COMPUTER MODULES FOR TEACHING STATISTICAL CONCEPTS

Mervyn Marasinghe
Department of Statistics
Iowa State University, USA

Marasinghe, Meeker, Cook, and Shin (The American Statistician, 1996) used graphical and simulation techniques to construct a system of computer-based modules for teaching statistical concepts. The software component of these modules consisted of a computer program written in LISP-STAT incorporating a highly interactive user interface. The instructional component is set of a prototype lessons providing information to instructors such as a description of concepts that may be illustrated with the program and possible exercises. Since then, the addition of several new modules have enhanced the usefulness of the system. In this paper we illustrate several of these modules useful for teaching concepts as different from how sample size and confidence level affects the width and coverage of confidence intervals to how variability affects precision of experimental results.

INTRODUCTION

Moore, Cobb, Garfield, and Meeker (1995), discuss the impact of using modern technology in statistics education and conclude that “Technology may at last really change higher education, but because our professional practice is already based on technology, we can welcome technology in our teaching. New styles of learning require active involvement of students in learning and interaction between teachers and students, but because statistical practice is active and interactive, it is easier for us to change our classroom practice.”

Computers are beginning to be used in statistics education in more innovative ways. Web-based courses, computer-aided instruction using multi-media, and software that incorporate dynamic graphics and computer simulation are a few examples. Marasinghe, et al. (1996) presented computer-based modules that use interactive graphics and simulation to teach statistical concepts for undergraduate courses. This paper extends that work to include modules that cover important statistical concepts underlying regression analysis and the design of experiments. These modules are flexible enough to be used in both undergraduate and graduate level courses in applied statistics.

COMPUTER-BASED MODULES FOR INSTRUCTION

This project is aimed at improving the effectiveness of undergraduate statistics courses. It's main goal is to develop computer-based modules that are more concept-driven than course-oriented. Thus they are designed to be more easily adaptable for general use and statistics instructors could use them to supplement and enhance their own teaching methods. The availability of these easy-to-use modules will enable them to present and illustrate statistical concepts more effectively than is possible using standard approaches. As these modules incorporate dynamic graphical displays that provide instantaneous visual feedback, they encourage *active* learning, long-recognized as a desirable characteristic of good statistics instruction.

The software component of these modules is a computer program written in a statistical programming language that incorporates dynamic graphical techniques, computer simulation, and a highly interactive user interface. The instructional component is a set of prototype lessons that includes a description of concepts to be covered, instructions on how to use each module, and exercises designed to reinforce the learning experiences. These modules can be used both as classroom demonstration tools and as self-paced exercises by individual students or small groups. Course instructors can use the lesson plans as templates to create their own lessons.

IMPLEMENTATION OF THE SOFTWARE COMPONENTS

The software component of the modules described here has been programmed in the Lisp-Stat language (Tierney, 1990). Lisp-Stat is a powerful object-oriented programming language that allows rapid prototyping and is an ideal environment for developing software of this type because it allows the incorporation of true dynamic graphics, linking of routines written in Fortran or C, and can be installed on the different instructional computing platforms in use at most universities and in industry (Unix workstations, Macintosh, and PC's running Windows software). In addition, it is supported by a large core of built-in statistical functions. Another important advantage of using Lisp-Stat for developing software is that, for noncommercial applications, it is available without cost. For a good introduction to Lisp-Stat and other Lisp-Stat based software, see de Leeuw (1994).

We have developed user-interfaces that are consistent across modules. Students need only execute one command to start a module. All further interaction is through a mouse-driven interface. One may click on a button to initiate action, click and drag on a menu button to select items from a pull-down menu, or click and move slide-bars to set values.

