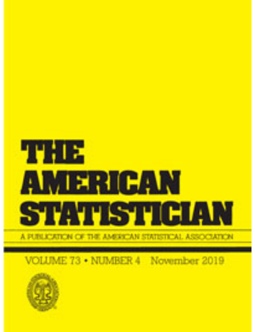

THE AMERICAN STATISTICIAN

A PUBLICATION OF THE AMERICAN STATISTICAL ASSOCIATION

VOLUME 73 • NUMBER 4 November 2019

The American Statistician

ISSN: 0003-1305 (Print) 1537-2731 (Online) Journal homepage: https://amstat.tandfonline.com/loi/utas20

Key Attributes of a Modern Statistical Computing Tool

Amelia McNamara

To cite this article: Amelia McNamara (2019) Key Attributes of a Modern Statistical Computing Tool, The American Statistician, 73:4, 375-384, DOI: 10.1080/00031305.2018.1482784

To link to this article: https://doi.org/10.1080/00031305.2018.1482784

Accepted author version posted online: 01 Jun 2018.
Published online: 05 Sep 2018.

THE AMERICAN STATISTICIAN
2019, VOL. 73, NO. 4, 375–384: Statistical Computing and Graphics
https://doi.org/10.1080/00031305.2018.1482784

Taylor & Francis
Taylor & Francis Group

STATISTICAL COMPUTING AND GRAPHICS

Key Attributes of a Modern Statistical Computing Tool

Amelia McNamara

Statistical and Data Sciences, Smith College, Northampton, MA

ABSTRACT
In the 1990s, statisticians began thinking in a principled way about how computation could better support the learning and doing of statistics. Since then, the pace of software development has accelerated, advancements in computing and data science have moved the goalposts, and it is time to reassess. Software continues to be developed to help do and learn statistics, but there is little critical evaluation of the resulting tools, and no accepted framework with which to critique them. This article presents a set of attributes necessary for a modern statistical computing tool. The framework was designed to be broadly applicable to both novice and expert users, with a particular focus on making more supportive statistical computing environments. A modern statistical computing tool should be accessible, provide easy entry, privilege data as a first-order object, support exploratory and confirmatory analysis, allow for flexible plot creation, support randomization, be interactive, include inherent documentation, support narrative, publishing, and reproducibility, and be flexible to extensions. Ideally, all these attributes could be incorporated into one tool, supporting users at all levels, but a more reasonable goal is for tools designed for novices and professionals to "reach across the gap," taking inspiration from each others' strengths.

ARTICLE HISTORY
Received September 2016
Revised May 2018

KEYWORDS
Bootstrap; Data visualization; Exploratory data analysis; Randomization; Reproducibility; Software design; Software evaluation

1. Introduction

Tools shape the way we see the world, and statistical computing tools are no exception. Affordances for building graphics, representing data, and modifying analysis all impact how users conceive of their statistical products. As our world becomes increasingly data-driven, it is important to critically examine the tools we are using and look toward the future of computational possibilities.

The use of the term "tool" to mean computer software or a programming language harkens to a time when computers do more than just amplify human abilities: they also augment them (Pea 1985). In the same way, physical tools allow us to do more than we could on our own, computers can allow humans to "see" and "think with" data in higher dimensions and with more clarity than they otherwise could.

Statistical computing tools have historically been delineated into tools for learning and tools for doing statistics (Baglin 2013; McNamara 2015). Interestingly, while statisticians have thought and written about principles underlying the tools for learning statistics, almost no critical work has been done to evaluate professional tools for doing statistics. In the educational context, Rolf Biehler's 1997 article Software for Learning and for Doing Statistics outlined principles for a statistical computing tool that would support novices in learning statistics and data analysis (Biehler 1997). It provided a framework for the assessment of statistical education software, and motivated the development of new tools. The motivation and criticism of professional tools is much less rich, and tends to focus on language properties (Ihaka and Gentleman 1996; Morandat et al. 2012). Today, the lines between educational and professional tools are starting to blur, and we believe this lowers the barrier to entry for statistical computing.

This article presents a list of attributes capturing features needed for tools for both novices and professionals. The attributes aim to be as broad as possible, but they are focused specifically on the development of more supportive environments. Hopefully, this list can be used to frame critical discussions of statistical computing tools of all types. The target audience of this article is software developers who are considering the development and improvement of statistical computing tools. Practicing educators may also find the article interesting, as it forms a scaffolding for evaluating existing tools and deciding which to use in a particular course. And, statisticians could do to think more critically about the tools they use and help create.

Although the attributes have been designed to be applicable to a broad range of users, it is useful to focus our discussion on "a user" rather than "the user" (Agre 1995). For purposes of discussion, we consider our target user to be a journalist looking to bring more computation into their work. The practice of data journalism has been accelerating, but journalism schools have been slow to modify the curriculum to teach computational skills (Berret and Phillips 2016). As a result, many journalists have limited experience with programming and statistics, but want to tell data-driven stories. They need to move from novices to producers very quickly. A key goal of journalism is communication, and as news publications have embraced the interactive web, journalists are at on the forefront of publishing modern, data-rich reports. Considering a data journalist as our

CONTACT Amelia McNamara amcnamara@smith.edu Department of Statistical and Data Sciences, Smith College, Northampton, MA 01063.

target user means prioritizing tools that are easy to learn but also powerful and flexible.

We could have focused just as easily on a number of other specific users. For example, a graduate student in a scientific or social scientific field who needs to use statistical methodology to complete their work. We may imagine those graduate students are already using a statistical computing tool, but many use Excel (Weiss 2017), and if they need to get up to speed with another package they are often self-taught (Lowndes et al. 2017). Once again, these users need tools that are easy to use, and become "invisible," allowing them to get their work done. Because of the increasing importance of reproducibility in science, they also need tools that allow them to document their work. For someone who uses a statistical computing package every day, it may be hard to imagine being new to it, or conceptualize how the tool could be improved. However, to broaden the use and understanding of statistics, we need to make it easier to do statistics, and to do statistics well.

Many of the ideas presented here are not new. In particular, these attributes attempt to distill principles and characteristics proposed by Rolf Bieher, Alexander Repenning, and John Tukey (Tukey 1965; Biehler 1997; Repenning, Webb, and Ioannidou 2010), while also considering the recent developments in data and computing. John Tukey was considering the "technical tools of statistics" in 1965, and describing a vision for the future of statistical programming tools, including "More of the essential erector-set character of data-analysis techniques, in which a kit of pieces are available for assembly into any of a multitude of analytical schemes" (Tukey 1965). Thirty years later, Rolf Biehler defined three primary problems, the "complexity of tool problem" (existing tools were too hard for novices to learn), the "closed microworld problem" (learning tools were designed for one particular type of problem or dataset and could not be extended) and the "variety problem" (because of the closed microworld problem, it was necessary to use many tools to do everything an instructor wanted to cover) (Biehler 1997). Most recently, Alexander Repenning, David Webb, and Andri Ioannidou outlined the six requirements for a "computational thinking tool," including having a low threshold, a high ceiling, and being equitable (Repenning, Webb, and Ioannidou 2010).

