

Expanding JMP with R and Python

(and SAS and CAS and Matlab)

Why use this connection?

- Use JMP's dynamic linkage, advanced visuals or powerful analytics, but also use special R or Python packages.
- Explore new methodology in an R package side-by-side with validated methods in JMP.
- Develop an add-in to JMP that calls R or Python so that future users can interact with the GUI add-in without needing to write the R or Python code themselves.
- Introduce specific open source packages in the classroom but within a point-and-click environment that wouldn't require coding by the student.

Agenda

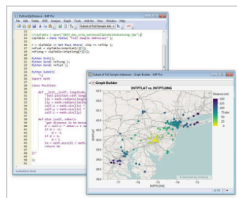
1. Generating Python (or C, JavaScript, SAS or SQL) to deploy in another environment.
 - Housing Prices example deployed in AWS
 - Valentine's Heart example deployed in AWS
2. Simple Python connection example
3. Simple R connection example
4. R add-in example: T-SNE and UMAP
5. R add-in Builder example
6. Making an Add-In
7. Help!

[Resource Center](#) > White Paper

Using JMP and JMP Pro With Python and R

JMP Synergies With Open Source

By Ruth Hummel, JMP

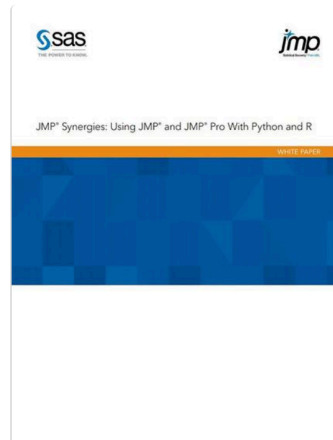


JMP is a standalone, full-featured data visualization and statistical analysis software from SAS for the Windows and Mac desktop. JMP has the interactivity and dynamic linkage that makes data exploration exciting, insightful and contains many advanced analytical options, fully satisfying the needs of data explorers and analysts. Still, there may be occasions where you'll want (or need) to use JMP in conjunction with open source tools, like Python or R.

In addition to providing you with the basics, this paper introduces the Python scoring-code generation, the Python in JMP scripting and the R in JMP scripting. You'll also discover sample code and advanced examples that will make using the connections and add-ins provided in JMP easy.

Whatever your motivation for connecting open source (or other) tools with JMP software's GUI, this guide will help you to get started using the Python and R connections in JMP.

Download Now



Generating Code: Housing Prices Example

<https://s3.amazonaws.com/stlouishousing.web/index.html>

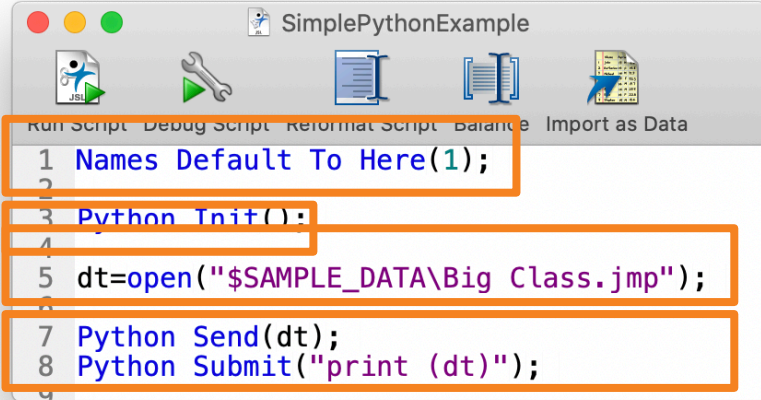
- Prediction model developed in JMP
- Generate Python code and deploy in AWS with a webpage wrapper

Generating Code: Valentine's Heart Example

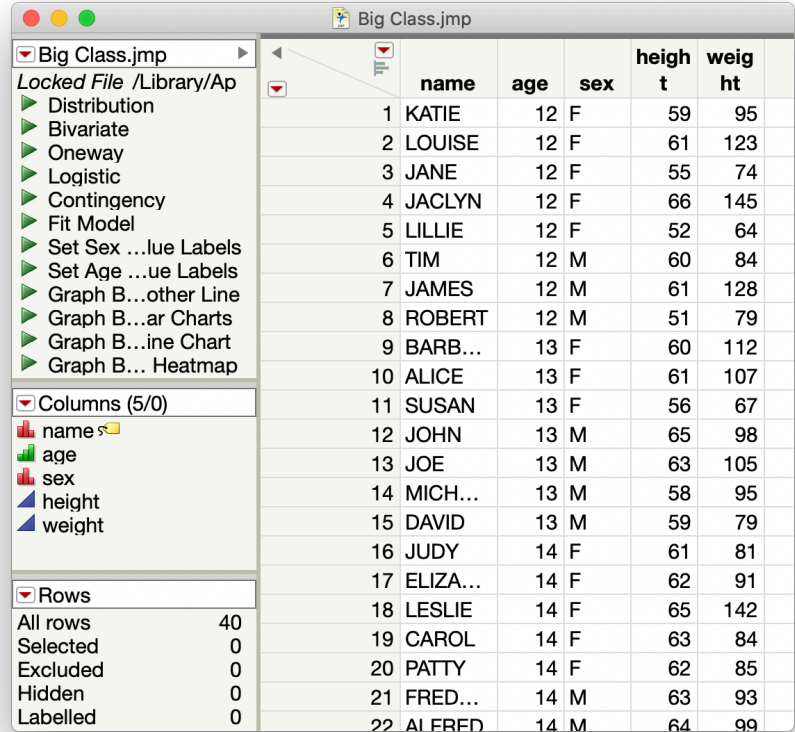
<https://s3.amazonaws.com/jmp.valentine/index.html>

- simple use case: creating a static (self-contained, no server calls) web app with an embedded model.
- neural net developed in JMP
- deployed it in JavaScript and “wrapped” a visualization and web page around it

Simple Python Example



```
1 Names Default To Here(1);
2
3 Python Init();
4
5 dt=open("$SAMPLE_DATA\Big Class.jmp");
6
7 Python Send(dt);
8 Python Submit("print (dt)");
9
```

