Using Graphics and Simulation to Teach Statistical Concepts

Mervyn Marasinghe
William Meeker
Dianne Cook and Tae-sung Shin
Department of Statistics
Iowa State University
Ames, IA 50011

January 6, 1995

Abstract

The value to students of *active* learning has been recognized. This has lead to the wide use of assignments in statistical methods courses where students use statistical software and computing equipment to analyze data. These assignments enable most students to master the *mechanics* of data analysis. The amount of experience that a student can get with such assignments is, however, limited. Thus, a sizable proportion of students have difficulty grasping some of the many *concepts* that are introduced in these courses. Nevertheless, these concepts are important for effective modeling and data analysis and instructors should focus on them. By using current computing technology, it is possible to supplement standard data analysis assignments and algebraic derivations and have students become actively involved in the learning of important statistical concepts. The learning experience can be enhanced by giving students additional statistical "experiences" by using combinations of carefully designed and implemented multiple simulations and dynamic graphics to illustrate key ideas. In this article we describe and illustrate several instructional modules and corresponding software that have been designed to assist instructors in teaching introductory statistics courses.

Keywords: Education; teaching; dynamic graphics; Monte Carlo; simulation

1 Introduction

1.1 Computers and statistics instruction

In the past 20 years, computers and statistical software have had a major impact on the practice of statistics and the teaching of statistical methods. More recently, popular statistical packages have been moved to microcomputers and workstations and new procedures and capabilities have been added to these packages. These recent changes have not, however, resulted in a substantive change in statistics instruction. Instructors continue to use computing software in teaching for routine data analysis almost exclusively. The purpose of this article is to describe a project at Iowa State University in which we have developed teaching materials and computer software that extend the application of computers in statistics instruction and make a substantial contribution toward improving the effectiveness of undergraduate statistics courses.

1.2 Statistical concepts

Our goal has been to develop new instructional tools and make these available for general use. These tools will provide statistics instructors with the capability of supplementing teaching methods by presenting and illustrating statistical concepts more effectively than is possible using mathematics or static graphical displays in the classroom. These tools allow dynamic display and exploration of important statistical concepts, in addition to flexible and interactive modeling and data analysis.

Traditionally, statistics instructors have used lectures and static visual displays to explain and illustrate important statistical concepts. Examples of some of the important concepts that we want our students to understand include

- Different graphical displays tend to highlight different aspects of a set of data. More than one graphical display is needed to thoroughly explore a set of data.
- When examining probability plots to assess the adequacy of a normal distribution as a model for data we expect to see departures from linearity, especially in the tails of the distribution, even when the underlying distribution is normal.
- The endpoints of a traditional confidence interval are random functions of our sample data and the confidence level really describes the *process* of constructing such interval.
- The adequacy of the central limit theorem, when used to justify the use of inference based on normal distribution theory, depends on the sample size *and* on the shape of the underlying distribution. There is no magic number (like the 30 or 50 or 100 suggested in so many text books) above which the approximation is adequate.

Having students work through homework assignments, lab exercises and class projects provides practice with mechanics, illustrates ideas, and clearly helps the process of learning these concepts. For most individuals, however, true *understanding* comes only after extensive experience. By using highly interactive graphical display of simulation results in carefully designed instructional modules, students can quickly acquire pseudo-experiences that will help solidify their understanding of important concepts. For example, students can use simulation and dynamic graphical displays to study and quickly determine the effect of changing sample size and underlying distribution on the adequacy of the approximation afforded by the central limit theorem.

1.3 Related work

Saunders (1986) describes the use of dynamic graphics to produce "moving visualizations designed to introduce difficult concepts, reinforce mathematical ideas, and explore the techniques of probability modeling." The visualizations were used in a distance-learning television program produced by the BBC and the paper describes visualizations to illustrate the effect that parameter changes have on binomial, Poisson and bivariate normal distributions, as well as a complicated simulation to illustrate concepts behind the theory of conflict. Doane, Mathieson and Tracy (1994) present a similar program, for personal computers, that allows users to explore distribution shapes. Groeneveld (1979) and Gnanadesikan, Scheaffer, and Swit (1987) provide a number of illustrations of the use of simulation in teaching

elementary statistical concepts. Almond (1992) describes `ElToY`, a program for elicitation of prior information for Bayesian statistics that allows the user to dynamically view probability distributions and convergence of distributions according to the central limit theorem.

1.4 Overview

The rest of this paper is organized as follows. Section 2 provides an outline of the components of our overall project to develop instructional modules based on simulation and graphics. Section 3 describes and illustrates six instructional modules that have been developed specifically for undergraduate statistics service courses. In Section 4 we explain how to acquire some of the materials that we have developed. Section 5 briefly describes other instructional modules aimed at higher-level courses. Section 6 contains some concluding remarks.

2 Outline of the Project

2.1 Instructional Modules

We have developed a number of instructional modules designed to illustrate concepts, provide important insights, and lead to more meaningful experiences and assignments with real or realistic problems of data analysis and inference. The software components of these modules use a combination of state-of-the-art computing hardware, statistical programming languages, high resolution color graphics, simulation, and a highly interactive user interface. The modules are independent of specific courses. Course supervisors and instructors can modify or build around these modules to create their own lessons.

