

Paper PO-12

Using Proc Gplot and Proc Reg Together to Make One Great Graph

William 'Gui' Zupko II, U.S. Census Bureau, Suitland, MD

ABSTRACT:

Regression is a helpful statistical tool to show relationships between two or more variables. However, many users can find the barrage of numbers at best unhelpful, and at worst, undecipherable. Using the shipments and inventories' historical data from the U.S. Census Bureau's office of Manufacturers' Shipments, Inventories, and Orders (M3), we can create a graphical representation of two time series with proc gplot and map out reported and expected results. By combining this output with results from proc reg, we are able to highlight problem areas that may need a second look. The resulting graph shows which dates have abnormal relationships between our two variables and presents the data in an easy to use format which even users unfamiliar to SAS® can interpret. This graph is ideal for analysts finding problematic areas such as outliers and trend-breakers or for managers to quickly discern complications and the effect they have on overall results.

INTRODUCTION:

Proc gplot is a valuable statistical tool that graphically represents relationships between two items. However, individual problems can be difficult to spot on the graph, as well as to identify specific observations. Proc reg is a useful way to identify outliers and determine other statistical information, but the output can be dense and difficult to interpret. By combining these two outputs, we can create a graph that highlights potential problem areas and represents trends and other valuable information.

To demonstrate this point, we will be using public data from the U.S. Census Bureau's survey of Manufacturers' Shipments, Inventories, and Orders (M3). M3 is an economic indicator survey that reports data from the Manufacturing sector (www.census.gov/m3). While studying relationships between two items, the total value of shipments (vs_total) and total value of inventories (ti_total) from January 1992 to April 2011, we can map out the data to determine trends and identify possible outliers. Although we use this program on confidential data, we are able to show the same effect with data released to the public.

We will be using the following data, seasonally adjusted in millions of dollars. Table 1 is a sample of the 5 observations of the 232 observations in the universe:

Table 1

Period	TI_total	Period	VS_Total
Jan-1992	377,747	Jan-1992	227,886
Feb-1992	376,638	Feb-1992	228,960
Mar-1992	377,648	Mar-1992	238,575

Apr-1992	377,531	Apr-1992	239,888
May-1992	377,654	May-1992	243,627

This data can be found at the U.S. Census Bureau's M3 website under the historical data link. The link is http://www.census.gov/manufacturing/m3/historical_data/index.html. Unfortunately, this dataset can be difficult to read. Accordingly, I used a new site, the Search Databases system, that allows the data to be formatted to a specific user's needs. This database can be found at <http://www.census.gov/econ/currentdata/search/m3?programCode=M3&geoLevelCode=US&yearStart=1992&yearEnd=2011&categoryCode=MTM&dataTypeCode=VS&adjusted=1¬adjusted=0&errorData=0submit=GET+DATA>.

Fig. 1

Source: Manufacturers' Shipments, Inventories and Orders (Definitions)
Total Manufacturing — Seasonally Adjusted Value of Shipments (Millions of Dollars)
 TXT XLS-V XLS-H

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1992	227,886	228,960	238,575	239,888	243,627	245,708	245,763	242,836	244,603	243,490	247,071	248,163
1993	247,798	249,667	250,209	248,805	251,941	252,936	249,094	249,398	252,572	256,506	255,374	255,982
1994	260,272	261,247	261,706	263,348	266,838	267,949	271,236	273,787	272,880	276,547	279,426	286,836
1995	287,105	288,543	284,558	289,663	285,586	288,701	287,937	289,702	295,112	292,999	291,615	300,577
1996	287,221	282,396	294,270	295,746	299,515	303,345	299,811	301,541	305,512	301,295	308,479	307,658
1997	306,977	314,531	315,603	317,686	315,287	320,528	322,600	322,149	323,819	322,226	328,562	326,506
1998	321,169	329,667	328,451	324,302	325,284	319,938	315,796	323,818	325,481	327,564	328,584	328,455
1999	328,480	332,452	328,838	330,167	335,194	333,653	334,244	340,453	338,604	343,111	343,605	344,306
2000	352,338	340,160	349,152	354,512	350,027	355,114	351,491	347,221	355,737	349,286	346,501	350,443
2001	342,474	346,204	340,459	330,526	338,811	330,972	327,969	330,804	321,520	320,508	319,528	322,291
2002	320,292	319,887	321,723	326,698	328,712	327,930	325,813	329,800	331,563	330,077	330,039	324,698
2003	329,297	332,730	337,001	325,544	328,434	332,584	336,592	332,740	340,301	340,496	340,123	340,947
2004	339,466	339,840	354,606	350,930	353,062	356,104	355,761	362,223	363,487	369,119	373,258	376,479
2005	382,149	382,258	385,607	390,747	389,197	391,153	388,728	398,153	405,096	405,681	408,000	415,856
2006	416,409	414,458	419,071	412,626	423,338	423,041	415,986	423,874	412,905	410,528	415,622	429,422
2007	419,566	427,395	435,243	438,289	444,854	442,764	445,213	445,091	446,177	452,258	462,938	461,934
2008	467,518	461,513	461,387	474,957	475,426	481,484	485,305	467,038	453,849	434,235	399,940	382,988
2009	369,445	370,864	360,570	357,454	356,007	362,735	367,357	368,630	374,858	378,727	383,894	389,564
2010	391,192	389,580	397,323	400,920	396,819	393,959	402,458	401,696	405,645	408,082	412,779	423,543
2011	431,064	431,886	445,386	444,522	NA							

