DYNAMIC, INTERACTIVE DOCUMENTS FOR TEACHING STATISTICAL PRACTICE

Deborah Nolan
University of California - Berkeley, United States
Duncan Temple Lang
University of California- Davis, United States
nolan@stat.berkeley.edu

Along with many others, we propose that statistical thinking and literacy are the important elements to teach rather than rules and methods. We propose that this is true for all levels of statistics education. We further argue that we must teach our students how statistics can be used to answer scientific questions and how to connect relevant statistical methods to these questions. We outline an approach that allows authors to create documents describing data analyses and tutorials that combine the description of the analysis with the computations performed along with the thought process. These documents can contain the different branches of exploration which the author pursued, along with the more traditional distilled approach. The document can also contain interactive controls that allow the reader to modify the computations. The documents are thus dynamic (outputs can be recalculated), interactive (controlled by reader) and contain the thought process of the author and also methods for reproducing and exploring the results.

INTRODUCTION

Leading statistics educators call for reforming the way we teach our introductory courses to place greater focus on statistical thinking and data analysis, and less on derivations and computational recipes (see Cobb and Moore, 1997, and Moore, 1997, 2005). In response, new introductory textbooks now include case studies to motivate statistical topics and activities for students to help convey basic concepts. Examples include Ramsey and Schafer (2002), Utts and Heckard (2003), Schaeffer *et al.* (1996), Utts (1999) and Watkins *et al.* (2003). We applaud these and others efforts to change the way statistics is taught by revealing the basic principles in statistics and the power that statistics has to address important issues in science. Yet, we have not gone far enough in this direction. And, in some respects, we are not necessarily positioning ourselves to be able to sustain even these initial directions in the future in terms of new applications and methodologies.

We fundamentally view statistics as a *science of data*, and we need to *rebalance statistical education* from this vantage point, where statistical concepts flow from contextual problem solving with data. It is ironic that many of us use the term “applications” to refer to examples of applying a particular method to some data. We feel that the process should be entirely in reverse - a scientific application calls for the selection and use of statistical methods. Of course we must illustrate methodology with examples by applying it to data, but this shows that our teaching is more in the spirit of tutorials on statistical procedures than the statistical process itself. We further contend that reform must extend beyond our introductory courses to the entire curriculum, both undergraduate and graduate.

Our attempts to address these problems have led us toward a novel design for interactive, dynamic, electronic documents that support the synthesis of theory and practice and employs multiple approaches to convey statistical concepts. We describe this design and provide an example of such an interactive document. Along with this example, we discuss the technical infrastructure that is in place to produce and interact with it.

STATISTICS AND THE SCIENTIFIC QUESTION

While many courses teach methodology, either the theory or the application, very few focus primarily on teaching the skills of approaching a scientific problem from a statistical perspective. Even case studies hide much of the thought process that is actually needed for real analyses. An analysis is typically presented as a completed work, but the thought process that led to the conclusions and approaches is usually omitted. There is rarely mention of alternative approaches that were not fruitful or that were not pursued for various reasons. We also do not typically identify alternative approaches that led, or would lead, to almost equivalent solutions to the one we present in the final analysis. For those learning data analysis and statistics, the intuition and experience that are used in data analysis are the hardest things to learn, and less often taught.

Intuition and experience in data analysis is somewhat of an art, and it is very hard 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 and data analysts. Instructors outside the realm of statistical consulting and research have great difficulty finding new, real, interesting scientific problems with accompanying data to use in their classes and have little exposure to the use of modern statistical methods in the scientific context.

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 and effort. The approach we outline later in the paper proposes a programming and authoring environment designed for professional statisticians that supports communication of statistical results and the data analysis process at all levels including students projects, consultant reports, topic tutorials and research papers. 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 "case" or data analysis studies must be able to easily provide information about the entire thought process and the analyses they performed to get to the final conclusion. The collaborative model of Open Source software development, and specifically of *R* within the statistical community, can enable this to happen. 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 over 700 packages for different tasks. 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 who 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.

AN EXAMPLE

We outline an example of one of these documents. We will not present the entire document but merely indicate the different sections and how the author and student/reader might interact with the document. The topic is spam - unsolicited email - and approaches for using statistics to classify a message as spam or the alternative, ham.