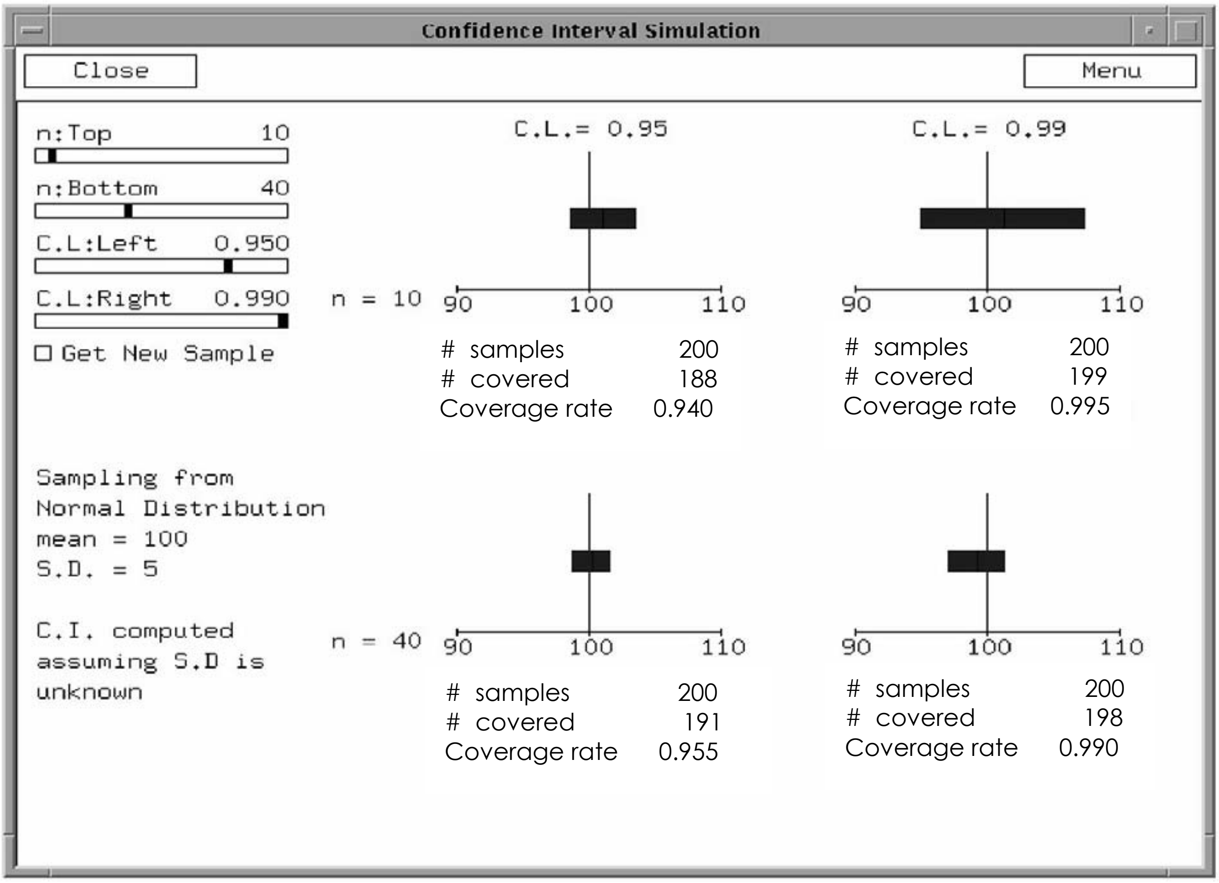
SELECTED EXAMPLES OF COMPUTER-BASED MODULES

In this article we describe a selected set of computer-based modules that are examples of their use in teaching introductory and more advanced courses on statistical methodology. The three modules are presented here illustrate a variety of statistical concepts.

MODULE *teach1*: UNDERSTANDING CONFIDENCE INTERVALS

The confidence interval (CI) module is used to illustrate some of the important concepts related to the use and interpretation of CI's. Some of these are to

- Illustrate the fact that constructed CI's vary from sample to sample.
- Illustrate "coverage" or "correctness" of a CI.
- Provide an understanding of what is meant by phrases like "95% confidence".
- Demonstrate the effects that changing the sample size and the confidence level have on CI's.



```
n:Top        10
n:Bottom     40
C.L:Left     0.950
C.L:Right    0.990

Sampling from
Normal Distribution
mean = 100
S.D. = 5

C.I. computed
assuming S.D is
unknown
```

```
n = 10, C.L.= 0.95              n = 10, C.L.= 0.99
#  samples         200          #  samples         200
#  covered         188          #  covered         199
Coverage rate    0.940          Coverage rate    0.995

n = 40, C.L.= 0.95              n = 40, C.L.= 0.99
#  samples         200          #  samples         200
#  covered         191          #  covered         198
Coverage rate    0.955          Coverage rate    0.990
```

Figure 1: Frame of the Confidence Interval Module Window after 200 Simulations

The CI module starts up with a graphic window that contains four slide-bars to control the sample size (n) and the confidence level ($C.L.$) as shown, respectively, on the top and left sides of the 2×2 array in Figure 1.