2.2 Documentation of the instructional modules

For each instructional module, we have developed, in LaTeX format, a sample 3-to-6 page lesson/instruction/exercise document. These documents consist of

- An introduction to and description of the important ideas behind the concept(s) to be covered.
- Objectives for the instructional modules.
- Instructions on how to fire up and run the software part of the module.
- Warm-up exercises to help the student become familiar with the software module.
- More formal exercises and questions requiring execution of the software, (usually following a reasonably precise set of instructions), careful thought, interpretation of results, and explanation of conclusions.
- Notes for the instructor, including comments on what one should get from doing the more formal exercises.

Instructors can either use these documents or customize them for their own purpose and class. The documents (perhaps without the notes-for-the-instructor part) can be made available to the student as a handout or on-line in the form of a dvi or postscript file.

2.3 Implementation of the software components

The software components of the modules described here have been programmed in the Lisp-Stat language. The Lisp-Stat system provides a computing environment with the following important features:

- It is a powerful object-oriented programming language that allows rapid prototyping and development.
- It allows development of modules using true dynamic graphics.
- It allows links to existing Fortran or C algorithms.
- It can be installed on the different instructional computing platforms in use at Iowa State University (Unix workstations, Macintosh, and IBM-compatible PC's)
- For noncommercial applications, it is available without cost.
- It has a large core of statistical functions

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 by using point-and-click or by moving slide-bars. Students do not need to have any knowledge of the Lisp-Stat programming language. In more advanced courses students may, however, begin to explore the Lisp-Stat language in greater depth.

3 Examples of Instructional Modules

We have developed modules to explore confidence intervals, univariate data plotting methods (probability plot, box plot, and histogram), power transformations, the central limit theorem, linear least squares regression, and sampling distributions of various statistics (sum, mean, median, standard deviation, variance, range and t-statistic). Each module is discussed in detail below by listing some important concepts targeted by each module, followed by brief comments on the operation of the accompanying software components. The individual instructional lessons and exercises (outlined in Section 2.2) provide more detail on how guided software operation can be used to reinforce the concepts.

3.1 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 "95% confidence".
- Demonstrate the effects of changing the sample size and the confidence level of a CI.

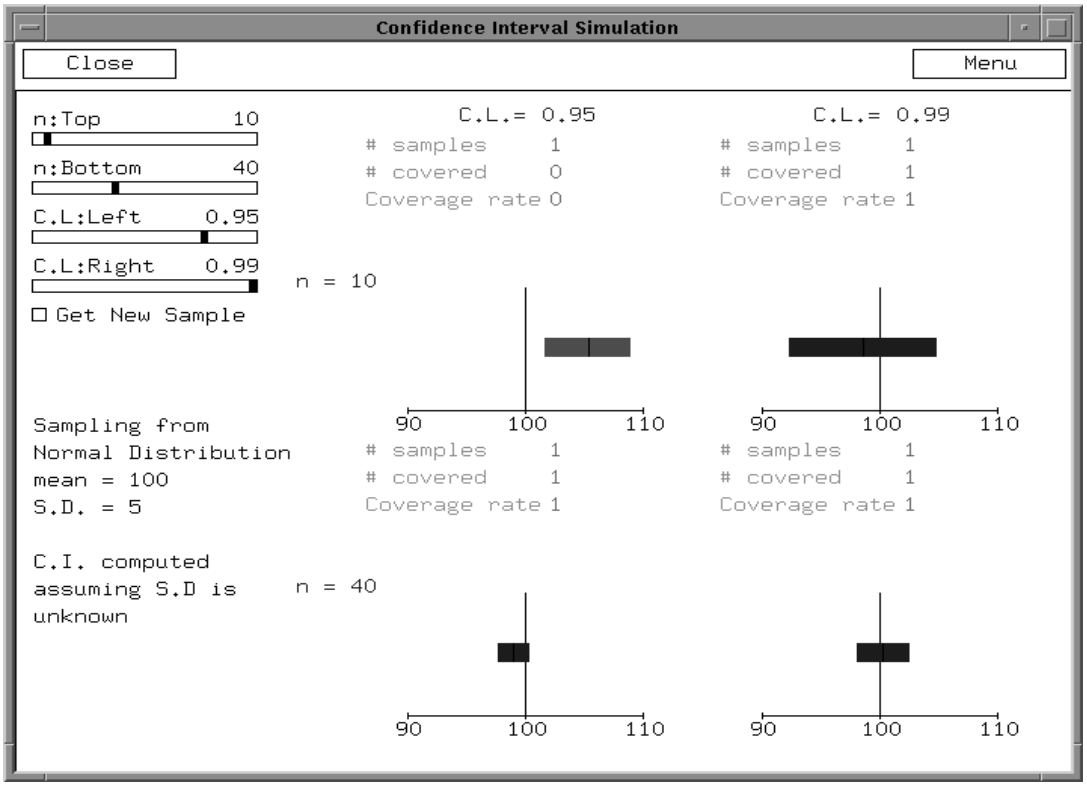

Figure 1: Initial set-up of the Confidence Interval Module Window

The CI module starts up with a graphic window shown in Figure 1. Four slide-bars 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 simulation replication, a random sample of data (actually pseudo-random sample) 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 (The first confidence interval generated for C.L.=0.95, n=10 in Figure 1 does not cover the distribution mean). Repeated samples are generated if `New Sample` button is held down; `Coverage Rate` is updated continuously. Figure 2 shows a frame of the window after one hundred such simulations.

