WUSS Paper 124-2022

Map It Out: Using SG Attribute Maps for Precise Control of PROC SGPLOT Output

Joshua M. Horstman, Nested Loop Consulting

ABSTRACT

The SGPLOT procedure, part of the ODS Statistical Graphics package, allows for extensive customization of nearly all aspects of plot output. These capabilities are commonly used to distinguish between groups or categories being compared through the use of distinct plot attributes, such as symbols and colors. However, there are times when it is advantageous to be able to associate specific plot attributes with specific data values. SG attribute maps provide functionality that does exactly that. This presentation will provide an introduction to the use of SG attribute maps in conjunction with PROC SGPLOT. A series of examples will demonstrate how attribute maps are used and why they are useful as a programming tool. Both discrete and range attribute maps will be used to modify a variety of plot attributes, such as plot marker symbols and colors, line styles and fill patterns.

INTRODUCTION

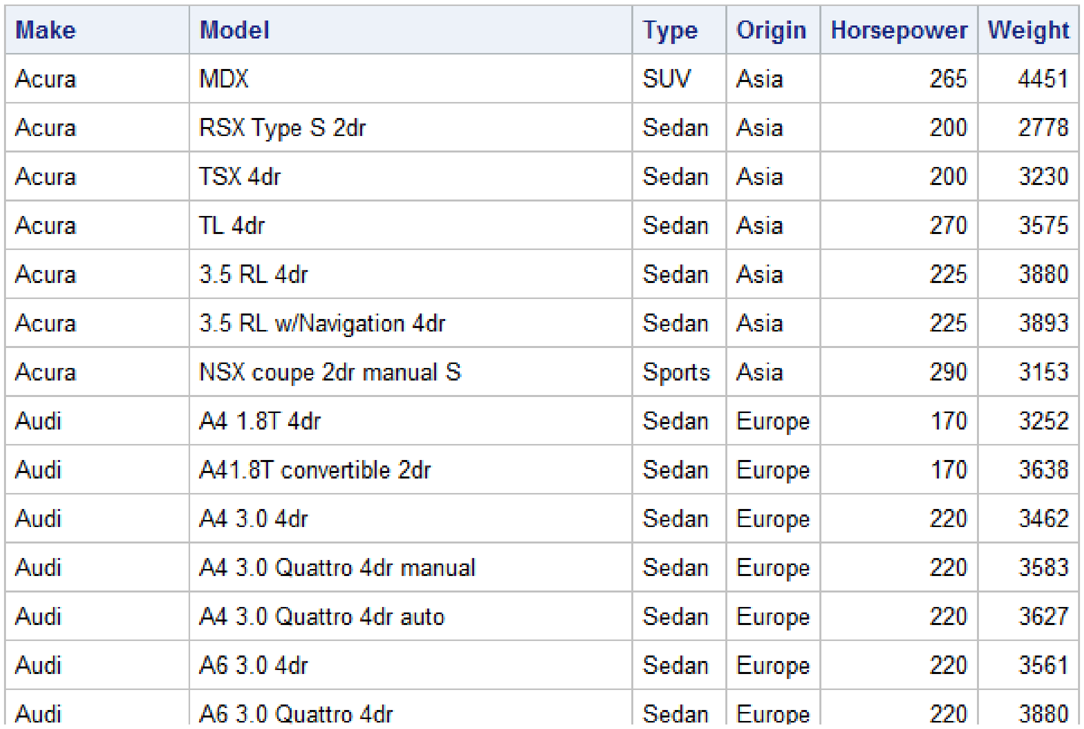
The ODS Statistical Graphics package is a highly flexible graphics suite that was added to SAS® software in version 9.2 and has been greatly enhanced in more recent versions. This paper discusses one feature of the package, SG attribute maps. Attribute maps allow us to associate specific visual plot attributes with specific data values or ranges of values.

In this paper, we begin with a simple scatter plot that does not use an attribute map. In the second example, we create a scatter plot of a subset of our data to motivate the need for attribute maps. Finally, we see examples demonstrating the use of both types of attribute maps – discrete attribute maps and range attribute maps.

