IBM SPSS® Modeler, a powerful analytic tool, supports all phases of a data mining process, including data preparation, model building, deployment, and model maintenance. The IBM SPSS Modeler UI makes use of a visual data mining workbench that provides built-in data preparation, modeling, and output nodes that enable rapid development of the analytic assets.

Visual programming in IBM SPSS Modeler is based on icons called SPSS Modeler nodes. The user creates a process that runs data through a series of nodes called streams. The nodes represent operations to be performed on the data, and the links between the nodes indicate the direction of data flow. Typically, you use a data stream to read data from a collection of data sources, you manipulate the data, and you send the data to a destination, which can be a table or a viewer.

The nodes can be linked to form a stream. The stream represents a flow of data through a number of operations to a destination. This user-friendly interface makes it possible to analyze complex data sets and create powerful predictive models without programming.

**Extend function for IBM SPSS Modeler with new nodes**

To add functions not included in SPSS Modeler, you can create nodes. The more nodes you have in SPSS Modeler, the more operations you can perform. Use the Component-Level Extension Framework, a mechanism that enables you to add user-provided extensions to the standard functions of SPSS Modeler.

Because SPSS Modeler 16 is integrated with the R programming language, you can now run R scripts. This article describes how to develop new extensions using R. Figure 1 shows three new nodes: R Transform, R (for modeling), and R Output. You can insert your R script directly into these nodes in SPSS Modeler.
What you'll need to get started
To run R code in SPSS Modeler, you need to download and install R 2.15.2 and the IBM SPSS Modeler Essentials for R Plugin.

SPSS Modeler is not intended to serve as a workbench to write the R code; rather, SPSS Modeler is only able to run R code. Use a workbench such as RStudio to write and test R code before porting it to IBM SPSS Modeler.

To learn how to configure IBM SPSS Modeler 16 and the R integration, see the developerWorks article "Calling R from SPSS."

R programming language and environment
R is a powerful open source statistical language and environment, which offers a rich analytic ecosystem for data exploration, visualization, statistical analysis, modeling, machine learning, simulations and more. It is an emerging competitor to proprietary platforms.

R is taught at universities; many statistics and math courses now use R. IT shops are adopting R, and many companies are integrating it into their products. A vibrant, active user community is growing around R and jobs, demand for R skills are on the rise, and the number of R packages are increasing. Also, emerging algorithms usually appear in the freely available R programming language, before they are available in commercial packages.

R does require the developer to overcome a learning curve. Before adopting R, organizations must consider the level of support offered because open source packages have varying levels of quality. R is based on an in-memory architecture.

The ability to create nodes for SPSS Modeler enables you to use the more than 5,500 R packages available to analysts, even if you don't have R programming skills, and even if you do not write any code.

Custom Dialog Builder
The Custom Dialog Builder (first available in IBM SPSS Statistics) enables you to create and manage nodes to be used in SPSS Modeler 16. To open the Custom Dialog Builder (shown in Figure 2), click Tools > Custom Dialog Builder for R in the main menu, which shows the following elements:

- Dialog canvas— Area where you design the layout of the node dialog.
- Properties pane— List of properties that make up the node dialog and the properties of the dialog, such as the node type.
- Tools palette— Set of controls that can be included in a custom node dialog.
Example 1. Create a geocoding node as your first node

As an example of how to create a new node, develop a node to perform geocoding of geographical data. Geocoding is the process to find geographic coordinates from other geographic data, such as street addresses, ZIP codes, and similar data. Many Internet companies — such as OpenStreetMap, Mapquest, Bing, and Google Maps — provide geocoding services. This article describes how to build a node based on the Google Maps API.

Step 1. Get and test the R code for geocoding

To create a new node, get the appropriate R code for the geocoding task. The R code can be written by any user with R skills or can be downloaded from the Internet. This article relies on the R code in Listing 1 from the article "Using Google maps API and R." Before you use this code in SPSS Modeler, test the code using RStudio or the R console.

1. Open Rstudio and run the code from Listing 1.
   - Create a variable with an address in a string, such as `address<-geoCode("The White House, Washington, DC").`
   - Run the commands `address[1]` to get the latitude and `address[2]` to get the longitude. The results should be "38.8976831" and "-77.0364972."

