SAS Java Library

A proposal and other stories

Introduction

This paper is about the Statistical Analysis System or SAS. It is also a discussion on the problems with the data collection to submission and all the bits in-between, and asks the question if the two are in fact related.

Brief History of SAS

To this day I still find myself correcting Analysis Plans which have SAS listed in the abbreviation section because since version 6, SAS Institute have no longer wished to emphasize the "statistical" heritage of their original Base product. The Institute's ambition was to create "a complete information delivery system which can be utilised at all levels of data preparation, manipulation and analysis". Up until that point, the original SAS was written in assembly, FORTRAN and PL/I, but was overhauled and rewritten, motivated in part by the fall in the number of PL/I programmers in comparison with the easy availability of C.

Fast forward to today, and the core product, the Base SAS system, behaves much as it has done in those intervening 25 years. Procedures and functions have been added, with the SASware ballot an excellent way to deliver improvements to the system driven by user-demand, but to the newbie and to the old hack, the syntax can be contradictory and confusing. In short, I think the time is now for a long overdue overhaul.

SAS Corporate Strategy

We have a glut of revenue-driven modules of various degrees of take-up and success, many of which seem to have a prefix of "Enterprise" in their name. Back in the dawn of the millennium, SAS embraced the idea of the thin-client, thick server basis of the Web by bringing out Web-prefixed version of existing modules (AF and EIS). These were killed off by the idea that SAS wanted to charge for these when a non- proprietorial solution was available through Java. Non-web AF had also undergone a re-write, with its syntax converted to Java-like dot notation – which only served to remind developers that again, a more comprehensive set of widgets could cheaply be obtained (and perhaps more easily built) using a more user-friendly, familiar Integrated Development Environment (IDE) in Java.

This last point defined all that is I believe to be the incorrect corporate strategy of the SAS Institute, but understandably are also the development strategies of any industry which has a working, successful product. Those examples were attempts to patch on changes in the outside world into the existing SAS development environment. And then patch on something else. In fact - anything new that comes along, let's bring out a new module - appears to be the thinking. Sometimes these modules are competing against each other for the same type of work, for instance SAS/IML with the existing STAT and GRAPH modules.

Rivalry

Other rivals have sprung up as challengers to performing the sorts of analysis that SAS does very well. Herein lies the seeds of destruction: SAS is used as the industry standard because it works well, as it integrates with data in many different forms and has a consistent look and function across operating systems. SAS is not alone in trying to do everything – and be all things to all people, but the reality is that its chief, unique selling point is its number-crunching and application of many statistical models.

Devolution

The introduction of ODS 10 years ago helped to improve the presentation of these results, but this has always lagged behind other products produced by other vendors (notably graphics). We even have a situation with the confusion of competing modules promoted by SAS itself. My argument though is that even rivals such as R should also stick to the statistics and not the presentation. Instead, dedicated graphics software should be used to present the data graphically, and textual output should also be presented by the dedicated desk top publishing, browser or word processing tools, rather than trying to integrate the language within SAS itself.

