The UCLA Statistics Textbook and Modules

Jan de Leeuw
UCLA Department of Statistics
Box 951554
Los Angeles, California 90095-1554

Abstract

We review the problems or organization and implementation involved in the construction of an electronic statistics textbook on the World Wide Web. Solutions currently chosen by the authors of the UCLA Statistics Textbook are discussed.

1 Introduction

The *UCLA Electronic Statistics Textbook* is an attempt to write a statistics textbook which is

1. freely available to everyone on the Internet;
2. independent of the level of the student, i.e. useful at the undergraduate, graduate, and postdoc level;
3. interactive, using graphics and demos;
4. complete, i.e. it covers most of statistical theory as traditionally taught.

We use the model of a classical statistics textbook to bring together hypertext describing the traditional concepts and techniques of statistics and graphical components illustrating these concepts and techniques. But we also use alternative forms of organization of the same material into modules, and into pages with calculators.

This particular mix, except for the hypertext, would just be a classical textbook that can be read online. But we also add interactive graphics, demos, and calculators to the mix, and thus we create something with combines a classical textbook with computer software for demonstrations. Ultimately, we also plan to add exercises, examples, glossaries, and instructors manuals.

UCLA Electronic Statistics Textbook is very much *under construction*, and it will most certainly be under construction forever. It is not a product, but a project. It is infinite in conception, and thus it will continue to grow. One of the major advantages of the electronic form is that there is no limit on the size of the book, and that material can be added at any point in time. Moreover the number of printings and the number of editions is also unlimited. And, as we shall show below, the same material can be organized in different ways to serve different purposes.

In this paper we will give an overview of the structure of the *UCLA Electronic Statistics Textbook*, and some of the technical problems in its implementation. We will *not* include graphical representations of the pages. It is much better if the reader browses the textbook while reading this paper. Thus, we suggest you now go to

`http://www.stat.ucla.edu/textbook/`

By the way, all other URL's given below are in the same WWW tree. Thus you should always put

`http://www.stat.ucla.edu`

in front of them. Unless, of course, we use pages on other servers, in which case we give the complete URL.

2 Organization

2.1 Hierarchical Organization

The classical hierarchical organization of the textbook mimics that of a real printed book, There are chapters, corresponding with major statistical topics, and sections and subsections of these chapters. Thus the textbook, seen in this way, is a tree, where the structure of the tree follows traditional patterns familiar from the statistical curriculum.

But there are some obvious additions to the traditional structure. In the first place there are the hypertext links, which make it possible to jump from one part of the textbook to another, and in many case to jump to pages on other computers on the net.

Second, the textbook has demos and calculators, which make it possible to show dynamic graphics and to do actual computations from the WWW pages. This is obviously a major extension of the textbook concept, even of the textbook-with-a-floppy-in-the-back.

Third, there are data sets and case studies linked into the textbook. They are more dynamic than the usual textbook examples, because of their interactivity, and because of the hyperlinks. This part is still rather underdeveloped in the current version of the textbook.

2.2 Modular Organization

It is quite conceivable that some readers only want to know about a specific statistical (class of) technique(s). Regression would be one example, confidence intervals and tests of hypothesis would be another. Although such topics often correspond with chapters and section of the hierarchical textbook organization, it can also be convenient to simply collect the components in a linear list of modules. This is done in

```
/textbook/modules
```

2.3 Functional Organization

The textbook contains a large number of what we call *calculators*. Some of these calculators really resemble the classical hand-held TI or HP calculators, but others have more specific statistical functionality. Thus there are t-test calculators, correlation calculators, cumulative distribution function calculators, and so on. The interface to the calculators is

```
/textbook/calculators
```

Besides the calculators there are also demos. The difference between the two is not absolute, but demos tend to be graphical, dynamic, and teaching-oriented. They illustrate concepts, such as confidence intervals or the influence of the bin-size on the shape of a histogram. A number of demos written in Xlisp-Stat are collected at

```
/textbook/demos/
```

3 Implementing the Textbook

Components of the textbook that require computation can be implemented on the server-side and on the client-side. This means calculations are either done on our machine, which runs the WWW server, or on your machine, which browses our WWW pages. Both alternatives have their strong and weak points.

3.1 The Server Side

3.1.1 The HTTP Server

The textbook relies in various places on the fact that we run the Apache HTTPD, with specific modules compiled into the server. This means, in terms of portability, that in order to "run" the textbook you need the same server, similarly configured.

3.1.2 CGI

The classical server-side programming uses the CGI interface to pass parameters from WWW pages to programs which then write their output back to a WWW page. The output can have any MIME type, it can be HTML, or plain text, or graphics, or VRML. CGI program are usually written in perl or C, and there are many libraries and headers around to make CGI programming easy (provided one is reasonably fluent in the base language.

A typical example of a CGI program are the calculators at

`/textbook/singles/describe_single/probmodels/calc.html`

which are currently being replaced by calculators using server-side scripting in PHP/FI (see the next section). Another example is the spinner at

`/calculators/spinner.html`

which takes input form an HTML form and uses this to write a VRML program, which is then loaded into a VRML World Viewer installed as a helper or plugin. Observe this combines the server and the client side.

3.1.3 Embedded Languages

Embedded languages are pieces of code written directly on the page. If we use a server-side approach, then such pages are usually called `foo.phtml` or `foo.shtml` or `foo.mhtml`. The server knows, from the extension, that the page should be interpreted by some program before being send to the client. The mother of all embedded languages are the *server side includes*, which do not require a separate program, because the server itself can interpret them.

