Sinergias con JMP®: Cómo usar JMP® y JMP® Pro con Python y R

DOCUMENTO TÉCNICO

Índice

Introducción

JMP es un software de visualización de datos y análisis estadístico autónomo y con múltiples funcionalidades de SAS para ordenadores Windows y Mac. Gracias a la interactividad y los enlaces dinámicos, JMP hace que la exploración de datos resulte emocionante y reveladora e incluye muchas opciones de análisis avanzadas, por lo que satisface completamente las necesidades de exploradores de datos y analistas. Aún así, puede que haya ocasiones en las que quiera (o necesite) usar JMP junto con herramientas de código abierto como Python o R.

Por ejemplo:

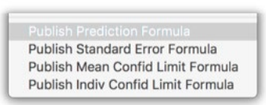
- Un estadístico de una empresa que no puede usar software no validado, como R, para hacer análisis primarios, pero que quizás quiera explorar nuevas metodologías con un paquete de R junto a métodos validados en JMP.
- Un programador de Python o R que quiera aprovechar los enlaces dinámicos, las visualizaciones avanzadas y los potentes análisis de JMP.
- Un departamento de análisis de datos de una empresa que quiera desarrollar un complemento de JMP que permita ejecutar código Python o R como complemento dentro de JMP, de manera que los futuros usuarios puedan interactuar con la interfaz gráfica sin necesidad de escribir ellos mismos el código Python o R.
- Un profesor de estadística que quiera presentar paquetes específicos de código abierto en clase, pero dentro de un entorno de «apuntar y hacer clic» con el que el estudiante no tenga que programar.

Sea cual sea su motivación para conectar herramientas de código abierto (u otras) con la interfaz gráfica de usuario del software JMP, esta guía le ayudará a empezar a usar las conexiones de este programa con Python y R.[1]

JMP® y Python

Cómo generar código Python desde un modelo de JMP Pro

Con JMP Pro puede generar código Python para los modelos predictivos que cree en JMP. Esto es útil para crear, explorar y comparar modelos en JMP Pro, pero implementarlos en un entorno de producción distinto. JMP Pro ofrece una manera fácil de construir múltiples modelos y compararlos entre sí, e incluso de crear una media de los modelos (modelo de conjunto) en la plataforma Almacén de fórmulas.



Primero, ajuste el modelo usando las herramientas del menú Análisis de JMP. Para guardar el modelo en el Almacén de fórmulas, use las opciones «Publicar fórmula de la probabilidad» (para modelos de clasificación) o «Publicar fórmula de la predicción» (para modelos de predicción). Normalmente, ambos están en las opciones de los triángulos rojos de los resultados del modelo o en «Guardar columnas».

[1] JMP también puede conectarse con SAS y Matlab (además de R y Python), y también genera código C, JavaScript, SAS y de puntuación de modelos SQL (además de Python). Para más información sobre cualquiera de estas conexiones JMP y para ver más consejos sobre cómo empezar y ejemplos, visite jmp.com/support/help/en/15.1/#page/jmp/extending-jmp.shtml.

Una vez que haya seleccionado la palabra clave «Publicar» (o utilizado la opción «Agregar fórmula desde columna»), su modelo se enviará al Almacén de fórmulas. El Almacén de fórmulas es una plataforma que le permite guardar uno o varios modelos que compiten entre sí, compararlos, crear modelos de conjunto e implementarlos en diversos entornos de producción.

«Generar código Python» es una opción que está bajo el triángulo rojo en cada modelo individual (p. ej., este modelo de «Ajuste por mínimos cuadrados» para predecir **pesos**) o bajo el triángulo rojo del Almacén de fórmulas para generar el código de todo el conjunto de modelos guardados en la ventana del Almacén de fórmulas.

El código Python resultante, que se muestra a continuación, incluye algunas dependencias obligatorias, que son llamadas en la primera sección del código. A continuación tenemos la información de derechos de autor, después un poco de código Python con los nombres y tipos de variables, y finalmente la fórmula que se usará para predecir la variable respuesta para las observaciones nuevas con valores conocidos para las variables predictoras (en este ejemplo, *la edad, la altura y el sexo*).