A survey of statistical computing tools (McNamara 2016) helps ground these ideas in the existing computational landscape. Again, since statistical computing tools have often been delineated into professional and educational tools, we take representative examples from each "category" when we refer to existing software. When tools for learning statistics are mentioned in this article, the most common examples will be TinkerPlots and Fathom, two interactive tools for statistics education (Finzer 2002; Konold and Miller 2005). These tools are graphical, drag-and-drop interfaces that make analysis highly visual. Most references to professional tools will be to the programming language R (R Core Team 2016) or to SAS, Stata, and SPSS.

Considering the various positive qualities of current tools for doing and teaching statistics alongside Biehler's goals (Biehler 1997) and combining them with ideas from Repenning, Webb, and Ioannidou (2010) and Tukey (1965), we developed a list of 10 attributes for a modern statistical computing tool. These are summarized in Table 1. While these attributes attempt to be exhaustive, they are also designed as a springboard for discussion. Because there has been relatively little recent critique given to statistical computing tools, this list is an attempt to start the conversation.

Table 1. Summary of attributes.

1. Accessibility
2. Easy entry for novice users
3. Data as a first-order persistent object
4. Support for a cycle of exploratory and confirmatory analysis
5. Flexible plot creation
6. Support for randomization throughout
7. Interactivity at every level
8. Inherent documentation
9. Simple support for narrative, publishing, and reproducibility
10. Flexibility to build extensions

Each requirement will be discussed in more detail in its respective section.

2. Accessibility

Tools should always be accessible, particularly to new users. As a baseline requirement, software should be affordable (or free), work with a variety of operating systems, and be easy to install (Dunham and Henessy 2008; Repenning, Webb, and Ioannidou 2010). In this context, most tools for teaching are accessible, because they are designed to work across platforms and are priced inexpensively. However, professional tools tend to be expensive and inaccessible for nonprofessional or occasional users. They are not accessible for small newspapers, nonprofits, individuals, or K–12 school systems.

Users in these contexts must consider if they have the funding for a tool, if it will work on the computers they have access to, and if they have the user privileges to install software. System administrators can be few and far between in newsrooms and underfunded school systems. One way to ensure accessibility is to create a web-based tool—cloud services allow users to access software from any machine with internet access.

Beyond the accessibility of a tool to the masses, it is also important to consider the needs of people with disabilities. For a tool to be required in public schools, it must be compatible with accessibility features on modern computers (Office of the Chief Information Officer 2001). Some progress has been made on programming languages accessible for blind users (Stefik, Hundhausen, and Smith 2011; Godfrey 2013), but given that many educational tools are visual, it is not clear if any of them are accessible to blind users. Of course, there are other disabilities that can impact a person's ability to use a tool. Considering "universal design" (a principle of designing things to be usable by all people) (Connell et al. 1997) should be an aspect of the development of any new statistical computing tool.

3. Easy Entry for Novice Users

Tools to be used by novices—and really, all tools—should make it easy to get started. This attribute comes directly from Reppenning et al.'s work on tools for computational thinking (Repenning, Webb, and Ioannidou 2010). It should be clear what the tool does, how to use it, and what the most salient components are. The tool should provide immediate gratification, rather than a period of frustration eventually leading to success.

Easy entry means users should be able to jump directly into "doing" data analysis without having to think about minutiae. Novices should be able to begin exploratory data analysis within the first 10–15 min of using a tool.

Biehler argued, "In secondary education, but also in introductory statistics in higher education, using a command language system is problematical. We are convinced that a host system with a graphical user interface offers a more adequate basis" (Biehler 1997). By Biehler's estimation, a system providing easy entry for novices will likely have a visual component, either initially or throughout.

Indeed, visual tools like TinkerPlots and Fathom allow novices to create linked plots and multivariate data visualizations within the first minute of beginning the software. Curriculum development using the programming language R has begun to put first emphasis on exploratory data analysis, rather than data structures (Pruim, Horton, and Kaplan 2014), so these goals can also be achieved in a scripting context. Given the success of the blocks-based language Scratch in computer science education (Resnick et al. 2009), it seems possible that a graphical system would be better for novices. However, there are many other ways by which easy entry could be achieved, such as the use of language levels (Hsia et al. 2005), or accessible IDEs (Kölling 2010).

4. Data as a First-Order Persistent Object

A data analysis tool must necessarily deal with data. A tool cannot be considered to be designed for statistical computing if it does not make data its primary object of interest. The way data are formatted and represented within the system is also of crucial importance. In this context, formatting and representation refer specifically to how the data appear to the user, not how they are stored within the computer's memory system.

Modern data analysis tools should make it easy to access common data types (flat files, hierarchical data formats, APIs, etc.) and "see" the full data (whether in a format allowing for value-reading or a higher-level view). Data should be a persistent object, with a reproducible workflow of wrangling to take raw data to clean.

4.1. Viewing Data

Many tools (including spreadsheets) make a view of the data the primary focus. In conversations with data journalists, they often mention scrolling through a spreadsheet, "reading" the data values as their first line of inquiry. Scientists also like to visually look through their data when they begin. While there are few recent studies to support this, an early experiment on Lotus 1-2-3 suggested users spend around 42% of their time viewing individual cells (Brown and Gould 1986). In contrast, programming languages like R and Python have traditionally not shown data to users when it is read in, instead requiring the use of function calls to view the first few rows of data. This can be a sticking point for users transitioning from spreadsheet programs, so Integrated Development Environments providing a data preview have become popular (RStudio Team 2014).

Whether provided by default or requested by the user, most current tools provide data such that users can immediately read each individual value. However, there are other ways to "see" an entire dataset. For example, Victor Powell's CSV Fingerprint creates a colored image as a high-level overview of the data (Powell 2014). Colors indicate data types (to see whether it is mostly numeric, categorical, integer, etc.) and missing data (Powell 2014). This simple visualization gives a lot of insight, and suggests that there may be other visual metaphors to represent raw data that could be equally helpful. The more a user can glean from an initial glance at their data, the easier it is for them to begin to dig into it.

4.2. Rectangular Versus Hierarchical Data

Analysts are typically accustomed to thinking of data in a tidy rectangular format, composed of rows and columns or observations and variables. Rectangular data can be considered "tidy" if every row represents one case (e.g., a person, gene expression, or experiment), and every column represents a variable (i.e., something measured or recorded about the case) (Wickham 2014a). Tidy data are often visualized as a spreadsheet, and spreadsheets are the most common way people around the world interact with data (Bryan 2016).

