Use of R as a toolbox for mathematical statistics exploration

In press, *The American Statistician*

Nicholas J. Horton*
Department of Mathematics
Smith College, Northampton, MA

Elizabeth R. Brown
Department of Biostatistics,
University of Washington, Seattle, WA

Linjuan Qian
Department of Mathematics
Smith College, Northampton, MA

August 25, 2004

*Address for correspondence: Dept of Mathematics, Smith College, Northampton, MA 01063. Phone: 413-585-3688, fax: 413-585-3786, email: nhorton@email.smith.edu. We are grateful to Ken Kleinman and Paul Kienzle for comments on an earlier draft of the manuscript, and for the support provided by NIMH grant R01-MH54693 and a Career Development Fund Award from the Department of Biostatistics at the University of Washington.

Abstract

The R language, a freely available environment for statistical computing and graphics is widely used in many fields. This "expert-friendly" system has a powerful command language and programming environment, combined with an active user community. We discuss how R is ideal as a platform to support experimentation in mathematical statistics, both at the undergraduate and graduate levels. Using a series of case studies and activities, we describe how R can be utilized in a mathematical statistics course as a toolbox for experimentation. Examples include the calculation of a running variance, maximization of a non-linear function, resampling of a statistic, simple Bayesian modeling, sampling from multivariate normal and estimation of power. These activities, often requiring only a few dozen lines of code, offer the student the opportunity to explore statistical concepts and experiment. In addition, they provide an introduction to the framework and idioms available in this rich environment.

Keywords: mathematical statistical education, statistical computing

1 Introduction

The R language (Ihaka & Gentleman 1996, Hornik 2004) is a freely available environment for statistical computing and graphics. It allows the handling and storage of data, supports many operators for calculations, provides a number of tools for data analysis, and features excellent graphical capabilities and a straightforward programming language. R is widely used for statistical analysis in a variety of fields, and boasts a large number of add-on packages that extend the system.

In this paper, we consider the use of R not for analysis, but as a toolbox for exploration of mathematical statistics. Following the approach of Baglivo (1995), we introduce examples to illustrate how R can be used to experiment with concepts in mathematical statistics. In addition to providing insight into the underlying mathematical statistics, these example topics provide an introduction to R syntax, capabilities and idioms and the power of this environment.

We begin by providing some motivation for the importance of statistical computing environments as a component of mathematical statistics education. In section 2, we review the history of R (and its connection to S/S-plus), detail resources and documentation available for the system, and describe an introductory R session. Each of the activities in section 3 include a listing of the R commands and output, and a description to help introduce new syntax, structures and idioms. Finally, we conclude with some overall comments about R as an environment for mathematical statistics education.

Over the past decade, there has been increasing consensus regarding the importance of computing skills as a component of statistics education. Numerous authors have described the importance of computing support for statistics, in the context of numerous curricular initiatives. As an example, Moore (2000) features papers on technology that fall within the mantra of "more data, less lecturing" (Cobb 1991). Curriculum guidelines for undergraduate programs in statistical science from the American Statistical Association *require familiarity with a standard software package and should encourage study of data management and algorithmic problem-solving* (American Statistical Association 2004). While we are fully in agreement with the need for software packages to teach introductory statistics courses, in this paper, we focus on aspects relating to algorithmic problem-solving, at the level of a more advanced course.

We also make a distinction between statistical computing and numerical analysis from the toolbox (or sandbox or testbed) for statistical education that we describe. While there is a crucial role for statisticians with appropriate training in numerical analysis, computer science, graphical interfaces and database design, in our experience relatively few students emerge from undergraduate or graduate statistics programs with this training. While somewhat dated, Eddy, Jones, Kass & Schervish (1987) consider the role of computational statistics in graduate statistical education in the late 1980's. A series of papers in the February 2004 *American Statistician* (Gentle 2004, Monahan 2004, Lange 2004) re-address these issues. Lange (2004) describes computational statistics and optimization courses that "would have been unrecognizable 20 years ago*". Gentle (2004) lists 6 areas of expertise valuable for statisticians:

1. data manipulation and analysis,
2. symbolic computations,
3. computer arithmetic,
4. programming,
5. algorithms and methods, and
6. data structures.