Fit_Least_Squares_weight

```
from __future__ import division
import jmp_score as jmp
from math import *
import numpy as np

""" ================================================================
Copyright(C) 2018 SAS Institute Inc.All rights reserved.

Notice:
The following permissions are granted provided that the
above copyright and this notice appear in the score code and
any related documentation. Permission to copy, modify
and distribute the score code generated using
JMP(R) software is limited to customers of SAS Institute Inc. ("SAS")
and successive third parties, all without any warranty, express or
implied, or any other obligation by SAS. SAS and all other SAS
Institute Inc. product and service names are registered
trademarks or trademarks of SAS Institute Inc. in the USA
and other countries. Except as contained in this notice,
the name of the SAS Institute Inc. and JMP shall not be used in
the advertising or promotion of products or services without
prior written authorization from SAS Institute Inc.
================================================================ """

""" Python code generated by JMP v14.1.0 """

def getModelMetadata():
    return {"creator": u"Fit Least Squares", "modelName": u"", "predicted": u"weight",
    "table": u"Big Class.jmp", "version": u"14.1.0", "timestamp": u"2018-08-08T04:02:07Z"}


def getInputMetadata():
    return {
        u"age": "float",
        u"height": "float",
        u"sex": "str"
    }


def getOutputMetadata():
    return {
        u"Pred Formula weight": "float"
    }


def score(indata, outdata):
    _temp_0 = np.nan
    _temp_1 = np.nan

    if (indata[u"sex"] == u"F"):
        _temp_0 = 2.0492068900361
    elif (indata[u"sex"] == u"M"):
        _temp_0 = -2.0492068900361
    else:
        _temp_0 = np.nan
    if (jmp.numeq(indata[u"age"], 12)):
        _temp_1 = 0
    elif (jmp.numeq(indata[u"age"], 13)):
        _temp_1 = -13.6855931672148
    elif (jmp.numeq(indata[u"age"], 14)):
        _temp_1 = -25.8472610743993
    elif (jmp.numeq(indata[u"age"], 15)):
        _temp_1 = -19.7698493666905
    elif (jmp.numeq(indata[u"age"], 16)):
        _temp_1 = -10.1590212259529
    elif (jmp.numeq(indata[u"age"], 17)):
        _temp_1 = 2.52025614257322
    else:
        _temp_1 = np.nan
    outdata[u"Pred Formula weight"] = -176.033203126788 + 4.72294023921341 * indata[u"height"]
    + _temp_0 + _temp_1

    return outdata[u"Pred Formula weight"]
```

Puede encontrar más información sobre este tema en jmp.com/support/help/en/15.1/#page/jmp/formula-depot.shtml.

Aspectos fundamentales para conectar JMP y Python

Además de generar código de puntuación para modelos Python desde JMP, también puede abrir la conexión entre JMP y Python y enviar y recibir datos, resultados y acciones (usando una combinación de JMP y Python) entre Python y JMP. (La conexión entre JMP y R funciona de la misma manera). Para utilizar la conexión con Python en JMP, usará el lenguaje de scripting de JMP (JSL) como contenedor para su código Python.

Aquí podemos verlo con un ejemplo simple. Necesitará tener Python[2] instalado en su ordenador, pero es el caso de la mayoría de los ordenadores nuevos.

(1) Abra la conexión con Python desde JMP.

(2) Ejecute una operación, como enviar datos de JMP a Python.

(3) Ejecute otra operación, como enviar código Python directamente a Python para crear una variable nueva.

(4) Ejecute otra operación más, como llevar de vuelta un resultado o un dato desde Python a JMP (quizás para visualizar el resultado en JMP).

(5) Cierre la conexión con Python.

Aquí tiene el código de este ejemplo simple. Para seguir los pasos, primero tendrá que abrir JMP. A continuación, diríjase a Archivo > Nuevo > Nuevo script para abrir una nueva ventana de script en blanco. A continuación, escriba lo siguiente:

SimplePythonExample

Run Script | Debug Script | Reformat Script | Balance | Import as Data

```
Names Default To Here(1);

Python Init();

dt=open("$SAMPLE_DATA\Big Class.jmp");

Python Send(dt);
Python Submit("print (dt)");

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)
]\");

dt2 = Python Get(dt);
dt2 << New Data View;

Python Term();

```

En el script de arriba hay nueve fragmentos de código JSL (indicados en azul). La primera línea ayuda a garantizar que las referencias a archivos o variables que hagamos estarán relacionadas con este script. La tercera línea, «**`Python Init`**();», abre Python.[3]

[2] python.org.

[3] Aquí puede encontrar los detalles de la instalación y versiones de Python: jmp.com/support/help/en/15.1/#page/jmp/install-python.shtml# y jmp.com/support/help/en/15.1/#page/jmp/troubleshooting-the-jmp-python-integration.shtml#ww822804.

La quinta línea, «**dt = Open**(**"$SAMPLE_DATA/Big Class.jmp"**);», le pide a JMP que abra un conjunto de datos que está en el directorio de datos de muestra de JMP y le asigna el nombre «dt».[4] El conjunto de datos de este ejemplo se llama «Big Class.jmp.».

La séptima línea, «**Python Send(dt)**;», envía la tabla de datos «dt» de JMP a Python. La tabla de datos de JMP se transferirá a Python como marco de datos de Pandas. La octava línea, «**Python Submit("print(dt)");**», envía código Python a Python. La información entre comillas es el script de Python; el resto de la línea es el contenedor de código JSL. Es útil hacer un seguimiento de sus envíos de código y solucionar los problemas, si es necesario, utilizando la ventana registro de JMP. Diríjase a la barra de menú de JMP y elija Ventana → Registro para abrir el registro JMP. Aquí puede ver todos los mensajes de error o resultados internos de Python. Por ejemplo, si revisamos el registro tras ejecutar este código hasta este punto, veremos la siguiente salida de código Python en el registro:

```
dt=open("$SAMPLE_DATA\Big Class.jmp");
/*:

Data Table( "Big Class.jmp" )
//:*/
Python Send(dt);
Python Submit("print (dt)");
/*:

      name  age sex  height  weight
0    KATIE   12   F      59      95
1   LOUISE   12   F      61     123
2     JANE   12   F      55      74
3   JACLYN   12   F      66     145
4   LILLIE   12   F      52      64
```

