PharmaSUG 2018 - Paper LD-19

Statistical Programming: Revolution, Evolution, or Status Quo?

Frank DiIorio, CodeCrafters, Inc.; Jeff Abolafia, Rho, Inc.

ABSTRACT

Opinions are plentiful about the radical transformation of statistical programming in the pharmaceutical industry in the past 20 years. They usually contend that programmers must become conversant in languages other than SAS®, acquire data science skills, and improve communication and other "soft" skills. But is all that really necessary? If not, what are the implications for programmers, managers, and their organizations?

This paper reviews the evolution of both the pharma environment and the role of the stat programmer, focusing on the skill sets required for submission deliverables. It also summarizes interviews with managers in pharmaceutical companies and CROs, focusing on how their technology and deliverables have changed. What new skills are required? Which have persisted? What are the implications of the changed submissions landscape for the statistical programmer?

Readers should come away with an understanding of the changing organizational skill set and the impact of this change for the statistical programmer, in particular.

INTRODUCTION

It's not news that the number and complexity of deliverables in FDA submissions has notably increased in the past decade. It's also no revelation that change in the pharmaceutical industry can be characterized as less than rapid. Some reasons for the industry's slow pace can be attributed to the requirements imposed by FDA compliance, the inherent caution of pharma to adopt new technology, and the general organizational issues that arise when large companies are faced with managing organizational change.

The statistical programming department is a critical cog in the deliverables "machine" and is the focal point of this paper. At the authors' workplace (described in detail in the next section), the department's responsibilities include the creation and validation of datasets, tables, listings, and displays that go into the Clinical Study Report. The group also prepares the Reviewers Guide and handles ad hoc requests from sponsors.

One approach to managing the FDA's increasing number and type of deliverables is to broaden the scope of statistical programming, requiring the stat programming skill set expand in order to create a wider variety of output. In this view of the pharma programming world, it is incumbent upon the programming team to gain familiarity with any tools that will satisfy timely, high-quality submissions. This places data science methods, new programming languages, and other technologies in the organizational repertoire of pharma and CROs.

But is this expanded stat programming role really necessary? Have the demands made upon the statistical programmer actually changed that much? If so, how does the programmers' organization respond? If not, what are the benefits and costs of only a *modestly* changed status quo? This paper addresses these questions. We begin with a comparison of the submission environment ca. 2000 and now, then discuss the implications for skill sets and workflow. After a discussion of the pros and cons of a near status quo approach, we summarize findings from a survey of stat programming departments in other companies.

WHO WE ARE: SOME CONTEXT

Our discussion and opinions about the evolution of statistical computing is, of course, informed by our own workplace. In 1984, Rho Inc. was, like many other tech companies, born in a basement. Thirty-five years later, it is now a full-service CRO with over 200 commercial clients, while remaining true to its original mission: "Deliver the highest quality work product and meet aggressive deadlines without sacrificing integrity or rigor." The work load is a mix of federal research and support for pharmaceutical companies. To date we have prepared over 100 FDA submissions.

Like many companies that produce submission packages, Rho is database agnostic but SAS-centric in its creation of deliverables. Clinical data supplied by clients or produced in-house is held in a variety of formats – SAS, Excel, Oracle, *et al*. Output, however (the TFLs and datasets for the submissions package), is SAS based. This is not simply because that's the way we've always done it and it's the best way. All our commercial contracts specify that datasets and TFLs be generated with SAS; some even require delivery of the SAS programs themselves.

How the submission workflow is handled is what sets Rho apart from many of its peers: since the early 2000's we have developed metadata-driven processes that reduce the creation time and improve the quality of submission deliverables. Rho's focus on metadata has convinced us that exploiting metadata is the best approach to satisfying new client requirements and for making existing processes more efficient. It also informs some unique organizational responses. First, statisticians and stat programmers are expected to create specs. Second, the metadata focus resulted in the creation of a team dedicated to development, maintenance, and expansion of the database, user interface, and end-user tools.

THE SUBMISSION ENVIRONMENT: THEN AND NOW

The computing needs of pharma and CROs are determined to a large degree by the FDA. Changes to its requirements (type and format of deliverables, proof of validation, need for supplemental documentation) typically result in increased and more complex compliance requirements for pharma/CROs. To understand the changing demands on stat programmers over the past 20 years, it's useful to provide some context by examining the evolution of submission requirements.