THE DATA

The examples in this paper are based on the SASHELP.CARS data set. The SASHELP library is included with most SAS installations and contains dozens of useful sample data sets. The SASHELP.CARS data set contains 428 records, each pertaining to a particular vehicle model.

The CARS data set has 15 columns containing various information such as the vehicle make and model as well as pertinent details such as the engine size, horsepower, and weight. A classification variable called ORIGIN has a value of Asia, Europe, or USA. Another classification variable named TYPE which tells whether the vehicle is a sedan, truck, wagon, sports car, SUV, or hybrid. A portion of the data set is shown in Figure 1.



Make	Model	Type	Origin	Horsepower	Weight
Acura	MDX	SUV	Asia	265	4451
Acura	RSX Type S 2dr	Sedan	Asia	200	2778
Acura	TSX 4dr	Sedan	Asia	200	3230
Acura	TL 4dr	Sedan	Asia	270	3575
Acura	3.5 RL 4dr	Sedan	Asia	225	3880
Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	225	3893
Acura	NSX coupe 2dr manual S	Sports	Asia	290	3153
Audi	A4 1.8T 4dr	Sedan	Europe	170	3252
Audi	A41.8T convertible 2dr	Sedan	Europe	170	3638
Audi	A4 3.0 4dr	Sedan	Europe	220	3462
Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	220	3583
Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	220	3627
Audi	A6 3.0 4dr	Sedan	Europe	220	3561
Audi	A6 3.0 Quattro 4dr	Sedan	Europe	220	3880

Figure 1. Portion of SASHELP.CARS Data Set

EXAMPLE 1: SIMPLE SCATTER PLOT – NO ATTRIBUTE MAP

We begin with a simple scatter plot. This example does not use an attribute map but provides a foundation upon which later examples will build. This scatter plot will have horsepower on the Y axis and weight on the X axis. Additionally, the vehicles are grouped by vehicle origin (Asia, Europe, or USA) so that each vehicle origin uses a different plot marker color and symbol.

```
ods graphics / attrpriority=none;

proc sgplot data=sashelp.cars;
  title "Horsepower vs. Weight";
  scatter y=horsepower x=weight / group=origin markerattrs=(size=10px);
  styleattrs
    datacontrastcolors=(IndianRed ForestGreen Navy)
    datasymbols=(CircleFilled SquareFilled TriangleFilled);
run;
```

Note the use of ATTRPRIORITY=NONE on the ODS GRAPHICS statement. This option specifies the way in which attributes are applied to group values when we are varying more than one attribute. In this case, we are using the STYLEATTRS statement to apply both marker colors and marker symbols based on a group variable. Setting ATTRPRIORITY to NONE causes SGPLOT to cycle through both lists of attributes in a pairwise fashion. That is, the first group is assigned the first color and the first symbol, the second group is assigned the second color and the second symbol, and so on.

Also, notice that the marker size (10 pixels) is specified using the MARKERATTRS option on the SCATTER statement, not the STYLEATTRS statement. This attribute will apply to all markers and will not vary by group.

Figure 2 shows the resulting plot. Note that vehicles originating in Asia are represented by red circles, those from Europe appear as green squares, and those from the USA are rendered as blue triangles.

