Paper PD02

A Beginner's Journey with SAS: One Small SAS for Man...

Dale Armstrong, Sekharico Informatics Ltd, Edinburgh, United Kingdom

Abstract

Four years ago, I had never even heard of SAS. After leaving my job in physical sciences and transitioning into a statistical programming role, I would like to share some of my insights and learnings from this process.

Why did I choose SAS out of the multitude of other languages and career choices? How did I get started, what was my mind set, what could I bring to the table? How was my first year, my first project, my first large project; what have I learnt over my current career? What went well, what went terribly? What's next?

Introduction

I started my career as a statistical programmer in April 2020, not long after the first lockdown period started in the UK. In this paper, I will summarise the overall process of transitioning into this career and hopefully illuminate aspects of this experience that will help the reader in navigating either their own journey or that of someone close to them, such as a trainee employee.

There are currently many ways to become a programmer, be it using SAS or any other language, from a structured academic or corporate pathway all the way through to starting out completely self-taught. Within each of these routes, differences abound in terms of learner circumstances, abilities and industry opportunities.

Due to the large number of pathways into programming, opinions on which route is most efficient can be just as varied, with many differing opinions on how to progress.

This paper will provide a year-by-year breakdown of the pathway that I took into the industry, highlighting a selection of what I believe to have been key points in my development. With each point, I will discuss how I feel it impacted my journey, including the benefits, drawbacks or struggles it presented.

Getting Started

Of all the things I have heard about becoming a programmer, the one that remains completely true for me is that it is a long process. So why would I embark on such an involved task?

In 2017, I had just bought a new high-powered laptop, and whilst it was amazing at providing entertainment, I wanted to see what else it was capable of.

Making my first foray into the world of programming, I stumbled across a unique culture that was home to hobbyists, professionals and everyone in between. Among programmers I encountered online, I noticed a general shared aim towards the overall betterment of everyone involved and the sharing of ideas and concepts. This not only felt worlds away from my current career, but appealed to me as something I could experiment with while being as involved or uninvolved as I wished. All I needed to get started was a computer.

The more I researched the programming industry, the more I discovered its reputation for good work-life balance, flexible working, rewarding pay, a wide range of opportunities and overall high-quality employment. Sifting through all

sides of the reputation of programming, from tales of highly paid geniuses solving the world's problems to stories of labour disputes and terrible working conditions, I took somewhere in the middle to be the closest to the truth.

Thinking ahead towards the prospect of a career with greater longevity and fulfilment than my current one, I realised I was keen to learn a skill that would provide me with a valuable trade. Whilst entry to my previous industry had involved a high barrier in terms of specific degree qualifications, with few prospects for even highly-skilled workers to progress beyond under-rewarded lab work, coding appealed to me for its relatively low barrier to entry. Through becoming a programmer, I saw the potential to acquire a skill that could be applied flexibly across multiple fields, with varied opportunities and competitive wages.

In other words, a major factor in my decision to make a career change was simply dissatisfaction and disillusionment with my job at the time. Whilst this alone was not the reason that I began programming, it was ultimately the catalyst that led to my next steps.

In summary, my journey into programming did not start with a calling. Rather, it was motivated by the pressures of an incompatible career giving me clarity on what I really wanted out of my work and the rest of my life. This clarity is what I used as a guide to choose programming.

Years 1–2 (2017–2018): First Steps into Coding

My initial exploration of programming was haphazard at best. Without any frame of reference or family friends to call upon, I had no means of immediately entering the industry.

My very first step was to consider what I could already bring to the table. My previous career in biotechnology had provided me with good soft skills, a solid grasp of general biological science and strong research abilities. Whilst most of these skills would not help me in the immediate term, I predicted that they would become more and more valuable the further I progressed.

In this initial stage, I decided to stay in my existing job, as its stable hours and income were key to allowing me the freedom for study and research.