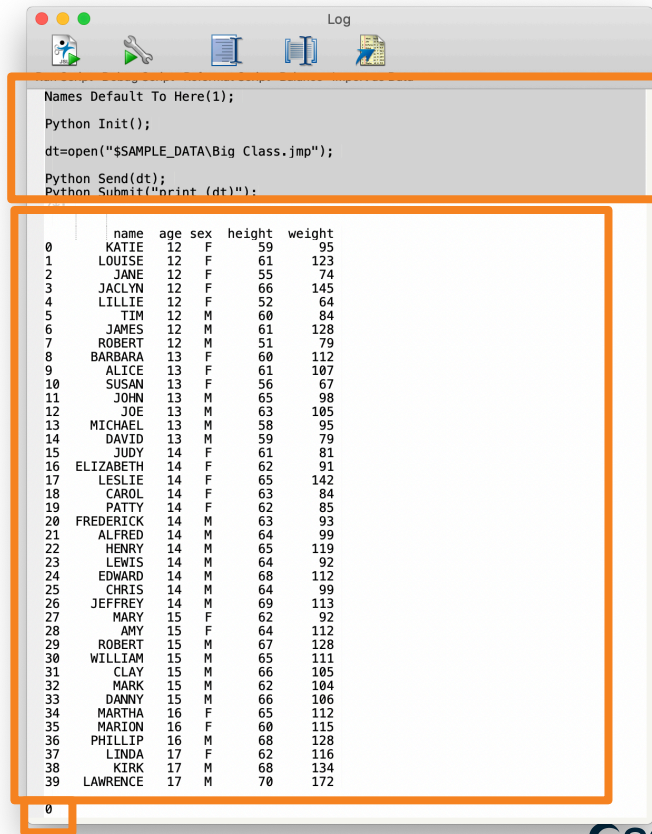
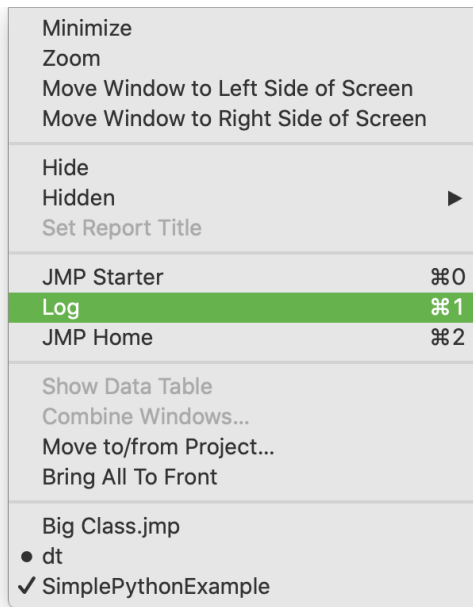
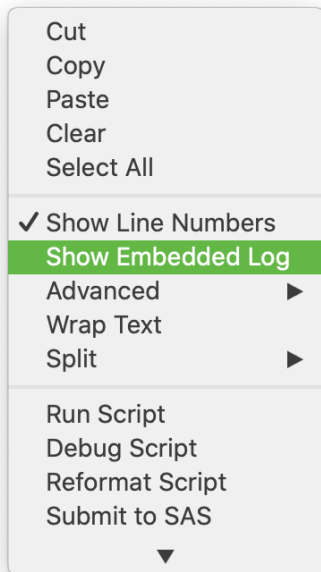


	name	age	sex	height	weight
1	KATIE	12	F	59	95
2	LOUISE	12	F	61	123
3	JANE	12	F	55	74
4	JACLYN	12	F	66	145
5	LILLIE	12	F	52	64
6	TIM	12	M	60	84
7	JAMES	12	M	61	128
8	ROBERT	12	M	51	79
9	BARB...	13	F	60	112
10	ALICE	13	F	61	107
11	SUSAN	13	F	56	67
12	JOHN	13	M	65	98
13	JOE	13	M	63	105
14	MICH...	13	M	58	95
15	DAVID	13	M	59	79
16	JUDY	14	F	61	81
17	ELIZA...	14	F	62	91
18	LESLIE	14	F	65	142
19	CAROL	14	F	63	84
20	PATTY	14	F	62	85
21	FRED...	14	M	63	93
22	ALFRED	14	M	64	99

Tip: Choose to view the Log

Either right-click within
the script window

Or go to
Window > Log



```
SimplePythonExample
Run Script Debug Script Reformat Script Balance Import as Data
1 Names Default To Here(1);
2
3 Python Init();
4
5 dt=open("$SAMPLE_DATA\Big Class.jmp");
6
7 Python Send(dt);
8 Python Submit("print (dt)");
9
10 Python Submit("\[
11 # The JMP Data table is transferred to Python as a pandas data frame
12 # Print out the column names
13 for col in dt.columns:
14     print( col )
15
16 # Create a new column and apply formula to data, creating pounds per inch
17 # weight / height
18 def my_formula(w,h):
19     return w / h
20
21 dt['lb_inch'] = dt.apply(lambda row: my_formula(row['weight'],row['height']), axis=1)
22
23 dt.head()
24
25 print(dt)
26
27
28 dt2 = Python Get(dt);
29 dt2 << New Data View;
30
31 Python Term();
32
```

