CHAPTER 1

Bring Order to your Bar Charts

Purpose: This chapter describes techniques you can use to control the order of the bars, and the stacking order of the colored bar segments, in Proc GChart bar charts.

Bar Order

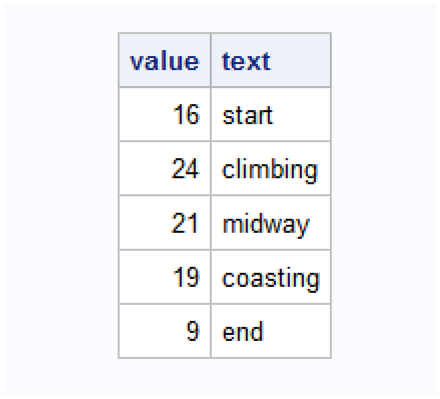
This first example shows how to control the order of the bars in your bar chart.

Proc GChart has two built-in ways to control the order of the bars in a bar chart – either in the alpha/numeric order (which is the default), or in ascending/descending order of the bar height (by using an easy option). But sometimes neither of those are exactly what you want. This example demonstrates how you can customize the chart to have the bars in any order you want.

First, we'll create a simple data set to use in this example:

```
data my_data;
input value text $ 4-12;
datalines;
16 start
24 climbing
21 midway
19 coasting
 9 end
;
run;
```

Here's what the dataset looks like:

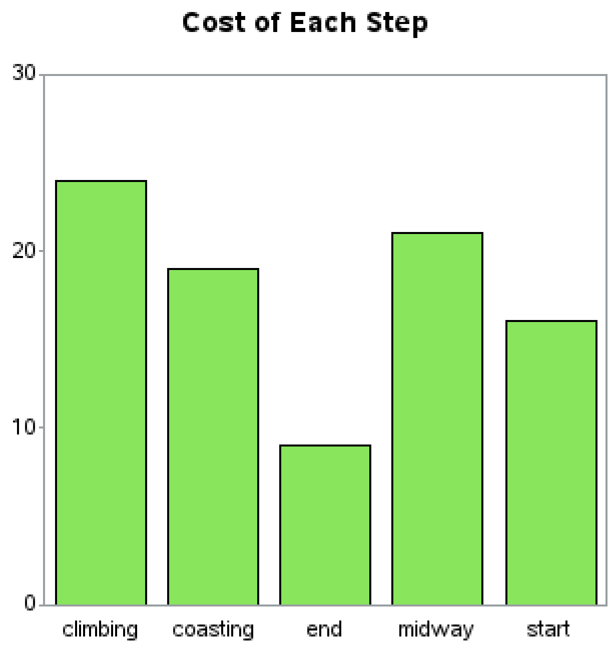


value	text
16	start
24	climbing
21	midway
19	coasting
9	end

You can easily generate a simple bar chart using code like this:

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

title1 ls=1.5 "Cost of Each Step";
pattern1 v=s color=bilg;
proc gchart data=my_data;
vbar text / type=sum sumvar=value
 raxis=axis1 maxis=axis2;
run;
```

The above code produces a chart like this, with the bars sorted in alphabetic order:

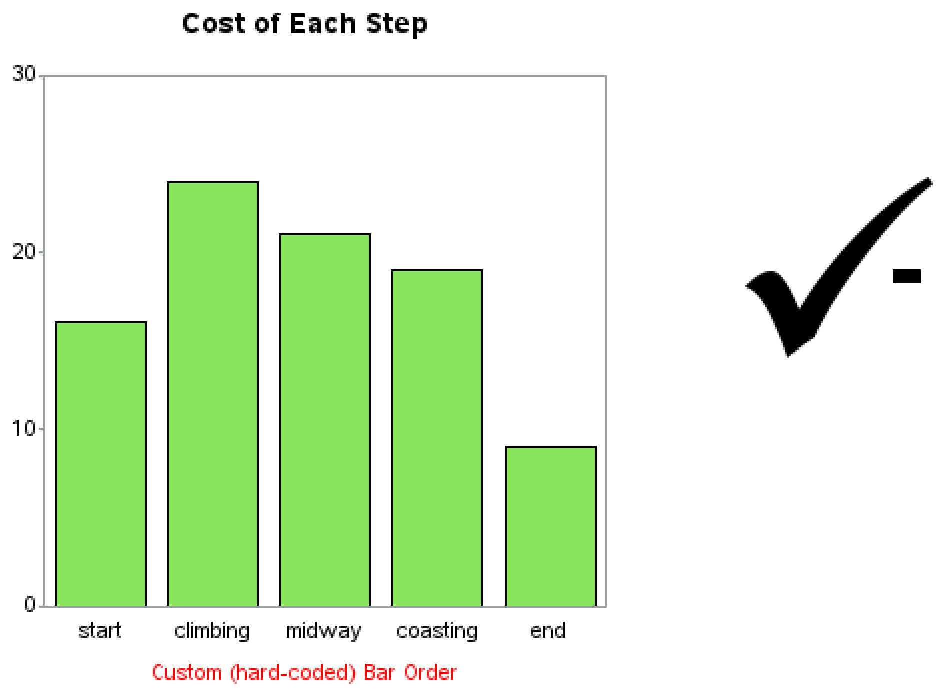


Default (alphabetic) Bar Order

Alphabetic order is ok for many bar charts, but in cases like this, the bars would make more sense if they were in a certain order. You can hard-code the desired order in an axis statement for the maxis, as shown in axis2 below:

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none
 order=('start' 'climbing' 'midway' 'coasting' 'end');

title1 ls=1.5 "Cost of Each Step";
pattern1 v=s color=bilg;
proc gchart data=my_data;
vbar text / type=sum sumvar=value
 raxis=axis1 maxis=axis2;
run;
```

