CHAPTER 4

Pie Charts

Purpose: This chapter demonstrates how to create pie charts, and discusses some of the issues to consider when deciding whether to use a pie chart.

Simple Pie

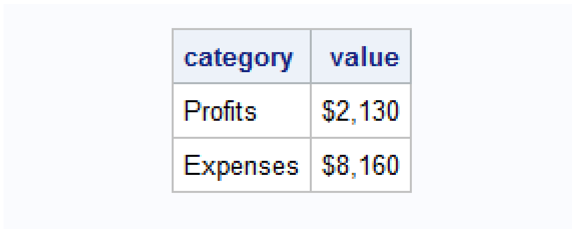
Most people *get* the concept of pie charts, and intuitively understand what "a piece of the pie" means. But people also tend to have trouble creating good pie charts, and frequently use them when other charts would be more appropriate.

Probably the most appropriate use of a pie chart is for a part-to-whole comparison, where the *part* is the one slice of interest, and the *whole* is the entire pie. The alternative would be to use a 100% stacked bar chart, containing just a single bar (which would be visually awkward).

In this first pie chart example, the whole pie represents the sales of an imaginary popcorn company, and it will have two slices. One slice will represent the amount of money used to pay expenses, and the other slice will represent profit. We will be most interested in the profit slice. Here is the data we will be using. This time, I'm assigning the values directly to variables, rather than setting up a DATALINES section.

```
data my_data;
length category $8;
format value dollar8.0;
category='Profits'; value=2130; output;
category='Expenses'; value=8160; output;
run;
```

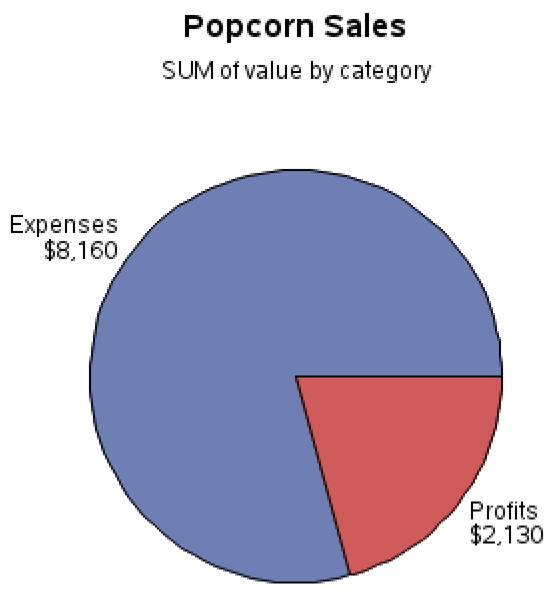
Here are the first few lines of data:



category	value
Profits	$2,130
Expenses	$8,160

Using minimal code, here is how to create the default pie chart.

```
title1 ls=1.5 "Popcorn Sales";
proc gchart data=my_data;
pie category / type=sum sumvar=value;
run;
```

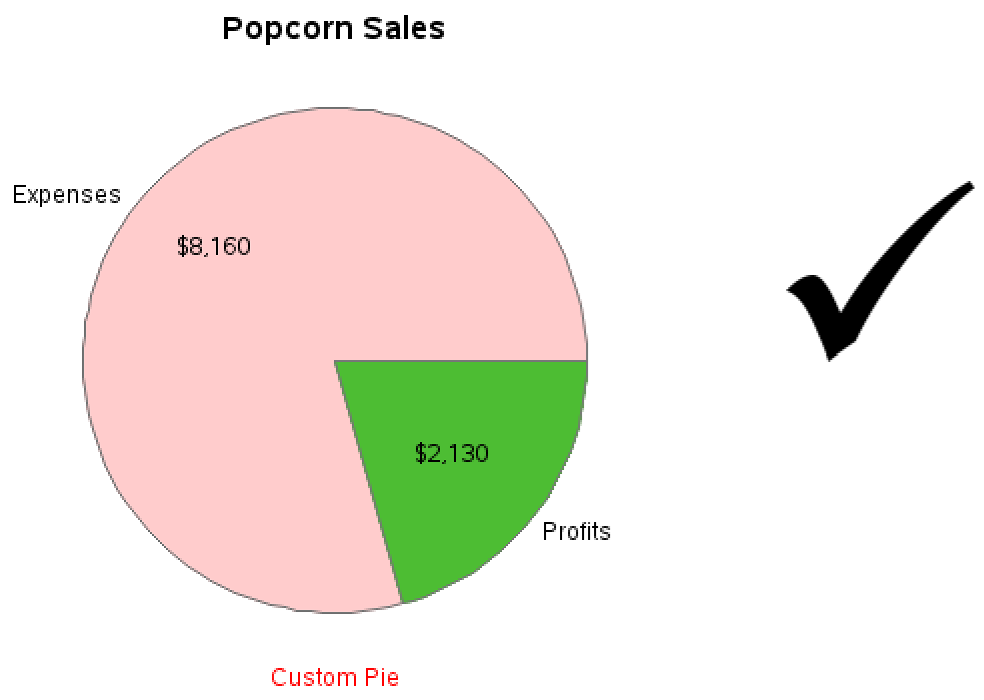


Default Plot

The default pie looks OK, but we can do much better by specifying a few simple options to customize it. For example, I recommend that you always use the NOHEADER option to get rid of the automatic header (“SUM of value by category”) below the title. This title is never really useful, and eliminating it allows the pie to reclaim that space and be drawn larger. I also recommend that you use the VALUE=INSIDE option to place the numeric values inside the slice rather than outside. Placing labels inside the pie provides more room outside, so the pie can be drawn larger. And let's also specify some custom colors, so the Profits slice (the slice of interest) is brighter and more attention-grabbing than the Expenses slice.