Las líneas 10-26 incluyen una instrucción «**Python Submit();**» más larga. Este fragmento de código Python, incrustado en la instrucción de código JSL «**Python Submit()**», imprimirá los nombres de columnas (en el registro) y después creará una columna nueva con la fórmula «lb_inch = weight/height,». Finalmente, imprimirá la nueva tabla de datos, que añade esta columna nueva a nuestra antigua columna de datos, en el registro:

```
/*:

name
age
sex
height
weight
      name  age sex  height  weight    lb_inch
0    KATIE   12   F      59      95  1.610169
1   LOUISE   12   F      61     123  2.016393
2     JANE   12   F      55      74  1.345455
3   JACLYN   12   F      66     145  2.196970
4   LILLIE   12   F      52      64  1.230769
5      TIM   12   M      60      84  1.400000
6    JAMES   12   M      61     128  2.098361
```

Las líneas 28 y 29 devuelven la tabla de datos actualizada a la memoria operativa del software y la abren como una nueva tabla de datos de JMP.

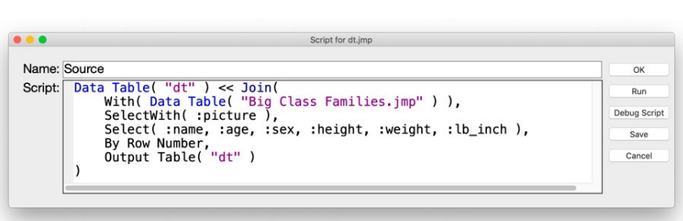
La última línea de código, «**Python Term();**», finaliza la conexión con Python.

[4] Nota: hemos utilizado una línea de código JSL ligeramente diferente en el ejemplo de R para apuntar a un archivo de datos abierto.
Esta es otra manera de apuntar a datos, en este caso para abrir un archivo de datos, utilizando JSL.

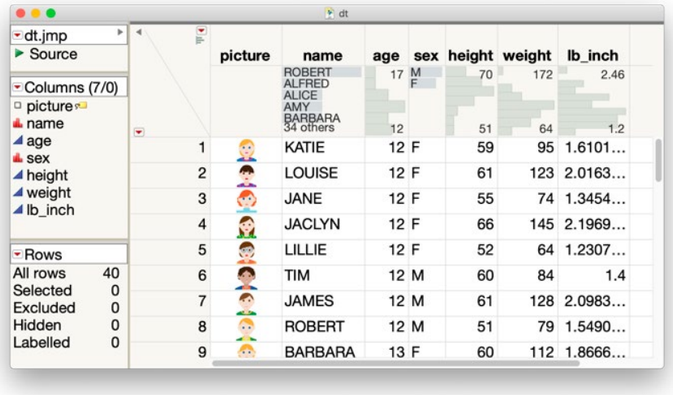
Un poco más de diversión con JMP : cómo añadir un marcador de imagen a nuestro gráfico

Aunque la última sección de este ejemplo simple no hace más uso de la conexión de JMP con Python, sí que ilustra los beneficios potenciales de utilizar JMP con Python (o con R u otros programas conectados). En especial, podemos trabajar de manera interactiva entre JMP y Python, aprovechando las ventajas de ejecutar código Python prescrito o paquetes especiales de Python y después mejorando nuestros análisis o visualizaciones finales utilizando las funcionalidades dinámicas de JMP.

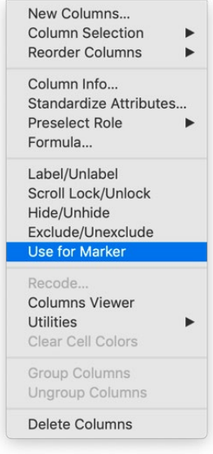
Por ejemplo, tras enviar la tabla de datos «dt» de vuelta a JMP, podemos usar algunas de las atractivas funcionalidades de visualización de JMP para mejorar nuestras gráficas con solo unos pocos clics o scripts breves de JSL. Para ilustrarlo, vamos a combinar nuestra tabla «dt» con el conjunto de datos de muestra «Grandes clases de familias», disponible en la Librería de datos de muestra de JMP, para añadir una columna de imágenes a nuestra nueva «dt». Primero, abra Grandes clases de familias (haciendo clic en Ayuda > Librería de datos de muestra > Grandes clases de familias o ejecutando el código «**dt = Open("$SAMPLE_DATA/Big Class.jmp");**») y después utilice Tablas > Combinar (o el script a continuación) para unir la columna de imágenes a nuestra tabla «dt».

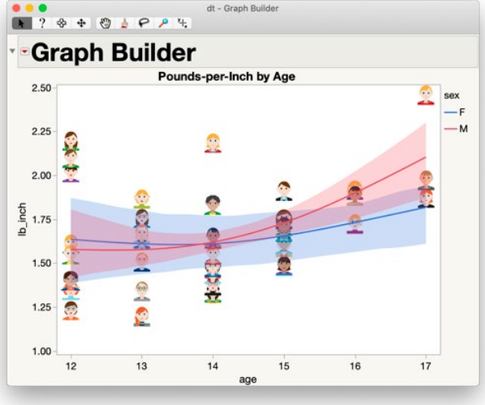


Estos pasos añaden una columna llamada «imagen» a nuestro conjunto de datos e importan las imágenes estudentizadas de la tabla de datos Grandes clases de familias en nuestra nueva tabla de datos combinada.



