EDA and the Internet

Leland Wilkinson [*]

February 7, 2001

Abstract

John Tukey's book, *Exploratory Data Analysis*, introduced a new way of analyzing data. Although written for those using pencils, its deepest effects have been on the computing community. Tukey has influenced not only how we write programs, use programs, and write about programs, he has also shaped the way we think about computing. The Internet offers new possibilities for extending Tukey's work, but at the same time presents difficult challenges.

1 Introduction

EDA, like FDA, or WWW, or JWT, is one of those acronyms that some people imagine themselves to understand simply because they know the words behind the letters. Acronym means a name viewed from on high. Unfortunately, EDA is not something that one can understand from on high. The details matter.

I do not fully understand EDA, despite many years of experience programming parts of John Tukey's famous orange book with that title. But I do know something. EDA, like IRS and INS, is an abused acronym. I say "abused" because it is used by many to express feelings rather than ideas. To stop discussion rather than to start it. For example, every statistics package I am acquainted with claims to do EDA, even though there is only one (Data Desk) of which I would say, "If you liked the book, you'll love the movie." People want to believe they are doing EDA even when they are, in spirit and practice, doing anything but EDA. Many people explore. Many analyze data. Few do EDA.

Why few? Perhaps because for EDA, there is more thinking than doing. More than a set of analytic methods, John Tukey has given us a way of thinking about inquiry - as a process of successive surprise and refinement. Tukey's scientist approaches data with many questions and few answers. After exploring, she has some answers and more questions. It is a difficult enterprise. Perhaps

[*]Leland Wilkinson is Sr. VP, SYSTAT Products, SPSS, Inc., 233 So. Wacker Drive, Chicago, IL 60606 and Adjunct Professor of Statistics, Northwestern University, Evanston, IL; e-mail: leland@spss.com. This talk was given at the Chicago chapter of the American Statistical Association Spring Conference, "The Impact of Tukey's Exploratory Data Analysis" May 5, 2000

more difficult, however, is dealing with colleagues and reviewers - hostile and friendly - who have a different methodological agenda. They come in many guises: one has all the answers, one has no questions, one asks questions that have no answers. Many say they are explorers, but instead are bureaucrats.

This may be part of a larger problem: many people who use statistics place technology before science. Science is about knowing, technology is about fashioning. Both are respectable. I hope so, because I am a technologist, not a scientist - a tool and die maker. I write the songs. I don't sing them. This circumstance has given me the opportunity to learn how researchers use statistics. Pat Fleury, head of tech support at SYSTAT, a very patient man, listened once as a researcher yelled over the phone, "I don't care what the program says, find a way to make my result significant."

Worship of statistical technology has many unfortunate consequences. First, it can make scientists suspend their disbelief; complex methods can freeze a scientist's judgment as surely as headlights freeze deer. Second, researchers are too often willing to frame questions to fit models rather than to frame models to fit questions. Third, statistics are too often used to intimidate rather than to inform - to deflect criticism rather than to encourage it. Fourth, statistics printed on paper by a computer are paradoxically more durable than inscriptions in granite - especially if they have lots of digits. Many of you doubtless have other examples. We have seen enough of them to know that in statistics, as Dostoevsky said of religion, we seek miracle, mystery, and authority.

I say this having had the privilege of serving recently on a committee advised by John Tukey. Howard Wainer, here today, was also on the committee. We were commissioned by the American Psychological Association to look into a proposal to ban significance tests in psychology journals. Our final report avoided this extreme, but along the way, we learned something about how statistics are used by psychologists. This is not a statistically naive group. Psychologists tend to study a lot of statistics in graduate school.

I was suprised to find, however, that after a 25 year vacation from reading psychology journals, I saw few changes in the way psychologists analyzed data. The graphics (or lack thereof) and statistics in these journals were pretty much the same as they were when I was in graduate school in the early 1970's. In my simple survey of APA journals, I could not find a boxplot. Medians, if they appeared at all, were used to split a sample in order to make a binary variable. (It seems that binary splits continue to be one of the most popular methods for discarding information: *data = split plus ignore*)

I am less qualified to speak of the influence of John Tukey on statistics than I am to speak of his influence on graphics and computing. The subject of graphics could be a talk in itself, and I suspect Howard Wainer will do just that better than I can. A few people imagine that Tukey's graphical thinking comprises box-plots and stem-and-leaf diagrams. Those who do should consult the Tukey collected works graphics volume edited by Bill Cleveland and published by Wadsworth some years ago.

