Paper 363-2013

A Beginner's Introduction to an Idiot's Guide to PROC TEMPLATE and GTL for Dummies

Christopher Battiston, SickKids, Toronto, Ontario, Canada

ABSTRACT

The aim of this paper will be an extremely gentle introduction to the very exciting and somewhat intimidating world of PROC TEMPLATE and Graph Template Language (GTL). As these two SAS® features are still relatively new, not many people have had time to learn to learn them and see what they are capable of accomplishing with minimal effort.

INTRODUCTION

"The SAS®/GRAPH *Graph Template Language* (GTL) is an extension to the Output Delivery System (ODS) that enables you to *create sophisticated analytical graphics that are not available from traditional SAS®/GRAPH procedure statements*…The GTL templates are defined with PROC TEMPLATE. The GTL includes conditional statements that can be used to determine what graph features are rendered, layout statements that specify the arrangement of graph features, plot statements that request specific plot types (such as histograms and scatter plots), and text and legend statements that specify titles, footnotes, legends, and other text-based graph elements... "

- Taken from Support.SAS.com page on Graph Template Language

Having used SAS® for about 2 years now, I still consider myself a newbie. I am literally learning something new everyday and so coming to these Conferences allows me to expand my very large horizons, connect with people that I can contact for assistance, and see different ways of approaching the same problems that I have. However, two areas that I've really been struggling with learning are PROC TEMPLATE and GTL. I don't like to admit defeat, but for some reason I was having a very difficult time picking up these two very daunting features in SAS®. The resources online were extensive, but every time I started reading one, I found I was so overwhelmed by the amount of information, I forgot what I was trying to learn. So, the idea came to me one weekend – I need to figure out how to use these in my day-to-day SAS® programming, and so I started just taking very basic notes – literally baby stepping my way gently into this vast area of templates, graphics, outputs, and more options than I know what to do with. When I was considering submitting this for SAS® Global Forum, I struggled with a title; it came to me as I was falling asleep one night. Being a huge fan of the "For Dummies" books, I've always liked their approach of starting with the fundamentals and working up. However, I wanted to show that this was even gentler; that's where the "An Idiot's Guide" came in. The "A Beginner's Introduction" was the final piece in the title and was put there for an over-the-top emphasis that this was about as basic as you can get. No, I'm not going to be showing you "this is a keyboard" or "this is how to start SAS®", I assume you already have that down pat. This is going to assume preliminary knowledge of SAS® PROCs and the DATA step, and will work from there. I must stress that I am learning these at the same time, and so any misunderstandings, mistakes or misinterpretations of any information are mine and mine alone. **This paper is not intended to be used for proper methods of data visualization. I intentionally keep the graphs simple, so expected data management like sorting are not done.**

WHAT IS SO DIFFERENT ABOUT PROC TEMPLATE?

Where PROCs like PROC FREQ, PROC UNIVARIATE, or PROC MEANS have one set of statements and options, PROC TEMPLATE is more like a tool-belt of things you can do with your data or your output. Unlike other PROCs that impact the data in a very specific way, PROC TEMPLATE allows you to change how your data looks at every point of your analysis, from the table for summary statistics to tagsets that allow you to include markup language tags for your output destination to your graphs being designed specifically to meet *your* needs. This paper will only be focusing on the Graphics part of GTL/PROC TEMPLATE as these can be the most confusing for people but also the most useful.

OVERVIEW OF PROC TEMPLATE AND THE GRAPHIC TEMPLATE LANGUAGE (GTL)

OK so let's just go ahead and start digging into these mysteriously powerful SAS® functions. The first critical point is to understand what happens when you use PROC TEMPLATE. Here's a very basic example; don't worry about the specifics just yet as we'll go through that in the next section.

```
proc template;
```

```
define statgraph sashelp.class;
begingraph;
       layout overlay;
              scatterplot x=height y=weight;
       endlayout;
endgraph;
end;
run;
```

When I run this, it appears nothing happens; there is no results window that opens, no dialog boxes appear. What did I do? If you go into the Log, you will see the following:

```
417   proc template;
NOTE: Writing HTML Body file: sashtml.htm
418   define statgraph sashelp.class;
419   begingraph;
420       layout overlay;
421       scatterplot x=height y=weight;
422       endlayout;
423       endgraph;
424       end;
NOTE: STATGRAPH 'Sashelp.Class' has been saved to: SASUSER.TEMPLAT
425       run;
NOTE: PROCEDURE TEMPLATE used (Total process time):
      real time           1.78 seconds
      cpu time            0.62 seconds
```

OK, so it looks like my code *did* work. And in fact what it did was save the template to the SASUSER.TEMPLAT folder, which means that you can now call this template when you run the other PROC that goes with this set, PROC SGRENDER (see below). I named the Template the same as the Data set, just to keep things simple. In fact you can (and should) have very specific names for your templates; as you build more and more into your library, it may become harder to find the one you want. Calling a template "OutlierDetection" is clearer than calling it "Template2".
I mentioned that there's another PROC that you have to be aware of, and that's PROC SGRENDER. What SGRENDER does is generate the graph based on your template, merging it with your data. Because we specified that we wanted a SCATTERPLOT with x = height and y = weight, we don't have to redo that. All we need is:

```
proc sgrender data=sashelp.class template=sashelp.class;
run;
```

Now, I know what you're thinking – why would I run two PROCs to get a scatter plot when I do it in one all the time? Remember what the quote said at the beginning – "*…create sophisticated analytical graphics that are not available from traditional SAS®/GRAPH procedure statements…".* So no, you wouldn't use PROC TEMPLATE for something basic like a standard scatter plot. However, I dare you to try and accomplish the scatter plot below, again using the SASHELP.CLASS data set but with a whole pile of features added (code provided in Appendix A). Including the PROC SGRENDER, this was 19 lines of SAS® Code.