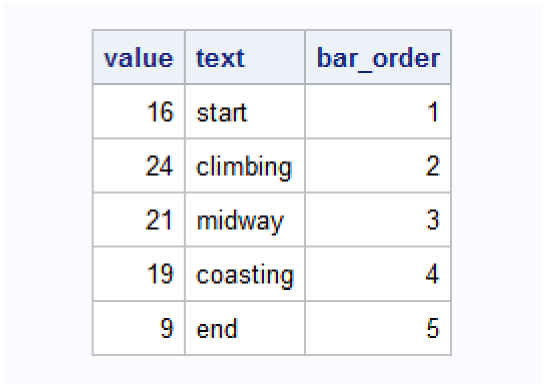
The resulting chart has the bars in the desired order, but there are a few drawbacks & possible gotchas to this coding technique. First, you must spell the text in your axis statement 100% correctly – if you make a typo, then a car could be left out of the chart. Also, if your data changes, you must remember to also change the hard-coded text in your axis statement. This puts a lot of responsibility on you, the programmer:



With this particular data, the observations in the dataset are actually sorted in the desired order. This is often the case with data, either as a direct result of the way the data were entered, or from sorting the data based on certain variables, etc. Therefore, it is useful to have a trick to sort the data chart in the **data order** (ie, the order that the observations appear in the dataset). We can 'capture' the data order by assigning **_n_** (an automatic variable containing the observation number) to a variable.

```
data my_data; set my_data;
bar_order=_n_;
run;
```

The dataset looks like this, with a numeric variable called **bar_order** added …

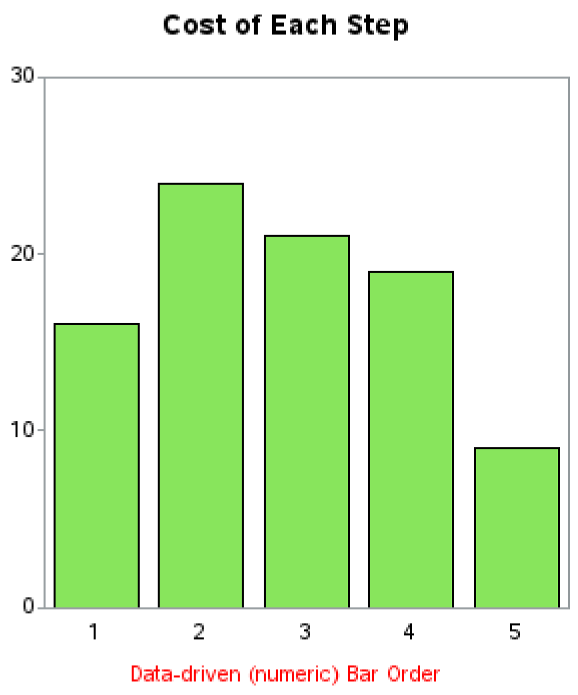


value	text	bar_order
16	start	1
24	climbing	2
21	midway	3
19	coasting	4
9	end	5

You can now generate a bar chart using the numeric variable for the bars, and they will be in the desired numeric order (note that when using a numeric variable for the bars, you must also use the 'discrete' option – otherwise the bars will be combined and summarized into a histogram) …