SAS re-invents C++ again

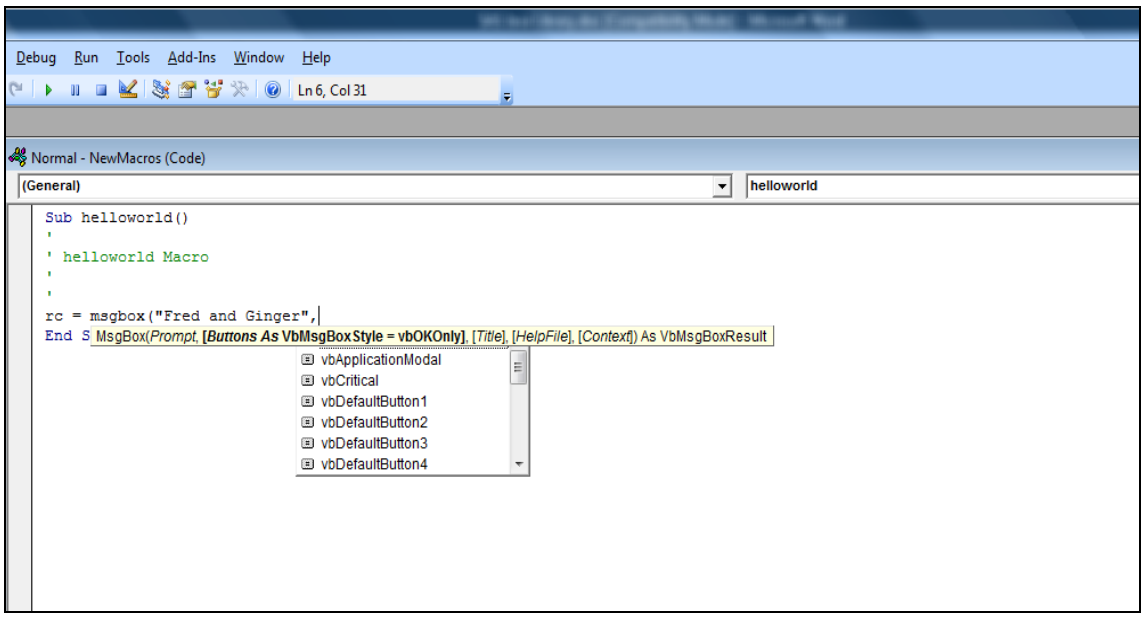
In short, SAS have taken their eye of the ball and it is time to stop. With so many fingers in so many pies, it looks (to me) that the original value of SAS as a number-crunching tool for the statistical modelling of data has been left behind and forgotten. Worse still, SAS is no longer an acronym for "Statistical Analysis System", which everybody got wrong anyway, confusing the "system" for "software" (which was never helped by lots of references in SAS documentation to the "SAS System").

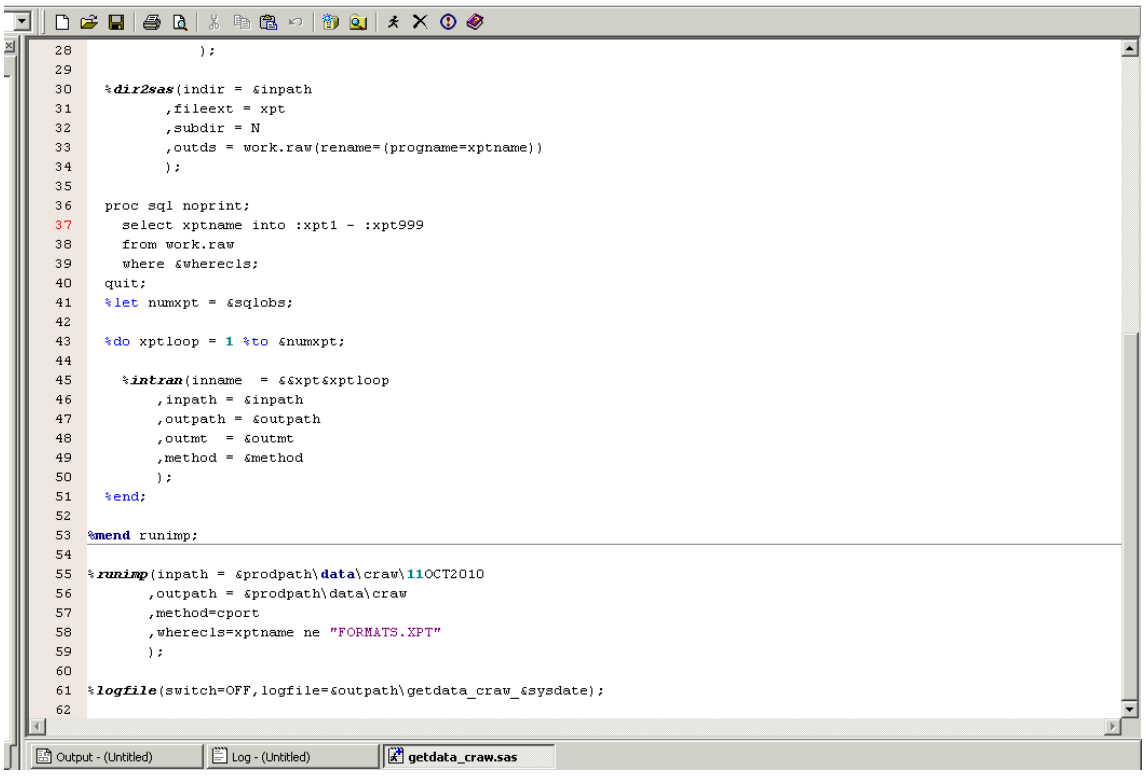
In the Base language, it is now possible to create hash objects by using the same add-in syntax that has also been developed concurrently in PL/I. Just as with the approach with web-enabling SAS using Java, it is analogous to the extension of the C language by clunking on object oriented techniques, rather than having those features built into the underlying design from the outset (like in Java). In particular, C++'s reputation has suffered because of this – and because of its poor garbage collection (the destroying of created objects in a session which otherwise would clog-up memory and subsequently performance).