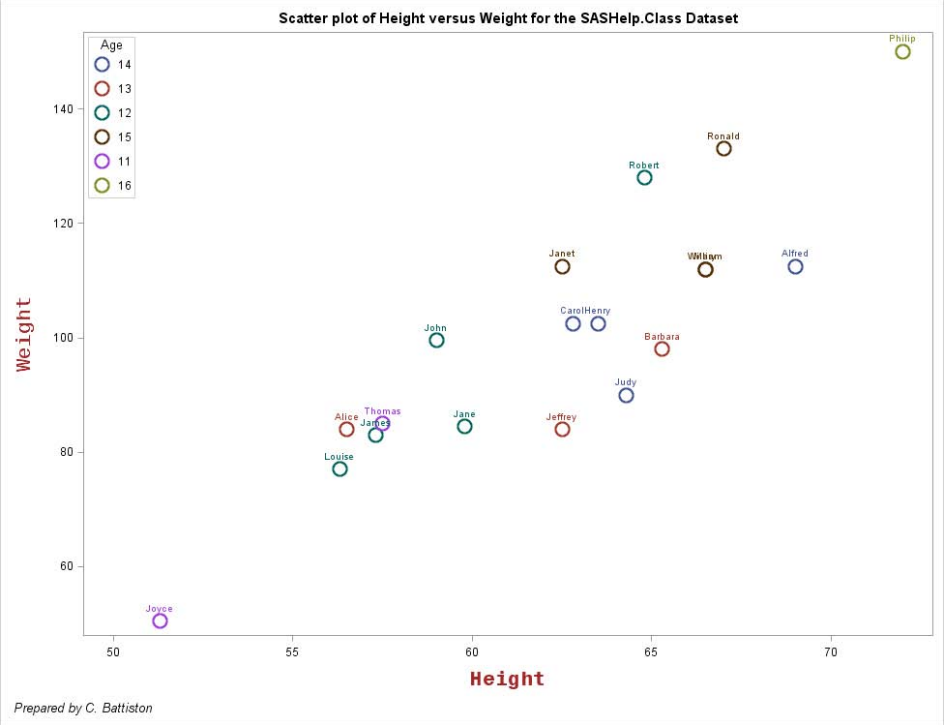

Figure 1. A Complex Scatter Plot Created with PROC TEMPLATE

So let's get into a couple of basic examples, and work through the code to see what the code does.

SOME VERY BASIC EXAMPLES AND A REVIEW OF THEIR STRUCTURE

SCATTER PLOTS

Going back to our first example, here's the code again:

```
proc template;
define statgraph sashelp.class;
begingraph;
       layout overlay;
              scatterplot x=height y=weight;
       endlayout;
endgraph;
end;
run;

proc sgrender data=sashelp.class template=sashelp.class;
run;
```

and the corresponding output:

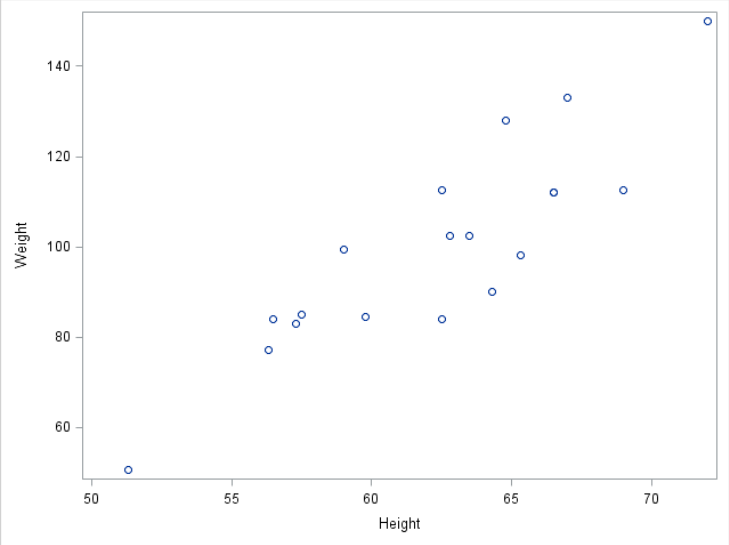

Figure 2. Output of Scatter Plot using PROC TEMPLATE

When calling PROC TEMPLATE, the `define statgraph` tells SAS® that you're going to be generating a statistical graph; the `sashelp.class` is what you want the template to be called. As I mentioned, I would recommend a short but descriptive name for your templates. `BeginGraph` is letting SAS® know that this is where the details of the template are going to be provided; if you were going to be using the template for a Table, for example, you would use `define style` and then obviously omit the `Begingraph`. However, the focus of this paper is on graphics, so we'll continue with our scatter plot. `Layout overlay` is where you can specify options that control the ticks, tick labels, axes, axis type, axis labels, grids, etc. – basically anything that's going to *overlay* your data on the graph. The `endlayout;` statement closes that section. Between that however is the critical piece of the whole PROC – `scatterplot x=height y=weight;`. This is identical to PROC SGPLOT, and makes sense as SGRENDER and GTL are from the same family as SGPLOT (the Statistical Graphics group of PROCs).
The last piece of the code is the PROC SGRENDER statement, and by far the easiest so far. `PROC SGRENDER data=sashelp.class template=sashelp.class; run;.` That's it – putting that together with the TEMPLATE, we get the scatter plot above.
Now, we'll move onto a couple of other graph types.