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

title1 ls=1.5 "Cost of Each Step";
pattern1 v=s color=bilg;
proc gchart data=my_data;
vbar bar_order / discrete type=sum sumvar=value
 raxis=axis1 maxis=axis2 width=12;
run;
```

The resulting bar chart is now in data order, but the bars are labeled with the numeric values (rather than the text) – not to worry, there is a 'trick' for fixing that!



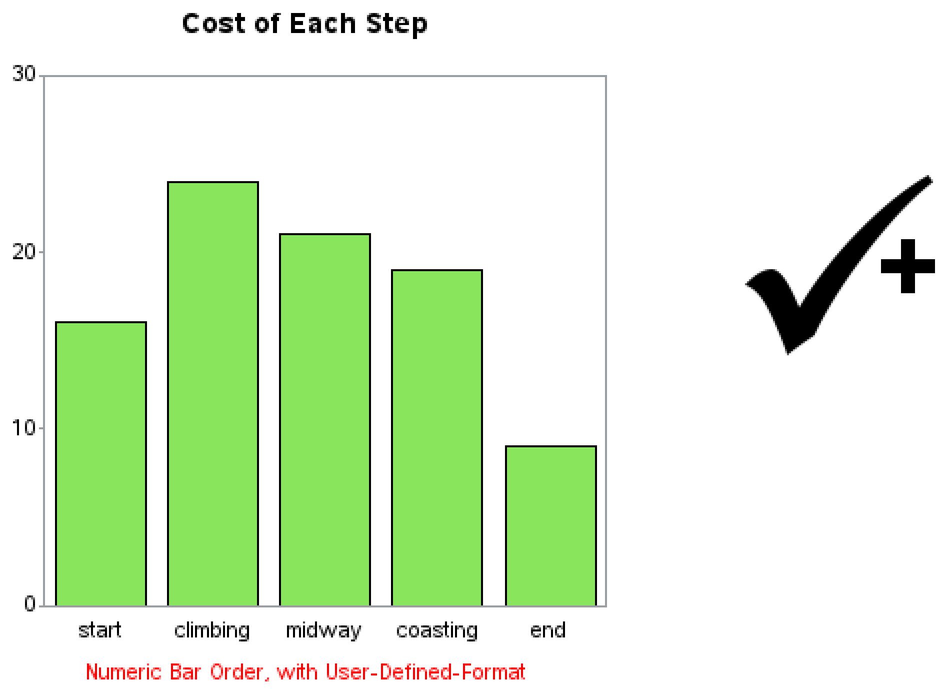
Of course, you could hard-code a user-defined format that would make the numbers print as the desired text … but that would have the same drawbacks & gotchas as hard-coding the order in an axis statement. We need a way to make this data-driven, so the user-defined format will automatically change any time the data changes.

The following code is the 'trick' for generating such a user-defined format directly from the data. Basically, you create a data set with the numeric values (named as 'start'), and the text values (named as 'label'), and then run it through Proc Format, as shown in the code below. You should definitely add this re-usable code to your bag of apprentice tricks!

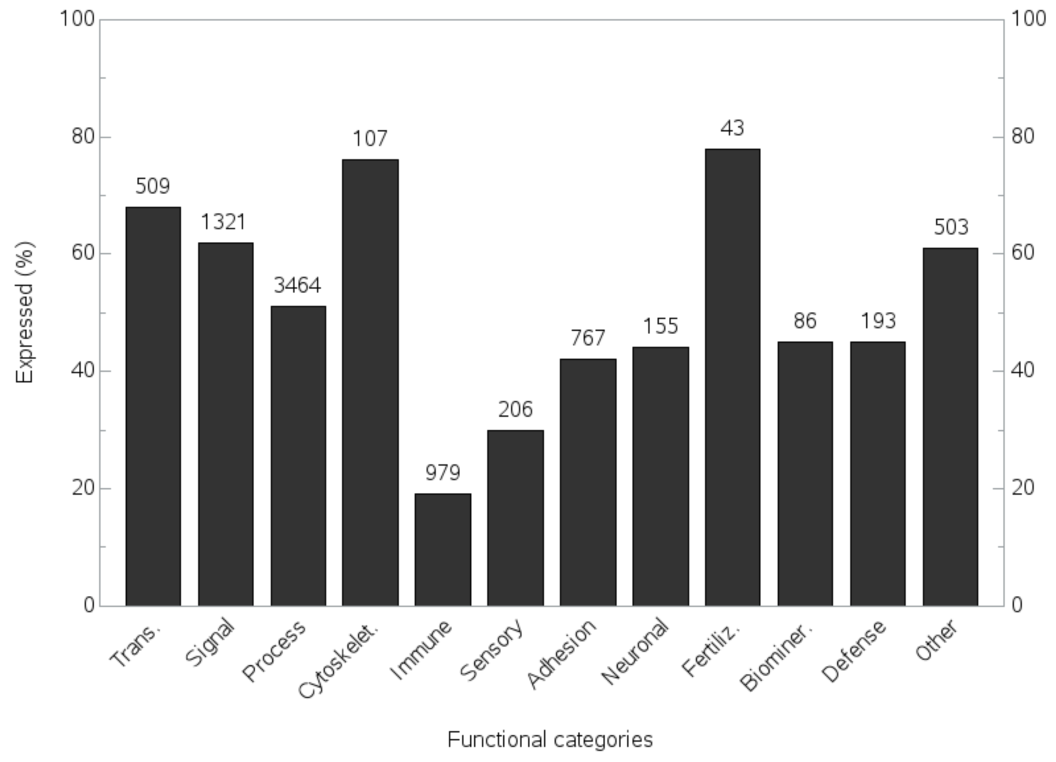
```
proc sql;
create table foo as select unique
 bar_order as start,
 text as label
 from my_data;
quit; run;
data control; set foo;
fmtname = 'barfmt';
type = 'N';
end = START;
run;
proc format lib=work cntlin=control;
run;
```

You can now generate the bar chart using the numeric bar order, and apply this user-defined format (which I called **barfmt**) to the numeric variable so that the bars will have the desired text printed instead of the numbers.

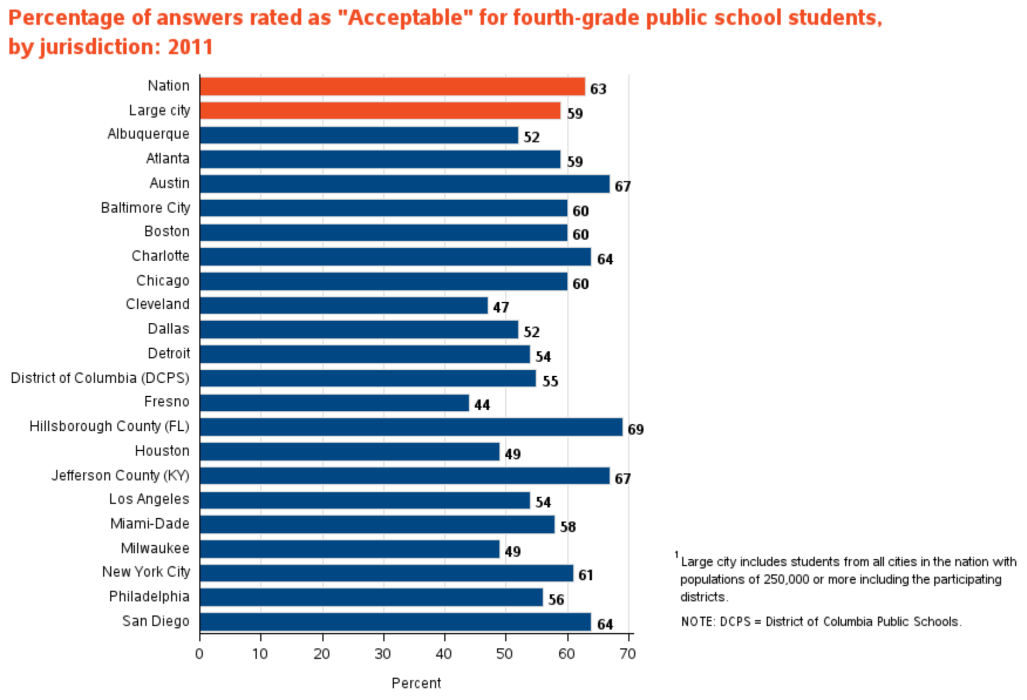
```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