Observing the sequence of samples illustrates to students that only some intervals capture the distribution mean, that higher confidence results in wider intervals, and that larger sample sizes can be used to improve precision.

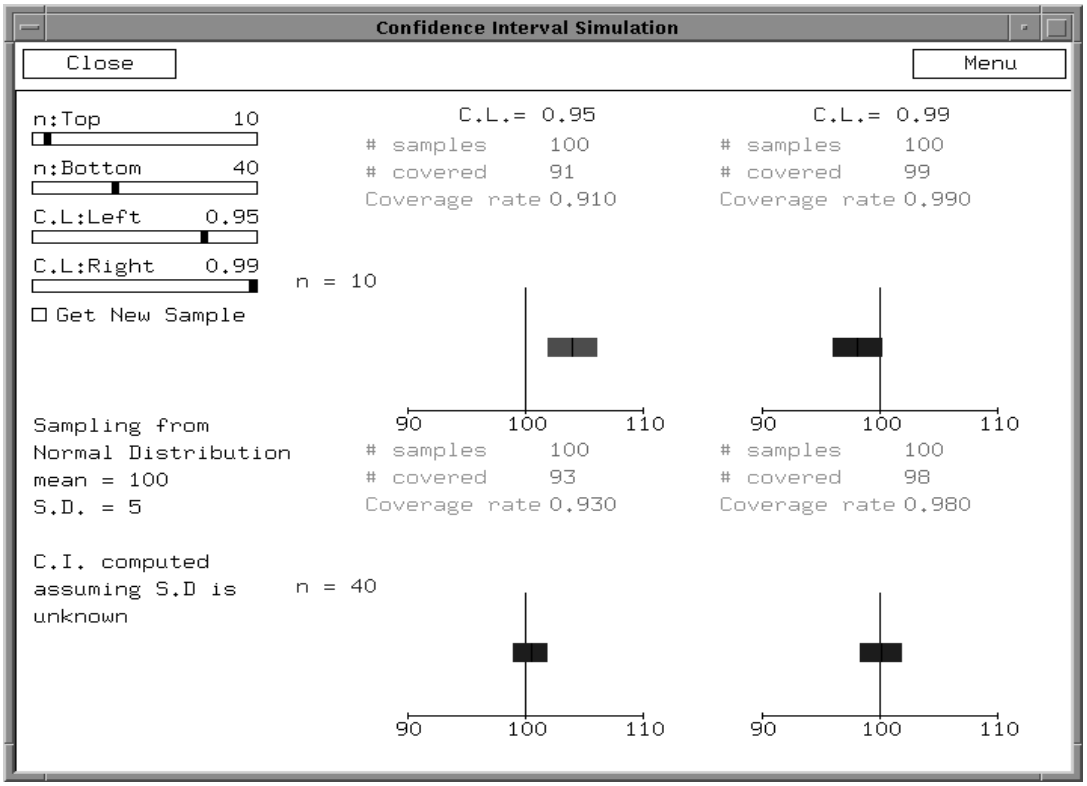

Figure 2: Frame of the Confidence Interval Module Window after 100 Simulations

3.2 Comparing univariate data plots

The univariate data plot (UDP) module allows students to explore and experiment with several graphical methods of displaying univariate data. Some of the concepts central to this module are:

- The histogram of data from a random sample can have a shape different from the parent distribution. Variability in the shape is caused by random variability (noise) in the data. The effect of noise is reduced in larger samples.
- The perception about data that one gets from a histogram depends on the number of bins used in its construction.
- Even when data are generated from a normal distribution, we expect to observe moderate departures from a straight line on a normal probability plot, especially in the tails of the distribution.
- In small samples from a normal distribution, it is possible for a normal probability plot to show distinct curvature. Also, with small samples from a nonnormal distribution, it is possible for a normal probability plot to appear linear. There is much more consistency with larger samples.

- The ability to detect nonnormality depends on how much the parent population distribution differs from a normal distribution and on the sample size.
- The different univariate plots complement each other. The histogram should correspond, approximately to the density. The box-plot allows us to focus attention more on the behavior in the tails of the distribution. The normal probability plot highlights departures from an assumed normal distribution.

