JMP Discovery Summit Developer Tutorial

Building Dashboards and Applications

Dan Schikore
JMP Principal Systems Developer
SAS
dan.schikore@jmp.com

Notes

Course Agenda

Building Dashboards and Applications

Introduction

This tutorial will take you through the process of combining multiple reports using JSL or the JMP Dashboard Builder and Application Builder, so that the process can be repeated on the same or different data tables. You will learn how to enhance your application using data filters and parameterization, as well as manage your end-to-end workflow with import from sources such as the SQL Query Builder and output to interactive HTML. You will also learn about the new dashboard features in JMP 13, with summary report views and drag-and-drop arrangement of live reports.

The accompanying file AppBuilderTutorial.jmpapp will help to guide you through the exercises within JMP. Open this file with JMP 13 and use the arrows at the top of the page to navigate through the major steps. This document will provide additional detail on the steps to be performed.

Combine Windows

JMP contains a large variety of statistical and graphing platforms for particular tasks. It is not uncommon for multiple platforms to be used in a coordinated way to answer a set of questions. Once you have created several reports that work well together, it may be helpful to be able to reproduce the same set of reports on the same table or on a new table. Combining the reports into a single window can reinforce the relationship between the reports.

In the accompanying live JMP tutorial, click the button *Launch Reports* to create two graphs.

To combine multiple reports into a window, start by positioning the reports on the screen in roughly the layout that you would like to see in the combined report. To combine the windows:

On Microsoft Windows (see Figure 1)
1. Select the reports using the toggle button at the lower-right corner
2. From the drop-down menu, select *Combine selected windows*

On either Mac (JMP 11-13) or Windows (in JMP 13)
1. Select *Combine Windows…* from the main Windows menu
2. Select the windows and click Ok

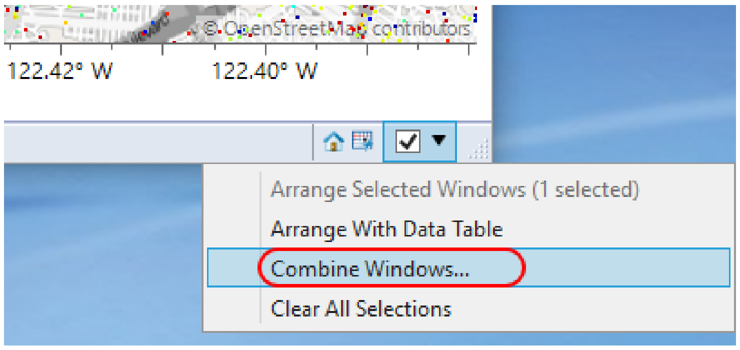
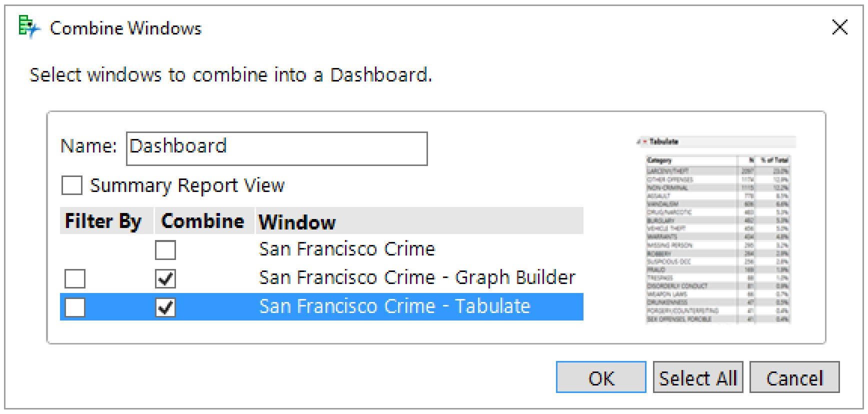


Figure 1: Combining multiple windows

With either approach, a new dialog will appear prompting you to select the windows to be combined, including both reports and data tables. Thumbnail views of the windows will be shown to help identify the windows of interest.



Combine Windows

Select windows to combine into a Dashboard.

Name: Dashboard

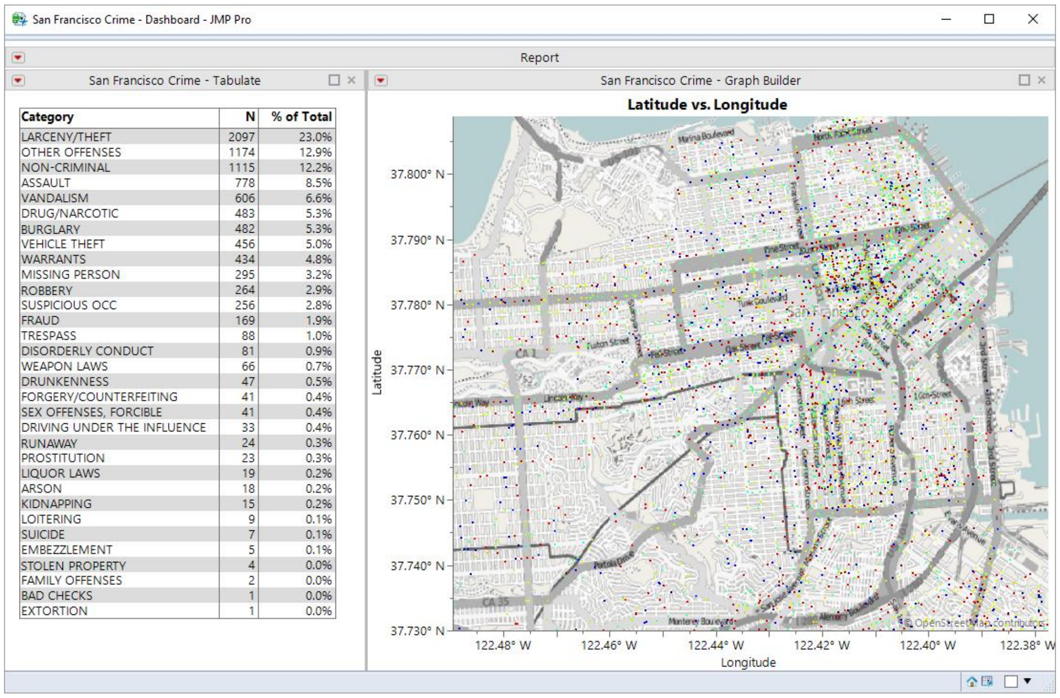
☐ Summary Report View

Filter By	Combine	Window
	☐	San Francisco Crime
☐	✓	San Francisco Crime - Graph Builder
☐	✓	San Francisco Crime - Tabulate

Figure 2: Combine Windows Dialog

After selecting windows and clicking ‘OK’, a new window will be created as shown in Figure 3. Note that the windows will be arranged depending on how they appeared on the screen. They could be reversed, or even arranged vertically, depending on how you positioned the windows.

Note that the reports are still fully interactive – performing a selection in one graph will highlight the linked rows in the data table. Each graph also has a red-triangle menu to continue working with the platforms.



San Francisco Crime - Dashboard - JMP Pro

Report

San Francisco Crime - Tabulate

Category	N	% of Total
LARCENY/THEFT	2097	23.0%
OTHER OFFENSES	1174	12.9%
NON-CRIMINAL	1115	12.2%
ASSAULT	778	8.5%
VANDALISM	606	6.6%
DRUG/NARCOTIC	483	5.3%
BURGLARY	482	5.3%
VEHICLE THEFT	456	5.0%
WARRANTS	434	4.8%
MISSING PERSON	295	3.2%
ROBBERY	264	2.9%
SUSPICIOUS OCC	256	2.8%
FRAUD	169	1.9%
TRESPASS	88	1.0%
DISORDERLY CONDUCT	81	0.9%
WEAPON LAWS	66	0.7%
DRUNKENNESS	47	0.5%
FORGERY/COUNTERFEITING	41	0.4%
SEX OFFENSES, FORCIBLE	41	0.4%
DRIVING UNDER THE INFLUENCE	33	0.4%
RUNAWAY	24	0.3%
PROSTITUTION	23	0.3%
LIQUOR LAWS	19	0.2%
ARSON	18	0.2%
KIDNAPPING	15	0.2%
LOITERING	9	0.1%
SUICIDE	7	0.1%
EMBEZZLEMENT	5	0.1%
STOLEN PROPERTY	4	0.0%
FAMILY OFFENSES	2	0.0%
BAD CHECKS	1	0.0%
EXTORTION	1	0.0%