Ahora podemos crear una visualización de la nueva variable de libras por pulgada, usando las imágenes como marcadores en nuestro gráfico. Primero convertimos la columna de imagen en un marcador haciendo clic en la columna «imagen» y yendo a Cols > Usar para marcador.





En este ejemplo simple, usamos la conexión de JMP con Python para abrir los datos en JMP, enviarlos a Python, crear una columna nueva en Python, añadir la columna nueva al marco de datos de Pandas de Python y después devolver los datos actualizados como tabla de datos de JMP, que podemos seguir analizando y manipulando en JMP.

Más ejemplos simples de JMP con Python

Puede encontrar algunos ejemplos simples más de interacciones de JMP con Python en la Ayuda de JMP, en Guía de scripting → Cómo ampliar JMP → Cómo trabajar con Python, o en jmp.com/support/help/en/15.1/#page/jmp/additional-python-integration-examples.shtml.

Aquí encontrará el código para:

- Enviar una tabla de datos a Python.
- Crear objetos en Python.
- Operaciones matriz en Python.

JMP® y R

Conectar JMP con el software de código abierto R[5] permite aprovechar fácilmente las capacidades de R junto con el enfoque dinámico e interactivo de análisis y visualización del software de JMP. También puede crear interfaces con menús fáciles de usar para los paquetes de R y combinar elementos de JMP y R. Todo esto es posible cuando tiene tanto JMP como R instalados en su ordenador. Puede encontrar una lista de las funciones de R para ejecutar estas conexiones, junto con sus descripciones (con ejemplos) en el Índice de scripts (Ayuda > Índice de scripts > Funciones de R).

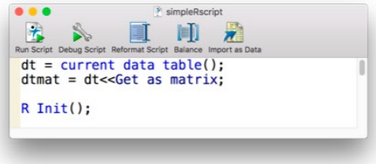
Aspectos fundamentales para conectar JMP y R

Para usar la conexión de JMP con R, tiene que ajustar su código R utilizando el lenguaje de scripting de JMP (JSL).

Por ejemplo, vamos a mover unos datos de JMP a R para analizarlos y manipularlos y después los devolveremos a JMP. El proceso es bastante sencillo.Escribimos un poco de código JSL para abrir R, enviamos datos de JMP a R, enviamos el código R a R, y volvemos a traer los datos de R a JMP.

Primero, asegúrese de que tiene tanto R como JMP instalados en el mismo ordenador.[6]

A continuación, abra una ventana de scripting en JMP desde **Archivo > Nuevo > Nuevo Script**.

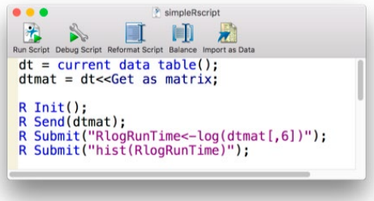


También vamos a abrir un conjunto de datos de muestra en JMP. Diríjase a **Ayuda > Librería de datos de muestra** y abra la tabla de datos **Fitness.jmp**. Para enviar esto a R, necesitamos usar dos líneas simples de JSL. La primera línea básicamente identifica la tabla de datos abierta, **Fitness.jmp**, y le cambia el nombre a «dt». La segunda línea convierte la tabla de datos recién llamada «dt» en un formato de matriz. Como las opciones de estructura de datos de R son ligeramente diferentes a las de JMP, el formato de matriz es un entorno simple que funciona con facilidad tanto en JMP como en R. A continuación, llamamos a R desde JMP. Si R está instalado en el mismo ordenador, JMP lo encontrará; no necesita configurar una conexión ni dirigir JMP a la instalación de R. Solo necesita escribir la línea JSL «`R Init();`». JMP buscará R en el ordenador y a continuación lo iniciará (o lo abrirá).

[5] cran.r-project.org.

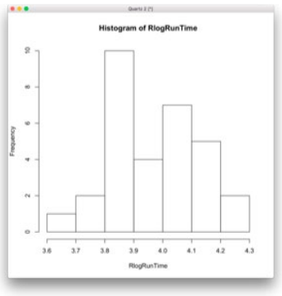
[6] Aquí puede encontrar los detalles de la instalación y versiones de R: jmp.com/support/help/en/15.1/#page/jmp/installing-r.shtml.

Ya estamos preparados para enviar y recibir información entre JMP y R. La primera línea del código a continuación envía nuestra versión matriz de los datos, llamada dtmat, a R. La siguiente línea envía código R para que se ejecute en R.



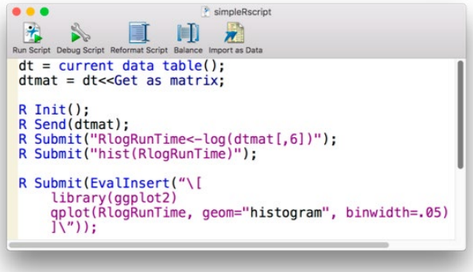
La línea «`RlogRunTime<-log(dtmat[,6]);`» toma la sexta columna de la matiz de datos de entrada (el «Mile Run Time» [«tiempo en correr una milla»] del conjunto de datos Fitness) y lo transforma logarítmicamente. Ahora, estos resultados se llamarán «RlogRunTime» (porque es el nombre que le hemos asignado en este fragmento de código R).