Most of these topics, while quite valuable, often require additional training beyond the standard prerequisites for a mathematical statistics class (Monahan 2004, Gentle 2004, Lange 2004). The use of "rapid prototyping" and "quick and dirty" Monte Carlo studies (Gentle 2004) to better understand a given setting is particularly relevant for mathematics statistics classes (not just statistical computing). For many areas of modern statistics (e.g. resampling based tests,

*though considerable overlap is noted with topics given by Eddy et al. (1987).

Bayesian inference, smoothing, etc.) or for topics that often confuse students (e.g. convergence of sample means, power) it is straightforward for students to implement methods and explore them using only a modest amount of programming. Instructors can help remove hurdles by providing students with sample code and specific instructions for how to use them. While these implementations are often limited and may be less efficient than programs written in environments such as C++, they are typically simpler and more straightforward[†]. It is this use of R as a prototyping environment for statistics exploration that we consider.

We also acknowledge that other approaches to activity based learning are extremely valuable (e.g. simulations using Fathom or Java-applets, see Appendix for more examples), but believe that these environments tap into complementary types of learning, and we do not further consider them.

Finally, we distinguish our work from the excellent efforts of Nolan & Speed (1999), who have proposed a model of extended case studies to encourage and develop statistical thinking. Their text (Nolan & Speed 2000) and website (see Appendix) also utilizes R as an environment for an advanced statistics course, but does so primarily for analysis, not as a testbed to explore methods in mathematical statistics.

2 Background on R

R was initially written by Robert Gentleman and Ross Ihaka of the Statistics Department of the University of Auckland (Ihaka & Gentleman 1996), and is freely redistributable under the terms of the GNU general public license. The environment is quite similar to the S language developed originally by Bell Laboratories and now licensed as S-plus by Insightful Corp. Since the late 1990's R has been developed by a core group of approximately 18 individuals, plus dozens of other contributers. The R project describes the functionality of the environment as:

> an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes:
>
> - an effective data handling and storage facility,
> - a suite of operators for calculations on arrays, in particular matrices,
> - a large, coherent, integrated collection of intermediate tools for data analysis,
> - graphical facilities for data analysis and display either on-screen or on hardcopy, and

[†]We note in passing that R can be set up to call C, C++ or Fortran compiled code, if efficiency is needed.

- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities (R Project 2004)

2.1 Documentation, resources and installation

A huge amount of background information, documentation and other resources are available for R. The CRAN (comprehensive R archive network), found at `http://www.r-project.org` and mirrored worldwide, is the center of information about the R environment. It features screenshots, news, documentation and distributions (both source code and precompiled binaries, for R and a variety of packages).

R's documentation includes internal help systems (see `help()`, `help.start()`, and `help.search()`), printed documentation (particularly "An Introduction to R" [approximately 100 pages], based on a document first written by Bill Venables and David Smith, and the "R Installation and Administration" guide [approximately 30 pages]), a frequently asked questions (FAQ) list (Hornik 2004) and a number of textbooks (see the R project web page for a comprehensive bibliography). Of the books, a classic reference is *Modern Applied Statistics with S* (aka "MASS" or "Venables and Ripley"), now in its fourth edition (Venables & Ripley 2002). Other books on R include Dalgaard (2002), Fox (2002), and Maindonald & Braun (2003). An excellent introduction to programming in R is found in Venables & Ripley (2000). In addition to the textbooks, the R project web page also features pointers to a dozen English language contributed tutorials and manuals, plus a smattering in Spanish, French, Italian and German.