Interestingly, novices who have not encountered data before often do not use rectangular formats to represent their data, but rather default to a list-based or hierarchical format (Lehrer and Schauble 2007; Finzer 2013, 2014). So, although rectangular data have become prevalent, it may not be the most natural format. There are hierarchical and list-based data formats like JSON and XML which are used commonly on the web. These types of data are important for data science (Nolan and Temple Lang 2014). However, they may require the development of new visual metaphors because the tidy rectangle will no longer suffice. How can you see a clear overview of an entire hierarchical dataset? One observation may stretch down the screen, and Powell's colored overview certainly does not directly translate.

5. Support for a Cycle of Exploratory and Confirmatory Analysis

Statistical computing tools should promote exploratory analysis, and its twin, confirmatory analysis. The complementary exploratory and confirmatory cycles were suggested by John Tukey in his 1977 book, and have been reemphasized by current educators (Tukey 1977; Weisberg 2005; Biehler et al. 2013). The use of the term "cycle" indicates how iterative the data analysis process is. Each step can lead back to prior steps. The cycle can include generating statistical questions, collecting data, analyzing data, and interpreting results (Carver et al. 2016). Hadley Wickham lists import, tidy, transform, visualize, model, communicate (Wickham 2014b). In a pedagogical setting, educators often talk about the PPDAC cycle: Problem, Plan, Data, Analysis, Conclusions, typically attributed to Wild and Pfannkuch (1999).

The idea of EDA is to explore data deeply by computing descriptive statistics and making many graphs—of one variable or several—to gain an understanding of the underlying structure. Although EDA can appear subjective, it sometimes comprises the best and richest method for analysis, particularly for finding patterns in data and performing informal inference (Rubin, Hammerman, and Konold 2006; Makar and Rubin 2014). Exploratory data analysis can also be used in the context of statistical modeling (Gelman 2004). If users find something interesting in a cycle of exploratory analysis, they need to follow with confirmatory analysis.

The difference between exploratory and confirmatory analysis (or informal and formal inference) is like the difference between sketching or taking notes and the act of creating the final painting or writing an essay. One is more creative and expansive, and the other tries to pin down the particular information to be highlighted in the final product. A system supporting exploration and confirmation should provide a workflow connecting these two types of activities. Users need "scratch paper"—a place to play without the results being set in stone. While data analysis needs to leave a clear trail of what was done so someone else can reproduce it, a scratch paper environment might allow a user to perform actions not "allowed" in the final product, like moving data points. Biehler called this capability "draft results" (Biehler 1997).

Many current systems for teaching statistics provide rapid exploration and prototyping (allowing users to manipulate data or play with graphic representations), but typically do not support the more formal final analysis. In contrast, professional tools tend to make it difficult to play with data (in R, creating multiple graphs takes effort, as does modifying parameter values), and they may not support cyclical exploration or rapid plot generation. Again, this is limiting, as a sense of play and discovery is important to data analysis. Data scientists repeatedly cycle back through questioning, exploration, and confirmation or inference, so analysis is never a linear process from beginning to end. A statistical computation tool should support this cyclical process.

6. Flexible Plot Creation

To fully support data analysis (both exploratory and confirmatory), a tool needs to emphasize plotting. Computational tools make it possible to visually explore large datasets in ways that would be difficult or impossible using just pencil and paper. Visualization greats John Tukey and Jacques Bertin both developed visualization methods for summarizing and visualizing patterns in data before computer graphics (Tukey 1977; Bertin 1983).

These static plots are still useful now that computers can generate them, but a statistical computing tool should give humans abilities beyond what they could achieve with pencil and paper. An exemplary method is the Grand Tour, which takes high-dimensional data and produces projections into a variety of two- and three-dimensional spaces, walking a user through many views of their data to expose clusters and trends (Buja and Asimov 1986). A simpler example that can also provide insight is the generalized pairs plot, which displays all two-variable relationships in the data (Emerson et al. 2013). These plots allow humans to look for patterns in higher dimensions than they could ordinarily conceptualize.

Providing easy plotting functionality of many variables should be a goal of every tool, whether for learning or for doing statistics. Tools, particularly those for novices, must choose whether to provide a few simple plotting functions or the ability to fully customize graphics. While it can seem simpler to provide a small set of standard data visualizations, creating visualizations from primitives both provides more flexibility for the user and reinforces the mapping between abstract data and visual aesthetics on the screen (Weisberg 2005; Wilkinson 2005; Wickham 2009). Ideally, a statistical programming tool would make it simple to begin plotting (to facilitate EDA) and to produce standard graphics, while also allowing users to create novel plot types.

7. Support for Randomization Throughout

Computers have made it possible to use randomization and bootstrap methods where approximating formulas would once have been the only recourse. These methods are not only more flexible than traditional statistical tests, but can also be more intuitive for novices to understand (Pfannkuch, Wild, and Regan 2014; Tintle et al. 2012). Randomization and simulation can help make inference from data, even if those data are from small sample sizes or nonrandom collection methods (Efron and Tibshirani 1986; Lunneborg 1999; Ernst 2004).

Randomization and the bootstrap can also be used to validate models (Gelman 2004; Buja et al. 2009; Majumder, Hofmann, and Cook 2013), provide a visual representation of uncertainty in a plot (Hullman, Resnick, and Adar 2015), or perform graphical inference, a method of assigning significance to plots by using a series of randomized plots to provide a "null" visualizations to compare true visualizations against (Buja et al. 2009; Wickham et al. 2010; Majumder, Hofmann, and Cook 2013).

These methods have been gaining popularity in statistics research and trickling down to the educational context as well. Several popular introductory statistics textbooks focus on randomization and simulation methods (Lock et al. 2012; Diez, Barr, and Çetinkaya Rundel 2014; Tintle et al. 2014), and other resources help get instructors up to speed (Hesterberg 2015). These materials avoid the issue that many introductory statistics courses fall into, where the course can begin to feel like a grab-bag of methods. Instead, they show randomization as a unifying method to answer many statistical questions using one framework.

The application of randomization and the bootstrap is a place where tools for teaching statistics shine. Popular applet collections provide simple randomization and bootstrap functionality (Chance and Rossman 2006; Lock Morgan et al. 2014). TinkerPlots and Fathom also provide intuitive visual interfaces for this (Finzer 2002; Konold and Miller 2005). However, professional tools have lagged behind. R provides the most complete functionality, but it is not always simple to use.

Because of their intuitive nature and generalizability, randomization and bootstrap methods can be helpful for novices and experts alike. They can be used in a variety of contexts, including graphical inference methods bridging the gap between exploratory and confirmatory analysis.

8. Interactivity at Every Level

Interactive systems enable users to be more engaged and playful with data. Rather than typing commands, users should be able to interact with their data. And the more direct the manipulation, the better. This means valuing pinch-zoom over a dropdown menu with an option for zoom, click-and-drag selection over a form allowing the user to enter filtering values, and linked plots and analysis over a set of disconnected products. Here, 'products' encompass anything that comes out of the analysis, including plots, model output, and summary statistics.

