Paper 1724-2018

Bricolage: My autobiography with SAS® procedures

AnnMaria De Mars, 7 Generation Games

ABSTRACT

When it comes to starting a fascinating career, SAS is as good a starting place as any, and probably better than most. All of my various careers have had their roots in statistics and SAS. Could I have learned the same concepts and gotten the same results using another programming language, taking a different path? Probably. But I didn't. I did it with SAS. In this paper, I'll explain the many ways in which I have built a career by bricolage – that is, from building stuff – programs, companies – using whatever was lying about. With its application in a wide range of fields, from psychology to biostatistics, SAS has a largely unrecognized potential to address the shortage and lack of diversity of software developers by introduction of fundamental building blocks to success outside of the standard computer science curriculum. In fact, the more you look at it, the clearer it becomes that you can generalize from SAS to a career using almost any language and almost anywhere, even Santiago, Chile.

INTRODUCTION

TL, DR; My point, which most people despair of me ever having, is that when it comes to starting a fascinating career, SAS is as good a starting place as any, and probably better than most. I thought the title of Al Franken's book, The Truth, with Jokes, was great and I wanted to do something just like it. Unfortunately, I'm not funny. So, since all of my various careers have had their roots in statistics and SAS, I decided to write my autobiography of my career with SAS procedures.

SPECIALIST OR GENERALIST? BOTH ARE AN OPTION WITH SAS

When speaking of careers, often, the discussion comes up among colleagues whether it is better for one's career to be a specialist or a generalist. It's a little (a lot) too late for me to become the world's foremost authority on PROC REPORT, besides, Art Carpenter, has that spot taken (Carpenter, 2007). Right after wondering whether anyone uses PROC REPORT any more and deciding not to read Art's book, I starting thinking of all of the basic concepts I learned from PROC REPORT that I apply regularly, mostly with PHP.

Maybe this example that I shamelessly lifted from the SAS documentation (SAS Institute, 2009) :

```
PROC REPORT DATA=grocery nowd;
   COLUMN manager department sales;
   RBREAK after / dol summarize;
   WHERE sector='se';
```

looks different from this PHP code to do reports on students playing an educational game:

```
$stmt = $conn->prepare("SELECT * FROM answers WHERE username =
:username");
// Run the query
$result = $stmt->execute(['username' => $username]);
echo '<h2>Data Requested for User: '. $username. '</h2> <p</p>' ;
$rows = $stmt->rowCount() ;
for ($j = 0; $j< $rows ; ++$j ) {
    $row = $stmt->fetch();
    echo '<tr><td>' . $row['question'] . '</td><td>' . $row['answer']
.'</td><td>' . $row['tries'] .'</td><td>' . $row['correct_answer'] .
</td></tr>' ;
}
```

However, both accomplish pretty much the same thing, that is, select rows based on meeting criteria specified with a WHERE, print out certain columns for every row, get a summary value. One might be a little more of a pain to write than the other, but the general idea is the same. In fact, the more you look at it, the clearer it becomes that you can generalize from SAS to a career using almost any language and almost anywhere, even Santiago, Chile.

If you decide to be a specialist, your career path probably looks something like this:

Figure 1. Specialist Career Path

You get a bachelors and a masters in statistics, you become a data analyst and work your way up to managing the entire division. If you follow this linear path, you'll move up the ranks to knowing absolutely everything about, for example, clinical trials of migraine drugs, and you'll probably end up with a nice house in the suburbs, a 401K and three weeks of vacation each year. At 60 or so, you can retire and spend more time at your lake house. That's a perfectly reasonable choice.

DON'T PAY ANY ATTENTION TO ANYONE CALLING YOU "JUST" A SAS PROGRAMMER

I want to emphasize that specialization is not your ONLY choice. You may hear a lot of discouraging words.

- You can't be a software developer,
- SAS isn't a real programming language,
- you have no experience in creating software applications,
- doing statistical analyses doesn't count

and a lot of other nay-saying. I'd recommend you follow the policy of our 7 Generation Games CEO, Maria Burns Ortiz (2014) who commented that she never complied with the suggestions made by a venture capitalist that she restrict her activities because, "I refuse to follow stupid advice."

Perhaps you are asking yourself,

> What can I do? I have an M.S. in statistics (or business or sociology) and two years of experience using SAS. What options do I have?

