# Verwendung von JMP® und JMP® Pro mit Python und R

WHITE PAPER

# Inhaltsverzeichnis

# Einführung

JMP ist ein eigenständiges, umfassendes SAS-Softwarepaket für die Erstellung von Datenvisualisierungen und die Durchführung von statistischen Analysen auf Windows- und macOS-Desktopgeräten. JMP-Anwender profitieren von einer großen Interaktivität, einer dynamischen Datenkopplung sowie unzähligen erweiterten Analyseoptionen. Sie haben dadurch eine größere Kontrolle über Ihre Daten und können mit einem hohen Maß an Autonomie aufschlussreiche und anschauliche Datenanalysen erstellen. Trotz dieser bereits umfangreichen Funktionen stoßen Sie mitunter womöglich auch auf Anwendungsfälle, bei denen Sie JMP in Verbindung mit Open-Source-Werkzeugen wie Python oder R nutzen möchten (oder müssen).

Hier ein paar Beispiele:

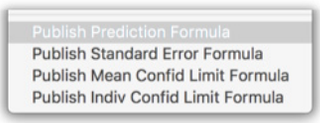
- Ein Unternehmensstatistiker, der keine nicht validierte Software (z. B. R) für primäre Analysen verwenden darf, könnte nichtsdestotrotz eine neue Methodik aus einem R-Paket in Kombination mit validierten Methoden in JMP erkunden wollen.
- Ein Python- oder R-Programmierer möchte von der dynamischen Kopplung, den fortschrittlichen visuellen Inhalten oder den leistungsstarken Analysen von JMP Gebrauch machen.
- Die Datenanalyseabteilung eines Unternehmens möchte ein Add-in für JMP entwickeln, mit dem ausgewählte R- oder Python-Codes als Add-ins in JMP ausgeführt werden können. Damit wird Benutzern ermöglicht, zukünftig einfach mit dem GUI-Add-in zu interagieren, ohne selbst den R- oder Python-Code schreiben zu müssen.
- Ein Statistiklehrer möchte seinen Schülern bestimmte Open-Source-Pakete in einer mausgesteuerten Umgebung zur Verfügung stellen, sodass die Schüler selbst keinen Code schreiben müssen, um die Pakete zu nutzen.

Wenn Sie beabsichtigen, Open-Source-Werkzeuge (oder andere Werkzeuge) mit der GUI von JMP zu verbinden, kann Ihnen dieser Leitfaden, der die Nutzung der Python- und R-Verbindungen in JMP darlegt, als gute erste Einführung in diesen Themenbereich dienen.[1]

# JMP® und Python

## Generierung von Python-Code in einem JMP Pro-Modell

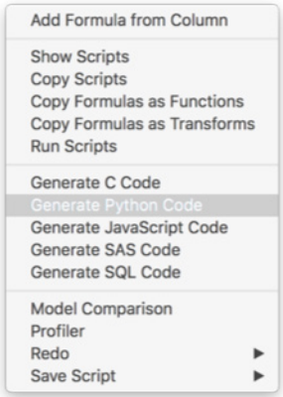

Mit JMP Pro können Sie Python-Codes für Vorhersagemodelle generieren, die Sie in JMP erstellt haben. Diese Vorgehensweise ist besonders nützlich, wenn Sie Modelle zwar in JMP Pro erstellen, analysieren und vergleichen, sie jedoch in einer anderen Produktionsumgebung bereitstellen möchten. JMP Pro ermöglicht es Ihnen, über das Formeldepot mühelos Modelle zu generieren, konkurrierende Modelle zu vergleichen und sogar einen Modellmittelwert (Ensemble) zu erstellen.



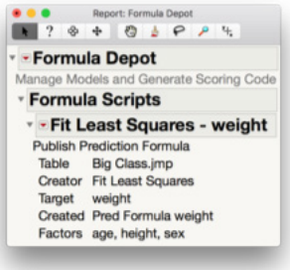

Erstellen Sie dafür zunächst in JMP mithilfe der Werkzeuge im Menü „Analysieren“ ein Modell. Um das Modell im Formeldepot zu speichern, verwenden Sie entweder die Option „Wahrscheinlichkeitsformel veröffentlichen“ (für Klassifikationsmodelle) oder „Vorhersageformel veröffentlichen“ (für Vorhersagemodelle). Diese Optionen können innerhalb der Modellergebnis-Fenster über die roten Dreiecke (Drop-down-Menüs) oder unter „Spalten speichern“ aufgerufen werden.

[1] Es gibt auch JMP-Verbindungen zu SAS und Matlab (zusätzlich zu R und Python) und JMP generiert auch Modell-Scoring-Code in C, JavaScript, SAS und SQL (zusätzlich zu Python). Weitere Informationen zu diesen JMP-Verbindungen und noch mehr Tipps zu den ersten Schritten mit JMP sowie entsprechende Beispiele finden Sie auf jmp.com/support/help/en/15.1/#page/jmp/extending-jmp.shtml.

Nachdem Sie eine Option mit dem Schlüsselwort „Veröffentlichen“ ausgewählt haben (oder die Option „Formel aus Spalte hinzufügen“), wird Ihr Modell an das Formeldepot gesendet. Beim Formeldepot handelt es sich um eine Plattform, auf der Sie ein Modell oder mehrere konkurrierende Modelle speichern und vergleichen sowie Ensemble-Modelle erstellen und diese in einer Reihe von Produktionsumgebungen bereitstellen können.




Die Option „Python-Code generieren“ im roten Dreieck-Menü eines Modells (in diesem Beispiel das „Fit Least Squares“-Modell [Kleinste Quadrate anpassen] für die Vorhersage des **Gewichts**) oder in dem roten Dreieck-Menü des Formeldepots ermöglicht es Ihnen, den Code für den gesamten Satz an Modellen, die im Formeldepot-Fenster gespeichert sind, zu generieren.




Der generierte Python-Code (unten abgebildet) enthält eine Reihe erforderlicher Abhängigkeiten, die im ersten Code-Abschnitt aufgerufen werden. Darauf folgen Urheberrechtsinformationen und ein Python-Code zur Generierung der Variablennamen und -typen. Ganz am Ende steht die Formel, mit der die Zielgröße für neue Beobachtungen mit bekannten Werten für die Prädiktorvariablen vorhergesagt wird (in diesem Beispiel *Alter, Größe und Geschlecht*).

