International Statistical Review (2007), 75, 3, 295–321 doi:10.1111/j.1751-5823.2007.00025.x

Dynamic, Interactive Documents for Teaching Statistical Practice

Deborah Nolan[1] and Duncan Temple Lang[2]

[1]*Department of Statistics, UC Berkeley, 367 Evans Hall, Berkeley, CA 94720-3860, USA*
[2]*Department of Statistics, UC Davis, 4210 Math Sci, 1 Shield Ave, Davis, CA 95616, USA*
E-mails: nolan@stat.berkeley.edu, duncan@wald.ucdavis.edu

Summary

Significant efforts have been made to overhaul the *introductory* statistics courses by placing greater emphasis on statistical thinking and literacy and less on rules, methods and procedures. We advocate broadening and increasing this effort to *all levels* of students and, importantly, using topical, interesting, substantive problems that come from the actual practice of statistics. We want students to understand the thought process of the "masters" in context, seeing their choices, different approaches and explorations. Similar to Open Source software, we think it is vital that the work of the community of researchers is accessible to the community of educators so that students can experience statistical applications and learn how to approach analyses themselves. We describe a mechanism by which one can collect all aspects or fragments of an analysis or simulation into a "document" so that the computations and results are reproducible, reusable and amenable to extensions. These documents contain various pieces of information (e.g. text, code, data, exploration paths) and can be processed to create regular descriptive papers in various formats (e.g. PDF, HTML), as well as acting as a database of the analysis which we can explore in rich new ways. Researchers, instructors and readers can control the various steps in the processing and rendering of the document. For example, this type of document supports interactive components with which a student can easily control and alter the inputs to the computations in a semi-guided fashion, gradually delve deeper into the details, and go on to her own free-form analysis. Our implementation for this system is based on widely used and standardized frameworks and readily supports multiple and different programming languages. Also, it is highly extensible which allows adaptation and future developments.

Key words: Education; Technology; XML; R.

1 Introduction

Leading statistics educators have long called for reform in the way we teach our introductory courses to place greater emphasis on statistical thinking and data analysis and less on mathematical derivations and computational recipes (Cobb & Moore, 1997; Moore, 1997, 2005). We applaud the efforts many have made in this direction by introducing textbooks that include case studies to motivate statistical topics and activities to help convey basic statistical concepts (e.g. De Veaux *et al.*, 2007; Ramsey & Schafer, 2002; Scheaffer *et al.*, 1996; Watkins *et al.*, 2003; Utts & Heckard, 2003; Utts, 1999). Yet, we have not gone far enough in this direction. Reform must extend beyond our introductory courses to the entire curriculum, both undergraduate and

graduate. To maintain a healthy discipline, we must not only improve quantitative literacy among consumers of statistics, but also work to attract and train bright, talented students.

We present a multi-faceted approach to change the way we teach statistics that takes a radical stance in its calls for reform. The ideas presented here encompass guiding principles for changing what we teach and how we teach, technologies that make this change feasible, and ways in which the statistics community can support a significant reform in educating future statisticians.

We advocate that it is important to do more than simply repair existing syllabi. We must break from the constraints of current curricula, and design syllabi from scratch, making use of the large collection of tools and experiences at our disposal. These tools should include *visualization*, *simulation* and *computing* as they offer powerful complements to mathematics. For example, simulation offers a (literally) constructive approach to studying statistical concepts, and for many students, hands-on constructive techniques aid understanding as much as or more than mathematical descriptions. Also, if students are facile with computing, then they will be in a position to work on interesting topical scientific problems, including current research problems. Through computing we can succeed in attracting and training bright, talented students to the field. Instead of focussing on methods and rules and relying primarily on mathematics, we must teach from the vantage point of statistical concepts flowing from contextual problem solving with real data. When this happens, alternative tools will take on more prominent and useful roles in statistics education and they will have the potential to radically influence what and how we teach.

More concretely, our proposal includes a novel design for interactive, dynamic, electronic documents that supports the synthesis of statistical theory and practice and that employs multiple approaches to conveying statistical concepts. These documents would provide resources to instructors to assist them in teaching in new ways because they would open up the thought process, intuition and experience behind a data analysis, which is hardest to teach and to learn. Technology plays a creative role in this proposal, and it places us in a position to be be able to sustain and improve upon the initial directions in the reform of the statistics curriculum. And, importantly, the proposed technological approach would support a model for cooperation between statisticians active in research and consulting and the community of statistics educators that would facilitate the flow of data analysis projects and current research methodologies into the curriculum.

2 Current Challenges

The current state of statistics education presents many challenges in this endeavor.

2.1 Historical Stranglehold on the Curriculum

We agree with Efron (as quoted by Rossman & Chance, 2003, p. 3) that theoretical statistics courses are caught in a time warp that bores teachers and subsequently bores students. We have overwhelming anecdotal evidence of this phenomenon. (Recall the response you receive when you introduce yourself as a statistician.) A lesson in hypothesis testing offers one example: statistics texts place too much emphasis on sets of rules developed in the 1950s for various test statistics and too little attention goes to the main notions behind testing. Even worse, we teach these same sets of rules for testing over and over again in our curriculum, in introductory, advanced undergraduate and graduate level courses. This approach fails to teach modern developments in statistics and fails to convey the excitement of statistical practice.

2.2 Limitations of Communicating in One Language: Mathematics

Advanced courses heavily, if not solely, rely on mathematics as the means for communicating and demonstrating statistical ideas. After all, that is how many of us were trained. This approach restricts our ability to convey statistical concepts in a fuller light. Although many tasks are easier to convey through mathematics, others are more appropriately conveyed through a plot, a simulation study or experience with data. These other equally valid approaches, such as visualization and simulation provide alternative insights into the field. See Moore (2005) and Wild & Pfannkuch (2004) for similar claims.

2.3 Missing in Action: The Statistical Thought Process

Traditional courses do 'not attempt to teach what we do, and certainly not why we do it' (Efron, as quoted by Rossman & Chance, 2003, p. 3), yet intuition and experience of methodology in the scientific context are essential to learning how to think statistically (Wild & Pfannkuch, 1999). While many courses teach methodology, either the theory or the application, very few focus primarily on the skills needed to approach a scientific problem from a statistical perspective. An 'application' is too often just an example of how to apply a particular statistical method to some data, rather than being a scientific application that calls for a statistical method. Further, simply letting students do projects, although a valuable exercise, is far from adequate when it is the student's sole encounter with statistical thinking in a real setting. For those learning statistics, the intuition and experience that are necessary for good data analysis are the hardest things to learn.

2.4 Lack of Resources

Finding interesting and topical scientific problems with accompanying data in a form accessible to instructors who want to teach in this experiential way is difficult. The Internet provides a great resource for data, but often falls short in supplying analysis and context. Articles that present applications are plentiful in research journals, but the analysis is typically presented as a completed work and the thought process that led to the conclusions and approaches are omitted. Also, rarely are the data available.

Exceptions do exist. For example, *Statistics* (Freedman *et al.*, 1998) offer case studies to illustrate important statistical concepts, but by their very nature, they can not engage students in all dimensions of statistical thinking. They teach valuable lessons about a statistical concept, however, they do not necessarily teach the student how to glean these insights themselves. *Statistics: A Guide to the Unknown* (Peck *et al.*, 2004) is another example. These stories present scientific questions with statistical approaches and solutions, yet, given their format, it is difficult for the student to scratch below the surface of the presentation. The text *Stat Labs* (Nolan & Speed, 2000) places the scientific problem as the centerpiece of the course, provides relevant data, and develops statistical theory to answer the subject-matter questions. However, the data analysis is left to the students and instructors to complete on their own. The approach of these books (with very different audiences) is in many ways good for statistics education, but there is room for improvement.

2.5 Inadequate and Unsupported Infrastructure

The last decade has seen a frenzy of activity around interactive tools for teaching statistics, often via the Web. Technology offers the ability to design new ways to engage students in statistical

thinking, but too often it is relegated to simple tasks such as seeing what happens when the bin widths of a histogram change, or to prescribed computations with synthetic, overly simplistic labs or class-generated data. Many have focused on implementing material from textbooks in this new medium, while it is far from obvious that content designed for a static medium (the textbook) is suited to an interactive one. The necessary technical details and skill sets to create interactive materials have led many instructors to focus less on how and what is being taught and more on implementation aspects. Also, because new technology typically does not include basic statistical methods, using it requires spending much time on redeveloping these methods and less time on considering how best to teach with the technology. Further, these activities often can neither be maintained by the owner or tailored by another instructor, resulting in limited use. Cobb warned against such a situation: 'few have the vision and understanding' to design computer activities and we will get in return a 'virtual landfill of multi-mediocre techno-trash' (Moore *et al.*, 1995). Unfortunately, the pedagogical expense of these mistakes are borne by the consumer, i.e. the student.

3 Guiding Examples

Two recent experiences have fundamentally guided us in addressing these challenges. One is our experience designing and running a week-long summer program to introduce undergraduates to modern statistical research problems with the aim of enticing them to pursue advanced studies in statistics. The summer program, entitled Data Visualization and its Role in the Practice of Statistics was funded primarily by the Institute for Pure and Applied Mathematics at UCLA. More information about it is available at http://summer.stat.ucla.edu. Hansen, Nolan and Temple Lang ran the Summer Statistics Program, where statisticians came for one or two days, bringing a research problem to work on with the 24 undergraduate participants. Past problems have been in areas such as bioinformatics, environmental science, health sciences, astronomy, imaging and engineering. For each research project, the students heard about the scientific problem and its importance; used R (R Development Core Team, 2006) to explore and visualize the researcher's data; presented their findings to the group; and discussed them with the researcher. We found that the students quickly mastered the computing and visualization tools provided. They were excited about how these tools enabled them to uncover the basic structure of the data to get to the statistical problems these data present and to get a sense of how statisticians approach large, complex problems. In general, the students were thrilled by their interactions with the researcher over his/her research.

The second experience was in designing a new course, required for statistics majors, on concepts in computing with data. In this course, students work on real problems where they perform large-scale exploratory data analyses using advanced data technologies and modern, computationally intensive, statistical methods. We have observed amazing transformations in our classes as students, who initially were unsure of their abilities in computing or otherwise reluctant to work with the computer, gained the confidence and skills to tackle a wide variety of data problems, and by the end of the course, were excited about the prospect of doing so.

In addition, our interactions with the researchers in preparing for the Summer Statistics Program has been a guiding example for how instructors might create this sort of experience for their students. In advance of the workshop, to assist the researchers in their presentations, we requested their data, articles on the subject, and a proposal for program activities. We then set about exploring, visualizing and analysing the data, discussing with the researchers what we found, what we didn't understand, and what alternative avenues might be worth presenting. Working together, we came up with an overall structure and approach for the student activities. The aim was to give students some guidelines to get started, to provide an opportunity for them

to explore avenues that they might find interesting, and to motivate the researcher's approach to the problem. It is this process/experience that we imagine an instructor wanting to emulate and reproduce for his students.