In my decidedly non-linear career, I have been an industrial engineer, professor in Schools fo Education, Engineering, Business, Liberal Arts and Human Services. I think the only one I have missed is Fine Art. I was supposed to teach a statistics course for Fine Arts majors this semester, but I ended up going to Santiago, Chile to open up our Latin American office instead. I've been a consultant, programmer, statistician, consultant again and now am president/co-founder of a gaming start-up. My career path is decidedly not linear. It looks more like this

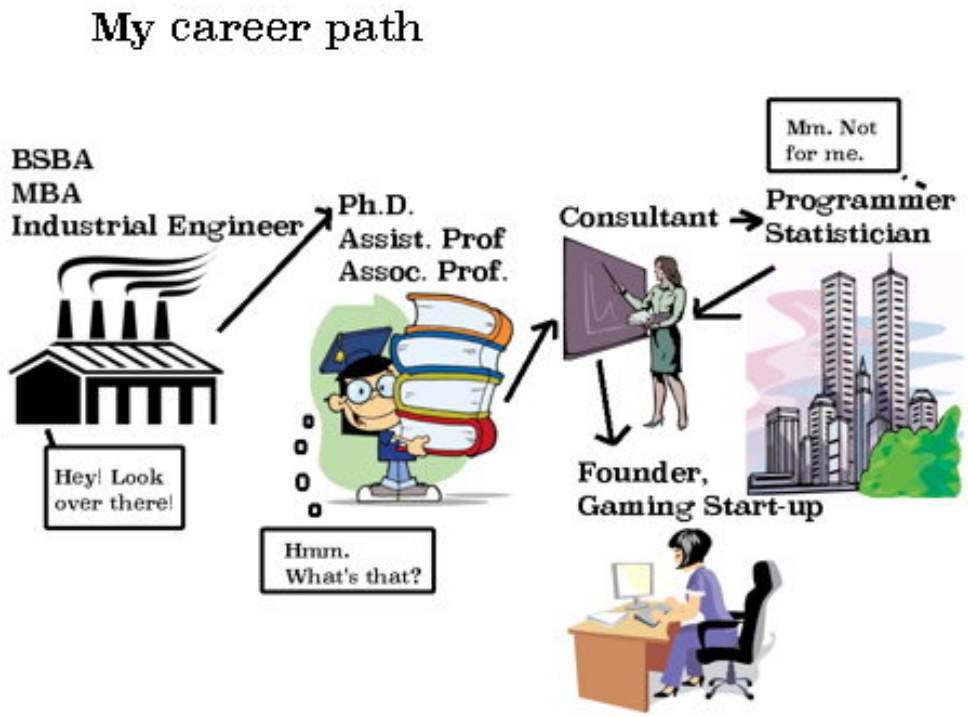

Figure 2. Generalist Career Path

The fact is, once you have the foundation, the choice of direction is up to you.

THE 5 BUILDING BLOCKS OF SUCCESS IN SOFTWARE DEVELOPMENT

If you are in any kind of quantitative field you have a VAST range of options, from working at some of the largest companies in the world in marketing research to performing efficacy studies for non-profits whose staff members can be counted on one hand. All of these broad number of opportunities require, at most, five building blocks:

1. Programming concepts – You need to understand scope, do-loop, arrays, functions
2. Working in a software development team – this is the part "self-taught" programmers are often not taught – documentation, testing and debugging
3. Statistics – coming from the age when we inverted matrices by hand with a piece of paper and a pencil (not kidding) SAS, SPSS, R, Stastica, JMP and even Excel have made this a hundred times easier from when I started in the field
4. Data management – The thousand ways that users can enter data, and how to keep it from screwing up your results
5. Domain specific knowledge – by that, I mean if you are working in aerospace know something about what a transmitter and receiver are, know that a male and a female plug is a thing. If you are in biostatistics, understand survival analysis, relative risk.

Since I started (mostly) with SAS, my examples focus on how starting as a SAS programmer can be like a Dr. Seuss book – "Oh, the places you'll go!" My main point, as I have said before, is that it doesn't matter what language you use in the beginning, although SAS has a few advantages as a "starter kit".

PROGRAMMING CONCEPTS

Let's start with programming concepts. SAS taught me basic concepts I use almost every day, like …

Arrays

One of the most frequent uses I make of SAS is to score tests, which requires creating an array of answers from a respondent and a second array of items scored correct or incorrect. Perhaps making a game where kids earn points by solving math problems which they then spend in a virtual village trading posts to outfit their wigwam is a far cry from scoring standardized tests. However, our game, Making Camp, that teaches multiplication and division, has a virtual trading post and a wigwam, both of which make extensive use of arrays. All of the items you can "buy" with the points you earned from solving math and history problems are in an array.