SERIES PLOTS

Another simple graph done in Base SAS®, Series Plots are very useful for looking at time-based data. In this example, I'm using the SASHELP.CITIDAY data, and I'm calling the Template "Graph" for simplicity's sake.

```
proc template;
define statgraph Graph;
dynamic _DATE _DFXWCAN;
begingraph / designwidth=1024 designheight=776;
        layout overlay;
         seriesplot x=_DATE y=_DFXWCAN / name='series';
     endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CITIDAY template=Graph;
dynamic _DATE="DATE" _DFXWCAN="DFXWCAN";
run;
```

Because we are doing a Series plot, we have a little more work to do. In this case, the first big difference is the `dynamic` statement. This is allowing the two variables indicated in this statement to be set by the

procedure and used to customize the template when you run it. The underscore allows SAS® to know that when you call it in SGRENDER that it was set up in PROC TEMPLATE as a dynamic variable. `Designwidth` and designheight are used to specify the dimensions of the graph; I really like this because often times I need a very specific size of graph, and this allows me complete control. Again we have the `layout overlay;` allowing us to change axes, etc. `seriesplot x=_Date y=_DFXWCAN` calls into the template the two variables that we specified above. Although it seems redundant to have the same variables indicated in two spots, this makes more sense when you start into more complex graph types and want to be able to specify certain variables for very particular portions of a graph. This would be seen for example working in a panel-type graph where you may be calling six or eight variables in the dynamic statement, but then using two per graph for each section. Using the `/ name='series'` provides a name for the series plot itself; again, if you're going to be generating multiple graphs, having short descriptive names are the best. `Endlayout` and `Endgraph` close out their respective statements, and that's it for PROC TEMPLATE.

Looking at the SGRENDER statement, it has an extra line compared to the Scatter plot, but that's because we need to specify the dynamic variables we want included in our output (again, more useful when dealing with multiple variables over a number of graphs). When we run the whole thing, we get:

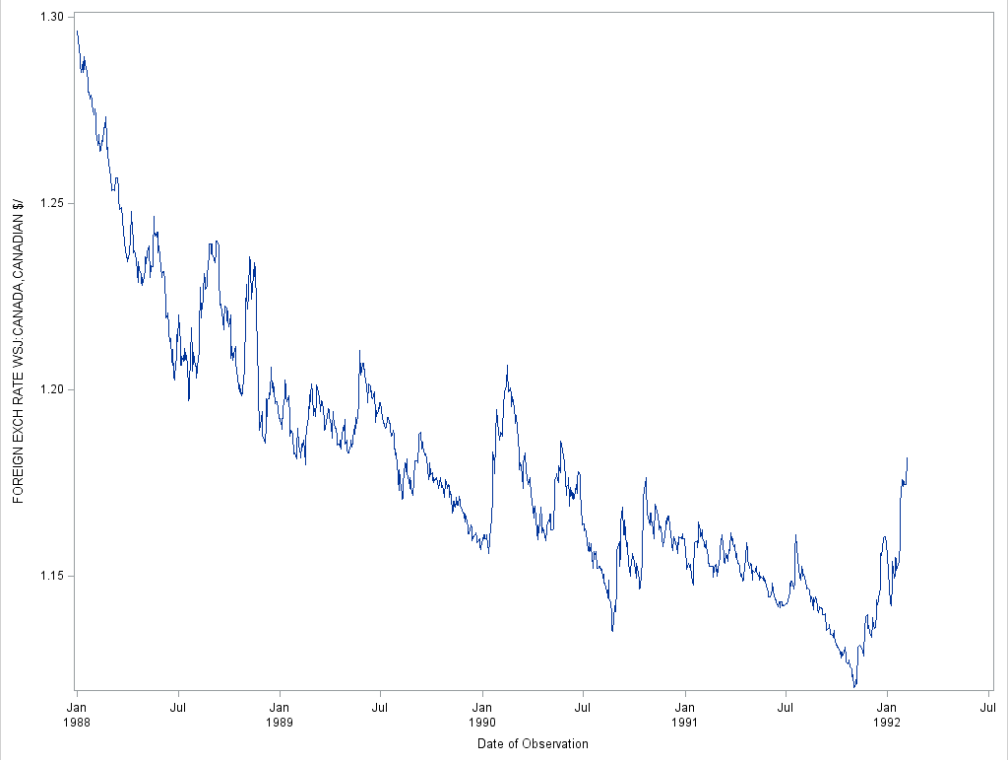

Figure 3. A Series Plot done in PROC TEMPLATE

Considering I'm Canadian, I'm not sure I like this graph! But it was easy enough to create, and is a perfect example of Series Plots.

BAR CHARTS

Another ubiquitous graph type is the Bar Chart, and it is very easy to create in Base SAS®, and comparably easy in PROC TEMPLATE. Here's the code:

```
proc template;
define statgraph Graph;
dynamic _MAKE _LENGTH;
begingraph / designwidth=869 designheight=656;
      layout overlay;
         barchart x=_MAKE y=_LENGTH / name='bar';
      endlayout;
endgraph;
```

```
end;
run;

proc sgrender data=SASHELP.CARS template=Graph;
dynamic _MAKE="MAKE" _LENGTH="LENGTH";
run;
```

Everything is identical to the previous chart types, with the obvious exception of the `barchart` line. **This is one graph that would be better displayed if the data was sorted – the purpose of this paper is to highlight how to create the graphs; it is up to the user to make the decisions on how to display the data.** Using the SASHELP.CARS data set, here is the output generated:

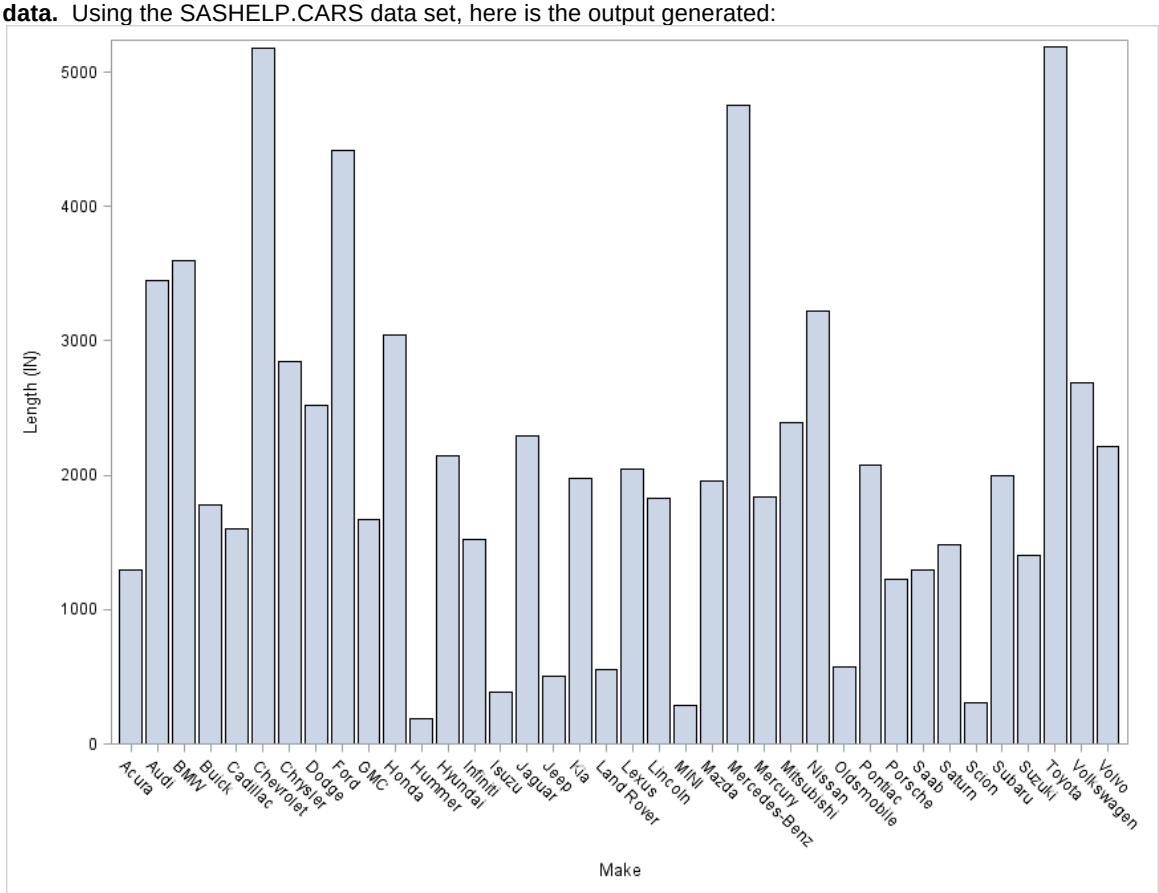

Figure 4. Bar Chart done with PROC TEMPLATE

Not nearly as intimidating as PROC TEMPLATE was made out to be, is it?

HISTOGRAMS

The last basic chart type I want to review are Histograms, because they are used frequently and providing a tool to make these graphs more powerful is definitely something I'd like you to have available.
So without further ado, here's the code:

```
proc template;
define statgraph Graph;
dynamic _AGEATSTART;
begingraph;
   layout overlay;
      histogram _AGEATSTART / name='histogram' binaxis=false;
   endlayout;
endgraph;
end;
run;
```

```
proc sgrender data=SASHELP.HEART template=Graph;
dynamic _AGEATSTART="AGEATSTART";
run;
```

You'll notice a couple of differences here; there's only one dynamic variable (AgeAtStart), and there's a new parameter `binaxis.` When I'm starting something new and I see an option set to True or False, I switch it to see the effect; I find that the visual change is the most effective. So, here is the output of the above code using the SASHELP.HEART data, with binaxis=false:

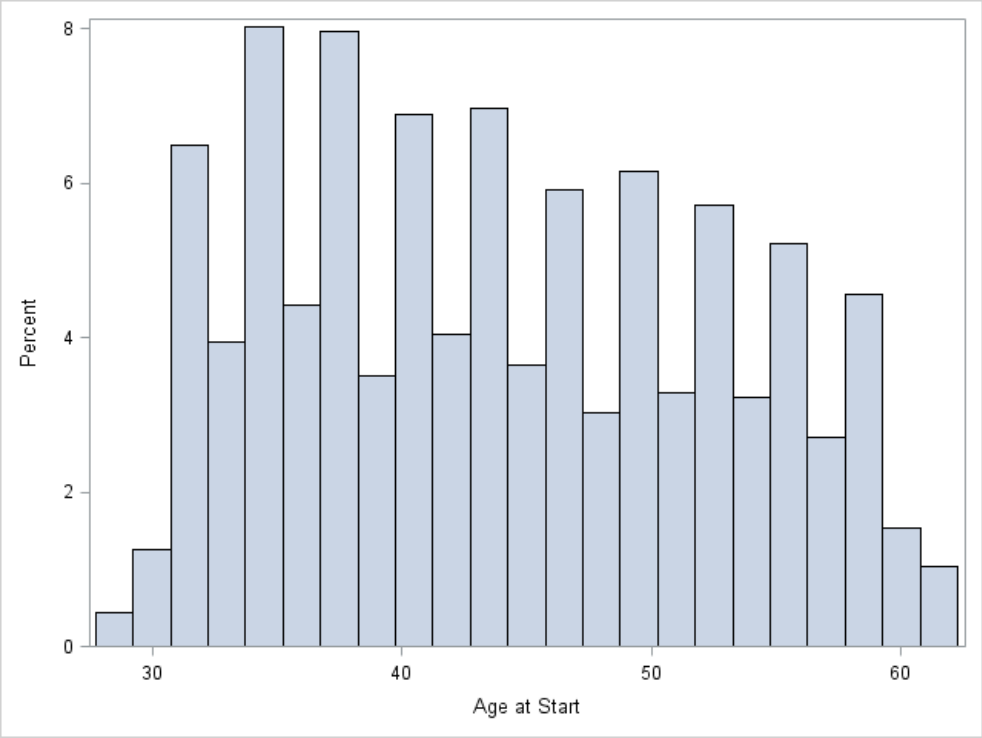

Figure 5a. Histogram without Binning in PROC TEMPLATE

And here's the same graph but with binaxis=True:

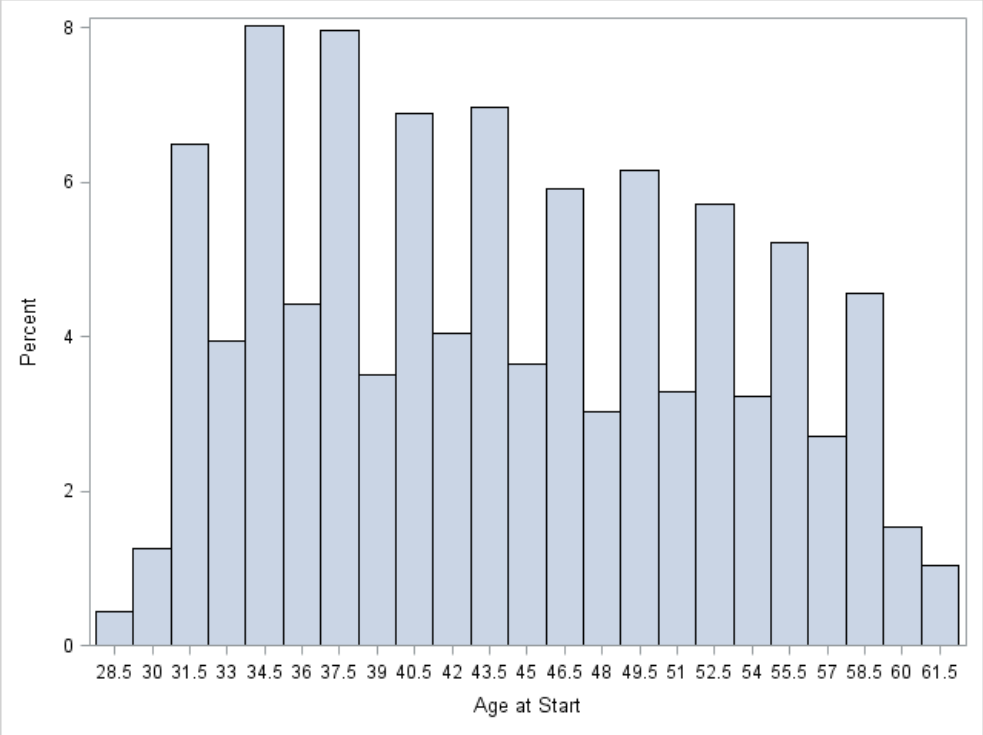

Figure 5b. Histogram with Binning done in PROC TEMPLATE

Identical graphs except the X-Axis shifted from whole numbers to "bins" with 0.5 increments every other tick. **It is understood that this is not the proper use of Binning; because the Age is being treated as continuous, the graph does not make sense. Again, the binning option is the key point here, not the actual output**. Depending on what your needs are, either one would be easy enough to create, but knowing about that parameter could make life a lot easier!

Now that we've got the basics down, let's explore a little further and see what else we can discover.

ENHANCING GRAPHICS

Given the wide variety and complexity of enhancements you can use in PROC TEMPLATE, I wanted to be able to focus in on some of the most useful ones; I reviewed a number of graphs that I've created both in PROC TEMPLATE, Base SAS® and elsewhere, and have come up with a list of common options I modify on a regular basis. It should be noted that these features and add-ons may be different depending on the type of graph you're using; I have not done a full-scale review of all the graphs available, so please proceed with caution!

GROUP BY

Because of the nature of my job, a lot of analysis and reporting I do has to be done overall but then broken down into the five different groups I work with. Therefore, having a quick and easy way to distinguish Group A from another one is key, and with PROC TEMPLATE, it's as easy as one extra option. Using the SASHELP.CLASS dataset once again,

```
proc template;
define statgraph Graph;
dynamic _HEIGHT _WEIGHT _SEX;
begingraph / designwidth=861 designheight=629;
      layout overlay;
         scatterplot x=_HEIGHT y=_WEIGHT / group=_SEX name='scatter';
      endlayout;
endgraph;
end;
run;
```

```
proc sgrender data=SASHELP.CLASS template=Graph;
dynamic _HEIGHT="HEIGHT" _WEIGHT="WEIGHT" _SEX="SEX";
run;
```

I've highlighted the differences – definitely not as scary as you thought, is it? Here's the output:

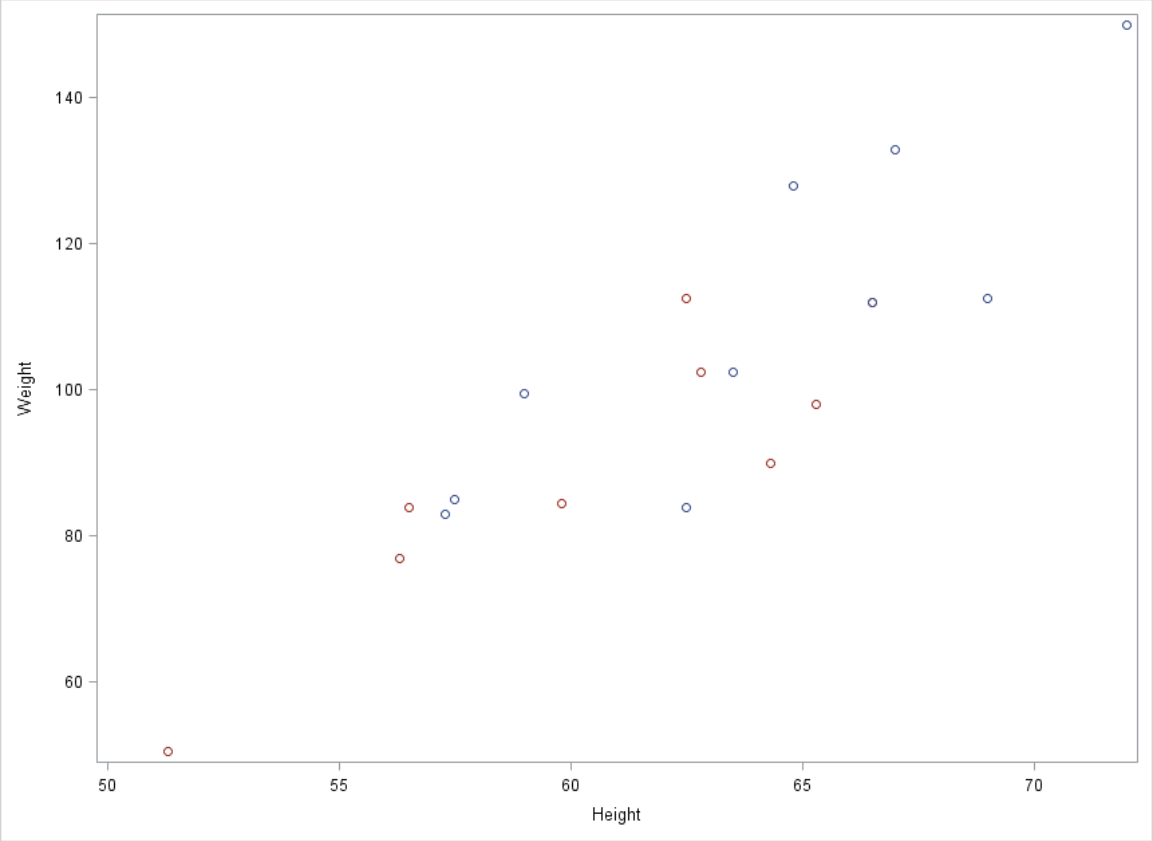

Figure 6. A Scatter Plot with GROUP BY

OK, so that's great – but what do the Blue circles and the Red circles mean? That brings us to Legends.

LEGENDS

Having something on the Graph to tell you "RED = Female" is key to good data visualization. Therefore, I was very happy to learn that with a few keystrokes, I could add a legend to my scatter plot – and again, with the power of PROC TEMPLATE, there are a great number of features and options you can use. I've kept it to the basics here however:

```
proc template;
define statgraph Graph;
dynamic _HEIGHT _WEIGHT _SEX;
begingraph / designwidth=861 designheight=629;
      layout overlay;
         scatterplot x=_HEIGHT y=_WEIGHT / group=_SEX name='scatter';
         discretelegend 'scatter' / halign=left valign=top across=1
location=inside;
      endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=Graph;
dynamic _HEIGHT="HEIGHT" _WEIGHT="WEIGHT" _SEX="SEX";
```

```
run;
```

I apologise if the code is getting to be repetitive but as mentioned I like to build slowly and I find that having the code shown every step of the way allows for maximum understanding. The `discretelegend` option is used, and we want it to refer to the "scatter" that was declared in the line above (this is where naming becomes key, especially in multi-graph outputs). `HALIGN` and `VALIGN` are horizontal and vertical alignment (because I know I have an outlier in the top right corner, I've put the legend in the top left). `Across` allows you to have the legend wide or long, depending on preferences, space, etc.

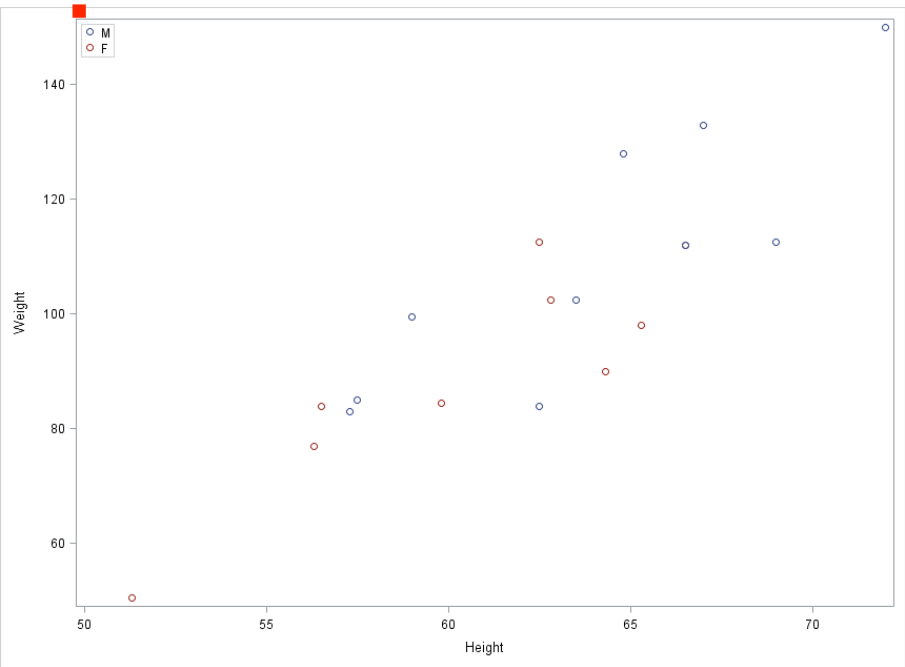

Figure 7. The scatter plot, now with a Legend

We are getting closer to what I would consider to be a complete graph. But what is this graph actually representing? I'd like some more details to be available, so that others will know what I'm reporting.

TITLES

Titles need to be short, descriptive, and provide the reader enough information to know precisely what is being reported. A good philosophy to have when it comes to data visualization is that a graphic should be so clear and explicit that it does not require text to explain it; a title plays a critical role in that.

```
proc template;
define statgraph Graph;
dynamic _HEIGHT _WEIGHT _SEX;
begingraph / designwidth=861 designheight=629;
   layout overlay;
         scatterplot x=_HEIGHT y=_WEIGHT / group=_SEX name='scatter';
         discretelegend 'scatter' / opaque=false border=true halign=left
valign=top displayclipped=true across=1 order=rowmajor location=inside;
         entry halign=center 'This is a scatter plot of the SAS CLASS
Data set comparing Height to Weight and Grouped by Gender' / valign=top
location=outside textattrs=(size=12 );
   endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=Graph;
dynamic _HEIGHT="HEIGHT" _WEIGHT="WEIGHT" _SEX="SEX";
```

```
run;
```

Holy cow, that's a lot of new code! But wait – it's not really. `Entry` is just a text entry put on to the graph; `HALIGN` is obviously the same as above, and then the text that I want in the title. `VALIGN` and `LOCATION` are self-explanatory. `Textattrs` are the options you want to modify for the text; in this case, I wanted to change the size of the font to 12.