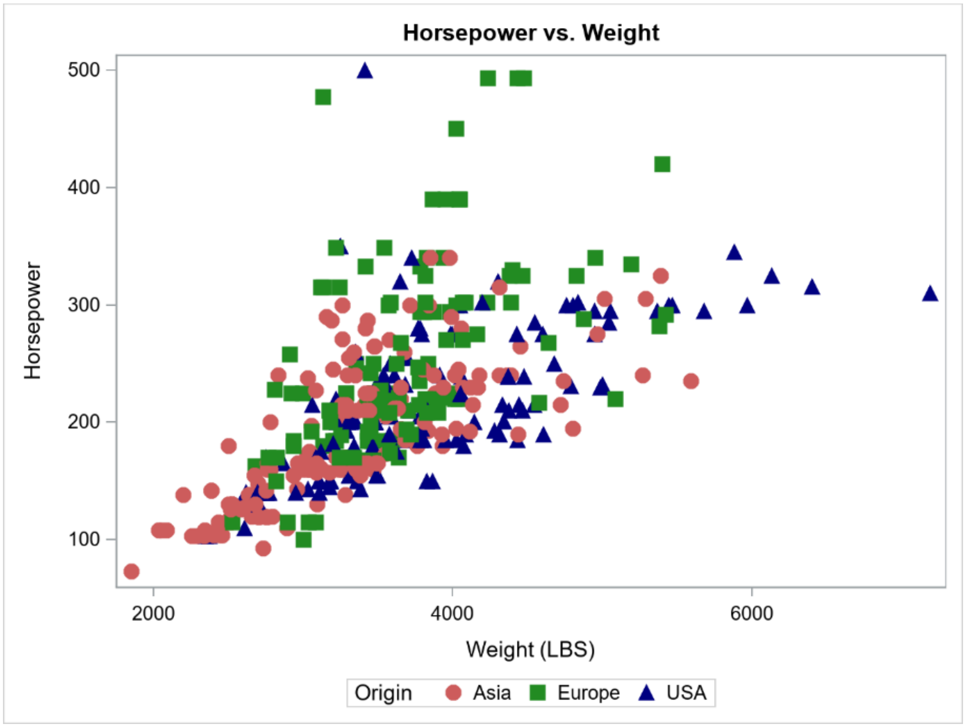


Figure 2. PROC SGPLOT Output for Example 1

EXAMPLE 2: SUBGROUP PLOT – NO ATTRIBUTE MAP

The second example is very similar to the first except that only a subset of the data is plotted. Specifically, Example 2 includes only those records in SASHELP.CARS corresponding to trucks (TYPE="TRUCK").

```
proc sgplot data=sashelp.cars;
  title "Horsepower vs. Weight - Trucks Only";
  where type='Truck';
  scatter y=horsepower x=weight / group=origin markerattrs=(size=10px);
  styleattrs
    datacontrastcolors=(IndianRed ForestGreen Navy)
    datasymbols=(CircleFilled SquareFilled TriangleFilled);
run;
```

Figure 3 shows the output for this subgroup plot. Observe that there are no vehicles of European origin in this subset. As a result, the assignment of plot attributes has changed so that vehicles from the USA are now represented by red circles while those of Asian origin are now green squares. For consistency across multiple plots, we would like to keep the original mapping of attributes as seen in Example 1 (Figure 2). In our next example, we will see how to use an attribute map to accomplish this.