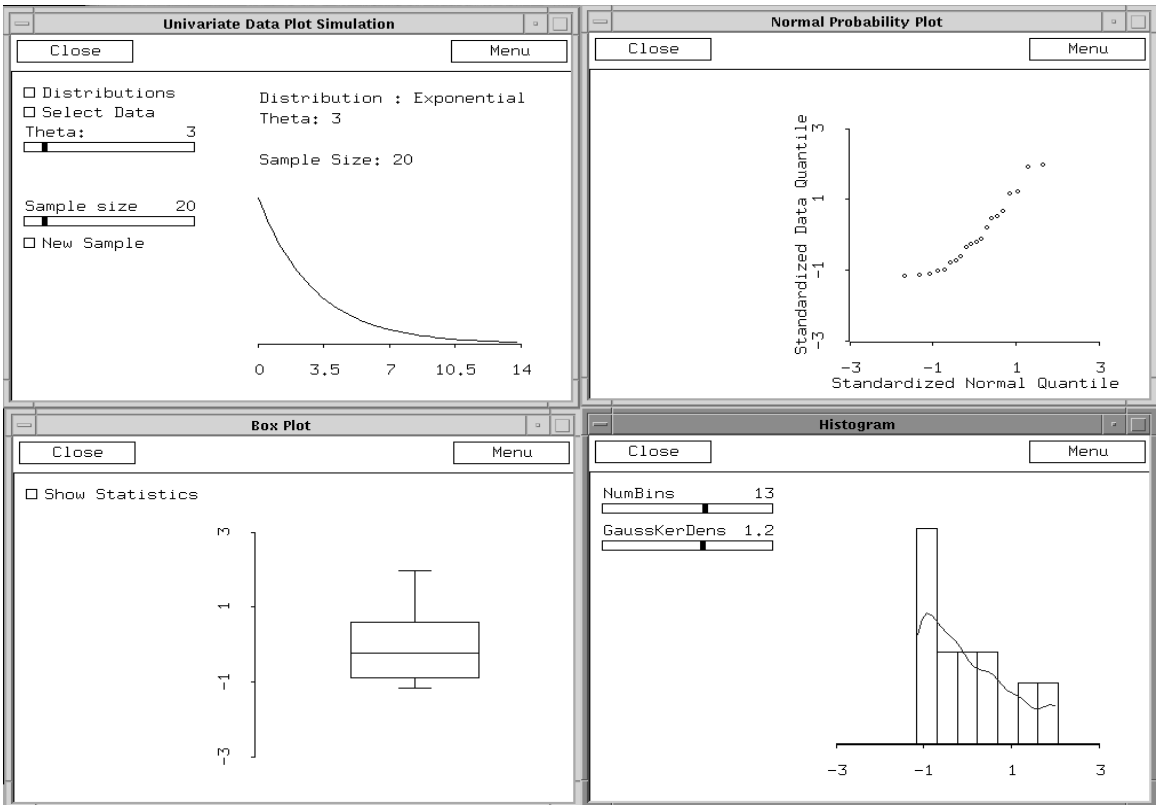

Figure 3: Frame of the Univariate Data Plot Module Windows

Figure 3 shows a frame from the UDP software module. Students might start off choosing a distribution (`Distributions` button), parameters, and sample size (with slide-bars), and looking at plots from a sequence of simulated data sets. A graph of the density or mass function of the selected distribution appears on the window. With each newly generated sample (obtained with the `New Sample` button), the data plots are redrawn. Plots of the different samples provide a sense of how the data deviates from the model as depicted by each type of plot. For the histogram, the student can dynamically adjust the number of bins or the smoothing parameter in the kernel density estimate. Experimenting with different underlying distributions (perhaps guided by exercise instructions) and sample sizes provides a range of experiences that help the student better extract useful information from plots. Finally exercises can move to looking at plots from a selection of *real* data sets chosen from

the menu brought up by the `Select Data` button/menu.

3.3 Exploring Box-Cox transformations

The Box-Cox transformation (TR) module is used to examine the use of the power transformation on simulated samples from a variety of distributions and actual data samples. Some of the main concepts are:

- The power transformation can be used to transform skewed data so that they are more symmetric and, in some cases, so that the data can be adequately described by a normal distribution.
- Graphical methods can be used to roughly gauge the appropriate transformations (values of the shift and power parameters) to use to make samples from some particular distributions have a shape that can be described, approximately, by a normal distribution. Graphical methods are also useful for demonstrating that the Box-Cox transformation *cannot* be used to achieve approximate normality when transforming samples from some other distributions (e.g., Cauchy).
- Graphical methods can also be used to examine the effect that power transformations have on actual data sets.

The TR module has four windows (transformation controls, probability plot, box plot, and histogram) displayed on start up. The plot windows are empty until the student selects a parent distribution (`Distributions`) or a data set (`Select Data`). In Figure 4 the parent distribution is chi-square with 1 degree of freedom and the sample size is 20. The plot in the control window shows the density function of the parent distribution or a description of the data. Clicking on the `power` slide-bar changes the power parameter (λ), and likewise the `shift` slide-bar controls the shift parameter (m) in the transformation

$$x^{(\lambda,m)} = \begin{cases} \frac{(x+m)^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log\,(x+m) & \lambda = 0,\ x+m > 0. \end{cases}$$

For the sample shown in Figure 4 a power of 0.3 and shift close to 1 appear reasonable: the points in the probability plot lie close to a straight line, the boxplot is close to symmetric and the histogram is reasonably bell-shaped.