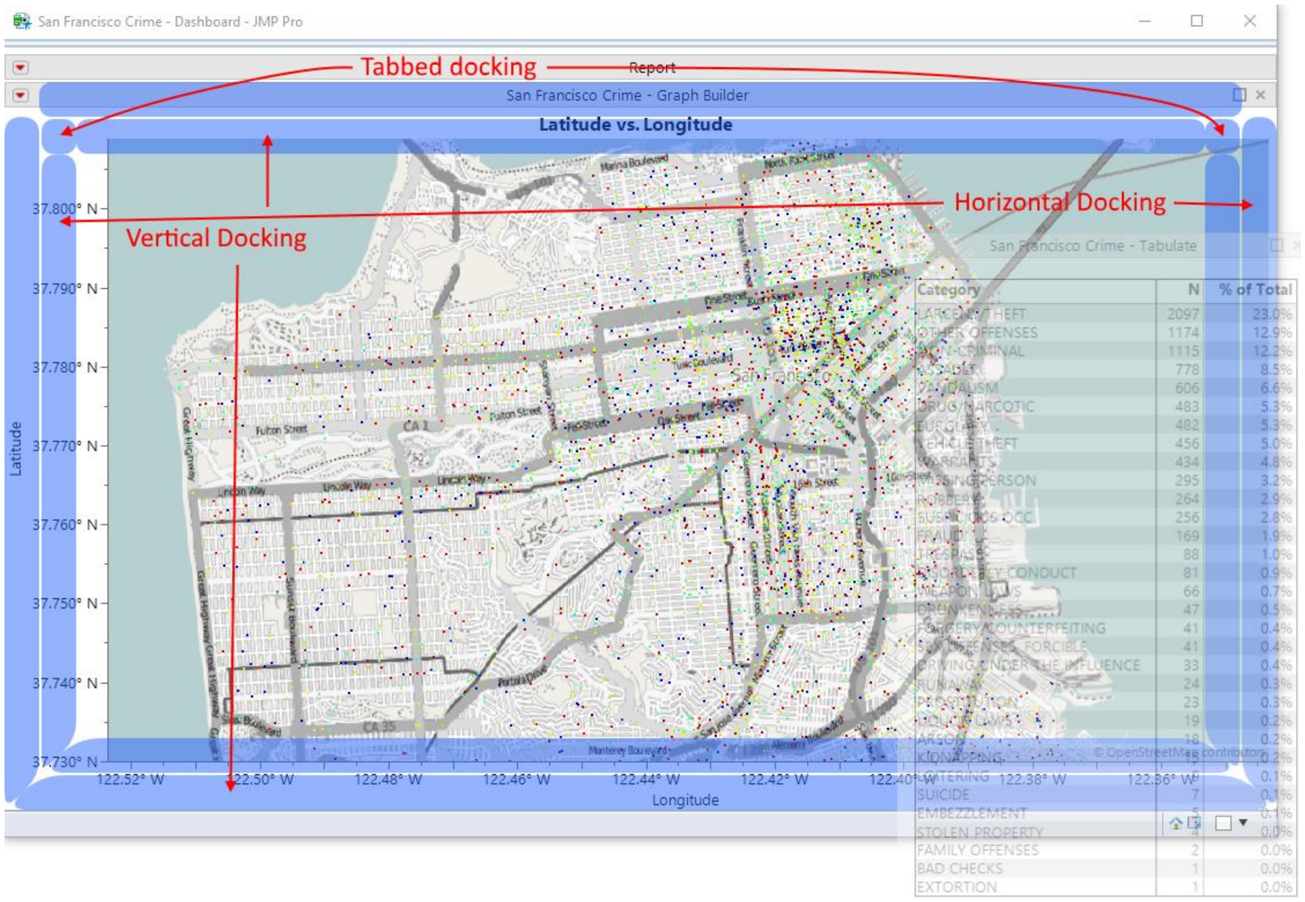
San Francisco Crime - Graph Builder

Figure 3: Combined Windows

Dashboard docking behavior

When using Combine Windows in JMP 13, you will notice that the reports in the Dashboard look different than the stand-alone reports. Dashboards use a Tab Page Box as a container for each report, and the tab page replaces the top-level Outline Box for the report.

Tab pages form the basis of a docking framework for reorganizing reports dynamically using drag-and-drop. By clicking and dragging from the tab page title a report can be repositioned in the window. There are drop zones for horizontal docking on the left and right, vertical docking on the top and bottom, and in the upper corners there are two drop zones that can be used to arrange multiple reports in a Tab Box. Figure 4 illustrates the docking action.



Category	N	% of Total
LARCENY/THEFT	2097	23.0%
OTHER OFFENSES	1174	12.9%
NON-CRIMINAL	1115	12.2%
ASSAULT	778	8.5%
VANDALISM	606	6.6%
DRUG/NARCOTIC	483	5.3%
BURGLARY	482	5.3%
VEHICLE THEFT	456	5.0%
WARRANTS	434	4.8%
MISSING PERSON	295	3.2%
ROBBERY	264	2.9%
SUSPICIOUS OCC	256	2.8%
FRAUD	169	1.9%
TRESPASS	88	1.0%
DISORDERLY CONDUCT	81	0.9%
WEAPON LAWS	66	0.7%
DRUNKENNESS	47	0.5%
FORGERY/COUNTERFEITING	41	0.4%
SEX OFFENSES, FORCIBLE	41	0.4%
DRIVING UNDER THE INFLUENCE	33	0.4%
RUNAWAY	24	0.3%
PROSTITUTION	23	0.3%
LIQUOR LAWS	19	0.2%
ARSON	18	0.2%
KIDNAPPING	18	0.2%
LOITERING	9	0.1%
SUICIDE	7	0.1%
EMBEZZLEMENT	5	0.1%
STOLEN PROPERTY	4	0.0%
FAMILY OFFENSES	2	0.0%
BAD CHECKS	1	0.0%
EXTORTION	1	0.0%

Figure 4: Dragging the Tabulate report activates drop zones for rearranging the dashboard

Saving a dashboard or application

When you have a combination of graphs that you would like to save to use again, there are several options. Note that in addition to the platform red-triangle menus, there is an additional red-triangle menu for the combined report. Much like an individual platform, this menu provides several options for saving the combined window for future use.

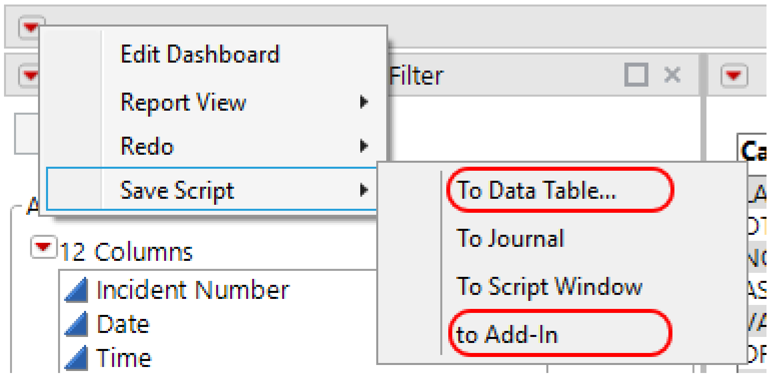


Figure 5: Saving an application

- Save Script to Data Table

  Saving the script to the Data Table is equivalent to the same command on an individual platform. The Application will appear in the list of scripts for the data table, and will reproduce the combined window for that specific Data Table.

- Save Script to Add-In

  Add-Ins are a mechanism for adding menus to the JMP interface and associating scripts with these menus. This option is most useful if you want to produce the same combined report for a different data table with the same columns. Add-Ins can also be shared with others who have a need to perform the same analysis. When an application is run from the Add-In menu, the reports will be based on the current data table.

Dashboard Builder

To this point, we have been using the Dashboard Builder infrastructure, but not Dashboard Builder itself. Combining windows is a quick way to produce a report. In Dashboard Builder you can produce the same report from scratch, or make additional changes to a report created with *Combine Windows*.

To modify a dashboard resulting from *Combine Windows*, select the *Edit Dashboard* option from the red-triangle menu.

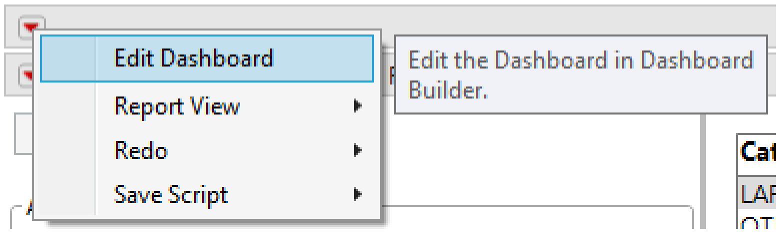


Figure 6: Editing a combined windows dashboard

To start from scratch, use File > New > Dashboard to open the Dashboard Builder. An opening screen will allow you to choose a template from a set of commonly used layouts, or you can start with a blank canvas.