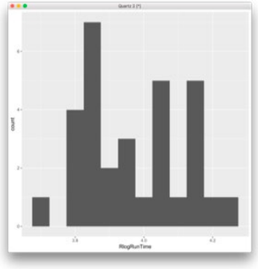
La línea «`R Submit("hist(RlogRunTime)");`» genera un histograma en R, que aparecerá en su escritorio una vez que haya ejecutado esta línea de código.



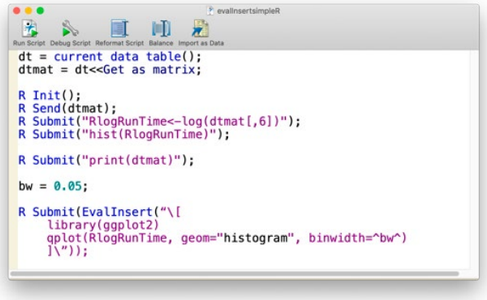
Para ejecutar el código, seleccione una o más líneas (cada línea debe terminar con «;») y haga clic en el botón «Ejecutar script» en la parte de arriba de la ventana de script:

Si tenemos que enviar un fragmento de código R de múltiples líneas a través de nuestro contenedor JSL, podemos usar «`R Submit(EvalInsert("\[____]\"))`» en lugar del simple «`R Submit("___")`», como puede observar en el código R a continuación, en el que utilizamos el paquete «`ggplot2`» en R para crear un histograma:

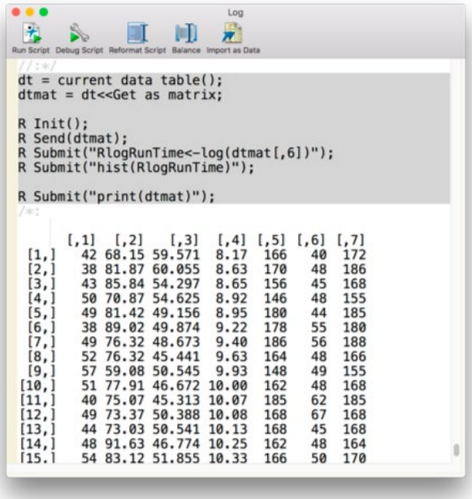




Este «`EvalInsert`()» también es útil en general cuando necesita hacer sustituciones o evaluar expresiones dentro de una cadena de caracteres en JSL. Por ejemplo, podemos especificar el ancho de la barra para el histograma R como variable en el script JSL y después llamar a esa variable usando el código «`EvalInsert`()» así:

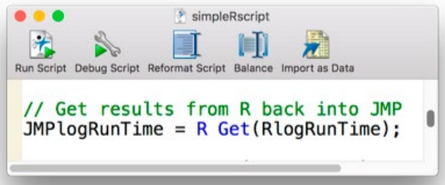


Haga un seguimiento de sus envíos de código y, en caso necesario, solucione los problemas utilizando la ventana Registro de JMP. Vaya a la barra de menú de JMP y seleccione Ventana → Registro para abrir el registro JMP. Aquí puede ver todos los mensajes de error o resultados internos de R.



Por supuesto, también podemos crear este histograma en JMP, utilizando la plataforma Distribución. Podemos utilizar las funciones de JMP de dos maneras diferentes, como se indica a continuación.

En primer lugar, tenemos que traer «**RlogRunTime**» de vuelta a R desde JMP. Esta sección de código toma los datos recién transformados, llamados «**RlogRunTime**» en R, y los trae otra vez a JMP con el nuevo nombre «**JMPlogRunTime**».

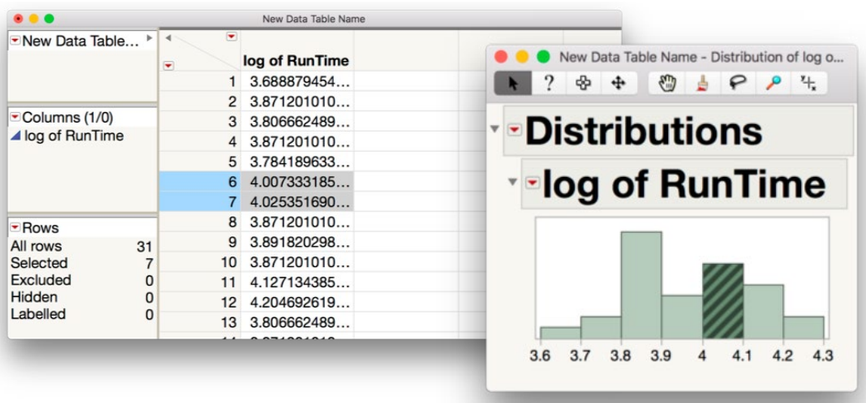


También necesitamos mover estos datos JMP nuevos desde la memoria interna hasta una tabla de datos de JMP nueva, de manera que podamos guardarlos y seguir operando con ellos. El código a continuación crea una nueva tabla de datos JMP con una columna, formada por los datos de «**JMPlogRunTime**».

A continuación, podemos escoger seguir usando funciones JMP de cualquiera de las dos maneras siguientes.

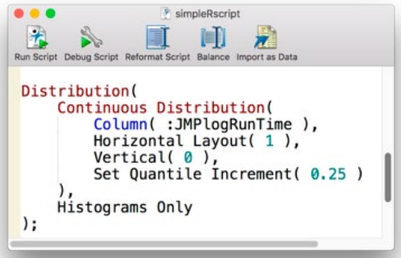
(1) Operando en la nueva tabla de datos de JMP, usando las funciones de JMP de los menús de apuntar y hacer clic.