In order to avoid the jokes that C++ has acquired, I believe that the SAS statistical modelling techniques and functions should be should be serialised as objects that communicate with the data in a development environment outside of SAS itself, rather than trying to clunk objects inside. A SAS Java library in short.

Integrated Development Environment (IDE)

The Enhanced Programming Editor (introduced with version 7) was a step forward in helping developers produce code with fewer errors, in response to the market in text editors which did a better job of indenting and manipulating and colour coding SAS statements according to their context. What developers like is the predictive capability of an IDE to anticipate and prompt for the next part of a program statement – especially if language functions are composed of many parameters. Visual Basic, Java, C++, in fact most computer languages have this facility. SAS doesn't, and can't – at least in its current form. This is principally because the SAS language itself is a hybrid of many differing style and contradictory syntax. My favourite is the difference between TRANSLATE(source, to, from) and TRANWRD(source, from, to). Ultimately then the colour coding of the uncompiled code may not be 100% accurate, especially when SAS code within macro code is concerned. The solution is surely to turn these SAS functions into messages triggering events in data objects. This would allow users to use an already familiar IDE – where the syntax will be checked and corrected as development proceeds.





Within SAS, the Enhanced Editor falsely recognises the word "data" as the start of a data step command – and colours it blue; and the "11" is coloured cyan as this is assumed to be a format length. Meanwhile, inside the macro definition section of the program, none of the PROC SQL commands have been colour coded.

Within the VB editor, not only are the statements colour coded in context, predictive syntax tool text pops up as the statements are being written.

Why things must change

For this brave new world to work, it requires other changes to happen in industry, and as so much of today's world depends on existing infrastructures and procedures to change. The justification for this statement requires me to offer an argument for the changing of almost everything in the drug development model.

Regulation

The FDA launched the Critical Path Initiative in 2004 in an attempt to arrest the decline in new drug applications (and as a direct result of this, approved drug patents). The graphs above are grim reading – in spite of an increase year-on-year in total R&D spending, the number of new applications is on a long-term downward trend. Meanwhile the market for generics (those drugs for whom the formula loses its patent – and revenue for the pharmaceutical company who developed them) has increased.

Some progress has been made – the focus of CPI is on identifying biomarkers and focussing on neglected areas such as paediatric medicine; however 2009 remained flat.

Boom and bust

As shown below, for several years now the pharmaceutical industry has been in decline. True, there are a few blockbusters which appear by accident during the development process, and there has been the mergers and acquisitions flurry which dominated in the 90s and 00s which stimulated profitability and share prices. None of this is sustainable, and does not replace the real growth of a company through long-term development of bringing product to market.