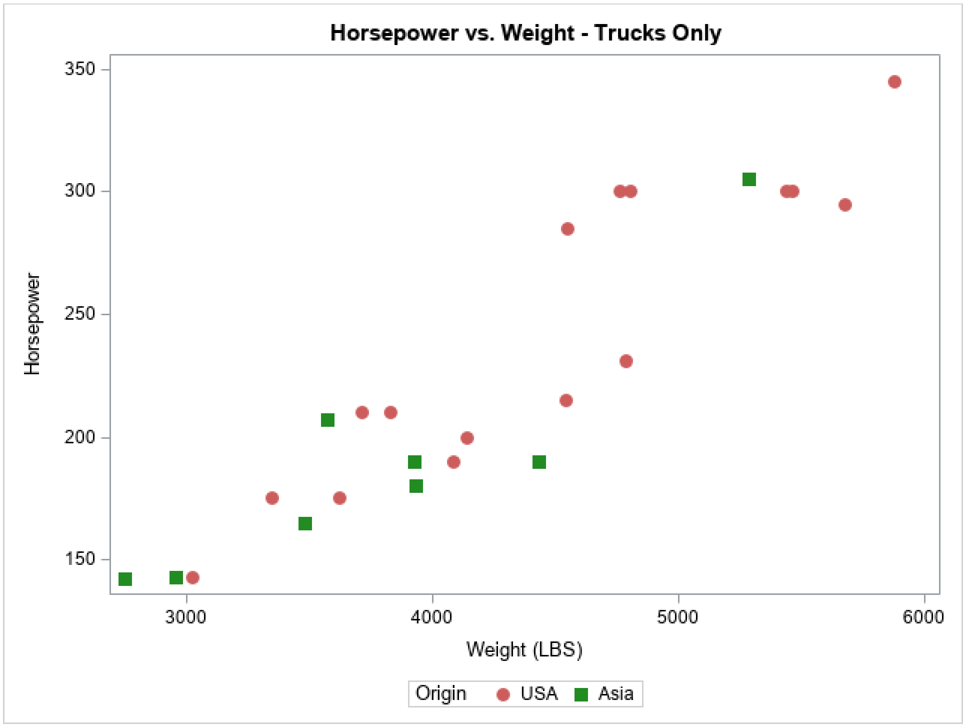


Figure 3. SGPLOT Output for Example 2

EXAMPLE 3: USING A DISCRETE ATTRIBUTE MAP

In our third example, we use an attribute map to control which specific plot attributes are assigned to each group. This eliminates the inconsistencies that can occur when we allow the ODS statistical graphics procedures to assign the attributes using the default method.

There are two kinds of attribute maps. Discrete attribute maps allow us to assign plot attributes to specific data values, while range attribute maps allow us to map certain plot attributes to entire ranges of data values. Here we use a discrete attribute map.

Using an attribute map is a two-step process. First, an attribute map data set must be created. Then, this attribute map data set is applied using one of the statistical graphics procedures such as SGPLOT.

The attribute map data set can be created using any of the various programming techniques that SAS provides for creating data sets, but it must conform to specific structure. Refer to SAS 9.4 ODS Graphics: Procedures Guide, Sixth Edition (2016) for full details on the attribute map data set.

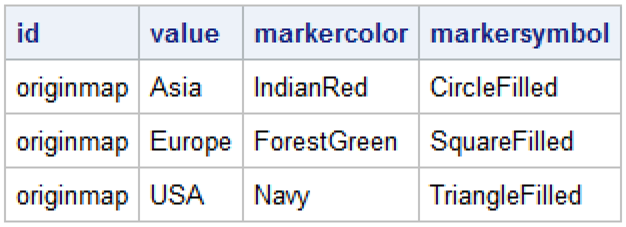
At a minimum, a discrete attribute map data set must contain variables called ID and VALUE. ID is a unique attribute map identifier that is used to distinguish between multiple sets of attribute mappings contained within the same data set. The VALUE variable contains the formatted data value to which attributes are to be applied.

In addition, several variables may be included within the discrete attribute map data set to specify the plot attributes. These variables control various marker attributes, line attributes, fill attributes, and text attributes. Refer to the SAS 9.4 ODS Graphics: Procedures Guide, Sixth Edition (2016) for a complete list. Here we use only MARKERCOLOR and MARKERSYMBOL.

The following DATA step creates the discrete attribute map data set for Example 3. The arbitrary name ORIGINMAP has been assigned to this mapping.

```
data myattrmap;
  length id $9 value $6 markercolor $11 markersymbol $14;
    input id $ value $ markercolor $ markersymbol $;
    datalines;
      originmap Asia IndianRed CircleFilled
      originmap Europe ForestGreen SquareFilled
      originmap USA Navy TriangleFilled
      ;
run;
```

Executing this DATA step will produce the data set shown in Figure 4.



id	value	markercolor	markersymbol
originmap	Asia	IndianRed	CircleFilled
originmap	Europe	ForestGreen	SquareFilled
originmap	USA	Navy	TriangleFilled

Figure 4. Discrete Attribute Map Data Set for Example 3

Once the attribute map data set has been created, it must be applied to our scatter plot. We specify the name of the attribute map data set using the DATTRMAP= option on the PROC SGPLOT statement. The ID value for our map is supplied with the ATTRID= option on the SCATTER statement to indicate that this plot request statement should use this mapping.

```
proc sgplot data=sashelp.cars dattrmap=myattrmap;
  title "Horsepower vs. Weight - Trucks Only";
  where type='Truck';
  scatter y=horsepower x=weight /
    group=origin
    markerattrs=(size=10px)
    attrid=originmap;
run;
```