Dashboard Builder is a drag-and-drop environment for building dashboards. The builder interface, shown in Figure 7, has two main areas:

1. Source Panel

   The source panel contains the display objects that can appear in a dashboard window. This includes reports, data tables, data filters, and select display boxes for pictures and text.

2. Workspace

   The workspace on the right is where the window layout occurs.

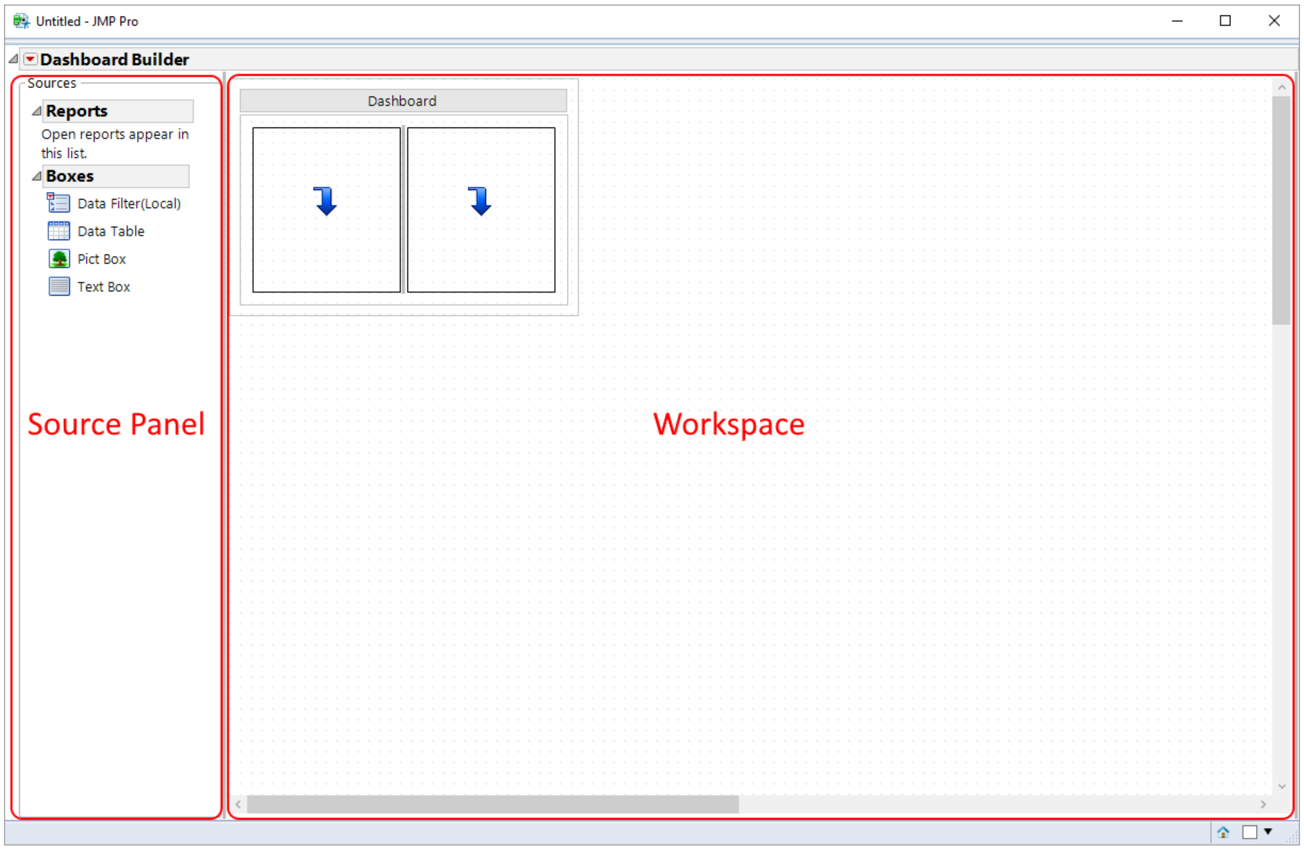


Figure 7: Dashboard Builder layout

Objects in the source list can be added to the workspace in several ways:

1. Drag the item and drop in an existing placeholder or other drop zone to create horizontal, vertical, or tab layouts.
2. Single-click to select the item and then single-click in a placeholder.
3. Double-click the item to populate the next placeholder, or append the report if no more placeholders exist.

Reports, Data Tables, and Filters are shown as thumbnails in Dashboard Builder, to make it easier to manipulate the layout. To see what your Dashboard will look like when it is run, select *Preview Mode* from the red-triangle menu.

Saving Application Files

Saving a combined window to the Data Table and Add-In are examples of deploying the application for use. In some cases your application may have a life of its own, going through multiple revisions to add new features, rearrange the report, or make other changes. There are two document types in JMP to support applications:

1. .jmpappsource

   A file with .jmpappsource extension is the native format produced by File > Save in the Application Builder. Opening a file with this extension will return you to the Dashboard Builder interface where you can continue making changes to your application. If you are producing dashboards that are consumed by others, this may be a file that you maintain but do not distribute.

2. .jmpapp

   A file with .jmpapp extensions is produced by File > Save As on Windows or File > Export on Mac. The content of this file is identical to the .jmpappsource, but in this case the application will run when the file is opened. This is an alternative to distributing an Add-In if you prefer not to add the application to the JMP menus.

Shared Local Data Filter

Most platforms within JMP support a Local Data Filter feature for selectively filtering one report without affecting other reports. Within Dashboard Builder, you can add a Local Data Filter that will be shared between multiple reports in a window. Begin by positioning your reports within the Dashboard Builder, and then drag the Local Data Filter object from the source panel and place it anywhere in the workspace, as shown in Figure 8.

Shared local data filters in a Dashboard are global to the window – all reports based on the same table will respond to the filter.

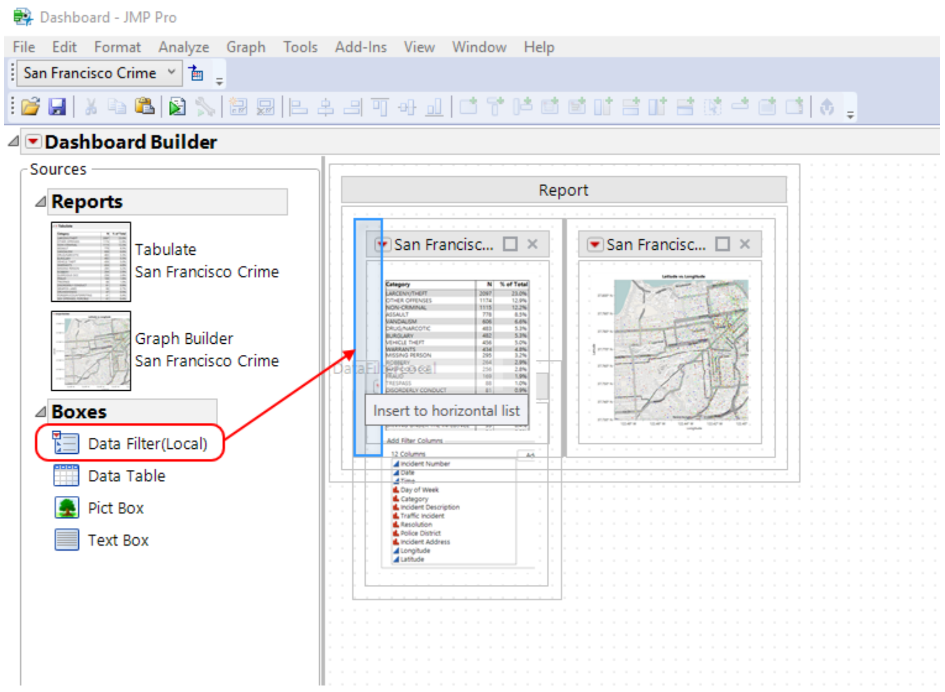


Figure 8: Adding a Local Data Filter to a Dashboard

Selection Filters

The built-in Data Filter interface provides multiple ways of selecting the data of interest for categorical and continuous data. When exploring data in JMP, we often use graphs to inform decisions on what to look at next. In Dashboard Builder, a JMP graph can be used as the data filter interface, which is then referred to as a *Selection Filter*.

