HANDLING COMPLEXITY IN THE DESIGN OF EDUCATIONAL SOFTWARE TOOLS

Clifford Konold
University of Massachusetts - Amherst, United States
konold@srri.umass.edu

Designers of educational software tools inevitably struggle with the issue of complexity. In general, a simple tool will minimize the time needed to learn it at the expense of range of applications. On the other hand, designing a tool to handle a wide range of applications risks overwhelming students. I contrast the decisions we made regarding complexity when we developed DataScope *15 years ago with those we recently made in designing* TinkerPlots*, and describe how our more recent tack has served to increase student engagement at the same time it helps them see critical connections among display types. More generally, I suggest that in the attempt to not overwhelm students, too many educational environments managed instead to under whelm them and thus serve to stifle rather than foster learning.*

INTRODUCTION

Put five statistics educators in a room with the objective of specifying what should be in a data analysis tool intended for young students. The list of essential capabilities they generate is guaranteed to quickly grow to an alarming length. And no matter how many capabilities are built into a tool, teachers and curriculum developers — even students — will still find things they want to do, but can't. If as a software developer you try to be helpful by including most of what everyone wants in a tool, it becomes so bloated that users then complain they can't find what they want. Thus when it comes to the question of whether to include lots of features in a software tool, it's generally "damned if you do, damned if you don't."

Biehler (1995) refers to this as the "complexity-of-tool problem." He suggests that one approach to addressing it is to design tools that become more sophisticated as the user gains expertise. This is just what successful computer games manage to do through a number of means (Gee, 2003), but it is hard to imagine implementing this in an educational software tool. The *Mini Tools* developed by Cobb, Gravemeijer and associates comprise three separate applications that the developers introduce in a specified order according to their understanding of how rudimentary skills in data analysis might develop over instruction (cf., Bakker, 2002). Perhaps this suite of tools is a simple example of the kind of evolving software Biehler had in mind.

In developing *DataScope* 15 years ago, we took a different approach to the complexity problem (Konold and Miller, 1994; Konold, 1995). *DataScope* is data analysis software intended for students aged 14-17. We conceived of it as a basic set of tools that would allow students to investigate multivariate data sets in the spirit of Exploratory Data Analysis (Tukey, 1977). To combat the complexity problem, we implemented only five basic representations: histograms (and bar graphs), box plots, scatterplots, one and two-way tables of frequencies, and tables of descriptive statistics. Our hope was that by limiting student choices, more instructional time could be focused on learning underlying concepts and data inquiry skills.

In many ways, we accomplished our goal with *DataScope*. Students took relatively little time to learn to use it, and it proved sufficiently general to allow them to flexibly explore multivariate data (Konold, 1995). However, one persistent pattern of student use troubled us. To explore a particular question, students would often select the relevant variables, then choose from the menus one of the five display options, often with only a vague idea of what the option they selected would produce. If that display did not seem useful, they'd try another, and another, until they found a display that seemed to suit their purposes. If they were preparing an assignment or report, many students generated and printed out every possible display. There are undoubtedly several reasons for this behavior; Biehler (1998) reports similar tendencies among older students using software with considerably more options. However, it seemed clear that the limited number of displays in *DataScope* explained in part this trial-and-error approach, as there was little cost in always trying everything. Had this behavior been prevalent only among novice users, it would have not been of much concern. But, it persisted as students gained experience.

When we were field testing *DataScope*, I had a fantasy that students would want to work with it outside of class — just for the fun of it, if you will. One day I walked into a class to discover that a student was already there. She had fired up the computer and was so engrossed that she didn't notice me. Trying not to disturb her, I quelled my excitement and tiptoed around her to see what data she was exploring. Alas, it was not the glow of *DataScope* lighting her face, but one of the rather mindless puzzles that early Macs included under the Apple menu. This was the closest I got in the *DataScope* days to realizing my fantasy.