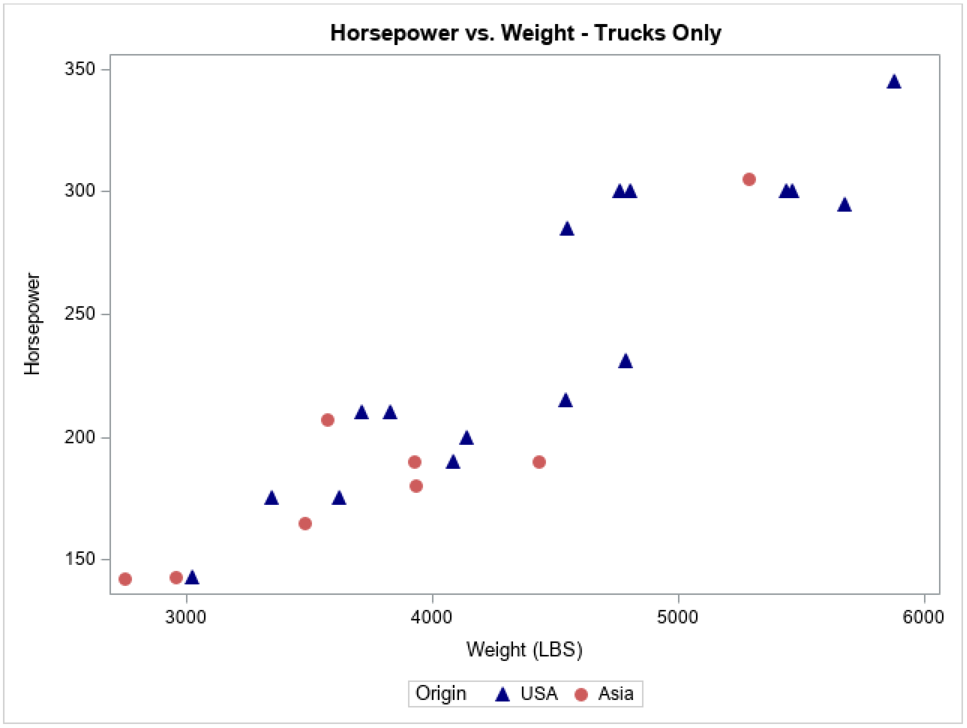


Figure 5. SGPLOT Output for Example 3

Figure 5 shows the output created by the SGPLOT procedure in conjunction with the attribute map. Note that the attributes have been mapped as specified. Asia is represented by a red circle while USA is indicated by a blue triangle.

Unfortunately, the group values are not displayed in the legend in the same order as was used in our original plot in Example 1. This is due to the order in which the data values appear in our subset of the data, which is not the same as the order in which the data values appear in the complete data set. We can fix this minor issue by adding a KEYLEGEND statement and using the SORTORDER=ASCENDING option to specify the order in which legend values should appear. The resulting SGPLOT output is shown in Figure 6.

```
proc sgplot data=sashelp.cars dattrmap=myattrmap;
  title "Horsepower vs. Weight - Trucks Only";
  where type='Truck';
  scatter y=horsepower x=weight /
    group=origin
    markerattrs=(size=10px)
    attrid=originmap;
  keylegend / sortorder=ascending;
run;
```