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"]
```

Weitere Informationen zu diesem Thema finden Sie unter jmp.com/support/help/en/15.1/#page/jmp/formula-depot.shtml.

## Grundlagen der Verbindung zwischen JMP und Python

Sie können innerhalb von JMP nicht nur Modell-Scoring-Code in Python generieren, sondern auch eine Verbindung zwischen JMP und Python öffnen, um Daten, Ergebnisse und Aktionen (mithilfe einer Kombination aus JMP- und Python-Code) zwischen Python und JMP auszutauschen. (Die Verbindung zwischen JMP und R funktioniert nach dem gleichen Prinzip.) Um die Python-Verbindung in JMP nutzen zu können, müssen Sie die JMP-Skript-Sprache (JSL) als Wrapper für Ihren Python-Code verwenden.

Sehen wir uns zur Veranschaulichung ein einfaches Beispiel an. Sie müssen dafür Python[2] auf Ihrem Gerät installiert haben (auf den meisten neuen Computern ist die Software standardmäßig verfügbar).

(1) Öffnen Sie in JMP die Python-Verbindung.

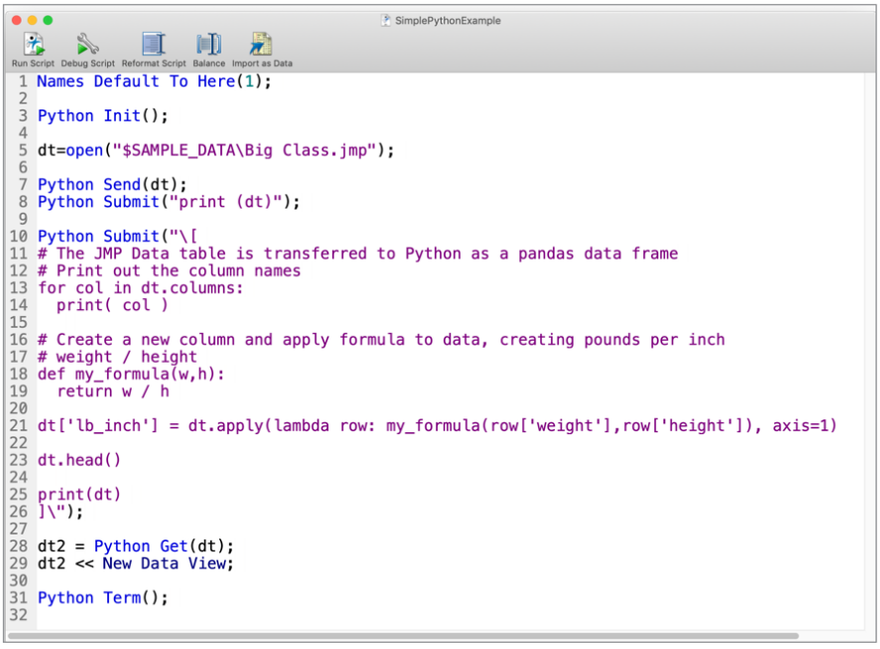
(2) Führen Sie eine beliebige Operation durch, z. B. Daten von JMP an Python senden.

(3) Führen Sie eine weitere Operation durch, z. B. Python-Code direkt an Python übermitteln, um eine neue Variable zu erstellen.

(4) Führen Sie eine weitere Operation durch, z. B. ein Ergebnis aus Python zurück an JMP übermitteln (indem Sie beispielsweise das Ergebnis in JMP visualisieren).

(5) Schließen Sie die Python-Verbindung.

Im Folgenden sehen Sie den Code für dieses einfache Beispiel. Um das Beispiel nachzubilden, müssen Sie zunächst JMP öffnen. Gehen Sie anschließend zu „Datei" > „Neu" > „Neues Skript", um ein leeres Skriptfenster zu öffnen. Geben Sie dann Folgendes ein:




```
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();
```

In dem Skript gibt es neun JSL-Code-Elemente (in Blau hervorgehoben). Mit der ersten Zeile sorgen wir dafür, dass jegliche Referenzen auf Dateien und Variablen sich auf dieses Skript beziehen. Die dritte Zeile unseres Skripts „**`Python Init`**();" öffnet Python[3].

[2] python.org.

[3] Die Details der Python-Installation und -Versionen können Sie aufrufen unter: jmp.com/support/help/en/15.1/#page/jmp/install-python.shtml# und jmp.com/support/help/en/15.1/#page/jmp/troubleshooting-the-jmp-python-integration.shtml#ww822804.

Die 5. Zeile „**dt = Open**(**"$SAMPLE_DATA/Big Class.jmp"**);" gibt JMP den Auftrag, einen im Verzeichnis der JMP-Beispieldaten vorhandenen Datensatz zu öffnen, und weist diesem die Bezeichnung „dt" zu.[4] Der Datensatz in diesem Beispiel trägt die Bezeichnung „Big Class.jmp."

Die 7. Zeile „**Python Send(dt)**;" sendet die „dt"-Datentabelle aus JMP an Python. Die JMP-Datentabelle wird als Pandas-Datenrahmen an Python übermittelt. Mit Zeile 8 „**Python Submit("print(dt)");**" senden wir einen Python-Code an Python. Bei den Informationen innerhalb der Anführungszeichen handelt es sich um das Python-Skript – der restliche Code in dieser Zeile ist der JSL-Code-Wrapper. Das JMP-Log-Fenster steht Ihnen zur Verfügung, damit Sie Ihren eigenen Code überwachen und bei Bedarf Fehler beheben können. Um es anzuzeigen, öffnen Sie die JMP-Menüleiste und wählen Sie „Fenster" → „Log". Hier werden Ihnen jegliche interne Python-Fehlermeldungen und -ergebnisse angezeigt. Wenn wir unseren Code bis zu diesem Punkt ausführen lassen, sehen wir etwa die folgende Ausgabe im Log:

```
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
```

Die Zeilen 10–26 enthalten eine längere „**Python Submit();**"-Anweisung. Dieser Python-Code-Abschnitt, der in der „**Python Submit()**"-Anweisung des JSL-Codes eingebettet ist, gibt zunächst die Spaltennamen im Log aus und erstellt dann eine neue Spalte aus der Formel „lb_inch = weight/height". Zum Schluss wird im Log die neue Datentabelle mit der folgenden neuen Spalte ausgegeben:

```
/*:

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
```

Mit den Zeilen 28 und 29 übermitteln wir die aktualisierte Datentabelle zurück in den Arbeitsspeicher und öffnen sie als neue JMP-Datentabelle.

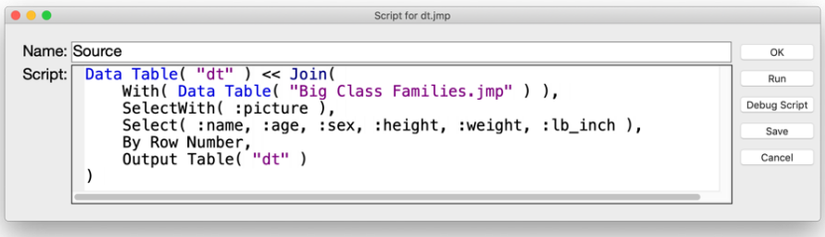
Die letzte Codezeile „**Python Term();**" beendet die Verbindung mit Python.

[4] Beachten Sie bitte, dass wir in dem R-Beispiel eine leicht veränderte JSL-Codezeile verwendet haben, um auf die offene Datendatei zu verweisen. Hierbei handelt es sich um eine weitere Methode, um auf Daten zu verweisen – in diesem Fall, um mithilfe von JSL eine Datendatei zu öffnen.

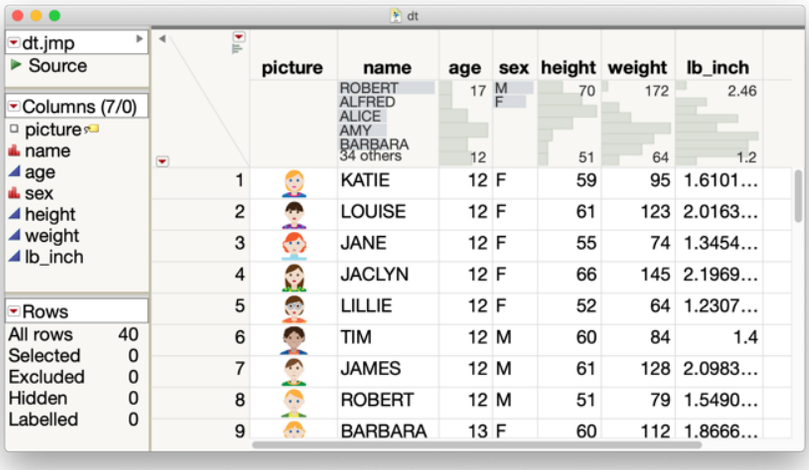
## Bildsymbole zu Diagrammen in JMP hinzufügen

In diesem letzten Abschnitt möchten wir uns noch einen Anwendungsfall von JMP ansehen, in dem wir zwar nicht die direkte Verbindung zwischen JMP und Python nutzen, der jedoch die potenziellen Vorteile der Nutzung von JMP und Python (oder R oder anderer verbundener Software) hervorhebt. Wir sehen uns an, wie wir interaktiv zwischen JMP und Python arbeiten können, indem wir einen vorgefertigten Python-Code oder spezielle Python-Pakete nutzen und unsere finalen Analysen oder Visualisierungen durch die Verwendung dynamischer Features von JMP verbessern.

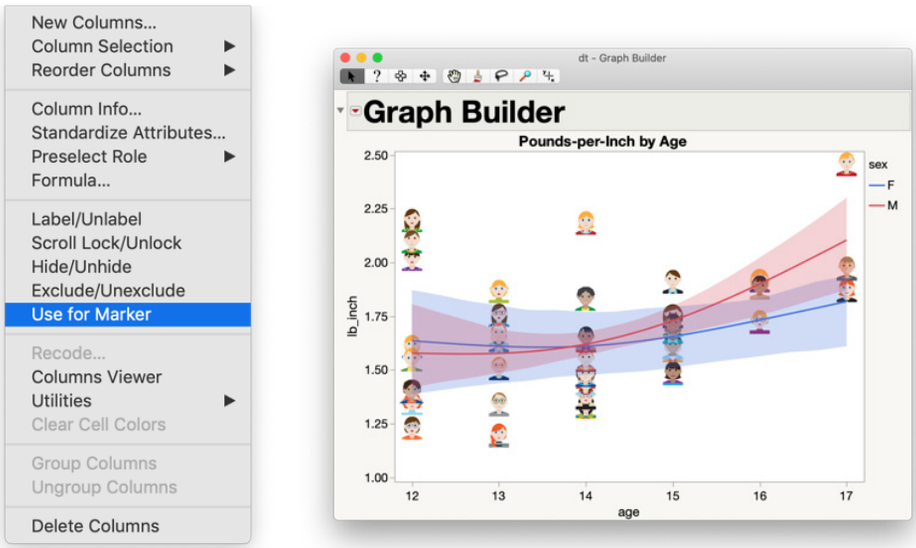
Nachdem wir die „dt"-Datentabelle zurück in JMP übermittelt haben, können wir beispielsweise die JMP-Visualisierungsfunktionen nutzen, um unsere Graphiken mit ein paar Klicks oder kurzen JSL-Skripten aufzuwerten. Hierfür verbinden wir zunächst unsere „dt"-Tabelle mit dem Beispieldatensatz „Big Class Families", der in der JMP-Beispieldatenbibliothek verfügbar ist, um der Tabelle eine Spalte mit Bildern hinzuzufügen. Öffnen Sie dafür zunächst „Big Class Families" (gehen Sie zu „Hilfe" > „Beispieldatenbibliothek" > „Big Class Families" oder führen Sie den Code „`dt = Open("$SAMPLE_DATA/Big Class.jmp");`" aus) und wählen Sie „Tabellen" > „Verbinden" (oder geben Sie das folgende Skript ein), um die Bilderspalte in Ihre „dt"-Tabelle aufzunehmen.




Mit diesen Schritten fügen wir eine Spalte mit der Bezeichnung „picture" (Bild) zu unserem Datensatz hinzu und importieren die Studierendenbilder aus der Datentabelle „Big Class Families" in unsere Datentabelle.



Nun können wir eine Visualisierung der neuen „Pounds-per-Inch"-Variable (Pfund-pro-Inch) erstellen und diese Bilder als Diagrammsymbole verwenden. Zunächst müssen wir bestimmen, dass die Inhalte der Bilderspalte als Symbole verwendet werden sollen. Klicken Sie dafür auf die „picture"-Spalte und wählen Sie „Spalten" > „Als Symbol verwenden" aus.




In diesem einfachen Beispiel haben wir nun also die JMP/Python-Schnittstelle verwendet, um zunächst Daten in JMP zu öffnen und an Python zu senden. Dann haben wir eine neue Spalte in Python erstellt, welche dem Python-Pandas-Datenrahmen hinzugefügt wurde. Anschließend wurden die aktualisierten Daten als neue JMP-Datentabelle ausgegeben, die wir dann (in dem letzten Abschnitt dieses Beispiels) in JMP weiter analysieren und bearbeiten konnten.

## JMP und Python – zusätzliche einfache Beispiele

Weitere einfache Beispiele für die Nutzung von JMP und Python finden Sie in der JMP-Hilfe, indem Sie „Scripting Guide" → „JMP erweitern" → „Mit Python arbeiten" auswählen oder indem Sie jmp.com/support/help/en/15.1/#page/jmp/additional-python-integration-examples.shtml besuchen.

Dort finden Sie auch den Code für folgende Aktionen:

- Datentabellen an Python senden
- Objekte in Python erstellen
- Matrix-Operationen in Python durchführen

