GAINING INZIGHTS FROM DATA

Chris J. Wild
Department of Statistics, University of Auckland
New Zealand
c.wild@auckland.ac.nz

Desktop data analysis tool iNZight and its web-based version iNZight Lite have been developed to accelerate the rate at which students can experience data exploration, especially multivariate data exploration. There are many "must-haves" in statistics education, and even more with the advent of "data science". However many of these imperatives actually conflict. Choices that make some things easy make others hard. iNZight has prioritised the ability to get useful output even when the user does not remember the names of appropriate techniques and to get people to "Aha" moments about data as quickly as possible without constantly being delayed by removable roadblocks. We will show something of the capabilities of iNZight and iNZight Lite but embed this in a deeper discussion of educational priorities.

INTRODUCTION

In this world there are many imperatives – if not entirely "must haves" then at least goals that almost of us would agree are very important indeed. Unfortunately these imperatives are often contradictory in the sense that the actions that they prioritise conflict. That leads us into trade-offs and compromises. That is true throughout statistics education, and in the design of statistical and educational software. We will explore some of this in the context of the iNZight system.

Right from its inception, the main aim of iNZight software project (www.stat.auckland.ac.nz/~wild/iNZight/) has been to provide an environment that allows even beginners to explore multivariate data rapidly and with a minimal learning curve. In addressing the abstract the conference presentation will show capabilities of iNZight and iNZight Lite whereas this proceedings paper emphasizes motivating philosophies and values.

THE FUTURE OF WORK, LONG-TERM VALUE, AND DEATH-DATED SKILLS

The thing that is most clear about "the future of work" is the rapidly increasing automation of virtually anything we are involved with, including working with data. In the long term, anything important that can be automated will be automated. Anything that is purely procedural can be automated. So what do we most want for our students so that they can survive and prosper in the future world of work? The ability to do things machines can't do! Human thought may mainly be required for formulating, clarifying and prioritizing ill-defined problems, or when it is sought by automated procedures that make allowances for human inputs.

In terms of what we learn, therefore, purely procedural skills are death-dated. They have no long-term value. Core big-picture conceptions (ways of thinking about things) are likely to have long-term value, but most details are also death-dated. There are two main reasons for this. The first is change. The ways we do things change over time, and at an ever-accelerating pace, even though the fundamental issues being addressed may be much more stable. The other reason is the limitations of human memory. Our memories for details fade fast. Almost none of what is "learned" in a university course, for example, sticks over the long term. This makes prioritizing a small number of big-picture learnings, to be targeted for long-term retention, very important.

The abilities to formulate goals, to find resources that can help in reaching these goals, and to follow instructions, these all have long-term value. But while the ability to operate a procedure/recipe/algorithm has great long-term value, the value of the ability to operate *any particular* procedure/recipe/algorithm is death dated. This applies to writing computer code. While the ability to code has long-term value, the ability to code *anything particular in any particular way* is death dated. Or as my contribution to Gould et al. (2018, p.460) has it, "Today's complex programming task is tomorrow's mouse-click."

We should be thinking of a world in which technology serves as an aid to human thinking and problem solving; an aid to be exploited to the full. It is most important that students be prepared to do the critically human work of: formulating desired outcomes and goals; recognizing what the technology can and cannot do in terms of assisting us in reaching those goals; and performing the thinking that is required because it extends beyond the limits of what the technology can do. In the workplaces of the future, as a consequence of the accelerating pace of change, *everyone* will have to be a life-long learner. It is our job to prepare students for lifelong learning, creative thinking and problem solving. One way we can begin to do that is to start to *populate their imaginations with possibilities* – "What sorts of things might it be possible to do?" or "In what sorts of ways might it be useful to think about this?" – as vital inputs to coming to new understandings and creative approaches to solving problems.

GETTING TO "AHA!"

The goal for early exposures to statistics that our group is beginning to prioritise most highly is the creation of excitement about '*what I can do with data and what data can do for me*'. If we can only create in students a passion for data and its possibilities for their lives ('What's in it for me?') the seeds of desire for further study and lifelong learning will have been sown. For most, this probably requires creating a broad awareness and as many 'aha!' and 'wow!' moments as possible from multivariate data in a short space of time" (Wild et al. 2017). A powerful way in to this is visualization. "Visualisation", said Martin Wattenberg, "is a gateway drug to statistics" (Aldous, 2011, p. 44).

I was reaching for a vision along these lines when I started riffing on Martin Luther King's "I have a dream" while writing this passage from Gould et al. (2018, p.459): "I have a dream of a multitude of students spellbound by the broad vistas of the data landscape. I have a dream of their flying on magic carpets that enable them to swoop effortlessly over this landscape exploring its nooks and crannies in search of its hidden treasures. I have a dream of students empowered to look at data, explore analysis systems and educational environments designed so that, like Alice in Wonderland, they keep crying 'Curiouser and curiouser!' and have the ability and confidence to go where that curiosity leads. I have a dream of educational and analysis environments designed to leverage the power of 'I wonder …?' to draw students in to learning more and more – the power of 'I wonder why …?', the power of 'I wonder what happens if …?', the power of 'I wonder what that does?', the power of 'I wonder what's around the next bend or just over the horizon?' I have a dream of software that finesses away the mundane, the mind-numbing and the soul-destroying difficulties. I have a dream of software that creates rich, virtual data-generating environments that mimic real real-world environments but enable real experiential learning in accelerated time frames. I have a dream."