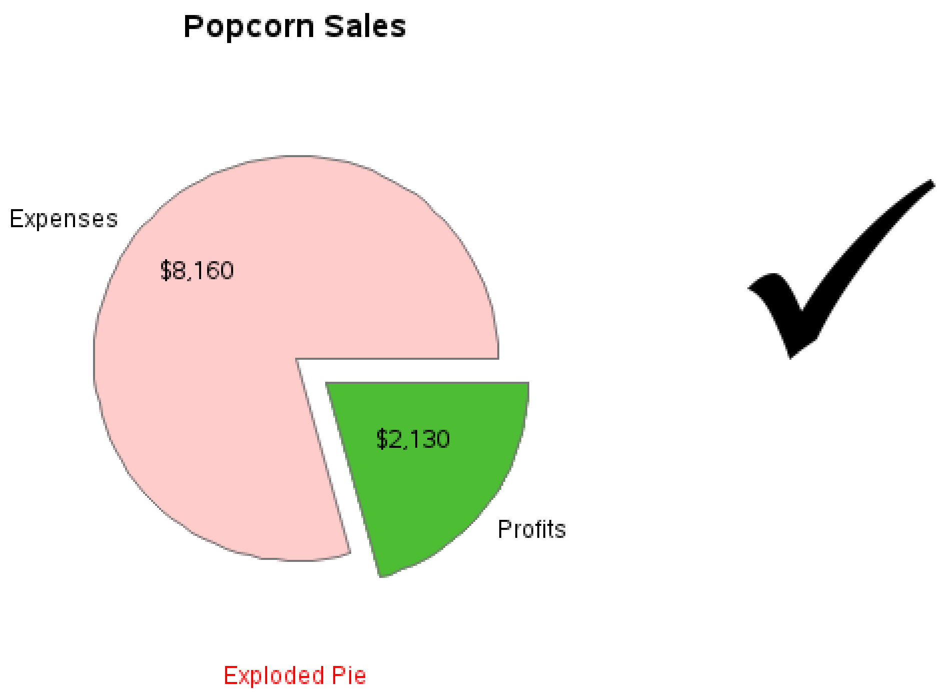
```
title1 ls=1.5 "Popcorn Sales";
pattern1 v=s color=cxFFCCCC;
pattern2 v=s color=cx4DBD33;
proc gchart data=my_data;
pie category / type=sum sumvar=value noheader
 slice=outside value=inside
 coutline=gray77;
run;
```

Notice that this custom pie is bigger (utilizing more of the available 400x350 pixel space), and much more visually pleasing than the default pie.



If you want to emphasize the slice of interest even more, you can use the EXPLODE option (see the following block of code) to physically bring that slice out away from the rest of the pie.

```
title1 ls=1.5 "Popcorn Sales";
pattern1 v=s color=cxFFCCCC;
pattern2 v=s color=cx4DBD33;
proc gchart data=my_data;
pie category / type=sum sumvar=value noheader
 slice=outside value=inside
 coutline=gray77 explode='Profits';
run;
```



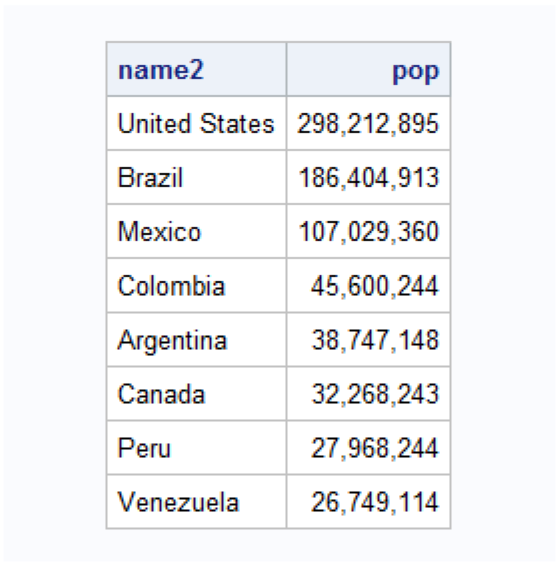
Crowded Pie

In general, it is best to limit the number of slices in a pie chart to a very small number. But if you must generate a pie chart with a larger number of slices, there are a few techniques you can use to help keep the pie readable.

In this example, we will use the SASHELP.DEMOGRAPHICS data which ships with SAS, and contains the population of each country. We'll create a pie chart of the countries in North and South America (region 'AMR'), and let the size of the pie slices represent the population. The country names are upper-case in the data, but we can use the PROPCASE() function to convert them to mixed-case. Here's the code used to subset the data and convert the country names.

```
data my_data;
  set sashelp.demographics (where=(region='AMR'));
name2=propcase(name);
run;
```

Here are a few rows of the customized data. There are 35 countries in the full data set.