# JMP® und R

Indem Sie JMP mit der Open-Source-Software R[5] verbinden, können Sie die Fähigkeiten von R zusammen mit den dynamischen und interaktiven Analysen und Datenvisualisierungen der JMP-Software nutzen. Zudem können Sie nutzerfreundliche Menü-Schnittstellen für R-Pakete erstellen und verschiedene Elemente von JMP und R miteinander kombinieren. Um von diesen Möglichkeiten zu profitieren, müssen Sie sowohl JMP als auch R auf Ihrem Computer installiert haben. Die R-Funktionen, die zur Einrichtung dieser Verbindungen notwendig sind, werden im Skriptindex („Hilfe" > „Skriptindex" > „R-Funktionen") mit Beispielen näher beschrieben.

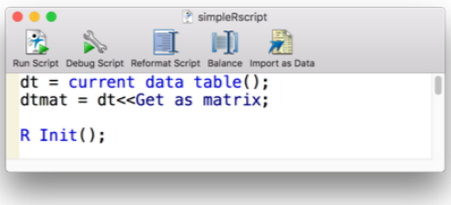
## Grundlagen der Verbindung zwischen JMP und R

Um die R-Verbindung in JMP nutzen zu können, müssen Sie zunächst Ihren R-Code mit der JMP-Skript-Sprache (JSL) umschließen.

Sehen wir uns dafür ein Beispiel an: Wir möchten Daten zur Analyse und Bearbeitung aus JMP in R übertragen, und sie dann wieder zurück nach JMP übermitteln. Hierbei handelt es sich um einen recht einfachen Prozess.
Wir schreiben dafür einen kurzen JSL-Code, um R zu öffnen, senden dann die gewünschten Daten von JMP an R, senden den R-Code an R und übermitteln dann die Daten zurück von R an JMP.

Zunächst sollten Sie sicherstellen, dass sowohl R als auch JMP auf Ihrem Gerät installiert sind[6].

Öffnen Sie anschließend ein Skriptfenster in JMP, indem Sie **„Datei" > „Neu" > „Neues Skript" auswählen**.

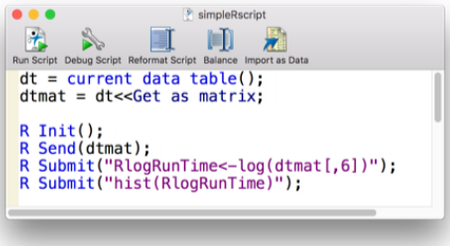



Anschließend öffnen Sie einen Beispieldatensatz in JMP. Gehen Sie dafür auf **„Hilfe" > „Beispieldatenbibliothek"** und öffnen Sie die Datentabelle **Fitness.jmp**. Um diese Tabelle nun an R zu senden, verwenden wir zwei einfache Zeilen JSL-Code. Die erste Zeile identifiziert die geöffnete Datentabelle **Fitness.jmp** und gibt ihr die neue Bezeichnung „dt". Die zweite Zeile konvertiert diese neue „dt"-Datentabelle in ein Matrixformat. Da sich die Datenstrukturierungsoptionen von R leicht von JMP unterscheiden, wurde als Matrixformat ein einfaches Framework gewählt, das sowohl mit JMP als auch mit R gut funktioniert. Jetzt rufen wir R in JMP auf. Wenn R auf demselben Gerät installiert ist, wird JMP die Software finden. Sie müssen dafür keine spezielle Verbindung einrichten oder JMP auf die R-Installation verweisen. Sie müssen lediglich die JSL-Zeile „`R Init();`" eingeben. JMP wird dann auf dem Gerät nach R suchen und die Software initialisieren (oder öffnen).

[5] cran.r-project.org.

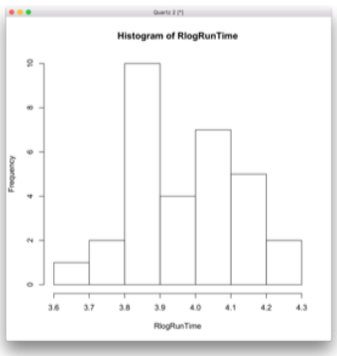
[6] Details der R-Installation und -Versionen finden Sie unter: jmp.com/support/help/en/15.1/#page/jmp/installing-r.shtml.

Nun können wir Informationen zwischen JMP und R austauschen. Die erste Zeile des folgenden Codes sendet unsere Matrixversion der Daten, mit der Bezeichnung „dtmat", an R. Die darauffolgende Zeile übermittelt den R-Code, der in R ausgeführt werden soll.



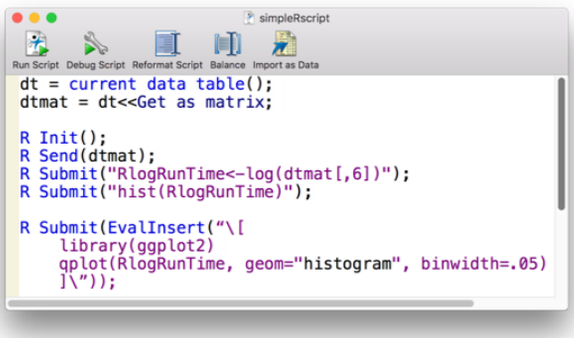

Die Zeile „**RlogRunTime<-log(dtmat[,6]);**" nimmt die sechste Spalte der Eingabedatenmatrix („Mile Run Time" aus diesem Datensatz) und wendet eine Log-Transformation an. Die Ergebnisse werden als „RlogRunTime" bezeichnet (da wir ihnen diese Bezeichnung im R-Code-Snippet zugewiesen haben).

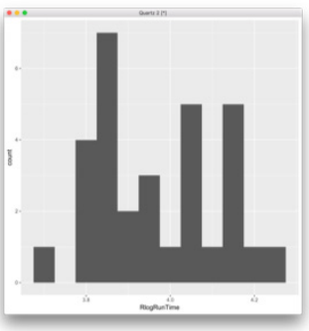
Die Zeile „**R Submit("hist(RlogRunTime)");**" generiert ein Histogramm in R, das als Pop-up auf Ihrem Gerät erscheint, wenn Sie diese Code-Zeile ausführen.



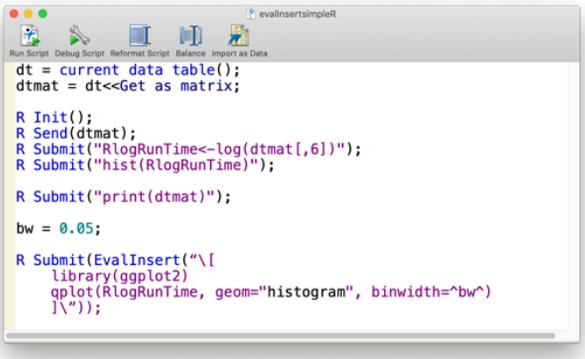