SUBMISSIONS CA. 2000

Today's detailed, structured submission standards were barely a gleam in the regulatory "eye" of the FDA in 2000. In an environment where both electronic and paper-based submissions were accepted, it shouldn't be surprising that the pace of innovation was slow. The absence of standards for datasets and their documentation meant reviewers often spent considerable time simply becoming familiar with a submission and learning the idiosyncrasies of the study's data and documentation. (About three quarters of reviewers' time was spent doing what amounts to data management rather than actual regulatory review.) This inevitably resulted in lengthy review times, which shrank profitable time on patent for the drug's sponsor.

The submission deliverables then would be recognizable today: raw and analysis datasets, TFLs, patient profiles – all SAS-generated – and documentation ("define.pdf"). A notable feature of these deliverables is that they were validated only to the internal standards used by the pharma and/or CRO; a single, well defined set of validation rules applicable to all sponsors and studies was not feasible. The well documented benefits of data standards (among these, consistent data representation, the possibility of standard validation) were an impetus for the creation of the data and documentation standards pioneered by CDISC, starting in 1997.

EVOLUTION

Paperless, electronic submissions became common, then mandatory, in the 2000's, moving the process a step forward. The advantages of paperless submissions were, however, still offset by reviewer ramp-up time. There was still a need for standardization of how data points were coded, identified, and organized. In 1997 CDISC began addressing this need, ultimately creating standards for raw data (SDTM), documentation (define.xml), and analysis data (ADaM). The data standards in particular had two major effects. Submission materials could be reviewed more efficiently, since data representation and

organization would be similar across sponsors. Also, standardization meant that different sponsors' data from similar therapeutic areas could be combined, thus facilitating meta-analysis.

The emergence of well-defined standards also created the possibility for the development of validation tools. The FDA endorsed the use of a standardized set of validation rules, meaning deliverables would need to adhere to *both* internal (pharma, CRO) and CDISC standards. The validation report confirms that the SDTM, ADaM, and documentation files adhere to the standard, leaving the sponsor to document why deviations from the standard could not be avoided. This documentation – part of the Reviewer's Guide – essentially became another new deliverable, and contains material that does not readily fit into define.xml.

TODAY'S DELIVERABLES

Still, today's deliverables are not appreciably different than they were 20 years ago: SDTM and ADaM datasets are essentially formalized descendants of raw and analysis datasets. TFLs are often more complex, but tools (both enhanced and new) have kept pace. define.XML is a more rigorous and technically demanding format than the earlier define.pdf, but can be easily created by home-grown and third-party tools. And while proof of conformance to CDISC standards is not a formal requirement for the submission package, it is hard to imagine a reviewer who would not want such a report (particularly when some data errors cannot be resolved). As with the define file, off-the-shelf validation tools are readily available. The impact of the validation requirement is, for the stat programmer, time spent on understanding and resolving errors. In a sense, the only truly new deliverable is the Reviewer's Guide; while not officially required, the FDA "strongly recommends" their inclusion in the package.

PRODUCING DELIVERABLES: SAS STILL RULES

So, while the deliverables comprising the submission have evolved somewhat over the years, the same isn't necessarily true for the tools used to produce the package. A stat programmer from the early 2000's would not feel out of place in today's environment. Indeed, as our survey (discussed in "What Do Our Peers Think?", below) suggests, even with the expansion and standardization of deliverables, the process is still dominated by SAS.

SAS has for decades maintained its role as the *lingua franca* of the pharmaceutical industry. Data must be submitted as SAS transport files. Since the transport format is in the public domain, other software can create the files. But if TFLs are produced using SAS (even if R might be used for Figures) it makes sense to keep as much of the process as possible within the SAS ecosystem.

The SAS language has grown over the years in a variety of ways:

- Increased ODS functionality
- Expanded DATA step capabilities
- New functions and options
- New and enhanced procedures
- Increased availability (platforms such as SAS Grid, SAS Enterprise Guide, and SAS Studio)

Pharma and CROs can exploit these enhancements to strengthen their stat programming code base so that high-quality, compliant deliverables can be created more efficiently. This product-centric approach is the polar opposite of "fad surfing," and is a defining characteristic of an industry that often eyes structural and technological change with caution. While SAS-centrism has clear and documented benefits, it also has the potential to lead to inertia and an almost reflexive resistance to new technology.

While many of the core components of the submission are fundamentally the same, albeit cast in different formats, the same cannot be said of the environment *surrounding* stat programming. The number and coordination of tasks required for submissions have changed significantly. EDC systems provide source data in more formats. Reviewers Guides and Case Report forms are more complex but can be at least partially automated.