4 The Student Experience

Our goal for the students is for them to tackle the same sorts of problems as in the summer program and the computing-with-data course, where they receive guidance from the researcher without having to program each and every detail. That is, we want to avoid simplifying the scientific/data problems, and instead simplify how statisticians do what they do. As an example, we take one of the class projects, extract the essential elements, and present it in an alternative setting with guided analysis, exploration and free-form analysis that do not require the same level of programming by the student. The example concerns the problem of how to build a spam filter for e-mail. The raw data are about 9000 e-mail messages that have been 'hand' classified as spam or ham (regular e-mail). Apart from the data processing challenges, this project aims for students to:

- understand the connection between the raw data, i.e. an e-mail message with header, body, and appendices, and data that statisticians can analyse;
- find a good predictor, taking into account that a good predictor is not necessarily the same as an accurate estimator and that errors in misclassifying ham as spam are worse than misclassifying spam as ham;
- dig deeper to find out where a method works and where it fails, and intelligently apply a method to a new set of data;
- make meaningful comparisons of competing techniques.

We present here several ideas on how to provide this experience for students through a 'document'. The document starts by describing the problem and asking the student to think about what characteristics of a mail message indicate a message is spam, and also to think about what identifies a message as being ham. This brings her towards identifying data at hand that can be used to solve this problem, such as the sender of the message and the topic of the subject line. Next, the student is introduced to information found in the mail message whose existence she may not be aware of, such as the route the message took to get to the mailbox. This approach shows that there is more information than initially meets the eye and also that she must decide which information is useful. The document presents the raw data through an interactive interface that displays the e-mail messages in a single folder (Figure 1). Like a regular e-mail reader (e.g. Thunderbird, Outlook, pine or mutt) the interface presents the messages as an ordered list by time, with each record having a subject, sender and date. When the student selects a message, she sees the raw text of the entire message including header information. The header is displayed as a table of name-value pairs which emphasizes that this is not just text but structured information. This interface is embedded directly within the document.

The next section of the document moves to a more traditional display of data that consists of 30 variables computed for each of the 9000 messages. The student can introduce his own messages, and the variables will be automatically computed for that data. The document provides numerical summaries of the different variables and collections of graphs of some potentially interesting sets of variables (Figure 2). Ideally, with this visualization tool, the student connects the records in a data sheet with the points in a plot and with the individual mail messages. For example, when the student identifies a point in a scatter-plot by clicking on it, the corresponding row

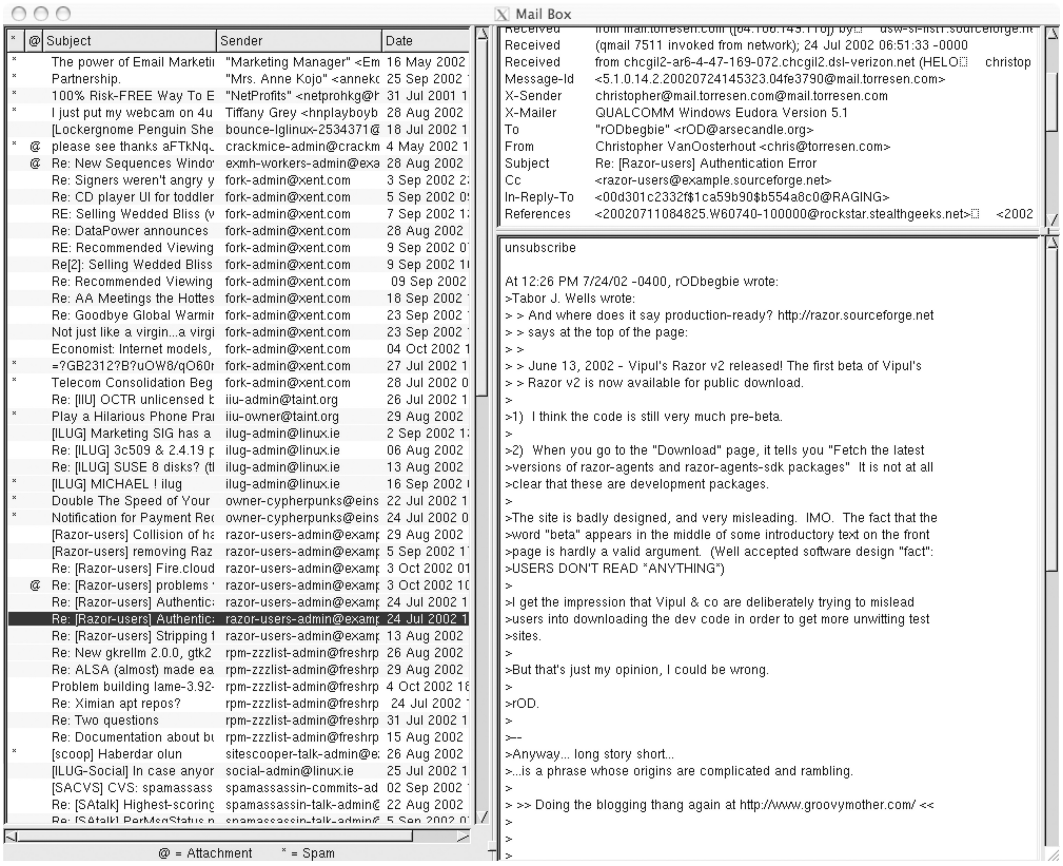

Figure 1. *The e-mail reader shown here has some of the basic functions found in regular e-mail readers such as Thunderbird and Outlook. For example, click on an e-mail message in the inbox (left-hand side) and the message is displayed to the right. Click on a column header, and the mail messages are sorted by that field, e.g. by subject, sender, date, or whether it is spam (∗) or has an attachment (@). In addition to the text of the message, the mailbox shows the message's complete header information in a table of name-value pairs (upper right-hand side). The mailbox entries can be linked to the rows in the more traditional display of the data in Figure 2.*

in the data sheet is displayed and the mail message is also displayed. Linked plots, e.g. with GGobi (Swayne *et al.*, 2007), iSPlot (Whalen, 2006) or iPlots (Urbanek & Wichtrey, 2007), would let the reader get a deeper sense of the data by exploring the multivariate nature of the relationship of these variables to spam and ham. The advanced reader can add, delete or modify the variables by providing an R function to compute the variable. This allows him to explore alternative measurements rather than sticking with what is presented.

The student is then introduced to the statistical methodology used to classify the messages, which in our example is a classification tree. The document has a link to a tutorial that introduces the concept of classification trees and then goes further into the details of using them. This includes aspects of pruning (Figure 3), cross-validation, surrogate splits and the code to fit and work with the trees. This material is not repeated in the document on spam filtering, but instead the document has a link to the tutorial so the reader can familiarize herself with the ideas. When the reader returns from the tutorial, she proceeds to fit the classification tree with the derived variables, and then explores how well it did. The document presents displays of the tree, the

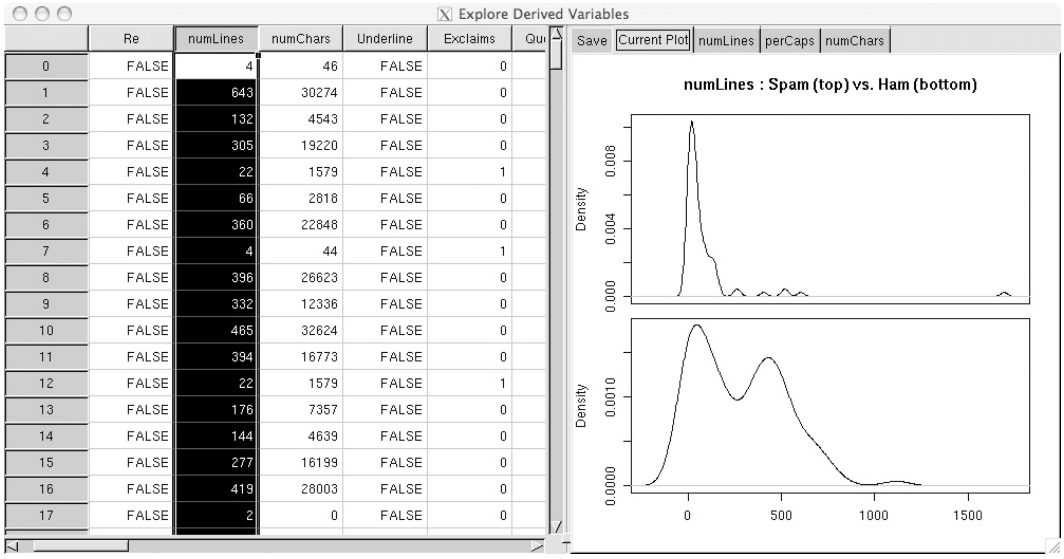

Figure 2. *This interface presents a specialized spreadsheet where the reader can explore the derived variables in two ways: by browsing the data in the grid format; and, by viewing plots of the data. The reader can save those plots which uncover structure in the data to create a notebook with pages of interesting plots.*

residuals and confusion matrix. The reader controls some of the parameters in the fitting of the tree, where the interactive components, such as the slider that controls the depth of the tree, are taken from the tutorial.

Having performed the diagnostics on the original data set, the student-reader is provided an entirely different set of messages to try on his classifier and see how it performs. The original document will have an analysis on a particular validation data set, and the reader can look at that analysis and read the comments of the author. If the instructor wants him to do more, a student might explore linked plots, tables and data sheets to try to understand where the classifier did well and where it did poorly. The instructor may also want the student to compare and contrast different classification methods, such as classification trees and k-nearest neighbor classifiers. The document presents a particular distance measure and value of k to use. However, the student can drill down into the document to examine other values that are contained in the document but not presented in the description of the analysis. The student explores these in parallel to the presented 'solution', and tries other combinations to come up with his own predictor. He then investigates the predictions for the validation set and the fits and misfits for his classifier. He investigates what makes these messages easy or hard to classify, for which observations his predictor disagrees with the one presented by the author, and importantly, why the classifiers perform differently on the original and the validation data sets.