SAS Arrays

```
Data scored ;
   set mydata.data2013 ;
   array ans{70} q1- q70 ;
   array correct{70} c1 – c70 ;
   array scored{70} sc1 – sc70 ;
```

JavaScript Arrays

```
var things = [
"art/tomahawk.png", "art/dog.jpg", "art/pottery.png",
"art/deer_skin_sm.png",
"art/bass_side.png", "art/arrows_and_quiver.png", "art/turtle.jpg", "",
"art/parfleche.png", "art/feather_sm_side.png",
"art/plate.png"
];
var things_name = [
"TOMAHAWK", "DOG", "POTTERY", "DEER SKIN", "BASS", "ARROWS AND QUIVER",
"TURTLE", "", "PARFLECHE", "FEATHER", "PLATE"
];
```

Yes, they look a little different but the basic concept is the same. In the SAS example, I'm creating three array which I will be matching up later in the program – the answer the students gave, the correct answer and the item scored correct or incorrect. In the JavaScript example, I am matching up two arrays; the source for the image file and the alternate text for that element. Waller (2010, p.1) says,

> "A SAS ARRAY is a set of variables of the same type that you want to perform the same operation on. The set of variables is then referenced in the DATA step by the array name. The variables in the array are called the "elements" of the array."

Every word of that applies in JavaScript except for "of the same type". In JavaScript you can have mixed type arrays and if SAS would add that, it would make me very, very happy. Other people think the loose typing in JavaScript is going to bring about the end of civilization. You can go onto Stack Overflow and argue with them if you want.

However, you feel about types, arrays are a fundamental concept to any programming language, so mastering that concept is a step forward.

Variables and functions (or how SAS became the basis for popular educational games)

Truly understanding variables is another foundational idea – not just that they are not constants, but the concepts of type, format and scope. So, let's talk about that a little bit.

If you know anything about SAS, you might be thinking that I used my mad data analysis skills to figure out what works and what doesn't for games. While that is somewhat true, it is not at all my point here. In fact, learning SAS first helped me a lot when it came to actually MAKING games. No, there is not a lick of SAS code in our games, but the concepts and ideas came to me fairly easily because of my experience using SAS.

(If you are thinking I could have learned everything here from Python or C or whatever your favorite language is, I am sure you are right. The fact is, though, I didn't.)

Let me give you another example from the new, premium version of Making Camp. The object of this part of the game is to match as many synonyms as possible in one minute. This is what has to happen:

- On loading the page, randomly select a word to display on the screen, start the timer and music
- Show the number of seconds on the page, going down every second
- On the page, show 7 other words, 3 that are synonyms and 4 that are not synonyms, making sure that the correct and incorrect words show up in random order.
- If the player drags a correct word into the box, it turns green and adds 1 point to the score.
- If the player drags an incorrect word, the box turns red
- If all three choice boxes are filled, all the boxes are cleared and a new word and choice boxes are shown
- When the time is up, if the player has a perfect score, show a happy image and appropriate text.
- If the player doesn't have a perfect score, show a less happy image and appropriate text.
- When time is up, show a button the player can click to play again.

Here's a tiny bit of code from our latest game

```
$(document).ready(function ()
    //Timer script ;
      var time = 60000;
      var timer ;
```

This first bit just starts a script, and the beginning of a function that will execute when the document is ready. That is, I don't want JavaScript to try acting on elements that aren't loaded yet. My first exposure to writing functions was in the 1980s. It was a very significant event. I swear, I even remember the cramped graduate assistant office at the University of California, Riverside where I read my first book of SAS macros written by users. This is how user groups distributed things before the Internet. I thought the idea of writing my own functions was the most amazing thing I had ever heard.

Imagine me in my twenties, saying,

> "Wait a minute – you mean if there is a PROC I wanted, like if I wanted something to do the parallel analysis criterion for factor analysis, I could write that myself?"