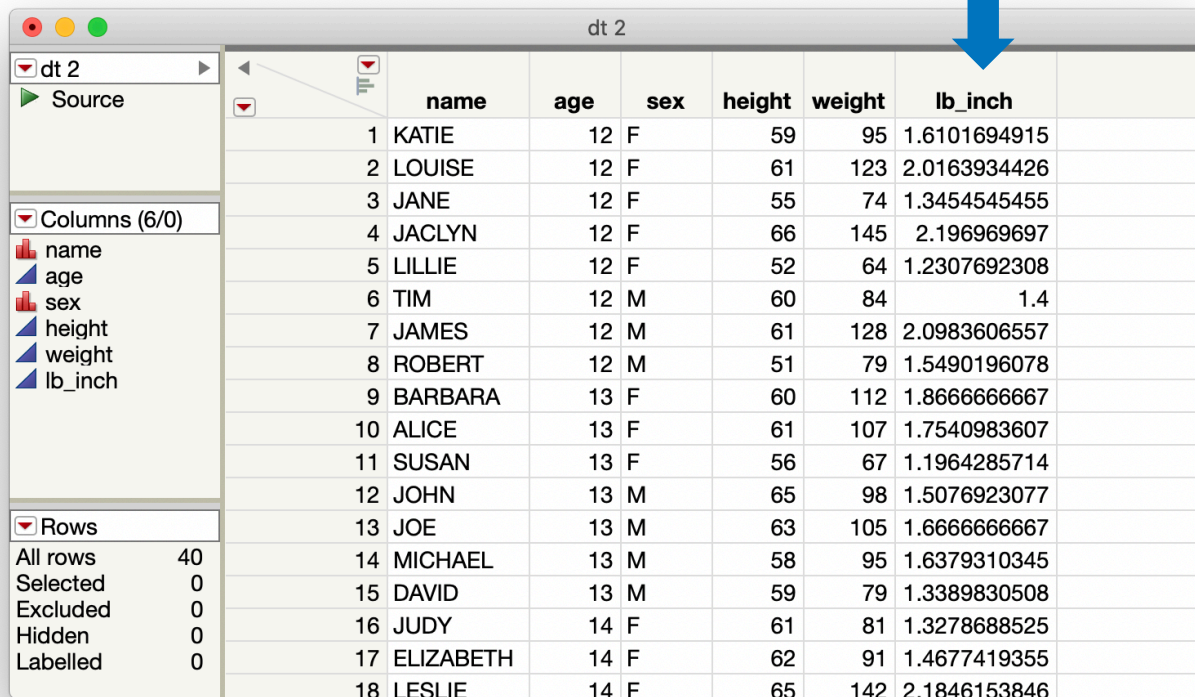
```
Log
Run Script Debug Script Reformat Script Balance Import as Data
Python Submit("\[
# The JMP Data table is transferred to Python as a pandas data
frame
# Print out the column names
for col in dt.columns:
    print( col )

# Create a new column and apply formula to data, creating pounds
per inch
# weight / height
def my_formula(w,h):
    return w / h

dt['lb_inch'] = dt.apply(lambda row:
my_formula(row['weight'],row['height']), axis=1)

dt.head()

print(dt)
\];
/*:
name
age
sex
height
weight
lb_inch
0 name age sex height weight lb_inch
1 KATIE 12 F 59 95 1.610169
2 LOUISE 12 F 61 123 2.016393
3 JANE 12 F 55 74 1.345455
4 JACLYN 12 F 66 145 2.196970
5 LILLIE 12 F 52 64 1.230769
6 TIM 12 M 60 84 1.400000
7 JAMES 12 M 61 128 2.098361
8 ROBERT 12 M 51 79 1.549020
9 BARBARA 13 F 60 112 1.866667
10 ALICE 13 F 61 107 1.754098
11 SUSAN 13 F 56 67 1.196429
12 JOHN 13 M 65 98 1.507692
13 JOE 13 M 63 105 1.666667
14 MICHAEL 13 M 58 95 1.637931
15 DAVID 13 M 59 79 1.338983
16 JUDY 14 F 61 81 1.327869
```



	name	age	sex	height	weight	lb_inch
1	KATIE	12	F	59	95	1.6101694915
2	LOUISE	12	F	61	123	2.0163934426
3	JANE	12	F	55	74	1.3454545455
4	JACLYN	12	F	66	145	2.196969697
5	LILLIE	12	F	52	64	1.2307692308
6	TIM	12	M	60	84	1.4
7	JAMES	12	M	61	128	2.0983606557
8	ROBERT	12	M	51	79	1.5490196078
9	BARBARA	13	F	60	112	1.8666666667
10	ALICE	13	F	61	107	1.7540983607
11	SUSAN	13	F	56	67	1.1964285714
12	JOHN	13	M	65	98	1.5076923077
13	JOE	13	M	63	105	1.6666666667
14	MICHAEL	13	M	58	95	1.6379310345
15	DAVID	13	M	59	79	1.3389830508
16	JUDY	14	F	61	81	1.3278688525
17	ELIZABETH	14	F	62	91	1.4677419355
18	LESLIE	14	F	65	142	2.1846153846