With this type of document, students have a lot of flexibility to control the analysis and computations. In short, the document is a 'live' worksheet with which the student can interact to modify some of the inputs and computations. Importantly, they are not programming the computations. Rather, they are working with the embedded and hidden computations to explore different scenarios and approaches. They are 'doing what we do', but are not responsible for the computational details. Also, they are being driven by statistical reasoning rather than mathematical formulae and applications of rules. And yet, mathematics can appear in tutorials to help them understand different concepts and methods. Figure 4 provides a schematic for

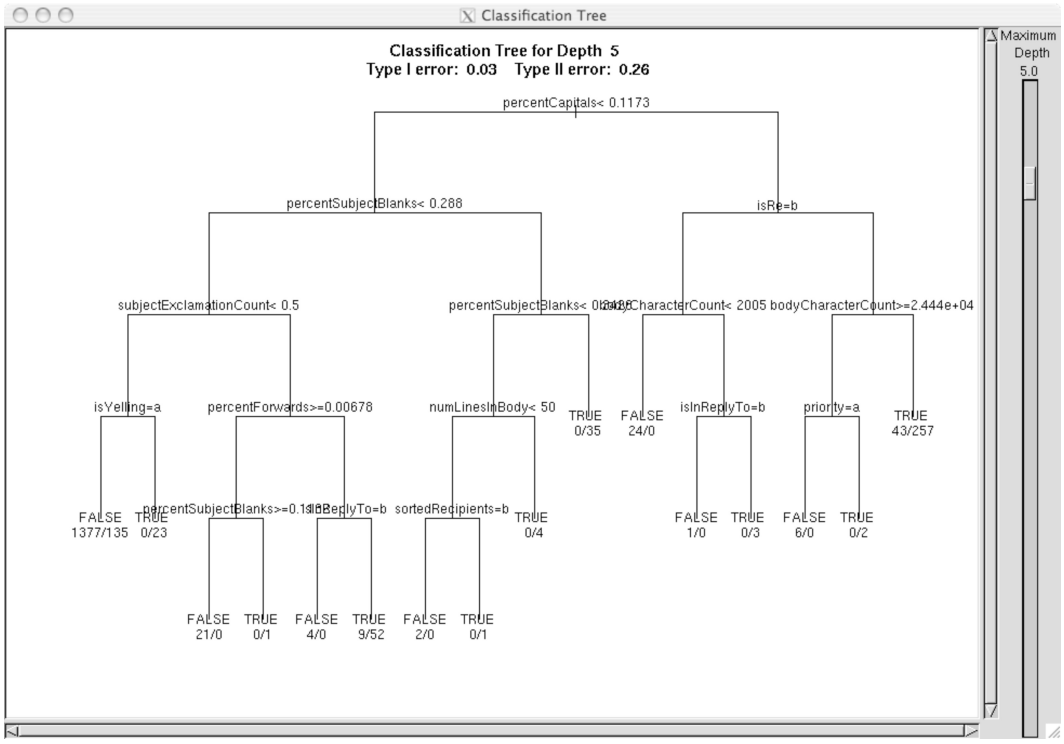

Figure 3. *The tutorial on classification trees allows the reader to explore various aspects of the statistical method. This particular graphical interface, allows the user to specify the depth of the tree by moving the slider on the right-hand-side of the window. The resulting classification tree and its Type I and II errors for misclassification are displayed in the graphics area. Note that we are using existing software in R for fitting and displaying these trees. This has the advantage that comes with building on other's work in that there is no need to write code for this complex procedure from scratch and new functionality can be readily incorporated.*

the control flow of this document. The ellipses denote specific computational tasks, such as loading the raw data, performing particular statistical tasks and tutorials. The rectangles represent interactive viewers and controls, such as sliders to change parameters in the tutorials or statistical tasks, or graphics devices to view the data. The ellipses and rectangles with dashed borders are optional tasks. That is, the instructor may choose to not present them to the reader, or the reader may choose to not follow the path to that computational task.

5 The Instructor Uses the Research Article

In order for the instructor to bring the statistical process of discovery described in Section 4 to her students, we imagine her engaging in an interaction similar to our encounters with the researchers in preparing for the Summer Statistics Program described in Section 3. To make the process more concrete, we take as an example the summer program presentation by Madigan on a problem related to location estimation in wireless networks. We have created a brief research article based on several articles on the topic (King *et al.*, 2006; Krishnan *et al.*, 2004; Madigan *et al.*, 2006; Youssef & Agrawala, 2004). Unlike a typical research article, this document includes the data, code from our analysis, and annotations from our discussions with Madigan (see Sections

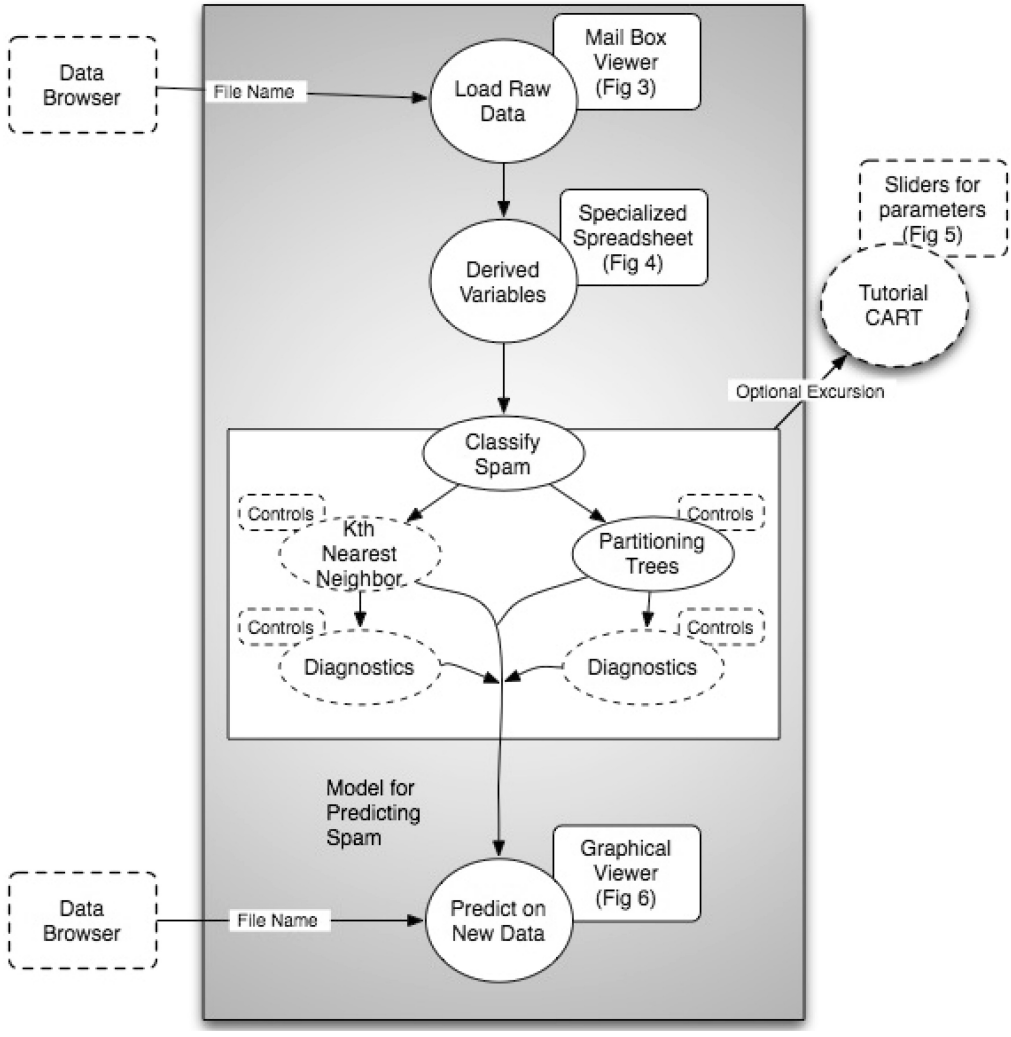

Figure 4. *A schematic for the computational flow of a sample document. Inputs for one task can be specified either in the document or interactively using different controls that are shown as options. Interactive controls are created for a particular view of the document. The outputs of one task become the inputs for the next task in the flow. Note the ellipses denote computational tasks, the rectangles represent interactive viewers and controls, and those shapes with dashed borders are optional tasks. We do not show the document text here, which would be interspersed with the tasks.*

7 and 8 for details about how such a document could be created). Using this document as a starting point, we began to think about how might we, as instructors, open up the researcher's analysis to students so they can explore and learn valuable lessons about how statisticians think and work. We provide several ideas as to how an instructor could augment a research article. (In what follows, 'author' refers to the researcher and original author of the article.)

- **Givens for the field.** An author might make a statement that is crucial to the analysis without necessarily checking it, as it is obvious to those in the field. An instructor, can have the student check and explore such statements so they better understand the scientific foundation of the problem before launching into the main data analysis. One example from our wireless data

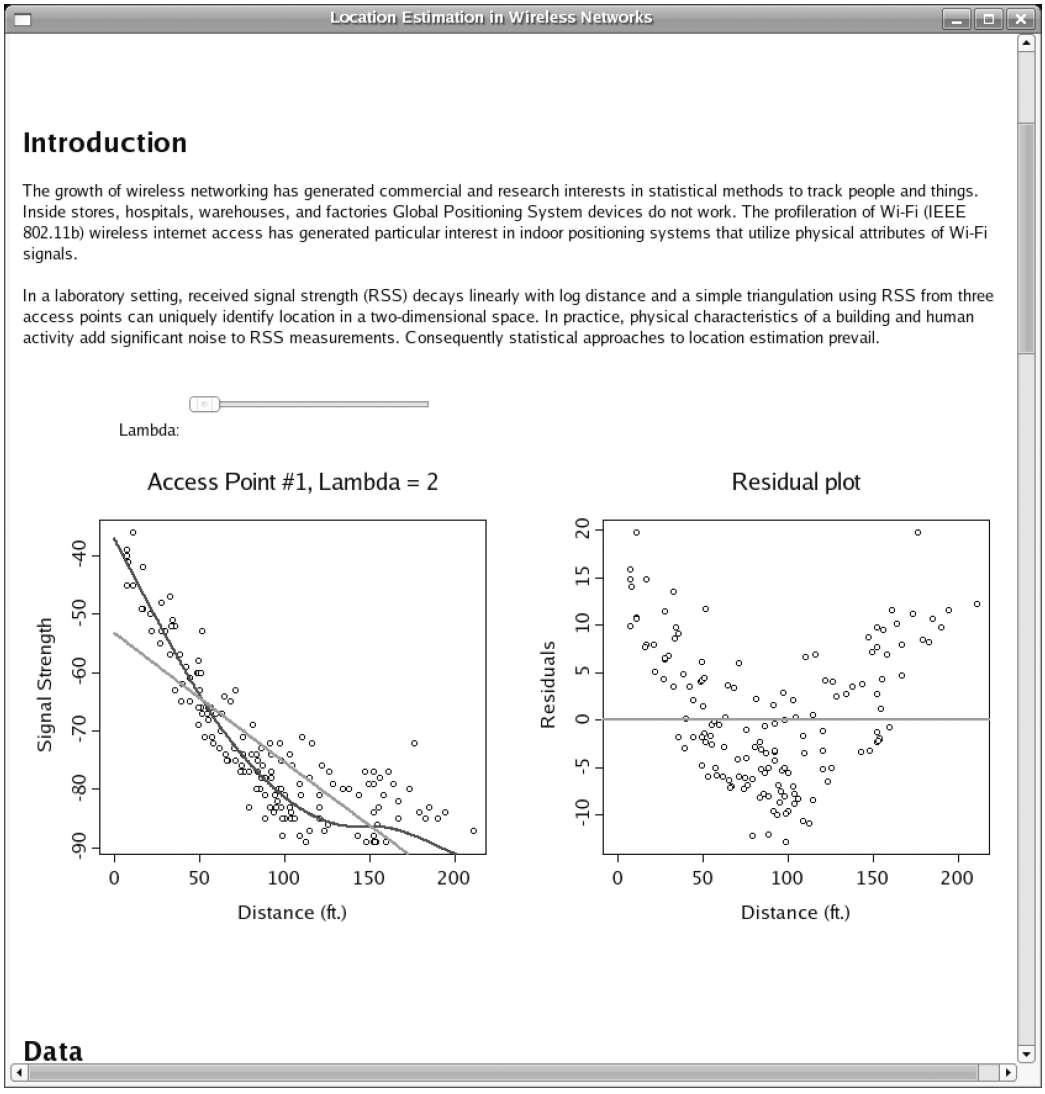

Figure 5. *In the introduction, the instructor has added a slider and linked plots so that the student can check on the author's statement that signal strength is linear in log distance.*

analysis is the mention that signal strength decays linearly with log distance. The instructor can provide the student an opportunity to explore this relationship via a simple plot and slider that fits a smooth curve to the data (Figure 5).