There are two ways to use a report as a selection filter in a dashboard:

1. Check the 'Filter by' box in the Combine Windows dialog

2. Place your reports within Dashboard Builder, and right-click on a report to add or remove selection filter behaviors, as shown in Figure 9.

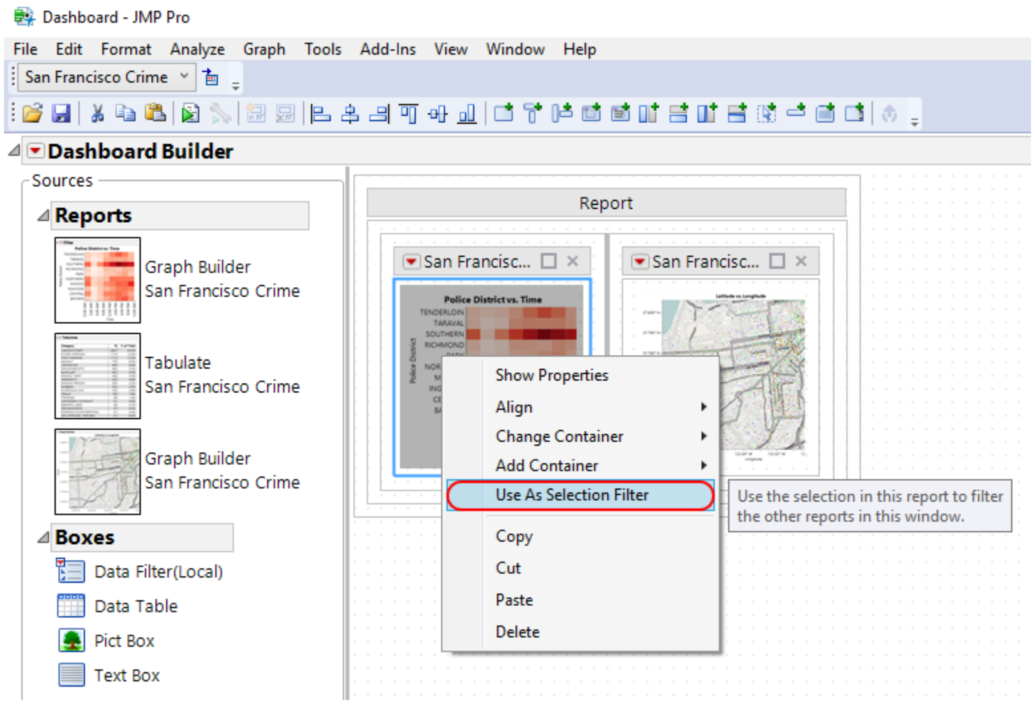


Figure 9: Specifying one of the graphs as a selection filter

When you run this Application, selections performed in the first report will filter the results in the other reports. Any JMP report that supports selection can be used as a selection filter, and the selection modes (Ctrl-selection, drag-selection, etc) are the same as in the stand-alone platform.

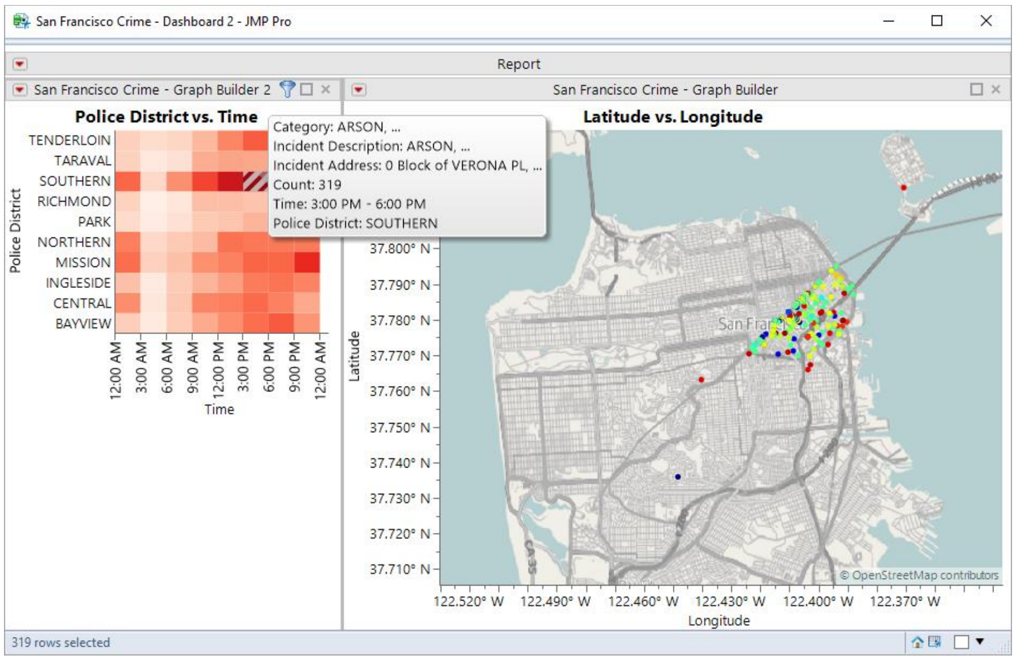


Figure 10: The Selection Filter in a running Application

Summary View

With the reconfigurable drag-and-drop nature of dashboards, stretchable graphs are some of the best content to use when designing a dashboard. Some JMP platforms are stretchable by default, such as Graph Builder, Bubble Plot, and Treemap. Many platforms do not stretch by default, due to the large amount of information presented in tables and other forms. A new Summary View option is available to make it easier to incorporate the graphs from such platforms into a dashboard. The option for Summary View can be toggled on from the Combine Windows dialog. There are two main effects on the reports:

1. Graphs in the report are made stretchable.
2. Most other content is hidden.

Some platforms customize this behavior – supplementary graphs may be hidden and important results in tables may be shown. The default behavior can also be customized through JSL scripting, as we will see later. Figure 11 illustrates a dashboard in Summary View. By making the reports stretchable, the window (or individual tabs) can be maximized to fill the screen for presentation purposes.