One of the strengths of R is the wide variety of add-on packages (or libraries) that have been contributed by users. These extensions bolster what is available in the base R system. A number of very useful routines are included in the `MASS` library, which is included in the standard R distribution. More than 100 other packages are available for downloading, including code to implement generalized estimating equations (`gee`), signal processing using wavelets (`waveslim`), analysis of complex survey samples (`survey`), and statistical process control (`spc`).

Other resources include *R News*, the newsletter of the R project that is published approximately 3 times per year, and a number of mailing lists. These include R-announce (low volume, for major announcements about R), R-packages (for information regarding new packages), R-help (high volume, dozens of messages per day, for problems and solutions using R) and R-devel (for developers).

2.2 Sample session

We begin with a short sample session in R (see Figure 1), to provide background for the examples we consider in section 3 (though the resources described in section 2.1 are better suited as an introduction for new users). From the Linux, Unix or Mac OSX command line, R can be run directly [line 1]. Alternatively, a graphical user interface version can be initiated through the Start menu (Windows) or Dock (MacOS). Lines 2-14 provide the version, copyright, disclaimer and background information. On lines 15-16, two vectors of length 3 are created (containing the integers 3,4,5 and 1,2,3, respectively). The `ls()` command lists "objects" within R that have been created. We can display the values of the vectors, and perform manipulations of the individual elements (e.g. lines 23-24) for the entire vector. Individual elements can also be manipulated [line 25]. A number of functions are available within R; lines 27-30 show how to calculate the mean and standard deviation. Finally, the `q()` command is used to exit, and the user is prompted whether the objects (e.g. variables) created within the session should be retained for future sessions (as a workspace image).

3 Example scripts

The extensibility and flexibility of R, combined with the collection of tools, makes it attractive as a testbed (the fact that it is freely available on almost all computing platforms helps as well!). One problem, however, is that the environment has a moderately steep learning curve, and as described earlier, can be termed "expert-friendly", i.e. very powerful once mastered, but non-trivial to learn. To attempt to address this issue, and to help motivate learning this environment, we follow the general approach of Baglivo (1995), and describe some problems that arise in mathematical statistics education. The sample code to solve problems in these areas are in most cases quite straightforward, and display the power and versatility of R as a statistical education environment.

All of these scripts are available for download from our website
`http://math.smith.edu/~nhorton/R`. Contributions of additional activities are encouraged.

3.1 Calculation of a running average

We begin with a simple example that displays the convergence (or non-convergence, for certain distributions) of a sample mean. As described earlier, vectors are a fundamental structure in R. Assume that $X_1, X_2, \ldots, X_n$ are independent and identically distributed realizations from some distribution with center μ. We define $\bar{X}_k = \sum_{i=1}^{k} X_i/k$ as the average of the first k observations. Recall that because the expectation of a Cauchy random variable is undefined

Figure 1: Sample R session

```
% R
R : Copyright 2003, The R Development Core Team
Version 1.7.0  (2003-04-16)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type ‘license()’ or ‘licence()’ for distribution details.

R is a collaborative project with many contributors.
Type ‘contributors()’ for more information.

Type ‘demo()’ for some demos, ‘help()’ for on-line help, or
‘help.start()’ for a HTML browser interface to help.
Type ‘q()’ to quit R.
> x <- c(3,4,5)
> y <- 1:3
> ls()
[1] "x" "y"
> x
[1] 3 4 5
> y
[1] 1 2 3
> x+2*y
[1]  5  8 11
> x[2]
[1] 4
> mean(x)
[1] 4
> sqrt(var(x))
[1] 1
> q()
Save workspace image? [y/n/c]: n
```

(Romano & Siegel 1986), the sample mean does not converge to the population parameter. Figure 2 displays the R code to calculate and plot the running average of a sample of Cauchy random variables with center equal to zero. Line 1 defines the function called `runave()`, which