En nuestro ejemplo, podemos usar la función Análisis > Distribución para crear un histograma interactivo en JMP.



(2) Usar JSL para seguir trabajando en una ventana de script para llamar a esas mismas funciones de apuntar y hacer clic de JMP.

Si está familiarizado con JSL, puede escribir más acciones de JMP directamente en su ventana de script. Si no está familiarizado con el JSL que se corresponde con sus acciones de apuntar y hacer clic, puede generar fácilmente ese JSL usando apuntar y hacer clic para crear esas acciones una vez. A continuación, haga clic en el triángulo rojo de la ventana de resultados y escoja Copiar script a la ventana de script. Para actualizar el script para una nueva tabla de datos, edite el script JSL para aplicarlo a la variable o variables apropiadas.



Más ejemplos simples de JMP con R

Para encontrar más ejemplos simples de cómo utilizar JMP en combinación con R, busque «R» en el menú Ayuda de JMP, en Guía de scripting → Cómo ampliar JMP → Cómo trabajar con R.

En la ayuda de JMP encontrará el código para:

- Enviar una tabla de datos a R.
- Crear objetos en R.
- Usar funciones y gráficos de R.
- Adición simple de matrices en R.
- Obtener intervalos de confianza con bootstrap en R.

Ejemplos avanzados

Una vez que haya entendido las bases de usar las interfaces de JMP con R y Python, puede ampliar este uso de maneras interesantes e interactivas.

A continuación encontrará enlaces a algunos ejemplos:

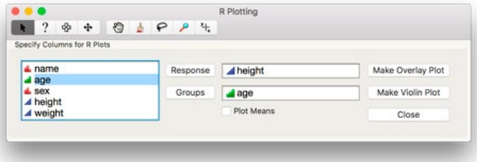
Llamador interactivo de JMP a GGPlot

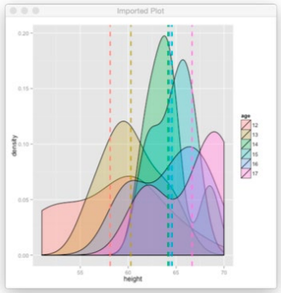
Este script utiliza la conexión entre JMP y R para crear gráficos de densidad con curvas lisas[7] con el paquete de R ggplot2,[8] separados por los niveles de una variable de agrupación.

[7] Nota: este tipo de vista de histograma también es una opción nativa del constructor de gráficos de JMP en la versión 15, así que el uso de R en este ejemplo es para ilustrar el proceso, no por necesidad.

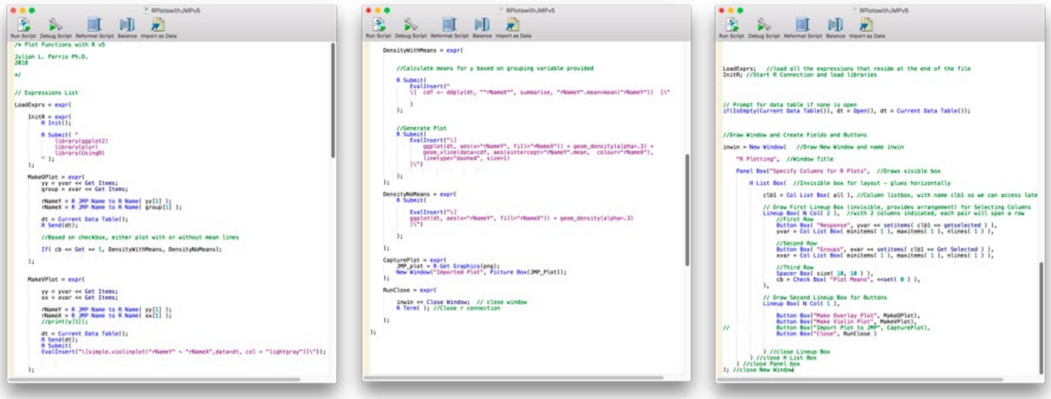
[8] Véase cran.r-project.org/web/packages/ggplot2/index.html.

Para ejecutarlo, escoja un archivo de datos y después abra y ejecute este script, que abre un cuadro de diálogo con las columnas de la tabla de datos abierta. Al interactuar con este cuadro, se crea el gráfico de R resultante.





Para usar este script, no necesita escribir código R ni JSL. Sin embargo, si quiere crear algo similar, el JSL subyacente (que contiene fragmentos de código R) se muestra aquí (puede descargarse desde el enlace a este ejemplo):



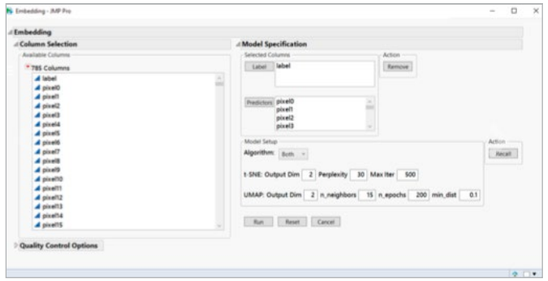
Para saber más y descargar este complemento, visite la comunidad de JMP en community.jmp.com/docs/DOC-6472.