Interactivity is becoming standard on the web. Users of Google maps know they can pan and zoom a map, and Apple has strong opinions on which direction is more "natural" to scroll. On smartphones we launch angry birds, drop pins on our location, and swipe left to reject a date.

Data analysis platforms need to follow suit. For novices, we want to "Teach about, and with, interactive graphics" (Ridgeway 2016) so they become adept at seeing data in this way. As Biehler suggests, we want to encourage direct manipulation rather than modifying a script (Biehler 1997). Today, educational tools provide this type of direct manipulation, but professional programming tools often do not. However, even textual programs can shorten the time between making a change in the code and seeing the results. Computer futurist Bret Victor has made shortening this loop one of his driving design principles, to provide users with the ability to see the direct results of their actions without waiting for something to compile (Victor 2012). The development of d3.express shows promise in bringing this paradigm to the visualization library d3 (Bostock 2017).

In the context of statistical programming, Deborah Nolan and Duncan Temple Lang make the distinction between dynamic documents (those that are compiled and then automatically include the results of embedded code), and interactive documents (those that let a reader interact with components like graphics) (Nolan and Temple Lang 2007). Given the goals of interactivity at every level, and the importance of publishing, a modern statistical programming tool should provide "dynamic-interactive" graphics, where users can interact with any component of the document and have the results update in real time.

Interactivity can take place at three levels. The first is in the context of developing an analysis. Ideally, users should be able to build their analysis interactively. Menus and wizards are a type of "interaction," but are not direct interaction and do not add any intuition about the process. Instead, a tool should aim to allow for the most direct manipulation possible.

The second level is within the analysis session, where all results should themselves be interactive. The tool should support graphs as an interface to the data (Biehler 1997). Behaviors like brushing and linking should do dynamic subsetting (Wilkinson 2005; Few 2010). All graphs should be zoomable, support brushing and linking, and allow for simple tooltips to identify data points. It should be easy to change the data cleaning methods and see how that change is reflected in the analysis afterward, and parameters should be easily manipulable. The system should also make it possible to see multiple coordinated views of everything in the user's environment. The importance of a coordinated view is supported by researchers who suggest allowing for multiple views of the same data may help people gain a more intuitive understanding (Shah and Hoeffner 2002; Bakker 2002).

Finally, the finished data product should be interactive. This means that the audience of a piece of data analysis—even if they do not know much about statistics—could play with the parameters and convince themselves that the data were not doctored.

As may be expected, standalone educational tools do a better job of providing interactivity than professional tools.

TinkerPlots and Fathom are highly interactive, allowing users to drag-and-drop variables onto their plots and supporting brushing and linking between plots. Highlighting cases in the data table highlight them in every plot. These tools make it easy to interactively develop analysis and play with it, but do not support sharing interactive results with someone who does not have the software.

On the other hand, interaction has historically been more challenging in professional tools. The history of statistical computing traces back to the pre-graphics era of computers, so most systems rely on static code. This paradigm means users are not incentivized to return to the beginning of their analysis to see how a code modification would trickle down. If a programmer wants to adjust a parameter value in their code, they must modify the code and rerun it, making the comparison between states in their head. Comparing two states in this way may be possible, but comparing more than two is difficult. This is a cognitive burden we no longer need to put on users (Victor 2012). If results were immediately accessible, it would make it possible to make hundreds of comparisons in just a few seconds.

In recent years, some of these possibilities have begun to emerge. "Notebook" functionality in several environments allows users to execute code chunks directly within their source file (Perez and Granger 2015; RStudio Team 2016). For experienced programmers, the production of interactive documents that respond to user input is possible (Bostock 2013; Chang et al. 2015; Satyanarayan et al. 2016). While these packages allow expert users to create dynamic graphics, they are too complicated for a beginner.

As a result, most current published work with interactive abilities is the result of a bespoke process. Because few tools exist to facilitate the development of fully interactive data products, people who want to generate such products must hard-code them for a particular application. Two exemplary pieces of journalism include a simulation-based look at hurricane impacts in Houston by ProPublica, which allows readers to manipulate parameters of the simulation (Satija et al. 2016), and the IEEE programming language ratings (Cass, Diakopoulos, and Romero 2014) which provides access to the weight parameters used for each data source in the rating algorithm.

The power and usefulness of a truly interactive data analysis platform is easy to imagine. If all parameters were adjustable, it would be easier to get an intuitive sense of the parameter space, and therefore the fragility of a particular piece of an analysis.

9. Inherent Documentation

Systems should provide inherent documentation, so computing tools "highlight the logic of what is going on" (Kaplan 2007). Most programming language documentation is hard for novices to comprehend, so we first want help that is helpful. However, the idea of an inherent documentation goes one step further, to help that is integrated into the process of using a tool. Instead of having to go to a second place to learn what a feature is or what a function does, objects should provide documentation as a unified part of themselves.

Ideally, every component of a system should visually show the user what it is going to do, versus just telling them. However, even in textual languages inherent documentation can be achieved by bringing the syntax of the language more in line with human language. Function names that describe what they do are more valuable than those that preserve keystrokes. Supportive features like tab completion can make documentation of parameters more inherent to the analysis process.

For example, if a tool is going to perform k-means clustering, the basic level of documentation should be the words "k-means." Ideally, the user should see a visual representation of the algorithm, and as it is applied to the data, interim steps should be visualized (Mühlbacher et al. 2014). Of course, using a computer is not the same as moving through the real world, so interface designers must think carefully about visual metaphors that make the most sense. Sometimes, this means mimicking the real world (as in the desktop metaphor, with folder icons and a trash can) and sometimes developing a new visual language (as may need to happen for visualization of models, database operations, and the like). Interactive controls of a system should give some idea of what they are going to do, either by their design or by the presentation of "scented widgets," embedded visualizations providing hints to users about what elements are capable of (Pousman, Stasko, and Mateas 2007).

10. Support for Narrative, Publishing, and Reproducibility

One important component of data science is the communication of results. We have already considered the importance of flexible plot creation, which is a form of visual communication. In addition to plots, almost all data analytic products require some form of narrative to accompany the work and contextualize it for readers. The products of a statistical computing system should be as easy to understand as the process of creating them, and they should be simple to share with others. Integrated narrative and button-click publishing will provide affordances that support reproducibility. Reproducible, interactive workflows may help to build confidence in results because they can be easily verified even by nonexperts.

10.1. Narrative

Historically, analysis workflows have tended toward a paradigm of doing analysis in one document and narrative in another. Programmers traditionally separate the documentation of their code from the code itself (code comments notwithstanding). Data analysts often create their data analysis code first, then go back to create a narrative surrounding the analysis. Data journalists refer to the process of performing analysis in Excel and writing about the results in Word as keeping a "data diary."