title1 ls=1.5 "Cost of Each Step";
pattern1 v=s color=bilg;
proc gchart data=my_data;
format bar_order barfmt.;
vbar bar_order / discrete type=sum sumvar=value
 raxis=axis1 maxis=axis2;
run;
```

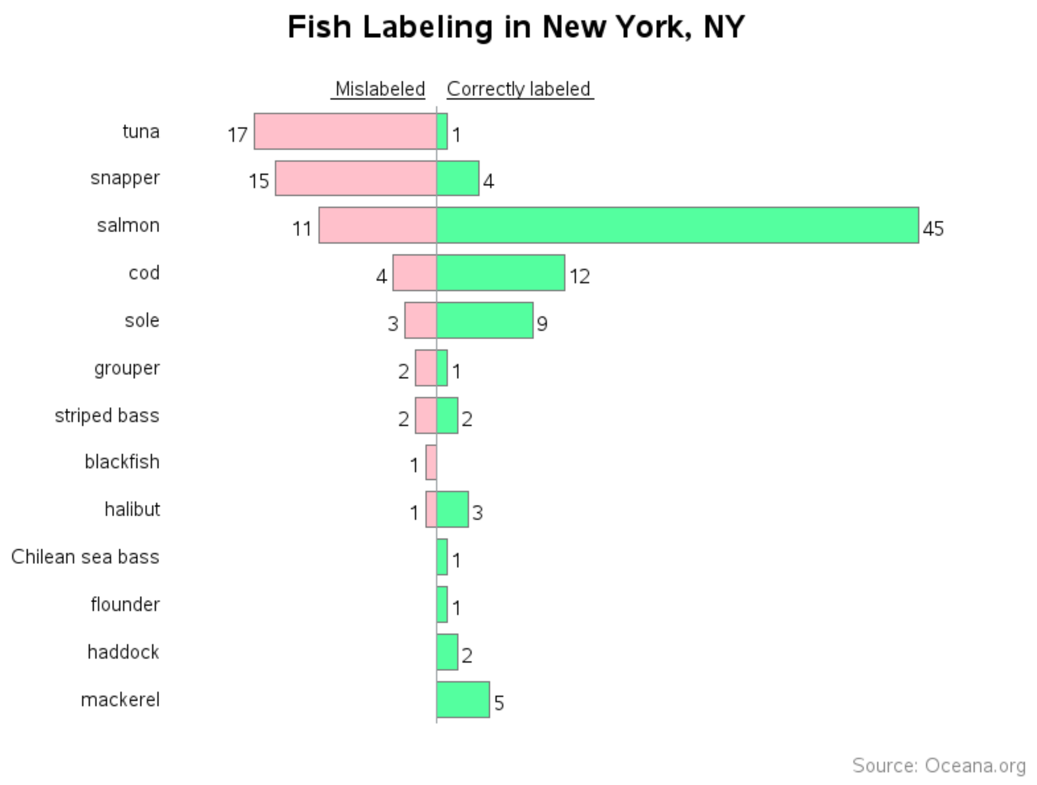


Here are a few other examples that demonstrate how this trick (using a numeric value to control the bar order, and a user-defined format to make it print as the desired text) can be used with real data:



Proof-of-Concept, using fabricated data





Let's Talk: When it comes to customizing a graph to look just like you want it, I like the saying "any problem can be solved with another level of indirection." And when it comes to getting axes to sort a certain way, but to visually appear with different (more readable) values, that indirection comes from formats … in many cases user-defined formats.

Stacking Order

This second example shows how to control the stacking order of the colored segments in your bar chart.

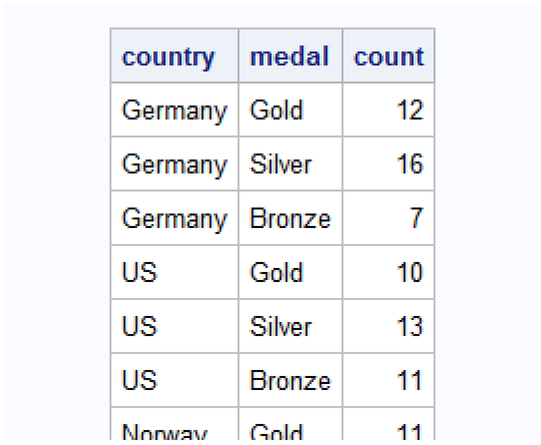
Most people just accept the default stacking order of the segments in a bar chart, being an intermediate-level wizard's apprentice, you are driven to go the extra mile and customize the stacking order when this can help make a better chart. This section will teach you the #1 best trick to accomplish that.

For these next few examples, we will use the following data, which shows the number of bronze, silver, and gold medals won by various countries in the Olympics.

```
data my_data;
length country $7 medal $6;
```

```
input country $ medal $ count;
datalines;
Germany Gold   12
Germany Silver 16
Germany Bronze  7
US      Gold   10
US      Silver 13
US      Bronze 11
Norway  Gold   11
Norway  Silver  7
Norway  Bronze  6
Canada  Gold    6
Canada  Silver  3
Canada  Bronze  8
Russia  Gold    6
Russia  Silver  7
Russia  Bronze  3
;
run;
```

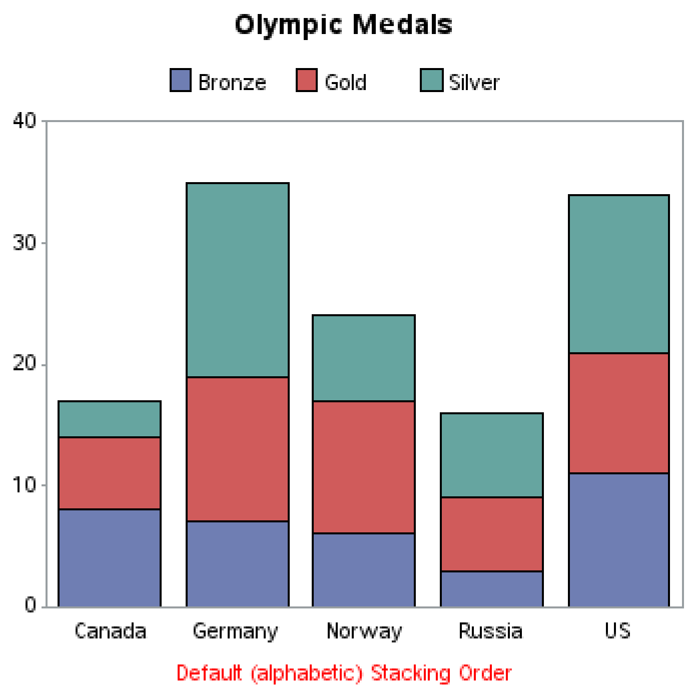
Here's what the first few lines of data look like in the SAS data set:



country	medal	count
Germany	Gold	12
Germany	Silver	16
Germany	Bronze	7
US	Gold	10
US	Silver	13
US	Bronze	11
Norway	Gold	11

If you create a simple stacked bar chart (using the subgroup= option), notice that the bar segments are stacked in alphabetic order by default – bottom-to-top, Bronze, Gold, and Silver.

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

legend1 label=none position=(top center)
 shape=bar(.11in,.11in);

title1 ls=1.5 "Olympic Medals";
proc gchart data=my_data;
vbar country / type=sum sumvar=count
 subgroup=medal legend=legend1
 raxis=axis1 maxis=axis2;
run;
```