Um den Code auszuführen, wählen Sie eine oder mehrere Zeilen (jede Zeile muss mit „;" enden) und klicken Sie oben im Skriptfenster auf die Schaltfläche „Skript ausführen".


Wenn wir einen R-Code-Abschnitt mit mehreren Zeilen durch unseren JSL-Wrapper senden möchten, können wir „**R Submit(EvalInsert("\[____]\"))**" nutzen anstelle der einfachen „**R Submit("___")**"-Anweisung. In dem folgenden R-Code sehen Sie zum Beispiel, wie wir auf diese Weise das „**ggplot2**"-Paket in R nutzen, um ein Histogramm zu erstellen:

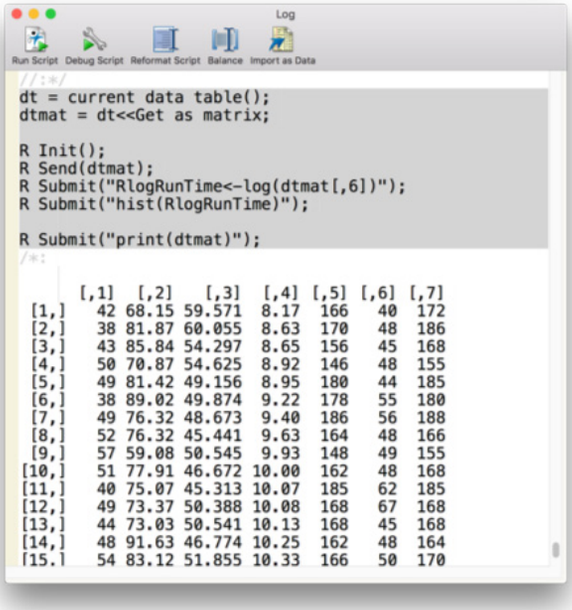


„**EvalInsert**()" ist generell auch hilfreich, wenn Sie innerhalb einer Zeichenkette in JSL Substituierungen machen oder Ausdrücke evaluieren möchten. Wir könnten beispielsweise die Klassenbreite des R-Histogramms als Variable im JSL-Skript angeben und dann diese Variable aufrufen, indem wir den „**EvalInsert**()"-Code wie folgt nutzen:

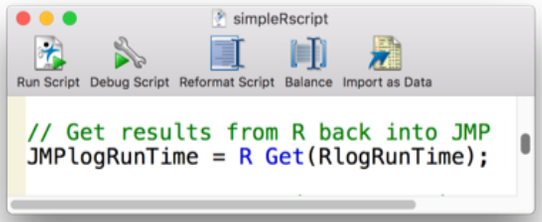



Das JMP-Log-Fenster steht Ihnen zur Verfügung, damit Sie Ihren eigenen Code überwachen und bei Bedarf Fehler darin beheben können. Um es anzuzeigen, rufen Sie die JMP-Menüleiste auf und wählen Sie „Fenster" → „Log". Hier können Sie nun jegliche interne R-Fehlermeldungen und -ergebnisse einsehen.

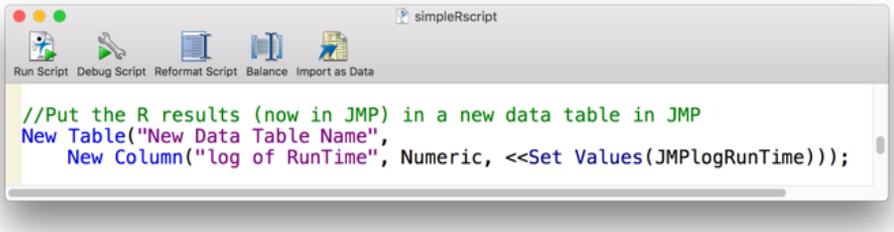


Wir können dieses Histogramm natürlich auch mithilfe der Verteilungsplattform in JMP erstellen. Die JMP-Funktionen können wir dabei auf zwei unterschiedliche Arten nutzen:

Zunächst müssen wir „**RlogRunTime**" aus R zurück an JMP übermitteln. Dieser Code-Abschnitt nimmt die transformierten Daten, die in R als „**RlogRunTime**" bezeichnet wurden, und bringt sie zurück in JMP unter der neuen Bezeichnung „**JMPlogRunTime**".



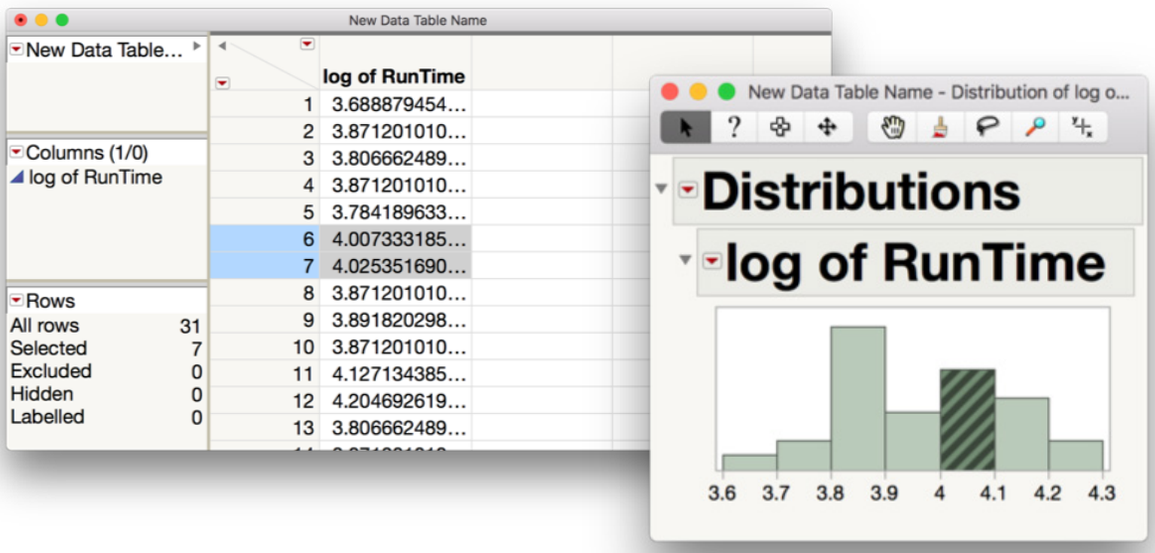

Diese neuen JMP-Daten müssen wir aus dem internen Arbeitsspeicher in JMP in eine neue Datentabelle verschieben, damit wir sie speichern und weitere Operationen darauf anwenden können. Der unten aufgeführte Code erstellt eine neue JMP-Datentabelle mit einer Spalte, welche die „**JMPlogRunTime**"-Daten enthält.




Anschließend haben wir nun also zwei unterschiedliche Möglichkeiten, um weitere JMP-Funktionen einzusetzen.

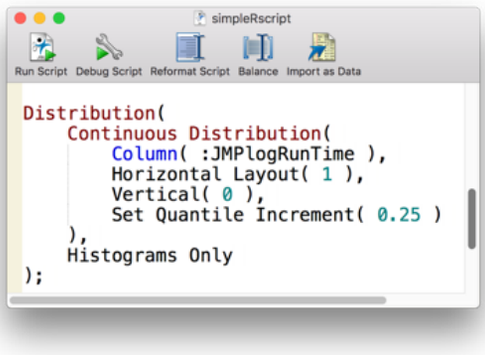
**(1)** Nutzen Sie die verschiedenen JMP-Funktionen der mausgesteuerten Menüs zur Bearbeitung der neuen JMP-Datentabelle.

In unserem Beispiel verwenden wir die Option „Analysieren" > „Verteilung", um ein interaktives Histogramm in JMP zu erstellen.



**(2)** Nutzen Sie JSL, um weiter in einem Skriptfenster zu arbeiten und die gleichen mausgesteuerten JMP-Funktionen zu nutzen.

Wenn Sie sich mit JSL bereits gut auskennen, können Sie weitere JMP-Aktionen direkt in Ihr Skriptfenster schreiben. Wenn Sie sich noch nicht so gut mit JSL auskennen, können Sie die gewünschten Aktionen einfach einmalig anhand der mausgesteuerten Aktionen durchführen und auf dieser Basis den entsprechenden Code generieren. Klicken Sie dann im Fenster mit den Ergebnissen auf das rote Dreieck-Menü und wählen Sie „Skript in Skriptfenster kopieren". Um das Skript für eine neue Datentabelle zu aktualisieren, bearbeiten Sie das JSL-Skript und wenden Sie es auf die entsprechende(n) Variable(n) an.




## JMP und R – zusätzliche einfache Beispiele

Weitere einfache Beispiele für die Nutzung von JMP und R finden Sie in der JMP-Hilfe, indem Sie dort nach „R" suchen oder indem Sie „Scripting Guide" → „JMP erweitern" → „Mit R arbeiten" auswählen.

In der JMP-Hilfe finden Sie Code für folgende Aktionen:

- Datentabellen an R senden
- Objekte in R erstellen