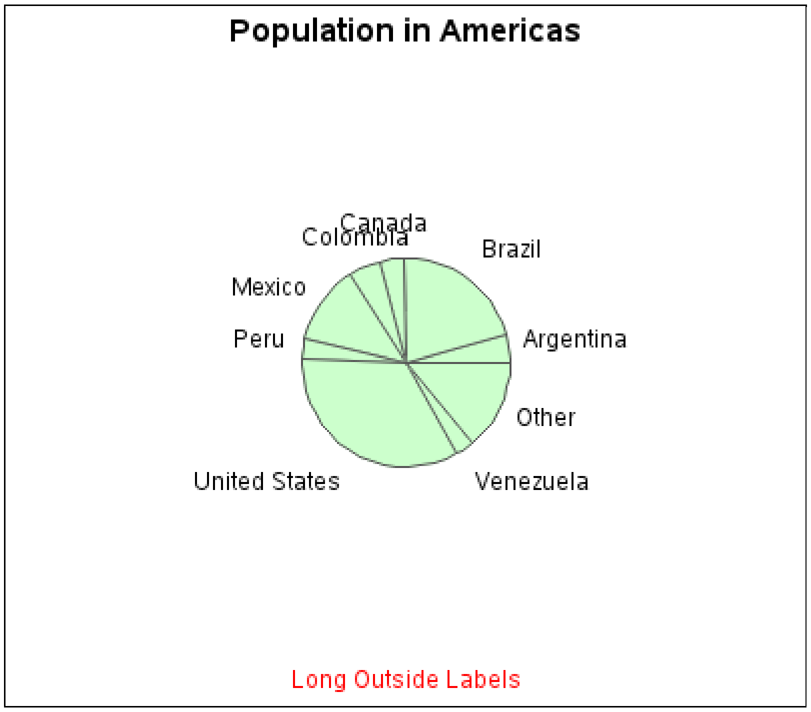
name2	pop
United States	298,212,895
Brazil	186,404,913
Mexico	107,029,360
Colombia	45,600,244
Argentina	38,747,148
Canada	32,268,243
Peru	27,968,244
Venezuela	26,749,114

We'll start with some fairly minimal code, but in cases like this where there are so many rows of data, it will take several iterations to refine the code and produce a *good* pie chart.

First, we'll use the VALUE=NONE option to suppress the population values from being printed in the chart – this will make the chart a little less crowded. Also, when there are so many potential pie slices (35 countries), PROC GCHART summarizes the smaller slices into the OTHER category and represents them with one single slice. By default, if a slice comprises 4% or less of the pie, then it is grouped into the OTHER slice. I experimented with some larger and smaller values, and determined that 3% works well with this particular data. I also used the OTHERLABEL option to specify a mixed-case name for the OTHER slice, so it would blend in better with the mixed-case country names.

```
title1 ls=1.5 "Population in Americas";
pattern1 v=s c=cxCCFFCC repeat=100;
proc gchart data=my_data;
pie name2 / type=sum sumvar=pop noheader
 slice=outside value=none coutline=gray55
 other=3 otherlabel='Other';
run;
```

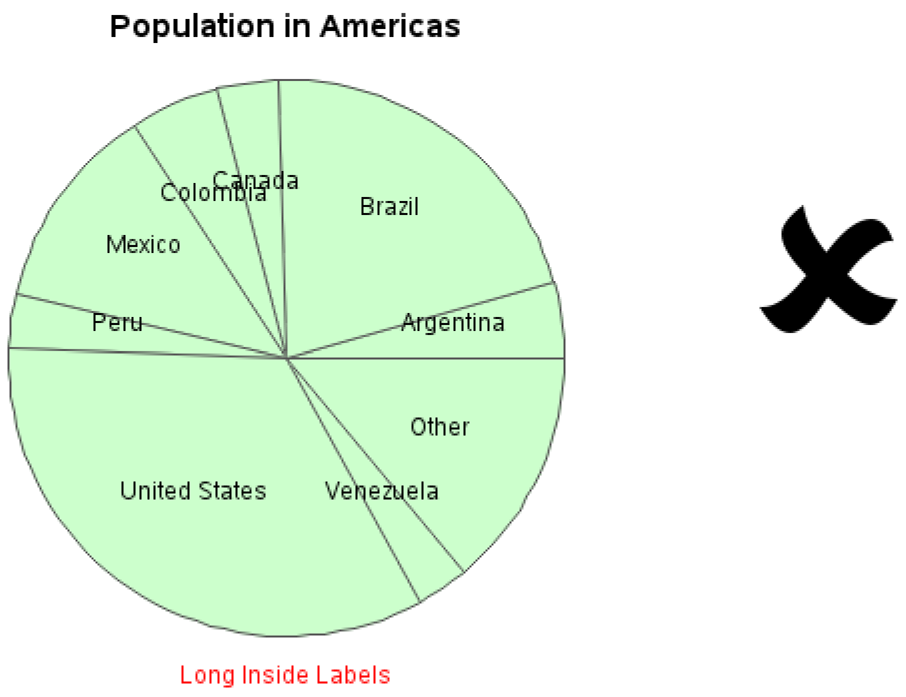
The resulting pie chart is still not great, but this is just the first iteration of improvements.