**Listing 1. R code performing geocoding using Google Maps API**

```r
#### This script uses RCurl and RJSONIO to download data from Google's API:
```
### Latitude, longitude, location type (see explanation at the end), formatted address
### Notice there is a limit of 2,500 calls per day

```r
library(RCurl)
l library(RJSONIO)
l library(plyr)
l
url <- function(address, return.call = "json", sensor = "false") {
  root <- "http://maps.google.com/maps/api/geocode/"
  u <- paste(root, return.call, "?address=", address, "&sensor=", sensor, sep = "")
  return(URLencode(u))
}

go GeoCode <- function(address, verbose=FALSE) {
  if(verbose) cat(address, "\n")
  u <- url(address)
  doc <- getURL(u)
  x <- fromJSON(doc,simplify = FALSE)
  if(x$status=="OK") {
    lat <- x$results[[1]]$geometry$location$lat
    lng <- x$results[[1]]$geometry$location$lng
    location_type <- x$results[[1]]$geometry$location_type
    formatted_address <- x$results[[1]]$formatted_address
    return(c(lat, lng, location_type, formatted_address))
  } else {
    return(c(NA,NA,NA, NA))
  }
}

## Test with a single address
#address <- geoCode("The White House, Washington, DC")
#address
#[1] "38.8976831"
#[2] "-77.0364972"
#[3] "APPROXIMATE"
#[4] "The White House, 1600 Pennsylvania Avenue Northwest, Washington, D.C., DC 20500, USA"
```

2. To verify that the coordinates are correct, point your browser to [Google Maps](https://maps.google.com) and paste the coordinates in the search box, as shown Figure 3. In this case, the geocoding worked well, and the coordinates point to the White House in Washington.
When you run the code in Listing 1, the following tasks are performed:

1. **The URL to the web service is created.** Because this function uses a web service, the first action is to create the URL to call the service. The address passed as an argument is written with spaces, but the spaces are removed to create a URL in the specific format required by the Google Maps API. The result can be in XML or JSON. In this example, the JSON output is used.

2. **The connection is created** and the JSON is received.

3. **The JSON is parsed to get the latitude and the longitude** of the address given between others.

Third-party APIs, such as the Google Maps API, can have usage limits. Free web services do provide geocoding, and the code has only to be adapted to use them. Data provided by free services might be less accurate, however.
Step 2. Create the new node on IBM SPSS Modeler 16

Now that the code has been tested, the next step is to develop the new node.

1. Open IBM SPSS Modeler 16 and open the Custom Dialog Builder.
2. In the properties of the node, put the same parameters as in Figure 4. For Node Icon, download the image geocode_icon.gif from the sample code for this article and use it as an icon for the new node you are creating.

**Figure 4. Properties to set in the Custom Dialog Builder**

3. From the tools, drag and drop Field Chooser, double-click Field Chooser, change identifier to address, and change Title to Address field.
4. Click Edit > Script Template and paste in the R code of Listing 2.

The node is almost created, but you must make some modifications in the R code to map the data coming from the SPSS Modeler stream. See the documentation on "Allowable Syntax" for SPSS Modeler. It describes the statements and functions recognized by R. For help, see the documentation for the script template, under Examples.

Specify control identifiers in the form %%Identifier%% at the appropriate location. Press Ctrl + Spacebar to show a list of available control identifiers. In this case, the value is %address%%.

```
location <- modelerData$%%address%%
```

The R object modelerData is a data frame that contains the original data. To add the new columns for latitude and longitude, use the `cbind` function to create a data frame with the original data plus the output generated, as shown.

```
modelerData<-cbind(modelerData,lat)
```

The R object var1 sets up a new field in SPSS Modeler for the data model that describes the type and structure of the new data generated. The name of the new field and the type of storage are specified in this new field.

```
var1<-c(fieldName="Latitude",fieldLabel="",fieldStorage="real",fieldFormat="",fieldMeasure="", fieldRole="")
```
The R object `modelerDataModel` contains the data model for the original data with the extra field generated. The extra field is called `Latitude` and it has characteristics specified.

```r
modelerDataModel <- data.frame(modelerDataModel, var1)
```

When you are working on your R code in SPSS Modeler, you can perform limited debugging of the R script by using commands including `print()` and `str()`.

Figure 5 shows the script template and the node at the end of development.

**Figure 5. Properties to set in the Custom Dialog Builder**

The following code listing shows the required modifications.