MINIMISING COGNITIVE DEMANDS

Studies reported by Cowan (2000) suggest that the average person can only hold two to six pieces of information in their attention at once! Cognitive overload continually trips people up or blocks their progress. Fig. 1 of Wild (2007) illustrates an obvious cognitive reality. It shows two young people separated by a large mountain. They look very lonely and dejected – each totally oblivious to the presence of the other. But in the next frame the mountain that divides them has been reduced to the merest hillock. Their eyes meet, love strikes, happiness ensues. The point is that when we want to connect two things, having to go off and think about something else in between is like that view-obliterating mountain. It is an impediment that will often prevent valuable connections from ever being made. We need to strive very hard to avoid this sort of thing in learning experiences, and also in the tools we have students use.

CODING VERSUS POINT-AND-CLICK

We have laid out a dreamscape in which the progression from an internal desire to see something to its appearance is effortless and immediate. And we have discussed some moderating cognitive realities. We will now pull the discussion back to software issues.

There is a trend in data-science influenced statistics towards statistics education based on coding in R (Python has also been suggested) rather than point-and-click systems. In "point and click" we include systems that rely on things like menus, drag and drop, sliders, phone-type hand gestures and the other metaphors used by modern interactive systems – in contrast to systems that rely on the user communicating with the device using a language. We will consider the strengths and weaknesses of both types of system and then, in the following section, discuss solutions that combine both.

I once watched Ross Ihaka, the co-creator of R, motivating his students to learn to program. He stomped his feet several times, crying “Whaaaa” and thrusting out his pointing finger. This, he explained, is how we communicate with point-and-click systems. It is communicating like a baby. As we get older we learn to speak. Our language skills then enable us to communicate much more complex and subtle messages. And that is what programming is like. While there is a good deal of truth in Ross’ images, babies can start getting a lot of their basic needs met long before they can talk despite knowing virtually nothing at all.

Good point-and-click systems can allow users to access a large number of capabilities with minimal learning curves, conferring the ability to do much more, and in a much shorter time frame, than their ability to code ever could. Learning to use any language takes time, and memories of how to do things via coding fade fast without the reinforcement of constant use. Using help resources to rediscover how to do something can exhaust a lot of time. In menu-based systems, however, the menu choices themselves act as reminders. These factors make point-and click systems good for beginners and for occasional users. Things like looking at the choices offered by menus also make it easier to see what sorts of capabilities are on offer. And initiating procedures that a point-and-click system prioritises can be very fast indeed.

Good point and click systems can enable us to see a whole range of things we can do with our data very quickly and with very little effort. Learning to code, on the other hand takes substantial time and effort. People need good motivations to make such investments. They need to believe there will be pay-off benefits that make it all worthwhile. I was reaching for a vivid metaphor for along these lines in Wild (2015), “The difference being emphasised is the contrast between providing visions of exciting holiday destinations and providing the ability ‘to work one’s passage’ (pay for a sea voyage by working on the ship). It is the former that arouses the desire to invest in the latter.” There are other external factors that help. For example, people are increasingly associating data science with lots of good high-paying jobs. This is a powerful external motivator for making an effort to master its tools.

To date we have been pointing out some advantages of point-and-click systems. It is now time to look at the flip side. Coding solutions have many advantages over point-and-click systems. The advantage that has perhaps attracted the most attention in recent times is *reproducibility* -- the ability coding gives for someone else to reproduce (and therefore check) an analyst’s results easily. Closely related to this is the automatic generation of an *audit trail* of the steps that were taken in doing an analysis. Most importantly, any changes made to the data along the way can be seen. It is all visible right there in the code.

Another advantage of coding is *flexibility* and *extensibility*. With point-and-click interfaces it can be next to impossible to do anything beyond what the software explicitly provides for. Then there are issues of *long-run time-efficiency*. Coding constructs such as looping *facilitate* the *automation* of repetitive tasks. The speed advantage of a work sequence produced by making a series of choices, using only point-and-click communication, disappears when it is realised that the data that was used should have been changed in some way so that you have to do it all over again. With point-and-click you have to go through making all those same choices all over again. With code you make changes to a small part of the code and then just rerun the slightly modified code. It doesn’t take too many repetitions until the coding approach is faster.

Closely related are *dynamic documents* – documents such as R Markdown documents or Jupiter notebooks in which expository text is interspersed with blocks of code that generate the desired data-analysis outputs such as tables and graphs. These documents are then compiled to produce something like a statistical report, the slides for a talk, a thesis, or even a whole book. The big advantage in working in this way is that when you discover you need to change something that affects the data and will have downstream effects on model fits, tables and graphs, you do not have to do a lot of re-analysis and copy-and paste rework, you just have to recompile the document. Even at first-course level, many students are already being taught to produce their homework assignments in this way.