Whilst I spent a great deal of time worrying over which language to start learning, in hindsight, the most important step was simply getting started. At the time, however, I had a lot of considerations on my mind. Which language was most in demand? Which would allow me to do what I wanted in future? Were some languages easier than others? Was there a lot of support for a given language?

Eventually I settled on Python, with its syntactical proximity to English and reputation as a great beginner language with a wealth of resources.

Despite my eventual settling on SAS, I still think this was a good first language to choose, as being so popular meant that there was a lot of support available to me.

The one downside of choosing Python was the sheer number of learning resources to choose from, which left me with almost too many options. Below, I outline the pros and cons of the approaches I tried.

My first port of call in the search for learning resources was to look online. I found the barrier to entry to these to be generally very low, with the vast majority being completely free. While paid materials did exist, I found them to be unnecessary. You could find anything you needed – or something close enough – for free. From Stack Overflow, YouTubers such as Corey Schafer, written papers, forums and dedicated help websites, there

was more than enough available to cover the bases.

That said, online sources required a preliminary check prior to committing to them. The main issue I came across was the hit or miss nature of some resources, in which quality of explanation would vary wildly or vitals steps would be skipped or assumed. As such, things which would seem simple to an experienced programmer would quickly derail a beginner like myself. This was only exaggerated by the fact that there was extremely limited scope for feedback from more experienced programmers. Learning on my own, I was largely reliant on whether a source had anticipated a given issue arising or if the person (or community) I was learning from was still active enough to respond to my queries.

The second resource I investigated was textbooks. These presented a higher barrier in the sense that they required learners to purchase a book and commit a great deal of time to that single resource.

I found textbooks to be extremely good at teaching fundamental ideas, covering topics with a lot more depth and comprehensiveness than many online resources, and building ideas on top of one another to quite complex levels. They were also excellent reference materials and made me aware of several knowledge areas that I simply had not encountered in online tutorials.

Picking the correct textbooks takes some research, as some can fall out of date. For example, it is an unnecessary hurdle to be learning from a book working in version 2.0 when the current version is 3.0. Whilst in hindsight this may not be a major issue, confusion can arise when it comes to using outside sources to solve in-book issues.

As with free online resources, feedback is also an issue. Although some books provide extra resources online, authors are not typically available to field questions.

Moving beyond self-directed, mostly budget-friendly learning methods, coding boot camps were another option to consider. These appealed to me as a potential route, as they offered high-intensity crash courses which introduced learners to programming and promised to help graduates find employment over the course of six months or so. I was lucky to have a boot camp, the now liquidated CodeClan, based locally to me. I inquired, attending some taster classes and open days to decide if it was the correct path for me.

CodeClan seemed well structured and connected with local industry, with a record of successful graduates and local government backing. However, with a full-time course starting at £8,000 for three months, it was very much not the type of commitment I wanted to make so early in my learning.

Out of all the approaches I tried, by far my favourite method of learning was through networking. The culture in the programming industry creates easy-to-access talks and conferences for amateurs to attend, learning from and meeting others involved in the programming industry. For new programmers, these networking opportunities also provide valuable insight into working for local companies. Personally, getting involved in such events allowed me not only to experience introductions to new topics by people well versed in them, but also to really see what kind of work I would be getting into. Moreover, it gave me an idea of the types of jobs available and what kind of people I might be working with – all while making connections in the hope of finding a role for myself.

Although I am a supporter of this type of networking, it does have some drawbacks. First, you will not become a better

technical programmer by attending these events. As such, maintaining a balancing act of both creating a network and doing the hard work of programming is key to success. Second, many events will not necessarily apply to your particular interests or be pitched at an appropriate level for beginner programmers.

Ultimately, the biggest pitfall I encountered was allowing these events to feel more important to my development than they actually were. Whilst they may form a key piece of the puzzle, they are generally not where you build projects or practise techniques.

To conclude, in my first year of studying I found that my preferred method of acquiring programming skills was to rely heavily on a textbook to start with before then gradually transitioning to using more and more online resources as I became able to interpret higher-level tasks.