3.4 Exploring linear least squares

The linear least squares module (LS) module is used to illustrate some of the important concepts related to simple linear regression.

- There is a simple connection between the numerical coefficients in the regression equation and the regression line.
- A single summary statistic like a correlation coefficient or R^2 does not tell the whole story. A scatter plot is an essential complement to examining the relationship between two variables.

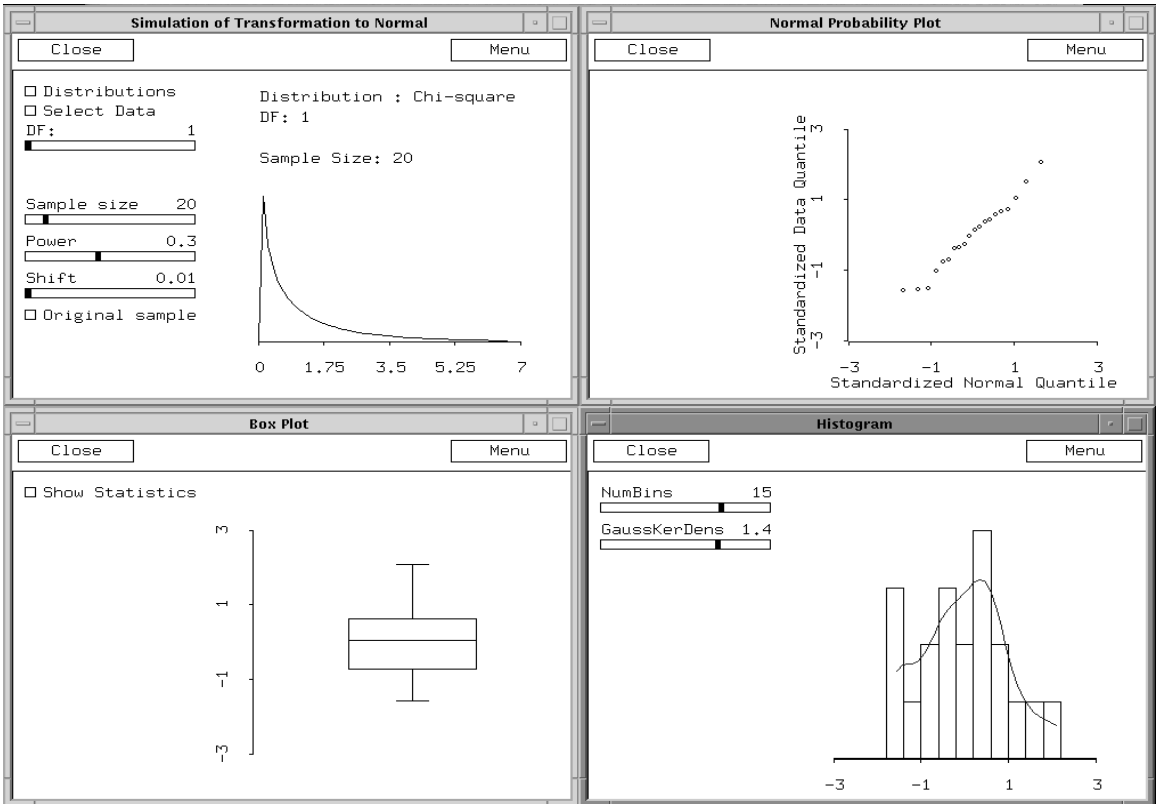

Figure 4: Frames of the Transformations Module Windows

- Finding a line to approximately minimize the residual sum of squares, by eye, is difficult, especially when there is a lot of residual variability in the data.

The frame on the left-hand side of Figure 5 shows the LS module in its initial form. 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. Using the `Select Data` button allows selection from a list of simple regression data sets. The frame on the right-hand side of Figure 5 shows the LS module after a data set has been chosen. Now the student can attempt to fit a line "by eye" (numerically and graphically); the line and the residual sum of squares will be updated dynamically. When satisfied with the "by eye" line, the student can use the `Fit Least Squares` button to display the exact least squares line and summary statistics (including regression coefficients, residual standard deviation, and the correlation coefficient). A comparison of these two lines provides the student with a better sense of the meaning of the least squares criterion for fitting a line to data.

3.5 Visualizing the central limit theorem

The central limit theorem (CLT) module was designed to reinforce the following important statistical concepts.