We start by describing the problem and asking the students to think about what characteristics of a mail message indicates to them that a message is spam. And we ask them also to think about what identifies a message as being ham. This brings them 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.

We then introduce information found in the mail message whose existence they may not be aware of such as the route the message took to get to the mailbox, or the application used to compose the message. We illustrate the anatomy of an email message made up of the header information (like an annotated envelope) and the body along with any attachments which have a similar header and body structure. We feel that it is important to connect the student to the raw data. It shows them that there is more information than initially comes to mind and also that they must decide which are useful. To show them these data, we present an interactive interface as a collection of email messages in a single folder. Like a regular email reader (e.g., *Thunderbird, Outlook*, *pine* or *mutt*) we present the messages as an ordered list by time, with each record having a subject, sender and date. When the reader selects a message, she sees the raw text of the entire message including header information, etc. We display the header as a table of name-value pairs, emphasizing that this is not just text, but structured information. This interface is embedded directly within the document. It is a “live” email reader application that allows the student to explore the data as they would interact with their own email.

We move to a more traditional dataset consisting of 30 variables computed for each of the 9000 messages in our dataset. (Note, the students can introduce their own messages at any point in time and the variables will be automatically computed for that data as the document contains the workflow and computational framework.) We provide numerical summaries of the different variables and collections of graphs of some potentially interesting sets of variables. We provide a visual tool to allow the student to connect the records in a data sheet with the points in a plot and with the individual mail messages. For example, when the students identifies a point in a scatter-plot by clicking on it, the corresponding row in the data sheet is displayed and the mail message is also displayed. The corresponding parts of the text in the mail message are highlighted to show how the values of the variables displayed in the plot were computed. For example, for the hour-sent variable giving the hour of the day the message was sent, a click on a point in the plot would highlight the Date entry in the header of the message and perhaps color the hour value in the entry. Identifying the presence of spam-related words in the subject would be done by highlighting those words in the subject entry of the header. Showing the absence of a feature (e.g., that the message is not in response to an earlier message via the absence of Re: at the start of the subject) is more difficult. Interesting ways to illustrate such “non-features” are needed.

We use linked plots (using *GGobi*, *iSPlot* or *iplots*) to explore the data. These plots are displayed within insets within the document, surrounded by the text. The goal is to let the reader get a sense of the data and to explore the multivariate nature of the relationship to spam and ham. Advanced readers can add, delete or modify the variables by providing an *R* function to compute the variable for each message. This allows her to explore alternative measurements rather than sticking with what the author has presented.

Next we move to introduce statistical methodology that can be used to classify the messages. We start with classification trees. 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, cross-validation, surrogate splits, and the code to fit and work with the trees. The tutorial is another of these dynamic, interactive documents written in the same manner, but here focused on statistical methodology rather than a particular analysis. We do not need to repeat this material in our document, but have a link to it so that the reader can familiarize herself with the ideas. (We can insist that they go and work through the tutorial before continuing, but that again is an option for different readers that can be controlled when generating the view from the original document and/or when initially displaying it to the reader.)

When the reader returns from the tutorial, we proceed to fit the classification tree for our messages and derived variables, and then explore how well it did. We present displays of the tree, the residuals, and confusion matrix to see how it performed. We let the reader control some of the parameters in the fitting of the tree. For example, we let them control the maximum depth of the tree by using a slider to specify a value between 1 and 30. A plot of the resulting tree is displayed beside the vertical slider and the different plots and tables for the fit are updated. The component containing the slider and plotting of the tree are taken from the tutorial. In this way, we reuse the elements and code from other documents. We have to connect the changes to our other elements and some assistance is provided for specifying this for the author.

Having performed the diagnostics on the original dataset, we then focus on using an entirely different set of messages to try our classifier and see how it performs. The original document will have an analysis on a particular validation dataset. The reader can look at that analysis and understand the comments of the author. For students or curious readers, we want them to do more. The reader can use the dataset we provide for this purpose or, preferably, bring in their own. We explore linked plots, tables and data sheets to try to understand where the classifier did well and where it did poorly. The students should try to find some underlying patterns in the cases where the classifier under-performed on the new data.