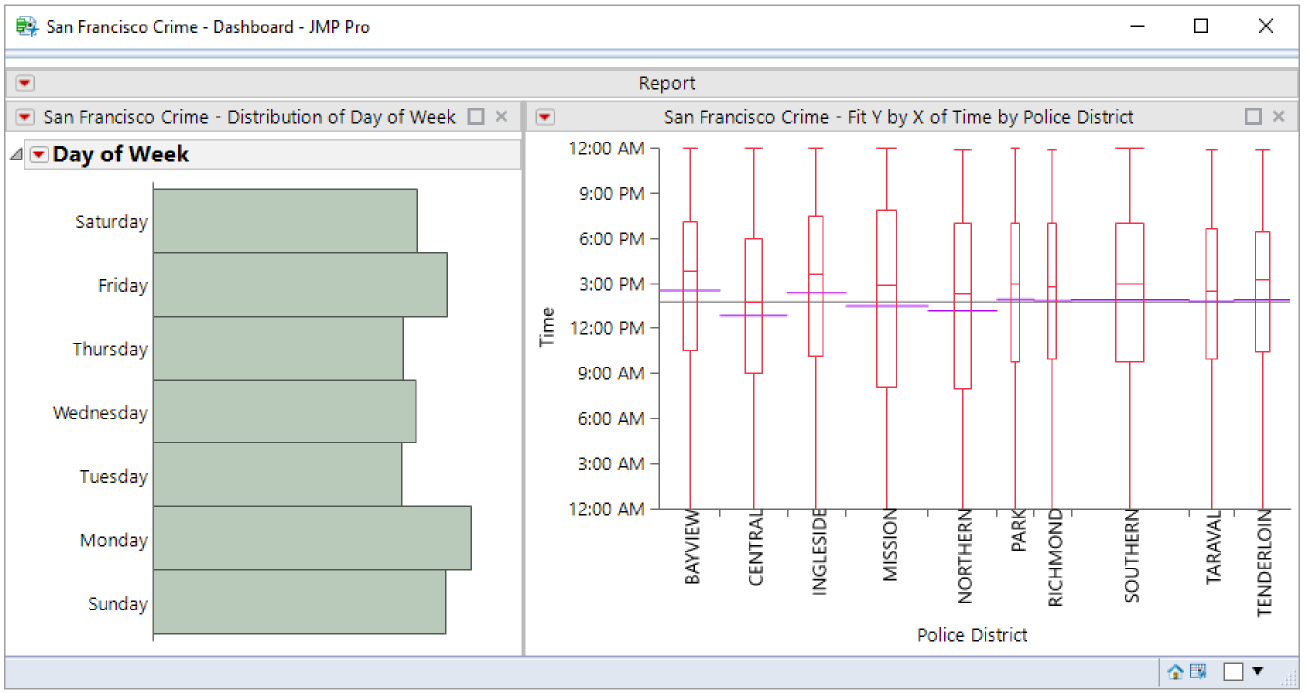


Figure 11: A dashboard in Summary View

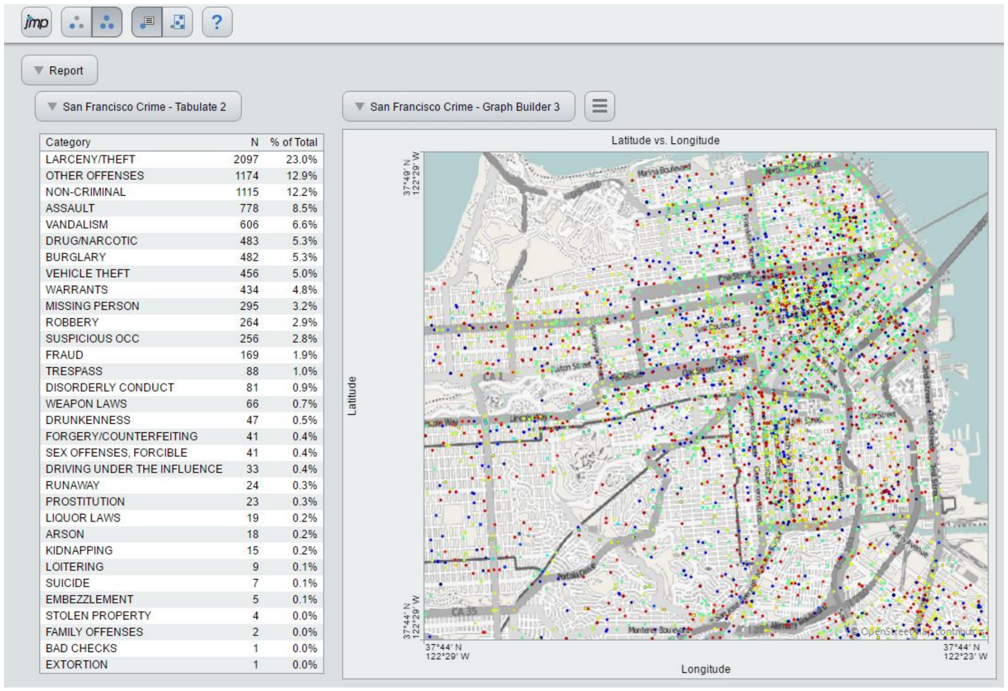
HTML5 output

JMP 11 introduced the ability to save a report to Interactive HTML, allowing interaction with a JMP report in a browser. With JMP 13 most Graph Builder elements are supported in Interactive HTML, which also makes HTML a great way to communicate dashboard results.

To output a report, choose File > Save As (Windows) and choose *Interactive HTML* as the file type. On Mac, the menu is File > Export. Note that local data filters and selection filters are not supported in HTML5 –

filtering requires the ability to re-compute the contents of the reports, while the current HTML output is limited to displaying results that have already been computed in desktop JMP.

When you save a report to Interactive HTML, the data will be stored directly in the HTML file along with a description of the display components to produce the graphs. You can publish this HTML on a web server or send the file to colleagues and collaborators. An example HTML report is shown in Figure 12.

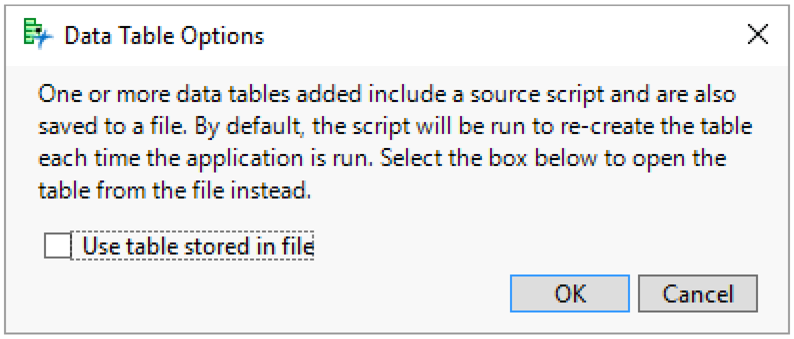

Report
San Francisco Crime - Tabulate 2
San Francisco Crime - Graph Builder 3
Latitude vs. Longitude
Latitude
Longitude

Category	N	% of Total
LARCENY/THEFT	2097	23.0%
OTHER OFFENSES	1174	12.9%
NON-CRIMINAL	1115	12.2%
ASSAULT	778	8.5%
VANDALISM	606	6.6%
DRUG/NARCOTIC	483	5.3%
BURGLARY	482	5.3%
VEHICLE THEFT	456	5.0%
WARRANTS	434	4.8%
MISSING PERSON	295	3.2%
ROBBERY	264	2.9%
SUSPICIOUS OCC	256	2.8%
FRAUD	169	1.9%
TRESPASS	88	1.0%
DISORDERLY CONDUCT	81	0.9%
WEAPON LAWS	66	0.7%
DRUNKENNESS	47	0.5%
FORGERY/COUNTERFEITING	41	0.4%
SEX OFFENSES, FORCIBLE	41	0.4%
DRIVING UNDER THE INFLUENCE	33	0.4%
RUNAWAY	24	0.3%
PROSTITUTION	23	0.3%
LIQUOR LAWS	19	0.2%
ARSON	18	0.2%
KIDNAPPING	15	0.2%
LOITERING	9	0.1%
SUICIDE	7	0.1%
EMBEZZLEMENT	5	0.1%
STOLEN PROPERTY	4	0.0%
FAMILY OFFENSES	2	0.0%
BAD CHECKS	1	0.0%
EXTORTION	1	0.0%

Figure 12: A JMP dashboard reproduced using Interactive HTML

Query Builder

In the examples we have used so far, the data comes from a JMP file. JMP 12 and JMP 13 have made it much easier to dynamically create tables from databases or other JMP tables, using the Query Builder. Tables produced by Query Builder have a Source script that will be used by Dashboard Builder to rerun the query. If the data table has also been stored to a file, you will be given the option to use the file on disk rather than rerunning the script.



One or more data tables added include a source script and are also saved to a file. By default, the script will be run to re-create the table each time the application is run. Select the box below to open the table from the file instead.

☐ Use table stored in file

Figure 13: Prompt for behavior of imported tables

Application Builder

Dashboard Builder is new in JMP 13, but it is based on the infrastructure of Application Builder, which has existed since JMP 10. Additional features that are available in Application Builder include:

1. Multiple windows
2. Support for more display boxes
3. Applications can include JSL, which can control the workflow of multiple windows or respond to input from display boxes
4. Parameterization allows you to change the columns being analyzed

To start creating an Application from scratch, use File > New > Application. You can also start with a dashboard created from Combine Windows or from Dashboard Builder, and toggle off *Dashboard Mode* in the red-triangle menu.

The Application Builder interface is very similar to Dashboard Builder. The source panel contains more boxes for building custom applications. The workspace shows multiple tabs – the first is for the scripts associated with the application, and there are tabs for each window. On the right are two new panels:

1. Object tree

   The object tree at the upper-right contains a hierarchical view of the application, including any data tables, modules (windows), and display boxes. The main purpose of the object tree is to provide an alternate way to select objects to operate on, rather than clicking in the workspace area.

2. Property Panel

   The property panel displays editable properties for the current selected objects. Use this area to both view the current settings and also to make changes.

As an example, select the Application object from the Object tree as shown in Figure 15. Edit the Name in the Property Panel to choose a name for the Application. This name will help you to distinguish between multiple applications. The name is used when saving the application to the Data Table, to an Add-In, and also to a file.

Unlike Dashboard Builder, Application Builder does not have a Preview Mode. To test your Application, choose Run Application from the red-triangle menu.