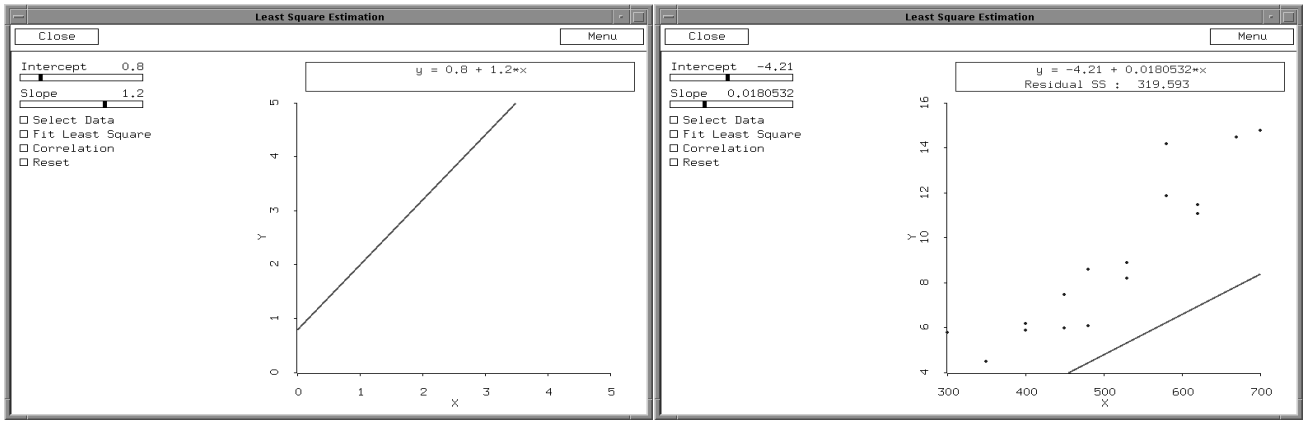

Figure 5: Two Frames of the Least Squares Module Window

- Under certain conditions, in large samples, the sampling distribution of the sample mean can be approximated by a normal distribution.
- The sample size needed for the approximation to be adequate depends strongly on the shape of the parent distribution. Symmetry (or lack thereof) is particularly important.
- For a symmetric distribution, even far from the shape of a normal distribution, an adequate approximation can be obtained with small samples (e.g., 10 or 12 for the uniform distribution).
- Typical textbook guidelines suggest that a sample of size 30 to 60 is needed for the central limit theorem to provide an adequate approximation. However, in some extreme cases (e.g. binomial with $p \approx 1, n = 1$) samples sizes far exceeding the typical guidelines are needed.
- For some distributions without first and second moments (e.g., Cauchy), the central limit theorem does not work.

Figure 6 shows the CLT module window. After a distribution, parameter(s), and sample size are chosen (in a manner similar to the Univariate Data Plot module), a graph of the parent density is shown. Each time the `New Sample` button is used, a new sample is generated and displayed as a dot plot below the density plot. The sample mean is highlighted with the symbol "x," and is added to the accumulated sample means displayed in the dynamically updated histogram on the right. If the `New Sample` button is held down, this process is repeated continuously. To accelerate the sampling process, the `Add 100 Samples` button can be used to immediately add 100 new sample means to the histogram. At any point in the process the student can click on the `Save Histogram` button to save a snapshot of the current histogram.

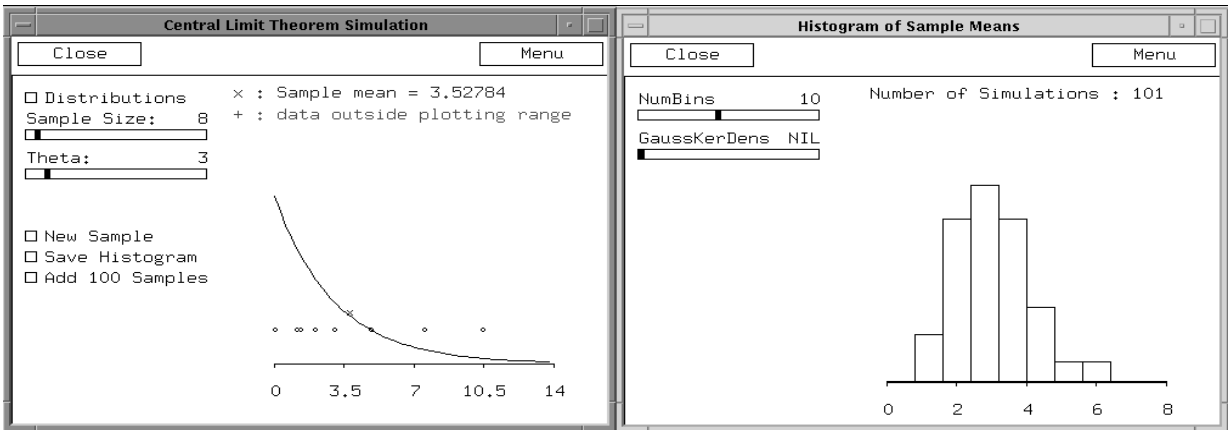

Figure 6: Frame of the Central Limit Theorem Module Windows

The `Save Histogram` button allows the student to take snapshots of the current histogram window to allow comparisons. Figure 7 shows snapshots of three histogram windows comparing sample means from samples of size 12 from three different parent populations.