I want to talk today, however, about how some aspects of computing have affected EDA since the orange book and how EDA has affected computing since the invention of the microcomputer. First, I will talk about the past. Then the present. And I will finish with the future.

2 The Past

My first exposure to EDA was in Bob Abelson's analysis of variance course at Yale in 1970. Abelson was one of a few lucky psychologists (unlike me) to learn from Tukey at Princeton. We had two texts in that course at Yale: the first edition of Winer and the mimeographed EDA manuscript. Every week we had to turn in a new problem set from Winer, painstakingly calculated on our Friden and Marchant machines. (Those of you who went through this experience know what it is like, after a half hour of button presses, to lose in one moment the precious string of digits in the Total Sum of Squares peeking through the tiny windows of that device. One-arm-bandits are more forgiving.) Abelson used Winer for nothing else. He recognized his obligation to tutor us in the methods the psychology journal editors would demand.

But Abelson's heart was elsewhere. Abelson spent his lecture time on letter values, median polish, additive models, conjoint measurement, monotone ANOVA, and other topics that, even today, most research psychologists would not recognize. We learned that EDA is about $data = fit\ plus\ residuals$. There are two ways to look at this equation. In model fitting we look at residuals as refractory and annoying. The devil is in the details. We do everything we can to tame them and bring them into normality - by transformations, augmenting, or changing our model. In EDA we look at residuals as our friends. God is in the details. Because this is where we find where our theory is inadequate or wrong.

The box plot is my favorite statistical graphic because it encapsulates this idea. The essence of the box plot is not the center. It is not a density estimator. Underlying the box plot are two collections of statistics: letter values, which are the results of successive splits of a batch of data (not the same as quantiles), and the fences, which tell us who is inside and who is outside. The surprises are outside the fences - those little stars and circles we see.

The box plot and other graphics in EDA were designed to be drawn by hand. Ironically, it is almost easier to compute them by hand than to do them on a computer. As David Hoaglin showed some years ago, there are several pitfalls when computing letter values on real data. One software company got the whole thing wrong from the start. They implemented boxplots by placing the center at the mean, the edges of the box at one sample standard deviation on either side of the mean, and the ends of the whiskers at two standard deviations on either side of the mean. To this company, it appears, symmetry is the highest form of art.

Although EDA was written for hand calculation, its major influence has been in the world of desktop computing. EDA and the robustness movement coincided with what we now call the personal computing revolution. Some date this revolution to the decade of the PC in the 1980's, but it really began in the

1970's with those of us who build our own machines. I built my first machine in 1977, under the tutelage of two friends Helmut Epp and Ernie Kent. Helmut's machine was a mess of wires and sockets stapled to two-by-fours and plywood; mine was Bauhaus pure, a black slab from 2001. Helmut left globs of solder and dirt on his circuit boards; I soldered, filed, and polished my circuit boards. Helmut put his computer on the floor of a messy office, to be occasionally peed on by the pet gerbils that wandered the house; I created a clean room for mine, a ventilated closet, a household computing shrine. Helmut's computer worked; mine crashed every two minutes. Helmut went on to found the school of computer science at DePaul and is now its dean. Messy offices do not imply messy minds.

By 1979, two years before the IBM PC was invented, my computer had 1 megabyte of RAM, two 8 inch floppy disk drives, each with 1.2 megabytes of storage space, an 8 megahertz processor, a drum plotter for my 3D graphics work, a 600 baud modem, and a dot matrix printer. Software included WordStar, BASIC, LISP, FORTRAN, Assembler, and an operating system that was more powerful than the Seattle Computer Products imitation DOS that Microsoft famously bought and sold to IBM as MS-DOS. In a profound sense, the IBM PC was a major step backward in the evolution of the personal computer - both in IBM's choice of a processor and in its choice of an operating system. We continue to feel the effects of this thoughtless decision today in the bugs and antiquated architecture of Microsoft Windows.