Figure 1: 10-Year Trends in Biomedical Research Spending

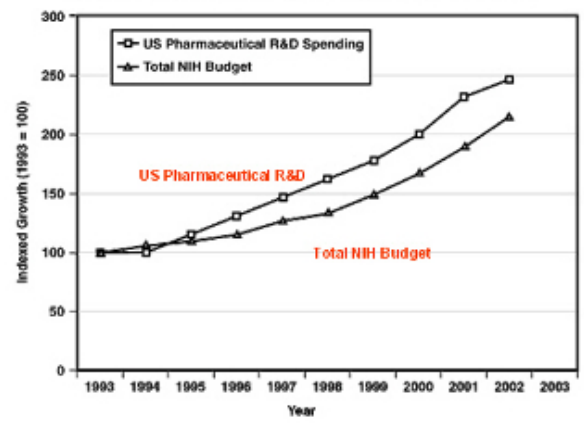
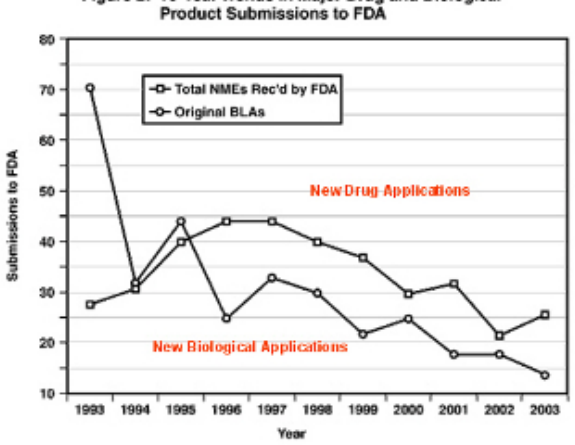


Figure 2: 10-Year Trends in Major Drug and Biological Product Submissions to FDA



(source: http://www.c-path.org/CPI.cfm)

Agency neglect

Meanwhile the FDA itself has suffered years of neglect by US Governments - keen to be seen to criticise for high profile safety scandals for drugs approved for market - but not supported by funding to assist in their review. Not only have jobs been cut (from 1994 to 2007 personnel fell from 9,167 to 7,856) spending has not even kept pace with inflation (it has been estimated at only 2/3 of the amount required – "*Who Strangled the FDA",American Prospect Dec 12 2007).* The number of AEs have increased in the 10 years to 2006 by 146 percent without the corresponding increase in staff to perform the review. In spite of this, FDA review times have fallen from around 40 months in the 1980s to 19 months in 2001 (Rehnquist review, Office of the Inspector General 2003).

Standardisation

The need for standardisation in programming in the pharmaceutical industry had been long overdue. It comes as no surprise then that the origins of PhUSE itself came from this environment. Ironically, when the need to adopt standards was greatest (fuelled by the many Mergers and Acquisitions in the industry), the opportunity has often been passed over in favour of office politics.

Arguably the role of CROs has also been a driver of standards, as the need to adopt client-neutral systems and programming is a commercial reality for these companies.

CDISC

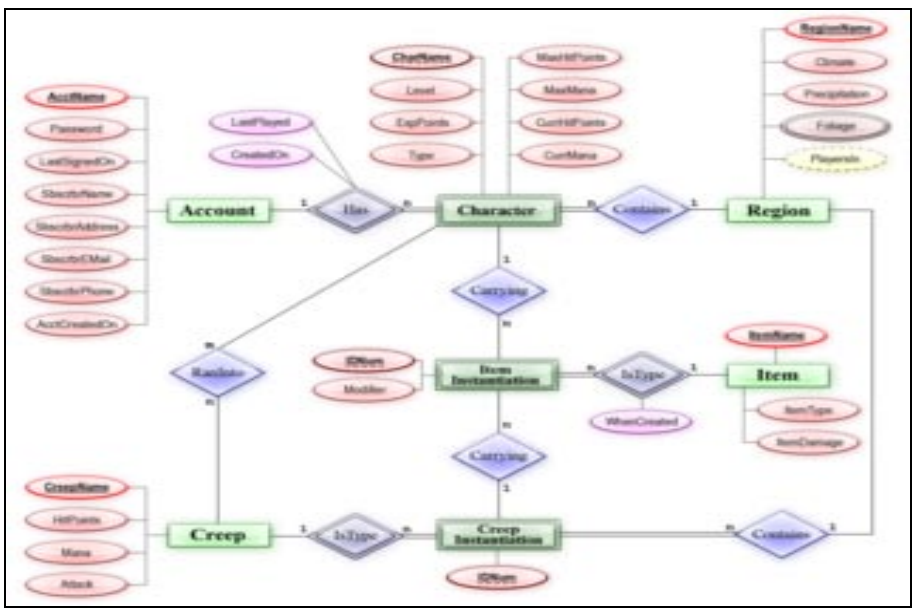
The FDA – perhaps frustrated by their dwindling resources – and need to monitor more effectively the long-term safety of all products – have led the way in creating their integrated data warehouse JANUS. For the meta analysis to succeed, the agency and pharmaceutical companies have co-operated in creating CDISC.