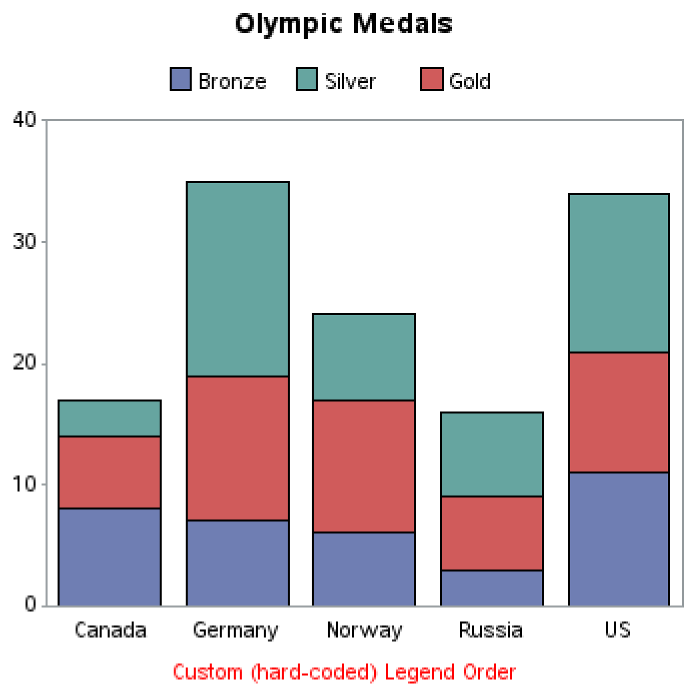


Similar to hard-coding the axis order (as shown in the bar order example), you could try hard-coding the legend order …

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

legend1 label=none position=(top center)
```

```
 shape=bar(.11in,.11in)
 order=('Bronze' 'Silver' 'Gold');

title1 ls=1.5 "Olympic Medals";
proc gchart data=my_data;
vbar country / type=sum sumvar=count
 subgroup=medal legend=legend1
 raxis=axis1 maxis=axis2;
run;
```

The custom legend does indeed control the order of the items in the color legend (Bronze, Silver, Gold), but it does not affect the order the colors are assigned to the bar segments – they are still assigned in alphabetic order, from bottom-to-top (Bronze, Silver, Gold). The colors still match (ie, no data integrity problem), but the order in the legend doesn't match the stacking order … therefore you've actually made the chart a little more difficult to read (rather than improving it).

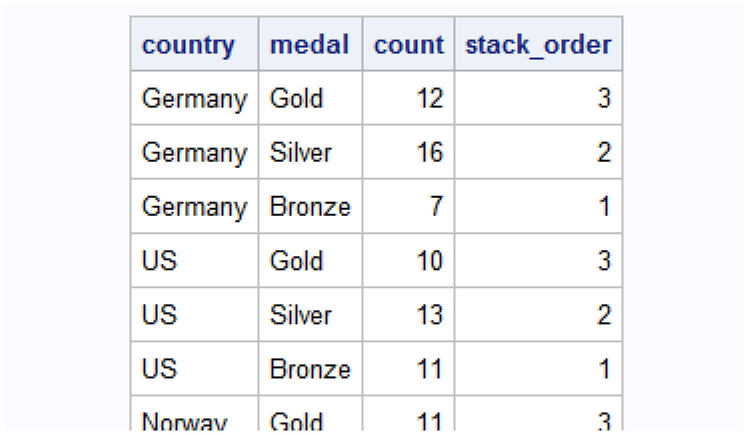


So, what trick **do** you use? As with many tasks dealing with custom ordering in SAS/Graph charts, you can assign numeric values to control the stacking order, and then have the numeric values print as the desired text using a user-defined format.

In this example, you can assign numeric values using a simple data step as follows:

```
data my_data; set my_data;
if medal='Bronze' then stack_order=1;
if medal='Silver' then stack_order=2;
if medal='Gold' then stack_order=3;
run;
```

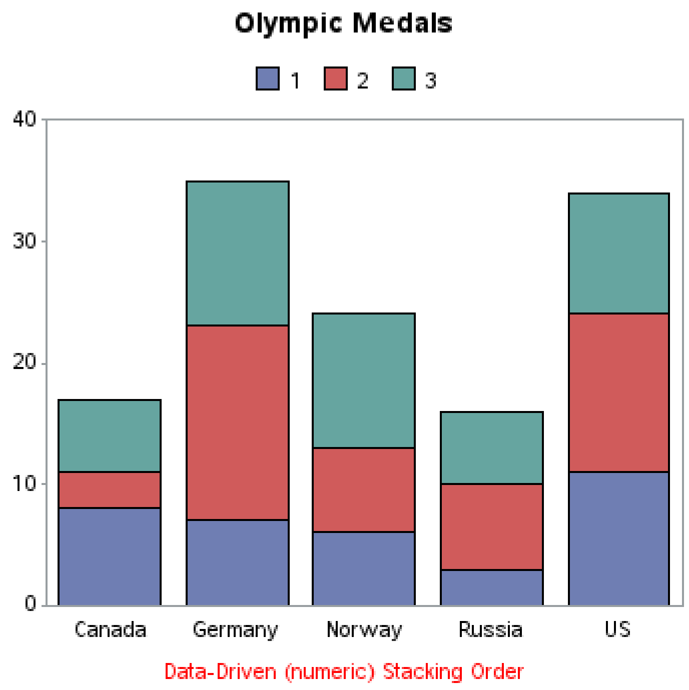
Here are the first few lines of the modified dataset, with the stack_order variable added:



country	medal	count	stack_order
Germany	Gold	12	3
Germany	Silver	16	2
Germany	Bronze	7	1
US	Gold	10	3
US	Silver	13	2
US	Bronze	11	1
Norway	Gold	11	3

And now you can easily color the bar segments by the numeric stack_order variable, producing a bar chart with the colored segments stacked in the desired order.

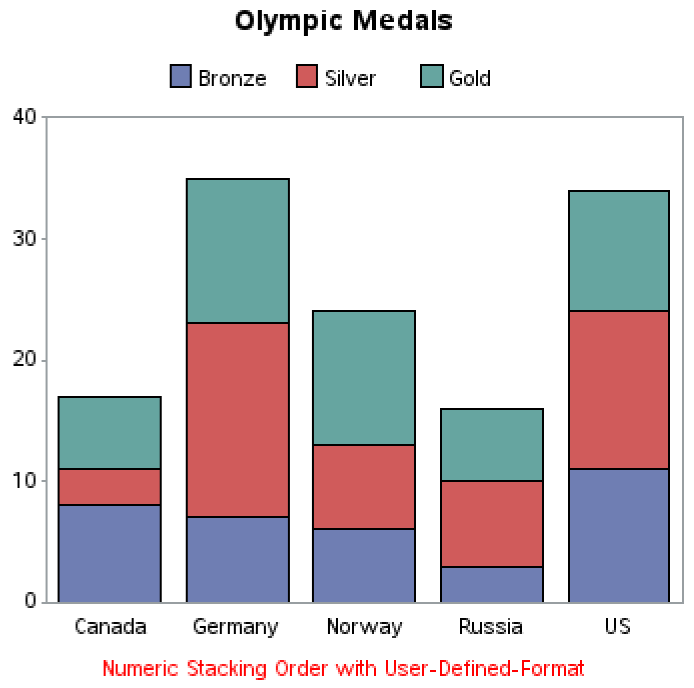
```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

legend1 label=none position=(top center)
 shape=bar(.11in,.11in);

title1 ls=1.5 "Olympic Medals";
proc gchart data=my_data;
vbar country / type=sum sumvar=count
 subgroup=stack_order legend=legend1
 raxis=axis1 maxis=axis2;
run;
```



With the above graph, you are half way there. The final step is to create a user-defined format so the numeric values (1, 2, 3) print as the desired text (Bronze, Silver, Gold). You can generate this user-defined format directly from the data, using the trick described in the previous (bar order) example. This is the same code we used before, with very slight modifications:

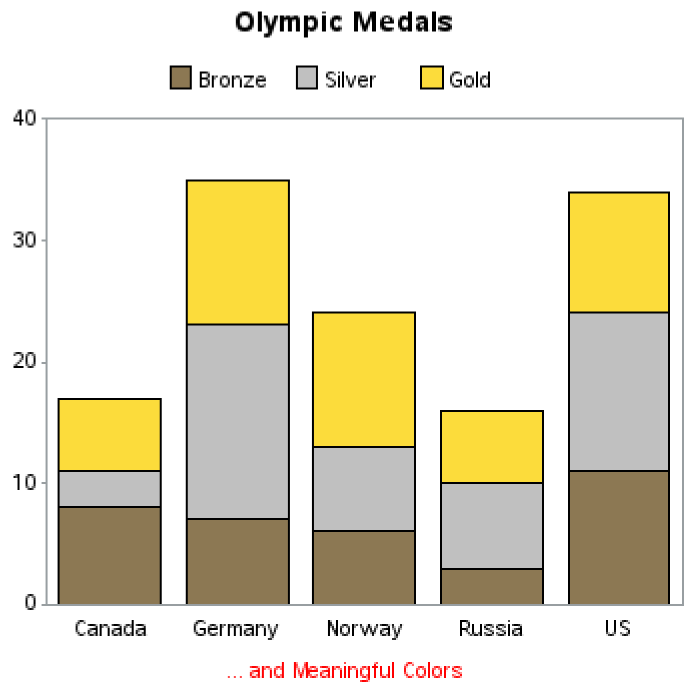
```
proc sql;
create table foo as select unique
 stack_order as start,
 medal as label
 from my_data;
quit; run;
data control; set foo;
fmtname = 'stackfmt';
type = 'N';
end = START;
run;
proc format lib=work cntlin=control;
run;
```

```
axis1 label=none minor=none offset=(0,0);
axis2 label=none;

legend1 label=none position=(top center)
 shape=bar(.11in,.11in) value=(justify=left);