Figure 2: R code to calculate and display running average

```
1  runave <- function(x) {
2        n <- length(x)
3        ret <- rep(0,n)
4        for (k in 1:n) {
5                ret[k] <- mean(x[1:k])
6        }
7        return(ret)
8  }
9  x <- rcauchy(100)
10 plot(runave(x))
11 title("Running average of Cauchy(0,1)")
```

takes a single argument x (a vector). Lines 4-6 consist of a `for` loop to calculate the mean of the first k observations, which are stored in a vector called `ret`. Line 9 generates a sample of 100 independent Cauchy random variables with center 0 and scale 1. For loops are not always the most efficient way to solve a problem in R (but they have the benefit of being simple); a more elegant, efficient but perhaps less clear solution can be implemented in one line:

```
runave <- function(x) { return(cumsum(x))/(1:length(x))}
```

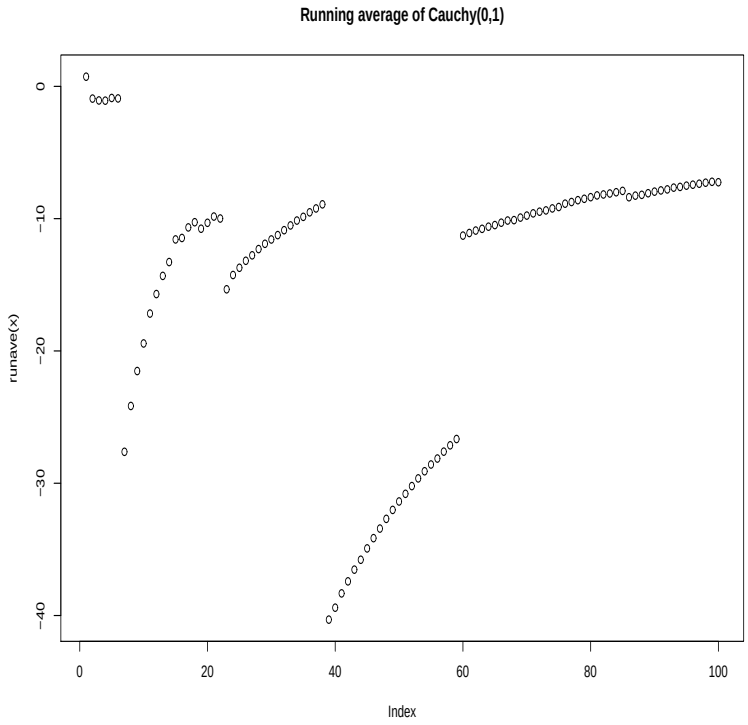
Finally, the returned values are plotted (see Figure 3), and a title is added to the graph [lines 10-11]. It is clear that the mean does not exist; periodically an extreme positive or negative value distorts the statistic.

3.2 Simulating the sample distribution of the mean

It is often useful to conduct simulation studies to explore the behavior of statistics in a variety of settings where analytic solutions may not be tractable (Thisted & Velleman 1992); R provides an extremely flexible environment to carry out such studies. We consider the sampling distribution of the mean of a set of independent and identically distributed exponential random variables. In this setting, this behavior is well-known, but can be implemented in less than a dozen lines of code. More complicated scenarios can be explored using this general framework, with only slight variations.

Figure 4 displays the R code to conduct this simulation 100 (`numsim`) times [line 2] for samples

Figure 3: Running average of the mean of a Cauchy random variables



of size 5, 25, and 100 from an exponential random variable with rate parameter equal to λ. Lines 3 and 4 create vectors of the correct length to store the results, and the `for` loop in lines 5-9 sample from the distribution and store the results. Finally, a boxplot is displayed [line 10], and a title added [line 11]. The boxplots are displayed in Figure 5; the sampling distribution appears to be symmetric when samples are of size 25 or greater, and the distribution is less variable when n=100.