This method allowed me to control my own pace, which in turn enabled me to focus on the materials that worked for me rather than being torn between the approaches of multiple learning guides. This also helped with effectively managing my time, resources and mental wellbeing, making the process a sustainable long-term undertaking that avoided burnout. As I progressed, I found myself able to commit to more without overly stretching myself. Overall, patience was key.

Year 3 (2019): Houston, We Have a Problem

Moving into the third year of my studies, I began to gain momentum as I worked, each step taking me closer to my goal of a career in programming. However, there came a point where I felt the progress drop off. As it became clear that the study methods I had followed thus far could no longer sustain my growth, it was time to take another approach in order to commit seriously to a career change.

Whilst I had by now covered the syntax and other basics of Python, being able to apply that knowledge to self-guided projects was still incredibly hard, particularly when it came to finding projects which were not too technical or involved. As I found myself getting tripped up on problems well above my ability to fix and becoming discouraged, I realised that something needed to shift.

During my networking events, I had managed to make a few friends and regular acquaintances, to the point where one of them, a Mr Anil Sekhari, offered to become my mentor. Anil was a statistical programmer with decades of experience in SAS, running a SAS consultancy business through which he had previously worked with newer programmers.

Through Anil's mentorship, I learnt of the role of programming in data science, clinical trials and ultimately SAS. As the first official step of our working relationship, we agreed to one year of supported study whilst I held down my existing job. This meant limited financial risk for both of us while also allowing us to assess my progress before proceeding to employment – our ultimate goal being that I end up a SAS programming guru.

The first benefit of this mentorship was gaining a clear and structured path to a legitimate goal, as prior to this I had not had anything concrete to aim for. This goal provided motivation, enabling me to better use my time. In fact, it overhauled the way I studied overnight.

With someone there to guide my training plan, I went from bouncing between projects which were either too complex or too simple to working on tasks with a gradual increase in complexity. Whilst the majority of this work was still self-driven, Anil was able to direct me to the topics and resources that would help me overcome programming issues I came across.

Another notable improvement was that I found myself finally able to access near-immediate feedback that was tailored to my exact problems, with Anil often foreseeing pitfalls such as the temptation to trust the frequency procedure to correctly calculate percentages, whereas best practice would be to calculate them manually. With this kind of support, I found myself no longer getting caught out or wasting time on dead-end solutions. Instead, I would find my own way to fulfil a given goal, after which we would talk about the strengths and weaknesses of that method. I would then be challenged to complete the task again via another method.

Through this process, I came to realise that I often erred towards code which was too complex. Through feedback, I learnt that simplicity was key when it came to ensuring that my code could be understood and re-used by other programmers.

Another change that came about through mentorship was a shift in the pressure I was under, as suddenly I was not just answering to myself. I essentially went from a hobbyist to someone gearing up for a job, which took a while to get accustomed to.

As my mentor was under similar time constraints, it was key that I managed my output in keeping with agreed deadlines to ensure that it could be reviewed in time. I did not want poor time management to cause me to miss out on Anil's contributions and feedback. While this did not mean that I always timed things up perfectly, with good communication, we managed to make good progress.

An unforeseen hurdle throughout my training was the difference in experience levels between the two of us. The things I was learning were second nature to Anil, so at times, it took a lot of back-and-forth to break down the basic meaning of certain concepts into a learnable format.

Despite these occasional issues, the wealth of experience and contacts in the industry provided by Anil represented something of incredible value: a potential foot in the door and a senior figure in the industry who would hopefully vouch for me when it came to future opportunities. By the same token, however, I had to be prepared to perform well given the chance at any one of those opportunities, as it was more than just my own reputation being put on the line.

