Thirty Years with Stata: A Retrospective

Edited by
ENRIQUE PINZON
StataCorp

A Stata Press Publication
StataCorp LP
College Station, Texas

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845
Typeset in $\LaTeX\,2_\varepsilon$
Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN-10: 1-59718-172-2
ISBN-13: 978-1-59718-172-3

Library of Congress Control Number: 2015940894

Contents

(Pages omitted)

Preface

Thirty years ago, Stata was created with a vision of a future in which computational resources would make data management and statistical analysis readily available and understandable to someone with a personal computer. The research landscape today validates this vision. We have increasing amounts of data and computational power and are faced with the necessity for rigorous statistical analysis and careful handling of our information.

Yet the increasing amount of information also poses daunting challenges. We have to disentangle the part of our data that communicates meaningful relationships (signal) from the part that is uninformative (noise). Distinguishing signal from noise was the task of researchers thirty years ago, and it is still today. The fact that we have more data and computational prowess does not mean that we have less noise lurking. The tools of careful data management, programming, and statistical analysis are fundamental to maximize the potential of our resources and to minimize the noise.

In this volume, we gather 14 essays and an interview that answer how Stata has helped researchers in the process of distinguishing signal from noise. In the first part, we begin with a new essay based on a speech given by Bill Gould during Stata's 2014 holiday celebration, which we follow by revisiting two contributions that were written to commemorate Stata's 20th anniversary. These three pieces provide a perspective from inside Stata. The first, the speech, discusses the decisions made regarding Stata's software architecture. The second is an interview of Bill Gould by Joe Newton that reveals the guiding principles behind Stata. The third, written by Sean Becketti, gives us insight into the culture and challenges of Stata in its developing stages.

The second part of the book represents points of view from the outside. Researchers from different disciplines answer how Stata has helped advance research in their fields and how their fields have evolved in the past three decades. Some of the contributions look at the discipline as a whole, while others speak about very specific experiences with Stata. The contributions in this part come from the disciplines of behavioral science, business, economics, epidemiology, time series, political science, public health, public policy, veterinary epidemiology, and statistics. Also in this part, Nick Cox writes about the history of Stata and devotes part of his essay to the conception and evolution of the Stata User Group meetings.

Having a vision from inside and a vision from outside is a fitting way to celebrate Stata's 30th anniversary. The vision from inside reminds us of the ideas that made Stata popular and the principles that guide Stata to this day. Yet, it is the researchers, their interests, their concerns, their active participation, and their interaction with Stata that help the software evolve. The relationship between Stata users and StataCorp is the fundamental reason that we are celebrating this anniversary.

To all that have been inside and to all those outside, this book is for you.

Acknowledgments

The authors in this volume contributed to the final product not only with their chapters but also with their suggestions and input during the entire process.

Shelbi Seiner and Stephanie White did a wonderful job, respectively, with the editing and the formatting of the book. The cover design is the work of Annette Fett.

To all of you, thanks.

Caveat Emptor

All the contributions in this volume were written before the release of Stata 14.

(Pages omitted)

1 Initial thoughts

William Gould

It is 2015 and Stata is celebrating its 30th anniversary. Stata 1.0 was released in January of 1985. That is a long time ago. That Stata still survives is remarkable. That Stata is not a dinosaur is even more remarkable.

Stata was born in 1985. Some of you were not yet born. Others may not remember 1985 as clearly as they would wish. So let me set the stage.

Ronald Reagan was president. Mikhail Gorbachev became head of the Soviet Union. Ronald Reagan would not say "Tear down this wall" for another two and one-half years (Stata 1.5), and the wall would not fall for another two and one-half years after that (Stata 2.05). NASA discovered a hole in the ozone over the Antarctic. Boom boxes were popular. Among the top movies of 1985 was *Back to the Future*.

In 1985, the expensive desktop computers had five-and-a-quarter inch floppy drives. Cheaper ones had cassette tape. Memory was measured in kilobytes, and in only two digits at that.

In 1985, the software available for the desktops was crude. PC-DOS 3.1 was released. So was Windows 1.0, but it was more a forward-looking experimental demonstration system than anything else. It was not much used, and Windows 3.1 was still seven years away.

The first Mac was released in 1984, the year before the release of Stata 1.0. Work on Stata 1.0 began in 1984.

