by {Xvar, Yvar} with replace
else
  sort table ActiveTable by {Xvar, descending, Yvar} with replace
end if
end tell
set enabled of push button "Copy/Paste & ROC" to true
activate current application
end if
end tell

The core work is done in the following script. It copies the values of the X and Y variables in the first two columns, opens the "ROC template" stationary pad, changes the column names to match the original data, and pastes the values into the first two columns of the template. The template may be renamed but is given the default name of "Sen & Spec". The paste causes all of the formulas in the data table to calculate. The logistic regression results appear (shown earlier), and then the ROC curve is drawn.

The points with the highest accuracy are labeled with x's.

Then, a subset table is formed which consists of all of the sensitivities and specificities and related terms. This data table is shown at the end of this paper.

Two plots then appear to illustrate how accuracy and sensitivity and specificity vary with the X value cut-offs.
**CONCLUSION**

AppleScript and its JMP implementation are emerging technologies that will allow applications to work together, allow the extension of existing applications, and facilitate the automation of repetitive tasks.

My experience is that since the software tools are not mature, the user will need to keep abreast of developments in this area. The Apple Script Editor is minimally sufficient to begin writing scripts. There are just now emerging other more complete environments for the writing and debugging of scripts. FaceSpan is clearly extremely useful for the building and scripting of a GUI. There are other tools, some free, that can be used to fill in gaps.

In order for the tools to mature, users must get involved with the developers. The JMP developers have shown a willingness to extend their AppleScript implementation and would appreciate constructive guidance as to how this should be done. What features need to be implemented? How would it be best to coordinate the cooperation of applications? How should these considerations be integrated with the same kinds of languages in the Windows environment. It is my sense that, as with any emerging technology, the more involved user can have a large impact on future developments.

<table>
<thead>
<tr>
<th>Cases By Cutoff</th>
<th>Cutoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Row %</td>
</tr>
<tr>
<td>0</td>
<td>96.30</td>
</tr>
<tr>
<td>1</td>
<td>8.57</td>
</tr>
</tbody>
</table>

In the last phase, the user must modify the formula for 'Cutoff' to reflect the chosen cut-off value. Then the following script results in a cross tabulation which can be used to determine the final sensitivities and specificities.
REFERENCES


Software Designs Unlimited Inc. (199x) *FaceSpan*, 1829 E.Franklin St, Suite 1020, Chapel Hill NC 27514-5861, sdu@applelink.apple.com.

ACKNOWLEDGEMENTS

JMP software is a registered trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

AI Best may be reached through the internet at: Best@Ruby.VCU.Edu or by mail at: Departments of Biostatistics and Psychiatry Box 980032, Medical College of Virginia, Virginia Commonwealth University, Richmond VA 23298-0032. 20 Feb 95

\[ \begin{array}{cccccccccc}
<table>
<thead>
<tr>
<th>C</th>
<th>L</th>
<th>N</th>
<th>C</th>
<th>C</th>
<th>N</th>
<th>C</th>
<th>Spec</th>
<th>Optimal Va...</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>0.130</td>
<td>0.971</td>
<td>0.91</td>
<td>0</td>
<td>0.870</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>0.074</td>
<td>0.943</td>
<td>0.93</td>
<td>0</td>
<td>0.926</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0.037</td>
<td>0.914</td>
<td>0.94</td>
<td>0</td>
<td>0.963</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1</td>
<td>0.019</td>
<td>0.886</td>
<td>0.94</td>
<td>1</td>
<td>0.961</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1</td>
<td>0.000</td>
<td>0.800</td>
<td>0.92</td>
<td>1</td>
<td>1.000</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>1</td>
<td>0.000</td>
<td>0.743</td>
<td>0.90</td>
<td>1</td>
<td>1.000</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>1</td>
<td>0.000</td>
<td>0.714</td>
<td>0.89</td>
<td>1</td>
<td>1.000</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>1</td>
<td>0.000</td>
<td>0.657</td>
<td>0.87</td>
<td>1</td>
<td>1.000</td>
<td>0</td>
</tr>
</tbody>
</table>
\]