As part of the mentorship programme, I was provided with a comprehensive training plan. According to this plan, I was to work using dummy data to understand topics in the following order:

- CDISC: Clinical Data Interchange Standards Consortium
- SDTM: Study Data Tabulation Model
- ADaM: Analysis Data Model
- TFLS: Tables, Figures and Listings
- Statistics: how to manipulate data into statistical models

The structure of this training plan was designed to replicate what would typically be expected in a SAS programming role. This approach gave me a good overview of a realistic industry workflow and was scaled so that the data was considerably smaller than a full study, allowing me to move quickly through each stage and have my work regularly reviewed without too much effort on Anil's part.

One example of the value of this method was in practising how to handle data, as up until that point I had been purely focused on the technicalities of the coding language and had not been aware of the practical importance of visualising data and anticipating how it may need to be handled, such as when transposing orientation.

As described above, having access to a learning plan and tailored feedback revolutionised my learning. Under this

structure, I was able to gain a taste for what a real SAS programming job was, with all the benefits of a safe space to learn in and the freedom to find flexible solutions to problems while honing my data handling skills.

With the ultimate goal of me being employed by Anil, it was important for him to get a feel for how I would perform in that environment and to assess whether the effort to train me would pay off.

In summary, the training plan we followed provided a ground-up introduction to working as part of a SAS programming team handling clinical trial data. It allowed me to see the entire clinical process, which may not have been possible with a more specialised company, and to manage my expectations regarding my anticipated future role.

Years 4–Current (2020–Current): Becoming a Programmer

With one year of mentorship under my belt, it was time to go for a professional contract. Whilst my studies were not quite complete, I was hitting the limit in what could be achieved with self-study. This meant quitting my current job and joining Anil's company, Sekharico Informatics Ltd, as a Junior Statistical Programmer. Despite some nerves around the risk, I was ready to commit to a career change.

The first consultancy project I was assigned was a small Covid-19 respiratory trial to be completed over the span of a month. I was to produce a handful of ADaM datasets with accompanying tables and figures.

The size of this project provided a number of benefits. First, I was excited to work on a live project. Second, it was only slightly more challenging than what I had already been working on, meaning less chance of dropping me in at the deep end. As Anil was the Validation Programmer and would be checking all details, I knew I had to perform. Finally, from Anil's point of view, it was easy to understand what was going on and help me if need be. With all of these factors in play, this first consultancy was a great learning experience that ended successfully and allowed for a good amount of reflection and feedback. Specifically, my main point for improvement was to speed up my work rate whilst maintaining quality.

Having gained experience of consultancy work, we began to look into placing me as a junior programmer in the same team that Anil was already part of, developing the ISS/ISE submission package deliverables for a new psoriasis compound. While Anil initially helped me considerably in understanding assignments, finding resources and focusing my priorities, this was a considerable step up in complexity and workload.

My first tasks, once I was orientated, were tables, listings and any lower-level extra tasks, such as creating minor macros. It was eye-opening to see that, coming into an already established study, there was less call for creating code from scratch. Instead, understanding, rewriting and repurposing already present code became a large part of my job.

It was around five months before I was placed on study level ADaM dataset reviews and simple dataset creation (ADSL). After around eight months, I was then entrusted with more complex work, including multiple imputation and complex figures, culminating in FDA response work where I was instrumental in supporting programmers drafted in from other departments.

Throughout the entire process, I felt as if whenever I got settled into one type of work I would have the rug pulled out from beneath me and be sent straight back to square one with the next task. In hindsight, however, nothing was ever beyond my understanding. In fact, I found

the constant pressure helped me to continually improve my ability.

That is not to say, however, that I did not make mistakes whilst on project. In anticipation of this, Anil did a good job of forewarning me of a lot of the errors to look out for, generally setting up good habits so they would not be an issue and keeping me away from critical infrastructure until I had shown competency on lower-level work. It was only then that he would expose me to more vital parts of the process, such as reviewing datasets from outsourced companies prior to creating in-house datasets.

With my workload ramping up and often having to work longer hours, it became clear that I would not be able to keep up with my existing self-study routine. Equally, however, I could not have my professional development dictated purely by whatever happened to be within my workload.