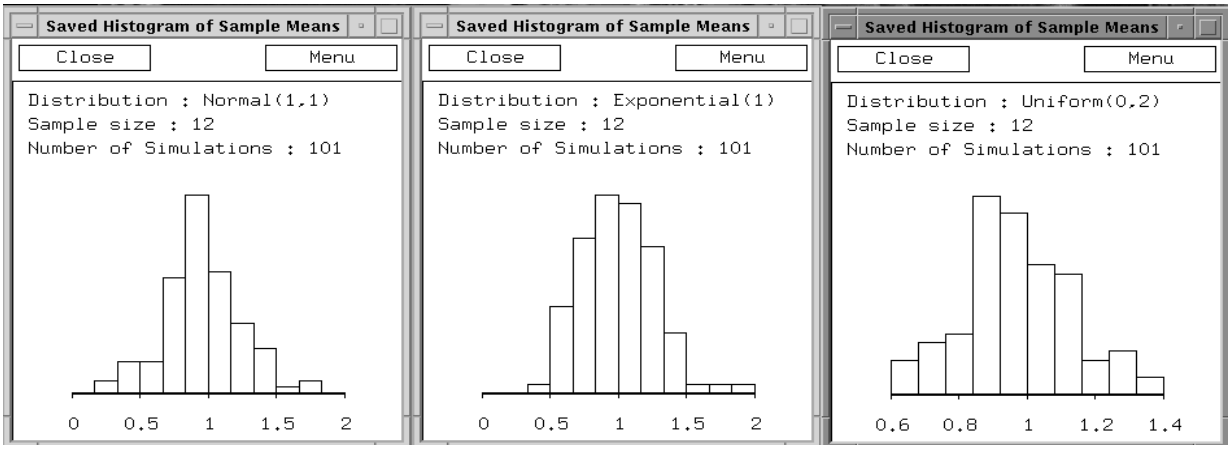

Figure 7: Three Frames of the Central Limit Theorem Software Windows

3.6 Experimenting with sampling distributions

The the sampling distribution (SD) module was designed to allow students to learn about the sampling distributions of various commonly used statistics (e.g., the sample sum, mean, median, standard deviation, variance, range and t-statistic). As such, this software can serve as the basis for a large number of different instructional modules. Some important concepts that can be demonstrated with this software include

- Different statistics have different sampling distributions with shape depending on (a) the specific statistic, (b) the sample size, and (c) the parent distribution.

- For most distributions, the variability in a sampling distribution can be reduced by increasing the sample size.
- In large samples, many sampling distributions can be approximated with a normal distribution.

Figure 8 shows a frame from the sampling distribution software. Choosing the parent distribution, parameters, and sample size is done as in the UDP module (Section 3.2). Here one also chooses a particular statistic to study and low and high sample sizes (an intermediate sample size is automatically chosen as the geometric mean of the low and high values). When the `Start Simulation` button is used, 300 samples are generated for each of the 3 sample sizes. The sample statistics are computed and displayed in histograms. A comparison of the histograms provides information on the effect that sample size has on the shape of the sampling distribution.