In the textbook we currently use PHP/FI, a C-like scripting language that can be nicely integrated with the Apache HTTPD server. In fact, we have extended PHP/FI in such a way that many statistics functions are now available as commands in the language. Because PHP/FI is compiled as a module into the HTTPD, this means our WWW server now talks statistics. You can write the statistics commands directly in your WWW pages.

Two examples of server-side scripting are the two-sample calculator, the normal distribution calculator, and the scatterplot maker at

`/calculators/twosamp/ /calculators/cdf/normal.phtml /calculators/scatter/`

3.2 The Client Side

3.2.1 Browsers

One of the reasons to implement as much of the textbook as possible on the server side is because it makes using the textbook much more independent from the capabilities of the particular browser the client uses. For the same reason, we avoid extensions of HTML which are understood by only a small number of browsers.

3.2.2 JavaScript

JavaScript is a client-side embedded language, which is interpreted by the browser itself (either Netscape or MSIE is necessary). Some calculators are still written in JavaScript.

3.2.3 Java

Java is currently client-side. If the browser sees a reference to a Java applet, it download the bytecode from the server and runs it in its build-in virtual machine. Modern implementations use a local just-in-time compiler to convert bytecode to object code for the client machine. This requires a lot of heavy machinery at the client side. For some of the larger applets, such as the idvi page viewer (see below), it require quite a bit of downloading time as well. There are some Java applets in the textbook, ultimately they will al be moved to the server-side. Sun's JavaSoft is working on server-side Java.

A good example of a Java applet is the one have borrowed from Webster West (SC) at

`http://www.stat.sc.edu/~west/javahtml/new1ConfidenceInterval.html`

3.2.4 Plugins

Plugins are shared libraries using the Netscape API. They are usually written in C, and they extend the functionality of the browser, often by defining their own specific MIME type. Files of that type will then be handled by the plugin. Plugins must be installed locally by the clients, and they must be maintained and upgraded on the local machine. They also add to the Netscape bloat, because they are loaded into the Netscape process at browser startup. For this reason, and also because writing a plugin is more complicated than writing a CGI program, there are no plugins planned for the textbook.

3.2.5 Helpers

Helpers are programs running on the local machine that handle specific MIME types. For the textbook, the most often used helper is Xlisp-Stat. If the browser encounters a file `foo.xli`, with the `xli` extension, in starts up Xlisp-Stat and loads the file into that program. Then Xlisp-Stat runs the lisp code in the file, and displays the output in its usual way. Even more than plugins, this has the disadvantage that a complete Xlisp-Stat must be properly installed on the client machine. Clients often have to close the resulting Xlisp-Stat processes, and the many windows that are created could easily clutter the screen.

Again, it is possible to run programs like Xlisp-Stat on the server-side as well. Examples, using CGI, are the Xlisp-Stat WWW Interface and the Power Calculator at

`/cgi-bin/Xlisp-Stat.cgi` `/calculators/powercalc/`

and an example using PHP/FI scripting is the generalized linear models calculator at

`/calculators/glm/`

3.3 Specific Elements on Pages

3.3.1 Formulas

There is, so far, no direct way available to write formulas in HTML. The HTML 3.0 standard did have formulas, but only one single experimental browser recognized the relevant tags. This is a major nuisance. By now, there are many ways to create formulas, but almost all of them are rather clunky.

It is possible to use the WebEQ applet in Java (and some other similar applets) to embed each formula in the parameters of an applet tag. It is also possible to view dvi or pdf files, with formulas, in the browser window. This uses plugins, such as ndvi or Adobe Acrobat, or it uses Java, such as the idvi system. This has the usual client side disadvantages.

The most common way to deal with formulas in WWW pages is to include them as graphics. This is done by the LaTeX2HTML filter, it is also done by the pphtml preprocessor. Basically these programs takes $\TeX$ commands as arguments, process them with $\TeX$, and translate the dvi files to gifs (often from dvi to ps, from ps to ppm, and from ppm to gif, a somewhat tedious process). In the textbook, we currently rely on pphtml. You can see some examples at

`/textbook/formulas/`

3.3.2 Graphics

GIF files are the format used most to incorporate graphics. One of the major advantages of GIF is that the graphics can be generated "on the fly", when needed, using tools such as embedded languages or CGI libraries. The scatterplot maker at

`/calculators/scatter/`

is a nice example, because it reads user parameters and then generates the GIF. This can also be done quite easily in Java. Sequences of GIF's can also be used in animation.

3.3.3 Multimedia

By now, there is much statistics course material that uses additional multimedia elements, such as sounds and videos. We have decided not to incorporate these in the WWW textbook yet, because both client side and server side ways of handling these additional elements are not standardized at all, and inherently more complicated. Also, using sound and video requires a much larger bandwidth than currently available to most people.

4 Conclusion

Writing an electronic textbook for the WWW is a desperate race to keep up with rapidly changing technology. Many of the components of the UCLA Statistics Textbook have been and will continue to be rewritten almost from scratch, because it only takes a year to become outmoded. Nevertheless, it is clear that some form of WWW textbook is needed and will in fact perhaps be the dominant statistics teaching tool in the not too distant future. We will continue to stumble towards that goal.