Just as we try to understand why a classification method does well or poorly on a validation set, we also compare and contrast two different classification methods. In our example, we use classification trees and *k*-nearest neighbor classification. Again, we defer the introduction of the concepts and details for *k*-NN to a tutorial. For these data, we have to select the distance measure and value of *k* to use. While the author proposes the final selected combination, the document contains but does not display information about other combinations. The student can drill down and examine these in parallel to the presented "solution." We can do this as part of the content the author provides, or alternatively provide an inset that provides a convenient interface for the reader to select different combinations.

If the author includes the results for the different combinations, the reader can explore them in either way without having to recompute the values. However, they can also perform the computations if they wish or with new data.

Having fit the two models (a tree and *k*-NN) in the preceding parts of the documents, we explore the predictions for the validation set and the fits and misfits for the two classifiers with respect to the predictor variables. We investigate what makes these messages easy or hard to classify, for which observations do the two sets disagree, and importantly, why do the classifiers perform differently on the original and the validation datasets. In addition to exploring potential over-fitting, we compare the distributions of the two datasets to see if they are the same.

With this type of document, the students have a lot of flexibility to control the analysis and computations with which they are working. They can change the depth of the tree or introduce new data. They can even change the variables used within the analysis. For example, some of the derived variables have thresholds associated with their "definition." For instance, we have a binary variable indicating whether the subject line of the email has any words from a set associated with spam. We can define this to be true if there are more than *m* words, where *m* can be changed to create new variables. For interested students, we allow them to vary *m* via a slider and to follow the effects of this modified variable on the subsequent computations and results in the document. Additionally, the student can go back to the top of the document and introduce new data to the entire problem, or introduce it as the validation set.

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 at the level of the analysis to explore different scenarios and approaches. They are "doing what we do," but are not responsible for the details. Also, they are being driven by statistical reasoning rather than mathematical formulae and application of rules. And yet mathematics can appear in tutorials to help them understand different concepts and methods.

TECHNOLOGICAL INFRASTRUCTURE

As we have mentioned above, we are building on top of the *R* environment. This is because it is widely used by professional statisticians 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. And cutting-edge methodology is often developed first in *R* and immediately made available to the statistical community via the *R* package mechanism. Importantly, *R* is a general purpose, high-level programming language which allows us and the students to express statistical computations relatively easily and which allows both groups to progress from simple calculations to the full software development. Additionally, the Omegahat project provides another repository of *R* packages with an emphasis on infrastructure and integrating systems. Over the past six years, the Omegahat project has developed a number of extensions to *R* and other programs that will be used within this project.

The most natural choice in the current technological climate for a format for representing dynamic, interactive documents is XML - the eXtensible Markup Language. XML is a markup language for representing arbitrary data content in a standard manner. It is a general version of HTML in that it uses the common HTML-like syntax to identify nodes within the markup tree. XML differs markedly from HTML in that we can use our own names for the elements rather than being restricted to a particular set. In this sense, HTML is a particular XML grammar. XML is very widely used in many contexts, and it is used as the primary representation for all data within the modern office tools (i.e., word processors and spreadsheets).

Many tools are available to manipulate XML. There are tools to parse XML documents in all common programming languages including *R* via the XML package from Omegahat. There is also flexible software that is used to transform XML documents to different formats such as HTML, PDF and raw text. This is XSL - the eXtensible Stylesheet Language. This allows us to specify rules for transforming XML elements or groups of elements in whatever way we choose. Again, we can use this directly from within *R* to transform XML documents to different views of our document.

To view the document we transform the original XML document to a specific format. It is natural to use a Web browser to interact with the HTML document because of the possibility of using interactive controls, e.g., buttons, menus, etc. are provided via HTML forms. We have explored this via the *SNetscape* package for *R*. However, rather than using a Web browser, we prefer a different approach. Since the computations are typically in *R* (or can be sent to other interpreters via *R*'s inter-system interface packages) and the XML document can be processed in *R*, it makes sense to control the displaying of the document also in *R*. Using the *RGtkHTML* package, we can display HTML documents within GUIs that we create entirely within *R*. This also allows us to have complete control over the HTML content and the events such as processing of links (hyper-links, images), form submissions, etc. The interactive controls such as sliders and other non-HTML elements come directly via the general *RGtk* package for developing professional GUIs using *R* code. This makes it easy to insert arbitrary composite GUI elements within a document just as we would in a regular, stand-alone GUI window. Again, we provide a library of such interactive components that can be easily included in documents.