Retaining old code *speeds you up* whenever you want to *do something* that is *similar* to something you have done before. You just make modifications to the old code rather than starting from scratch. “Modular code”, made up of small modules, each serving a narrowly tighly-defined purpose, provides the best opportunities for repurposing. Furthermore, code *solutions can be shared* with others, thus enabling others to gain the same sorts of benefits from work you have done.

All this makes it clear to many in statistical education that students who are on a pathway to becoming statistical professionals should learn to code. They should also learn good working practices, such as: reproducible workflows, version control, collaborative development-environments, dynamic documents and all the rest. But for introductions, service courses, and people who will be at most occasional users, the case for incorporating all this additional baggage when time is constrained is weak at best. It comes at the expense of statistical fundamentals and the opening up new vistas.

As my contribution to Gould et al. (2018, p.459) has it, “We should be educating large numbers of students (I would argue virtually all students) to think with data and have some facility with conducting and critiquing real-world investigations, and much smaller numbers of people who can develop new methodologies and turn them into new tools. For the larger group, we need to get them fast to broad vistas and the big issues to create a sense of possibility and potential for their lives – to open eyes, quicken hearts, liberate the imagination, and empower – then temper this with proper caution. All of this should be facilitated by the best tools available, by software that is part of the solution (liberators, accelerators) not part of the problem (time-sinks, shackles, quicksand). If struggling to get the right stuff into and out of software chews up an appreciable proportion of these students’ time then it’s the wrong software.

The small group needs the big pictures of destinations fast too – to ground their thinking, but then back-filled by the technical computer and/or mathematical understandings and skills that will enable them to push boundaries and to improve or create new tools with even more powerful capabilities.”

ON HAVING OUR CAKE AND EATING IT TOO

Our contrast between point-and-click systems and coding solutions has been looking at extremes. Statistical point-and click systems for professionals usually provide some form of macro or scripting capabilities to increase their power and flexibility. But this is not bridging point-and-click versus coding world-views, it is a papering over of the cracks that appear towards the edges of point-and-click systems.

A much truer bridge between the worlds is provided by point-and-click systems that not only do things in response to user instructions (like produce graphs), they also generate and make available the computer code that implements those actions. R-Commander, which is probably the best known menu-driven interface to R, has been doing this for many years. My system iNZight has started to do this too. The main reason for doing it is as an aid for students in learning to code. But it also provides the other benefits: it provides code that can be re-run, shared, or put into dynamic documents, and audit trails. Why do with this with iNZight when people can already do it with R-Commander? Because iNZight provides powerful data transformation, analysis and graphics capabilities with a much shallower learning curve and a vastly reduced need to know/remember the names of things than R Commander does. This enables us to get places faster.

CONCLUSION

Broadening a message of Wild (2015) slightly by use of xxx, “I worry about starting too early with xxx. Yes we need to teach statistics majors to deal with xxx. But extracting jewels from gloop is not something most people do because they love messing around in gloop. They want the jewels. But first they have to know (i) that jewels exist, and (ii) they might be in there. So let’s first have them discover jewels in places where they are easier to find. ... all these things slow down what you can see and how fast you can see it. There should be a sniff test. Is this an enticing element of courtship? Or do I feel the skin-pricks of glass shards? So should we save it for after marriage? Or at least till after moving in?” iNZight has been designed to facilitate speed-dating and the initial phases of courtship. With code writing we are also getting some wedding cake.

REFERENCES

Aldhous, P. (2011). Culturelab: The art of data. *New Scientist*, *5*, 44.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioural and Brain Sciences*, *24*(01), 87-114.

Gould, R., Wild, C.J., Baglin, J., McNamara, A., Ridgway, J., McConway, K. (2018). Revolutions in Teaching and Learning Statistics: A Collection of Reflections. In D. Ben-Zvi., K. Makar & J. Garfield (Eds.), *International Handbook of Research in Statistics Education* (pp. 457-472). Springer International Handbooks of Education. Springer, Cham.

Wild, C.J. (2007). Virtual Environments and the Acceleration of Experiential Learning. *International Statistical Review*, *75*, 322-335.

Wild, C.J. (2015). Further, Faster, Wider. Invited discussion of Cobb, G. W. (2015), "Mere renovation is too little too late: We need to rethink our undergraduate curriculum from the ground up," *The American Statistician*, *69*, 266-282. Retrieved from https://s3-eu-west-1.amazonaws.com/pstorage-tf-iopjsd8797887/2615430/utas_a_1093029_sm0202.pdf .

Wild, C.J. (2016). Discussion: Locating statistics in the world of finding out. *International Statistical Review*, *84*(2), 194-202.
Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/insr.12152/epdf.

Wild, C.J., Pfannkuch, M., Regan, M., & Parsonage, R. (2017). Accessible conceptions of statistical inference: Pulling ourselves up by the bootstraps. *International Statistical Review*, *85*(1), 84-107. Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/insr.12117/epdf.