We recently completed the development of *TinkerPlots*, a data analysis tool for students ages 10-14 (Konold and Miller, 2005). Many of our design decisions were driven by our view of what data analysis is and how students learn it. In this article, however, I focus on design decisions that were driven more by the fantasy of students enjoying it and using it purposefully. These decisions resulted in a tool that in some ways is a complete opposite of *DataScope*. Rather than working to reduce the complexity of *TinkerPlots*, we purposely increased it. With rare exceptions, students are extremely enthusiastic with *TinkerPlots* and frequently ask to work with it outside of class. I believe that a big part of *TinkerPlots*' appeal has to do with its complexity. In what follows I attempt to describe how we managed to build a complex tool that motivates students rather than overwhelms them.

CONSTRUCTING DATA DISPLAYS USING TINKERPLOTS

On first opening the plot window in *TinkerPlots*, individual case icons appear in it haphazardly arranged (see Figure 1). Given the goal of answering a particular question about the data, the immediate problem facing students is how to impose some suitable organization on the case icons. *TinkerPlots* comes with no ready-made displays — no bar graphs, pie charts, or histograms. Instead, students build these and other representations by progressively organizing data icons in the plot window using basic operators including *order*, *stack*, and *separate*.

Figure 1 shows data I typically use as part of a first introduction to *TinkerPlots*. I ask the class whether they think students in higher grades carry heavier backpacks than do students in lower grades. I then have them explore this data set to see whether it supports their expectations. Figures 2 - 4 is a series of screen shots showing one way in which these data might be organized with *TinkerPlots* to answer this question.

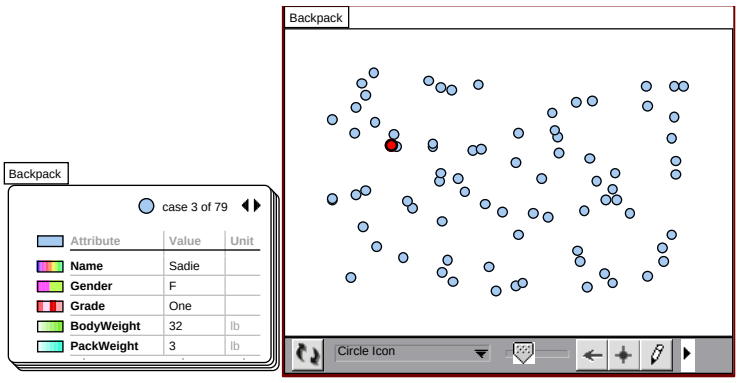

Figure 1: Information on 79 students along with their backpack weights displayed in *TinkerPlots*. Each case (student) is represented in a plot window (right) as a case icon. Clicking the Mix-up button (lower left of the plot window) sends the icons into a new random arrangement. The case highlighted in the plot window belongs to Sadie, whose data appears in the stack of Data Cards on the left.

In Figure 2, the cases have been separated into four bins according to the weight of the backpacks. This separation required first selecting the attribute PackWeight in the Data Cards and then pulling a plot icon to the right to form the desired number of bins. To progress to the representation shown in Figure 3, the icons were stacked, then separated completely until the case icons appeared over their actual values on a number line. Then the attribute Grade was selected (shown by the fact that the plot icons now appear various shades of red). With Grade selected, the grade-five students were separated vertically from the other grades. If we were to continue pulling out each of the three other grades one by one, we'd then see the distributions of PackWeight for each of the four grades in this data set (grades one, three, five, and seven). We could go on to place *dividers* to indicate where the cases cluster, or to display the location of the means of all four groups (see Rubin, Hammerman, Campbell, and Puttick (2005) for a description of the various *TinkerPlots* options that novices used to make comparisons between groups).