In contrast, a statistical programming tool should have affordances to encourage narrative alongside or mixed in with the code to facilitate the integration of storytelling and statistical products. Donald Knuth calls this "literate programming" because it is easier for humans to read and understand (Knuth 1984).

Currently, the most successful tools allow users to write formatted text and delimited code, then process the document to create a final product with text, code, and code output (Xie 2014; Perez and Granger 2015). Even those tools leave something to be desired. They feel constrained, and do not lend themselves to the type of expressive work that characterizes data science. Delimiting code chunks is a fairly lightweight process, but it does require some additional syntax. And including incidental numbers into narrative sentences can be tricky. A better solution would allow for explicit linking between code chunks (or, automatic detection of reactive connections), and the ability to drop any piece of an analysis into the text.

10.2. Publishing

Ideally, data analysis results and related products could be published with ease. Journalists could create a data-driven website, citizen scientists could share insights in the data they helped create with their friends and family, and people working together across an organization (or across the globe) could stay up-to-date on their collaborators' contributions. In all these scenarios, the publishing format should allow for exploration (discussed in more depth in Section 8). In fact, the ideal case would be a finished product allowing for full access to all the computation in the analysis. In this way, users could continue to explore the data, modify the analysis, and see the effects of their changes on the analysis and visualizations.

As the expected user base for analysis publication is wide (encompassing both novices and experts), the language the analysis is written in should be the same as the language it is published in. Currently, it is often necessary to translate from one format to another to share analysis. For example, a data journalist using RMarkdown to document their analysis will need to format it after the fact using their newspaper's content management system. To achieve the goal of native publishing, it is likely new linkage pipelines will need to be developed to streamline these transitions.

In data journalism, simple publishing abilities for fully interactive results of a data analysis could empower journalists to produce richer articles. Such articles could be accompanied by the reproducible code that produced them, allowing readers to audit the story. Similarly, as reproducibility becomes more valued in the academic community, data products are more often accompanied with fully reproducible code. If the code were interactive, it would widen the potential audience of the academic work.

10.3. Reproducibility

Reproducibility supports the aims of science, and should therefore be integrated with the work of data science (Buckheit and Donoho 1995; De Leeuw 2009; Ince, Hatton, and Graham-Cumming 2012; Sandve et al. 2013). Teaching novices to use tools that support reproducibility can help ensure it becomes an integral part of their statistical and data workflow (Carver et al. 2016).

There are many definitions of reproducibility. Here, we take a somewhat narrow view. A reproducible analysis is one that can be rerun (potentially years later, or by a different person) with the same data to produce exactly the same result. A slight extension to this is an analysis that can be rerun with a modified version of the original data to get analogous results (Kandel et al. 2011; Sandve et al. 2013; Broman 2015). For example, the initial analysis was done on 2016 data but needed to be run again on 2017 data, or the initial analysis used corrupted data that should be replaced by a corrected version.

It may sound simple to achieve this goal. However, in practice there are many factors that make it challenging. Software versions can change, package dependencies can get broken, and—most disruptive to the process—authors often do not manage to document their entire process. They may have done data cleaning outside the main software package (e.g., the bulk of the analysis was done in R but the author did data cleaning in Excel before the analysis), or run analysis steps without adding them to the code document. They may provide out-of-date code, or code with bugs that need to be addressed before it will run. These problems can be at least partially addressed with tooling.

Integrated narrative and simple publishing will necessarily encourage reproducibility. If analysis developers are writing narrative as they write code, the results will be easier to interpret and more likely to be housed in the same place. If it is easy to publish this type of document, readers will have access to a richer version of the analysis than is typically shared. Therefore, the products of statistical computing tools should continue to become more reproducible.

However, there is more work to be done before any statistical computing tool can be said to fully support the entire spectrum of reproducibility.

A fundamental feature supporting reproducibility is the ability to save the data analysis process. Some teaching tools (e.g., applets) do not allow state to be saved in any way. In other systems, like Fathom and Excel, analysis is not reproducible because it was produced interactively. Even in 1997, Rolf Biehler was aware of this drawback to interactive systems; "It may be considered a weakness of systems like Data Desk that the linkage structure is not explicitly documented as it is the case with explicit programming or if we had written the list of commands in an editor. An improvement would be if a list of commands or another representation of the linkage structure would be generated automatically" (Biehler 1997). Most interactive tools do allow the user to save the environment that produced the product, but do not document the steps taken within the environment. An independent researcher could use the saved document to explore the analysis, but may not be able to discover the steps to produce the final product. These types of tools also make it impossible to rerun the analysis on slightly different data.

Again, professional tools allowing for the integration of narrative and code are beginning to support some of these goals. Using R and RMarkdown, for example, users can now author entire analyses within a single document, fulfilling Broman's "everything with a script" and "turn scripts into reproducible reports" (Xie 2014; Broman 2015). Some of these tools are simple enough to be integrated in introductory college statistics courses (Baumer et al. 2014). However, even experts trying to implement reproducible workflows have found it difficult to fully document their process (Garijo et al. 2013; FitzJohn et al. 2014). For novices, full reproducibility is even more challenging (Garijo et al. 2013).

Future systems should therefore be designed to support reproducibility more fully. This may entail saving a version of the computer's state, tracking all "scratch work" alongside code put into a "final draft," automatically recognizing dependencies on files, packages, and custom functions, and providing a visual representation of those dependencies to the user. This vision would move close to Nolan and Temple Lang's vision of dynamic, interactive documents (Nolan and Temple Lang 2007).

11. Flexibility to Build Extensions

Of course, a statistical computing tool must have statistical methods built into it. While these attributes have outlined elements that approach methods (such as graphics and randomization) they shy away from specifying any particular models or techniques. This is because statistics is always changing, so one of the most important attributes of a statistical computing tool is the ability to extend it.

The flexibility to build extensions is necessary to prevent a tool from becoming obsolete. Users must be able to create new components of the system as methods are developed, computers improve, or scientific discoveries are made. To be a computational thinking tool, building extensions is a required feature such that the system has a "high ceiling," preventing users from "aging out" or "experiencing out" of a system (Repenning, Webb, and Ioannidou 2010). In a statistical computing tool, it should be possible to develop new visualization types and data processes from other modular pieces.

Professional tools can be looked to for inspiration, because they tend make it easier to create new components of the system using old ones. R even has a centralized repository where other users can easily find and import others' work (R Core Team 2015). Currently, the tools easiest for novices to use fail to provide a high ceiling, although Biehler argued that "adaptability (including extensibility) is a central requirement for data analysis systems to cope with the variety of needs and users" (Biehler 1997).

Any system hoping to stay the test of time must provide the flexibility to build extensions.