title1 ls=1.5 "Olympic Medals";
proc gchart data=my_data;
format stack_order stackfmt.;
vbar country / type=sum sumvar=count
 subgroup=stack_order legend=legend1
 raxis=axis1 maxis=axis2;
run;
```

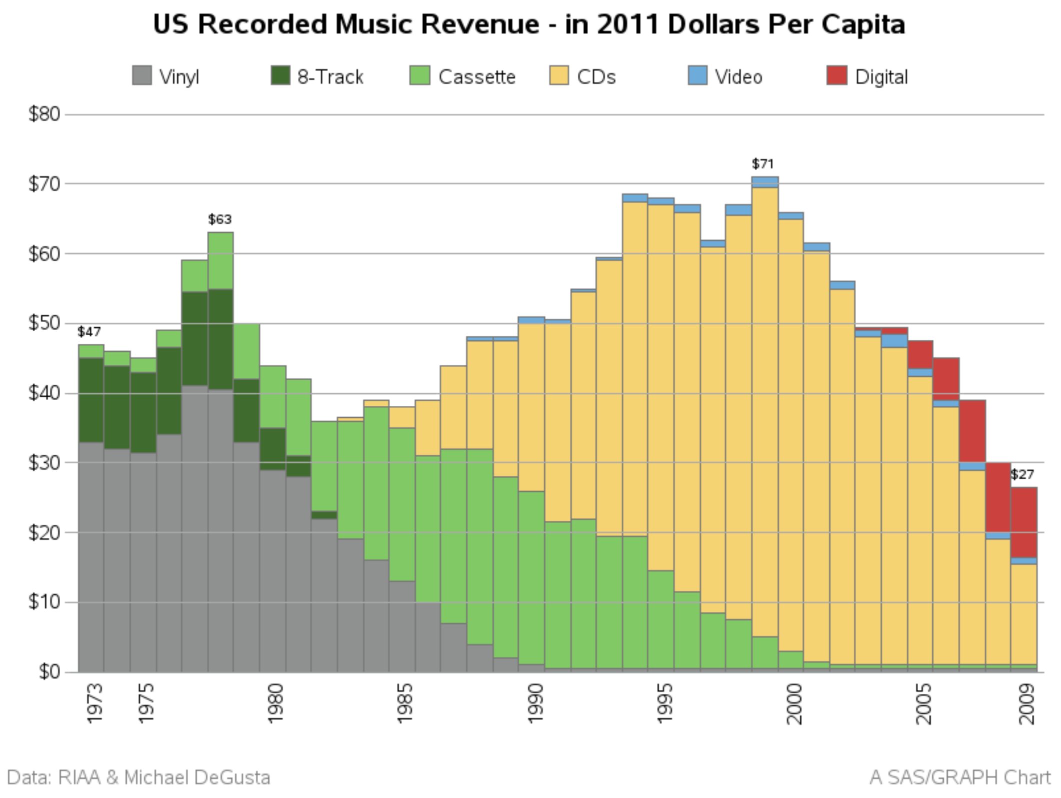


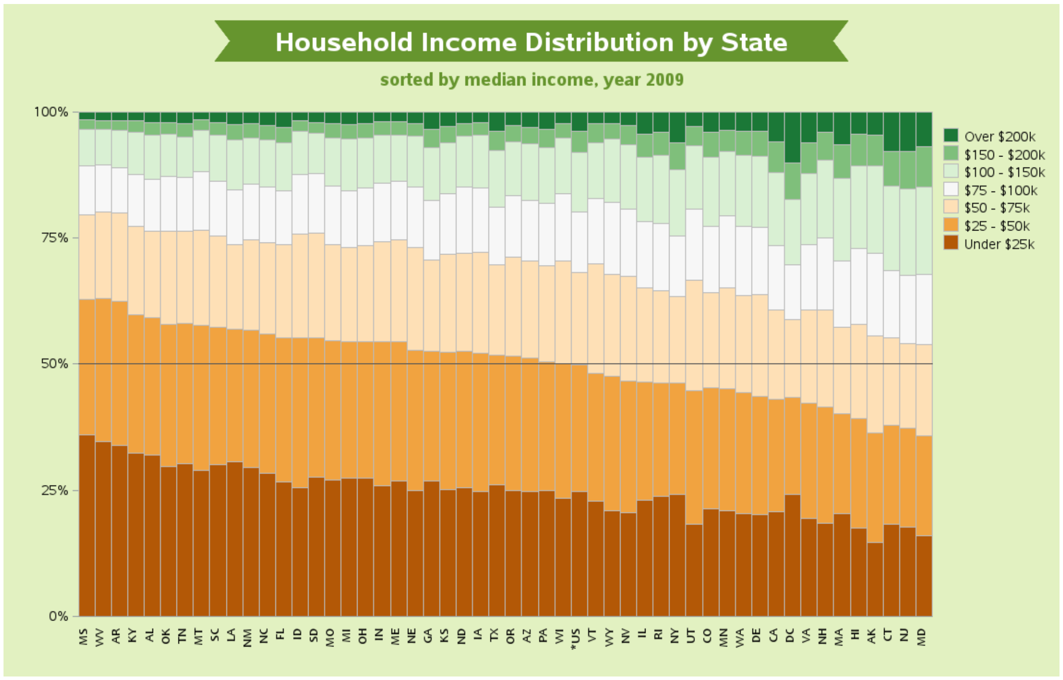
Now the bar segments are stacked in a meaningful/logical order, but what about those colors?... Let's make them match the real-world colors of the medals, which will make the chart even more intuitive, and easier to read. We can control the bar colors using pattern statements (notice that I'm using hex codes in the form of cxRRGGBB to get the exact colors I want):

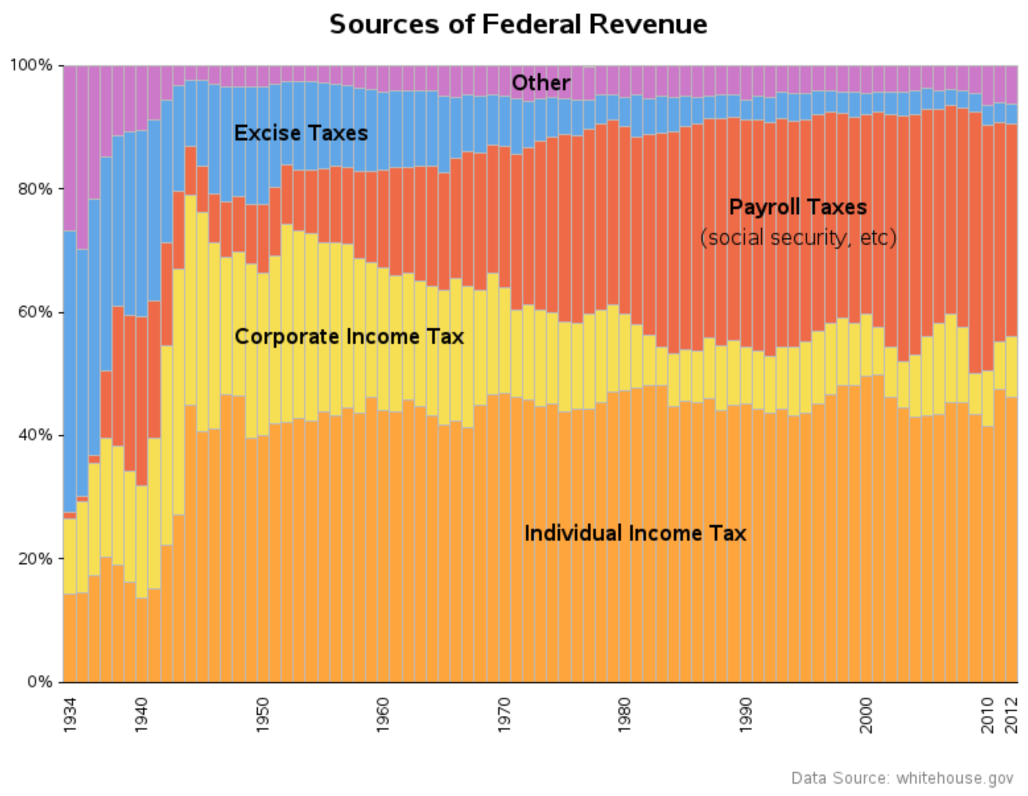
```
pattern1 v=s c=cx8C7853; /* bronze */
pattern2 v=s c=cxC0C0C0; /* silver */
pattern3 v=s c=cxFCDC3B; /* gold */