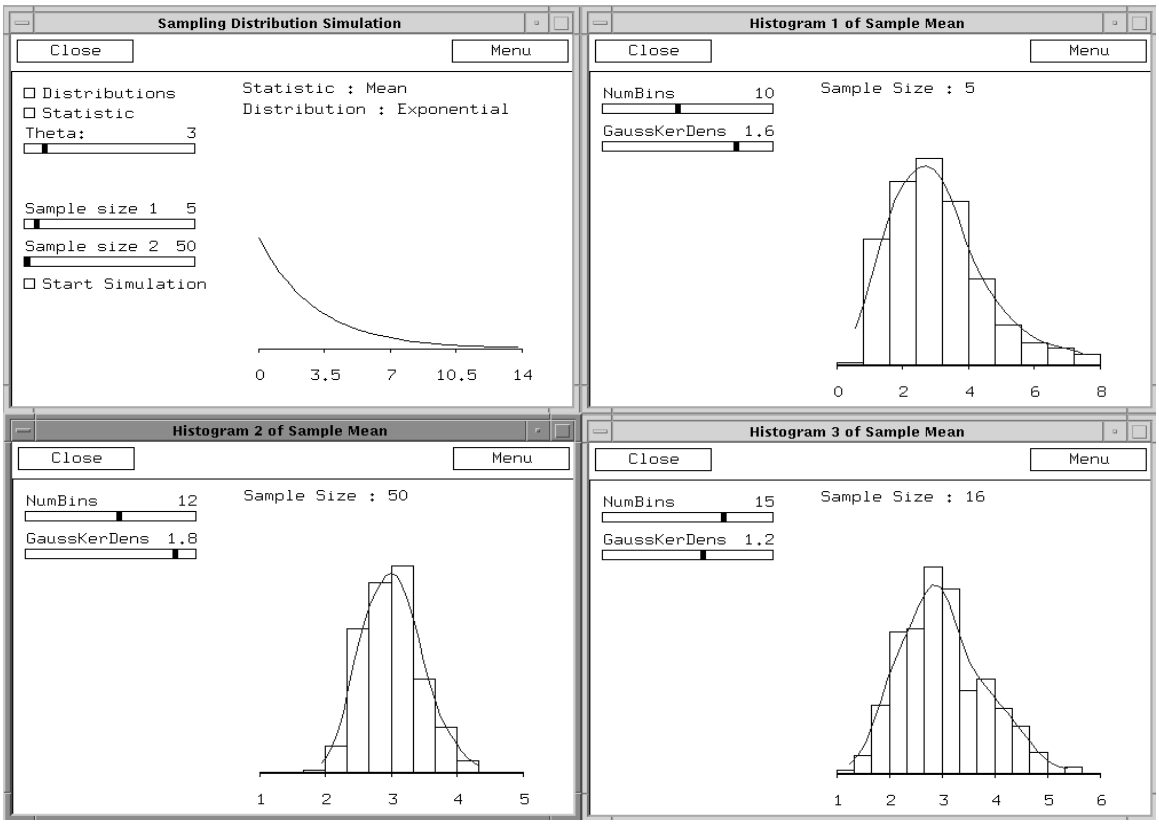

Figure 8: Frame of the Sampling Distribution Software Windows

4 Availability

The Lisp-Stat source code for the software components of our modules is available via anonymous ftp from Iowa State University. To obtain these use the command `ftp las2.iastate.edu`

with "anonymous.mervyn" as the username and "yourusername@your.email.host" as the password. This should get you into the first author's directory named "anonymous". The subdirectory "Teach" contains Readme files describing three versions of the software. If you are accessing files from a Unix host, the shell archive "Teach.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 (Teach.exe for PC/Windows and Teach.sea.hqx for Macs). These files are self-extracting binary archives that unbundle when executed. Other Readme/Install files will be found in each package after unbundling.

Postscript files of the current version of this paper and several instructional module lessons designed for use with the software modules are available in the subdirectory Teach/Docs. We are planning to make the software and lessons available in the future via `statlib`, so that they can also be accessed via electronic mail, ftp, gopher or mosaic.

`Lisp-Stat` is freely available from `umnstat.stat.umn.edu` or from `statlib`. Our software currently runs under version 2.1 Release 2 of `Lisp-Stat` and will be upgraded to run under Release 3 as soon as we have completed testing the software under that version.

5 Extensions and on-going work

This article has focused on the collections of instructional modules that we have developed specifically for our introductory undergraduate statistics service courses. We have also been developing instructional modules based on more advanced statistical concepts. These modules are being developed and used for courses that have as their primary audience advanced undergraduates and graduate students. Most of the software components of these modules have been developed in the Splus language, taking advantage of the powerful collection of graphics, modeling, and analysis functions available in this computing environment. Examples include

- Exploration of the shapes of various response surface models.
- Separating signal from noise when using autocorrelation and partial autocorrelation functions to identify ARIMA models.
- Assessing the effect of sample size and degree of censoring on bias and variance of maximum likelihood estimators in survival analysis applications.

6 Concluding Remarks

The availability of powerful computing hardware and tools to develop instructional software has brought statistics education to a watershed. The way that we teach is changing and we can expect the rate of change to increase. Using a judicious combination of traditional data analysis exercises and new exercises based on carefully designed and implemented simulation-based experiences has the potential of improving the students' level of understanding of statistical methods. We have an opportunity and a challenge to use the new technology to make important improvements in our instructional methods.

Acknowledgments

We would like to thank Bob Stephenson and Mark Kaiser for helpful suggestions on how to develop some of our modules. Luis A. Escobar made a number of helpful comments on an earlier versions of the paper. Computer hardware for this project was funded jointly by NSF Instructional and Lab Improvement Program grant number USE-9250829 and Iowa State University student computing fees. We also appreciate the willingness of Luke Tierney to make Lisp-Stat available without cost. Some parts of our software modules incorporated code kindly provided to us by Sandy Weisberg. Finally, we would like to thank Dean Isaacson, Head of the Department of Statistics and Director of the Statistical Laboratory at Iowa State University, for his encouragement and support of this project.

References

Almond, Russell G. (1992) "`ElToY`: A tool for Education and Elicitation," Documentation for `ElToY` system, distributed through `statlib@stat.cmu.edu`.

Doane, D. P., Mathieson, K., and Tracy, R.L. (1994), "Visualizing and describing the shape of distributions," Paper presented at the Joint Statistical Meetings, Toronto, August 1994.

Gnanadesikan, M., Scheaffer, R.L., and Swit, J. (1987) *The Art and Techniques of Simulation*, Dale Seymour Publications.

Groeneveld, R. A. (1979), *An Introduction to Probability and Statistics*, New York: Dekker.

Saunders, D. J. (1986), "Computer graphics and animations for teaching probability and statistics," *International Journal of Mathematical Education in Science and Technology*, **17**, 561-568.

Tierney, L. (1991), *Lisp-Stat*, Wiley.