Stata was written in C, and that was an odd choice. The popular mainframe computer languages were PL/I, FORTRAN, COBOL, and various assemblers. The popular small-computer languages were assembler, BASIC, and PASCAL. All the buzz was about PASCAL. In 1984, Apple released PASCAL and Borland released TurboPASCAL.

1984 was the year of PASCAL and Stata was written in the wrong language, everybody told me. Real software, of course, was implemented in FORTRAN, but FORTRAN was not available for small computers, which just went to show that the small computers were not real computers. "Little computers for little minds" was a popular saying among real programmers.

Some real programmers worked on minicomputers, but even that was not fully respectable. A lot of that work involved something called ARPANET, which many believed to be going nowhere, and in 1985, that view was proven correct when ARPANET was transferred to the academic backwater of NSFNET. Later, NSFNET would develop into a major part of the Internet backbone.

Despite all of this, interest in small computers was growing, and the dead-enders from ARPANET escaped to Santa Barbara, Palo Alto, and Berkeley. And it was exciting. Nonexperts were coming out of the woodwork to write software. Stata was not the first statistical system for microcomputers, nor the second.

Stata did not have bright prospects, and it would not have had bright prospects even if it had been written in the right language. Because Stata survives, some might conclude that I am brilliant. Yet I did not predict the powerful computers that we have today, the demise of the mainframe, or the rise of the Internet. I merely thought that PCs were powerful enough to perform data analysis on smaller datasets and that they could do it cheaper and better.

I do not know what my partner, Finis Welch, was thinking when he trusted me enough to run with it. At that time, I did think that C was the language of the future, and I had strong opinions about Stata's design, opinions that would be reconstructed and refined with the collaboration of Sean Becketti.

I cannot take all the credit for Stata, but I am proud of two of my contributions: I was right about C, it was the language of the future. And I was right about Stata's design, which has proven flexible enough to accommodate all the other changes I did not predict.

2 A conversation with William Gould

H. Joseph Newton
Texas A&M University
jnewton@stat.tamu.edu

2.1 Beginnings of Stata

Newton: How did the first version of Stata come about, and what sorts of things could it do?

Gould: The first version of Stata was a regression package and really nothing more than that. It did a little bit in the way of calculations, and it did some summary statistics, but it was all built around a regression engine. It was written over a one-year period by me initially and by Sean Becketti, who helped me later. I wrote the C code; Sean Becketti helped me a lot with the design. I would say that half of the design is mine and half the design is Sean's in terms of what the user actually saw. A number of things became available just at that time when we started this project, and it was those things that actually caused the project to start. The first C compiler was available for the PC.

Newton: When was this?

Gould: 1984. This is our 20-year anniversary. Our official release date was 1985. The actual release date was December 1984. There was the *American Economic Association* meeting in Dallas. That was where we first announced Stata and started selling it. The meeting is usually around Christmas or New Year. I started about a year before that, so that puts the beginning date at about January of 1984. The Lattice C compiler had just become available for the PC, but no one cared because nobody was interested in C—everyone talked about Pascal as *the* language. I had learned C earlier, and I was very interested in C. No part of Stata was developed under Unix, but I was subtly familiar with the stuff, and so once those development tools became available, that really made it possible for me to write something like Stata.

Newton: When you started, did you have syntax, grammar, or anything like that in mind?

Gould: Yes, right from the beginning. For two reasons: The major reason—What was the term that [Brian W.] Kernighan used when describing the creation of Unix?—"Salvation through scarcity". Stata benefited exactly from that. These were really small

(Pages omitted)

4 Then and now

Sean Becketti

4.1 Introduction

I have used Stata for a long time. A very long time. Since before it was released. Since before it was called Stata.

As I described in an earlier article (Becketti 2015), I had the good fortune to work with Bill Gould as he was developing what would become Stata. My initial job was to test—that is, to break—early versions of the code. As the program matured and breaking it became more difficult, I spent more time talking with Bill about what Stata should do and how it should do it. I have designed a few pieces of my own software over the years, but none of that was as much fun as the discussions Bill and I had in those early days about Stata.

In the first couple of years of Stata's life, I remained involved in the discussions about its evolution. Over time, though, my professional responsibilities led me in different directions. I do not remember exactly when the balance tipped. I remember working with Bill on the first version of the Stata `graph` command (which I still love) and on the `anova` command. I also recall assisting with improving the matrix inversion algorithm that underlies the `regress` command. That project led to Bill's first prototype of a matrix language for Stata. Anyway, at some point around then, I stopped working on Stata. A bit later, Bill asked me to edit the *Stata Technical Bulletin* for a couple of years. However, my professional obligations eventually took over all the time I had, and I cut the cord. I did return to the fold temporarily. Thanks to the global financial crisis, I had some free time in 2009, and Bill persuaded me to write *Introduction to Time Series Using Stata* (Becketti 2013) for the Stata Press. As it happened, I landed a new job fairly quickly, and I ended up trying Bill's patience by taking an unconscionably long time to finish what is a fairly straightforward book.