Now, for the timer. Everyone knows what a variable is, or you do if you did anything with any language. Here, I am initializing the time to 60,000 milliseconds. Initializing a variable is another basic idea I learned from SAS. I'm going to use that other variable, timer, later to execute the myTimer function. For now, let's take a look at some of my code for the game:

```
//Timer script ;
function myTimer() {
    if (time > 0) {
        var nowTime = time/1000 ;
        document.getElementById("timer").innerText = nowTime  ;
        time = time - 1000;
    }
    else if (time <= 0) {
        document.getElementById("timer").innerText = "0";
        clearInterval(timer);
            $("#form1").hide();
            //IF ALL OF YOUR ANSWERS WERE CORRECT ;
            if (correct === boxmove)
            {
                $("#correct").text("PERFECT! You answered " + thesepts +
                    " correctly.");
                $("#correcto").slideDown('slow');
                playAudioLocal("../../sounds/correct1");
            }
            else  {
                $("#wrongo").show();
                $("#incorrect").text("You answered " + thesepts +
                   "correctly.").slideDown('slow');
                playAudioLocal("../../sounds/flute");
            }
            $("#redo").show();
    }
```

Except for a few specific details, everything in the script above, I learned or improved from using SAS.

IF- THEN-DO-END – instead of DO and END , I have an opening { and a closing } but it's the same thing.

If the time is greater than 0, the variable nowTime is going to be set to time divided by 1,000 since most people would prefer to see their time in seconds rather than milliseconds. By the way, nowTime is a local variable, defined within a function. Local variables is another idea I first learned from SAS macros, thank you very much.

The text of the element in the page named ‘timer’ is now set to whatever the number of seconds remaining is (nowTime). We deduct another milliseconds from time. ELSE – DO is another common SAS bit of code . If there is no time left, do all of this stuff, e.g., set the time value to 0, stop calling the timer function.

You can have nested IF-THEN-DO code in SAS, as I do here in my JavaScript.

Here, we have a JavaScript text function where I’m concatenating a string with a variable and then another string.

```
$("#incorrect").text("You answered " + thesepts + "
correctly.").slideDown('slow');
```

While SAS didn’t introduce me to text functions, because I’d had a couple of programming courses in college, it’s where I learned a lot of them. Concatenating texts, substrings of texts, index functions to find a character or string within a text – all of these and more are functions I became familiar with through SAS.

Two-dimensional Arrays

I have used SAS arrays since they first came out and were implicitly indexed. In other words, it’s been a minute. If one-dimensional arrays were great, two-dimensional arrays were great-squared. Some people will tell you that JavaScript does not have two-dimensional arrays and rather, you have an array of arrays. To those people, I say, “Bah, humbug!”

```
// This is the array of words for my game. The first is displayed as the word to match ;
// The next three words are synonyms and the last four words are incorrect answers ;
var words = [
  ["large", "big","enormous","gigantic","awkward","introspective","sane", "bulbous"],
    ["fast", "rapid","quick","speedy","awkward","boring","dull", "bulbous"],
    ["fat", "stout", "thick", "overweight", "thin", "unprofitable", "sense", "dazzling"],
    ["bad", "terrible", "not good", "awful", "couch", "sad", "ugly", "usual"],
    ["strange", "odd", "queer", "weird", "couch", "sad", "ugly", "happy"],
    ["rare", "uncommon","unusual","not typical","irate","musical","aromatic", "within"]
];
```

Also, if you didn’t know, now you know, SAS 9.4 has multi-dimensional arrays (SAS Institute, 2016)

STATISTICS

Systematic Random Sampling Saves the Day

You might wonder where statistics enters into making a game to teach synonyms. My challenge was to make sure that the choices were put in random order so that the first 3 boxes weren’t always the correct answer. I went through a lot of possible solutions where I tried to splice the array to pull out a word randomly used, then pull another random choice from the shortened array, using the length attribute.

Sometimes you can be too clever. After all of that, I realized there was a really simple solution. Pull out a random number. Take that and the rest of the items in the row, then start at the beginning again. Systematic random sampling. Yep. Super simple. Every useful programming language on earth has a random number function, another basic function I first encountered using SAS. First, we randomly pull a row out of the array. Then, we start with the n+1 word in that array, when n is a random number between 1 and 7.

```
var qnum =1 +Math.floor((Math.random() * 7)) ;
```