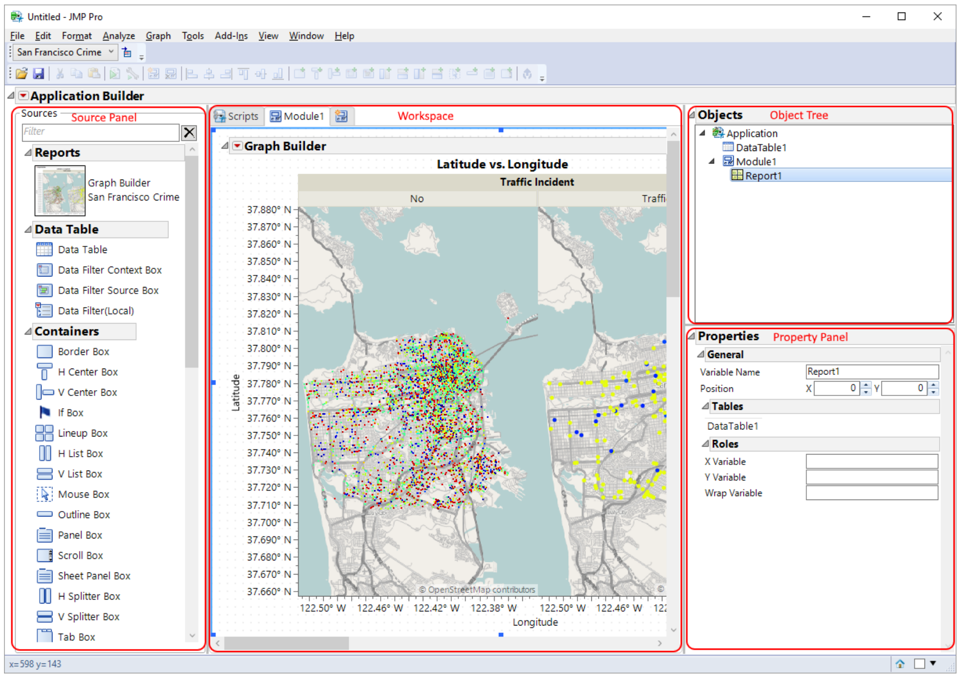
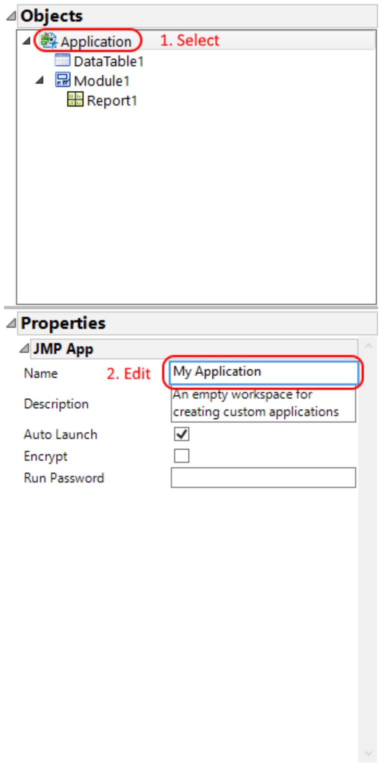


Figure 14: Application Builder layout



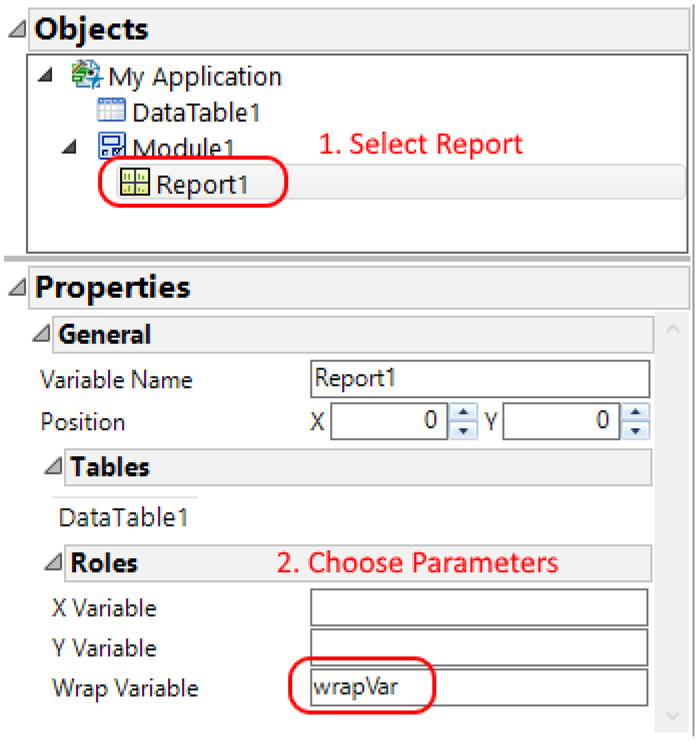
Name: My Application
Description: An empty workspace for creating custom applications
Auto Launch: ✓
Encrypt: ☐
Run Password:

Figure 15: Editing the name of the Application

Parameterizing Applications

Most JMP platforms begin by asking which columns you would like to analyze or graph. When you save a script to a table or script window, these choices are fixed unless you write some additional JSL. The same has been true with the dashboards and applications that we have created so far, but Parameterization allows you to create an application that has more flexibility.

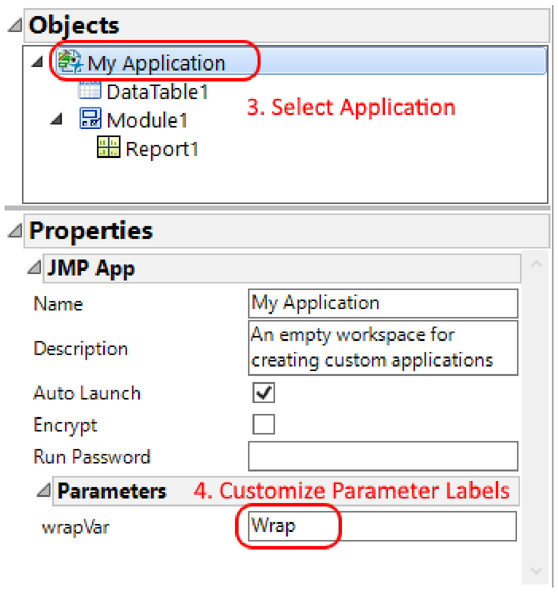
The first step in parameterization is to select which platforms, and which roles, you want to parameterize. As shown in Figure 16, when a Report is selected (either in the Workspace or in the Object Tree), the roles are displayed in the Property Panel. If the role value is left blank, then App Builder will launch the platform with the original set of columns. You can choose to set a parameter value using a *JSL variable name* to indicate that you would like to vary this parameter when the application is run. You can choose to parameterize some roles and not others – for this demonstration we will parametrize the Wrap role using the variable *wrapVar*.



Variable Name	Report1
Position	X 0 Y 0
DataTable1	
X Variable	
Y Variable	
Wrap Variable	wrapVar

Figure 16: Parameterizing roles for a platform

Setting the parameter variables is sufficient to create a parameterized application. Having chosen one or more roles to parameterize, JMP will prompt to select the columns to be used when the application is run. Optionally, you can customize the labels that will be displayed in the launch dialog, as shown in Figure 17. Customizing labels will give you an opportunity to give a look and feel that is specific to the particular task of your application, rather than using general terms such as *X*, *Y*, *Wrap*, and *Color*.



Name: My Application
Description: An empty workspace for creating custom applications
Auto Launch: ✓
Encrypt:
Run Password:
Parameters
wrapVar: Wrap

Figure 17: Customizing labels in a parameterized application

Multiple Windows

There are many reasons that you might want to have multiple windows in an application. You might have multiple report windows that open simultaneously or in sequence, or you might use input from one window to create other windows.

In the parameterized example, we saw that Application Builder provided a launch window to choose the column for the wrap role. We will extend this example to create a second application window to handle the launch of the report. Figure 18 shows the 'Add Module' button that is used to create a new tab that we will use for the launch window.

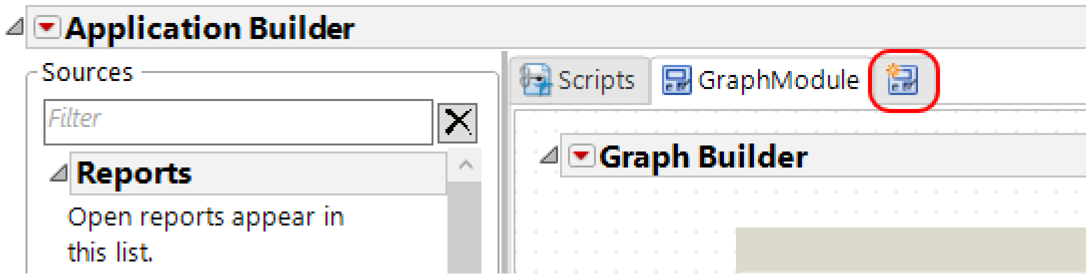


Figure 18: Adding a module for a second application window

Within the new module, we drag display boxes from the source panel to the workspace. In addition to containers to organize the display, the main boxes are a ColListBox to select a column, and a ButtonBox to create the graph, as shown in Figure 19.