Creating time for self-study required a reeling back a lot of my networking meetings, which were still recovering after the pandemic; adjusting my expectations regarding how quickly I could complete a given topic, and generally working through textbooks over a longer period of time.

Around one year after the start of this consultancy project, I moved into introducing myself to statistical procedures: a complex topic, but with the basics being covered at work, I was free to explore more advanced concepts in my own time.

During this period, one thing I was conscious of was the risk of burning out from doing too much coding. Indeed, I found it quite intimidating to be working alongside such experienced programmers (with an average of around 15 years in the industry). However, the team turned out to provide quite the benefit to my development, and I found that if I was honest about my experience while also showing a willingness to get involved, I could still be an integral part of the group.

Whilst one thing I was not expecting was a communication issue in that I could not “talk shop” and distil complex ideas into simple terms like those around me were able to, constant exposure ultimately remedied this issue.

One factor in my success at ingratiating myself into the team as quickly as I did was my soft skills, which I had developed in previous jobs. Despite being unable to contribute fully on the technical side of things, I did not disrupt team cohesion.

As I progressed on this long-term consultancy assignment, something I had to be very aware of was avoiding bad habits. With so many different programs being worked on by multiple programmers under varying levels of pressure, there was always a good variety of code being produced by those around me. A key part of my job became not just to use this code, but to try to learn it and improve upon it. As the code was often written by people far more experienced than myself, my first instinct was to assume their practice was the best. With time, however, I was made aware that it may well be the case that much of the code I was looking at was simply the best that could be created in the time made available to the programmer under tight timelines.

With my relatively inexperienced outlook, I had a recurring issue of being overly trusting of the existing code written by much more experienced people without fully verifying it myself to look for any potential issues. In other words, I struggled to look at the work produced by others from an objective standpoint, which caused quite a few issues when it came to verifying work done prior to me joining the team.

As I progressed, I began to routinely look for where corners may have been cut and how out of date the code was due to

dataset changes. I also learnt to spot code that was overly complex or had been hastily squeezed in from another program. It was then down to me to use what limited time I had to make it more accessible. Anil was a big help when it came to this kind of task, quickly unearthing previous programs which were a close fit for what I would need.

All in all, I considered my first year of working on this long-term consultancy project a success. Although ups and downs were present, the client was happy to extend my involvement in this project for another year thanks to my performance.

Conclusion

Throughout this paper I have shared my experiences, the steps I took, the people involved and the overall journey to starting my programming career.

If someone else were to replicate this process, I would recommend the following:

- Understand what you want out of programming. This will be key when it comes to deciding whether you truly want to undergo the challenging process of becoming a programmer.
- When starting out, focus on getting out into programming networks first and foremost. This will help you to create a space in which you can stay aware of any opportunities or changes to the local coding scene and establish a support system to help tackle any hiccups you encounter along the way.
- Start coding with a beginner-level textbook and then use that as a strong foundation from which to expand into online tutorials. When you feel ready, you can then use your network to try and find a mentor who will take you that last step and give you an introduction into the industry proper.
- Once you start your first role, seek support from the team around you. They will get you up to speed faster than any other method. Engage with them regularly and you will progress much faster than trying to go it alone. Be aware of your mistakes and your successes both to motivate you and to show you what your next improvements need to be.

While this is what worked and continues to work for me, I do not doubt that others have found their own paths to success.

I hope that this provides a useful case study for anyone who is starting out or supporting someone in a similar position.

ACKNOWLEDGMENTS

Thank you to Anil Sekhari for helping through my professional journey and to Elly Darrah for proofreading my paper.

CONTACT INFORMATION

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

Dale Armstrong

Sekharico Informatics Ltd

5 South Charlotte Street

Edinburgh

UK

EH2 4AN

Email: dale.armstrong@sekharico.com

Web: https://www.sekharico.com/