One of the main problems with the pie chart above is that the country names are long, and therefore the pie has to be sized very small to fit the names and the pie into the available space (in this case 400x350 pixels). As an experiment, let's try placing the country names inside the pie slices, using the SLICE=INSIDE option.

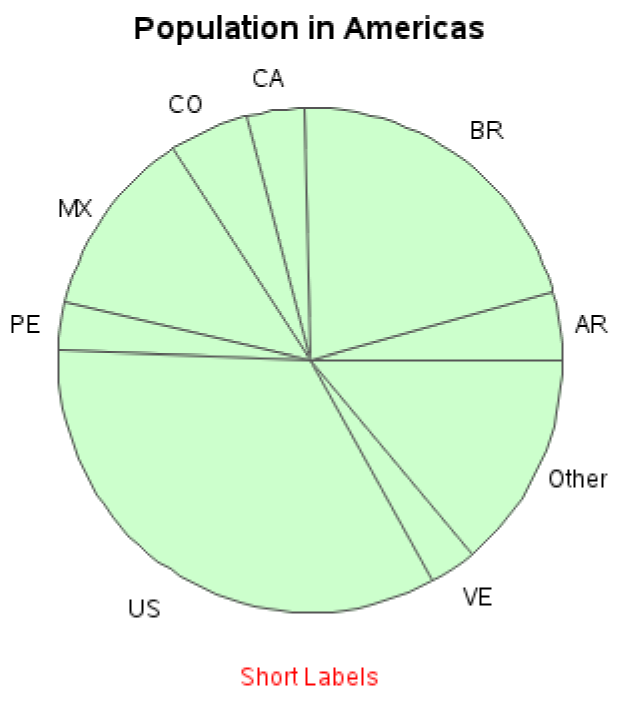
```
title1 ls=1.5 "Population in Americas";
pattern1 v=s c=cxCCFFCC repeat=100;
proc gchart data=my_data;
pie name2 / type=sum sumvar=pop noheader
 slice=inside value=none coutline=gray55
 other=3 otherlabel='Other';
run;
```

The pie itself is bigger now, but the some of the country names are overlapping and the pie still doesn't look good in general.



Perhaps if the country names were shorter, the pie could be even larger, and the text would not overlap. Therefore, let's merge in the 2-character country abbreviations (GLCALPHA2) from the MAPS.NAMES data set, and use them as the slice labels.

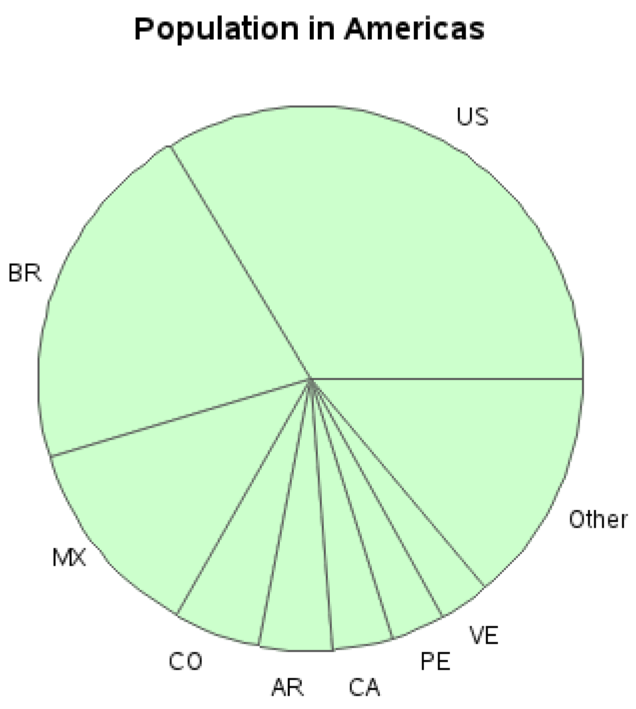
```
proc sql noprint;
create table my_data as
select unique my_data.*, names.GLCALPHA2
from my_data left join maps.names
on my_data.name=names.name;
quit; run;

title1 ls=1.5 "Population in Americas";
pattern1 v=s c=cxCCFFCC repeat=100;
proc gchart data=my_data;
pie GLCALPHA2 / type=sum sumvar=pop noheader
 slice=outside value=none coutline=gray55
 other=3 otherlabel='Other';
run;
```