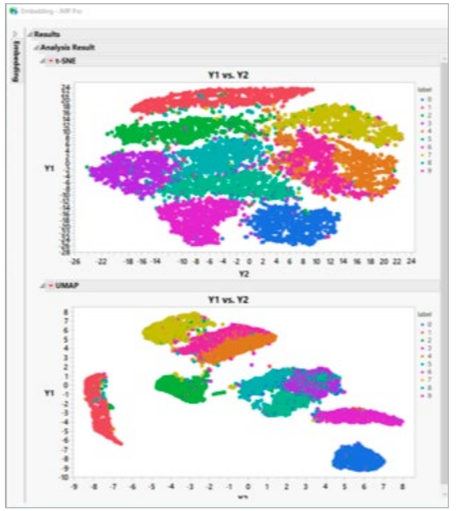
Visualización de datos con el complemento de t-SNE y UMAP:

Este script, que utiliza los paquetes de t-SNE y UMAP[9] de R, es aún más fácil de usar cuando se combina con un complemento de JMP.

La incrustación de vecinos estocástica con distribución t (t-SNE) y la aproximación y proyección múltiple uniforme (UMAP) son algoritmos no lineales de reducción de dimensiones y visualización que se están popularizando en campos como el procesamiento de imágenes, la minería de texto y la genómica. Este complemento ofrece una interfaz fácil de usar para estos dos paquetes de R, que habilita navegación por tablas de datos, control de la calidad de los datos, gestión de la dispersión, parametrización intuitiva e interpretación de resultados interactiva.

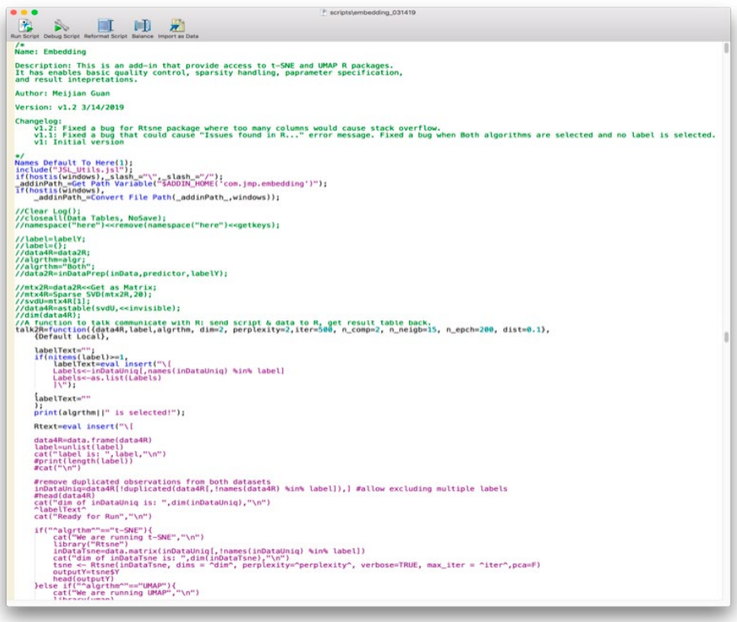
[9] Véase cran.r-project.org/web/packages/Rtsne/ and cran.r-project.org/web/packages/umap/.





Para crear un complemento como este, solo necesita que un usuario desarrolle el código Python o R, además del código JSL que le sirve de contenedor. A continuación, este usuario lo combinará todo en un complemento de JMP con el Constructor de complementos de JMP.[10] El archivo .jmpaddin resultante puede enviarse a otros usuarios por correo electrónico o publicarse en la web.[11] Este proceso permite que los siguientes usuarios ejecuten el complemento sin necesidad de escribir, ver ni editar el código. En lugar de ello, interactúan con la visualización de apuntar y hacer clic fácil de usar de JMP.

El código que no estamos viendo y que se está ejecutando en segundo plano empieza así (el código R está en color violeta):



[10] Para más información sobre cómo crear un complemento de JMP desde su script JSL, consulte jmp.com/content/dam/jmp/documents/en/academic/learning-library/01-add-in-builder.pdf.

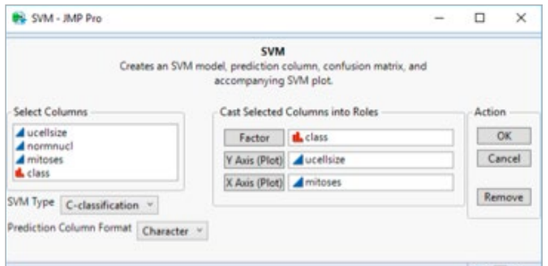
[11] Publíquelo en nuestra comunidad de usuarios de JMP en community.jmp.com para que otros usuarios puedan localizarlo.

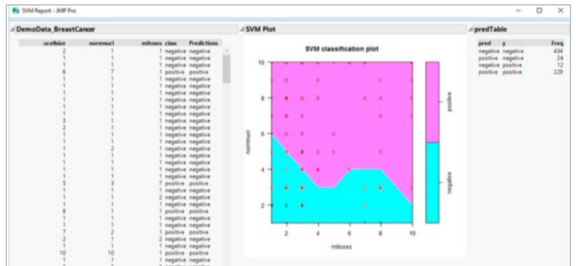
Una vez escrito, los siguientes usuarios no necesitarán ver este código ni interactuar con él en absoluto. Esta funcionalidad le permite escribirlo una vez e introducirlo en un complemento. Ahora dispone de una interfaz de apuntar y hacer clic para llamar a funciones de R o de Python.