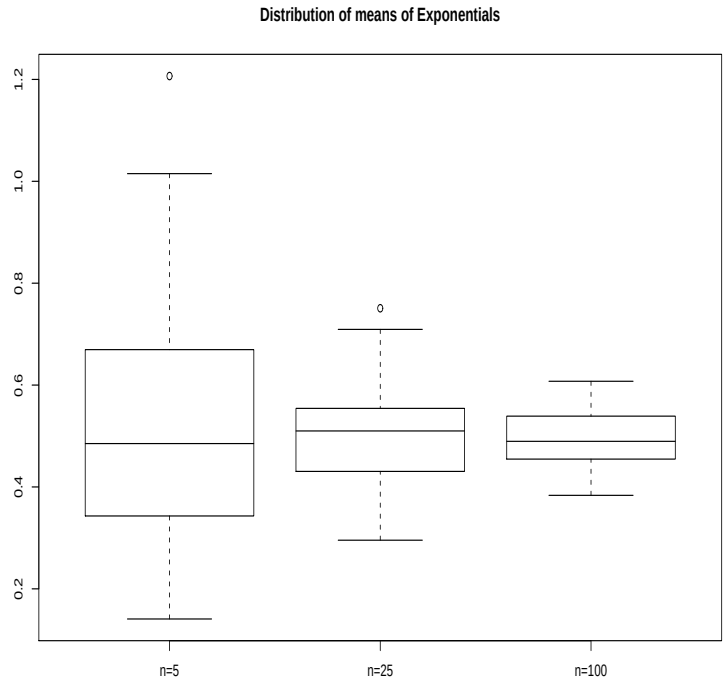
3.3 Sampling from multivariate normal distribution

As described earlier, one of the strengths of R is the facility with which new functions and routines can be prototyped and implemented. As an example, consider sampling from a multivariate normal (MVN) distribution. While routines exist to directly sample from many distributions (see, for example `rcauchy()` from a prior example, plus more commonly utilized functions such as `runif()` [uniform], `rnorm()` [univariate normal], `rf()` [F], `rgamma()` [Gamma], or in the MASS library, `mvrnorm()`), it is straightforward to utilize existing linear algebra functions (e.g.

Figure 4: R code to simulate the sampling distribution of the mean

```
lambda <- 2
numsim <- 100
mean5 <- rep(0,numsim)
mean25 <- mean5; mean100 <- mean5
for (i in 1:numsim) {
    mean5[i] <- mean(rexp(5,lambda))
    mean25[i] <- mean(rexp(25,lambda))
    mean100[i] <- mean(rexp(100,lambda))
}
boxplot(mean5,mean25,mean100,names=c("n=5","n=25","n=100"))
title("Distribution of means of Exponentials")
```

Figure 5: Boxplots of the sampling distribution of the mean



the singular value decomposition(Harville 1997) (chapter 21) to generate MVN samples directly. The function `rmultnorm()`, in Figure 6, (original author unknown) first tests that its arguments