I wrote SYSTAT on the machine I built. Two characteristics of that machine had interesting consequences. First, it was an extremely fast development environment but an extremely slow runtime environment because it was efficient at integer register operations (needed for sorting, editing, and compiling) but it lacked floating point registers. Thus, I could compile the entire SYSTAT package in a minute or two, but it could take an hour to compute a set of principal components on a 25 by 25 matrix. Second, and more importantly, the machine had only 64K of address space. Like the IBM PC, it could address more by using pages of memory, but it was safer for me to write assuming users had only 64K of memory for the operating system, the program, and the data. Thanks to an overlay linkage editor called Plink, I was able to implement the only full-featured statistics package ever to run in a small memory space. This limitation had a profound effect on my thinking that remains with me today. I dislike any subroutine, procedure, or method that has more than 100 lines of code. I've written longer when I'm lazy, but I usually try to break things up in smaller pieces. It often clarifies one's thinking.

It is hard to convey the excitement we had in owning our own computers at that time. We built microcomputers to defy the authorities: the computer center staff who hid behind cement walls and plexiglass windows, the university administrations who doled out tiny portions of ”funny money” for cpu time, and the mainframe companies who wrote canned software and largely shunned Tukey's methods. We saw and envied what could be really done with computers, notably by the group at Bell Labs. But we learned the hard way that the Bell Labs BLNFY system meant for us, Bell Labs Not for You.

Anatomy is destiny. It is easy to ignore how hardware influences software. Software is pure ideas. It should run anywhere. This seems to be true for legacy systems - thirty year old software frozen, bandaged, wrapped, and dressed up to look like new. But software has a real life that ties it to a platform in a way that can be only superficially disguised. This was especially true of SYSTAT. It was written to be small for a small world. To my surprise, the largest group of SYSTAT users was ecologists. They took their Kaypro portables to the Panama jungles, African veldts, Australian outback, and Antarctica glaciers and did their plots and analyses there.

Despite what some reviewers said, and despite my including box plots and stem-and-leaf plots, SYSTAT was not an EDA package. It was Paul Velleman who, in DataDesk, introduced the personal computing world to EDA. Velleman was the first to see the EDA potential in the Macintosh. I remember John Hartigan describing to me in the mid 1980's his ideal statistical computing environment, and I remember thinking how hard it would be to rearchitect SYSTAT to fit his vision. Months later, Velleman introduced DataDesk. It incorporated many of the features Hartigan wanted.

There were other Mac programs at the time, but they were driven by dialog boxes instead of widgets and tools. Dialog boxes are the curse of Mac and Windows software that besets us even today. They enable people who do not want to get involved with their computer - indeed, who dislike computers. Dialog boxes and drop-down menus inhibit play and exploration. I nominate them for the interface design most stifling and harmful to statistical computing and data analysis.

3 The Present

It is time to move to the present. If the age of the personal computer put *analysis* in the hands of ordinary people, the age of the Internet has put *data* in the hands of people. The Internet is not about dot coms, it is about distributed computing and access to data. A new language has arisen to enable this: Java. Despite continuing and determined sabotage and subterfuge from Microsoft and despite Sun's repeated attempts to shoot itself in the foot, Java has revolutionized the capabilities of the Internet. Java is the only widely-used language designed for distributed computing. And, unlike C++, its object-oriented capabilities have been designed in from the start rather than grafted onto a foreign stalk.

The simplest way to illustrate the significance of distributed computing is by showing some examples. The first is a linked map and bar graphic. Note that it incorporates brushing and linking in a manner similar to a desktop package, but it is running in a Web browser. The sources of data may be anywhere on the net.

The second is an ordinary application far from the world of statistics or data analysis that is nevertheless derived from ideas introduced by Tukey almost 30 years ago. This graphic is an example of a new way to order airline tickets. It looks like a bar graphic, but it is much more. We may link by departing or arriving airport, airline, or other variables. And the final purchase is made simply by clicking on one of the route bars. This takes one to the web site of the airline for that specific flight.

Distributed computing means sharing data but it also means sharing computing. I remember Tukey giving the keynote address at an Interface conference some years ago. He suggested that statisticians begin devising difficult problems that would be amenable to time slicing in a distributed environment. As an example, he suggested computing convex hulls on large datasets. This is one method for identifying outliers in high-dimensional data, but it is impractical on a single-processor machine. I am not sure if anyone has taken him up on his idea, but I have seen similar ideas implemented. Some of you may have Berkeley screen savers on machines at home or the office searching for signals from extra-terrestrials at this moment, although why anyone would want to contact beings whose first gesture would be to vaporize us is beyond my understanding.