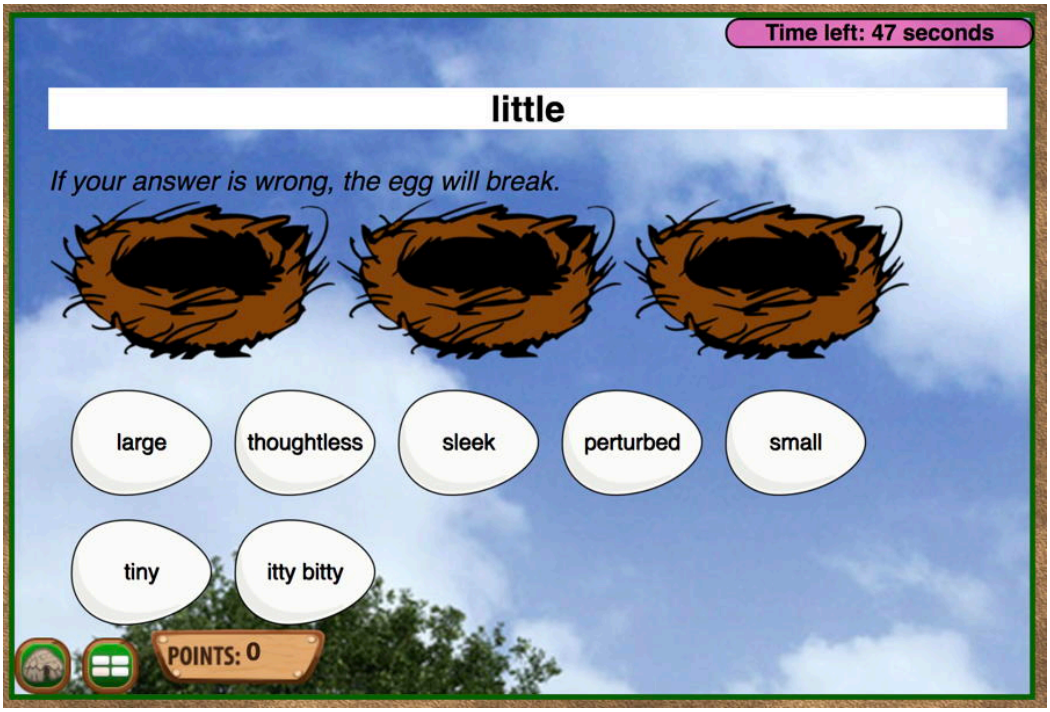

Figure 3. Synonym Game Screenshot

We pull the word that is in the n+1 position in the row and assign it to the first box. Then, the next box gets the next word in order. When we get to the end of the row, the next box will have the first synonym. So, if my random number is 5, the boxes for the choices are words # 5, 6, 7, 1 , 2, 3, 4 and boxes 4-6 are the correct answers.

SAS LETS YOU FOCUS ON THE STATISTICS THAT GET YOU PAID

One Friday last year, I was on the Spirit Lake Dakota Nation in North Dakota working with the Spirit Lake Vocational Rehabilitation Project, an impressively effective group of people who help tribal members with disabilities get and keep jobs. A research project used SAS to analyze the data to try to identify predictors of employment. Due to a delayed flight, I spent the night with my friend in Minot, discussing, among other things, the decline in native speakers of Cree, and not the herd of deer in her backyard, which was common place enough to pass without comment.

Saturday, I was back home in California, on a dinner cruise in Marina del Rey. We were discussing how to analyze the data on persistence in our games to show that the re-design, with a longer lead-in story line and a higher proportion of game play early on was effective. I suggested maybe we could use survival analysis. Really, it's the same scenario as how many people are alive after 2, 3 or 4 months or how many people kept playing the game after the 2nd, 3rd or 4th problem.

The deer, the large loud sea lion on the dock and I spent the exact same amount of time discussing the probability mass function for a Poisson distribution and proving the Central Limit Theorem. My point is, that everywhere I go, and that is a REALLY broad range of places, people are interested in the application of statistics, but SO much of school is focused on teaching how to compute the area under the normal curve or how to prove some theorem or computing coefficients using a calculator and plugging numbers into a formula, inverting matrices. I'm not sure how helpful that was to me as a student and I can guarantee you that the last time I computed the sums of squares without using a computer was about 35 years ago.

Whether you are using SAS, SPSS, Excel, R, JMP or any one of a dozen other statistical packages, it lets you focus on what's really important. Does age actually predict whether or not someone is employed? Do rural school districts have fewer bureaucratic barriers? Is this a reliable test? Did students who played these games improve their math scores?

When I was young, and many of the current statistical packages were either very new and limited or didn't yet exist, someone asked me if I was worried that I would be out of a job. I laughed and said no, because what computers were replacing was the computational part of statistics, and except for that tiny proportion of people who were going to be developing new statistics, the jobs were all going to be in applying formula, not proving them and certainly not computing them with a pencil and a piece of paper. A computer allows you to focus on what's important.