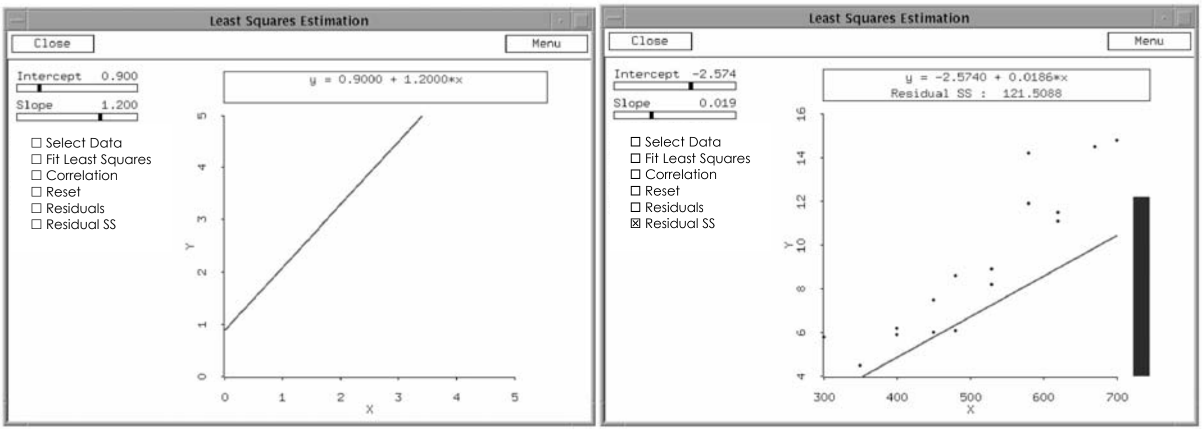
For each simulated replication, a pseudo-random sample of data is generated from a normal distribution with $\mu = 100$ and $\sigma = 5$. The CI's for the mean of the normal distribution are computed (assuming that σ is unknown) using the standard Student's t method. Clicking on Get New Sample generates new samples and displays the corresponding CI's in all four panels. The CI's are shown in blue if correct (i.e., if they cover the distribution mean) and red otherwise.

Repeated samples are generated if the *Get New Sample* (button) is held down, while *Coverage Rate* is updated continuously. Figure 1 shows a frame of the window after two hundred such simulations. Observing the sequence of samples illustrates to students that only some intervals capture the distribution mean, that higher confidence levels tend to result in wider intervals (there may be exceptions due to the variability in the sample variance), that larger sample sizes can be used to improve precision, and that the observed coverage rate tends to converge to (but will not necessarily be equal to) the nominal confidence level.

MODULE *regteach1*: EXPLORING LEAST SQUARES FITTING

The *regteach1* module is useful for illustrating some of the fundamental concepts related to simple linear regression. For example:

- A single summary statistic like a correlation coefficient or R^2, by itself, cannot be used to interpret the strength of a relationship. A scatter plot is an essential complement to examining the relationship between two variables.
- It is important to understand the idea of least squares fitting. It can be demonstrated that one may not always be minimizing the sum of squared deviations when "fitting a line by eye".
- Magnitudes of the residuals from a regression depend on the fitted line. Thus a simple residual plot can reveal a lot about the goodness of the fit.