1. Sent data to
Python

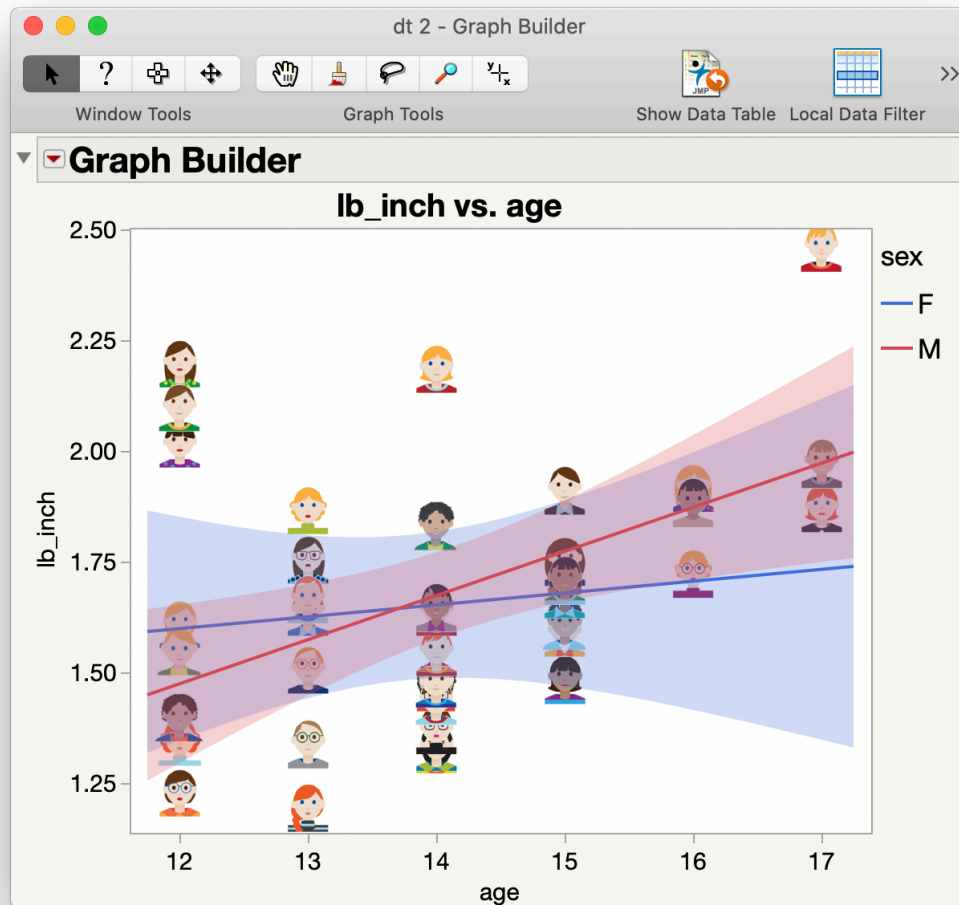
2. Ran some
Python code
on it

3. Brought the
resulting data
back into JMP

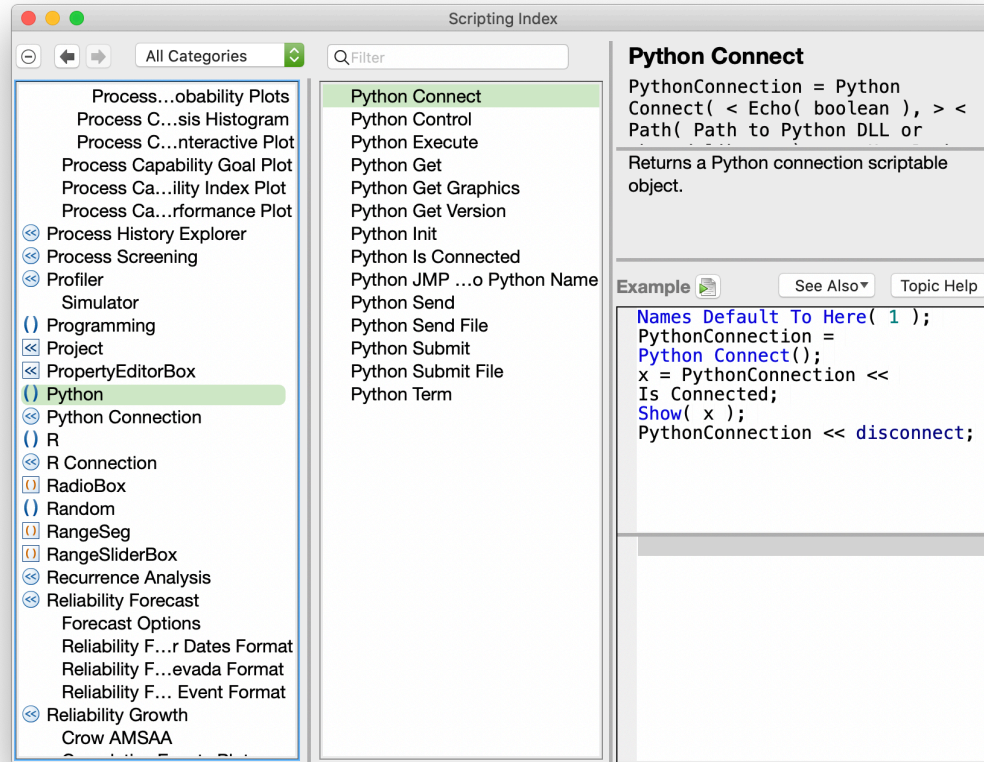
Now we can do more cool things in JMP with the results we brought back from Python...

```
SimplePythonExample
Run Script Debug Script Reformat Script Balance Import as Data
27
28 dt2 = Python Get(dt);
29 dt2 << New Data View;
30
31 Python Term();
32
33 dt3 = open("$SAMPLE_DATA\Big Class Families.jmp");
34
35 Data Table("dt") << Join(
36     With( Data Table(dt3)),
37     SelectWith( :picture),
38     Select( :name, :age, :sex, :height, :weight, :lb_inch),
39     By Row Number,
40     Output Table("dt")
41 )
```

	picture	name	age	sex	height	weight	lb_inch
		ROBERT	17	M	70	172	2.46
		ALFRED					
		ALICE					
		AMY					
		BARBARA					
		34 others					
1		KATIE	12	F	59	95	1.6101694...
2		LOUISE	12	F	61	123	2.0163934...
3		JANE	12	F	55	74	1.3454545...
4		JACLYN	12	F	66	145	2.196969697
5		LILLIE	12	F	52	64	1.2307692...
6		TIM	12	M	60	84	1.4
7		JAMES	12	M	61	128	2.0983606...
8		ROBERT	12	M	51	79	1.5490196...
9		BARBARA	13	F	60	112	1.8666666...
10		ALICE	13	F	61	107	1.7540983...
11		SUSAN	13	F	56	67	1.1964285...
12		JOHN	13	M	65	98	1.5076923...
13		JOE	13	M	63	105	1.6666666...
14		MICHAEL	13	M	58	95	1.6379310...
15		DAVID	13	M	59	79	1.3389830...



If you need help, go to Help > Scripting Index



Simple R Example

simpleRscript

Run Script

1 Na
2
3 //
4 dt
5 dt
6
7 R
8 R
9 R
10 R
11
12 R
13
14 //
15 JM
16
17 //
18 Ne
19
20
21 D
22
23
24
25
26
27
28
29)
30 |
31 R

New Data Table...

Columns (1/0)

log of RunTime

Rows

All rows 31
Selected 12
Excluded 0
Hidden 0
Labelled 0

New Data Table Name

	log of RunTime
1	3.688879454...
2	3.871201010...
3	3.806662489...
4	3.871201010...
5	3.784189633...
6	4.007333185...
7	4.025351690...
8	3.871201010...
9	3.891820298...
10	3.871201010...
11	4.127134385...
12	4.204692619...
13	3.806662489...
14	3.871201010...
15	3.912023005...

Fitness.jmp

Name	Sex	Age	Weight	Oxy	Runtime	RunPulse	RstPulse	MaxPulse	
1	Gonna	F	42	68.15	59.57	8.17	166	40	172
2	Gracie	F	38	81.87	60.06	8.63	170	48	186
3								5	168

New Data Table Name - Distribution

Window Tools

Distributions

log of RunTime

3.6 3.7 3.8 3.9 4 4.1 4.2 4.3

<https://community.jmp.com/t5/JMP-Add-Ins/Data-visualization-with-t-SNE-and-UMAP/ta-p/177969>

JMP-R connection used for JMP Add-In



MJ STAFF

Choose Language

Data visualization with t-SNE and UMAP

FEB 27, 2019 8:34 AM



mnist.jmp



Embedding.v1.2.jmpaddin



Description

Recently, non-linear dimension-reduction and visualization algorithms, most notably t-Distributed Stochastic Neighbor Embedding (t-SNE) and uniform manifold approximation and projection (UMAP), have been widely applied to various research areas such as image processing, text mining, and genomics. This Add-in provides access to both t-SNE and UMAP R packages. It offers a user-friendly interface enabling data table navigation, data quality control, sparsity handling, intuitive parameterization, and interactive results interpretation.