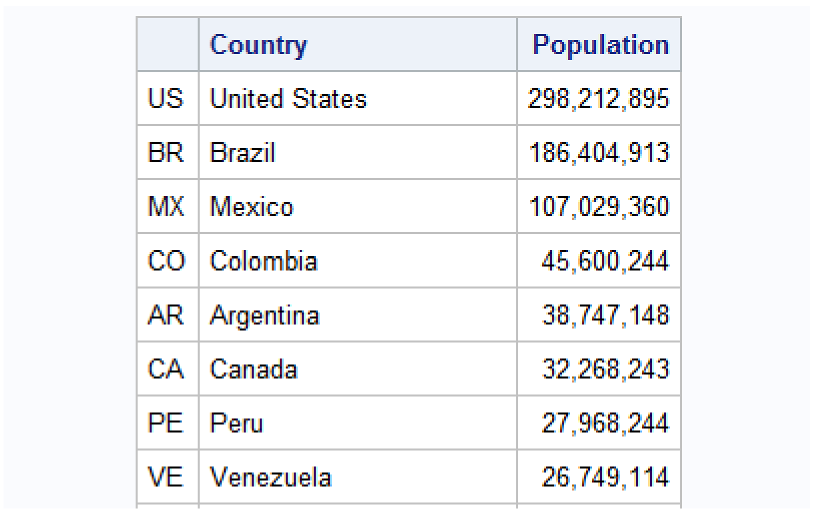


Now we've got a nice large pie, and the labels do not overlap … but the layout of the pie does not really give you much insight into the data, and many people might not know which countries the abbreviations stand for. Therefore, let's sort the pie slices by descending size, and follow the pie chart with a table showing both the country names and abbreviations.

```
title1 ls=1.5 "Population in Americas";
pattern1 v=s c=cxCCFFCC repeat=100;
proc gchart data=my_data;
pie GLCALPHA2 / type=sum sumvar=pop noheader
 slice=outside value=none coutline=gray55
 other=3 otherlabel='Other' descending;
run;


proc sort data=my_data out=my_data;
by descending pop;
run;

title;
proc print data=my_data noobs label;
label GLCALPHA2='00'x name2='Country' pop='Population';
var GLCALPHA2 name2 pop;
run;
```

Now you can easily tell the order/rank of the countries, and if you do not know the country name that goes with the abbreviation you can look it up in the table below the graph (to save space I've just screen-captured a portion of the table below).





	Country	Population
US	United States	298,212,895
BR	Brazil	186,404,913
MX	Mexico	107,029,360
CO	Colombia	45,600,244
AR	Argentina	38,747,148
CA	Canada	32,268,243
PE	Peru	27,968,244
VE	Venezuela	26,749,114

Multiple Pies

If one is good, then more are even better! … So the saying goes.

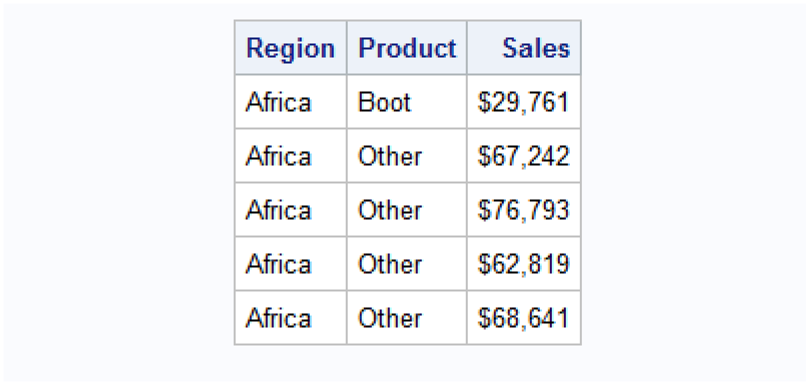
This might, or might not, apply to pie charts – depending on the situation.

There are several ways to create multiple pie charts. We could do it with brute-force, and write multiple blocks of Proc GChart code to produce multiple (separate) pie charts. Or, if there is a convenient variable in the data, we could use a BY statement and generate a pie for each value of that variable (which also produces separate pie charts). Or a third option – we could use the ACROSS and DOWN pie chart options to produce a grid of pie charts, all on the same page. The examples in this section demonstrate the latter two techniques.

For these examples, we'll use the SASHELP.SHOES data. Let's see how much of the shoe sales $ were boots, versus non-boots, in each region. The following code will transform the data such that the PRODUCT variable is 'Boots' or 'Other'.

```
data shoes; set sashelp.shoes;

if product^='Boot' then product='Other';

run;
```

Here are the first few lines of the resulting (transformed) data:



Region	Product	Sales
Africa	Boot	$29,761
Africa	Other	$67,242
Africa	Other	$76,793
Africa	Other	$62,819
Africa	Other	$68,641

In order to use a BY statement, the data should be sorted by the by-variable. Let's use Proc Sort to do that:

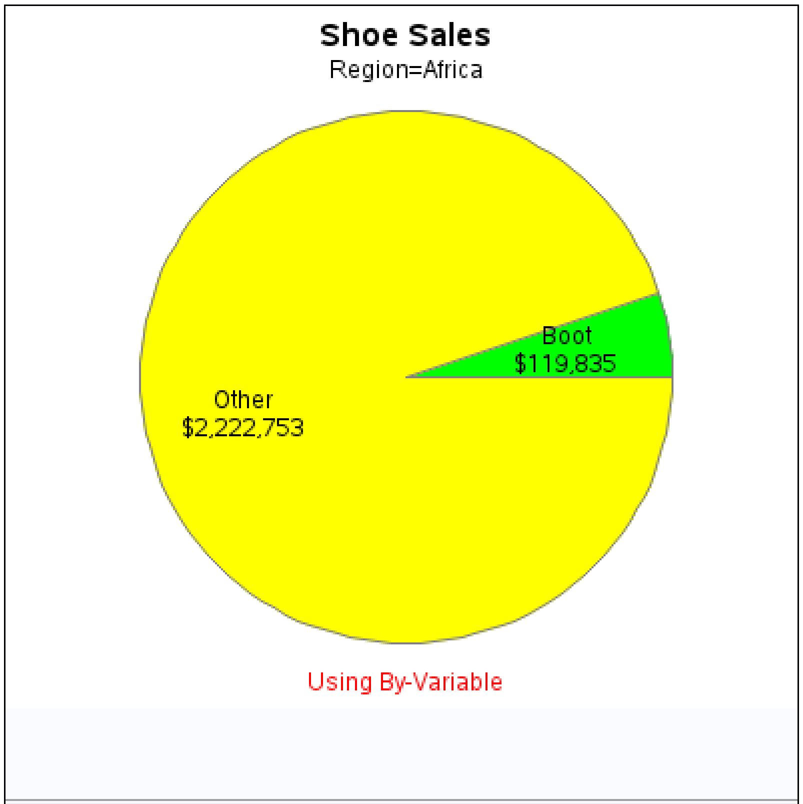
```
proc sort data=shoes out=shoes;

by region;
run;
```

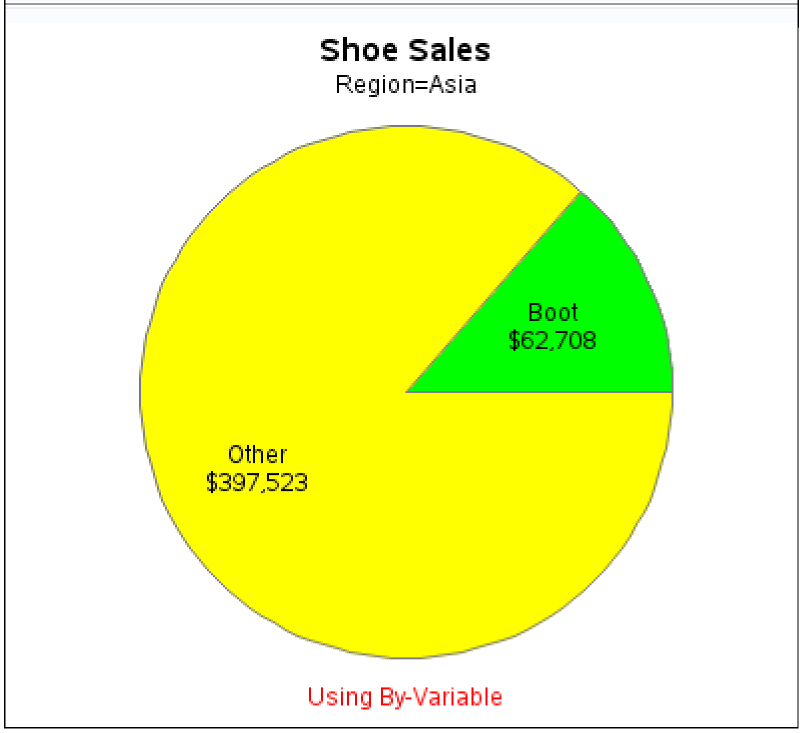
With the sorted data, we can easily generate a separate pie chart for each region using the following code, using a BY statement:

```
title1 ls=1.5 "Shoe Sales";

goptions reset=pattern;

pattern1 v=s c=lime;

pattern2 v=s c=yellow;

proc gchart data=shoes;

by region;

pie product / type=sum sumvar=sales noheader

 slice=inside value=inside coutline=gray;
run;
```

With this code, GChart creates a separate chart for each region, and writes them to the output one after the other. Each pie automatically gets a second title denoting the BY value represented in that chart. The following page shows a screen-capture of the first two pie charts (if you were viewing the actual output, you could scroll through all ten pies, but I'm showing just a small subset).

Using this technique each chart is treated separately, and the charts only go down the page, not across – this can be a big disadvantage when you have small charts, because it wastes a lot of horizontal space. The main title is repeated on each pie chart, and each chart is a separate PNG file.


