

From Good to GREAT: The (Not-So-Technical) Qualities of Great Programmers

Tamar Roomian, MS MPH, Stryker Neurovascular

ABSTRACT

This paper aims to present qualities of programmers that employers and your fellow employees value that are beyond technical SAS programming skills, and specific examples of how these soft skills can translate to SAS programming. This paper will discuss how to think critically and contribute intellectually, work backwards to create the program, understand your subject matter, and build your problem-solving muscles. This presentation is aimed at beginner programmers who are just entering the workforce; however, any programmer could benefit from the tips and discussion presented in this paper. These soft skills will distinguish you as a GREAT programmer – someone who delivers better results and is a pleasure to work with.

INTRODUCTION

This paper aims to present qualities of programmers that employers and your fellow employees value that are beyond technical SAS programming skills, and specific examples of how these soft skills can translate to SAS programming. This presentation is aimed at beginner programmers who are just entering the workforce; however, any programmer could benefit from the tips and discussion presented in this paper. These soft skills will distinguish you as a GREAT programmer – someone who delivers better results and is a pleasure to work with.

THINK CRITICALLY AND CONTRIBUTE INTELLECTUALLY

THINK CRITICALLY

As programmers, we are frequently in situations where there is a business ask, and we are expected to simply deliver the asked results. Often, the person with the request does not realize what they actually need. They may not be aware that their request will not answer their question. The programmer who has more intimate knowledge of the data should think critically when collaborating and fulfilling requests. Great programmers will start with probing questions:

1. What are you trying to know?
2. What is the goal here?
3. Why are you doing this?

This gives the programmer the opportunity to evaluate if what the requester wants will actually achieve the end goal. If not, the programmer can give recommendations.

CONTRIBUTE INTELLECTUALLY

Based on the responses, the programmer should feel free to contribute intellectually and ask clarifying questions. The person with the request may not be aware of the assumptions that need to be made. An example could be explaining the different sources of data that could be used, and the advantages and limitations of each, and how they may differ from each other.

Here is an example conversation within the context of a healthcare research study:

Requestor: I want to know the percentage of 12-month visits completed for each site.

Programmer: Who do you want to include in your denominator?

Requestor: What do you mean?

Programmer: Do you want to include visits where today's date is after the minimum visit window? This would include visits that are within window, but not yet overdue. Or do you want to limit to visits where today's date is after the maximum visit window?

Requestor: Let's limit to visits with today's date after the maximum visit window.

Programmer: Okay. One more question – if subjects discontinued the study prior to completing the visit, do you want that to count against the site? Or would that count towards completion?

Requestor: That's a great question. We do not want to penalize the site if a subject left prior to completing the visit.

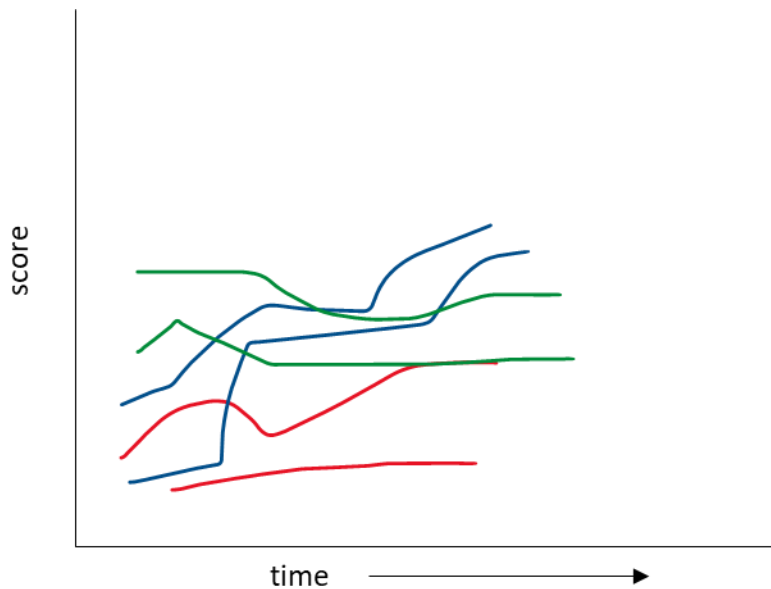
The programmer could have made assumptions about what the requester wanted. However, contributing intellectually and asking clarifying questions will ultimately produce better results for the requester in a more efficient way.