Usage Example

Here is a screenshot of the interface with MNIST data loaded. Under Model Specifications, I selected the label column as Label and all the pixels as predictors. I chose both t-SNE and UMAP as the algorithms.



<https://community.jmp.com/t5/JMP-Add-Ins/The-JMP-to-R-Add-In-Builder/ta-p/43879>



[juliagong](#) STAFF (RETIRED)

Choose Language

ARTICLE LABELS

The JMP to R Add-In Builder

AUG 31, 2017 8:11 AM



Update 27Jun2018: JMPtoRAddinBuilder.jmpaddin now contains unencrypted JSL source files.

Version 2 Update

I am happy to announce that the JMP to R Add-In Builder now supports the output of all output components (table, plot, etc.) as a single report! To enable this feature, install the attached "JMPtoRAddinBuilderWithReport". When defining your output components, check the "Return all results as report" box.

To see an example of the updated output format, scroll to the corresponding section in the post below. The SVM function with the report output enabled is also attached.

What is the JMP to R Add-In Builder?

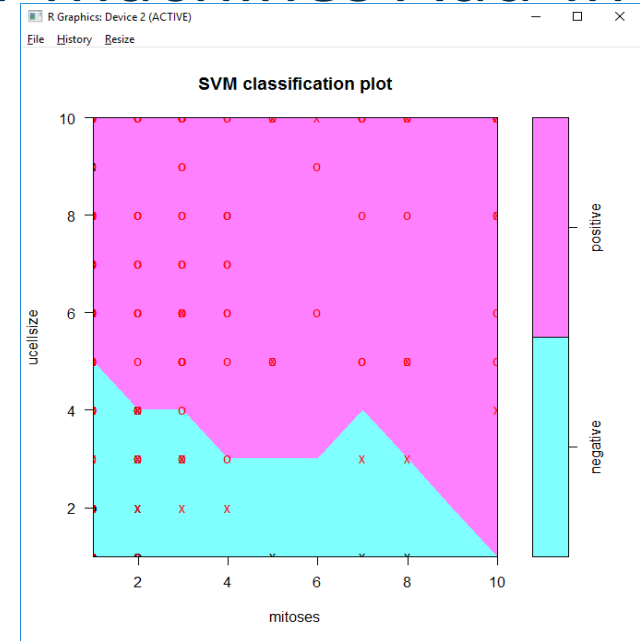
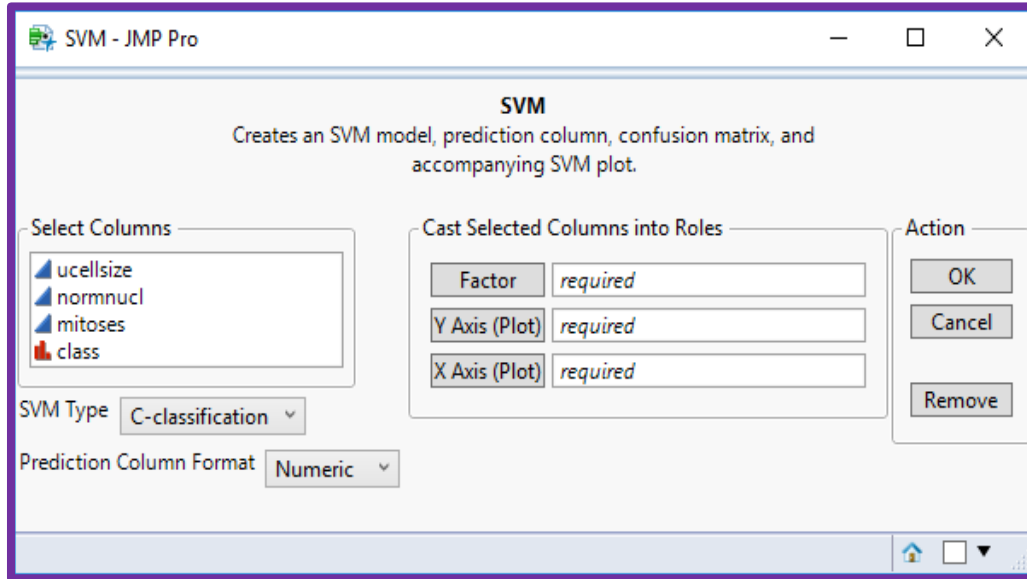
While JMP is a very powerful statistical software, JMP users may want to extend JMP beyond its included

Easily build a
JMP Add-In
calling R,
without using
any jsl!!!

[Jeff_Perkinson](#)

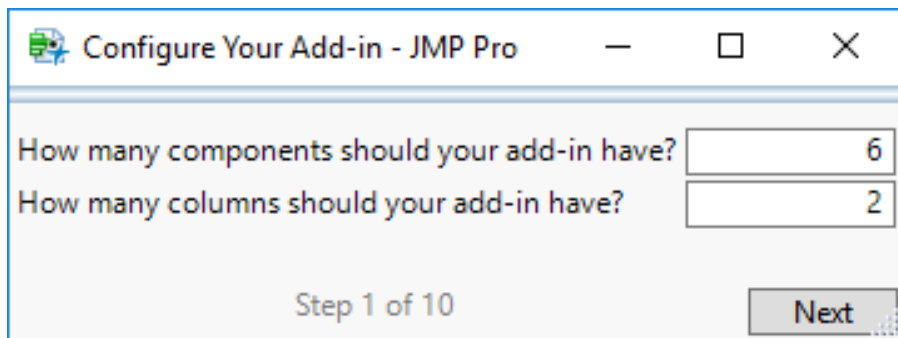


Goal: Make this Support Vector Machines Add-In



Predictions			
	pred	y	Freq
1	negative	negative	434
2	positive	negative	24
3	negative	positive	12
4	positive	positive	229

1.