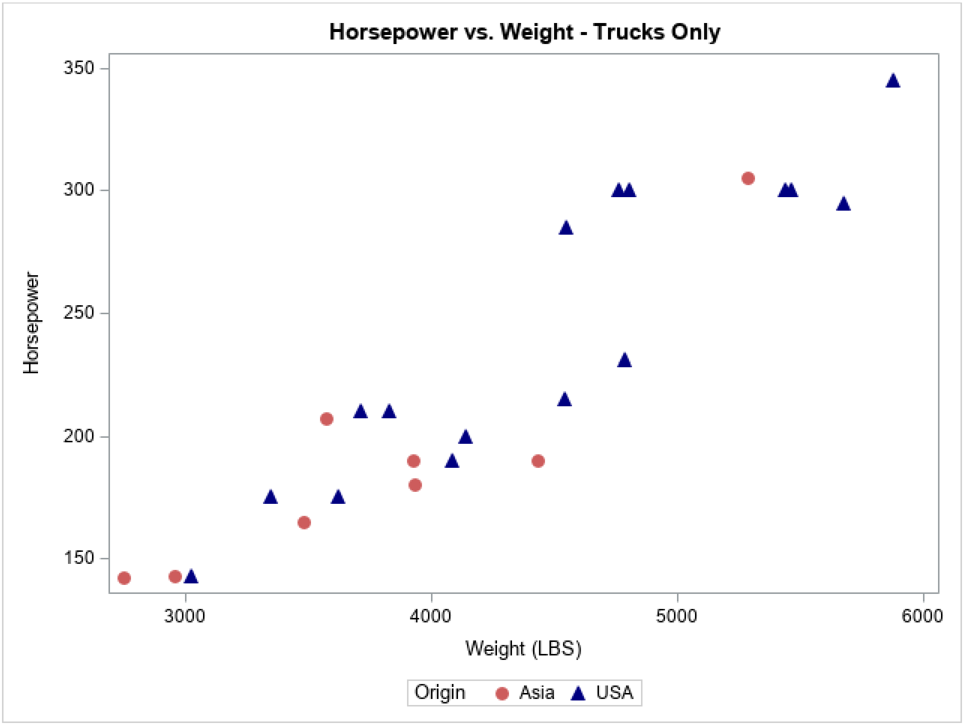


Figure 6. SGPLOT Output for Example 3 with Sorted Legend Entries

To make this plot more consistent with our original plot from Example 1, there is one more minor adjustment we wish to make. We would like the legend to include Europe, even though there are no records in this specific subset of the data with an ORIGIN of Europe.

This can be accomplished by adding a new variable called SHOW to the discrete attribute map. If we give this variable a value of "ATTRMAP" (as opposed to "DATA"), the legend will display the corresponding entry from our attribute map regardless of whether the value appears in the data. We update the DATA step that creates the discrete attribute map as follows:

```
data myattrmap;
  length id $9 value $6 markercolor $11 markersymbol $14 show $7;
  input id $ value $ markercolor $ markersymbol $ show $;
  datalines;
    originmap Asia IndianRed CircleFilled attrmap
    originmap Europe ForestGreen SquareFilled attrmap
    originmap USA Navy TriangleFilled attrmap
    ;
run;
```

Utilizing this updated attribute map with the same PROC SGPLOT code produces the result shown in Figure 7. Note that the legend is now consistent with the one from Example 1 (Figure 2).

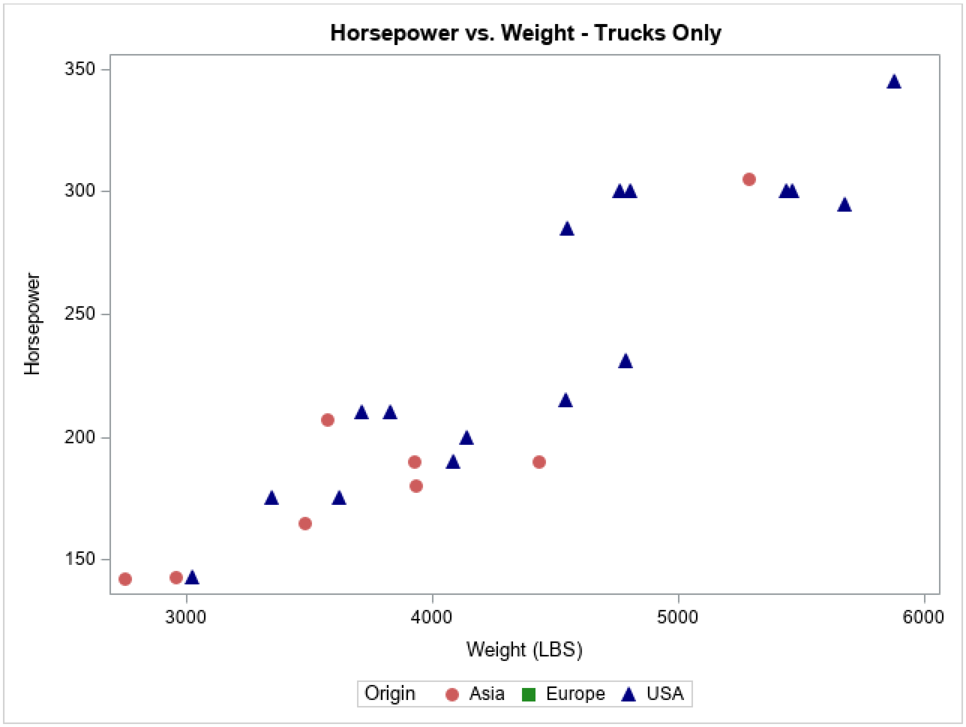


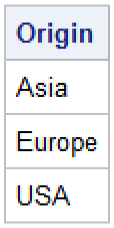
Figure 7. SGPLOT Output for Example 3 with Full Legend Entries