All of this is to establish that I have been involved with Stata for its entire life. Stata has changed quite a bit from the pre-Stata I first tested in 1984 to the software system it is today. The ways I use Stata have changed as well.[1] I will spend the next few pages highlighting some of the significant milestones along the way—for Stata, for time series, and for modeling in financial services (my field of endeavor).

1. Surprisingly, I have remained exactly the same as I was in 1984.

4.2 Stata, then and now

Stata launched just about the time everyone else in the world decided to introduce a statistics program for personal computers. PCs were weak; Bill started developing Stata before hard disks were offered on PCs. They had just become powerful enough to consider moving some real work from the mainframe (which was expensive to use) to the PC (which was expensive to buy, but free to use thereafter).

Many colleagues and competitors tried to convince Bill that he was introducing his "nice little regression program" (their description) too late—the market was already crowded with programs that had many more features than Stata 1.0. In those days, potential customers were fixated on comparative lists of available features (even when they had no plans to use many of these features). Did Stata include analysis of variance? Not yet. Did it include time-series analysis? Not yet. Did it include graphics? Too few to tout. To make matters worse, many customers were waiting for the two behemoths of statistical software—SAS and SPSS—to offer PC versions and crush all the upstarts. What these observers missed were Stata's considerable advantages.

Three things set Stata 1.0 ahead of the competition, more than making up for Stata's initial paucity of statistical features.

Stata's first advantage is its "modeless" approach. In many competing programs, the user has to navigate from module to module to get anything done. The user has to run a data-preparation step, then run another module to estimate a model, then save the estimation results, and then, finally, generate predictions and diagnostics. Each step requires exiting one module and starting another, and results that suggest alternative models require the user to start the process over from the beginning.

In contrast, all functions of Stata are available at all times. To do something in Stata, you—as the Nike slogan goes—Just Do It. As a result, Stata users typically do a better, more thorough job of exploring all aspects of their data than they would if they were using equally powerful, but more cumbersome, tools.

The second advantage, closely related to the "modeless" approach, is the data rectangle (my term, not Stata's). Other programs operate on files stored on disks. In data-preparation steps, other programs retrieve one observation at a time from an input file, manipulate it, and then store the results in an output file—again one observation at a time. Then the output file is passed to one of the statistical modules. This file-based approach dictates the module-to-module processing described above.

As you know, Stata stores its data in memory in a rectangle.[2] Each column contains all the observations on one variable. Each row contains one observation of every variable. For spreadsheet users, this is a natural, intuitive structure. In addition, it is easy to use *varlists* and the `if` and `in` clauses in Stata to restrict the operation of a command to any desired subrectangle without having to actually subset the data. It is difficult to overstate the usefulness of this approach.

2. At least, you can visualize it as a rectangle.

Initially, the data rectangle was regarded as a disadvantage of Stata relative to its competitors. After all, only a limited amount of data can fit in the computer's memory, and in the early days, that amount was substantially less than it is today. With the expansion of computer capacity and simultaneous reduction in cost, there are relatively few problems that do not fit in memory today. The data rectangle is now a clear advantage, and file-based approaches remain clumsy in comparison.

The third advantage is Stata's rigid syntax. Things have loosened up slightly over time, but initially there was only one way to say anything in Stata,[3]

```
by varlist: command varlist = exp if exp in range [weight], options
```

where everything except the command name is potentially optional. This may not seem like an advantage or a disadvantage, or even anything worth mentioning. But, while this syntax might seem confining at first, it actually simplifies learning Stata. If there is only one way of saying things, then you have to learn only one way of saying things. As a result, it is usually easy for users to guess how to use new commands. Even guessing what a Stata command might be called (is there a common English verb that describes this action? `regress`? `summarize`? `describe`?) works surprisingly often. This simplicity drastically shortens the learning curve for Stata.[4]

As I mentioned before, early users were "feature freaks". The important things—modeless operations, data rectangle, consistent syntax—are not immediately appealing, and their benefits are obvious only after one uses Stata for a while. As a result, sales grew slowly at first. Eventually, though, loyal users started spreading the word. Moreover, Stata users started inventing Stata commands, expanding the range of things Stata could do conveniently.