12. Conclusion

This list of 10 attributes aims to encompass the most important qualities for a modern statistical computing tool. We have focused on an idealized data journalist as our target user, but hopefully the attributes are more broadly relevant, encompassing some of the needs of science and social-science graduate students, novices at a variety of other ages, and seasoned statistics professionals.

Of course, there are other features that one might desire for their tools. The list focuses on things that could be built into a system by an engineer, which overlooks the importance of a welcoming and supportive community of users. It also has not touched on the language attributes commonly cited by computer scientists, such as speed and completeness, and it assumes tools would be stable and free of errors. Does the ideal tool need to support Bayesian statistics? Should it include an algebra solver? While some of these elements can be captured by the "flexibility to build extensions," there are certainly open questions. More than anything, this list of attributes was designed to start a critical conversation about the design of statistical computing tools.

Considering the existing tools for statistical computing, McNamara (2016) suggested that none of them fulfill all the attributes outlined above. Most tools can be described as either a tool for learning statistics or a tool for doing statistics. Those for learning statistics tend to be better at accessibility, easy entry, exploratory data analysis, flexible plot creation, randomization, and interactivity. For example, TinkerPlots and Fathom are highly interactive and intuitive, but make it difficult to share results. Spreadsheets like Excel are highly accessible to a broad audience, but obscure the computational processes taking place. In contrast, professional tools like R privilege data as a first-order object, support reproducibility, and have the flexibility to build extensions, but are harder to get started using and the data-analytic products they create are usually not interactive. For more details, see McNamara (2016).

No existing tools currently satisfy all the attributes, which suggests the need for new or improved software. It would be ideal to conceive of a single tool that could support users at all levels. For example, a blocks programming language with streamlined domain-specific language could step novices into more complex analysis. However, there are few examples of similar tools in other domains so it seems unlikely such a system will emerge, and indeed, projects which try to be all things for all people often fail.

If we acknowledge that users will likely have to move from one type of tool to another, software developers should be looking for ways to "bridge the gap" between the two types of tools (McNamara 2015). In other words, in tools with traditionally difficult learning curves, designers should consider how to lower the barrier to entry, while in tools where users tend to "experience out," designers should build (either technically or pedagogically) an onramp toward the next tool. R has historically been difficult to get started using, but curricula and packages have been developed to lower the barrier to entry (Baumer et al. 2014; Pruim, Horton, and Kaplan 2014). Researchers have also begun studying instruction methods that best support learning of both statistics and statistical computing (Baglin 2013). These efforts have not solved the problem of easy entry, but are easing the transition. More work needs to be done, but other tools could take inspiration from these initial efforts.

As new tools are developed and existing ones are refined, statistical practitioners need to remain actively engaged in their development and critique to ensure they can support learning as well as doing statistics. Hopefully, this article can act as a guide as we begin to engage more fully with this conversation.

Acknowledgments

The author is grateful to Nick Horton and Rob Gould, who read drafts of this article and gave many insightful comments. Thanks also to Alan Kay and Mark Hansen, who read early versions of this work in author's doctoral dissertation.

ORCID

Amelia McNamara http://orcid.org/0000-0003-4916-2433

References

Agre, P. E. (1995), "Conceptions of the User in Computer Systems Design," in *The Social and Interactional Dimensions of Human-Computer Interfaces*, ed. P. J. Thomas, Cambridge, UK: Cambridge University Press, pp. 67–106. [375]

Baglin, J. (2013), "Applying a Theoretical Model for Explaining the Development of Technological Skills in Statistics Education," *Technology Innovations in Statistics Education*, 7, 1–17. [375,382]

Bakker, A. (2002), "Route-Type and Landscape-Type Software for Learning Statistical Data Analysis," in *Proceedings of the 6th International Conference on Teaching Statistics*, pp. 1–6. [379]

Baumer, B., Çetinkaya Rundel, M., Bray, A., Loi, L., and Horton, N. J. (2014), "R Markdown: Integrating a Reproducible Analysis Tool into Introductory Statistics," *Technology Innovations in Statistics Education*, 8, 1–29. [381,382]

Berret, C., and Phillips, C. (2016), "Teaching Data and Computational Journalism," Technical Report, Columbia Journalism School and Stanford University. [375]

Bertin, J. (1983), *Semiology of Graphics*, Madison, WI: University of Wisconsin Press. [378]

Biehler, R. (1997), "Software for Learning and for Doing Statistics," *International Statistical Review*, 65, 167–189. [375,376,377,378,379,381]

Biehler, R., Ben-Zvi, D., Bakker, A., and Makar, K. (2013), "Technology for Enhancing Statistical Reasoning at the School Level," in *Third International Handbook of Mathematics Education*, eds. M. A. Clemenets, A. Bishop, C. Keitel, J. Kilpatrick, K.-S. L. Frederick, New York: Springer Science and Business Media. [377]

Bostock, M. (2013), "D3.js: Data-Driven Documents," available at *http://d3js.org/* [379]

——— (2017), "d3.express," available at *https://medium.com/@mbostock/a-better-way-to-code-2b1d2876a3a0* [379]

Broman, K. (2015), "Initial Steps Toward Reproducible Research," available at *http://kbroman.org/steps2rr/* [381]

Brown, P. S., and Gould, J. D. (1986), "An Experimental Study of People Creating Spreadsheets," *ACM Transactions on Office Information Systems*, 5, 258–272. [377]

Bryan, J. (2016), "Spreadsheets. useR! Conference," available at *https://github.com/jennybc/2016-06_spreadsheets* [377]

Buckheit, J. B., and Donoho, D. L. (1995), "Wavelab and Reproducible Research," in *Wavelets and Statistics. Lecture Notes in Statistics* (vol. 103), eds. A. Antoniadis and G. Oppenheim, New York: Springer, pp. 55–88. [380]

Buja, A., and Asimov, D. (1986), "Grand Tour Methods: An Outline," in *Proceedings of the Seventeenth Symposium on The Interface*, Elsevier Science Publishers B. V., pp. 63–67. [378]

Buja, A., Cook, D., Hofmann, H., Lawrence, M., Lee, E.-K., Swayne, D. F., and Wickham, H. (2009), "Statistical Inference for Exploratory Data Analysis and Model Diagnostics," *Philosophical Transactions of the Royal Society A*, 367, 4361–4383. [378]

Carver, R., Everson, M., Gabrosek, J., Horton, N. J., Lock, R. H., Mocko, M., Rossman, A., Rowell, G. H., Velleman, P., Witmer, J. A., and Wood, B. (2016), *Guidelines for Assessment and Instruction in Statistics Education: College Report 2016*, Alexandria, VA: American Statistical Association. [377,380]

Cass, S., Diakopoulos, N., and Romero, J. J. (2014), "Interactive: The Top Programming Languages: IEEE Spectrum's 2014 Rating," available at *https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages*. [379]