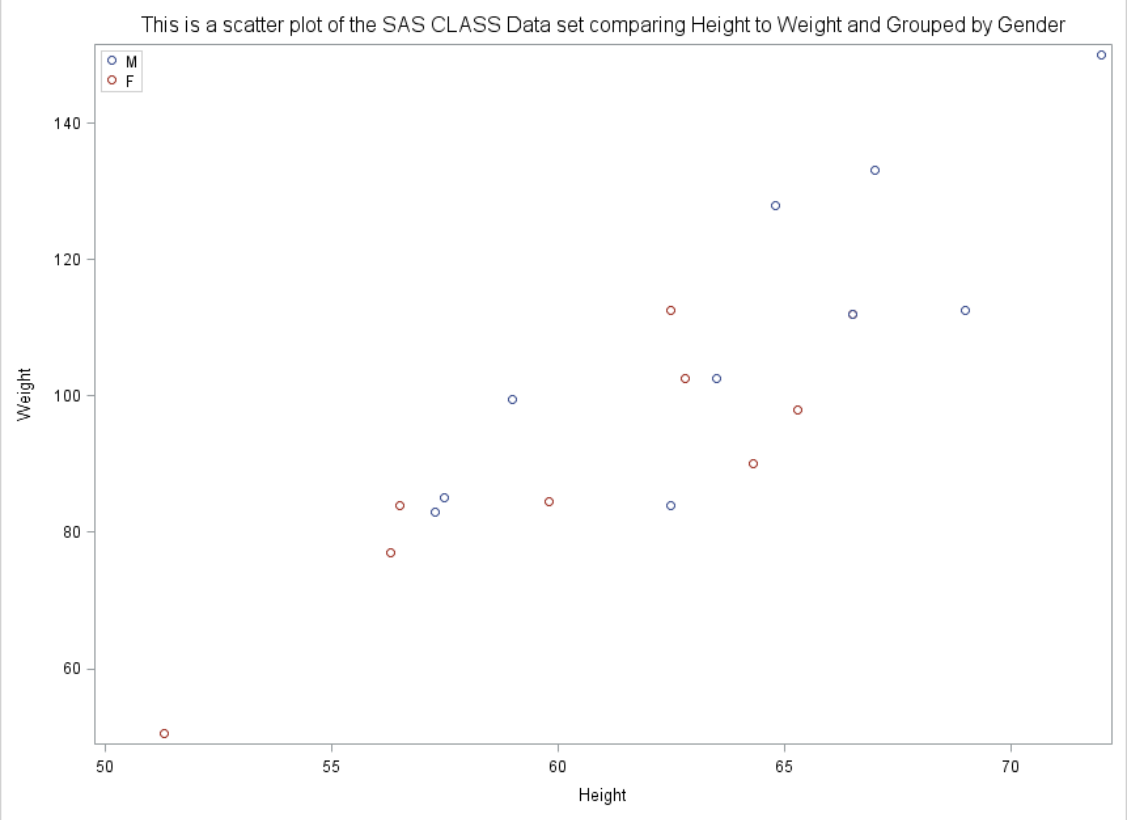

Figure 8. Having added a title through PROC TEMPLATE

We're getting a pretty good looking graph, and yet all we've done is added a few simple lines of code to our very basic PROC TEMPLATE – and we're not done yet!

DATA LABELS

The last piece of the PROC TEMPLATE puzzle that I wanted to cover is Data Labels. These are not something you should be putting on every graph; but when used properly they are an effective tool to allow your reader an enhanced understanding of the data.

```
proc template;
define statgraph Graph;
dynamic _HEIGHT _WEIGHT _SEX _NAME;
begingraph / designwidth=861 designheight=629;
   layout lattice / rowdatarange=data columndatarange=data rowgutter=10
columngutter=10;
      layout overlay;
         scatterplot x=_HEIGHT y=_WEIGHT / group=_SEX datalabel=_NAME
name='scatter';
         discretelegend 'scatter' / opaque=false border=true halign=left
valign=top displayclipped=true across=1 order=rowmajor location=inside;
         entry halign=center 'This is a scatter plot of the SAS CLASS
Data set comparing Height to Weight and Grouped by Gender' / valign=top
location=outside textattrs=(size=12 );
```

```
      endlayout;
   endlayout;
endgraph;
end;
run;

proc sgrender data=SASHELP.CLASS template=Graph;
dynamic _HEIGHT="HEIGHT" _WEIGHT="WEIGHT" _SEX="SEX" _NAME="NAME";
run;
```

To add data labels, it is 5 words and two equals signs? Pretty easy, isn't it?