What IS important in data analysis?

The number of contracts I've had where people wanted me to tell them what to do I can count on one hand – and I've been in business 30 years. Generally, whether it is an executive in an organization where I'm an employee or a client for my consulting services, people don't want me to tell them what to do,

> "Hey, you should do a repeated measures ANOVA."

Nope, they want me to DO it. It's funny how often I find myself doing the same procedures for vastly different organizations, everywhere from the middle of Missouri to downtown Los Angeles to American Indian reservations in North Dakota to Santiago, Chile. Here are a couple of my go-to procedures.

PROC FREQ and PROC MEANS = More than data management

I always start off with a PROC FREQ for categorical data, a PROC MEANS for numeric data and a PROC CONTENTS to check for any obvious programs with data quality, such as a high proportion of missing data and variables out of range. This is a basic fact of life one learns as an experienced programmer – data are never exactly what you expect them to be. I could tell a lot of stories about people who embarrassed themselves in high profile situations by stating confidently that the mean age for fourth-graders was 13 because they didn't spot the one student's age that was entered as 99.

With PROC FREQ and PROC MEANS, not to mention UNIVARIATE, PROC PLOT for checking for outliers and linear relations and the CHARACTERIZE DATA task in SAS Enterprise Guide, checking your data is so easy it becomes a habit.

However, PROC FREQ and PROC MEANS are more than just a check for outliers. Fisher's Exact Test is one reason why PROC FREQ is my favorite proc.

Last year I computed Fisher's Exact Test using this teeny bit of code:

```
PROC FREQ DATA = install ;
    TABLES rural*install / CHISQ ;
```

to test whether urban school districts have significantly more bureaucratic barriers to using educational technology than rural districts (they do). Because there were only 17 districts in the sample, chi-square would not be appropriate. You might be asking,

> "Wait a minute! Isn't that just a PROC FREQ and a chi-square? How the heck did I get a Fisher's exact test from that?"

Well, it turns out that if you have a 2 x 2 table, SAS automatically computes the Fisher exact test, as well as several others. It also turns out that this is the same code (with different variables) I used for an analysis of whether patients treated in specialized units were less likely to die than those in the general hospital wards. See how easy it is to move from one field to another?

Confidence Limits for the Mean

Working with small samples in rural communities, I often don't have the luxury of a control group. I know this makes me sound like a terrible researcher and that I never read a quantitative methods or experimental design textbook. However, let me give you an example of the types of conversations I have all of the time.

> Me: I'd like to use your program as a control group. I'll come in and test all of your students and then two months later, I'll test them all again.

Principal/ Superintendent/ Program Director: You mean you want me to take up two periods of class / counseling time for your tests?

Me: Yes.

Them: You wouldn't actually be giving our students any services or educational program, you'd just be taking two hours from all of our students.

Me: Yes, and then I'll compare their results to those of the students who do get services.

Them: What do our students get out of it?

You can see where this conversation is going. Institutional Review Boards are cautious about having substantial incentives because then they feel very low income might be coerced into participating – for some of the people on our research, $10 is a lot of money. The result is that I don't always have a control group, but all is not lost. Being smarter than I look (yes, really), I often use standardized measures for which there is a lot of research documenting the mean and I can do a one-sample test.

```
proc means data=cesd_score alpha=.05 clm mean std ;
var cesdtotal ;
```

This will give me the 95% confidence interval for the mean and I can see if my sample is significantly different from the mean. For example, with a sample of 18 children from an American Indian reservation, the mean score on the CESD – C, a measure of depression, the mean score was 21. The cutoff for considering the respondent as showing depressive symptoms is 15. With a confidence interval from 15.6 to 26.4 I can say that there is a greater than 95% probability that the population mean fits the cutoff for depressive symptoms. Notice that the lower confidence limit still is above the screening cutoff point of 15.

The Big Five Procs

Because I do a lot of research that involves tests and surveys, two psychometric procedures come in for measures of everything from depression to math achievement, PROC FACTOR and PROC CORR. How many hypothetical factors does this test (of anything) measure? Is it reliable? Are there differences between groups on a numeric variable (PROC GLM/ PROC MIXED) ? Is the production of one cost center in a factory significantly higher than another? Did the performance of one school improve more than the other? PROC LOGISTIC - the same procedure used to predict which missile would blow up on the launch pad is used to predict which family will choose to place their child with a disability in an institution. Yes, the content area knowledge is important, but if you have gotten all of the other pieces, the knowledge of biostatistics or aerospace manufacturing might be provided by another member of your team until you get up to speed. Speaking of team …

TEAMWORK

No matter where you start, whether with SAS at an insurance company or C# at a manufacturing plant, certain key skills will transfer to any kind of programming because you are that most coveted of commodities – an EXPERIENCED programmer.