Making these displays in *TinkerPlots* is considerably more complex than it would be in *DataScope*, *Tabletop*, *Fathom*, or most any professional or educational tool. In almost all of these packages, one would simply specify the two attributes and the appropriate graph type (e.g., stacked dot plot). As we have seen, making such a stacked dot plot in *TinkerPlots* requires perhaps ten separate steps. What is important to keep in mind, however, is that the students, particularly when they are just learning the tool, typically do not have in mind a particular graph type they want to make as they organize the data. Rather, they take small steps in *TinkerPlots*, each step motivated by the goal of altering slightly the current display to move closer to their goal — in this case of being able to compare the pack weights of the different grades. Because each of these individual steps is small, it is relatively easy for students to evaluate whether the step is an improvement or not. If it is not a productive move, they can easily backtrack. The fact that with each step the icons animate into their new positions also helps students to determine the nature of, and evaluate, each modification.

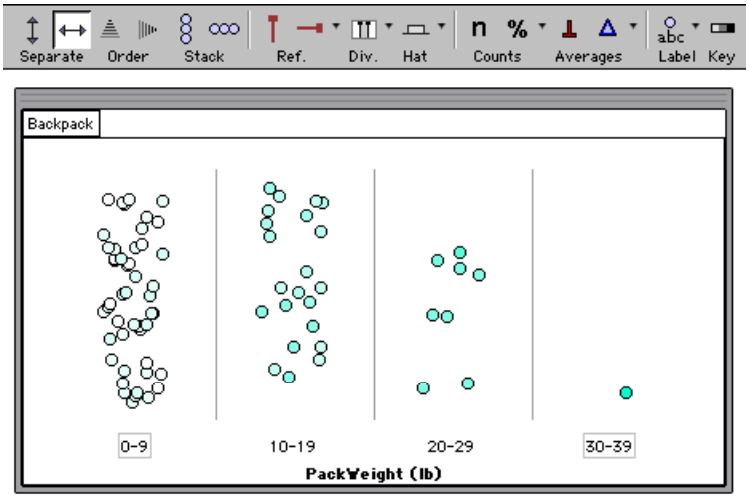

Figure 2: Plot icons separated into four bins according to the weight of students' backpacks. Shown above the plot window is a tool bar which includes various plotting options. When one of these buttons is pressed, it appears highlighted (as the horizontal Separate button currently is). Pressing that button again removes the effects of that operation from the plot.

There are a number of reasons we designed *TinkerPlots* as a construction set. A primary objective was that by giving students more fundamental choices about how to represent the data, they would develop the sense that they were making their own graphic representation rather than selecting from a set of pre-formed options. When I have students investigate the backpack data with *TinkerPlots*, I give them the task of making a graph that they can use to answer the question posed above. Having a specific task, especially when first learning *TinkerPlots*, is crucial. Without a clear goal, students would have no end to inch toward and thus no basis for evaluating their actions.

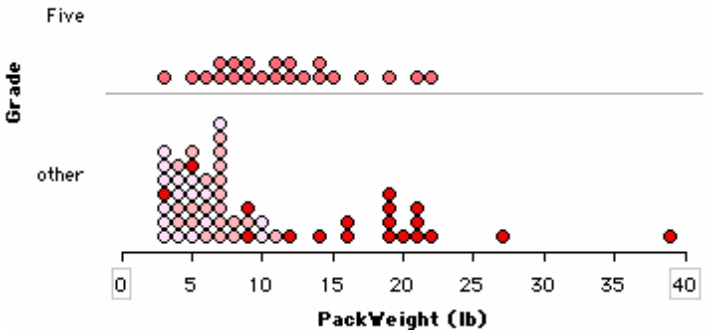

Figure 3. Cases have been stacked, then fully separated on the *x* axis until there are no bins. Then the grade five students have been separated out vertically, forming a new y axis. The cases are now colored according to Grade, with darker red indicating higher grade levels.