- **Dynamic imitation.** The author checks the data and assumptions extensively to understand how to proceed with an analysis, but the results of this stage of the analysis are often given cursory mention in the finished presentation. It is helpful for students to learn how these assumptions were checked. Students may gain insight into data analysis by imitating the author's work with similar explorations. Rather than having the student read/look through piles of plots and statistics, the instructor could give a portal into the process to rapidly redo pieces of this initial stage and go further. A simple starting point is the presentation of the data

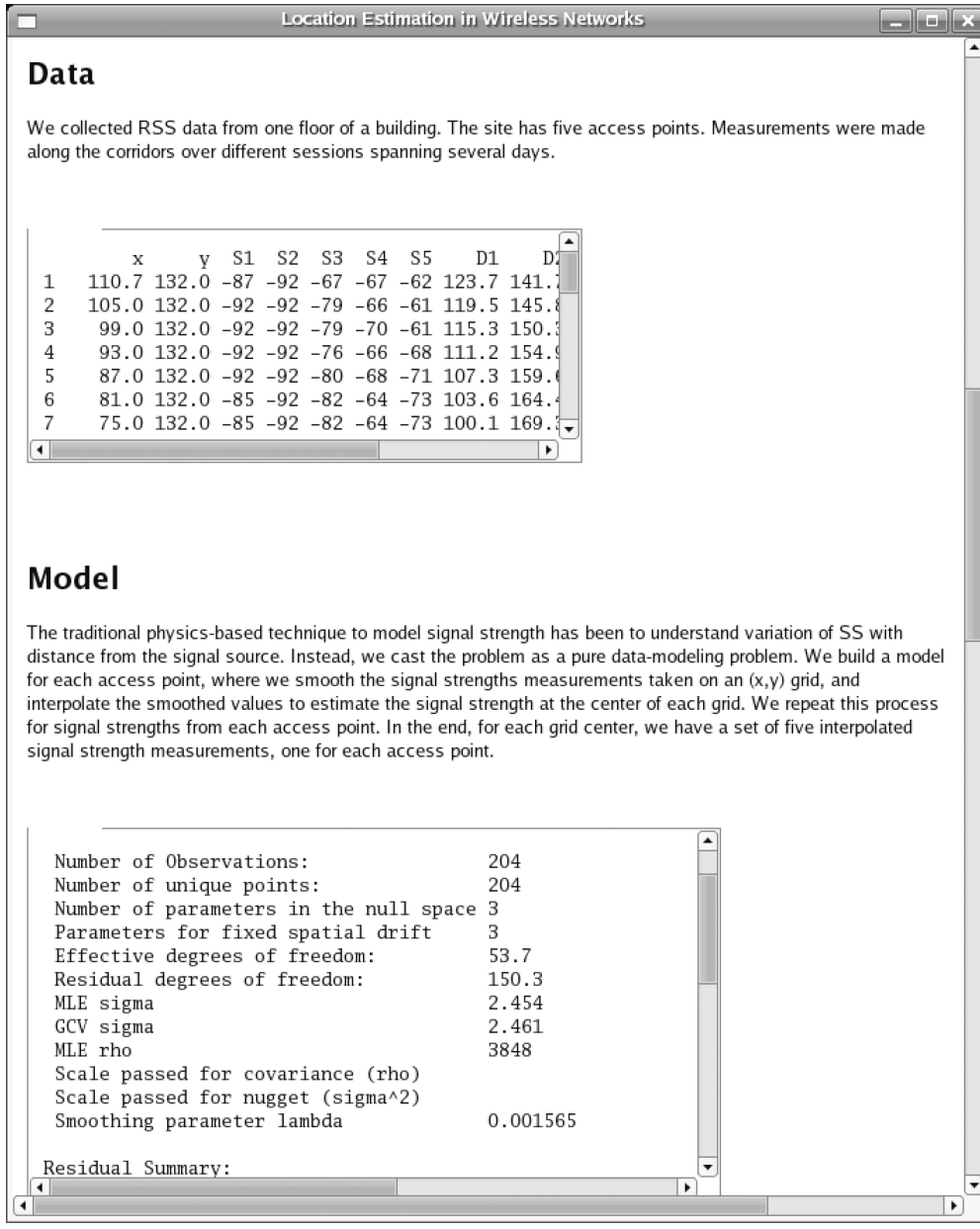

Figure 6. *A viewport for the data is automatically provided when the document is processed for interactive viewing. The instructor may wish to augment this simple display to include linked plots and the ability to create new variables. The output for the model fitting is also automatically provided for viewing, and a simple R input window or a more sophisticated collection of controls can be added for the student to explore other models.*

as in Figure 6, where by virtue of making the document interactive, this view of the data is provided 'automatically'.

- **Parallel avenues.** As the author conducts her analysis, she rules out different avenues or discovers avenues that are essentially the same as the approach she reports on in the article. It's important for students to see that the researcher didn't necessarily start in the direction

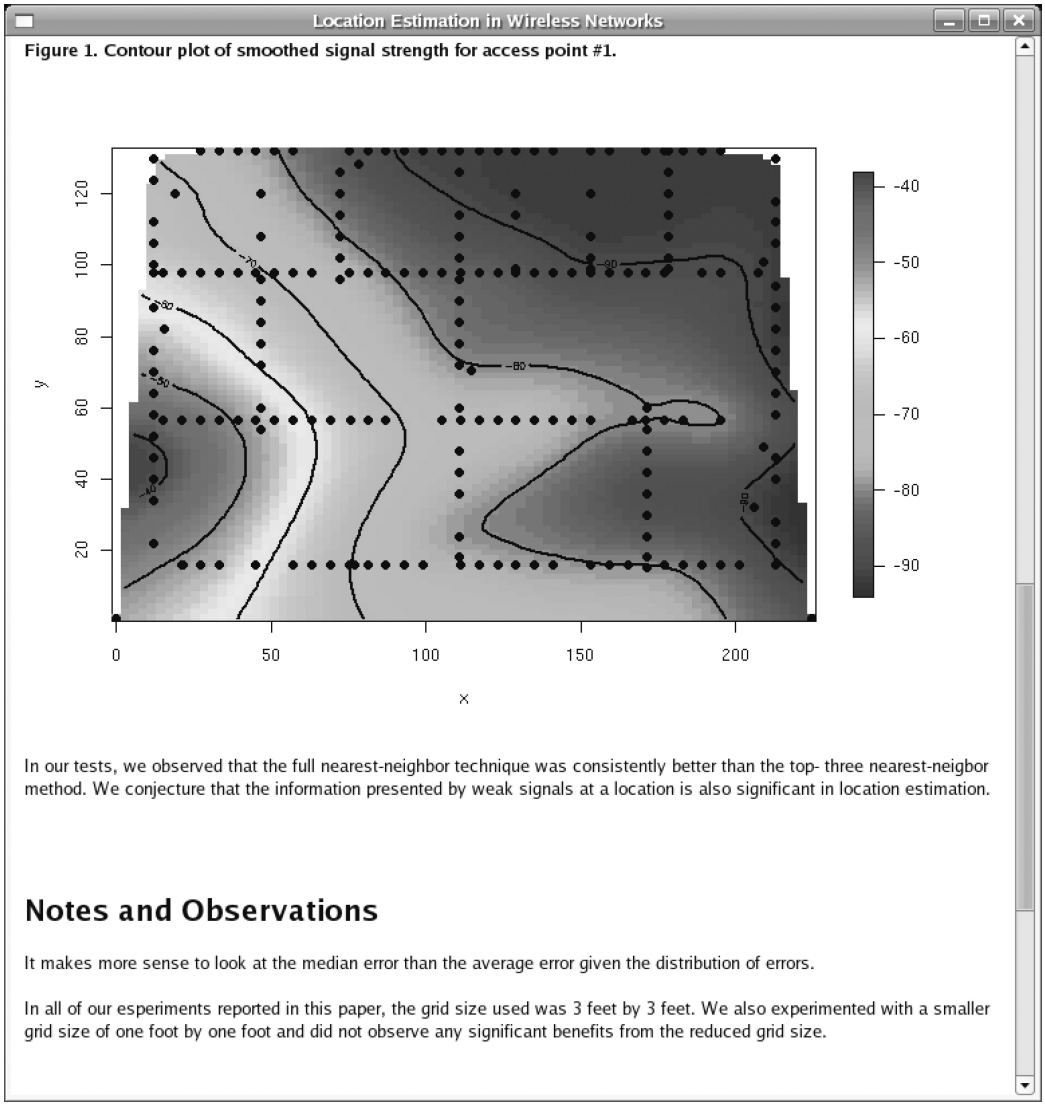

Figure 7. *Interactive documents lend themselves to alternative visualizations of the data, such as the one shown here, that the researcher would not normally consider including in an article.*

that appears in the final report and that she needed to check out other approaches before proceeding with confidence in this direction. An instructor may want students to: explore a related, possibly simpler, direction (as in Figure 7); critique and compare alternatives; find a 'next best' approach; figure out how to improve upon one of the approaches presented; or check them against a different set of data. Any of these can be accomplished by the instructor or the student building on the code that the author has supplied in the document. That is, the instructor and student can introduce their own code into the document.