As an experienced programmer, you are aware that the 'latest thing' is not always the best thing. I'm not against the latest thing, whether it is react or ember or Python games on Raspberry Pi or whatever it is today. My objection is to the fallacy that it is the only thing or even the most important thing. In fact, up to this year, I have used chapters in a 2005 book by Cody and Smith to teach factor analysis and MANOVA because these still apply. (I do supplement these with readings on PROC MIXED and PROC LOGISTIC because sometimes the newer things are cool, too).

There are several things you learn as an experienced programmer:

Testing. Testing. Testing. I said it three times because it was important. I think I will say it again. Testing. Testing. If you are developing an application, you need to test EVERYTHING. If I had a dollar for

every time someone told me, “I tested everything but …” I would never need to seek investor funding again, I would just pull money from the piles in every room in my house. However much you think you need to test your software, you are wrong. The answer is, “More.” You need to test it on other machines besides yours. I learned this from SAS code that ran on Mac (yes, there was SAS on Mac a very long time ago) but not on Windows or on Windows but not on Unix. You can’t develop apps that don’t work on anything older than Windows 10 because that is only half of people who run Windows and less than 20% of the total market. SAS is actually a good starting point for learning this because it runs on a lot of devices with few changes but you do need to change the LIBNAME and FILENAME statements, for example. Similarly, we make games now that run on Mac, Windows, iOS and Android . At a minimum, you need to do a separate build, but sometimes you need to make major changes. For example, Android has some limitations on app size that iOS does not. Test whether your software installs. Test whether it opens. Test the most basic applications. For SAS, this would be creating a temporary data set, reading in data with a DATALINES statement and doing a PROC MEANS. For our educational games, it might be playing all the way through getting all of the answers correct. Test extreme cases. For SAS this might be merging several enormous datasets, applying user created formats, calling macros to manipulate the data and then performing a Multivariate Analysis of Variance. For our games, it would mean getting every single problem wrong and quitting the game and logging back in many times, maybe after every problem. It would include entering completely illogical numbers, say, that you had picked 9,145,087 berries and seeing if the program really tried to put over 9 million berries in the baskets. I’m sure you can think of some more extreme cases, but you get the idea.

I can’t emphasize testing enough. Experienced developers realize that their work will be used by a public that does not completely understand how the software is supposed to work, or doesn’t care. Real testing includes things like wandering off the path in a game with the path clearly marked, “just to see what would happen”. It is having people enter “as often as I can” instead of male or female for sex.

Debugging is most of your life as a software developer. Basically, you write code for a few minutes and then swear and debug it for hours. Once you have a little experience, you learn to test and debug as you go and never write huge blocks of code that you then find doesn’t work and you have to figure out where in there the bugs occurred. You will learn all types of tricks of the trade for debugging. These include, printing out the first few records of a data set to make sure it looks like you expect. With JavaScript it might be writing the value of a variable to the console. Either way, the point is the same, you are testing little bits of code as you go and seeing that the result is what you expected. You also learn to debug all the way through. With SAS, you might apply the statements you have written to a data set in the documentation and verify that you got the same results. With a game, you might collect all of the objects in a scene and then check that the variable recording the number of objects is equal to what you expect.

In any program that you are writing, you learn to break it into modules and test each of those modules. So you are debugging it in chunks by writing out the values of some number both in small steps, say even after each statement if you are really running into problems, and also in medium steps, say, at the end of each SAS data step or procedure, or after the execution of each function. I’m not saying that inexperienced programmers don’t debug their code because obviously they do. No one always writes code that works perfectly the whole time. What I am saying, is that the more experience you have, particularly working as part of a team, the more debugging techniques that you have figured out for yourself, and the more ideas you have picked up from your colleagues.