This brings me to the next—and, to my mind, most important—milestone in Stata's progress: the ability to program Stata in Stata. It was always possible to create a Stata script to package a long or complex or just plain tedious sequence of operations.[5] A Stata program—a script that appeared to the user to be just another Stata command—took longer to appear. But once it arrived, the focus of Stata development changed.

Being a lazy sort, I frequently would ask Bill to add a feature or command to Stata that would make my life easier. Bill would counter that a resourceful person (clearly not me) could produce the requested result with a user-written Stata program, obviating the need to clutter Stata with a new feature. Then things got competitive. I would try to prove that it was impossible to fulfill my request with existing Stata features.

3. I am fudging the facts ever-so-slightly. For example, the `set` command.
4. This syntax did not always make things easy for us when Stata was being invented. We struggled first to pick straightforward, descriptive names for the commands (as you can tell from the current manuals, this became harder over time), then we struggled to figure out how to translate a natural English command (perform this type of estimation in the following way with certain adjustments that are very important to us) into the Stata syntax. More than once, we stared at the keyboard together for 10 or 15 minutes to see if there was an unused special character that might simplify things.
5. I do not remember exactly when the `do` command was introduced, but I do not believe it was available in Stata 1.0. Nonetheless, there were work-arounds that offered similar functionality. Anyway, the `do` command was introduced very early.

Bill would try to show how, in fact, my task could be done with Stata. On those rare occasions where it turned out I was correct, Bill would add another programming construct (looping, macros, etc.) to the Stata program toolkit, and that would solve the problem.

As we tackled more and more complex tasks, the Stata programs became more complex, and a serious problem arose. Stata programs were often difficult to write—not because the calculations were complicated, but because it was tedious to interpret the user's command line. And, more troubling still, Stata programs violated the syntax introduced in Stata 1.0. Abbreviation of variable and option names typically could not be supported. Indeed, options usually could not be supported, at least in the form used by built-in official Stata commands. Stata programs were vulnerable to simple user typos in ways that built-in commands were not.

And then a miracle arrived—the `syntax` command. Now, with almost no effort on the part of the programmer, a Stata program could be virtually indistinguishable from a built-in command. Typos triggered exactly the same error handling and error messages produced by a built-in command. Stata programs became much shorter and easier to read. I believe the `syntax` command was the catalyst for the explosion of Stata programs—and, hence, the explosion of Stata features—that followed. Other commands (`set trace on/off`, `pause`,[6] extended macro commands and the like) have helped, but `syntax` is the secret ingredient. The Stata programming features available today make Stata one of the most efficient platforms for complex statistical program development.

At this point, you are probably thinking, "What is wrong with this guy? The `syntax` command is so important? Come on. What about Stata's incredible range of statistical capabilities? What about Mata? What about Stata graphics and the Graph Editor? Are not those the signature features of Stata?" Of course all of those things and more that I did not mention are important, and they are typically the reason many users start using Stata. Many other programs offer a broad range of statistical features and graphics and even matrix languages, but they are not Stata. I have used many other programs, and many of them are very good, but none of them quite capture Stata's balance between already-built-in features and the convenience of extending and reusing these built-in tools in new and creative ways.

I would be remiss, however, if I did not point out the Stata team's absolute commitment to statistical integrity and accuracy. Those of us who survived the early days of statistical software remember some of the serious shortcomings of many of the early commercial programs. Working statisticians developed Stata. They insisted on a high-quality, no-compromises program to use in their own research, and they have firm opinions about the right way to do statistics. Of course, at times, the market demanded that Stata include a statistical technique the Stata team would prefer on professional

6. I believe my constant complaining played a small part in Bill's invention of the `pause` command. One could argue that my primary contribution to Stata has been a steady litany of (constructive) complaints. If StataCorp ever creates a position for a "complainer-in-chief", I think I have got a good chance of getting the job.

grounds to omit.[7] Even in those instances, the developers insisted on implementing the highest-possible-quality version of the technique.