Configure Your Add-in - JMP Pro

How many components should your add-in have? 6

How many columns should your add-in have? 2

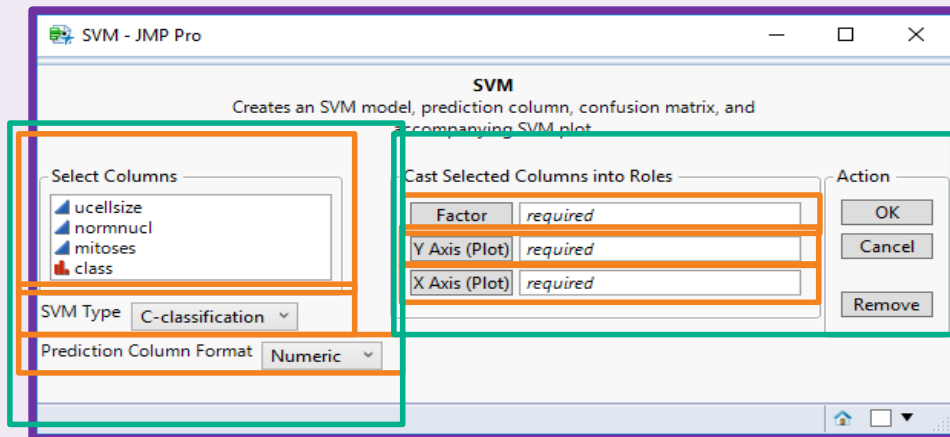
Step 1 of 10

Next

Component Types Available:

- Check Box
- Column List
- Column Select
- Dropdown Menu
- Numeric Value

Our goal:



SVM - JMP Pro

SVM

Creates an SVM model, prediction column, confusion matrix, and accompanying SVM plot

Select Columns

- ucellsize
- normnucl
- mitoses
- class

SVM Type C-classification

Prediction Column Format Numeric

Cast Selected Columns into Roles

Factor	required
Y Axis (Plot)	required
X Axis (Plot)	required

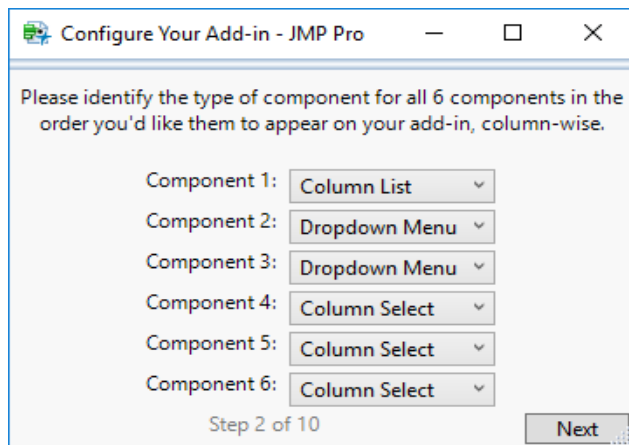
Action

OK

Cancel

Remove

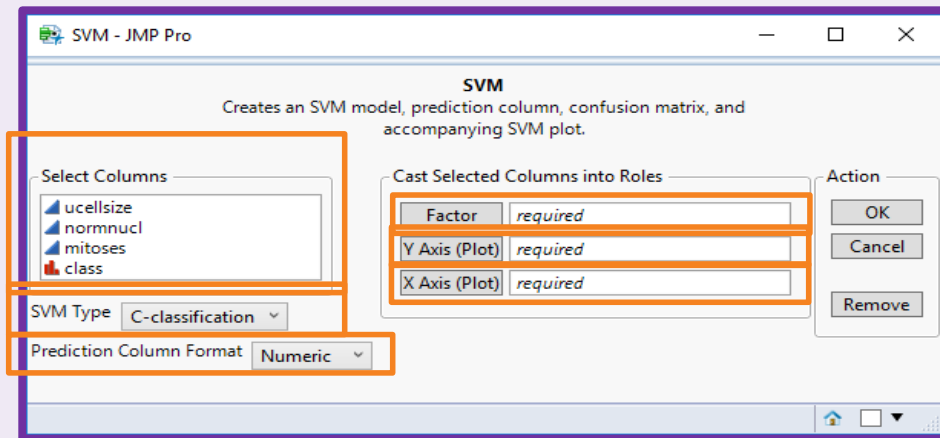
2.



Component Types Available:

- Check Box
- Column List
- Column Select
- Dropdown Menu
- Numeric Value

Our goal:



3.

Configure Your Add-in - JMP Pro

Please define what each component should do and in which column it should go.
Note: All column selects must go in the same column.

Column List: This is updated from your table. Column 1

Dropdown Menu: SVM Type,C-classification,eps-regression Column 1

Dropdown Menu: Prediction Column Format,Numeric,Character Column 1

Column Select: ☒ Required ☐ Multiple Columns ☐ Numeric Column 2

Column Select: ☒ Required ☐ Multiple Columns ☐ Numeric Column 2

Column Select: ☒ Required ☐ Multiple Columns ☐ Numeric Column 2

Step 3 of 10 Next

Choose the column for each component.

Give titles and option text.

4.

Configure Your Add-in - JMP Pro

Please define the names of each of your column select boxes.

Column Select 1 Name: Factor

Column Select 2 Name: Y Axis (Plot)

Column Select 3 Name: X Axis (Plot)

Step 4 of 10 Next

Our goal:

SVM

Creates an SVM model, prediction column, confusion matrix, and accompanying SVM plot.

Select Columns

- ucellsize
- normnucl
- mitoses
- class

SVM Type: C-classification

Prediction Column Format: Numeric

Cast Selected Columns into Roles

Role	Column	Required
Factor		required
Y Axis (Plot)		required
X Axis (Plot)		required

Action: OK Cancel Remove

5.

Configure Your Add-in - JMP Pro

Preview of Your Add-in:

Select Columns

- ucellsize
- normnucl
- mitoses
- class

SVM Type: C-classification

Prediction Column Format: Numeric

Cast Selected Columns into Roles

Factor	required
Y Axis (Plot)	required
X Axis (Plot)	required

Action

OK

Cancel

Remove

Next

Step 5 of 10

See a preview.

Give name and description.

6.

Configure Your Add-in - JMP Pro

Please give a name and optional description to your add-in.