EXAMPLE 4: AUTOMATING CREATION OF THE ATTRIBUTE MAP DATA SET

Up to this point, we have created our attribute map data set by hard-coding all the values into a DATA step using a DATALINES statement. While this method provided a simple way to illustrate the use of attribute maps, it is often advantageous to use more dynamic techniques for creation of the attribute map data sets. Example 4 produces the same output as Example 3 but automates certain aspects of creating the attribute map data set.

First, we begin by obtaining the unique values of ORIGIN from SASHELP.CARS by using PROC SORT with the NODUPKEY option. This results in the small data set show in Figure 8. It contains only one variable, ORIGIN, with just one row per unique value.

```
proc sort data=sashelp.cars out=origins(keep=origin) nodupkey;
  by origin;
run;
```



Origin
Asia
Europe
USA

Figure 8. Data Set Containing Unique Values of ORIGIN

With this data set as an input, we use a DATA step to construct our attribute map data set. The variable ORIGIN is renamed to VALUE, one of the required variables in a discrete attribute map data set. Additionally, we utilize the CHOOSEC function to dynamically assign values to MARKERCOLOR and MARKERSYMBOL for each record in the data set. The CHOOSEC function returns a value from a list of arguments based on an index specified in the first argument. Refer to Horstman (2017) for more examples with the CHOOSEC function. Finally, we assign static values to ID and SHOW.

```
data myattrmap;
  set origins(rename=(origin=value));
  length id $9 markercolor $11 markersymbol $14 show $7;
  id = 'originmap';
  markercolor = choosec(_n_,'IndianRed','ForestGreen','Navy');
  markersymbol = choosec(_n_,'CircleFilled','SquareFilled',
    'TriangleFilled');
  show = 'attrmap';
run;
```

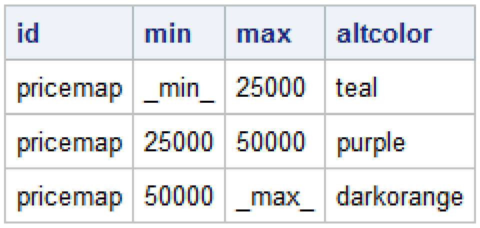
The attribute map data set produced by this DATA step is identical to the final attribute map data set created in Chapter 3. However, it was created without hard-coding the values of ORIGIN.

EXAMPLE 5: USING A RANGE ATTRIBUTE MAP

Our final example uses a range attribute map. Range attribute maps work much like discrete attribute maps except that attributes are assigned to an entire range of values. Consequently, the structure of the attribute map data set differs somewhat. While an ID is still required, the VALUE variable is replaced with two variables, MIN and MAX, that define a range of values.

In Example 5, we modify our original scatter plot (using the full CARS data set) so that the plot marker colors represent the price (MSRP) of the vehicle. We group the prices into ranges so that vehicles under $25,000 are colored teal, those between $25,000 and $50,000 are purple, and those above $50,000 are dark orange. The following DATA step creates the range attribute map data set, which is shown in Figure 9.

```
data myattrmap2;
  length id $9 min $5 max $5 altcolor $10;
  input id $ min $ max $ altcolor $;
  datalines;
    pricemap _min_ 25000 teal
    pricemap 25000 50000 purple
    pricemap 50000 _max_ darkorange
  ;
run;
```



id	min	max	altcolor
pricemap	_min_	25000	teal
pricemap	25000	50000	purple
pricemap	50000	_max_	darkorange