The embedding of SAS in stat programming department and the changes external to it support the maintenance of status quo: a good deal of the change in the submission workflow is due to the

introduction of standards. New technology to support these standards *surrounds*, but is not necessarily *part of*, the stat computing environment.

IMPLICATIONS FOR THE STATISTICAL PROGRAMMER

Let's summarize the major changes in the submissions environment before discussing how our stat programming group (indeed, any stat computing environment throughout the pharma industry) has been affected by them.

- Datasets are coded to clearly defined external (CDISC) standards
- Documentation (the define file) is more detailed and is stored in what is an unfamiliar format (XML) for most programmers
- Datasets and documentation must be syntactically and semantically valid
- TFLs are more numerous and often more complex
- Additional documentation (the Reviewers Guide) is now required for describing complex calculations, explaining unresolved validation issues, and other items that do not fit neatly into the structure defined by define.xml
- As the number of clients increases, so does the variety of source data formats, requests for data points and documentation that exceed CDISC standards, and *ad hoc* programming.

There are likely as many organizational responses to these challenges as there are pharma and CROs: deliverables can be outsourced; CDISC-specific job titles can be created; statisticians and stat programmers can be re-trained. The chosen approach to meeting the added and changed submission requirements can be based on a mix of past experience and intuition. Rho's story over the last 10-15 years demonstrates both.

EXTERNAL CHANGES: CREATING THE BUFFER

Since the early 2000s, Rho has steadily migrated information related to study setup, dataset creation, and components of TFLs to metadata. Without this focus on metadata, these tasks and products would have used an assemblage of Word documents, DOS bat files, and brute-force SAS programming. The combination of metadata interfaces (for entry and editing) and tools (to make the metadata easily accessible) provides a buffer between the programmers and the tables and views, run-time transformations, and other manipulations required for a task. This approach also has another, implicit benefit: as deliverables change, so do the tools; when we move to Version 2 of define.xml, the change from the programmers' perspective is training on the new metadata requirements and how to use the new macro.

The development of the interface and tools was originally a research and development effort, one that could not be pursued by staff working on client projects. It became evident to Rho management that creating a small team dedicated to the metadata/tool development process would be the most effective approach. Relatively free of day-to-day project work, this group could design the database and, with input from programmers and statisticians, design and continually enhance the tools that deploy the metadata into applications throughout the study life cycle.

Other, established departments at Rho also provide data and services that help control the scope of stat programming. Data from an internally developed project tracker is used to populate filters and text in TFLs. The data management department produces datasets that are at or close to standard-compliant. The bottom line here is that dedicated metadata teams, not bound to a particular client or study, can anticipate and react to programmer needs in what has become an increasingly complex and fast-changing regulatory environment.

These departments and the metadata team effectively form a buffer between the regulatory, standards-driven environment and stat programming. With a separate metadata team, project management tools and more standardized raw data, stat programming can do what it does best – create high-quality datasets, TFLs, and documentation. These outputs would be considerably more difficult to produce if the buffer's suite of tools and services did not exist.

INTERNAL CHANGES

So what has changed for the stat programmer? Knowledge of CDISC standards is required, as is the ability to validate datasets and documentation (i.e., define.xml). The greater number and complexity of handoffs among departments (datasets, CRFs) has resulted in an increased emphasis on communication, project management, and related "soft" skills. Non-programming training (*e.g.*, CDISC) has become important, since the programmer is now working in a standards-based environment. The training is multi-sourced: informal, in-house, courses, and professional conferences, among others. Training in internally developed tools is essential to ensure they are used effectively.

Finally, the programming environment itself has changed. We have evolved from SAS Version 8 on desktops to SAS 9.4 in Enterprise Guide, SAS Studio, and the SAS grid computing environment. These platforms present opportunities for workflow change. And common to all of them are the increases in SAS software functionality. A dataset or TFL program created 10-15 years ago will run today, but it is likely that taking advantage of changes to Base SAS (ODS, functions, hash tables, *et al.*) could make the program faster and/or more robust. Building and maintaining SAS skills continues to be a fundamental part of the programmer's agenda.

UPSIDE, DOWNSIDE

We've described above an environment whose organization selectively buffers the programmer from workflow changes created by clients and regulatory agencies.

What is the up side to this approach?