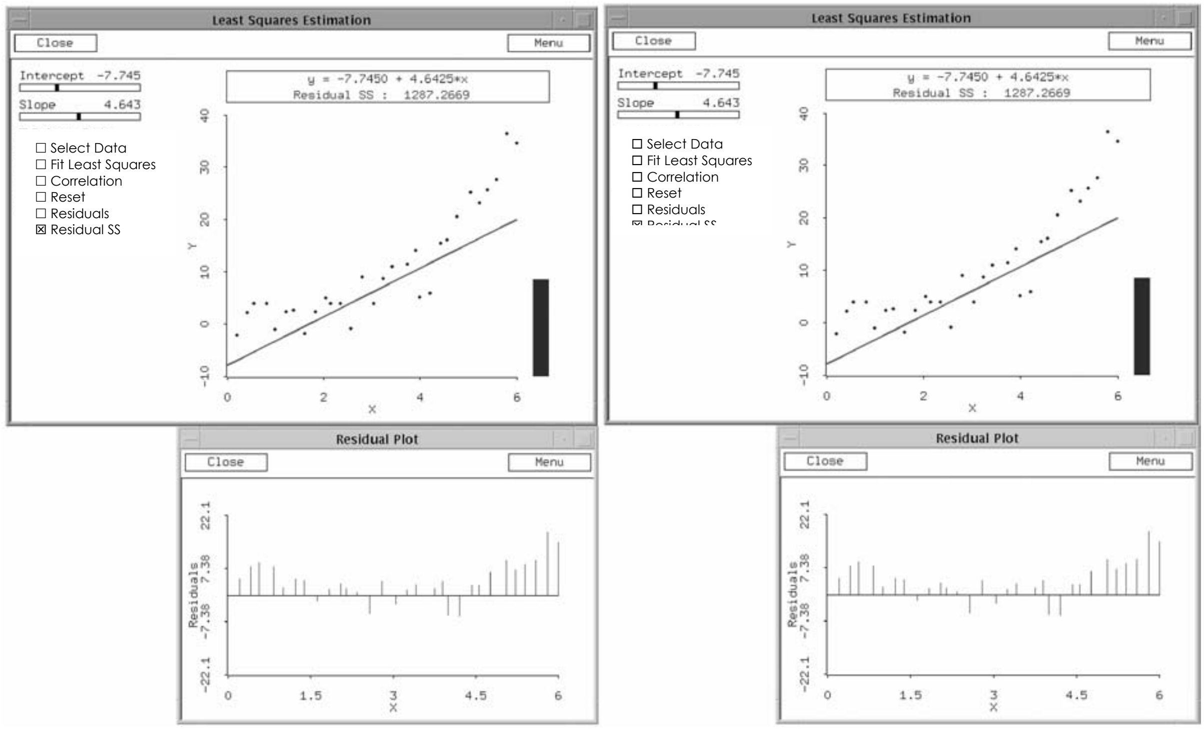
Intercept 0.900
Slope 1.200
y = 0.9000 + 1.2000*x

Intercept -2.574
Slope 0.019
y = -2.5740 + 0.0186*x
Residual SS : 121.5088

Figure 2: Two Frames of the *regteach1* Module Window

The frame on the left-hand side of Figure 2 shows the initial view of the *regteach1* module window. Changing the slope and intercept values in the slide-bars will dynamically change the slope and intercept of the plotted line and update the numerical coefficients in the box above the plot. Pressing the *Select Data* menu button and using the resulting pull down menu allows the user to select one from a list of simple regression data sets. The frame on the right-hand side of Figure 2 shows the re*gteach1* module after such a data set has been selected.

Pressing the *Residuals* button will produce an additional window containing a plot of residuals from the current fitted line. The signed deviations are displayed as line segments drawn from a zero baseline, plotted at the corresponding x-values. This can be used to demonstrate dynamically, the dependence of the magnitudes of the residuals on the fitted line (such as the fact that the fit improves as the residuals get smaller), as well as identifying various patterns in the residuals (such as curvature) for diagnosing the fit. Figure 3 shows the residual plots corresponding to two different lines fitted to a data set.



Least Squares Estimation

Intercept -7.745

Slope 4.643

y = -7.7450 + 4.6425*x

Residual SS : 1287.2669

□ Select Data
□ Fit Least Squares
□ Correlation
□ Reset
□ Residuals
☒ Residual SS