Add-in Name: SVM

Add-in Description (optional): Creates an SVM model, prediction column, confusion matrix, and accompanying SVM plot.

Next

Step 6 of 10

Steps 7, 8, 9, and 10 are about your R code:

R Code for this example:

```
library(e1071)
library(gridExtra)

theText <- paste("x <- subset(dt, select=-", colnames(factor)[1], ")")
eval(parse(text=theText))
y <- factor
svm_model <- svm(x, y, type=svmType)
modelText <- paste("second.svm <- svm(", colnames(factor)[1], " ~ ., data = dt)")
eval(parse(text=modelText))

summary(svm_model)
summary(second.svm)
pred <- predict(svm_model, x)
pred <- as.vector(pred)


if(is.list(y)) {y <- unlist(y)}
predTable <- table(pred,y)
predTable <- as.data.frame(predTable)

eval(parse(text=paste(colnames(y_var)[1], "<- as.vector(as.matrix(y_var))")))
eval(parse(text=paste(colnames(x_var)[1], "<- as.vector(as.matrix(x_var))")))

max1 <- eval(parse(text=paste("max(", colnames(y_var)[1], ")")))
max2 <- eval(parse(text=paste("max(", colnames(x_var)[1], ")")))
gridSize <- eval(parse(text=paste("max(c(", max1, ",", max2, ")")))

eval(parse(text=paste("plot(second.svm, dt, ", colnames(y_var)[1], " ~ ", colnames(x_var)[1], ", fill=TRUE, grid=gridSize)"))
```

7.

 Configure Your Add-in - JMP Pro

Please define your R input and output variables.

R input variable name for SVM Type:

R input variable name for Prediction Column Format:

R input variable name for Factor:


R input variable name for Y Axis (Plot):

R input variable name for X Axis (Plot):

List all R output variable names here, separated by commas (','), no spaces:

Step 7 of 10 Next

8.

 Configure Your Add-in - JMP Pro

Enter all of the R code that your add-in will run. Enter the number of output display actions your function will have.

Type your R Code (in R syntax) here. Make sure your code references only the variables you defined earlier.

```
library(e1071)
library(gridExtra)

theText <- paste("x <- subset(dt, select=-", colnames(factor)[1], ")")
eval(parse(text=theText))
y <- factor
svm_model <- svm(x, y, type=svmType)
modelText <- paste("second.svm <- svm(", colnames(factor)[1], " ~ .,
eval(parse(text=modelText))
```

Number of output displays for this function:

Step 8 of 10 Next

9.

Configure Your Add-in - JMP Pro

Choose the output display format for each output.

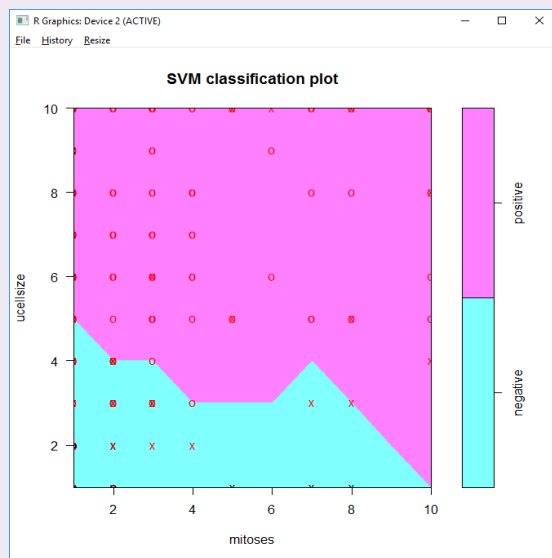
Output 1 type: Append Column in Existing Table

Output 2 type: Open Table

Output 3 type: Most Recent R Plot

Step 9 of 10

Next



Our goal:

Predictions
negative
negative
negative
positive
negative
negative
negative
negative

	pred	y	Freq
1	negative	negative	434
2	positive	negative	24
3	negative	positive	12
4	positive	positive	229

10.

Configure Your Add-in - JMP Pro

Provide the appropriate parameters for each output display.

Column Append Arguments

Column name: Predictions

Column type: User's Choice

Name of R output variable (list) with values: pred

Table Arguments

R input/output variable (contains target table): predTable

R Plot Arguments

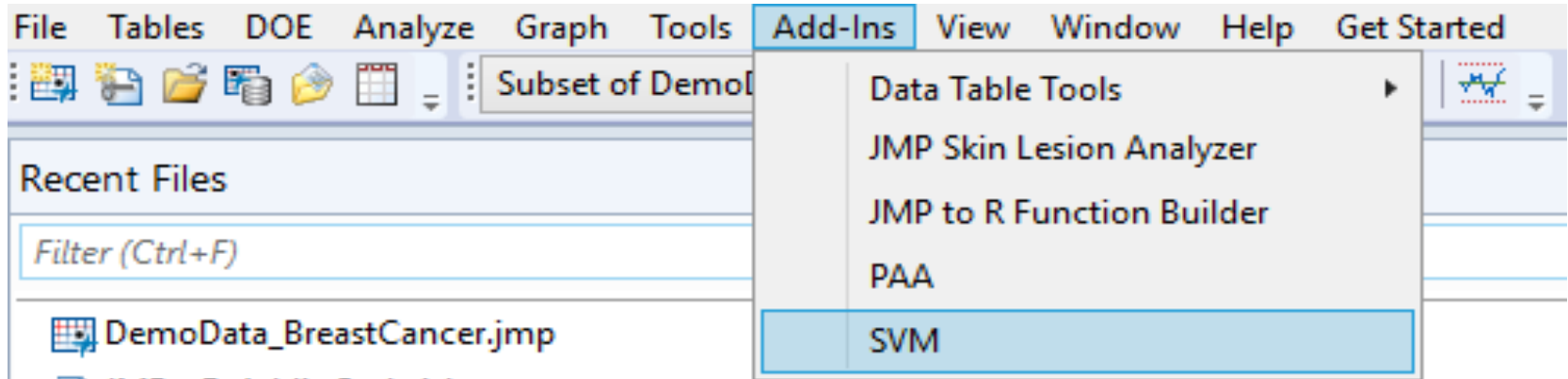
Graphics format (for most recently generated plot in R code): jpg

Save Add-in to... Desktop

Step 10 of 10

Next

Now you can share your Add-In for others to install and use:



Making an Add-In

Add-In Builder

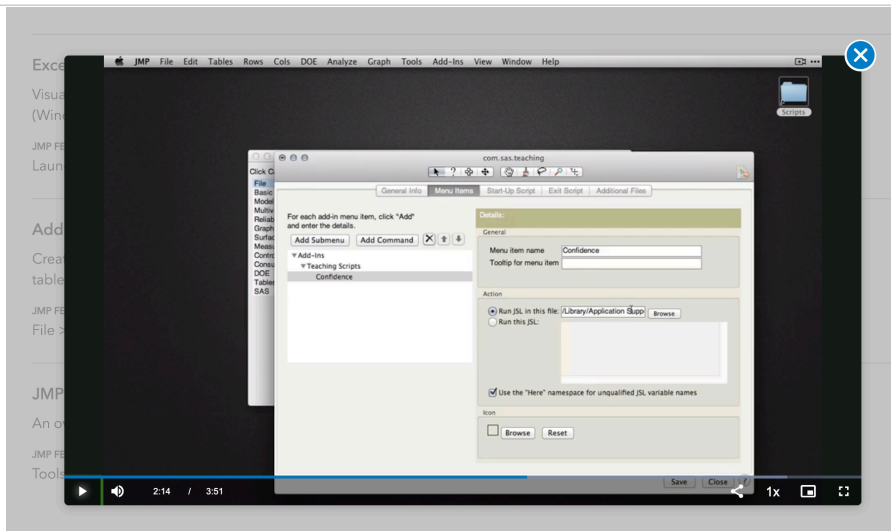
Create custom menus and easily distribute JMP scripts, applications, data tables and more.

JMP FEATURES DEMONSTRATED:

File > New > Add-In (or File > New > Add-In on the Mac)

> Video

> One-page guide (PDF)

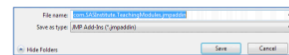
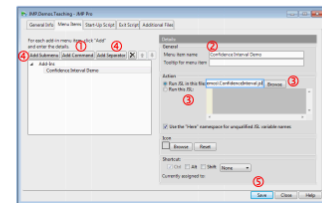
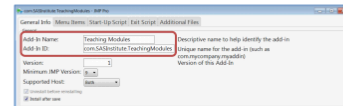


Add-In Builder

Use a **JMP® Add-In** to create custom menus and easily distribute JMP scripts, applications, data tables and more.

Create an Add-In with the Add-In Builder

1. Choose **File > New > Add-In** (or **File > New > New Add-In** on the Mac).
2. In the **General Information** tab, provide an Add-In Name and an Add-In ID.
3. In the **Menu Items** tab, for each menu item:
 - Click **Add Command** to add the menu item ❶ and provide a menu item name ❷.
 - **Browse** for the JSL script the menu item will run. Or, copy and paste the script next to **Run this JSL** ❸.
 - Click **Add Submenu** (if needed) and provide a name to group items below a single heading, or **Add Separator** to add a separating line ❹.
4. If desired, use the **Start-Up Script**, **Exit Script**, or **Additional Files** tabs to add other scripts, graphics or data tables to the add-in.
5. Click **Save** ❺. Edit the file name if desired (the Add-in ID will default, with a .jmpaddin extension), and click **Save**.



By default, the add-in will automatically install under the Add-In menu in JMP when saved. Test the menu items to ensure that everything works as intended, and fine-tune as needed. The Add-In Builder will stay open until you close it, and saved changes will override the previously installed version of the add-in.

Share the add-in with other users. When the file is opened in JMP, all files are extracted into the appropriate folder, and the add-in is installed in the JMP Add-in menu.

Tips and Limitations

- To **disable** (remove from menu) or **unregister** (uninstall) an add-in, use **View > Add-Ins**.
- To edit an add-in use **File > Open**, then click the arrow next to Open and select **Open Using Add-In Builder** (on Mac, first click "Options" then check "Edit after opening").
- To save an application as an add-in, select **Script > Save Script to Add-In** from the Application Builder red triangle (saving an application as an add-in requires JMP 10 or higher).



Note: For more information on developing and deploying add-ins, search for "Add-In" in the JMP Help or in the JMP Scripting Guide (under **Help > Books**). Explore [jmp.com/addins](https://www.jmp.com/en_us/learning-library/using-jmp.html) for available add-ins.

How-to-make-an-Add-In: https://www.jmp.com/en_us/learning-library/using-jmp.html

Links in the JMP Help

- JMP Help and more examples:
www.jmp.com/support/help/en/15.1/#page/jmp/extending-jmp.shtml
- JMP Help for the code generation:
<http://www.jmp.com/support/help/en/15.1/#page/jmp/formula-depot.shtml>
- Python installation and versioning:
<https://www.jmp.com/support/help/en/15.1/#page/jmp/install-python.shtml> and
<http://www.jmp.com/support/help/en/15.1/#page/jmp/troubleshooting-the-jmp-python-integration.shtml>
- R installation and versioning can be found here:
<http://www.jmp.com/support/help/en/15.1/#page/jmp/installing-r.shtml>
(R 3.3.3 or earlier for Mac OS:
<https://www.jmp.com/support/notes/61/841.html>)

- JMP with R and Python: https://www.jmp.com/en_us/whitepapers/jmp/jmp-synergies-using-jmp-and-jmp-pro-with-python-and-r.html
- SVM Add-In and JMP-to-R Add-In Builder: <https://community.jmp.com/t5/JMP-Add-Ins/The-JMP-to-R-Add-In-Builder/ta-p/43879>
- T-SNE and UMAP: <https://community.jmp.com/t5/JMP-Add-Ins/Data-visualization-with-t-SNE-and-UMAP/ta-p/177969>
- How-to-make-an-Add-In: https://www.jmp.com/en_us/learning-library/using-jmp.html

- Help with the R and Python commands in jsl: In JMP, go to Help > Scripting Index
- Help with jsl: So many resources!
 - www.jmp.com > Learn JMP > Mastering JMP videos
 - Training through SAS Education or free courses through your academic JMP campus license (contact academic@jmp.com to get access)
 - Ask and search questions in community.jmp.com
 - <https://community.jmp.com/t5/Mastering-JMP/What-Resources-are-available-to-learn-JSL-JMP-Scripting-Language/ta-p/23370>

Questions?

jmp.com