Chance, B., and Rossman, A. (2006), "Using Simulation to Teach and Learn Statistics, ICOTS-7," available at *http://www.ime.usp.br/abe/ICOTS7/Proceedings/PDFs/InvitedPapers/7E1_CHAN.pdf* [378]

Chang, W., Cheng, J., Allaire, J. J., Xie, Y., and McPherson, J. (2015), *Shiny: Web Application Framework for R*, R Package Version 0.12.0. [379]

Connell, B. R., Jones, M., Mace, R., Mueller, J., Mullick, A., Ostroff, E., Sanford, J., Steinfeld, E., Story, M., and Vanderheiden,

G. (1997), "The Principles of Universal Design," Technical Report, The Center for Universal Design, available at *https://projects.ncsu.edu/ncsu/design/cud/about_ud/udprinciplestext.htm*. [376]

De Leeuw, J. (2009), "Statistical Software—Overview," Technical Report 570, Department of Statistics, University of California, Los Angeles. [380]

Diez, D. M., Barr, C. D., and Çetinkaya Rundel, M. (2014), *Introductory Statistics with Randomization and Simulation*, OpenIntro. [378]

Dunham, P., and Henessy, S. (2008), "Equity and Use of Educational Technology in Mathematics," in *Research on Technology and the Teaching and Learning of Mathematics* (Vol. 1), eds. M. K. Heid, and G. W. Blume, Reston, VA: National Council of Teachers of Mathematics. [376]

Efron, B., and Tibshirani, R. (1986), "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy," *Statistical Science*, 1, 54–77. [378]

Emerson, J. W., Green, W. A., Schloerke, B., Crowley, J., Cook, D., Hofmann, H., and Wickham, H. (2013), "The Generalized Pairs Plot," *Journal of Computational and Graphical Statistics*, 22, 79–91. [378]

Ernst, M. D. (2004), "Permutation Methods: A Basis for Exact Inference," *Statistical Science*, 19, 676–685. [378]

Few, S. (2010), "Coordinated Highlighting in Context: Bringing Multidimensional Connections to Light," Technical Report, Perceptual Edge. [379]

Finzer, W. (2002), *Fathom: Dynamic Data Software (Version 2.1), Computer Software*, Emeryville, CA: Key Curriculum Press. [376,378]

——— (2013), "The Data Science Education Dilemma," *Technology Innovations in Statistics Education*, 7, 1–9. [377]

——— (2014), "Hierarchical Data Visualization as a Tool for Developing Student Understanding of Variation of Data Generated in Simulations, ICOTS9," available at *http://iase-web.org/icots/9/proceedings/pdfs/ICOTS9_9B1_FINZER.pdf* [377]

FitzJohn, R., Pennell, M., Zanne, A., and Cornwell, W. (2014), "Reproducible Research is Still a Challenge," Technical Report, rOpenSci, available at *https://ropensci.org/blog/2014/06/09/reproducibility/*. [381]

Garijo, D., Kinnings, S., Xie, L., Xie, L., Zhang, Y., Bourne, P. E., and Gil, Y. (2013), "Quantifying Reproducibility in Computational Biology: The Case of the Tuberculosis Drugome," *PLOS ONE*, 8, e80278. [381]

Gelman, A. (2004), "Exploratory Data Analysis for Complex Models," *Journal of Computational and Graphical Statistics*, 13, 755–779. [378]

Godfrey, A. J. R. (2013), "Statistical Software from a Blind Person's Perspective," *The R Journal*, 5, 73–79. [376]

Hesterberg, T. (2015), "What Teachers should know about the Bootstrap: Resampling in the Undergraduate Statistics Curriculum," *The American Statistician*, 69, 371–386. [378]

Hsia, J. I., Simpson, E., Smith, D., and Cartwright, R. (2005), "Taming Java for the classroom, SIGCSE'05," available at *https://www.cs.rice.edu/javaplt/drjava/papers/drjava-language-levels.pdf* [377]

Hullman, J., Resnick, P., and Adar, E. (2015), "Hypothetical Outcome Plots Outperform Error Bars and Violin Plots for Inferences about Reliability of Variable Ordering," *PLOS ONE*, 10, e0142444. [378]

Ihaka, R., and Gentleman, R. (1996), "R: A Language for Data Analysis and Graphics," *Journal of Computational and Graphical Statistics*, 5, 299–314. [375]

Ince, D. C., Hatton, L., and Graham-Cumming, J. (2012), "The Case for Open Computer Programs," *Nature*, 482, 485–488. [380]

Kandel, S., Heer, J., Pleasant, C., Kennedy, J., van Ham, F., Henry Riche, N., Weaver, C., Lee, B., Brodbeck, D., and Buono, P. (2011), "Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data," *Information Visualization*, 10, 271–288. [381]

Kaplan, D. (2007), "Computing and Introductory Statistics," *Technology Innovations in Statistics Education*, 1, 1–16. [379]

Knuth, D. E. (1984), "Literate Programming," *The Computer Journal*, 27, 97–111. [380]

Kölling, M. (2010), "The Greenfoot Programming Environment," *ACM Transactions on Computing Education*, 10, 14:1–14:21. [377]

Konold, C., and Miller, C. D. (2005), *TinkerPlots: Dynamic Data Exploration, Computer Software*," Emeryville, CA: Key Curriculum Press. [376,378]

Lehrer, R., and Schauble, L. (2007), "Contrasting Emerging Conceptions of Distribution in Contexts of Error and Natural Variation," in *Thinking with Data*, eds. M. C. Lovett, and P. Shah, New York: Lawrence Erlbaum Associates, pp. 149–176. [377]

Lock, P. F., Lock, R. H., Lock, D. F., Lock Morgan, K., and Lock, E. F. (2012), *Statistics: Unlocking the Power of Data*, New York: Wiley. [378]

Lock Morgan, K., Lock, R. H., Lock, P. F., Lock, E. F., and Lock, D. F. (2014), "StatKey: Online Tools for Bootstrap Intervals and Randomization Tests, ICOTS-9," available at *http://www2.stat.duke.edu/kfl5/Lock2014.pdf* [378]

Lowndes, J. S., Best, B. D., Scarborough, C., Afflerbach, J. C., Frazier, M. R., O'Hara, C. C., Jiang, N., and Halpern, B. S. (2017), "Our Path to Better Science in Less Time using Open Data Science Tools," *Nature Ecology & Evolution*, 1, 0160. [376]

Lunneborg, C. E. (1999), *Data Analysis by Resampling*, Boston, MA: Cengage Learning. [378]

Majumder, M., Hofmann, H., and Cook, D. (2013), "Validation of Visual Statistical Inference, Applied to Linear Models," *Journal of the American Statistical Association*, 108, 942–956. [378]

Makar, K., and Rubin, A. (2014), "Informal Statistical Inference Revisited," in *ICOTS-9*. [378]