Para saber más, y para descargar este complemento, visite la comunidad de JMP en community.jmp.com/t5/JMP-Add-Ins/Data-visualization-with-t-SNE-and-UMAP/ta-p/177969.

Complemento de SVM:

Este complemento llama a la función svm desde el paquete e1071.[12] El usuario del complemento SVM interactúa con este cuadro de diálogo para obtener el informe de clasificación SVM[13] resultante:





Lo que no ve el usuario es el código que se está ejecutando en segundo plano:

```
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)")))
```

Crear este complemento es aún más fácil con el constructor de complementos de JMP a R (véase el siguiente ejemplo). El constructor de complementos de JMP a R le permite saltarse la programación de JSL en su totalidad y limitarse a interactuar con algunos cuadros de diálogo, dejando el código R o Python en el cuadro de peticiones.

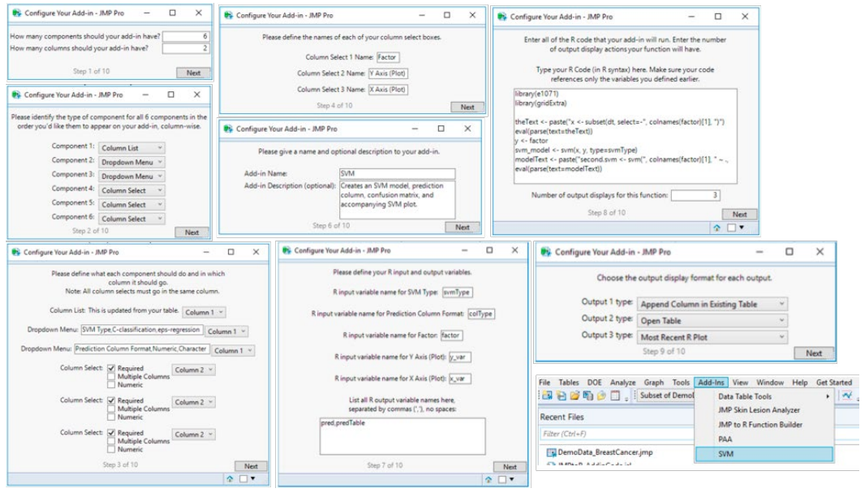
[12] Véase cran.r-project.org/web/packages/e1071/index.html.

[13] JMP Pro añadió SVM a la suite de modelos predictivos en la versión 15 así que, de nuevo, este ejemplo sirve para ilustrar el potencial de la conexión a R pero no es necesario para los usuarios de JMP Pro 15.

El constructor de complementos de JMP a R: Cómo usar la conexión de JMP a R sin JSL

Esta herramienta permite a los usuarios definir y utilizar complementos JMP personalizados que ejecutan funciones R. Para usar este creador de interfaces, primero necesita desarrollar el código R que quiere enviar a R, pero no tiene que escribir ningún código JSL. De manera similar, los usuarios finales de su nueva interfaz de apuntar y hacer clic no necesitarán utilizar ni ver ese código R: solo verán la interfaz de apuntar y soltar fácil de usar. (Nota: este complemento solo funciona en el sistema operativo Windows, no en Mac).

Este complemento le permite usar la conexión de JMP con R saltándose la parte del JSL. En su lugar, interactuará con algunos cuadros de diálogo para crear los cuadros, botones y peticiones que quiera incluir en su complemento. La mayoría de los pasos se muestran aquí:



El complemento de SVM del ejemplo anterior se creó utilizando este constructor de complementos de JMP a R.

Para saber más y descargar este constructor de complementos de JMP a R, visite la comunidad de JMP en community.jmp.com/t5/JMP-Add-Ins/The-JMP-to-R-Add-In-Builder/ta-p/43879.

Conclusión

En esta guía hemos visto una introducción a la generación de código de puntuación Python, el scripting de Python en JMP y el scripting de R en JMP. También hemos facilitado enlaces a ejemplos adicionales, incluyendo algunos ejemplos de R avanzados que utilizan la funcionalidad de constructor de complementos de JMP. Para más información sobre cualquiera de estos temas, consulte las notas a pie de página y los enlaces facilitados a lo largo del artículo. Si tiene alguna pregunta, necesita ayuda para implementar alguna de estas técnicas o simplemente quiere compartir algo que ha descubierto, publique en nuestra comunidad de usuarios de JMP en community.jmp.com.

Acerca de SAS y JMP

SAS, el líder mundial en análisis, creó JMP (pronunciado «jump», como «salto» en inglés) en 1989 para que los científicos, ingenieros y otros analistas pudieran explorar y analizar datos de manera visual e interactiva. Desde entonces, JMP ha pasado de ser un producto único a ser una familia de herramientas de descubrimiento estadístico, cada una de ellas adaptada para satisfacer necesidades específicas. John Sall, el cofundador y vicepresidente ejecutivo de SAS, lidera la unidad de negocio de JMP.

SAS Institute Inc. Sede mundial +1 919-677-8000

JMP es una solución de software de SAS. Para conocer más acerca de SAS, visite **sas.com**

Para obtener información de ventas de JMP en Estados Unidos y Canadá, llame al teléfono 877-594-6567 o visite **jmp.com**