We have only begun to imagine what we can accomplish with distributed computing - the sharing of data *and* analyses.

4 The Future

Now, a brief look at the future. I am excited to be involved in the latest developments in Internet design and programming. One of the thrills is to be able to work with people under 30. As surely as yellowed lecture notes signal you are dead wood in academics, talking about the good old days of computing to people under 30 signals you don't understand the Internet - that you don't "get it." One thing my 30 years of computing has done, however, is to have made me nervous about some recent developments.

I fear we may be returning to the era of the mainframe - to centralized control of data and analysis. Commerce inevitably drives science and technology, and the centralization we are beginning to see is a normal consequence of shakeouts in technological revolutions. But those who say the architecture of the Internet prevents governmental and corporate control have not lived in a time or place of governmental or corporate control. It can happen and, in my mind, is the most likely scenario for the future of the Internet.

To make my point, I will play some word games. Biblical scholars use the terms *exegesis* and *eisegesis* to describe two different ways of interpreting a word or text. Exegesis means to interpret a term by uncovering its history, literally to bring out its meaning in context, kind of what William Safire tries to do. Eisegesis means to read into a term, literally to bring meaning to it. I want to turn three terms on their head by doing eisegisis. These "hot" terms are thin client, XML, and data mining.

First, thin client. Thin client is a term that computing executives say will solve our Internet security problems. It provides a radical solution to security by eliminating direct user access to server data and programs: keep programs off your laptop or palmtop and they can do no harm to the database back in Cleveland. Now for the eisegesis. Thin Client means you can do only what your browser allows you to do - more specifically, what Microsoft wants you to do. Thin Client means keep Java off the client. Put it only on the server, which is to destroy the reason for Java's existence. Thin client means buttons instead of sliders, pictures instead of graphs, little giggling ducks instead of real 3D rotation. Companies that tell you they are doing "thin client solutions" are telling you they do not want you to explore data. Their solutions are your conundrums.

XML means extensible markup language. It is being promoted as the method for organizing data and analyses on the Internet. XML is simply a tree of text. It began as a reasonable proposal for enriching HTML, the hypertext markup language of Internet Web pages. Now for the eisegesis. XML is no substitute for analytics. Those who would suggest that XML enables interactive exploration are ignoring the fact that XML is a tree, not a program. XML is a file format. Nothing more. Exploratory data analysis is not about pre-organizing data or pre-answering questions.

Finally, data mining is all too familiar to most of you already. If you haven't changed your title from statistician to data miner, you are already too late. Data Mining means having a programmer at Oracle, Microsoft, or IBM look in a statistics book and program a procedure in Sequential Query Language. Or, it means having a programmer at SPSS or SAS supply Oracle, IBM, and Microsoft with code that can be packaged inside their database so that users will no longer need SAS or SPSS. Data Mining means the database manager controls the data and your analysis.

Data Mining means something else. The hottest word in mining right now is "clickstream." These are the records of where you click on a Web page. Some believe there is gold in sequences of clicks. There may be, but some enthusiasts want to show you patterns without your own thinking or exploring. This has nothing to do with EDA, although it has a lot to do with invasion of privacy.

5 Conclusion

Ed Tufte, Howard Wainer, Bill Cleveland, and others have campaigned against "chart junk" and other excesses in graphics. I worry that the same obligatory sanctification that Tufte's books have received has already befallen EDA. People keep copies of Tufte's beautiful books on their home coffee tables and go to work and produce 3D pie charts in PowerPoint and Excel. People sing praises to EDA and then use an automated data mining package to tell them that men who enter supermarkets buy beer and diapers. They praise with equal enthusiasm neural networks and EDA.

We will serve John Tukey best, I think, by continuing to be sceptical toward those who apply the same model, or a hot new model, to every dataset they see. John Tukey liberated us from canned computer programs and cookbook statistics. We must be aware, however, that canned program peddlers have an Orwellian tendency to use the term EDA to describe their wares.

Lastly, we must refuse to let corporations and governments define our data and our analytic tools. The seeming chaos of the Internet can lull us into thinking it is inevitably liberating, but behind that liberty lies a very real threat of control. This is a possibility we must work actively to prevent. If we do not, it will surely happen.

References

[1] Tukey, J.W. (1977). *Exploratory Data Analysis* . Reading, MA: Addison-Wesley.