The data can then be put into an Excel or text file in the horizontal format seen here in Figure 1, or use the XLS-V option, which formats the data into a vertical format. This last option is the one that I chose for this paper.

SETTING UP THE DATA:

In order to use proc reg output in proc gplot, we need to format this output properly. There are two ways to put that output into the graph; with a macro variable or by formatting the data. If you need to use the data in the title, header, or footnotes, then it is better to create a macro variable. If you need the information in the graph, then it is better to format the data in the dataset. For either of these ways to work, we will need to run proc reg. The following code runs proc reg and also creates a dataset which we can use to format our data.

```
proc reg data=mergevsti;
    model ti_total=vs_total;
        output out=regvs_totalti_total rstudent=rstud ;
quit;
```

This code sets the variable vs_total as the independent variable and ti_total as the dependent variable. The output is put into a new dataset and includes the variable rstud, which is the studentized residual. The studentized residual for an observation is its residual divided by the estimate of the standard deviation of the residuals.

Now that we have fit a model, we need to test some of our assumptions as well, such as errors being normally distributed and having constant variance. The best way to do this is with proc univariate.

```
proc univariate data=regvs_totalti_total normaltest ;
var rstud;
run;
```

One test for normality is if the p values for the tests for normality of the studentized residuals are greater than .05. Our test shows that the p value is less than that, so this shows that some assumptions for a linear regression are not met. However, since we are only looking for outliers, this might not matter. Appendix 3 shows how the test for normality can be incorporated into the final graph.

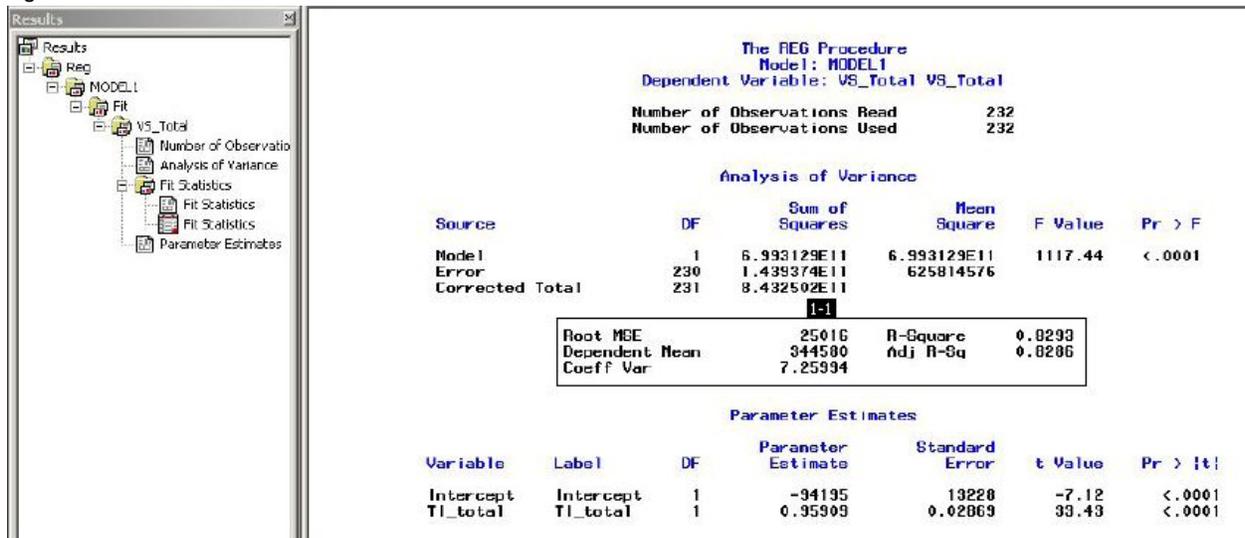
CREATE A MACRO VARIABLE:

The macro variable that I am creating contains the r^2 statistic that is included in proc reg's output. By using ODS Output, we can convert the output that is printed in the output window and put it in a DATA step instead. The following code will only print the output subgroup Fit Statistics. We can determine the name by running proc reg first and determining which result has the information we need.

```
ods output "Fit Statistics"=fit_total;
```

The highlighted area in Figure 2 is the part of the output that is created in the dataset.

Fig. 2



The ODS Output function creates a dataset called fit_total. The dataset does not look like it does in the output window, but one very nice feature included in this is a number value and a character value, denoted by nvalue and cvalue. In order to determine which values we need, we have to take a look at the actual dataset. In this case, we need the cvalue2, which we can easily determine after looking at the dataset, shown in Figure 3.

Fig. 3

	Model	Dependent	Label1	cValue1	rValue1	Label2	cValue2	n
1	MODEL1	VS_Total	Root MSE	25016	25016	R-Square	0.8293	
2	MODEL1	VS_Total	Dependent Mean	344580	344580	Adj R-Sq	0.8286	
3	MODEL1	VS_Total	Coeff Var	7.25994	7.259938			

Now that we have created our dataset, we can create a macro variable that contains the r^2 variable. Call `symputx` is a great function to retrieve the variable from the dataset and put it into the macro library.

```
data _null_;
    set fit_total;
        if label2='R-Square' then call symputx("rq",cvalue2);
run;
```

After the DATA step is run, a new macro variable called `rq` has been created that can then be used anywhere else in the program.

CREATE A NEW VARIABLE:

Creating a new variable allows us to show the data in the actual graph. If we want to show or highlight data in our final graph, a new variable needs to be created that only has data if certain criteria are met. In our case, we are looking for observations where the studentized residual, `rstud`, has an absolute value greater than 2. These are observations that we are setting as outliers or deserving more attention. The 2 comes from the fact that the studentized residuals follow a t-distribution with $n-p-1$ degrees of freedom, where n is the number of data points and p is the number of independent variables (in our case $p = 1$). Therefore, by chance alone, we'd expect approximately 5% of the studentized residuals to have an absolute value greater than 2, which we consider a rare event deserving of further attention. Note that any criteria can be user defined with an if-then statement.

```
data regvs_totalti_total;
    set regvs_totalti_total;
        if abs(rstud)>2 then ti_totalp=ti_total;
run;
```

In order to use this variable, the value of `ti_total` is put into a new variable called `ti_totalp`. This variable only has data if the criteria are met.

FORMAT THE GRAPH:

Now that our data is in an appropriate format, we can begin to format our graph. In order for us to format the graph, we should first reset all graph options to their default settings. If we do not reset the settings, SAS® can get confused and start putting different graph commands into the wrong graphs. Afterwards, we can put our own graph commands in to format the graph to our specific needs.

The `symbol` command controls the type of symbols that appear. `Symbol1` is the symbol that I used to designate observations that I desire to look at, such as outliers or trends. `Symbol2` is the symbol that I used to display all of the observations.

`Symbol1` has no value, since it will receive a value from the `symbol2` command. However, by using the `pointlabel` command, we can designate specific observations to show not a dot or some other symbol, but the actual information contained in the observation. We do this by setting the `pointlabel` to display the `#period`, which gives the period information for that specific observation. `Symbol2` has a value of `%`, which gives the appearance of a shamrock. By creating an option `interpol`, we are able to see a line showing the regression equation and 95% confidence intervals for the predicted values. Finally, we set the color to green, because shamrocks are always green.

```

goptions reset=global;

symbol1 value=none pointlabel=(position=middle '#period' color=red)
      color=red ;
symbol2 value=% interpol=rlclm95 color=green;