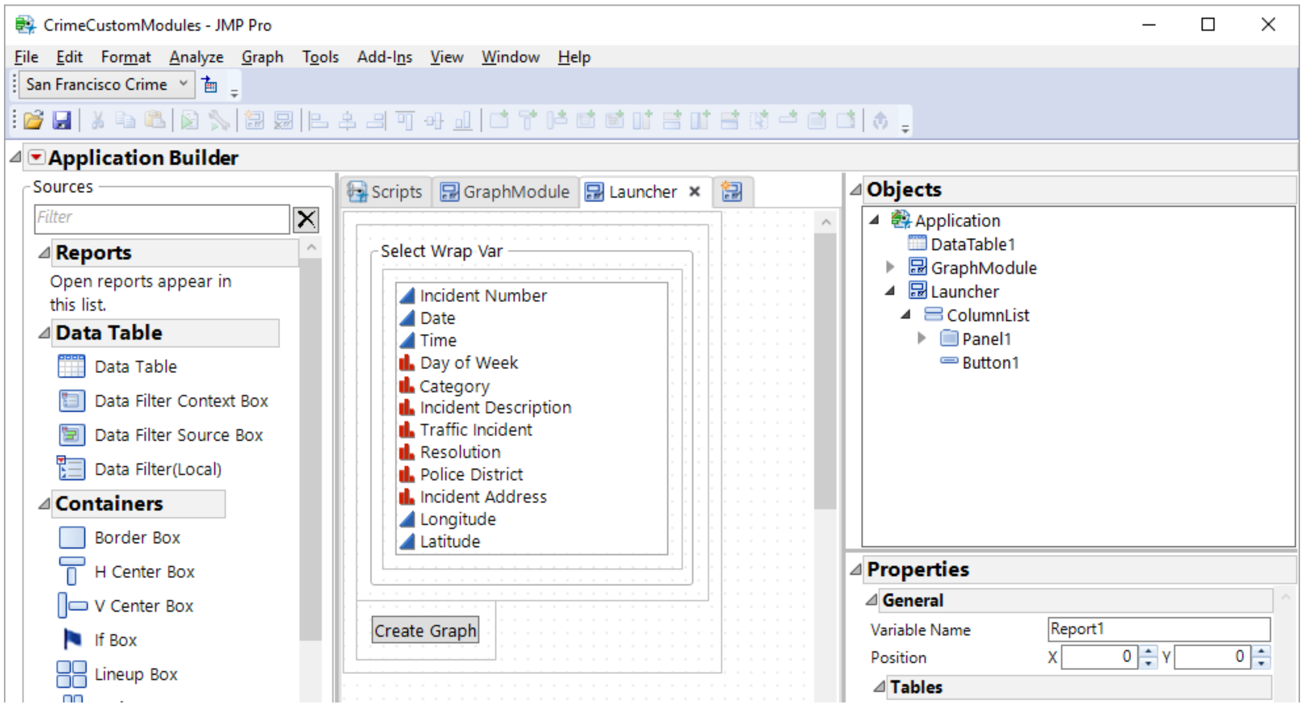


Figure 19: Custom boxes for launch of application

We need to change two default settings in order to get the launch behavior that we are looking for. First, because the GraphModule was parameterized, the default behavior at run-time will be for Application Builder to prompt for the value for *wrapVar*. To disable this prompt, select the Application object and toggle off the Auto Launch property.

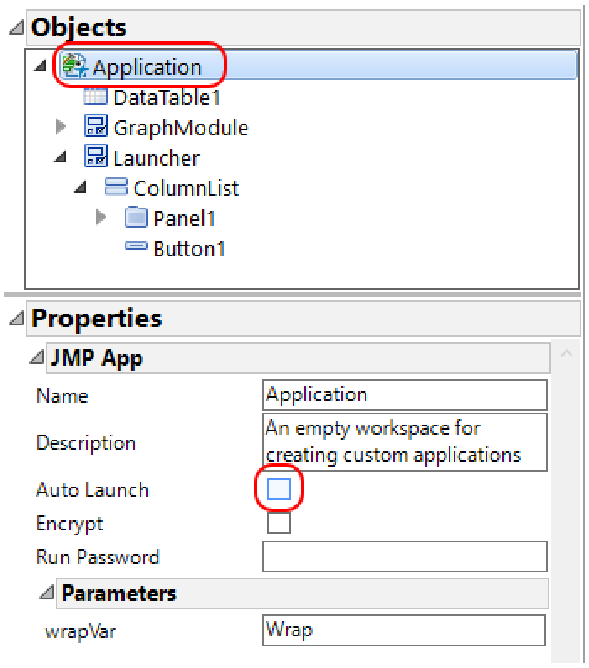
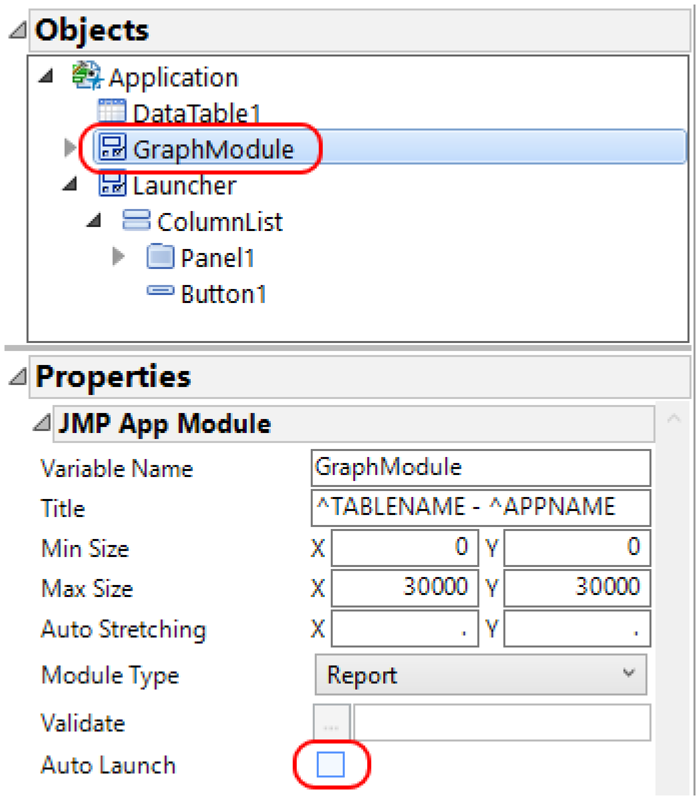


Figure 20: Toggle off Auto Launch to disable the built-in launch dialog

The second change is related to the two modules. Under default settings, one instance of each module is created at run-time. In this case, we want to create the Launch Module instance, but we do not want to create a GraphModule instance until the button is pressed. The Auto Launch property on the module determines whether this default instance is created when the application is run.


Objects
Application
DataTable1
GraphModule
Launcher
ColumnList
Panel1
Button1

Properties
JMP App Module

Variable Name	GraphModule
Title	^TABLENAME - ^APPNAME
Min Size	X 0 Y 0
Max Size	X 30000 Y 30000
Auto Stretching	X . Y .
Module Type	Report
Validate	
Auto Launch	☐

Figure 21: If Auto Launch is set for a module, one instance of the module will be created automatically at run-time

Custom scripts

With the auto launch settings complete, at run-time our application will display one instance of the Launcher module at run-time. The user will then select a column, and press the *Create Graph* button. We will need to add two short scripts to (a) handle the behavior when the button is pressed (b) initialize the graph module when it is created.

There are multiple scripts associated with an application. First is the Application script, which is the first thing to run. By default this script is empty, but this can be a good place to do global initialization within your application. The remaining scripts are the Module scripts, one per module defined in the application. This script is run once for each *instance* of a module. Module scripts contain some default JSL for processing input and creating the boxes associated with the window.

Many input boxes can have one or more scripts associated with them. An easy way to add a script is to right-click on the box in the workspace, and select one of the scripts in the drop-down menu, as shown in Figure 22. In the case of Button Box, the script that you define will be called when the button is pressed.

When you use this approach to add a script, a function will be added to the module script, and a property will be set on the Button Box to call this function when the button is pressed.