- Continuity of the corporate skill set (a *de facto* aversion to the "fad surfing" that we commonly see in our industry).
- Continued focus on programming. The effective use of tools to access metadata, deeper knowledge of SAS language and coding environment raises the level of the collective skill set.
- Creation of a strong code base (macros, template dataset/TFL programs) for routine tasks improves the rate and quality of deliverables.
- Standards make it easier to move from study to study. Programmers are doing basically the same thing, just with different data. The process – enter/import metadata; use access tools to create deliverables; validate – stays the same.
- The familiar is comfortable.

And the down side?

- Sunk costs inhibit adoption of new tools and technologies. A prime example is Rho's home-grown metadata repository. When, in 2006, we designed the data and tools for define.xml, there were few third-party products available. That isn't the case today, but the switching costs (license and training, among others) are prohibitive.
- The stat programming role as described here might not be appealing to new hires or those who want to gain experience with leading-edge technology. Our processes efficiently create quality deliverables. If viewed objectively, though, they might be seen as idiosyncratic, and not taking full advantage of new technology.
- Programming to a standard can inhibit creativity. This isn't unique to our workplace, but is worth noting. Aside from ad hoc requests from clients, the endpoint for the majority of the programming – compliant data - is the same, and generally uses the same toolset repeatedly.
- The familiar creates stagnation.

WHAT DO OUR PEERS THINK?

We anticipated that our opinions about this topic might be not be consistent with those elsewhere in the industry. Therefore, we reached out to our peers with a brief, open-ended survey. We also collected some information from our workplace's contracts to assess the impact of client requirements on how we create deliverables.

METHODS

Data for the paper was collected using two methodologies: a survey and a review of our contract database. The first method involved sending out a brief survey (**Figure 1**, below) to industry leaders in the area of statistical computing. Leaders were chosen based on their level of participation in PharmaSUG, PhUSE and CDISC. Surveys were sent to 25 industry leaders employed at both pharmaceutical companies and CROs. The survey is not meant to be random or representative of any particular population, but is based on a sample carefully selected by the authors and is limited to the pharmaceutical industry. (Academia was not included.) Figure 1 presents the survey instrument.

Figure 1. Survey Instrument Sent to Industry Leaders

Deliverables

Please describe the deliverables (i.e. analysis datasets, tables, figures…) that statistical programmers were responsible for producing 10 or more years ago.

Please describe the deliverables (i.e. analysis datasets, tables, figures…) that statistical programmers are responsible for producing today.

Programming languages

What programming languages were required to produce the deliverables described in question 1 (10 or more years ago)?

What programming languages are required to produce the deliverables described in question 2 (deliverables produced today)?

Skills sets (not language-specific)

Please describe the skills required for producing the deliverables described in question 1 (10 or more years ago).

Please describe the skills required for producing the deliverables described in question 2 (deliverables produced today).

The second source of data was the contract database at Rho. We looked at contracts since the year 2000 with pharmaceutical companies that contained a statistical computing or biostatistics component. Within each contract, we examined the list of deliverables and whether SAS or any other programming language was required to produce them.

RESULTS

Ten of the 25 industry leaders responded to the survey that we sent out. Six were from large pharmaceutical companies and four were from CROs. While our sample size is small, responses showed a clear consistency.

Table 1 presents the deliverables produced by statistical programmers today compared to greater than 10 years ago. Our respondents report that 10 years ago, programmers primarily were responsible for creating analysis data sets (n=10), tables, figures, and listings (n=10), and to a lesser extent patient profiles (n=5). Only three subjects mentioned the define file (define.pdf) and only two cited producing dataset specifications.

As expected, the greatest change in deliverables from 10 years ago is associated with the introduction of CDISC standards. Participants report that programmers are now responsible for producing ADaM and SDTM datasets (n=10), define.xml (n=7), CRFs annotated for SDTM (n=5), a reviewer's guide (n=4), and specification for SDTM or analysis datasets (n=5).

There is some continuity amidst the change. As was the case 10 years ago, programmers are still tasked with producing tables, figures, and listings (n=10).