```

Also, to make the graph a little easier to read, we need to format the vertical and horizontal axes. Although the horizontal axis only requires a title, the vertical axis requires a different orientation to provide more room for the graph. Also, because we have two graphs in one with the same horizontal axis, the vertical axis is different. It is easier to provide a title for the variable than to have SAS® pick one.

```

axis1 label=(a=90 'Total Inventories');
axis2 label=('Value of Shipments');

```

Finally, we need to have a title so that people know what our graph is for. This is our first chance to use our new macro variable that we created from the r^2 . The macro variable is in the title2. So, by pulling the macro variable from proc reg, we have now used some of our proc reg statistics in our proc gplot.

```

title1 "Relationship between Total Inventories and Value of Shipments January
1992-April 2011";
title2 "R-squared is &rq";

```

FINALLY, THE GRAPH:

Since gplot is a graphing procedure, graphs automatically are put into the catalog GSEG. However, if you want more control over the graphs or to put them in another catalog, the gout command will create a new catalog for the data to go to or simply put the newest graphs in the existing catalog. This will allow more control over data and also reduce the chance of accidentally deleting it. However, if the catalog is not deleted before a new graph is created, then the catalog can quickly be overrun by many graphs that are essentially the same. To control this overrun, it can be helpful to delete the catalog before or after each run. If we do not want to manually open the requisite library and delete the catalog after each run, we can do so with proc datasets.

```

proc datasets lib=work memtype=catalog;
      delete regression;
run;
quit;

```

I put this code at the beginning of the program because I want to make sure that the catalog is deleted before a new graph is created. This helps with maintenance, especially if the catalog is not temporary like the work library.

Now that we have formatted our dataset and our graph, we can finally create the graph. For the most part, this will be a standard proc gplot. However, for the combined data to work, we are going to need to create an overlay, or put two graphs together into one. As one graph will have all available data, the second will only be populated with outliers or specific observations. These two graphs are combined with the overlay function.

```

proc gplot data=regvs_totalti_total gout=regression;
plot ti_totalp*vs_total=1 ti_total*vs_total=2 /overlay vaxis=axis1
haxis=axis2;
bubble ti_totalp*vs_total=rstud/blabel vaxis=axis1 haxis=axis2;