A third part of being a grown-up software developer is learning to document the software. Comments are your friend. I had a colleague who made fun of me for how much I would put comments in the code but when the next year we had to do a similar project again I could turn to him and say, “who’s laughing now?” I have never met the programmer who enjoyed writing documentation. I have met a lot of programmers who were happy they had written it. if you are always chasing the latest thing, you might not be in that situation where you need to revisit something that you did a year or two ago. If you are not part of the team, you probably are not worrying whether some nonexistent team member can understand your code. On the contrary, you might be trying some really cool new ideas just because they’re interesting. I’m not against that, in fact, I completely understand. However, you need to document those cool new things. And if you take the attitude, “well, everyone should be expected to know the function call to integrate Lua with PHP”, you have some hard life lessons ahead of you.

Here's why being part of a software development is usually a crucial aspect of your career progress – all of the things I mentioned, most people don't really want to do. Testing isn't nearly as fun as writing code. No one likes to write documentation. Everyone knows that debugging is crucial but it usually seems at the time as if putting in all of those statements to check every single variable's value after every manipulation is so time-consuming when you are just sure it was correct anyway. When you're on a team, you can't get away with cutting corners and skipping the not fun parts nearly so much. You also realize how crucial those parts are when other people on the team have no idea what you were doing when you wrote that function or macro or nested do loop. Sorry, but I don't think a weekend hackathon is any substitution no matter how many prizes you won. Not unless you had to return to the same hackathon six months later and update the project with a completely new set of people.

I don't want to leave you all depressed if your only qualification is as a self-taught programmer with pet projects. So, I do have two pieces of advice. For the debugging part there are plenty of software conferences you can attend (like SAS Global Forum), and find sessions on tips for debugging software. You may also meet people at those conferences that you could end up working with on a team for some project interests all of you.

CONCLUSION

An astute reader (no doubt the only type of reader who would read this paper) may have noticed that there is not a lot of detailed SAS code. That's the point. It's not the specific SAS procedures that make you valuable in a wide range of industries and positions. It's the procedures you have learned as far as solving a problem, documenting code, debugging, validating your data and working on a team. It's also the concepts you have understood that are the foundation of programming.

There is another key point in "SAS as foundation" that is often overlooked. Technical fields in the U.S. have a severe under-representation of female, Hispanic, Native American and African-American professionals, in part related to the lower number of those groups in Computer Science majors. This isn't a complete explanation, for example, Stanford University Computer Science program is 30% female but Google workforce is only 18% female, Facebook, 16% (Cueto, 2015). Still, drawing technical staff from areas with greater diversity could both increase the number of people available for the technical workforce and the diversity of that pool.

Where do people learn SAS? Often in majors with greater gender and ethnic diversity than Computer Science. Students often learn SAS when they return to graduate school for a degree in psychology, education, public health or other 'non-technical' major. Pursuing a decades-long career in clinical trials or credit risk analysis is one option. However, for an experienced SAS programmer, who understands the basic structure of software applications and the processes of development, the possibilities are wide open.

REFERENCES

Burns Ortiz, M. (2014). How "Easy" It Is for Women-Led Startups To Get Funding. http://www.7generationgames.com/2014/11/14/show-me-the-money-how-easy-it-is-for-women-led-startups-to-get-funding/

Carpenter, A. (2007). Carpenter's Complete Guide to the SAS REPORT Procedure. Cary, NC. SAS Press.

Cody, R. P. & Smith, J. K. (2005). Applied Statistics and the SAS Programming Language (5th ed).

Cueto, J. (2015). Race and gender among Computer Science majors at Stanford. https://medium.com/@jcueto/race-and-gender-among-computer-science-majors-at-stanford-3824c4062e3a

Franken, A. (2005). The truth (with jokes). New York: Penguin Books.

SAS Institute Inc. (2009). Base SAS ® 9.2 Procedures Guide. Cary, NC: SAS Institute Inc.

SAS Institute (2016). Multidimensional Arrays: Creating and Processing. In

SAS® 9.4 Language Reference: Concepts, Sixth Edition, p.573-4.

Waller, J. L. (2010) How to Use ARRAYs and DO Loops: Do I DO OVER or Do I DO i? Paper presented at SAS Global Forum. http://support.sas.com/resources/papers/proceedings10/158-2010.pdf

ACKNOWLEDGMENTS

Thanks to Adekola Togunloju for making the synonyms game much prettier and to Dennis De Mars for his insightful comments.

CONTACT INFORMATION

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

AnnMaria De Mars
7 Generation Games
565 Monjitas, Piso 5
Santiago Centro, Santiago, Chile
(56) 990006617
annmaria@7generationgames.com
www.7generationgames.com