In addition to the interactive controls, we can also include live graphics displays in the HTML viewer. The *gtkDevice* by Drake, Plummer and Temple Lang allows us to use any Gtk drawing widget as a regular *R* graphics device and so we can display state-of-the-art, sophisticated graphical displays within these documents and update them separately from the display of the document. *GGobi* is a visualization tool for linked, dynamic graphics for exploratory high-dimensional data analysis. Not coincidentally, *GGobi* uses gtk as the underlying GUI toolkit and is tightly connected to *R* via the *Rggobi* package. This allows us to put linked *GGobi* displays within our document view. And the extensible nature of the *Rggobi* package allows programmers to easily add new forms of linking between *GGobi* displays and other parts of the document or other visual components. The *iSPlot* is another simpler but powerful linking facility for *R* that uses the *gtkDevice* package and so is immediately compatible with our framework.

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. To create a document, we envisage the following setup. There will be an *R* session where the author issues different commands which are collected into a tree structure and displayed. The nodes of the tree represent different sequential steps in the analysis. Within each node, there may be alternative computations or branches in the analysis. And within each of these nodes, there can be multiple commands that make up a single "action." A path in the tree represents a particular analysis from beginning to end. The author can connect the analysis with text by dragging it from the tree into a word processor. The code for the action is hidden within the document unless the author chooses to view it in that mode. Also, a template for the output of the action is created such as a table or a plot. This can be customized using a dialog that allows the author to control the way the output appears or to specify an *R* command to create the desired output. To add interactive controls or generally include material from the library of existing content, the author brings up another tool that provides access to these components that have been created by us and other authors. These tools can then be dragged into the document.

Temple Lang has developed *R* packages that provide a general way to connect *R* and applications such as *Word*, and *Excel*. These are the *DCOM* packages for *R* from the Omegahat repository. They allow *R* to communicate with Word and to find out all about the documents it is working on and to modify them from within *R*. This allows us to insert the action along with its code and other components using pure *R* code. This high-level programming interface from within *R* makes it relatively easy for us to create various advanced interfaces for the author to manage the content within the documents she is preparing. In addition to being a useful interface for authoring these documents, the R-Word connection can also be used to display the interactive, dynamic document to the reader.

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.
- *Excel* – Microsoft's spreadsheet application that is part of the Office suite.
- *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*.
- Gtk - A toolkit for creating graphical user interfaces. This underlies the Gnome desktop and numerous professional applications such as the *Gimp*, *Gnumeric*, *AbiWord*.
- *Ggobi* - An interactive, dynamic graphics system for data visualization. Built on Gtk, it can be readily used directly within *R* via the *Rggobi* package.
- *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. and Moore, D. (1997). Mathematics, statistics, and teaching. *The American Math Monthly,* 104, 801-823.

Moore, D. (1997). New pedagogy and new content: The case of statistics. *International Statistical Review,* 65, 123-165.

Moore, D. (2005). Quality and relevance in the first statistics course. *International Statistical Review,* 73, 205-206.

Ramsey, F. and Schafer. D. (2002). *The Statistical Sleuth*. Belmont, CA: Duxbury Press.

Scheaffer, R., Gnanadesikan, M., Watkins, A., and Witmer, J. (1996). *Activity-Based Statistics*. Emeryville, CA: Key Curriculum Press.

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

Utts, J. and Heckard, R. (2003). *Mind on Statistics* (2nd edition). Belmont, CA: Duxbury Press.

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

WEB REFERENCES

Drake, C. and Temple Lang, D. *R DCOM package*, www.omegahat.org

Drake, C. and Plummer, M. and Temple Lang, D. *gtkDevice package* http://cran.r-project.org

The Omegahat Project for Statistical Computing www.omegahat.org

The R Project for Statistical Computing www.r-project.org

Swayne, D. and Temple Lang, D. and Buja, A. and Cook, D. *Rggobi package* www.ggobi.org

Temple Lang, D. *RGtk package* www.omegahat.org

Temple Lang, D. *RGtkHTML package* www.omegahat.org

Temple Lang, D. *SNetscape package* www.omegahat.org

Whalen, E. *iSPlot package* www.bioconductor.org

Uranek, S. and Theus, M. *iplots package* www.rosuda.org/R