```

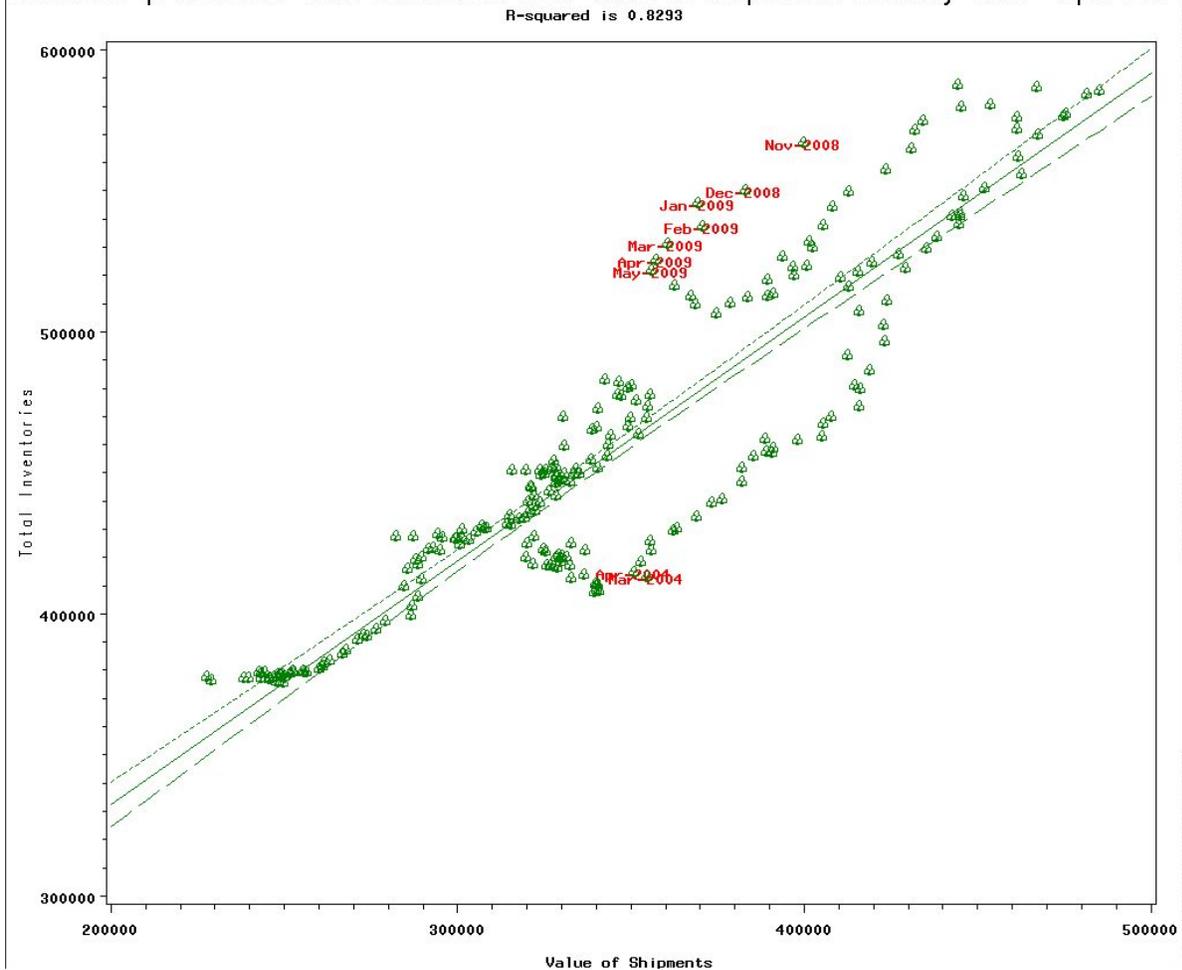
```
run;
quit;
```

Because we have two graphs, each one can have its own symbol. In order to show which symbol will be with which graph, the equal sign designates the appropriate symbol. In this case, the outliers were given the symbol1, while all observations were given symbol2.

The overlay command melds the two graphs together. If the overlay was not there, two separate graphs would have been created. The vaxis command and the haxis command finish the graph formatting that we planned earlier in the paper. And just like that, we have created a graph!

Fig. 4

Relationship between Total Inventories and Value of Shipments January 1992—April 2011



Now, we can see that the green color for all observations is appearing, as well as the highlighted areas. However, while it is easy to follow the progression in some areas, after 2006 the numbers get very difficult to discern. So, what we need are references to better determine the path of the information.

Unfortunately, using our pointlabels and overlays makes it so that we can only compare two plots at once. So, in order to add yet more information in, we will need at least one more model to plot. We can accomplish this by using the plot2 command. The plot2 command will allow a second model to be plotted using the same horizontal axis as before. One additional variable, ti_total, was created by showing the date for each year in January and the shamrocks were replaced with lines connecting the observations.