Table 1. Deliverables >10 years Ago Compared to Today (N=10)

Deliverable	> 10 Years Ago N	Today N
Create Analysis /ADaM Dataset	10	10
Spec Analysis/ADaM Dataset	2	5
TFLs	10	10
Listings	8	1
Patient Profiles	5	0
define.pdf	3	2
Annotate CRF	0	5
Data Anonymization	0	1
Create SDTM Dataset	0	10
Spec SDTM Dataset	0	5
define.xml	0	7
Reviewers Guide	0	4

Table 2 looks at the programming languages used to produce deliverables today compared to 10 or more years ago. One respondent provided this succinct evaluation: “Sadly, very little has changed in the past ten years”. All 10 participants reported that SAS was the language used to produce almost all deliverables both 10 years ago and today. One respondent stated that some R was used 10 years ago. While four surveys mentioned that some R was used today, two of these four said that R was not used for deliverables and the other two reported that R was only used for a few graphs. These findings from our external survey align with our review of Rho contracts with sponsors since the year 2000. We reviewed the Rho contract database for approximately 200 sponsors and found that *every* contract required the use of SAS to produce the deliverables listed in Table 2 (although one sponsor did inquire about our R capabilities).

Table 2. Programming Languages Used >10 years Ago Compared to Today (N=10)

Language	10 Years Ago N	Today N
SAS	10	10
Some R	1	4*
Extensive use of R	0	0
Other (SPSS, JMP)	1	0

* Respondents stated that R was used for a “few graphs”. Two of the four respondents stated that R was not used for deliverables.

Table 3 summarizes non-programming language skills today compared to over 10 years ago. Not surprisingly, what stands out is the current need for CDISC knowledge, understanding of FDA guidance, spec-writing skills, and the overall need for an understanding of clinical research compared to over 10 years ago. The need for these skills was attributable to the emergence of CDISC as the required standard for study data in a regulatory submission. Three respondents noted the need for better communication skills today. Their comments attribute this to the increase in outsourcing over the past 10 years. Lead programmers are now responsible for interaction with CROs and managing and validating work produced by CROs.

Table 3. Non-language Skill Sets >10 years Ago Compared to Today (N=10)

Skill Set	10 Years Ago N	Today N
Management skills for lead programmers	1	1
Communication	0	3
CDISC knowledge	0	9
Some statistics knowledge	1	1
Understanding of clinical research	1	6
Writing specifications	0	6
FDA Guidance	0	7
Understanding of 1 or more operating Systems	2	1
Data Modeling	0	2
Scripting	1	2

DATA SCIENCE TO THE RESCUE?

A host of recent papers suggest that statistical programmers in the pharmaceutical industry must now also possess data scientist skills. This means being conversant in metadata and data modeling, statistics, advanced computing tools, predictive modeling, and data analytics. In fact, at a recent PhUSE conference there were a good number of papers on data visualization and data exploration that used languages other than SAS, such as R and Python.

However, our brief survey and review of contracts with over 200 sponsors suggest otherwise. Almost all of the work performed by statistical programmers still uses SAS to produce clinical and analysis datasets and TFLs. What *has* changed is the need to be knowledgeable about CDISC data standards and related FDA guidance, as well as the ability to manage work outsourced to CROs.

Why is there this such a disconnect between the literature and our findings? One possible explanation is that the statistical programming role has not actually transformed. Rather, the data scientist is a new and distinct role that has emerged in the past 10 years. Scanning various job descriptions for data scientists, these skills are typically cited:

- Understanding machine learning techniques and algorithms such as k-NN, Naive Bayes, SVM, Decision Forests
- Common data science toolkits such as R, Weka, NumPy, MatLab
- Data visualisation tools such as D3.js, GGplot
- Query languages such as SQL, Hive, Pig Experience with NoSQL databases, such as MongoDB, Cassandra, HBase
- Applied statistics skills such as distributions, statistical testing, regression
- Scripting and programming skills

Clearly, data science is the shiny, brand new aisle of the computer science toy store.

CONCLUSION

Our view is that for the programmer tasked with creation of FDA deliverables, the skills above are at best complementary: *nice to have, but not necessary*. To produce the deliverables documented in our survey, we still need to hire statistical programmers with SAS expertise, and we will continue to have the need to provide them with tools and processes that ensure their productivity. This is likely to be the case until FDA submission requirements significantly evolve. Given the amount of time it has taken to implement CDISC standards, current submission guidelines and the amount of time it has taken industry to respond, we believe that SAS expertise will remain the primary skill for statistical programmers for the foreseeable future.

ACKNOWLEDGEMENTS

We appreciate the input from our colleagues who responded to our survey. And thank you, April Sansom, proofreader extraordinaire. Her red pen got an extra hard workout reviewing our early drafts. The final product reflects her diligence, persistence, and familiarity with the Oxford comma.

CONTACT INFORMATION

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

Frank Dilorio
CodeCrafters, Inc.
FrankDiIorio@gmail.com

Jeff Abolafia
Rho, Inc.
Jeff_Abolafia@RhoWorld.com