UNDERSTAND THE GOAL AND WORK BACKWARDS

Having a "beginning to end" mentality can make it hard to get started. If having trouble getting started, working backwards from the end makes the steps clearer. Ask yourself "What is the end product?" "What is the ultimate goal?" Then, ask yourself "What is the preceding step?" "How would the data need to be structured?" Then continue to work backwards. "How is the data structured now?" Then, the path forward becomes clear.

An example:

You want to make a graph that tracks scores of subjects over time. Each subject will have a line. The subjects were assigned different treatment groups, so each group will have a different color. You visualize the end product first with a rough sketch:



In order to achieve this, you're envisioning a program that looks like this:

```
proc sgplot data=dataset;
1series x=time y=person1/markers markerattrs=(symbol=circle color=blue)
lineattrs=(color=blue);
run;
```

¹Need to find a way to automate the creation of a series statement for each person

Each person will be a series in proc sgplot. You are not going to manually write out every subject as a series, so you need to structure the data in a way that lets you write a program to fill in that section. To do that, the data should be structured like this where each subject is a column:

Time	Person1	Person2	Person3	Person4
0	8	3	1	6
19			2	
24				
30		5		
37				
52	10			
59		4		
60			3	
65				8
95				9
100	11			
104				
150			4	

And your current data structure is like this:

Time	Person_id	Treatment	outcome
0	1	Trtmt1	8
24	1	Trtmt1	.
52	1	Trtmt1	10
100	1	Trtmt1	11
0	2	Cntrl1	3
30	2	Cntrl1	5
59	2	Cntrl1	4
104	2	Cntrl1	.
0	3	Trtmt2	1
19	3	Trtmt2	2
60	3	Trtmt2	3
150	3	Trtmt2	4
0	4	Trtmt1	6
37	4	Trtmt1	.
65	4	Trtmt1	8
95	4	Trtmt1	9

You need to transpose the data so that each subject is a column.

You also need to find a way to assign a different color for each treatment in proc sgplot:

```
proc sgplot data=dataset;  
  1series x=time y=person1/markers markerattrs=(symbol=circle  
  color=blue2) lineattrs=(color=blue2);  
run;
```

1. need to automate the creation of a series statement for each person
2. need to assign the color for each subject

You use the SQL procedure to create the series of text that will go into proc sgplot, but then you realize you don't have a dataset that assigns a color based on treatment:

```
proc sql;
select
cat("series x=time y=", name, "/markers markerattrs=(symbol=circle
color=", color, ") lineattrs=(color=", color, ");")

into :series separated by " "

from ?
;
quit;
```

You need to make a dataset with the color assignment that will be used above which can be easily created in a DATA step or SQL Procedure:

name	treatment	color
person1	Trtmt1	blue
person2	Cntrl1	red
person3	Trtmt2	green
person4	Trtmt1	blue

Then, the macro variable series can be called within proc sgplot:

```
proc sgplot data=outcome_wide;
/*macro variable goes below, no semi-colon needed at the end*/
&series
run;
```

Final tweaks (adding axis labels) added:

```
proc sgplot data=outcome_wide;
/*macro variable goes below, no semi-colon needed at the end*/
&series
yaxis label="Outcome";
xaxis label="Time (weeks)";
run;
```

Working backwards helped create the program.

UNDERSTAND YOUR SUBJECT MATTER

Having a basic understanding of the subject matter will help the programmer:

1. Know what questions to ask
2. Define variables correctly
3. Catch unreasonable results, programming mistakes, or data entry mistakes

An example conversation:

Requestor: I'm interested in looking at factors associated with dysgeusia (loss of taste) among patients with head, neck, and throat cancer. Here's a list of variables I'm interested in (hands over list).

Programmer: Thanks for the list. I noticed smoking status is not on here. Was that an oversight? I would think smoking could impact taste, or at least be a confounder with the other things you have here.

Requestor: We definitely need to include smoking status.

BUILD YOUR PROBLEM-SOLVING MUSCLES

TROUBLE SHOOTING IS A SKILL

When programmers approach colleagues with SAS programming problems, the first questions to address the problem are often:

1. What did the log say?
2. Did you run the program step by step?
 - a. At what point in the program did you encounter the problem?
3. What have you tried already?
 - a. What were the results?

Going through these basic questions on your own first can often identify the exact problem and illuminate the solution. Moreover, investigating these steps before consulting colleagues will show that you made an honest effort and save your colleagues' time.

DID YOU GOOGLE IT?