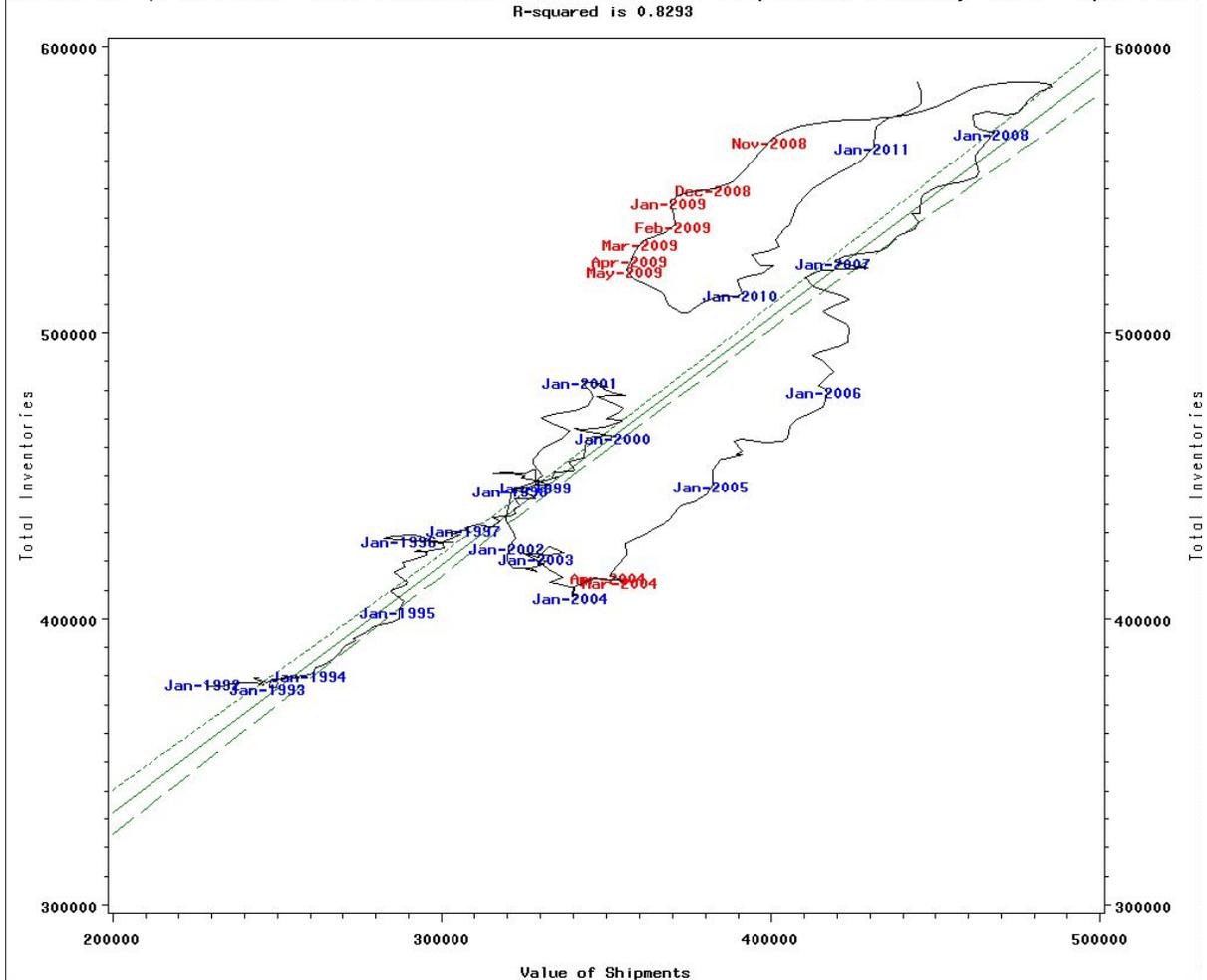
Now that we have four different symbols to identify, we need to expand our options for each symbol. Since we do not actually need any points to show up except for our pointlabels, all our values are set to none. Additionally, we are using the Interpol=spline option, that makes a smooth line throughout the observations.

```
proc gplot data=regvs_totalti_total gout=regression;
plot  ti_totalp*vs_total=1 ti_total*vs_total=2 /overlay vaxis=axis1
haxis=axis2;
plot2 ti_totallr*vs_total=3 ti_total*vs_total=4 /overlay vaxis=axis1
haxis=axis2;
run;
quit;
```

The only change to this code is the plot2 line. Additional symbols were inputted, and the full code can be found in appendix 2. The following graph, Figure 5, shows these changes and how the graph becomes easier to read and follow.

Fig. 5

Relationship between Total Inventories and Value of Shipments January 1992–April 2011



One difference that I wanted to show is that since we are melding four graphs into one, another set of vertical axis labels appears. Since our vertical axis is the same on both sides, they both look the same. However, this can be used to compare graphs along different vertical scales as well.

This graph also highlights a problem the pointlabel can have. If two observations are close to one another, e.g. Jan. 1998 and Jan 1999, then there can be overlap on the graph. Also, Jan 2009 already appears in red, so we would not want to also get a label to show in blue as well. The way the code is written, if both observations were to appear, these observations would have appeared in blue, which would have been considered of lesser importance to being considered an outlier.

So, in the end, the programmer either decides what is important or lets SAS® decide. Almost everything with proc gplot is customizable to some degree. We were able to add output from proc reg into the graphic or into the title. This gives us many options like adding the regression equation into the title (see Appendix 3) or other information that might be considered important.

CONCLUSION:

Proc gplot is a great way to show relationships between two variables in a way that anybody can interpret, regardless of their SAS® experience. This is a great medium for presentations, papers, or reports. With the techniques discussed in this paper, we are able to highlight important points or provide pertinent information for management or analysts. This provides a quick and comprehensive look at a dataset so that attention can be focused on cases that break the trend or are deemed important in some way.