Figure 9. Range Attribute Map Data Set for Example 5

We also need to make some slight changes to our SGPLOT code. First, on the PROC SGPLOT statement itself, we use the RATTRMAP= option (instead of the DATTRMAP= option) to specify the name of the range attribute map data set. Secondly, on the SCATTER statement, the ID of our mapping is specified using the RATTRID= option (instead of the ATTRID= option). Finally, we need to tell SGPLOT which variable to use in applying the range attribute map, which is done using the COLORRESPONSE= option. Note that the MARKERATTRS= option is also to specify marker attributes that should apply to all markers. The output is shown in Figure 10.

```
proc sgplot data=sashelp.cars rattrmap=myattrmap2;
  title "Horsepower vs. Weight by MSRP";
  scatter y=horsepower x=weight /
    colorresponse=msrp
    rattrid=pricemap
    markerattrs=(symbol=CircleFilled size=10px);
run;
```

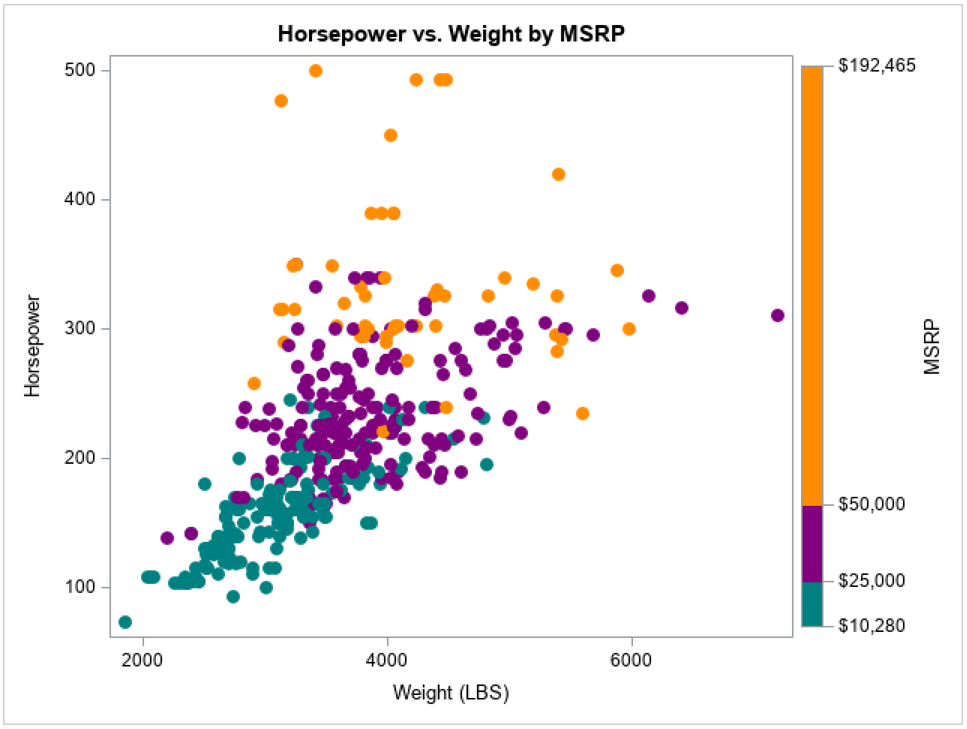


Figure 10. SGPLOT Output for Example 5

CONCLUSION

Attribute maps provide a convenient and flexible way to associate specific visual attributes with specific data values (or ranges of values) in a plot. This functionality can be used to provide consistency across multiple plots, eliminate dependencies upon the order of the data in the input data set, and include legend entries not found in the data.

In this set of examples, we focused on the use of attribute maps with the SGPLOT procedure. However, this functionality is also available in several other ODS statistical graphics procedures including SGPANEL, SGSCATTER, and SGPIE.

We have only scratched the surface of what can be done using attribute maps. For more details, refer to SAS 9.4 ODS Graphics: Procedures Guide, Sixth Edition (2016).

REFERENCES

Horstman, Joshua M. 2017. “Beyond IF THEN ELSE: Techniques for Conditional Execution of SAS® Code.” Proceedings of the SAS Global Forum 2017 Conference. https://support.sas.com/resources/papers/proceedings17/0326-2017.pdf

SAS Institute Inc. 2016. *SAS® 9.4 ODS Graphics: Prcoedures Guide, Sixth* Edition. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Joshua M. Horstman
Nested Loop Consulting
josh@nestedloopconsulting.com