Least Squares Estimation

Intercept -7.745

Slope 4.643

y = -7.7450 + 4.6425*x

Residual SS : 1287.2669

□ Select Data
□ Fit Least Squares
□ Correlation
□ Reset
□ Residuals

Figure 3: Least Squares Module: Two Fitted Lines and Corresponding Residuals

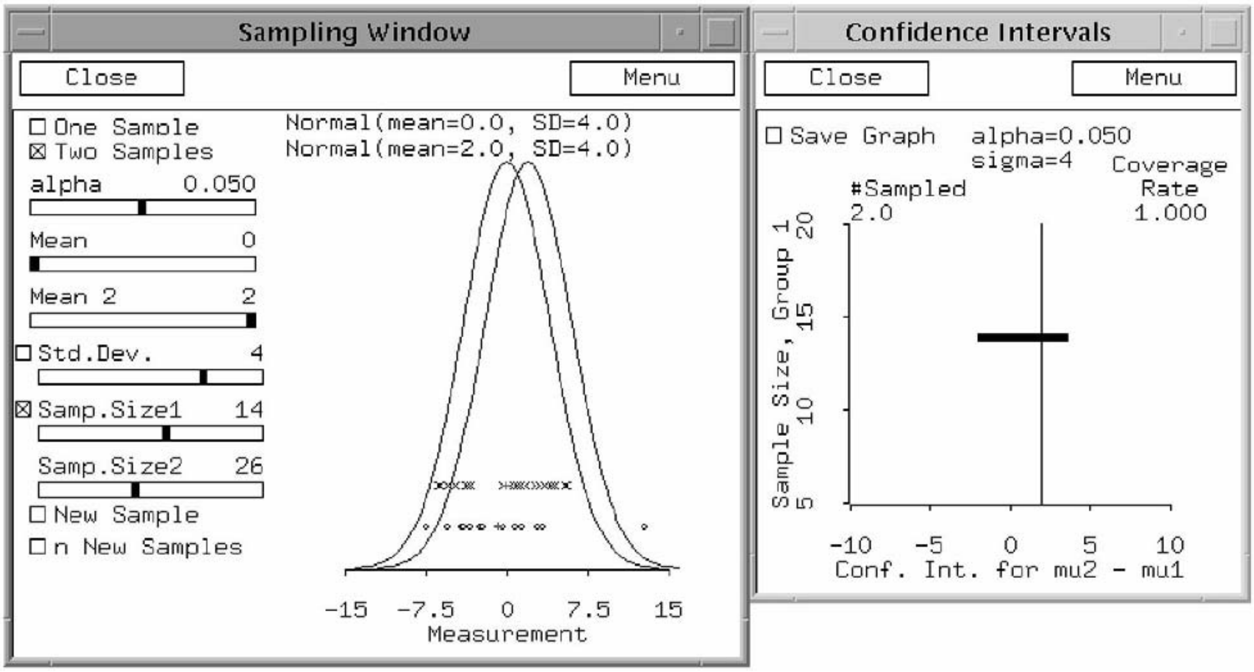
The user can attempt to find the best fitting line to the selected data “by eye” (graphical fitting) using the slide-bars to change the slope and intercept; the plotted line and the residual sum of squares [Residual SS (or RSS); hidden, by default] will be updated dynamically. When satisfied with the line fitted “by eye”, the user can press the *Fit Least Squares* button to display the least squares line fitted to the data and the resulting summary statistics (including regression coefficients, residual standard deviation, and the correlation coefficient). Comparison of the lines from different “by eye" fits with the least squares line, illustrates the “minimization” concept associated with the least squares criterion. In addition to a graphical fit, a user may also attempt to fit the best line numerically, by attempting to minimize the value of the RSS, or better, by using the RSS “thermometer” displayed on the right margin of the primary window, both activated by pressing the *Residual SS* button.

MODULE *dsnteach1*: VARIABILITY AMONG EXPERIMENTAL UNITS

Variation among experimental observations is due to inherent differences in experimental material and variation in the measurement procedure. If smaller differences among treatment effects are to be found, the *precision* of the experiment becomes critical. The precision of an experiment, defined as the ability to detect significant treatment effects, is measured by the standard deviation of the estimates (or differences in estimates) of the treatment effects. This depends directly on the experimental error variation, i.e., the variation among *replications*. Replications are observations obtained from random repetitions of an experimental treatment or a test condition. The precision of an experiment thus depends upon both the choice of the size of the experiment as determined by the number of replications (*sample size*) of each application of a treatment, the homogeneity of experimental material, and the order of measurement as governed by the experimental design in use.

This module, described in Iversen and Marasinghe (2001), examines these ideas by constructing confidence intervals for the mean of one sample and for the difference of the means of two samples. A shorter confidence interval for a population mean, for example, implies that a null hypothesis of mean is equal to a specified value may be rejected more readily than when the interval is longer. Thus the experiment that produced the shorter interval is more precise. The width (or length) of a computed confidence interval is a function of the estimated standard deviation of the sample means and the sample size, and since the estimates depend on the true variance among the observations, confidence intervals can be used to compare precisions of different experimental procedures. The concepts illustrated by this module are:

- The precision of an experiment can be measured by the widths of confidence intervals on the quantity of interest. The smaller the width the higher the precision and vice versa.
- The precision of an experiment depends on the sample size and it increases with the sample size if the error variance is constant.
- The precision depends on the experimental error variance. With smaller experimental error variance, the experiment will be able to detect smaller differences.