Shoe Sales
Region=Africa
Boot
$119,835
Other
$2,222,753
Using By-Variable

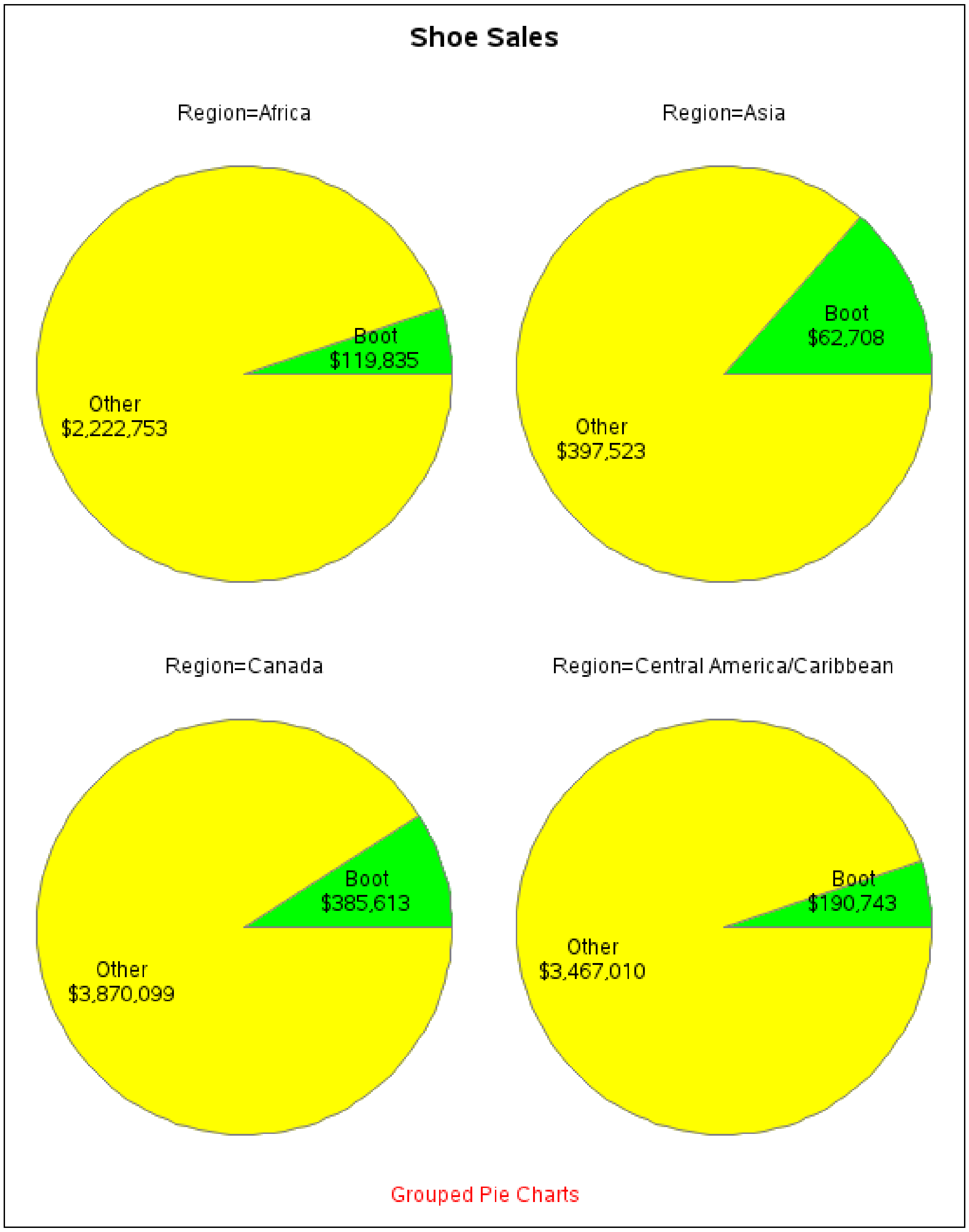

Shoe Sales
Region=Asia
Boot
$62,708
Other
$397,523
Using By-Variable

Now, let’s create a grid of pies, tiling them both across and down to, fit more pies on each page. As you might guess, we use the ACROSS and DOWN options to tell GChart how many pies to try to fit on a page. And rather than using a BY statement (as in the previous example), we use the GROUP. Note that the title is now shown once per page rather than being repeated in each of the individual charts, and the entire page (consisting of 4 pie charts and the title, in this case) is one PNG file rather than a separate PNG file for each chart.

```
title1 ls=1.5 "Shoe Sales";

pattern1 v=s c=lime;

pattern2 v=s c=yellow;

proc gchart data=shoes;

pie product / type=sum sumvar=sales noheader

 slice=inside value=inside coutline=gray

 group=region across=2 down=2;
run;
```

The image on the next page shows the first page of output – it contains four pie charts (two across, and two down), and has a single title for the entire page. This gridded arrangement of the charts is more space-efficient, and allows users to more easily view and compare multiple pies. I’m placing the check-plus here, singe there’s no room for it beside the wide chart ☺

✓+


Shoe Sales
Region=Africa
Boot
$119,835
Other
$2,222,753
Region=Asia
Boot
$62,708
Other
$397,523
Region=Canada
Boot
$385,613
Other
$3,870,099
Region=Central America/Caribbean
Boot
$190,743
Other
$3,467,010
Grouped Pie Charts

3D Pie

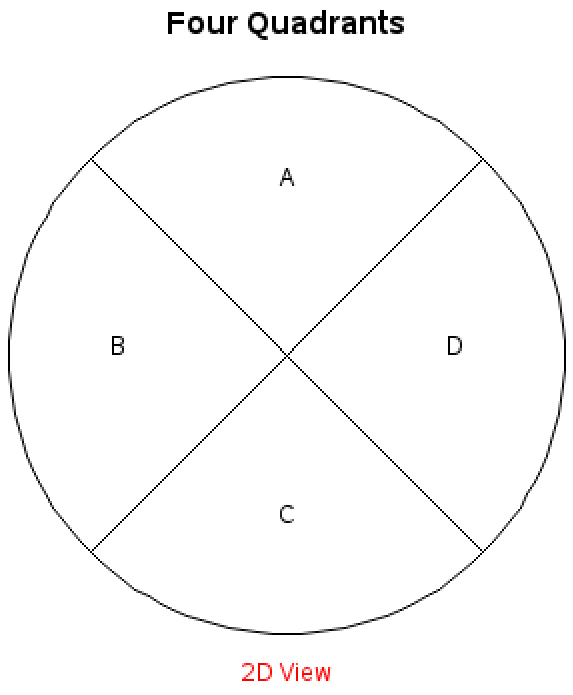
Now that you have learned to create good standard pie charts, you are probably eager to create something fancier – such as 3D pie charts. But please don't do it! This example is not to show you how to create 3D pie charts, but rather to show you why a 3D pie chart is a bad idea.