- **Relevant tutorials.** An author might perform a more specialized case of a particular statistical method which the instructor might choose to focus on in greater detail. The instructor can fill

in the preliminary steps for the student who is learning about the method by providing, for example, a tutorial on the subject that uses the author's data. This approach can make learning the method more compelling and more easily understandable as it appears in a motivating context.

- **Pushing the boundaries.** The author can only present simulation results for a finite set of initial conditions. The instructor may ask the students to study different parameter settings, or augment the simulation. This could involve the instructor or student supplying their own code to better understand the behavior of the method under study.

In summary, intuition and experience in data analysis is somewhat of an art, and it is very hard to teach. Perhaps that is why we don't do so more. Another reason is that there are few good examples from which to work. At the heart of this problem is the missing link between statistics educators and researchers. In our preparation for the Summer Statistics Program, we were in the role of educators and we worked with the researchers to help prepare the data analysis activities. This experience has guided our development of these documents and our vision of how these documents can be created by researchers and made available to educators.

6 Synergistic Development

Our goal is to provide a prototype of infrastructure that would enable researchers and consultants to easily make their problems available to educators in a useful form. The main thrust of our position is that we need active statisticians contributing case studies to the education community, and to facilitate this flow between the two groups we need the cooperation to be cost-free in terms of time. The approach we outline in Sections 7 and 8 proposes a programming and authoring environment designed for professional statisticians that supports communication of statistical results and the data analysis process. Thus educators can take documents written by researchers and enhance them for students by, for example, adding controls to explore the computations or contrasting alternative approaches side-by-side within the context of the document. To make this remotely feasible, authors of these data-analysis studies must also be able to easily provide information about the entire thought process and the analyses they performed to get to the final conclusion. They must be easily able to include the computations and the results that lead them through the process to a conclusion. Not all of these steps need to be shown to the reader, but they should be available so that instructors can access, use, and augment what was actually done, not just what was reported.

In our experience with the Tools for an Interactive Learning Environment project (TILE was funded in part by the National Science Foundation and conducted between 1995 and 2001) we found it too costly to rebuild statistical methods in a new environment, e.g. Java for the Web. For example, a large Java-based project requires Java programmers knowledgeable in statistics, numerical analysis, graphics, etc. to develop adequate statistical algorithms, and as a result wastes resources re-implementing algorithms that have already been well-implemented in existing statistical programming languages. In addition, such a project needs leaders knowledgeable in software development and management of software projects. Alternatively, projects must have limited scope and ambition. For example, students can be hired to program small disparate applets or JavaScript. Unfortunately, these activities typically can neither be maintained by the owner nor tailored by another instructor.

The collaborative model underlying the development of open-source software can overcome many of these problems in the development of pedagogical content. It provides

- facilities that allow us to build pedagogical applications on existing, well-tested, well-designed code;
- a community of users to contribute and collaborate on projects;
- quality control through potential peer review, both formal and through use.

The R project offers an excellent illustration of this opportunity within the statistics community. As much as the R environment has provided high quality software to the statistics profession, perhaps more importantly, it has shown the incredible benefit of a community of similarly focused people developing a very large and comprehensive library of packages covering a vast array of statistical methodology. The CRAN, BioConductor and Omegahat repositories combined contain approximately 1000 contributed packages. Building from StatLib, R has fostered a spirit where researchers and developers contribute useful software as add-ons to the base system to provide enormous value.

We believe that good infrastructure for statistical tutorials, research, and pedagogy can have the same passive accumulation effect. By leveraging many researchers, some of whom also teach, we have the opportunity to build up a large collection of data analysis problems, tutorials about statistical methods and glossaries of terms that no individual group could hope to create.

7 The Interactive, Dynamic Document

The document displayed in Figures 5–7 is both interactive and dynamic. By dynamic we mean that the code for the analysis and plots are contained in the document and this code is run to create the view of the document shown in the figures. The document is interactive in that the reader interacts with, for example, the slider by dragging the slider thumb. When this happens, code is re-evaluated and the updated output is displayed. The infrastructure that we have already implemented provides a prototype for how to build and view such documents. It is based upon the R computational environment. We describe here how an instructor can use this infrastructure to view, modify, and write a document.

7.1 The Document in XML

The document in Section 5 is a collection of text, code, and output, and is written in XML—the eXtensible Markup Language. XML syntax is similar to HTML having elements or nodes of the hierarchical form:

```
<element attribute="value" ...>
  <child attr="...">...</child>
</element>
```

However, XML differs from HTML in that we have created our own names for the elements rather than being restricted to a specific set. In this sense, HTML is a particular XML grammar. We have chosen Docbook (Walsh & Muellner, 1999) as the foundation for the grammar for these technical documents. Docbook provides a rich vocabulary for technical documents, including the usual constructs such as article, section, title, paragraph, table, hyper-link and so on. It has been used to publish entire books and a vast number of articles. To Docbook, we have added special XML elements for designating R code, expressions, and variables, as well as elements for interactive components. For example, to include a slider in a document, such as the one shown

in Figure 5, the instructor would use the following markup.

```
<i:slider var="lambda" min="2" max="100"/>
```

In this slider tag, the attribute 'var' specifies that the slider is to be connected to the R variable `lambda` and the attributes 'min' and 'max' denote the range of possible values for `lambda`, i.e. from 2 to 100.

Below is a snippet of the XML file, *wirelessArticle.xml*, for the document displayed in Section 5. Note that the *r:plot* element contains the code to produce the plot, i.e. the plot is produced *dynamically* by this code. Further note that the plot depends on the spline fitted to the data using the nuisance parameter `lambda`. When the reader *interacts* with the slider, the input value is assigned to `lambda`, the associated R variable; the relevant code is re-evaluated with the new value; and the output is displayed. In this case, the document updates the spline fit and re-plots the figure.

```
<r:code id="spline" i:display="false">
spl = smooth.spline(w$D1, w$S1, df = lambda)
pred = predict(spl, x)
</r:code>

<table>
  <tr>
    <td align = "center">
    Lambda:
  <interactive ref="setVars">
       <i:slider var="lambda" min="2" max="100"/>
  </interactive>
    </td>
    <td> </td>
  </tr>
  <tr>
    <td>
      <r:plot i:update='true'>
      plot(S1 ~ D1, data = w[w$S1 > -90,],
          xlab = "Distance (ft.)", ylab = "Signal Strength")
      lines(predAuthor, col="blue", lwd = 2)
      lines(pred, col="green", lwd = 2)
    </r:plot>
  </td>
...
</table>

<section>
  <title> Data</title>
  <para>
We collected RSS data from one floor of a building. The site has
five access points. Measurements were made along the corridors over
different sessions spanning several days.
  </para>
</section>
```

7.2 Viewing the Document

The R package **IDynDocs** (Temple Lang & Nolan, 2007) provides the functionality to transform and display the XML documents. To view the wireless XML file, *wirelessArticle.xml*, the interactive tags are replaced by interactive controls, the R code in the document is run, the output is inserted into the transformed document, and the output, controls, and text are arranged in the viewer. The viewer is a special HTML browser. A simple call to the function `viewIDoc()` in **IDynDocs** performs these actions.

```
viewIDoc("wirelessArticle.xml")
```

The **IDynDocs** package depends on several other R packages, which are automatically installed with **IDynDocs**. These are the R packages: **RwxWidgets** (Temple Lang, 2007b), **RwxDevice** (Temple Lang, 2006a), **Sxslt** (Temple Lang, 2007c), and **XML** (Temple Lang, 2006b).

The interactive controls and the HTML widget/browser are provided by a general-purpose Graphical User Interface (GUI) toolkit called wxWidgets. The packages **RwxWidgets** provides functionality to allow programmers to use the facilities in the wxWidgets GUI toolkit and the **RwxDevice** allows us to use regular R graphics anywhere within wxWidgets-based GUIs. The interactive controls are created via R functions provided in **IDynDocs** that interface to the GUI toolkit. The wxWidgets GUI toolkit, the Docbook grammar, and a third party XML transformer must all be installed as part of these dependencies. For the most part, these software packages are reasonably simple to install and will be done invisibly by R or require one-time, relatively simple commands to install the third party libraries. For example, the binaries for wxWidgets will accompany the Windows version of **RwxWidgets**, install easily on the Mac using macports (`www.macports.org`), and are available for most versions of Linux using one's favorite package manager. On some platforms, third party software will also need to be installed, i.e. wxWidgets.

7.3 Modifying the Document

An instructor can modify the file *wirelessArticle.xml* to add additional interactive components and code, or other branches of exploration. As a simple example, the instructor might want to add to the article's introduction (Figure 5) the capability to toggle between plots of signal strength against distance on a linear or a log scale. To begin, she adds the following interactive component to the XML file.

```
<i:checkbox var="logscale" label="Use Log Distance"/>
```

This results in a checkbox with the label 'Use Log Distance' being added to the display (see Figure 8). When the checkbox is checked or unchecked, the variable `logscale` is assigned TRUE or FALSE, respectively. **IDyndocs** handles this re-assignment of the logical variable, and the instructor simply adds the R code to make the appropriate plots according to the value of `logscale`.

Specifically, the instructor augments the code for the spline fitting and plot as shown below. Note that the variable *lpred* contains the predicted values for the splines based on log distance.

```
<r:code id="spline" i:display="false">
spl = smooth.spline(w$D1, w$S1, df = lambda)
predlin = predict(spl, x)
lspl = smooth.spline(log(1+ w$D1), w$S1, df = lambda)
predlog = predict(lspl, x)
```

```
</r:code>
...
<r:plot i:update='true'>
if (!logscale) {
  ls = ""
  predA = predAuthor
  pred = predlin
}
else {
  ls = "x"
  predA = lpredAuthor
  pred = predlog
}
xlab = paste("Distance (ft.)", " on log scale"[logscale], sep="")
plot(S1~ (1+D1), data = w[w$S1 > -90, ], log=ls,
  xlab = xlab, ylab = "Signal Strength")
lines(x, predA$y, col="blue", lwd = 2)
lines(x, pred$y, col="green", lwd = 2)
</r:plot>
```

With these small changes to *wirelessArticle.xml*, the instructor has modified the document to add more interactive controls. Ideally, we would have an authoring system that would facilitate the instructor (and original author) in editing the document. For example, the system could have a facility for representing the document as a tree that is expanded and collapsed for viewing and editing bits of code as needed. Ideas for how to build such a system are discussed in greater detail in Section 8.

7.4 Transforming the Document

Figure 9 shows a schematic for how the functionality in **IDynDocs** works, i.e. the background process involved in transforming XML content into a format for viewing. This is invisible to the reader and the author of a dynamic, interactive document but we describe it here for completeness and also to illustrate that it is easily extended and adapted.