CDISC is not a "registered" standard as such (yet), but it has gained substantial recognition within the industry and the FDA. It ought to stand a good chance of success as it is a collaborative project – it is only as successful as its users. However this strength is also its weakness, as it allows considerable flexibility and (mis)interpretation. The model is as close to a standard as you can get, and at least a serious attempt to explore how data is collected, databased, analysed, reported and submitted has begun. However, there are still gaps in the representation of data in various therapeutic areas and especially in regards to efficacy. In the absence of these, companies have developed their own versions of sponsor-defined domains – which currently battle for dominance. Arguably this battle erodes the meta-analysis project itself, though the primary driver (as far as the FDA are concerned) is arguably patient safety.

CDISC and HL/7 and mapping

The language of HL/7 is littered with references to the sending of messages. This is straightforward language for the representation of the world by object classes. CDISC is a curious hybrid of the two schools, which misleadingly refers to data "domain models", but also to the "observation classes", which mixes the relational theory of the "domain of discourse" with the object oriented nomenclature. Moving forward, it is the aim to move CDISC standards down the HL/7 route, and this is the aim of the creation of the BRIDGE project, and is part of the FDA's strategic plan.

The aims of this are to capture the lost links across data, which is currently only viewed as a series of flat database files. My view is that this will only happen if the database model is thrown away and an Object Oriented model is used. At the very least, a re-evaluation of the analysis of the pharmaceutical world using an Entity-Relationship diagram is required, as what is required is a new database design that builds the links as part of its schema. Analysis of the links between data sets cannot take place if the links themselves have not been created (or thought of) in the data base model.



Source: http://en.wikipedia.org/wiki/Entity-relationship_model

Round pegs, square holes

With between $1 to $3M a day in sales once approved, there is tremendous pressure on companies to get their drug to market. Timelines get squeezed at each stage if things go wrong to prevent delay. (The £1M/day is a much-used quote, but there is no equivalent for the price of failure by rushing through something with mistakes). Of course, there is no room to re-invent the wheel each time, but the consequences (at best) are that database designs and programs get re-used, as do specifications and very specific interpretations of standards. My main issue is good old fashioned data analysis seems to be neglected in all this, with so many examples of trying to adapt something that worked in one study to another project – with the danger that the subtle differences of one project dataset or variable is lost in the process.

End-to-end: the myth

If blind re-use is also coupled with functional areas working in isolation from each other, with problems being 'thrown over the wall', (in other words, errors left unchecked to be resolved by the next stage in the process), then you get quite a toxic mix. In fact this happens in most organisations, with protocol development commencing at one end leading on to the data capture activities (data base design and CRFs, development of the RDBMS) and independently from the Statistical Analysis and the final reporting activities at the other. The Protocol could also be amended once all of these activities at both ends have already begun.

Basically, the end-to-end process does not work. Within Biostatistics, the final deliverable is reversed engineered from the style of the outputs to be delivered (namely Tables and Listings and Submission-ready data sets). At the other end of the process, the Case Report Forms (paper or electronic) drive the structure of the transactional RDBMS data. The two meet in the middle and clash.

The statistician and programmer must therefore understand the issues surrounding the data and its context, looking to ensure that nothing has been missed at either end.

Previous focus on "end-to-end" process has been driven from data capture, normalisation and standardisation. The irony is that the Protocol is initially driven by the statistical model of what the study is trying to achieve – How can we prove the drug works and is safe? How many subjects do we need to recruit? This is may not a static proof as scientific innovation and theories evolve all the time, and so the plan may be refined in accordance to data monitoring and review activities (if lucky).

The argument must surely be that there should be a greater say in front-end design

activities by the statistician and programmer – and be prepared to iterate and change. That this may not always happen (or resistance to change the design) comes down to the failure of the use of RDBMS as a means to capture data.