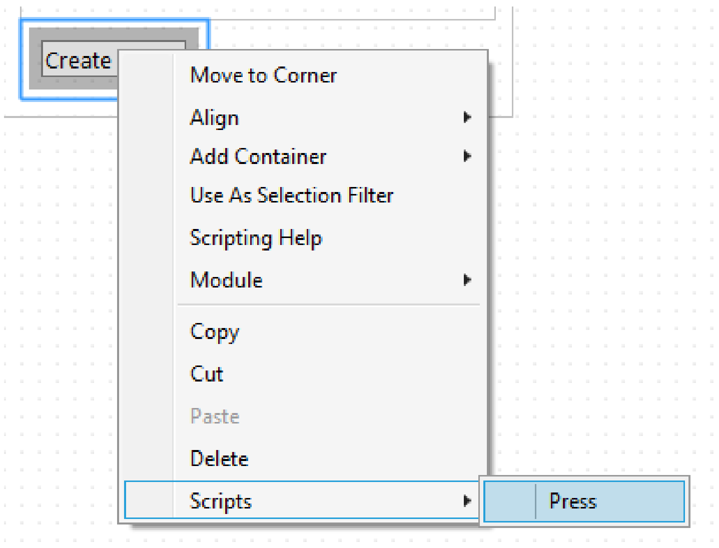


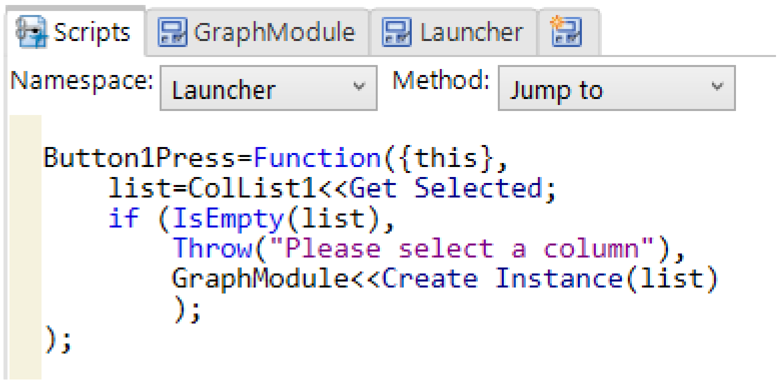
Figure 22: Adding a script for a Button Box

The module script will automatically show the newly created function. Below is the function that we have filled in to get the desired behavior:

1) First we get the selected column from ColList1. Every object that appears in the Object Tree has a JSL variable name associated with it, and we can use this variable name to get or set properties on the object.

2) For simple error checking, we check to see if any columns were selected. If not, then we pop up an error message.

3) If a column was selected, then we create an instance of the GraphModule and pass the selected column to it. GraphModule is an object of type JMPAppModule, which is a type that you will only see in a JMP Application.



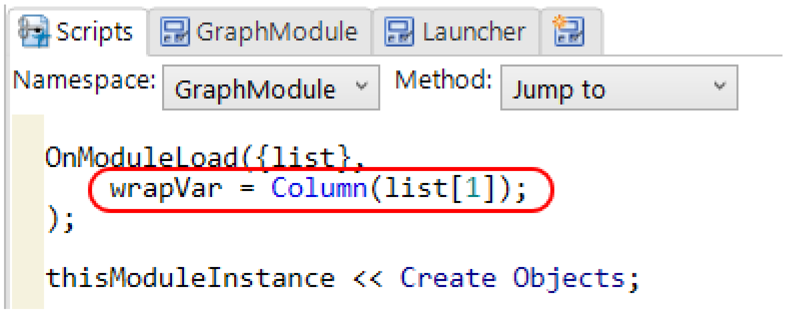
Namespace: Launcher Method: Jump to

```
Button1Press=Function({this},
    list=ColList1<<Get Selected;
    if (IsEmpty(list),
        Throw("Please select a column"),
        GraphModule<<Create Instance(list)
        );
);
```

Figure 23: Script to define the behavior when Create Graph button is pressed

This completes the behavior of the Launcher module. The final step is to define the behavior of the *GraphModule*. Recall that we disabled the Auto Launch function of the Application. The main purpose of the Auto Launch was to define the *wrapVar* parameter of our application. Because we are using a custom launcher, we need to define *wrapVar* ourselves, as shown in Figure 24.

When the launcher called <<CreateInstance, it passed as a parameter the selection list from the column dialog. The GraphModule script is receiving this parameter through a special function *OnModuleLoad*, and it uses this parameter to initialize *wrapVar*. Note that this initialization takes place before the call to *Create Objects*, which is when the boxes and reports will be created. A JSL error would occur if we called *Create Objects* before having initialized *wrapVar*.



```
OnModuleLoad({list},
    wrapVar = Column(list[1]);
);

thisModuleInstance << Create Objects;
```

Figure 24: The GraphModule script is executed for each module instance that is created

JSL

Dashboard Builder and Application Builder give you the ability to create certain classes of applications without using JSL, but all of the functionality is also available for JSL programmers to use. The new tab docking report infrastructure is based on three new features within existing display boxes:

1. Tab Page Box can be created without an associated Tab Box.

2. Tab Page Box has a new <<Moveable(1) option.

3. Tab Box and Splitter Box have new <<Dockable(1) options

The script below illustrates the usage of the new Tab Docking infrastructure from JSL. A Tab Page Box that does not have a Tab Box parent will draw its own title.

```
dt = Open( "$SAMPLE_DATA/San Francisco Crime.jmp" );
New Window( "Dashboard",
      H Splitter Box(
            Size(900,600),
            Tab Page Box("Tabulate",
                  dt << Run Script("Tabulate: Category, Summary, Percentage"),
                  <<Moveable( 1 )
            ),
            Tab Page Box("Graph Builder",
                  dt << Run Script( "Graph Builder Street Map" ),
                <<Moveable( 1 )
            ),
            <<Dockable( 1 )
      )
);
```

Shared Data Filters and Selection Filters can also be configured directly from JSL. The following script does not make use of the new dashboard features, so it will work in either JMP 12 or JMP 13. To create a selection filter there are two display boxes to add to your application:

1. Data Filter Context Box

   This is a container box that holds all of the graphs involved in the filter – both those doing the filtering and those being filtered.

2. Data Filter Source Box

   This is a container within the Data Filter Context Box that holds the reports that are acting as the selection filter. If you are using the built-in Local Data Filter, you don't need to add this box.

```
dt = Open( "$SAMPLE_DATA/San Francisco Crime.jmp" );
New Window( "Selection Filter",
   Data Filter Context Box(
      H List Box(
         Data Filter Source Box(
            dt << Graph Builder(
               Size( 313, 293 ),
               Show Control Panel( 0 ),
               Show Legend( 0 ),
               Fit to Window( "Off" ),
               Variables( X( :Time ), Y( :Police District ) ),
               Elements( Heatmap( X, Y, Legend( 4 ) ) ),
               Report View( "Summary" ),
               SendToReport(
                  Dispatch(
                     {},
                     "Graph Builder",
                     OutlineBox,
                     {Set Title( "Filter" )}
                  ),
                  Dispatch(
                     {},
                     "Police District",
                     ScaleBox,
                     {Label Row( Show Major Ticks( 0 ) )}
                  ),
                  Dispatch(
                     {},
                     "400",
                     ScaleBox,
                        {Legend Model(
                              4,
                              Properties(
                              0,
                              {
                                 gradient(
                                    {Color Theme( "White to Red" ), Width( 12 )}
                              )}
                           )
                        )}
                     )
                  )
               )
            ), /* end Data Filter Source Box */
            dt << Run Script( "Graph Builder Street Map" )
         )
      ) /* end Data Filter Context Box */
);
```

Conclusion

This tutorial has provided an introduction to the JMP Dashboard Builder and JMP Application Builder and demonstrated several features that help you to create reusable JMP Applications for specialized tasks. There are more features that we have not had time to cover, including:

1. Multiple data tables

   Application Builder supports multiple data tables and graphs/reports from multiple tables within a single application.

2. Complex filters

   In our examples of the Local Data Filter and the Selection Filter, we always had a single filter. It is also possible to use multiple filters configured hierarchically for both dashboards and applications. In an application you can also have different filters associated with different reports.

Whether you work with Dashboard Builder, Application Builder, or JSL, please join the JMP Community at http://jmp.com/community to see what others are doing and share your questions and experiences with the group. In addition to JMP Technical Support, the community is a good way learn from others and let others learn from your experience. This tutorial will be posted and maintained in the community, and there is a dedicated document called “Using Selection Filters in JMP 12” that goes into further detail on selection filters, including hierarchical examples.