After about 30 minutes, most of the students have answered the question to their satisfaction. I then have them walk around the room to observe the displays that other students have made. What they see is an incredible variety, which immediately presents them with the problem of learning how to interpret these different displays, all of which are purportedly showing the same thing. But more importantly, seeing all these different graphs makes it clear to them that *TinkerPlots* is not doing the representational work for them. Rather, they are using it as they might a set of construction blocks to fashion a design of their own making. They are in the

driver's seat, which means they have to make thoughtful decisions; mindlessly pressing buttons will most likely give them a poor result. Indeed, it is quite easy in *TinkerPlots* to make cluttered and useless displays.

There are numerous factors that affect the interpretability of a data display (Tufte, 1983). Many of these factors are ordinarily controlled by a software tool. In *TinkerPlots*, we chose to leave some rather fundamental display aspects under direct user control. Figure 4 shows the four levels of Grade separated out on the y axis. But the plot icons are so large that they spill over the bin lines, and any subtle features of the four distributions are obscured. This sort of plot crowding routinely occurs as students are making various graphs in *TinkerPlots*, and it is up to them to manually control the size of icons, which they quickly learn to do. It is a control they seem to enjoy exercising.

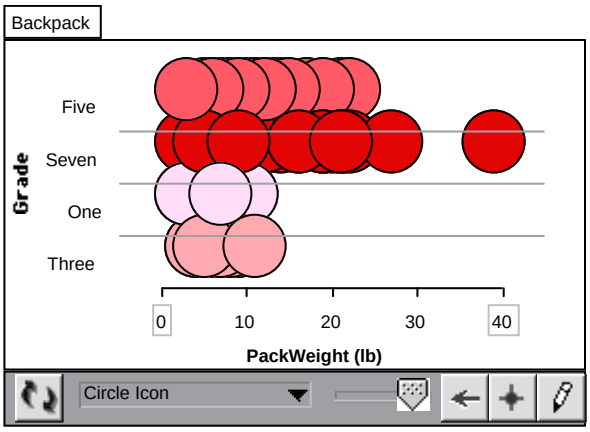

Figure 4. The plot icons in this graph are so large they obscure much of the data. Their size is under user control via the slider located on the tool bar below the plot.

Note, too, in Figure 4 that the four levels of Grade are not ordered sensibly. The current order resulted from the particular way each group was pulled out of the "other" category visible in Figure 3. In creating this data set, we intentionally entered the values of Grade as text rather than as numbers so that students would tend initially to get a display like this, with values of Grade not in an order ideal for comparing them. The ordering can be quickly changed, however, by dragging axis labels to the desired locations. Once ordered, students can sweep their eyes from bottom to top to evaluate the pattern of differences among the groups without having to continually refer back to the axis labels. In fact, it is this type of ordering from which graphic displays of data derive much of their power.

Leaving such details to the student further increases the complexity of the program. However, having to take control of things like icon size, bin size, and the ordering of values on an axis helps students to become explicitly aware of important principles that underlie good data display. Furthermore, leaving these fundamental responsibilities to the student is yet another way of communicating to them that they, and not the software tool, are ultimately in control of what they produce. Finally, these are factors which most students seem to enjoy having direct control over. Part of this satisfaction undoubtedly comes from the fairly direct nature of the control, and would be lost if instead we had used dialogue boxes.

MAKING THE COMPLEXITY MANAGEABLE

Certainly, it is not the complexity itself that makes *TinkerPlots* compelling, but the nature of that complexity. Indeed, one of the ways Biehler (1995) suggested to make a complex tool manageable is to build it around a "conceptual structure … which supports its piecewise appropriation." We chose the operators *separate*, *order*, and *stack* after having observed how students (and we ourselves) organized data on a table when it was presented as a collection of cards with information about each case on a separate card. We then worked to implement these operation in the software in a way that would allow students to see the computer operations as akin to what they do when physically arranging real-world objects. This sense — that one already knows what the primary software operators will do — becomes important in building up expectations about how the various operators will interact when they are combined, because it is this ability to combine operators in *TinkerPlots* that makes it complex, and powerful.