An XML document can be turned into a variety of formats, e.g. PDF, HTML, and what we call 'interactive HTML'. We have used XSL, the eXtensible Stylesheet Language, to make this transformation. Essentially, XSL allows us to specify rules for transforming XML elements into a target format. We use it directly from within R with the **Sxslt** package (Temple Lang, 2007c), which allows us to have regular XSL rules that might also call R functions to evaluate the computations in the document.

Sxslt processes the code elements in the document, using dynamic XSL stylesheets that we have provided. These dynamic XSL stylesheets generate HTML, or FO (Formatting Objects) or LaTeX either of which can be further processed to generate PDF. Additional processing is needed to create interactive documents. **IDynDocs** uses the dynamic stylesheets to transform the XML into HTML, and then converts the inline interactive elements, such as "i:slider", into HTML OBJECT elements within the new document. The resulting document is then displayed using the special HTML viewer described earlier.

The HTML widget parses the HTML content and lays it out. As it encounters each OBJECT element in the HTML, it calls an R function that creates the appropriate GUI widget. Support for common controls such as sliders, buttons, checkboxes, radio buttons, combo boxes, etc.

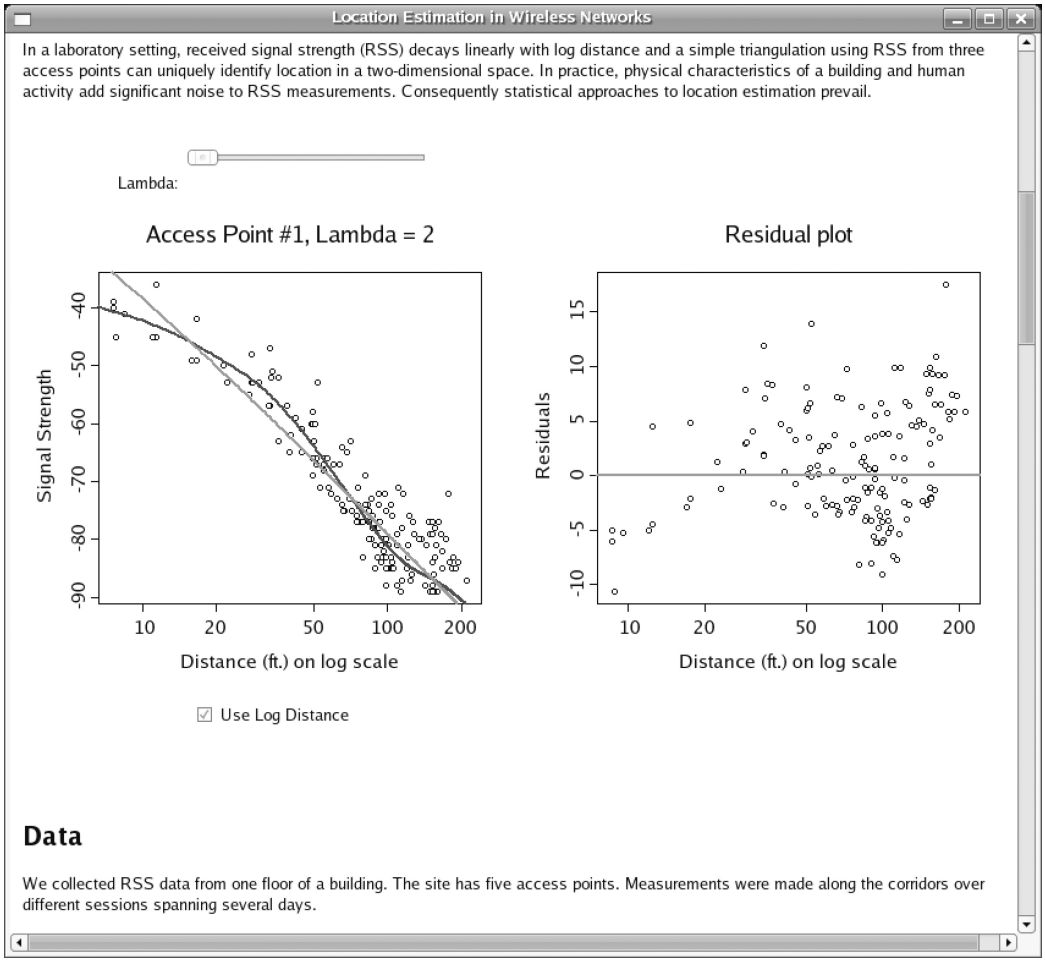

Figure 8. *Another instructor can easily update the document to add a check box to the introductory display in Figure 5 (below the left-hand plot) so the student can toggle back and forth between views of signal strength against distance on a linear or log scale.*

is provided through the **IDynDocs** package (Temple Lang & Nolan, 2007). This allows the instructor to easily insert such controls (Figure 8), and she can insert composite widgets, which she or another instructor has created, that combines several widgets to create richer, more complex interactions (Figure 2). Further, the instructor can supply her own R function for special, more flexible handling of the user input—handling new OBJECT types or customizing the response to user actions. Also, regular graphical output from R can be displayed within the dynamic, interactive document. The output can have a static format, such as JPEG or PNG (e.g. Figure 7), or it can be 'live' or dynamic. The dynamic output is created by embedding an **RwxDevice** graphics device within the HTML widget whose contents can be updated based on reader interactions (e.g. Figure 5).

A more complete description of the infrastructure, available and planned, and a discussion of the pros and cons of the future possibilities appear in Section 8.

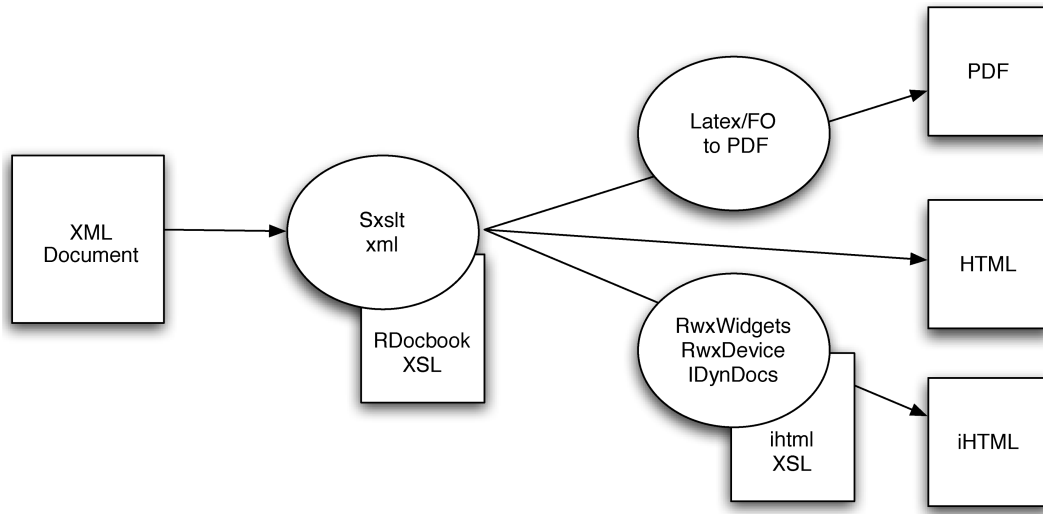

Figure 9. *A schematic of the control flow for processing a document for viewing. The original XML document is processed in R using the XSLT processor and Docbook stylesheet that we have extended for these purposes. This processing occurs using the R packages* ***Sxslt*** *and* ***XML****. The output could be HTML, or FO and LaTeX, which can be further processed to generate PDF. To create an interactive document, we use the GUI toolkit wxWidgets to create GUI controls and components. The toolkit has been made available from within R via the packages* ***RwxWidgets*** *and* ***RwxDevice****. We use the wxWidgets HTML viewer (also available via* ***RwxWidgets****) to display the transformed document. Inlined OBJECT elements within a HTML document are processed by user-level R functions that create the appropriate interactive control and connect its actions to evaluating the relevant code within the document and updating the display.*

8 An Overview of the Technological Infrastructure

The prototype described through the example shown in Sections 5 and 7 involves researchers, data analysts, instructors and others creating documents that contain detailed descriptions of what they have done in their work at various levels. The author or instructor can then add controls to allow the student to explore and control the analyses and computations. Students can explore the computations, add new ones, and organize them in a 'lab book'. And there must be tools to assist the authors in all these steps, along with facilities for readers to navigate and interact with the document in various different ways.

The system seems ambitious, and it is. When these concepts were germinating in our thoughts around 1996, it was evident that the technology was emerging that would make them feasible. Few could have predicted how things would have emerged, but the confluence of developments have left us in a very good position. Since 1998, Temple Lang has been working on a variety of infrastructural developments that make much of the authoring and reading tools outlined here possible within the short term.

The infrastructure that we have implemented and most of what we propose for the future is built upon the R environment. This is because it is widely used by practicing statisticians and researchers and has a broad collection of existing statistical methods. It is quite common that authors of data analysis studies will have done them using R or that they are readily adapted to use the functionality in R. Importantly, R is a general purpose programming language which allows the student to grow statistically from using GUIs to expressing their own computations. Also, cutting-edge methodology is often developed first in R and immediately made available to the statistical community. However, there is very little within our infrastructure that cannot be replicated within other environments such as S-Plus, Python, Perl or Matlab. The intersystem

interface packages for R available from the Omegahat project (Temple Lang, 2007a) even allow us to work with documents containing code written using multiple languages.

In what follows, we describe the current technological state of the proposal, map out future directions, and discuss the reasons for proceeding along these paths. We invite others to join in the development of the tools that would enable these new ideas and paradigms to take place.

8.1 The Format of the Document

The most natural choice in the current technological climate for a format for representing technically oriented, dynamic, interactive documents is XML and Docbook. From a technical point of view, we have chosen XML over, say LaTeX, because XML is more structured and makes it easy to process content for many purposes. From a social perspective, XML is one of the 'hottest' information technologies around at present and is increasingly widely used. All of the major Office suites (i.e. Microsoft Office, Gnome Office and KOffice) use XML as one of their regular formats for all of their applications. And data representation and exchange is done by many communities using customized XML grammars, e.g. Graph eXchange Language (GXL), Geographic Markup Language (GML), Systems Biology Markup Language (SBML) and many Web sites that disseminate large collections of data provide the content in XML. Sophisticated and flexible graphical displays can be done in XML using Scalable Vector Graphics (SVG). Mathematical content can be included using MathML. Good Web browsers now provide support for rendering both SVG and MathML. XML is also used to implement Web services via an approach termed SOAP—Simple Object Access Protocol. And XML is the 'X' in AJAX—Asynchronous JavaScript and XML—which is an important technology underlying dynamic Web pages. We should further note that XML and LaTeX are not mutually exclusive. We can use XML to separate the text which can be formatted using LaTeX from the code elements within the document. Additionally, there are tools to transform Docbook XML documents to LaTeX.