McNamara, A. (2015), "Bridging the Gap Between Tools for Learning and for Doing Statistics," Ph.D. dissertation, University of California, Los Angeles. [375,382]

McNamara, A. (2016), "On the State of Computing in Statistics Education," available at *https://arxiv.org/abs/1610.00984*. [376,382]

Morandat, F., Hill, B., Osvald, L., and Vitek, J. (2012), "Evaluating the Design of the R Language: Objects and Functions for Data Analysis," in *ECOOP'12 Proceedings of the 26th European Conference on Object-Oriented Programming*, Purdue University Computer Science Department, Beijing, China, pp. 1–27. [375]

Mühlbacher, T., Piringer, H., Gratzl, S., Sedlmair, M., and Streit, M. (2014), "Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations," *IEEE Transactions on Visualization and Computer Graphics*, 20, 1643–1652. [380]

Nolan, D., and Temple Lang, D. (2007), "Dynamic, Interactive Documents for Teaching Statistical Practice," *International Statistical Review*, 75, 295–321. [379,381]

——— (2014), *XML and Web Technologies for Data Sciences with R*, New York: Springer-Verlag. [377]

Office of the Chief Information Officer. (2001), "Requirements for Accessible Electronic and Information Technology Design," Technical Report, Department of Education, available at *https://www2.ed.gov/fund/contract/apply/clibrary/software.html?exp=0*. [376]

Pea, R. D. (1985), "Beyond Amplification: Using the Computer to Reorganize Mental Functioning," *Educational Psychologist*, 20, 167–182. [375]

Perez, F., and Granger, B. E. (2015), "Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science," Technical Report, Project Jupyter, available at *http://archive.ipython.org/JupyterGrantNarrative-2015.pdf*. [379,380]

Pfannkuch, M., Wild, C., and Regan, M. (2014), "Students' Difficulties in Practicing Computer-Supported Data Analysis: Some Hypothetical Generalizations from Results of Two Exploratory Studies," in *Using Tools for Learning Mathematics and Statistics*, eds. T. Wassong, D. Frischemeier, P. R. Rischer, R. Hochmuth, and P. Bender, New York: Springer, pp. 393–403. [378]

Pousman, Z., Stasko, J. T., and Mateas, M. (2007), "Casual Information Visualization: Depictions of Data in Everyday Life," *IEEE Transactions on Visualization and Computer Graphics*, 13, 1145–1152. [380]

Powell, V. (2014), "CSV Fingerprints," available at *http://setosa.io/blog/2014/08/03/csv-fingerprints/* [377]

Pruim, R., Horton, N. J., and Kaplan, D. (2014), *Start Teaching with R*. Project MOSAIC. [377,382]

R Core Team (2015), "Comprehensive R Archive Network," available at *http://cran.r-project.org/* [381]

——— (2016), *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing. [376]

Repenning, A., Webb, D., and Ioannidou, A. (2010), "Scalable Game Design and the Development of a Checklist for Getting

Computational Thinking into Public Schools, SIGCSE'10," available at *https://dl.acm.org/citation.cfm?id=1734357* [376,381]

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009), "Scratch: Programming for All," *Communications of the ACM*, 52, 60–67. [377]

Ridgeway, J. (2016), "Implications of the Data Revolution for Statistics Education," *International Statistical Review*, 84, 528–549. [379]

RStudio Team (2014), "RStudio: Integrated Development for R," available at *http://www.rstudio.com/products/rstudio/* [377]

——— (2016), "R Notebooks," available at *http://rmarkdown.rstudio.com/r_notebooks.html* [379]

Rubin, A., Hammerman, J. K., and Konold, C. (2006), "Exploring Informal Inference with Interactive Visualization Software," in *Research Papers from ICOTS 7*. [378]

Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013), "Ten Simple Rules for Reproducible Computational Research," *PLoS Computational Biology*, 9, e1003285. [380]

Satija, N., Collier, K., Shaw, A., and Larson, J. (2016), "Hell and High Water," *The Texas Tribune*, available at *https://projects.propublica.org/houston/*. [379]

Satyanarayan, A., Russell, R., Hoffswell, J., and Heer, J. (2016), "Reactive Vega: A Streaming Data Flow Architecture for Declarative Interactive Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 22, 659–668. [379]

Shah, P., and Hoeffner, J. (2002), "Review of Graph Comprehension Research: Implications for Instruction," *Educational Psychology Review*, 14, 47–69. [379]

Stefik, A., Hundhausen, C., and Smith, D. (2011), "On the Design of An Educational Infrastructure for the Blind and Visually Impaired in Computer Science, SIGCSE'11," available at *https://dl.acm.org/citation.cfm?id=1953323* [376]

Tintle, N., Chance, B., Cobb, G., Rossman, A., Roy, S., Swanson, T., and VanderStoep, J. (2014), *Introduction to Statistical Investigations*, New York: Wiley. [378]

Tintle, N., Topliff, K., Vanderstoep, J., Holmes, V.-L., and Swanson, T. (2012), "Retention of Statistical Concepts in a Preliminary Randomization-Based Introductory Statistics Curriculum," *Statistics Education Research Journal*, 11, 21–40. [378]

Tukey, J. W. (1965), "The Technical Tools of Statistics," *The American Statistician*, 19, 23–28. [376]

——— (1977), *Exploratory Data Analysis*, Boston, MA: Addison-Wesley Publishing Company. [377,378]

Victor, B. (2012), "Inventing on Principle," available at *http://worrydream.com/#!/InventingOnPrinciple* [379]

Weisberg, S. (2005), "Lost Opportunities: Why we need a Variety of Statistical Languages," *Journal of Statistical Software*, 13, 1–12. [377,378]

Weiss, C. J. (2017), "Perspectives: Teaching Chemists to Code," *Chemical & Engineering News*, 95, 30–31. [376]

Wickham, H. (2009), *ggplot2: Elegant Graphics for Data Analysis*, New York: Springer. [378]

——— (2014a), "Tidy Data," *Journal of Statistical Software*, 59, 1–23. [377]

——— (2014b), "Why dplyr? useR! Conference," available at *https://www.youtube.com/watch?v=dWjSYqI7Vog* [377]

Wickham, H., Cook, D., Hofmann, H., and Buja, A. (2010), "Graphical Inference for Infovis," *IEEE Transactions on Visualization and Computer Graphics*, 16, 973–979. [378]

Wild, C. J., and Pfannkuch, M. (1999), "Statistical Thinking in Empirical Enquiry," *International Statistical Review*, 67, 223–265. [377]

Wilkinson, L. (2005), *The Grammar of Graphics (Statistics and Computing)*, New York: Springer Science and Business Media. [378,379]

Xie, Y. (2014), *Dynamic Documents with R and knitr (The R Series)*, Boca Raton, FL: Chapman & Hall/CRC. [380,381]