Implementing these intuitive operators in the software was harder than we initially expected, however. In our first testable prototype, about half of the representations that students would make by combining operators were nonsensical. To remedy this, we had to reinterpret what some of the operations did in various contexts. *Stack*, for example, works as one might expect with the case icon style used in Figures 1-4. However, there are other icon styles where the stack

operation behaves a bit differently so as to produce reasonable displays. For example, icons can be changed to *fuse rectangular*, a style used to make histograms (see bottom of Figure 5). In this case, *stack* not only places case icons on top of one another, but also widens them so that they extend across the entire length of the bin they occupy. With the icon style *fuse circular*, case icons become wedges that fuse together into a circle (pie graphs). In this case, *stack* has no function and thus if it is turned on, it does nothing. In general, the user is unaware of these differences, but pays no price for this ignorance.

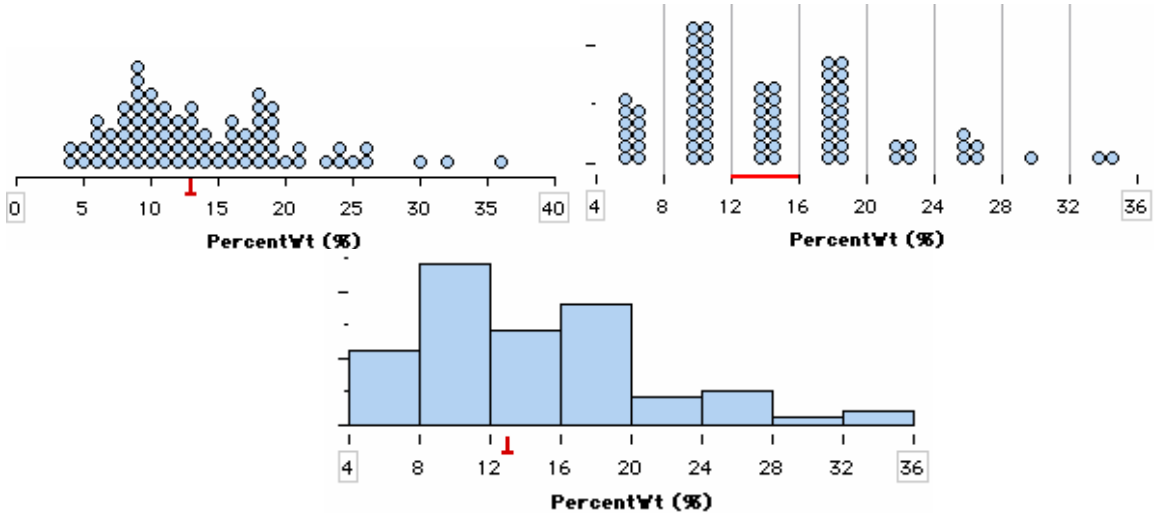

Figure 5: These graphs display the percentage the backpacks are of body weight. The top left graph shows the location of the median (inverted red T) at 13. In the binned dot plot on the top right, the median now appears as a red line below the bin, indicating that the median is in the interval 12 – 16. Changing the icon style to "fuse rectangular" makes a histogram, which now again displays the precise location of the median.

We avoid using error messages to instruct students, primarily because we worried that they would erode the attitude we are working hard to create — that the student, not the software, is in control. In some cases, applying an operator does nothing to the plot, and the button dims to indicate that it is in a suppressed state (as happens with *stack* in the context of pie graphs). Again, this goes mostly unnoticed.

However, whenever we can, we show some change in the plot, even if it is of only limited use. For example, when a numeric attribute is fully separated on an axis, students can click the median button to display the location of the median below the axis (see top of Figure 5). With a binned dot plot, however, it would be misleading to show the median as a specific point on an axis. But rather than have nothing happen when students turn on the median in this state, we display the median as a line running the length of the interval in which the median occurs (middle graph in Figure 5). While not providing much information about the value of the median, this display does help communicate the fact that when we place different values into the same bin we are, for the moment, considering them to be the same. This binned dot plot can be changed into a histogram by selecting the icon style *fuse rectangular* (see bottom graph of Figure 5). Now the median symbol once again appears at a precise location on a continuous axis. The animation from the binned dot plot to the histogram shows the cases growing in width to the edges of the bin lines, hinting at yet another change in how we are thinking of the values in a common bin.