☐ One Sample
☒ Two Samples
alpha 0.050
Mean 0
Mean 2 2
☐ Std.Dev. 4
☒ Samp.Size1 14
Samp.Size2 26
☐ New Sample
☐ n New Samples

Normal(mean=0.0, SD=4.0)
Normal(mean=2.0, SD=4.0)

☐ Save Graph
alpha=0.050
sigma=4
#Sampled 2.0
Coverage Rate 1.000

Figure 4: dsnteach1 Module Windows

Figure 4 shows the first window (the *sampling window*) that is displayed when this module (*dsnteach1*) is activated. Data from a one-sample or a two-sample experiment may be randomly generated, and the sample data are displayed superimposed by the normal distribution curve from which the sample was drawn. Slide-bars are used to set values for the significance level (α), the mean of each sample, the standard deviation (σ) and the sample size (n).

A second window, *the confidence interval window*, shows a confidence interval for the mean or mean difference of the sampled data (Figure 4). New samples are generated and confidence intervals are computed via the *New sample* buttons in the sampling window. Either the standard deviation or the sample size can be displayed on the vertical axis. This is controlled by check boxes in the sampling window. For each value on the vertical axis, only the latest interval is displayed, but a running tally is kept of how many intervals contained the true effect, and the coverage rate is displayed on the graph. These confidence intervals demonstrate the effect of σ or n on the interval width and, thus, on the precision of the experiment. Changing any slide-bar other than the checked one causes the confidence interval window to be reset.

SOFTWARE AVAILABILITY

The instructional modules and Lisp-Stat source code for the software components of our modules are available via anonymous ftp from Iowa State University. To obtain these use the command *ftp las2.iastate.edu* with "anonymous.stat" as the username and "yourusername@your.email.host" as the password. This should get you into the departmental directory named "anonymous". The subdirectories "Teach", "RegTeach", and "DsnTeach" contain Readme files describing three versions of the software.

If you are accessing files from a Unix host, the shell archives such as "Dsnteach.shar" should be sufficient to obtain a version that can be installed on the Unix, PC or Macintosh platforms. Otherwise obtain the file appropriate for your platform (e.g. Dsnteach.exe for PC/Windows and Dsnteach.sea.hqx for Macs). These files are binary archives that unbundle when executed. Other Readme/Install files will be found in each package after unbundling.

Lisp-Stat is freely available from *umnstat.stat.umn.edu* or from *statlib*. It is also available from UCLA via the URL *http://www.stat.ucla.edu*. Our software currently runs under Version 2.1 Release 3.50 of *Lisp-Stat*.

CONCLUDING REMARKS

This article has focused on a collection of instructional modules that we have developed specifically for teaching statistical concepts. These modules are being developed and used in courses that have as their primary audience advanced statistics undergraduates and graduate students from other disciplines.

The availability of powerful computing hardware and tools to develop instructional software is revolutionizing the delivery of post-secondary statistics education. Using lessons and exercises based on simulation and dynamic graphics will not only improve students' level of understanding of statistical methodology but also enhance their intellectual curiosity and interest. The time, effort and resources required for developing this type of material is enormous. However, it provides the statistics profession one avenue by which to incorporate new technology into their curriculum.

REFERENCES

de Leeuw, Jan (1994). The Lisp-Stat Statistical Environment. *Statistical Computing and Graphics Newsletter*, *5(3)*, 13-17.

Iversen, P, and Marasinghe, M.G. (2001). Dyanamic Graphical Tools for Teaching Experimental Design and Analysis Concepts. *The American Statistician*, *55(4)*, xxx-yyy.

Marasinghe, M.G., Meeker, W.Q, Cook, D, and Shin, T (1996). Using Graphics and Simulation to Teach Statistical Concepts, *The American Statistician*, *50(4),* 342-351.

Moore, D.S., Cobb, G.W., Garfield, J., and Meeker, W.Q. (1995). Statistics Education Fin de Siècle. *The American Statistician*, *49(3),* 250-260.

Tierney, L. (1990). *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*. New York: John Wiley and Sons.