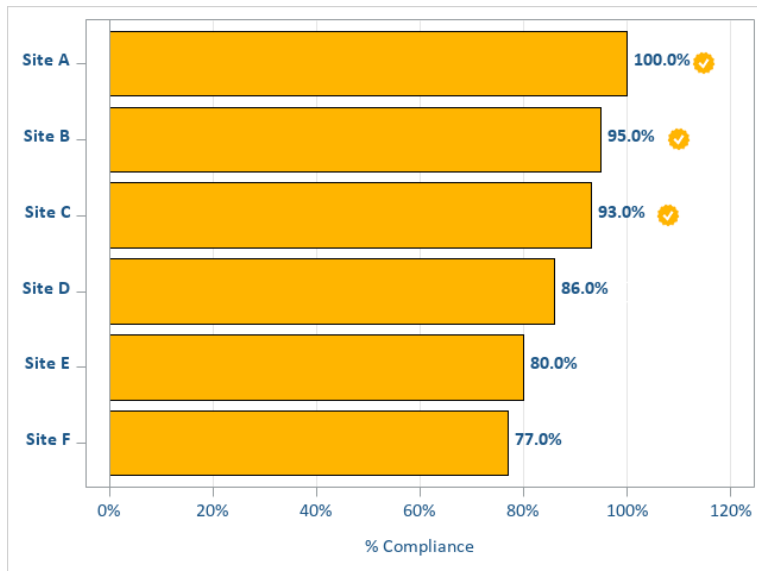
It sounds like a trite question. Believe it or not, using a search engine or finding the answer online is a skill that can be developed. Learning how to search and identifying the right search terms can make a difference in finding a solution to a problem.

As an example, searching for "color coding SAS report" will yield different results than "traffic lighting SAS report" although both terms describe similar concepts.

Besides the Google search engine, search SAS Blogs (<https://blogs.sas.com/>) or LexJansen (<https://www.lexjansen.com/>) to find what you need.

Sometimes, it's hard to find a single tutorial that will produce exactly what you want.

As an example, you want to create a graph like this:



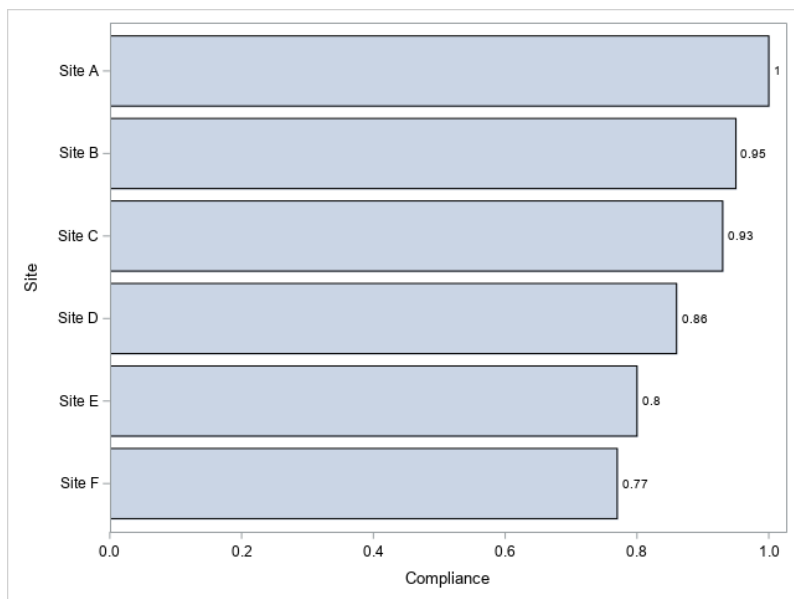
The graph would need to be produced regularly, so automating in SAS would be ideal.

1. You search a basic term ("bar chart") in sas Blogs search bar (<https://blogs.sas.com/>), then browse to find examples close to what you want.
2. You realize that what you have is a horizontal bar chart. Clicking through a few examples, you realize you could try HBAR or HBARPARM statement. Using Google, you searched "hbar vs hbarparm sas." The SAS documentation for HBARPARM is in the search results:

"Parameterized bar charts require a response variable in addition to the category variable. The response variable contains pre-summarized computed values such as a sum or a mean for each unique value of the category variable. This is the key distinction between HBARPARM and HBAR." (The SAS Institute 2024d)

3. Given that your data is already summarized by site, hbarparm seemed like the way to go. You start with a basic SGLOT Procedure using hbarparm:

```
proc sgplot data=compliance ;  
hbarparm category=site response=compliance;  
run;
```



Right away, you notice that the colors and font you want are different than the SAS defaults. You will need to find a way to change the font and colors for the axes and the data labels. You will also need to find a way to add the checkbox at the end of the bar to only sites that met the threshold of 90%.