ACKNOWLEDGEMENTS:

I would like to thank the U.S. Census Bureau and SAS® for this opportunity to write and present this paper. I would also like to thank Jan Lattimore, Justin Smith, Anne Linonis, and Danielle Corteville, at the U.S. Census Bureau, who took time to edit and check this paper to make sure that it was factually and grammatically sound.

SAS® and all other SAS® Institute Inc. product or service names are registered trademarks or trademarks of SAS® Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION:

William 'Gui' Zupko II

U.S. Census Bureau

4700 Silver Hill Road

Suitland, MD 20746

William.e.zupko.ii@census.gov

APPENDIX 1: ORIGINAL PROGRAM

```
proc printto new log='C:/SESUG 2011/log.log' ;
run;
/*data downloaded from
http://www.census.gov/econ/currentdata/search/m3?programCode=M3&geoLevelCode=
US&yearStart=1992&yearEnd=2011&categoryCode=MTM&dataTypeCode=VS&adjusted=1&no
tadjusted=0&errorData=0submit=GET+DATA*/
proc import datafile='H:/total vs.xls' out=tvsv dbms=excel2000 replace;
run;
proc import datafile='H:/total ti.xls' out=tti dbms=excel2000 replace;
run;
data mergevsti;
merge tvsv tti;
run;
proc datasets lib=work memtype=catalog;
delete regression;
run;
quit;

title1;
ods output "Fit Statistics"=fit_total;
proc reg data=mergevsti ;
model ti_total=vs_total;
output out=regvs_totalti_total rstudent=rstud ;
```

```

quit;
ods output close;
data _null_;
    set fit_total;
    if label2='R-Square' then call symputx("rq",cvalue2);
run;
data regvs_totalti_total;
    set regvs_totalti_total;
    if abs(rstud)>2 then ti_totalp=ti_total;
run;
/*Graph Options*/
goptions reset=global;

symbol1 value=none pointlabel=(position=middle '#period' color=red) color=red
;
symbol2 value=% interpol=rlclm95 color=green;

axis1 label=(a=90 'Total Inventories');
axis2 label=('Value of Shipments');

title1 "Relationship between Total Inventories and Value of Shipments January
1992-April 2011";
title2 "R-squared for &titlevar is &rq";
/*Create Graph*/
proc gplot data=regvs_totalti_total gout=regression;
plot ti_totalp*vs_total=1 ti_total*vs_total=2 /overlay vaxis=axis1
haxis=axis2;
bubble ti_totalp*vs_total=rstud/blabel vaxis=axis1 haxis=axis2;
run;
quit;
title1;
proc printto;
run;
/*Prints Log Problems*/
filename guilog 'h:/log.log';
data guizer;
infile guilog lrecl=200 pad ;
input @1 intext $200. ;
run;
title1 'Log Errors';
proc print data=guizer;
where index(intext,'ERROR:') or index(intext,'WARNING:');
run;

```

APPENDIX 2: REVISED GRAPH

```

proc printto new log='C:/SESUG 2011/log.log' ;
run;
/*data downloaded from
http://www.census.gov/econ/currentdata/search/m3?programCode=M3&geoLevelCode=
US&yearStart=1992&yearEnd=2011&categoryCode=MTM&dataTypeCode=VS&adjusted=1&no
tadjusted=0&errorData=0submit=GET+DATA*/
proc import datafile='H:/total vs.xls' out=tvsv dbms=excel2000 replace;

```

```

run;
proc import datafile='H:/total ti.xls' out=tti dbms=excel2000 replace;
run;
data mergevsti;
    merge tvs tti;
run;
proc datasets lib=work memtype=catalog;
    delete regression;
run;
quit;

title1;
ods output "Fit Statistics"=fit_total;
proc reg data=mergevsti ;
    model ti_total=vs_total;
    output out=regvs_totalti_total rstudent=rstud ;
quit;
ods output close;
data _null_;
    set fit_total;
    if label2='R-Square' then call symputx("rq",cvalue2);
run;
data regvs_totalti_total;
    set regvs_totalti_total;
    if abs(rstud)>2 then ti_totalp=ti_total;
    else if index(period,'Jan') then ti_totalr=ti_total;
run;
/*Graph Options*/
goptions reset=global;

symbol1 value=none pointlabel=(position=middle '#period' color=red) color=red
;
symbol2 value=none interpol=rlclm95 color=green ;
symbol3 value=none pointlabel=(position=middle '#period' color=blue)
color=blue ;
symbol4 value=none interpol=spline color=black;

axis1 label=(a=90 'Total Inventories');
axis2 label=('Value of Shipments');

title1 "Relationship between Total Inventories and Value of Shipments January
1992-April 2011";
title2 "R-squared for &titlevar is &rq";

/*Create Graph*/
proc gplot data=regvs_totalti_total gout=regression;
plot ti_totalp*vs_total=1 ti_total*vs_total=2 /overlay vaxis=axis1
haxis=axis2;
plot2 ti_totalr*vs_total=3 ti_total*vs_total=4 /overlay vaxis=axis1
haxis=axis2;
run;
quit;