Before I turn to the evolution of time-series analysis over the last 30 years, I want to say a few words about the implementation of time series in Stata. It took a long time to get built-in time-series capabilities in Stata. For many years, I relied on my own home-brew Stata programs to handle my time-series analysis needs. When Stata finally introduced its time-series suite, I was reticent to leave my familiar ado-files behind. My programs retained the modeless approach of Stata. In contrast, the Stata time-series tools required me first to define the frequency of the data and the variable that contained the date and time information. Many of the time-series commands used the "two-part" syntax that Stata has adopted for features that are too complicated to squeeze into the Stata 1.0 syntax. For example, to estimate and apply an exponentially-weighted moving-average smoother—a bread-and-butter operation in the world of time series—you cannot just type, say, `ewma`. You have to type `tssmooth exponential`, and the `tssmooth` command includes a family of related smoothers that are specified by the second word of the command. This syntax works, but it lacks the simplicity and elegance of equally sophisticated cross-section commands (for example, `regress`).

I held out for quite a while, struggling along with my more-limited ado-files. Over time, though, I found that my staff readily adopted the Stata time-series suite, and soon they were able to do easily what took me a lot of work. I started using one or two of the time-series commands that I just could not live without, but I did not transition completely until I wrote *Introduction to Time Series Using Stata* (Becketti 2013). That experience forced me to catch up with the rest of Stata users, and I am glad I made the change. I believe the Stata design makes time-series analysis about as easy as it can be. The complexity of the time-series commands reflects the complexity of the statistical techniques.

4.3 Time-series analysis, then and now

There is an element of irony in talking about changes over time in time-series analysis.

In the last 30 years, there have been a remarkable number of major breakthroughs in time-series analysis.[8] When I began my professional career, there was a handful of techniques for pure time-series forecasting; a fair amount of attention to seasonal adjustment; and some careful thought, but few tools, devoted to analyzing business cycles. Autocorrelation in the error term was regarded as a particularly annoying variety of heteroskedasticity that required "correction".

Box and Jenkins reinvigorated research in time-series analysis by providing a unified, tractable, and intellectually satisfying perspective on dynamic analysis. Just as

7. To avoid giving offense, I will not name any examples.
8. Warning: In this section, I am going to ignore a lot of interesting aspects of time-series analysis. I come to the field as an applied economist with a strong preference for time-domain models. Extensive work has taken place on frequency-domain models and in related fields that use time-series techniques. I am limited by my range of experience.

important, in their 1970 book, Box and Jenkins published usable algorithms—recipes for time-series software that put these techniques within reach of most statisticians.[9]

Then the pace of progress accelerated. Hendry and Mizon (1978, 1980) inverted the view of autocorrelation as a problem to be corrected and focused attention on the substantive information in a dynamic specification. Engle's (1982, 1987) introduction of autoregressive conditional heteroskedasticity gave researchers a way to model the impact of transient disruptions to dynamic processes. Granger's (1969, 1987) concept of cointegration provided a better way of thinking about long-run relationships. Sims's (1980) analysis of vector autoregressions challenged traditional thinking about structural econometric models. At least three Nobel prizes recognize the importance of this work and several more reward work that draws upon these contributions.

In some ways, though, the central topics in time-series analysis have hardly changed.

- Dynamic models of stationary variables provide a useful but dim flashlight into a dark future. They improve our ability to gauge the impact of transitory disturbances but, by construction, their predictive power decays rapidly.
- Small differences in a trend make a big difference in long-run outcomes. For instance, a difference of half a percentage point in the annual rate of gross domestic product growth generates a gap of ten percentage points in the cumulative growth over 20 years. Unfortunately, it is difficult to identify trends precisely or to distinguish them from unit roots.
- An irreducible tension remains in time-series analysis between pure forecasting models such as vector autoregressions and structural models. Sims offered vector autoregressions as a way to break free of the often-implausible simplifying assumptions in structural econometric models. But, without some structural assumptions, policy analysis—the ability to predict the outcome of counterfactual interventions—is impossible.

The statistical advances of the last 30 years have improved our understanding of these challenges, but the challenges remain.

4.4 Rocket science, then and now

For the last couple of decades, I have worked in and led modeling and analytics teams in financial services firms. I was lured into this line of work by newspaper and magazine stories about the glamorous lives of the Wall Street "rocket scientists"—mathematicians and statisticians who unlocked the secrets of unlimited wealth.[10] While I am not averse to unlimited wealth (I am still looking for it), I was also attracted to the opportunity to use my econometrics skills to do more than study the real world. I wanted to change the world.

9. In those days, most researchers had to write their own programs to use new techniques.
10. There may have been a touch of hyperbole in these accounts.