- R-Funktionen und -Graphiken nutzen
- Einfache Matrizen in R hinzufügen
- Bootstrapping-Verfahren für Konfidenzintervalle in R durchführen

## Fortgeschrittene Beispiele

Nachdem Sie die Grundlagen der Verwendung von JMP-Schnittstellen für R und Python kennengelernt haben, können Sie Ihr neues Wissen auf der interaktiven Oberfläche von JMP anwenden.

Im Folgenden haben wir Links zu entsprechenden Beispielen aufgeführt:

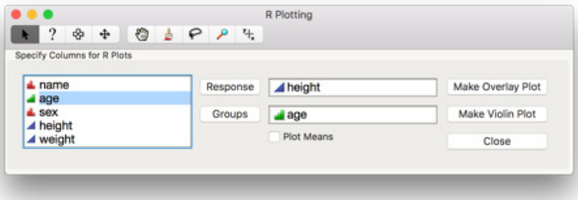
### JMP mit GGPlot – interaktiver Aufrufer

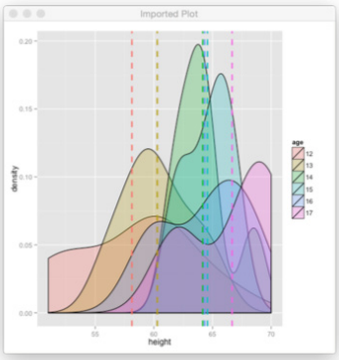
Dieses Skript nutzt die JMP-Verbindung mit R, um unter Verwendung des R-Pakets „ggplot2"[8] Dichtediagramme mit glatten Kurven zu erstellen[7], die nach den Stufen einer Gruppierungsvariablen aufgeschlüsselt sind.

[7] Bitte beachten Sie, dass diese Art der Histogramm-Ansicht ab Version 15 auch eine native Option der Graphik-Erstellungsfunktion ist. Wir verwenden R in diesem Beispiel also nur zur Veranschaulichung.

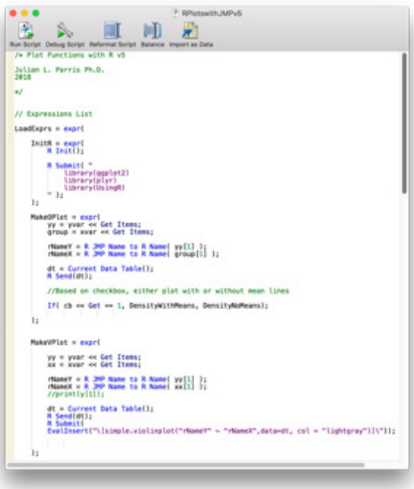
[8] Siehe cran.r-project.org/web/packages/ggplot2/index.html.

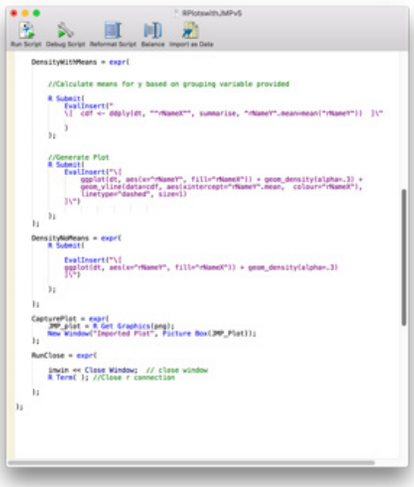
Wählen Sie zunächst eine Datendatei, öffnen Sie das Skript und führen Sie es aus. Es erscheint ein Dialogfeld, das die Spalten aus der offenen Datentabelle anzeigt. Über dieses Fenster können Sie das entsprechende R-Diagramm erstellen.

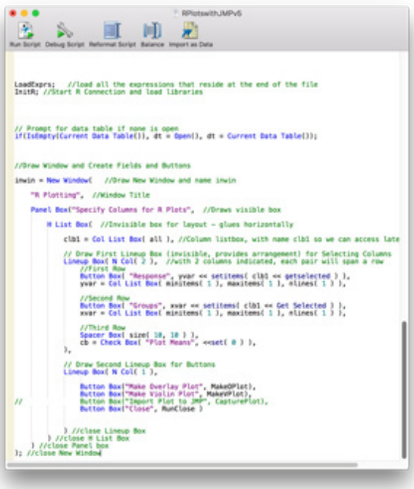



Um dieses Skript verwenden zu können, müssen Sie keinen R- oder JSL-Code schreiben. Für den Fall, dass Sie ein ähnliches, abweichendes Diagramm erstellen möchten, stellen wir Ihnen den entsprechenden JSL-Code mit R-Code-Snippets hier (unter dem Link zu diesem Beispiel) zu Verfügung.







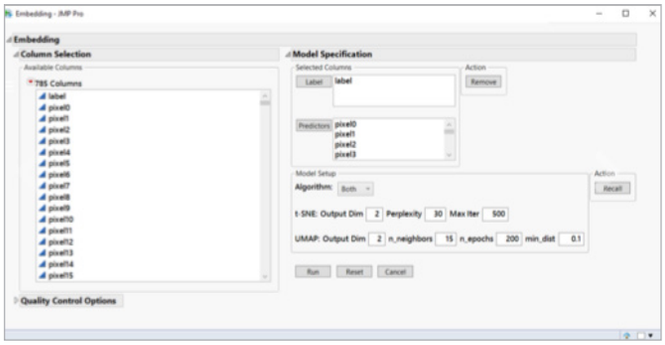
Für weitere Informationen und um dieses Add-in herunterzuladen, besuchen Sie bitte die JMP-Benutzergemeinde unter community.jmp.com/docs/DOC-6472.

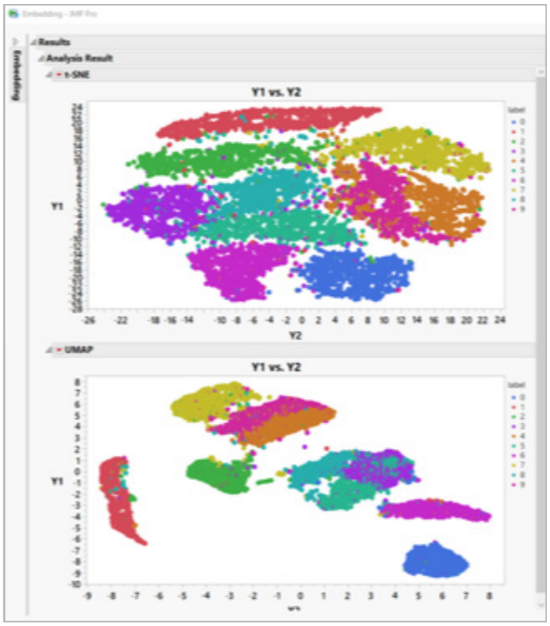
## Datenvisualisierungen mit t-SNE- und UMAP-Add-ins

Das folgende Skript, in dem wir die t-SNE- und UMAP-Pakete[9] in R verwenden, werden noch benutzerfreundlicher, wenn sie in einem JMP-Add-in bereitgestellt werden.

Bei t-Distributed Stochastic Neighbor Embedding (t-SNE) und Uniform Manifold Approximation and Projection (UMAP) handelt es sich um nichtlineare Algorithmen zur Dimensionsreduktion und Visualisierung, die in verschiedenen Anwendungsgebieten wie der Bildverarbeitung, dem Text-Mining und der Genomforschung zunehmend Verwendung finden. Dieses Add-in stellt eine benutzerfreundliche Schnittstelle für diese beiden R-Pakete bereit, damit Nutzer leicht in Datentabellen navigieren, die Datenqualität kontrollieren, dünnbesetzte Matrizen handhaben, intuitive Parametrisierungen vornehmen und Ergebnisse auf interaktive Weise interpretieren können.

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




Für die Erstellung eines solchen Add-ins muss ein Benutzer einmalig den R- oder Python-Code sowie den JSL-Code erstellen, der diesen umschließt, und den Code dann über die JMP-Add-in-Erstellungsfunktion in einem JMP-Add-in zusammenführen.[10] Die daraus resultierende .jmpaddin-Datei kann anschließend an andere Benutzer per E-Mail versendet oder im Web veröffentlicht werden.[11] Zukünftige Benutzer können das Add-in dann einfach ausführen, ohne selbst den Code schreiben, anzeigen oder bearbeiten zu müssen, indem sie einfach auf die benutzerfreundliche, mausgesteuerte Nutzeroberfläche von JMP zugreifen.

Der „unsichtbare" Code, der im Hintergrund ausgeführt wird, beginnt folgendermaßen (R-Code ist violett hervorgehoben):