```

```

title1;
proc printto;
run;
/*Prints Log Problems*/
filename guilog 'h:/log.log';
data guizer;
infile guilog lrecl=200 pad ;
input @1 intext $200. ;
run;
title1 'Log Errors';
proc print data=guizer;
where index(intext,'ERROR:') or index(intext,'WARNING:');
run;

```

APPENDIX 3: REGRESSION EQUATION ADDED TO THE TITLE AND NORMALITY TEST IN FOOTER

```

proc printto new log='C:/SESUG 2011/log.log' ;
run;
/*data downloaded from
http://www.census.gov/econ/currentdata/search/m3?programCode=M3&geoLevelCode=
US&yearStart=1992&yearEnd=2011&categoryCode=MTM&dataTypeCode=VS&adjusted=1&no
tadjusted=0&errorData=0submit=GET+DATA*/
proc import datafile='H:/total vs.xls' out=tvsv dbms=excel2000 replace;
run;
proc import datafile='H:/total ti.xls' out=tti dbms=excel2000 replace;
run;
data mergevsti;
merge tvsv tti;
run;
proc datasets lib=work memtype=catalog;
delete regression gseg;
run;
quit;

title1;
ods output "Fit Statistics"=fit_total "Parameter Estimates"=par_est "Tests
For Normality"=tnormal;
proc reg data=mergevsti ;
model ti_total=vs_total;
output out=regvs_totalti_total rstudent=rstud ;
quit;
ods output close;
ods output "Tests For Normality"=tnormal;
proc univariate data=regvs_totalti_total normaltest ;
var rstud;
run;
ods output close;
data _null_;
set fit_total;
if label2='R-Square' then call symputx("rq",cvalue2);
run;
data _null_;

```

```

        set par_est;
        if variable='Intercept' then call symputx("int",estimate);
        else call symputx("vs_est",estimate);
run;
data _null_;
    set tnormal;
    if _n_=1 and pvalue>.05 then call symputx("norms","Pass");
    else call symputx("norms","Fail");
run;
data regvs_totalti_total;
    set regvs_totalti_total;
    if abs(rstud)>2 then ti_totalp=ti_total;
    else if index(period,'Jan') then ti_totalr=ti_total;
run;
/*Graph Options*/
goptions reset=global;

symbol1 value=none pointlabel=(position=middle '#period' color=red) color=red
;
symbol2 value=none interpol=rlclm95 color=green ;
symbol3 value=none pointlabel=(position=middle '#period' color=blue)
color=blue ;
symbol4 value=none interpol=spline color=black;

axis1 label=(a=90 'Total Inventories');
axis2 label=('Value of Shipments');

title1 "Relationship between Total Inventories and Value of Shipments January
1992-April 2011";
title2 "R-squared is &rq";
title3 "Regression Equation: Inventories=%substr(&vs_est,1,5)*Shipments +
%scan(&int,1,.)";
footnotel "Test For Normal Distribution: &Norms";
/*Create Graph*/
proc gplot data=regvs_totalti_total gout=regression;
plot ti_totalp*vs_total=1 ti_total*vs_total=2 /overlay vaxis=axis1
haxis=axis2;
plot2 ti_totalr*vs_total=3 ti_total*vs_total=4 /overlay vaxis=axis1
haxis=axis2;
run;
quit;
title1;
footnotel;
proc printto;
run;
/*Prints Log Problems*/
filename guilog 'h:/log.log';
data guizer;
infile guilog lrecl=200 pad ;
input @1 intext $200. ;
run;
title1 'Log Errors';

```

```
proc print data=guizer;
where index(intext, 'ERROR:') or index(intext, 'WARNING:');
run;
```

Fig. 6

Relationship between Total Inventories and Value of Shipments January 1992—April 2011