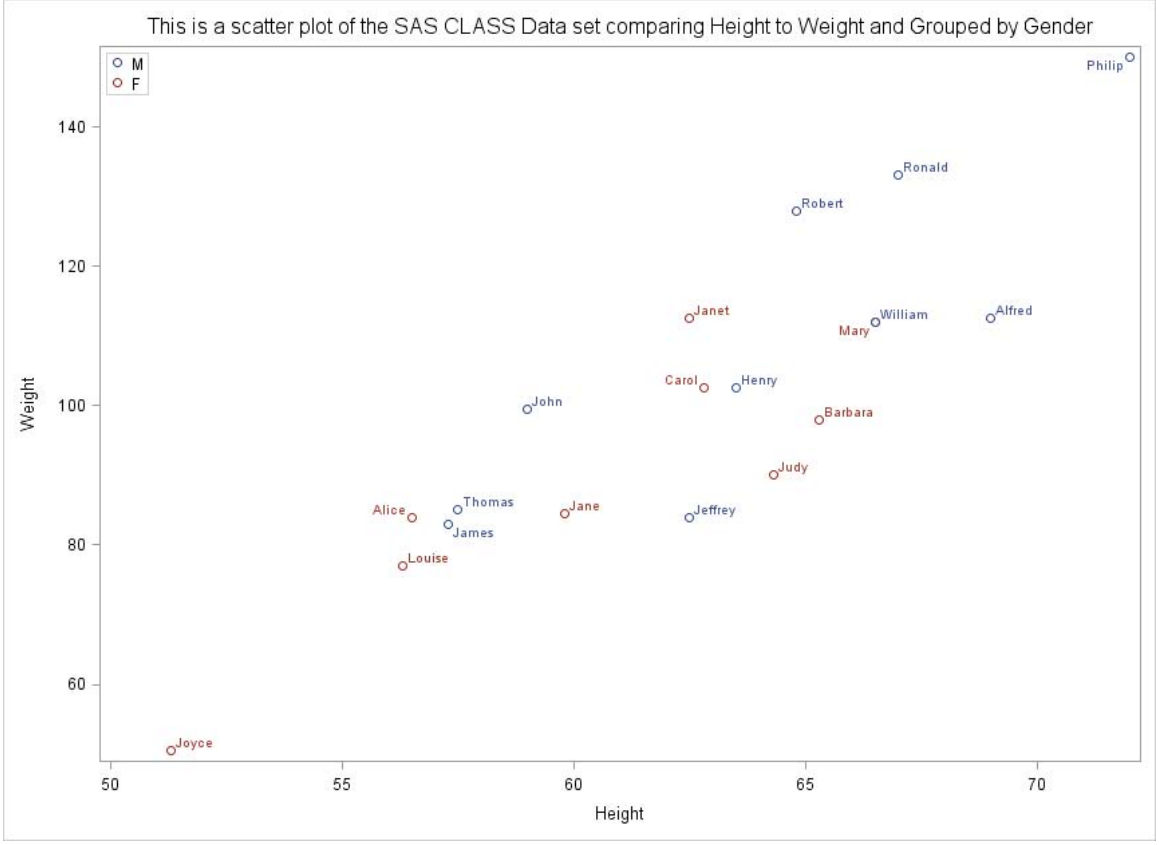

Figure 9. Addition of Data Labels

Does this graph look at all similar? Look back to Figure 1 – although there were a few more tweaks and modifications in that graph, we are very, very close – and it was not all that painful, was it?

CONCLUSIONS

PROC TEMPLATE and the Graph Template Language are extremely versatile, powerful and flexible tools available to any SAS® User. Like anything in SAS®, the complexity lies in knowing that there is the unknown; once you start working with it, changing / adding / removing options, working through examples using the SASHELP data sets, you'll find that you are up and running in much less time than you thought possible. I find the ability to make graphs that people look at and say "Oh wow, you did that in SAS®?" is a very invigorating feeling and having these two new features at my finger tips has even made experienced SAS® users not believing this was feasible.

REFERENCES

Kuhfeld, W. 2010. Statistical Graphics in SAS: An Introduction to the Graph Template Language and the Statistical Graphics Procedures. Cary, NC: SAS Institute.
Kuhfeld, W. 2010. The Graph Template Language and the Statistical Graphics Procedures: An Example-Driven Introduction. Cary, NC: SAS Institute. Available at: http://support.sas.com/resources/papers/proceedings10/334-2010.pdf
SAS Institute. 2012. SAS® 9.3 Graph Template Language Reference, Third Edition. Cary, NC: SAS Institute. Available at: http://support.sas.com/documentation/cdl/en/grstatgraph/65377/PDF/default/grstatgraph.pdf
Cartier, J. SUGI31. A Programmer's Introduction to the Graphics Template Language. SUGI31 Proceedings. Available at: http://www2.sas.com/proceedings/sugi31/262-31.pdf
Mantage, S. 2009. Introduction to the Graph Template Language. WUSS09 Proceedings. Cary, NC: SAS Institute. Available at: http://www.wuss.org/proceedings09/09WUSSProceedings/papers/dpr/DPR-Matange1.pdf

ACKNOWLEDGEMENTS

Thanks to Murphy Choy, section chair for Reporting and Information Visualisation, for allowing me the opportunity to present such an intimidating topic. Also, I am heavily indebted to Warren Kuhfeld, whom I was fortunate enough to see at my very first Global Forum, and who sparked my interest in this relatively new field in SAS®.

CONTACT INFORMATION

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

Name: Christopher Battiston
Institution: Hospital for Sick Children
Address: 555 University Avenue
City, State ZIP: Toronto, Ontario, Canada M5G 1X8
E-mail: darth.pathos@gmail.com

Appendix A

PROC TEMPLATE Code for Figure 1

```
proc template;
define statgraph sgdesign;
dynamic _HEIGHT _WEIGHT _AGE _NAME;
begingraph / designwidth=976 designheight=751;
   entrytitle halign=center 'Scatter plot of Height versus Weight for the
SASHelp.Class Dataset';
   entryfootnote halign=left 'Prepared by C. Battiston';
     layout overlay / xaxisopts=( labelattrs=(color=CXA52829 family='SAS
Monospace Bold' size=14 style=NORMAL weight=BOLD )) yaxisopts=(
labelattrs=(color=CXA52829 family='SAS Monospace' size=14 style=NORMAL
weight=BOLD ));
         scatterplot x=_HEIGHT y=_WEIGHT / group=_AGE datalabel=_NAME
name='scatter' datalabelposition=TOP markerattrs=(size=15 weight=bold );
         discretelegend 'scatter' / opaque=false border=true halign=left
valign=top title='Age' displayclipped=true across=1 order=rowmajor
location=inside;
      endlayout;
   endgraph;
end;
run;
proc sgrender data=SASHELP.CLASS template=sgdesign;
dynamic _HEIGHT="HEIGHT" _WEIGHT="WEIGHT" _AGE="AGE" _NAME="NAME";
```

```
run;
```