```
/*
Name: Embedding

Description: This is an add-in that provide access to t-SNE and UMAP R packages.
It has enables basic quality control, sparsity handling, paprameter specification,
and result intepretations.

Author: Meijian Guan

Version: v1.2 3/14/2019

Changelog:
    v1.2: Fixed a bug for Rtsne package where too many columns would cause stack overflow.
    v1.1: Fixed a bug that could cause "Issues found in R..." error message. Fixed a bug when Both algorithms are selected and no label is selected.
    v1: Initial version

*/
Names Default To Here(1);
include("JSL_Utils.jsl");
if(hostis(windows),_slash_="\",_slash_="/");
_addinPath_=Get Path Variable("$ADDIN_HOME('com.jmp.embedding')");
If(hostis(windows),
    _addinPath_=Convert File Path(_addinPath_,windows));

//Clear Log();
//closeall(Data Tables, NoSave);
//namespace("here")<<remove(namespace("here")<<getkeys);

//label=labelY;
//label={};
//data4R=data2R;
//algrthm=algr;
//algrthm="Both";
//data2R=inDataPrep(inData,predictor,labelY);

//mtx2R=data2R<<Get as Matrix;
//mtx4R=Sparse SVD(mtx2R,20);
//svdU=mtx4R[1];
//data4R=astable(svdU,<<invisible);
//dim(data4R);
//A function to talk communicate with R: send script & data to R, get result table back.
talk2R=function({data4R,label,algrthm, dim=2, perplexity=2,iter=500, n_comp=2, n_neigb=15, n_epch=200, dist=0.1},
    {Default Local},

    labelText="";
    if(nitems(label)>=1,
        labelText=eval insert("\[
        Labels<-inDataUniq[,names(inDataUniq) %in% label]
        Labels<-as.list(Labels)
        )\");

    labelText=""
    );
    print(algrthm||" is selected!");

    Rtext=eval insert("\[

    data4R=data.frame(data4R)
    label=unlist(label)
    cat("label is: ",label,"\n")
    #print(length(label))
    #cat("\n")

    #remove duplicated observations from both datasets
    inDataUniq=data4R[!duplicated(data4R[,!names(data4R) %in% label]),] #allow excluding multiple labels
    #head(data4R)
    cat("dim of inDataUniq is: ",dim(inDataUniq),"\n")
    ^labelText^
    cat("Ready for Run","\n")

    if("^algrthm^"=="t-SNE"){
        cat("We are running t-SNE","\n")
        library("Rtsne")
        inDataTsne=data.matrix(inDataUniq[,!names(inDataUniq) %in% label])
        cat("dim of inDataTsne is: ",dim(inDataTsne),"\n")
        tsne <- Rtsne(inDataTsne, dims = ^dim^, perplexity=^perplexity^, verbose=TRUE, max_iter = ^iter^,pca=F)
        outputY=tsne$Y
        head(outputY)
    }else if("^algrthm^"=="UMAP"){
        cat("We are running UMAP","\n")
```

[10] Weitere Informationen dazu, wie Sie ein JMP-Add-in auf Basis Ihres JSL-Skripts erstellen können, finden Sie auf jmp.com/content/dam/jmp/documents/en/academic/learning-library/01-add-in-builder.pdf.

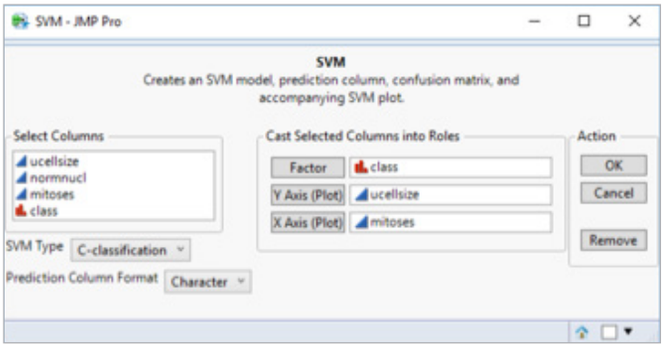
[11] Veröffentlichen Sie die Datei in der JMP-Community unter community.jmp.com, damit auch andere Anwender darauf zugreifen können.

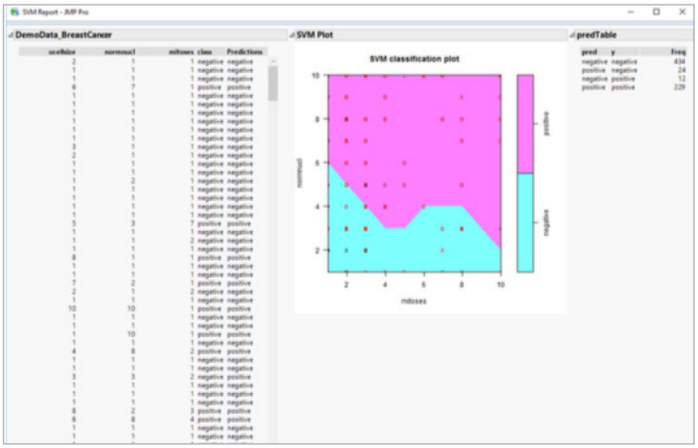
Nachdem der Code also einmal geschrieben wurde, kann er in einem Add-in verpackt werden, sodass zukünftige Benutzer nicht direkt damit interagieren müssen. Dadurch haben Sie eine mausgesteuerte Schnittstelle erstellt, über die Sie bestimmte R- oder Python-Funktionen aufrufen können.

Für weitere Informationen oder um dieses Add-in herunterzuladen, besuchen Sie die JMP-Benutzergemeinde unter community.jmp.com/t5/JMP-Add-Ins/Data-visualization-with-t-SNE-and-UMAP/tap/177969.

## SVM-Add-in

Dieses Add-in ruft die svm-Funktion aus dem e1071-R-Paket auf.[12] Wenn Benutzer das SVM-Add-in nutzen möchten, interagieren sie mit dem folgenden Dialogfeld, um den entsprechenden SVM-Klassifikationsbericht zu erhalten[13]:




Auch in diesem Fall sehen Benutzer nicht den im Hintergrund ausgeführten Code:

```
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)")))
```

Die Erstellung dieses Add-ins ist sogar noch leichter, wenn Sie die R-Add-in-Erstellungsfunktion in JMP nutzen (nächstes Beispiel). Die Add-in-Erstellungsfunktion ermöglicht Ihnen, die JSL-Kodierung komplett zu umgehen, indem Sie einfach nur mit den Dialogfeldern interagieren und dort den R- oder Python-Code eingeben.

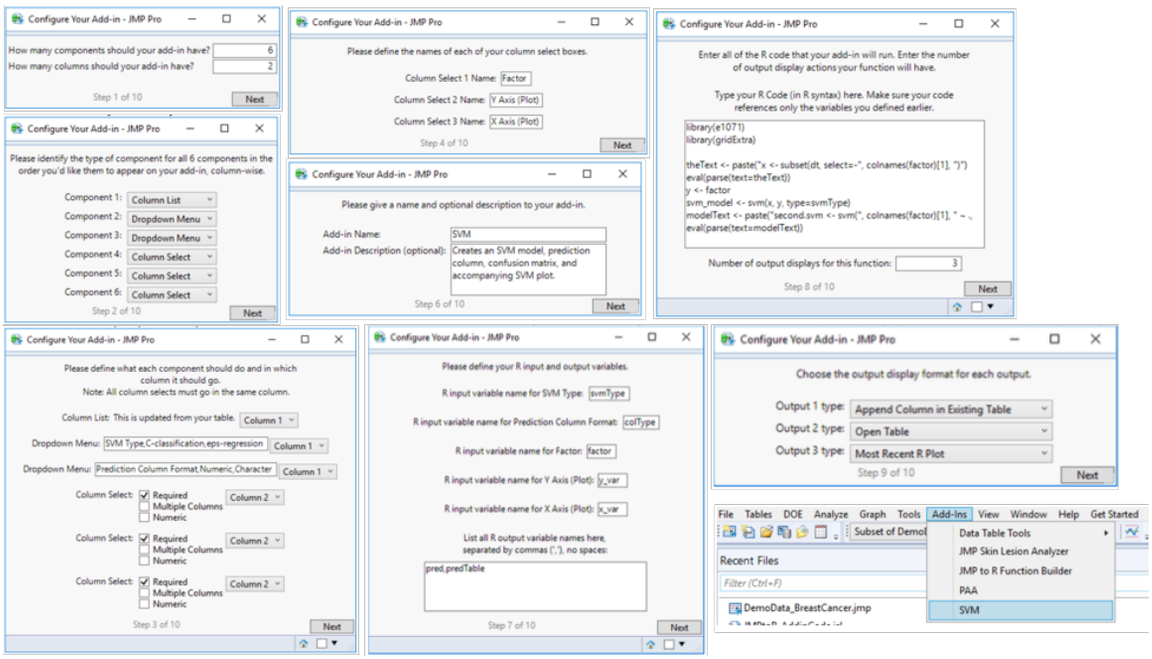
[12] Siehe cran.r-project.org/web/packages/e1071/index.html.

[13] SVM wurde in Version 15 von JMP Pro zu den Vorhersagemodellen hinzugefügt. In diesem Beispiel wollten wir also wiederum nur zeigen, wie Sie die Verbindung mit R effektiv nutzen können (doch für Benutzer von JMP Pro 15 ist dieses Vorgehen nicht mehr erforderlich).

## Die R-Add-in-Erstellungsfunktion in JMP: Verwendung der JMP/R-Verbindung ohne JSL

Mit diesem Werkzeug können Benutzer benutzerdefinierte JMP-Add-ins zur Ausführung von R-Funktionen definieren und anwenden. Um diese Funktion zur Schnittstellen-Erstellung zu verwenden, müssen Sie zunächst den R-Code schreiben, den Sie an R senden möchten. Sie müssen dafür jedoch KEINEN JSL-Code schreiben. Die Endnutzer Ihrer neuen mausgesteuerten Schnittstelle wiederum werden diesen R-Code weder sehen noch aktiv nutzen – sie sehen lediglich die benutzerfreundliche, mausgesteuerte Schnittstelle. (Hinweis: Dieses Add-in funktioniert nur auf Windows, nicht auf macOS).

Wenn Sie dieses Add-in nutzen, müssen Sie bei der Verwendung der JMP/R-Verbindung keinen JSL-Code einsetzen. Stattdessen interagieren Sie einfach nur mit ein paar Dialogfenstern, um die Felder, Schaltflächen und Abfragen zu erstellen, die Sie in Ihr Add-in einfügen möchten. Die meisten dieser Dialogfenster sind im Folgenden aufgeführt:




Das obige Beispiel eines SVM-Add-ins wurde mithilfe der R-Add-in-Erstellungsfunktion in JMP konfiguriert.

Für weitere Informationen oder um dieses Add-in für die R-Add-in-Erstellungsfunktion in JMP herunterzuladen, besuchen Sie bitte unsere JMP-Benutzergemeinde unter community.jmp.com/t5/JMP-Add-Ins/The-JMP-to-R-Add-In-Builder/ta-p/43879.

# Zusammenfassung

In diesem Leitfaden haben wir Ihnen die Generierung von Python-Scoring-Code sowie die Erstellung von Python- und R-Skripten in JMP nähergebracht. Darüber hinaus haben wir Ihnen einige weiterführende Beispiele bereitgestellt, darunter fortgeschrittene Beispiele für R, für die Sie die Add-in-Erstellungsfunktion in JMP nutzen können. Weitere Informationen zu diesen Themen finden Sie in den in diesem White Paper angegebenen Fußnoten und Links. Falls Sie Fragen haben, Hilfe bei der Implementierung dieser Techniken benötigen oder einfach nur eigene Ideen teilen möchten, dann veröffentlichen Sie doch einfach einen Beitrag in unserer JMP-Community unter community.jmp.com.

## Über SAS und JMP

SAS, der weltweit führende Anbieter von Analytics-Lösungen, entwickelte JMP (ausgesprochen „Jump") im Jahre 1989, um es Wissenschaftlern, Entwicklern und anderen Datenanalysten zu ermöglichen, Daten auf visuelle und interaktive Weise zu erkunden und zu analysieren. Seitdem ist JMP von einem Einzelprodukt zu einer Familie von Softwareprodukten für die statistische Datenanalyse gewachsen, die jeweils auf die spezifischen Anforderungen der Anwender zugeschnitten sind. SAS-Mitbegründer und Executive Vice President John Sall ist Leiter des Unternehmensbereichs JMP.


SAS Institute GmbH In der Neckarhelle 162 69118 Heidelberg Deutschland Tel. +49 6221 415-123, Fax +49 6221 415-145

JMP ist eine Softwarelösung von SAS. Um mehr über SAS zu erfahren, besuchen Sie bitte **sas.com.**

Den JMP-Vertrieb in den USA und Kanada erreichen Sie unter +1 877 594 6567 oder informieren Sie sich auf der Website **jmp.com**