CONCLUSION

In helping students learn a complex domain such as data analysis, we inevitably must find effective ways to restructure the domain into manageable components. The art is in finding ways to do this that preserve the essence and purpose of the pursuit. It is all too common in classrooms to find students succeeding at learning the small bits they are fed, but never coming to see the big picture nor experiencing the excitement of the enterprise. Of course, *TinkerPlots* by itself cannot change this, and much depends on how teachers and curriculum developers put it to use. Just as I

have watched in frustration as students in traditional classrooms spend months learning to make simple graphs of single attributes and never get to a question they care about, I now have had the experience of watching students work through teacher-made worksheets to learn *TinkerPlots* operations one at a time, "mastering" each one before moving on to the next. This despite the fact that the parts cannot be mastered in isolation or out of context.

After class, I spoke with the teacher who had created the worksheets and gently offered the observation that students could discover and learn to use many of the commands he was drilling them on as a normal part of pursing a question. He informed me that they didn't have time in their schedule to have students "playing around." While his response added to my despair about the direction education in the US seems to heading under the pressures of the testing/accountability movement, I also took it as another indicator that we succeeded with *TinkerPlots* in developing the tool we had hoped to — that in the absence of the strict regime of a worksheet, students seem to actually enjoy using it to explore data.

ACKNOWLEDGMENTS

I am grateful to Amy Robinson for her comments on an earlier draft. *TinkerPlots* is published by Key Curriculum Press and was developed with grants from the National Science Foundation (ESI-9818946, REC-0337675, ESI-0454754). Opinions expressed here are my own and not necessarily those of the Foundation.

REFERENCES

Bakker, A. (2002). Route-type and landscape-type software for learning statistical data analysis. In B. Phillips (Ed.), *Proceedings of the Sixth International Conference on Teaching of Statistics*, Cape Town. Voorburg, The Netherlands: International Statistical Institute.

Biehler, R. (1995). Toward requirements for more adequate software tools that support both: Learning and doing statistics. Revised version of paper presented at ICOTS-4. Occasional Paper 157. Bielefeld: University of Bielefeld.

Biehler, R. (1998). Students - statistical software - statistical tasks: A study of problems at the interfaces. In L. Pereria-Mendoza, L. S. Kea, T. W. Kee, and W-K. Wong (Eds.), *Statistical Education - Expanding the Network: Proceedings of the Fifth International Conference on Teaching Statistics*, (pp. 1025-1031), Singapore. Voorburg: The Netherlands: International Statistical Institute.

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy*. New York: Palgrave Macmillan.

Konold, C. (1995). Datenanalyse mit einfachen, didaktisch gestalteten Softwarewerkzeugen für Schülerinnen und Schüler. *Computer und Unterricht*, 17, 42-49. (English version: "Designing data analysis tools for students.")

Konold, C. and Miller, C. (1994). *DataScope*. Santa Barbara: Intellimation Library for the Macintosh.

Konold, C. and Miller, C. D. (2005). *TinkerPlots: Dynamic Data Exploration.* Emeryville, CA: Key Curriculum Press.

Rubin, A., Hammerman, J., Campbell, C., and Puttick, G. (2005). The effect of distributional shape on group comparison strategies. In K. Makar (Ed.), *Reasoning about Distribution: A Collection of Current Research Studies. Proceedings of the Fourth International Research Forum on Statistical Reasoning, Thinking, and Literacy (SRTL-4)*, [CD-ROM]. Brisbane: University of Queensland.

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

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Cheshire, CT: Graphics Press.