Figure 6: R code for function to generate multivariate normal random variables

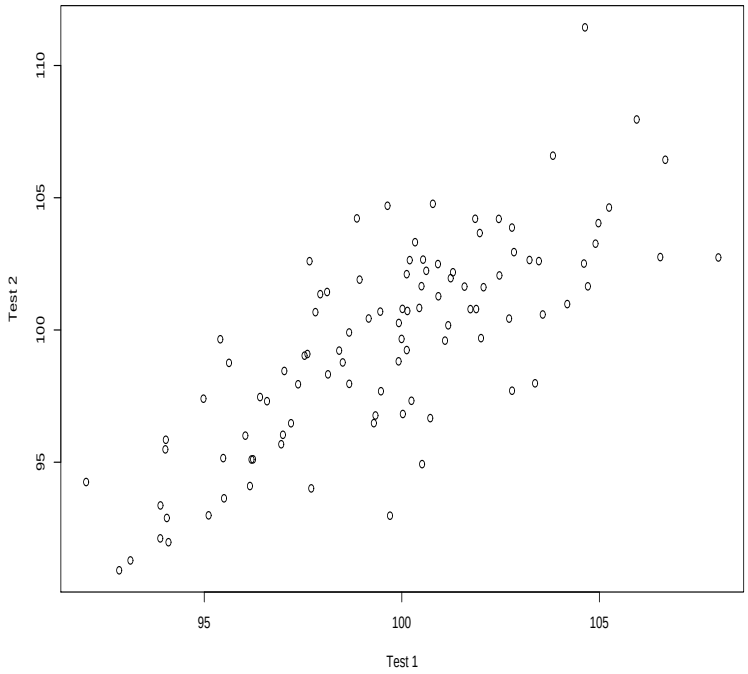
```
1    rmultnorm <- function(n, mu, vmat, tol = 1e-07)
2    # a function to generate random multivariate Gaussians
3    {
4      p <- ncol(vmat)
5       if (length(mu)!=p)
6           stop("mu vector is the wrong length")
7       if (max(abs(vmat - t(vmat))) > tol)
8           stop("vmat not symmetric")
9       vs <- svd(vmat)
10      vsqrt <- t(vs$v %*% (t(vs$u) * sqrt(vs$d)))
11      ans <- matrix(rnorm(n * p), nrow = n) %*% vsqrt
12      ans <- sweep(ans, 2, mu, "+")
13      dimnames(ans) <- list(NULL, dimnames(vmat)[[2]])
14      return(ans)
15   }
16   norms <- rmultnorm(100,c(100,100),matrix(c(14,11,11,14),nrow=2))
17   x1 <- norms[,1]
18   x2 <- norms[,2]
19   plot(x1,x2,xlab="Test 1",ylab="Test 2")
```

are appropriately specified (length of the mean vector equals the dimension of the variance covariance matrix [lines 5-6] and the variance covariance matrix is symmetric [lines 7-8]). The singular value decomposition function returns three objects named `u`, `d`, and `v`, where $vmat = u * d * t(v)$. These values can be used in conjunction with the sweep operator (Goodnight 1979) to generate multivariate normals from a set of $n * p$ univariate normals [lines 9-12]. Note that the `%*%` operator is used for matrix multiplication [line 10] (the `*` operator is used for scalar multiplication), and the `t()` function transposes a matrix. Finally, the `dimnames()` command adds descriptive names to the columns of the matrix (of dimension $n\times$ length(μ)) with the results [line 13]. While this activity might better be suited to a statistical computing rather than mathematical statistics course, it shows off the matrix algebra capabilities of R.

The function is called [on line 16] to generate a set of 100 paired binary outcomes with mean 100, variance 14, and covariance 11 (which corresponds to a correlation of 11/14=0.79). The arguments to the `matrix()` function include a list of values, and an indication of the dimension of the matrix (here it has 2 rows, and because there are 4 entries, the number of columns equals

2). The values in the example were chosen to reflect a set of IQ test results with a mean of 100 and standard deviation of 14. Separate vectors `x1` and `x2` are created from the 100×2 matrix `norms` [lines 17-18] and a scatterplot of the relationship between the two variables is displayed [line 19, see Figure 7].

Figure 7: Plot of bivariate normal random variables



3.4 Power and sample size calculations (analytic)

Power and sample size are confusing concepts for students at all levels, and hands-on activities can be helpful in demystifying their calculation. One approach to power is to determine the precision of an estimate. We consider the width of a 95% confidence interval for a proportion, which is given by $2 * 1.96 * \sqrt{p * (1-p)/n}$, where p is the true proportion and n is the sample size. Figure 8 displays the R code and output for a program that calculates and displays these widths for a set of sample sizes ranging from 100 to 1500 [line 1] and proportions ranging from 0.10 to 0.50 [line 2]. Lines 6-13 display a header for the output file, including use of the `paste` command to turn an R object (in this case the `date()` function) into a string [line 8] and the `cat`