RDBMS versus OODBMS

For analysis and reporting, this is most usefully performed in SAS which can successfully number crunch and analyse data, provided that it is in a structured format. This is usually in the form of data tables that have been organised in a relational database system and developed using SQL – this is an easy fit for SAS. It is also an easy fit for Data Management within the company as RDBMSs are sold by many vendors who tailor their products to suit the needs of the pharmaceutical industry. The problem is that the tailor is not a bespoke tailor, but an "off-the -peg" one. The database management system may have set solutions for different types of data that can be copied rather than re-modelled. In Statistics and Programming, there is also similar pressure on squeezing timelines by copying one solution from protocol to protocol (and sometimes between differing therapeutic areas), rather than starting again.

The consequences of having to change or adapt an existing design of database are awful in an RDBMS solution. Having to rebuild the model from scratch once the data entry stage has commenced can lead to a catastrophic loss of data, and considerable down-time whilst the re-build and validation takes place. An OODBMS could cope more easily with such change by using the concept of inheritance, where a new data object can be built that keeps all the attributes of the older object without the loss of that data.

Niche, networking or nepotism?

Nowadays fewer computer science graduates have procedural-based languages amongst their weaponry, as Pascal, C and even C++ have been replaced with Java, Smalltalk and other object-oriented general-purpose programming languages being taught at university instead. SAS itself is being taught less, with many universities preferring to use R, an open-source version of the proprietorial S-PLUS (both S-PLUS and R are based on S, a rival statistical computer language which uses objects and classes).

The consequence for the industry is that a failure to attract new SAS developers and programmers means an ever-diminishing pool of talent, which in turn drives prices up. "It's a small industry" is an oft-heard phrase, but the reality of this covers-up the fact that it is not unusual to bump into "serial contractors" - the same people working for different companies over the course of a short time. Recruiting an ex-colleague or friend is a good guarantee of quality, as you will have a working knowledge of their skills and personality, but this could also be viewed as favouritism or nepotistic to an outsider.

In Summary

The new world is as follows:

- Data to modelled and represented using OODBMS. Links and messages between objects in that world will function in that world rather than being mapped from existing RDBMS technologies. Mapping to fit standards is the root cause of data loss.
- SAS functions, statistical modelling and mining to be available as an object library – to be applied to the data objects.
- Data objects to be modelled based on HL/7. CDISC role will be to create the model around which the industry will flock.
- Java IDE to be used to manipulate this world in data. Existing tools that specialise in displaying text (word processors, desk top publishing, web browsers) should be used to display this textual data (not SAS modules). Existing tools that specialise in displaying graphics should also be used to display these graphics. And not SAS.

One year on…

A year ago in Basel, Dr Diane Jorkasky delivered the keynote speech for PhUSE 2009. She spelt out the need for the industry to change by adopting new technology, standardisation and change. Whether this is possible remains to be seen – there is a lot of wealth created and earned by continuing to use the existing technologies – RDBMS vendors, recruitment agencies, CROs, SAS, even you and me. In short, this poster is a virtual death sentence for the way in which I currently work today (and I am a little uncomfortable with that!)

Just as 25 years ago, PL/I and FORTRAN programmers were replaced with C programmers within SAS and S, the opportunity is never better to have a re-think in the pharmaceutical industry regarding programming, data acquisition and reporting, and of how SAS fits into that model.

It might be best to think of the evolutionary process of technology and machines. When the first twin-tub washing machines were invented, they motorised the existing human manual washing processes. This was just the first stage in an evolutionary process. Real change comes from changing those processes entirely – the pharmaceutical industry equivalent of the automatic washing machine.

Further reading

Bertino, Catania and Zarri: Intelligent Database Systems; Addison-Wesley ISBN 0-201-87736-8

Muller: Database Design for Smarties; Morgan Kaufmann Publishers Inc ISBN 1-55860-515-0

Burleson: Inside the Database Object Model; CRC Press ISBN 0849318076

Contact Information

Comments and questions are valued and encouraged. I can be contacted at

Michael Auld, Michael Auld Consulting

Email: sas@w3mac.com