title1 ls=1.5 "Olympic Medals";
proc gchart data=my_data;
format stack_order stackfmt.;
vbar country / type=sum sumvar=count
 subgroup=stack_order legend=legend1
 raxis=axis1 maxis=axis2;
run;
```

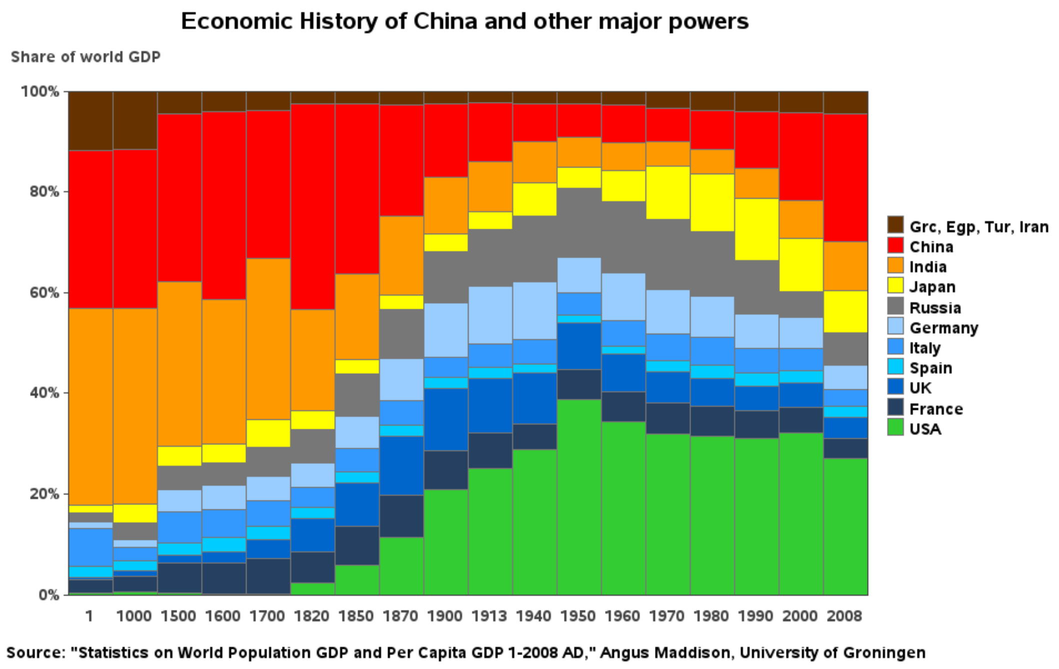


Here are a few other examples that demonstrate how this trick (using a numeric value to control stacking order, and a user-defined format to make it print as the desired text) can be used with real data:


US Recorded Music Revenue - in 2011 Dollars Per Capita
Vinyl
8-Track
Cassette
CDs
Video
Digital
$80
$70
$60
$50
$40
$30
$20
$10
$0
$47
$63
$71
$27
1973
1975
1980
1985
1990
1995
2000
2005
2009
Data: RIAA & Michael DeGusta
A SAS/GRAPH Chart


Household Income Distribution by State
sorted by median income, year 2009
100%
75%
50%
25%
0%
Over $200k
$150 - $200k
$100 - $150k
$75 - $100k
$50 - $75k
$25 - $50k
Under $25k
MS WV AR KY AL OK TN MT SC LA NM NC FL ID SD MO MI OH IN ME NE GA KS ND IA TX OR AZ PA WI *US VT WY NV IL RI NY UT CO MN WA DE CA DC VA NH MA HI AK CT NJ MD


Sources of Federal Revenue
100%
80%
60%
40%
20%
0%
Other
Excise Taxes
Payroll Taxes
(social security, etc)
Corporate Income Tax
Individual Income Tax
1934
1940
1950
1960
1970
1980
1990
2000
2010
2012
Data Source: whitehouse.gov


Economic History of China and other major powers
Share of world GDP
100%
80%
60%
40%
20%
0%
1
1000
1500
1600
1700
1820
1850
1870
1900
1913
1940
1950
1960
1970
1980
1990
2000
2008
Grc, Egp, Tur, Iran
China
India
Japan
Russia
Germany
Italy
Spain
UK
France
USA
Source: "Statistics on World Population GDP and Per Capita GDP 1-2008 AD," Angus Maddison, University of Groningen