**Listing 2. R code with modifications for geocoding**

```r
library(RCurl)
library(RJSONIO)
location <- modelerData$address
print(location)
root <- "http://maps.google.com/maps/api/geocode/
u <- paste(root,"json", "?address=", location, "&sensor=", "false", sep = "")
u <- gsub(' ','%20',u) #Encode URL Parameters
print(u)
require("plyr")
doc <- aaply(u,1,getURL)
json <- alply(doc,1,fromJSON,simplify = FALSE)
coord = laply(json,function(x) {
  if(x$status=="OK") {
    lat <- x$results[[1]]$geometry$location$lat
    lng <- x$results[[1]]$geometry$location$lng
    return(c(lat,lng))
  } else {
    return(c(NA,NA))
  }
```

Create new nodes for IBM SPSS Modeler 16 using R
Step 3. Save and install the new node

After the development is finished, save the node to distribute it to colleagues or to other SPSS Modeler users. Click File > Save in the Custom Dialog Builder. The file is saved with the extension .cfd. Install the new node. Click File > Install. Close the Custom Dialog Builder. The new node is in the Record Ops palette as specified in the properties.

Step 4. Test the geocoding node

Test the node by generating manual data using the User Input node:

1. Click the Sources palette and drag and drop the User Input node into the canvas.
2. Double-click the User Input node, create a new field called Location, and select String as storage. Specify addresses as values. In this example, the following addresses are used: "New York City" "San Francisco, California" "Paris, France".

1. Click Preview to visualize the generated data in a table.

   **Figure 6. Data manually generated with the User Input node**

2. Select the new Geocoding node just created from the Field Ops palette. Connect the User Input node to the Geocoding node and a Table node as output.

   **Figure 7. Stream to test new geocoding node**

3. Double-click the Geocoding node. In the Address field, select Location.
4. Run the stream. The expected output is a table with three columns: location, latitude, and longitude.
The first new node using R is now created and working.

**Example 2. Create a visualization node for geospatial data**

In Example 1, some addresses have been converted into coordinates. The output is a table that confirms that the code is running properly. However, there is no direct visualization to check that these coordinates are correct. The best way to visualize geospatial data is by using a map. This second example shows how to create a node that plot dots in a dynamic map that is run in a web browser. Because the map uses the Google Maps API, the name of the new node is the Google Maps node.

To create this node, use the R code described in "Mine spatial data with space-time-boxes in IBM SPSS Modeler and visualize the data with R." This R code uses the R package plotGoogleMaps, which provides an interactive plot for handling the geographic data within web browsers.

**Step 1. Test and understand the code in RStudio**

Test the code before you use SPSS Modeler. With Rstudio, you can easily modify and debug the code if it's not working.

1. Open Rstudio and create the latitude and longitude variables as shown in Listing 3. For example, the following values are for the White House in Washington, D.C.

   **Listing 3. Longitude and latitude variables**
   ```r
   latitude<- 38.8976831
   longitude<- -77.0364972
   df=data.frame(latitude,longitude)
   ```

2. Run the code in Listing 4. A browser opens to a page with the map and the dot is in the correct place.
Listing 4. Example of space-time-box geohash

```r
install.packages('plotGoogleMaps')
library(plotGoogleMaps)
coordinates(df)<-~longitude+latitude # convert to SPDF
proj4string(df) <- CRS('+init=epsg:4326')
# adding Coordinate Referent Sys.
# Create web map of Point data
m<-plotGoogleMaps(df,filename='C:/MyNewMap.htm')
```

The code gets the coordinates and transforms them to the coordinate system that the Google Maps API understands by using the R function `proj4string`. The function `plotGoogleMaps`, which is included in the plotGoogleMaps package, is run. In this example, the output is stored on the C drive with the name of MyNewMap.htm.

**Step 2. Create the Google Maps node in SPSS Modeler 16**

Create the node by using the Custom Dialog Builder, then map the data:

1. Open the Custom Dialog Builder and select the properties shown in Figure 9. Download the icon file GoogleMaps.gif from the sample code for this article. Note that now the Node Type is Output.

**Figure 9. Properties for Google Maps node**

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog Name</td>
<td>GoogleMaps</td>
</tr>
<tr>
<td>Title</td>
<td>GoogleMaps</td>
</tr>
<tr>
<td>Help File</td>
<td></td>
</tr>
<tr>
<td>Script Type</td>
<td>R</td>
</tr>
<tr>
<td>Required Add-Ons</td>
<td>Integration Plug-in for R</td>
</tr>
<tr>
<td>R Script</td>
<td>library(plotGoogleMaps)inLongitude=modelerData$%lon%\nlatitude=modelerData$\nLabel...</td>
</tr>
<tr>
<td>Score from Model</td>
<td>False</td>
</tr>
<tr>
<td>R Scoring Script</td>
<td></td>
</tr>
<tr>
<td>Modeler Properties</td>
<td></td>
</tr>
<tr>
<td>Node Type</td>
<td>Output</td>
</tr>
<tr>
<td>Palette</td>
<td>Output</td>
</tr>
<tr>
<td>Node Icon</td>
<td>C:\Users\Administrator\Desktop\googleMaps.gif</td>
</tr>
</tbody>
</table>

2. Select two Field Chooser elements and put them in the canvas; one is for the latitude and the other is for the longitude. The result looks similar to Figure 10.
3. Click **Edit > Script** and add the code from Listing 5. As explained, adapt the R code in **Listing 4** to SPSS Modeler.