As if all of the widespread use was not enough to make a compelling case for why we have chosen to use XML, the extensive availability of tools to manipulate XML is quite attractive. There are tools to parse XML documents in all common programming languages, including R via the **XML** package from the Omegahat project (Temple Lang, 2007a). The XPath language provides a rich form of expression for identifying one or more nodes in a document tree that have particular characteristics, e.g. relationships to other nodes and/or certain attributes or values. Using XSL we have been able to easily specify rules for transforming XML elements or groups of elements into a target format. XSL tools are available in all widely used programming languages and we use these directly from within R to transform XML documents to different views. Via the **Sxslt** package (Temple Lang, 2007c) for R, we can use XSL in the usual way, but we also allow the XSL rules to call R functions, e.g. to evaluate the computations in a document. In addition to these primary technologies and standards, there is a collection of XML-related technologies such as XPointer, XInclude and XLink that both illustrate the momentum of XML and how comprehensive its design and development has been.

The **XML** and **Sxslt** packages in R give us the infrastructure and expertise to easily manage the different stages of the document creation, transformation and rendering. Other XML-related packages, e.g. **SSOAP** (Temple Lang, 2006b), and general distributed computing, intersystem/language interfaces also provide us with useful facilities for performing the complete collection of computations we may need in different documents.

With the functionality provided in these packages, authors can easily generate a fixed/static 'final' version of their work that combines text, computations and output. The format of these static documents is PDF or HTML. The markup of the original documents consists of two

intermingled types: the Docbook related elements containing the text of the document, and the code-related elements that contain the R code to evaluate. By using Docbook's markup for the text, we gain from a widely used and well documented vocabulary, and we make use of the XSL stylesheets that come with the Docbook software to transform the document to HTML, XHTML or FO (Formatting Objects), the last of which can then be processed into PDF via, e.g. the Java application FOP from the Apache project. As the name implies, XSL is extensible and our XSL stylesheets extend the standard Docbook rules to include additional rules for processing and rendering the code-related elements. By leveraging the Docbook infrastructure, we continue to inherit any improvements in their tools in the future and yet we still have the flexibility to do what we want.

To create a regular document displaying the text and the code and not the results of the computations, the author applies one of our XSL stylesheets using any XSL transformation programs such as xsltproc. To evaluate the R code elements, R is needed and to that end, we use the XSL transformer from libxslt (which underlies xsltproc) embedded within R. To process the code elements in the document, we use a slightly different set of XSL rules which call back to R functions to evaluate the code and obtain the results. We provide these dynamic XSL stylesheets for generating HTML or FO and PDF.

The combination of **Sxslt** and these XSL stylesheets provide equivalent facilities as Sweave (Leisch, 2002) but is more flexible as it uses an XML format that can accommodate new markup with more than two types of content (text and code) and which can support LaTeX and other text formatting systems. Moreover, the framework can be extended via XSL or R and the documents can be processed in rich ways, allowing them to be used as 'databases' of information.

8.2 Displaying the Document

To view the document, we transform the original XML document to a specific format. Of course, if this is one of the standard formats such as PDF, then the reader simply uses one of the viewers for the resulting document's format, e.g. Adobe Acrobat for PDF. The interesting case arises when the document has interactive controls within an HTML format as in the examples provided in this paper.

Rather than using a Web browser, we prefer a different approach. We use an HTML viewer available in a general-purpose GUI toolkit. We have used both Gtk—the Gnome toolkit—and wxWidgets, a multi-platform native GUI toolkit. We plan to explore both the Qt toolkit and Mozilla framework. The key point is that such a toolkit provides a way to display an HTML document, and it allows us to customize how interactive controls can be displayed within the document.

To do this, we transform the original XML document into HTML and convert the inline elements, such as i:slider, into HTML OBJECT elements within the new document. We then pass this document to the HTML widget which parses the HTML content and lays it out. As it encounters the OBJECT elements, it calls our R functions that process these general OBJECT elements and embed the GUI components into the document. Both the gtkhtml and wxHtmlWindow widgets allow us to provide functions in R that perform this processing. The interactive controls are created via R functions that interface to the GUI toolkit, e.g. the R package **RwxWidgets** provides an R interface to the wxWidgets. We provide support for creating the common widgets and arbitrary GUI components.

RwxWidgets allows us to insert into the document simple widgets, such as buttons, sliders, and data grids, graphics devices (via the **RwxDevice** (Temple Lang, 2006a) or **gtkDevice** (Drake *et al.*, 2005) packages), the code and its output, and composite widgets that combine several

widgets to create richer, more complex interaction. In many cases, an interactive control will be quite simple. It will be a single widget that is connected to the value of a variable as in the slider example in Section 7. In order to facilitate this common style of interactive component, the R package **IDyndocs** supports markup of the form

```
<i:control var="varName" ...>
```

The package provides XSL rules and R functionality for several commonly used interactive controls. That is, it provides the basic functionality to handle the user interaction and update that widget and others within the document to create a flexible interface mixing text, graphics and interactive controls. Additionally, the author can customize these by specifying R code within the interactive XML elements.

Overall, this approach has several advantages: it reduces the number of languages involved relative to a browser-based approach and allows the smoother progression from interactive controls within a document to full-fledged graphical user interfaces. Since the author's computations within the document are typically in R (or can be sent to other interpreters via R's intersystem interface packages) and the XML document can be processed in R (directly and with XSL), it makes sense to control the displaying of the document also in R. However, it seems natural to use a Web browser to interact with HTML documents. Buttons, menus, etc. are provided via HTML forms. Sliders and other interactive components can be provided using a combination of Java and JavaScript. As the user controls the computations via the interactive elements, the R code must be evaluated and the new output displayed. An approach to creating the connection between R and the Web browser is to provide R as a browser plug-in, similar to Java or Flash. We did this in 2000 for the Netscape browser and may implement the same facilities for the Firefox browser in the future or alternatively integrate the Firefox rendering engine into R. This would give us support for Cascading Style Sheets used to customize HTML displays, and JavaScript used for rich effects within an HTML document. Also, we would gain facilities for rendering Scalable Vector Graphics (SVG) and give some interactivity within inlined R graphics displays. In short, it would give us access to a widely used set of facilities that are part of an ongoing, leading Web browser project.

On Windows, another alternative is to use the R-DCOM facilities (available at www.omegahat.org) to establish R as a compute server. We can use R to control the Web browser and to create controls and register R functions to handle user inputs and events. Alternatively, we can use R as a compute server in much the same way as a browser plugin. In either case, the user events cause the computations to be sent to R and the output is displayed by R in the web browser.

8.3 Word Processors, Editors and Authoring Tools

Of course, the first stage in the life cycle of one of these dynamic, interactive documents is the author writing the text and doing the computations for the analysis. And this is where the myriad of different styles of human computer interaction and writing of documents raises becomes problematic. Some authors of these documents will be familiar with LaTeX and will use an editor such as Emacs, vi/vim, bbedit or one of several others. Others would prefer to use a word processor such as Microsoft Word or OpenOffice. To provide the end-user interface for all of these requires more resources than we have. However, the choice of XML, Docbook and XSL does provide us with facilities that interact well with modern suites of office applications, e.g. Microsoft Office, and make transforming from one type of document to another feasible.

The approach we currently use for writing these types of documents is to use Emacs. We have developed customizations of the nxml mode for Emacs for editing XML documents with

knowledge of our added markup elements, e.g. r:code, i:slider. These additions provide keystroke bindings for inserting code elements within the document and sending the contents of a code element to an R process, much like Emacs Speaks Statistics (ESS) (Rossini *et al.*, 2004). We plan to add facilities for inserting the output from R directly into an r:output node in the XML document. Additionally, our extensions to the Docbook RNG schema allow for 'on the fly' validation of the document as it is being written and completion for what elements and attributes are allowed at the current input position. Currently, the author writes text in the document and then creates a code element and inserts the R code within that. Sometimes, the author works in R and then cut-and-pastes the code from the R session into the XML document. In other cases, he writes the R code directly within the XML element and sends it to R to be evaluated. All of these approaches are supported, and we plan to create an outline mode that will enable the author to see the large scale structure of the document or focus on all of the details and content.

As the author progresses linearly through the document, adding text and code, things are relatively simple. However, we additionally envisage that the author will explore different approaches to an analysis or simulation. Some of these will be dead-ends and others will be paths that are tangential to the main parts of the paper. The XML document will support the concept of a branch or path, including large segments of content within a branch node. When there are two or more possible approaches being considered for the same end, e.g. the use of different classification methodologies to create a classifier, these will be parallel branches and we want the reader to be able to see them as such and work with them as alternatives. Each branch is essentially a mini-document made up of text, code and interactive elements and any other content that may be relevant, e.g. data. It is connected to the other parts of the document in that it can access the earlier inputs and results. Furthermore, we can add markup for parallel branches to identify their inputs and outputs much like type specification for functions. This will facilitate readers switching between them and introducing new ones.

In addition to working with the document in an editor such as Emacs, it will be helpful to view non-trivial documents that have branches and paths by looking at a collapsed view of this hierarchical structure displaying the alternative paths. This allows the author to see the structure and to drag code elements, text or entire branches to other parts of the document and to view particular parts in the interactive HTML browser. This is similar to an outline mode but it would also be used by both author and reader and is more interactive and allows for active creation of the document rather than just viewing the existing structure.

Since word processing applications, and specifically Microsoft Word, are so widely used, we have thought about how we might leverage that familiar interface to write these documents. The **R-DCOM** tools (Temple Lang, 2005), developed as part of the Omegahat project, allows us to control Word from within R and to treat R as a computational server from within Word, Excel, etc. This bi-directional interface is richer than other R-DCOM bridges as it allows us to not only create from within R interactive controls that appear within the Word interface and document, but additionally to register R functions to respond to user actions within these controls and also generally for events on the document and application, e.g. creating a new document or selecting a region. We think that we can leverage this infrastructure to create an environment in which the R and Word sessions work in tandem, sending code from Word to R and R inserting results and controls into the document in response. And, we can create toolbar items for adding markup for, e.g. r:code, r:plot and interactive components such as i:slider and i:datagrid.

Since Microsoft Office now has quite extensive support for XML and XSL transformations, we expect that this R-Word link will be a convenient interface for authoring these dynamic, interactive documents for many who are already familiar with R. And it will be possible to render the interactive documents directly in Word, again simplifying the interface for the reader. Moreover, as inputs are changed and computations update, the code can be re-evaluated in R

(via the event handling) and the results inserted into the document to present the up-to-date view for the author or the reader.

Other tools such as SciViews, Relax and Rkward exist for creating documents in conjunction with R sessions and we hope others will explore these.