Let's Talk: This is a trap that almost every beginner falls into -- once they have learned the basics, they want to create fancy 3D graphs. These users eventually figure out that 3D graphs are all 'flash' and no substance, and return their focus to creating better 2D graphs. I encourage you to save a lot of wasted time & effort, and skip the "3D phase" altogether!

To demonstrate the problems with 3D pies, let's start with a simple pie chart containing 4 equal slices -- each slice contains 25% of the pie, or in other words, each slice makes up 90˚ of the 360˚ pie. The actual values used in the data set below are not important, as long as each slice has the same value.

```
data my_data;
quadrant='A'; value=90; output;
quadrant='B'; value=90; output;
quadrant='C'; value=90; output;
quadrant='D'; value=90; output;
run;

title1 ls=1.5 "Four Quadrants";
pattern1 v=s c=white repeat=4;
proc gchart data=my_data;
pie quadrant / type=sum sumvar=value noheader
 angle=45 slice=inside value=none;
run;
```

This creates a very simple 2D pie chart (shown on following page), and we can easily see that each of the slices covers an equal amount of area, and an equal angle. This pie *looks* exactly the way that we know the data to be.

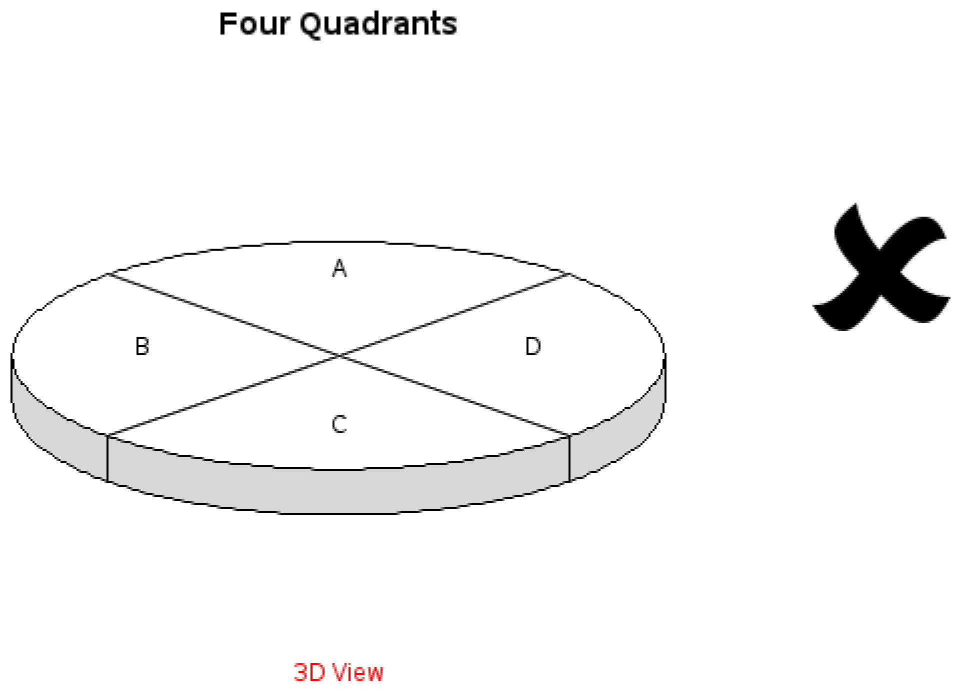


To create a 3D version of this pie chart, all you have to do to change PIE to PIE3D in the code. But that *little* change produces a *BIG* misrepresentation of the data (see the results on the next page).

```
title1 ls=1.5 "Four Quadrants";
pattern1 v=s c=white repeat=4;
proc gchart data=my_data;
pie3d quadrant / type=sum sumvar=value noheader
 angle=45 slice=inside value=none;
run;
```

Let's Talk: I cannot emphasize enough that 3D pies are bad! If you do not believe me, then do a search in your web browser for keywords such as: 3D, pie, misrepresent. You will find many experts giving you the same advice, and you will see many examples showing how 3D pie charts misrepresent the data.

With the 3D pie, you lose most of the visual cues that made a pie chart useful in the first place. For example, both the angles and the areas are now distorted. Since the 3D pie is displayed from a perspective view, the angles of the pie slices are no longer 90˚ - slices A and C look bigger than 90˚, and slices B and D look smaller. It is also difficult to visually compare the areas of the slices, since the proportions of A and C are now shorter and wider than B and D.


Four Quadrants
A
B
D
C
3D View