Despite not finding a single tutorial for exactly what you wanted to create, you can piece together components from searching the SAS documentation and SAS blogs to get the result you want:

```
/*add the additional variables needed*/
proc sql;
create table compliance2 as
select
site
, compliance format=percent8.1
/*the position of the checkbox*/
, compliance+.15 as checkbox_value

/*conditional logic of the checkbox*/
, case when compliance>=.90 then "compliance"
      else "nocompliance"
      end as checkbox
from compliance
;
quit;

/*data attribute map to assign symbols based on the conditional logic
above*/
data myattrmap;
length value $15;
length markersymbol $15;
ID="myid";
value="compliance";
markersymbol="sym1"; output;
```

```

ID="myid";
value="nocompliance";
markersymbol="circle";
markercolor="white";
output;

run;

proc sgplot data=compliance2 dattrmap=myattrmap noautolegend;

3hbarparm category=site response=compliance/datalabel
datalabelattrs=(size=12pt color=CX1C5687 weight=bold family=Calibri )
DATALABELFITPOLICY=NONE fillattrs=(color=CXFFB500);

1symbolimage name=sym1 image="C:\image.jpg "; /*path to image goes
here*/

axis valueattrs=(size=12pt color=CX1C5687 family=Calibri) label="%
Compliance" labelattrs=(size=12pt color=CX1C5687 family=Calibri) grid;
yaxis valueattrs=(size=12pt color=CX1C5687 weight=bold family=Calibri)
labelattrs=(color=CX1C5687 size=12pt family=Calibri) display=(nolabel)
;

scatter x=checkbox_value y=site /attrid=myid markerattrs=(size=15pt)
group=checkbox;

run;

```

¹You want to add an image (a check box) for only bars that meet a threshold. You searched hbarparm in sas Blogs search bar. The following looked promising:

Use an image in a graph in SAS (Mantange 2017)

²Assign the symbol based on a condition (assigning a checkmark to sites having compliance over a threshold >90%):

SAS Help Center: Discrete Attribute Map Data Sets (The SAS Institute 2024c)

³Finally, you need to change colors and fonts for every aspect of the graph:

- a. You can use the SAS documentation for the SGPLOT Procedure, HBARPARM to find the statements for each graph attribute (The SAS Institute 2024d).
- b. You can use the SAS documentation to assign the exact colors you want. (The SAS Institute 2024b).

By searching the SAS documentation and SAS blogs, it was possible to piece together the components needed to get the result you want.

CONCLUSION

Anyone can have advanced programming skills. Thinking critically and contributing intellectually, working backwards, understanding your subject matter, and building your problem-solving muscles are soft skills that will distinguish you as a GREAT programmer.

REFERENCES

- Mantange, Sanjay. 2017. "A graph with visual categories." Accessed April 7, 2024. <https://blogs.sas.com/content/graphicallyspeaking/2017/04/12/graph-visual-categories/>
- The SAS Institute. 2024a. "SAS Blogs." Accessed April 7, 2024. <https://blogs.sas.com/>
- The SAS Institute. 2024b. "SAS Help Center: Color-Naming Schemes documentation.sas.com." Accessed April 7, 2024. https://documentation.sas.com/doc/en/pgmsascdc/v_049/grstatgraph/p0edl20cvxxmm9n1i9ht3n21eict.htm
- The SAS Institute. 2024c. "SAS Help Center: Discrete Attribute Map Data Sets." Accessed April 7, 2024. https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/n18szqcwlr8q2nn10od9hhdh2ksj.htm
- The SAS Institute. 2024d. "SAS Help Center: Syntax: PROC SGPLOT HBARPARM documentation.sas.com." Accessed April 7, 2024. https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grstatproc/n0a5svhjs7b505n1vk4kuth2x3q.htm

RECOMMENDED READING

- Warrillow, Emma. 2017. "From Coder to Collaborator: Tips and Tricks for Being a Better Marketing Analyst" *SAS® Global Forum 2017*. Available at <https://support.sas.com/resources/papers/proceedings17/0850-2017.pdf>

CONTACT INFORMATION

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

Tamar Roomian
tamar.roomian@stryker.com