9 Conclusions

We have described an approach to creating and reading research documents that capture what the author actually did, including the false starts and what would be unreported results. The document acts as a sort of catch-all database for the author's work and allows different elements to be readily extracted and used in other computations by author, instructor, reader, and student alike. The format supports transforming the document in different ways for different audiences and to different formats. Importantly, this format is highly extensible and supports the inclusion of markup to create interactive controls for the computations. The reader can directly manipulate the computations using these GUI controls *while* she is reading the document and explore alternative choices and approaches. We provide an HTML browser in R that supports the creation and rendering of these interactive, dynamic documents, and the reader need only install the software once and then browse the documents with a single command.

The above describes the technical, extensible framework on which we can format and render these interactive, dynamic documents. However, it is the non-technical aspects that we think are equal in importance. We hope that researchers will gradually adopt this as it provides a flexible notebook from which they can generate their final documents and capture computations, and as these research documents become available, other researchers and students can read them in greater detail than a regular paper. Most importantly for us, instructors can add interactive controls and annotate these documents so that students can explore the material in an active, hands-on manner. We believe that statistics education is in dire need of exposing students to real scientific and social problems in which statistics plays a role. These documents will allow readers to see what statisticians do on a day to day basis and how. As the collection of these dynamic documents grows, we hope that they can be combined into a library and provide a flow from researchers to instructors which will enhance modern statistical education.

There is more work to be done on integrating this approach with the commonly used authoring tools. However, the framework we have proposed and developed provides a rich and relatively complete foundation for this work. We hope that it will encourage the sharing of documents and so the exchange of examples of actual statistical practice and provide an alternative to method-first/only pedagogy. The approach provides benefits for researchers, instructors and readers, and for this reason there is some chance it will be adopted.

Acknowledgement

The authors were supported in part by NSF grants DUE-0127557 and DUE-0618865.

Glossary of Technical Terms

HTML The Hypertext Markup Language used in creating Web page content.

XML The eXtensible Markup Language is a modern and simpler version of SGML and is similar in style to HTML. It allows new dialects or grammars to be specified as it permits new elements to be defined. Classes of documents, i.e. the grammars, can be formally defined using XML schema or the older Document Type Definitions (DTDs).

XSL The eXtensible Stylesheet Language is a particular grammar for XML that is used to specify rules for transforming XML content to other formats and outputs such as PDF. XSL-FO (formatting objects) is the common mechanism for generating PDF from XML content.

SVG Scalable Vector Graphics is the XML dialect for describing Postscript- or PDF-style graphics. SVG also supports interactivity and animation.

Docbook DocBook is a dialect of grammar of XML for marking up technical documents. DocBook uses presentation-neutral markup, i.e. it describes the content not the form of a document element. In this way a document written using Docbook markup can be published in various formats.

Microsoft Office Microsoft's suite of applications, which includes the word processor Word, the spreadsheet application Excel, and the presentation program PowerPoint.

Gnome Office The office suite for the GNOME desktop which is part of the GNU project. The Gnome office applications are free software, and include the word processor AbiWord and the spreadsheet application Gnumeric.

KOffice The free office suite for KDE, the K Desktop Environment. KOffice includes the word processor KWord, spreadsheet program KSpread, and presentation application KPresenter.

OpenOffice An open-source office suite that includes a word processor that often goes by the same monikar.

emacs The name Emacs is derived from Editor MACroS. Emacs is a text editor that has extensive feature sets of macros.

ESS Emacs Speaks Statistics (ESS) is an emacs package that provides a mode for editing statistical languages, including R. With ESS, the statistician can edit R code in one emacs buffer, and execute that code in another buffer.

LaTeX A type-setting system for authoring articles, books and generally documents. Based on Knuth's TeX, it excels at handling mathematical content.

FO The Formatting Object is a markup language for XML document formatting which is used to generate PDF. Unlike HTML, it has no semantic markup and it stores all of the document's data within itself.

PDF The Portable Document Format is an open standard for a file format created for device-independent and display-independent document exchange.

R A high-level statistical programming language and interactive data analysis environment. It is an Open Source implementation of the S language, and similar to the commercially available S-Plus.

JavaScript JavaScript is a scripting language that is used for client-side web development. JavaScript is unrelated to Java.

Java Java is an object-oriented programming language that is similar in syntax to C++. Java is platform independent.

Gtk A toolkit for creating graphical user interfaces. This underlies the Gnome desktop and numerous professional applications such as the Gimp, Gnumeric, AbiWord.

wxWidgets A toolkit for creating graphical user interfaces. wxWidgets runs on all platforms and uses the native widgets of the operating system, e.g. Carbon on Mac OS X, and Gtk on Linux.

GGobi An interactive, dynamic graphics system for data visualization. Built on Gtk, it can be readily used directly within R via the Rggobi package.

Firefox A modern, open-source Web browser that is gaining significant popularity.

DCOM Distributed Component Object Model from Microsoft that allows for client-server communication in a language-independent machine across machines. This is very similar to CORBA, the Common Object Request Broker Architecture, but is Microsoft-specific.

References

Cobb, G.W. & Moore, D.S. (1997). Mathematics, statistics, and teaching. *Am. Math. Mon.*, **104**, 9, 801–823.

De Veaux, R.D., Velleman, P.F. & Bock, D.E. (2007). *Stats: Data and Models*, 2nd ed. Boston: Addison Wesley.

Drake, L., Plummer, M. & Temple Lang, D. (2005). gtkDevice: Loadable and embeddable Gtk device driver for r. `http://cran.r-project.org/src/contrib`. R package version 1.9-4.

Freedman, D., Pisani, R. & Purves, R. (1998). *Statistics*, 3rd ed. New York: W. W. Norton & Company.

King, T., Kopf, S., Haenselmann, T., Lubberger, C. & Effelsberg, W. (2006). Compass: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, pp. 34–40. New York: ACM Press.

Krishnan, P., Krishnakumar, A.S., Ju, W.-H., Mallows, C. & Ganu, S. (2004). A system for lease: Location estimation assisted by stationery emitters for indoor rf wireless networks. In *Proceedings IEEE INFOCOM 2004,* the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong-Kong, China, March 7–11. IEEE 2004. http://www.informatil.initrier.de/~ley/db/conf/infocom/infocom2004.html#KrishnanKJG04.

Leisch, F. (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. In *Compstat 2002—Proceedings in Computational Statistics*, Eds. W. Härdle & B. Rönz, pp. 575–580. Heidelberg, Germany: Physika Verlag, ISBN 3-7908-1517-9.

Madigan, D., Ju, W.-H., Krishnan, P., Krishnakumar, A. & Zorych, I. (2006). Location estimation in wireless letworks: A bayesian approach. *Stat. Sin.*, **16**, 495–522.

Moore, D.S. (1997). New pedagogy and new content: The case of statistics. *Int. Stat. Rev.*, **65**, 2, 123–165.

Moore, D.S. (2005). Quality and relevance in the first statistics course. *Int. Stat. Rev.*, **73**, 2, 205–206.

Moore, D.S., Cobb, G.W., Garfield, J. & Meeker, W.Q. (1995). Statistics education fin de siècle. *Am. Sta.*, **49**, 3, 250–260.

Nolan, D. & Speed, T. (2000). *Stat Labs: Mathematical Statistics Through Applications*. Springer-Verlag New York: Springer Texts in Statistics.

Peck, R., Casella, G., Cobb, G.W., Nolan, D., Starbuck, R. & Stern, H., Eds. (2004). *Statistics: A Guide to the Unknown*, 4th ed. Belmont: Duxbury Press.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, ISBN 3-900051-07-0.

Ramsey, F. & Schafer, D. (2002). *The Statistical Sleuth*, 2nd ed. Belmont: Duxbury Press.

Rossini, A.J., Heiberger, R.M., Sparapani, R.A., Mchler, M. & Hornik, K. (2004). Emacs speaks statistics: A multiplatform, multipackage development environment for statistical analysis. *J. Comput. Graph. Stat.*, **13**, 1, 247–261.

Rossman, A. & Chance, B. (2003). Notes from JSM 2003 Mathematical Statistics session: Is the Math. Stat. course obsolete? http://www.amstat.org/sections/educ/SFPanelFinal.doc.

Scheaffer, R.L., Gnanadesikan, M., Watkins, A. & Witmer, J.A. (1996). *Activity-Based Statistics*, 2nd ed. Emeryville: Key Curriculum Press.

Swayne, D., Cook, D., Temple Lang, D. & Buja, A. (2007). The GGobi data visualization system. http://www.ggobi.org/.

Temple Lang, D. (2005). RDCOM bundle of packages. http://www.omegahat.org/RDCOMBundle.

Temple Lang, D. (2006a). RwxDevice: R graphics device using wxwidgets. http://www.omegahat.org/RwxDevice. R package version 0.2-1.

Temple Lang, D. (2006b). SSOAP: Client-side SOAP access for S. http://www.omegahat.org/SSOAP/. R package version 0.4-3.

Temple Lang, D. (2006c). An XML package for the S language. http://www.omegahat.org/RSXML/.

Temple Lang, D. (2007a). The omegahat project. http://www.omegahat.org.

Temple Lang, D. (2007b). RwxWidgets: Interface to wxwidgets. http://www.omegahat.org/RwxWidgets. R package version 0.5-6.

Temple Lang, D. (2007c). The Sxslt package. http://www.omegahat.org/Sxslt/.

Temple Lang, D. & Nolan, D. (2007). IDynDocs: Interactive dynamic documents in R. http://www.statdocs.org.

Urbanek, S. & Wichtrey, T. (2007). iplots: iplots - interactive graphics for r. http://rosuda.org/iPlots/. R package version 1.0-7.

Utts, J. (1999). *Seeing Through Statistics*, 2nd ed. Belmont: Duxbury Press.

Utts, J. & Heckard, R. (2003). *Mind on Statistics*, 2nd ed. Belmont: Duxbury Press.

Walsh, N. & Muellner, L. (1999). *DocBook: The Definitive Guide*. Sebastopol: O'Reilly & Associates, Inc. See also http://www.docbook.org.

Watkins, A., Scheaffer, R.L. & Cobb, G.W. (2003). *Statistics in Action: Understanding a World of Data*. Emeryville: Key Curriculum Press.

Whalen, E. (2006). iSPlot: Interactive Scatter Plots. http://www.bioconductor.org/packages/1.9/bioc/ html/iSPlot.html. R package version 1.8-0.

Wild, C. & Pfannkuch, M. (1999). Statistical thinking in empirical enquiry (with discussion). *Int. Stat. Rev.*, **67**, 3, 223–265.

Wild, C. & Pfannkuch, M. (2004). Towards an understanding of statistical thinking. In *The Challenge of Developing Statistical Literacy, Reasoning & Thinking*, Eds. D. Ben-Zvi & J. Garfield, chapter 1, pp. 17–46. New York: Kluwer Academic Publishers.

Youssef, M. & Agrawala, A. (2004). On the optimality of wlan location determination systems. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*.

[*Received June 2006, accepted September 2007*]