**Listing 5. R script to insert in the Custom Dialog Builder**

```r
library(plotGoogleMaps)
Longitude=modelerData$%%lat%%
Latitude=modelerData$%%lon%%
coordinates(modelerData)<-~Longitude+Latitude
proj4string(modelerData) <- CRS('+init=epsg:4326')
m<-plotGoogleMaps(modelerData,filename='C:/myNewMap.htm')
```

Save and install the node.

**Step 3. Test the node**

To test the node, plot the data coming from the data previously created. Select the Google Maps from the Output palette and connect it to the Geocoding node.

**Figure 11. Final stream with the new Google Maps node as an output**

Run the stream. The browser opens a page with a dynamic map, which plots the addresses generated.
Dependencies to be aware of

The nodes created in this article use web services and maps that require an Internet connection. Before the nodes can be used, the libraries used by the nodes must be installed. In the two nodes created in this article, the R packages plotGoogleMaps, Rcurl, RJSONIO, and plyr must be installed. To install a package, build a stream as shown in Figure 13.

Figure 13. Stream used to install R packages in SPSS Modeler 16
Double-click the R node, insert the code in Listing 6, and run the stream, which downloads and installs the library. This task must be performed only once. You can also run this code directly in the R console or RStudio.

**Listing 6. Example of space-time-box geohash**

```r
install.packages("Rcurl")
install.packages("RJSONIO")
install.packages("plyr")
install.packages("plotGoogleMaps")
```

You can create extensions using CLEF. Code development using CLEF is more complex than using R, but with CLEF, you can ensure that extension modules can look and act the same as native SPSS Modeler modules, and perform tasks at the same or similar speed and efficiency as native nodes.

**Conclusion**

This article describes how to use the integration of SPSS Modeler and R to create new nodes. Users who do not have R skills can use all of the powerful R packages, available for free. The R functions in SPSS Modeler are extended with technologies including SQL pushback to run the code faster. Features not originally included in SPSS Modeler can be added. An additional benefit is that nodes can be shared with the user community to make analytics technology available to non-experts.

To share the nodes in your organization, use the repository feature included in SPSS Collaboration and Deployment services. A public repository is available from the IBM SPSS DevCentral developerWorks community.
## Downloads

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample code</td>
<td>files.zip</td>
<td>66KB</td>
</tr>
</tbody>
</table>
Resources

Learn

• : Learn more about SPSS software.
• Check out SPSS Modeler to explore the differences between Modeler Professional and Server editions.
• Visit developerWorks Business analytics for more technical analytics resources for developers.
• The Comprehensive R Archive Network is the main site for the R project and each R package. The help pages and manuals that are associated with optimx, nlmrt, and Rcgmin are detailed. Numerous references are provided.
• Read "Do I need to learn R?" (Catherine Dalzell, developerWorks, September 2013) to learn why R is a valuable tool for data analytics that was expressly designed to reflect the way that statisticians think and work.
• "Calling R from SPSS" describes how to use R code inside IBM SPSS Modeler 16.
• Read "Using Google maps API" to discover how to use Google Maps API with R.

Get products and technologies

• Download the R plug-in for SPSS plugin.
• Download the R 2.15.2 for Windows package.

Discuss

• Visit the IBM SPSS DevCentral developerWorks community to share tips and experiences with other IBM SPSS developers.
• Follow developerWorks on Twitter to be among the first to hear about new resources.
About the author

Armand Ruiz Gabernet

Armand Ruiz is an IT Specialist staff member at IBM since 2011 when he was a university student. He participated in the IBM innovation program, Extreme Blue, for the most promising students from universities around the world while in Brussels (Belgium). Afterward, Armand did his Master Thesis at IBM La Gaude (France). He currently teaches business partners and ISVs the new solutions of IBM for big data and analytics around Europe, the Middle East, and Africa. Armand holds a degree of Electrical engineering at Universitat Politecnica de Catalunya (Spain) and Université Catholique de Louvain (Belgium). He also holds a Master of Business Intelligence at Universitat Oberta de Catalunya (Spain) and is doing a Professional Certification in 'Mining Massive DataSets' at Stanford University (US) for the past two years.

© Copyright IBM Corporation 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)