Figure 8: R code to calculate confidence interval width (plus output)

```
1   sampsize <- c(100,300,500,1500)
2   prop <- c(0.10,0.25,0.50)
3   rval <- 2
4   myfile <- "OUTPUT"
5
6   cat("WIDTH OF 95% CONFIDENCE INTERVAL FOR A GIVEN PROP/SAMPLE SIZE\n",
7      file=myfile,append=F)
8   cat(paste(date()),"\n\n",file=myfile,append=T)
9   cat("N=",file=myfile,append=T)
10   for (i in 1:length(prop)) {
11      cat("\t",prop[i],sep="",file=myfile,append=T)
12   }
13   cat("\n\n",file=myfile,append=T)
14
15   for (i in 1:length(sampsize)) {
16      width <- 2*1.96*sqrt(prop*(1-prop)/sampsize[i])
17      cat(sampsize[i],":  ",sep="",file=myfile,append=T)
18      for (j in 1:length(width)) {
19         cat(round(width[j],rval),"\t",sep="",
20            file=myfile,append=T)
21      }
22      cat("\n",file=myfile,append=T)
23   }
```

```
OUTPUT:
WIDTH OF 95% CONFIDENCE INTERVAL FOR A GIVEN PROP/SAMPLE SIZE
Thu May 27 12:16:04 2004

N=    0.10   0.25   0.50

100:  0.12   0.17   0.20
300:  0.07   0.10   0.11
500:  0.05   0.08   0.09
1500: 0.03   0.04   0.05
```

command to output a string [line 6]. For each sample size, the `for` loop on lines 15-23 calculates the width for each of the proportions (note that the object called `width` is a vector with length equal to the length of `prop`). The function `round()` is used to improve the formatting of the results.

We note that R can be used to execute arbitrary functions in the underlying operating system (similar to the use of `date()` on line 8) using the `system()` command. For example, it is straightforward for R to run other packages (e.g. symbolic mathematics programs such as Maple or Mathematica (Baglivo 1995) or any other Unix tool) and read output from those packages).

3.5 Power calculations (empirical)

While formulas to calculate power analytically exist for many settings, there are many situations where such formulas may be intractable or otherwise unappealing. The use of R to simulate data from a posited distribution and empirically calculate power can more realistically model the study design, and may be quite useful in a number of settings. We illustrate this approach using the design of a study of brain volumes in schizophrenics, their unaffected relatives, and a set of normal controls (Seidman, Pantelis, Keshavan, Faraone, Goldstein, Horton, Makris, Falkai, Caviness & Tsuang 2003). Here the investigators were interested in pairwise comparison between brain volumes of schizophrenics and their relatives. One complication is that the patients and relatives were clustered within families. We consider a simplified version of the study of Seidman and colleagues, where there are 50 families with 2 schizophrenic patients, and 1 first-degree relative, and 50 families with 1 schizophrenic patient, and 2 first-degree relatives.

For the comparison of schizophrenics and their relatives, it is straightforward to conduct empirical power studies by simulating from a trivariate normal distribution, after assuming the difference between group means (effect size, if the standard deviations are set to 1), and covariance between observations within a family, then fitting a random effects regression model (Laird & Ware 1982) using the `lme()` function of the `nlme` package of Pinheiro & Bates (2000). Finally, we can summarize the results as the proportion of observed p-values less than $\alpha = 0.05$.

Figure 9 displays the R code and output for this empirical power calculation. Much of the example code involves housekeeping to construct the design matrix (1's indicate schizophrenics and 0's indicate controls [lines 15-16]), and the indicators of family id's (with families 1-50 corresponding to data yielding 2 patients and 1 control, and families 51-100 corresponding to data derived from families with 1 patient and 2 controls [lines 17-18]).

Within the loop, the `rmultnorm()` function, defined in Figure 6 is used to generate trivariate

Figure 9: R code to estimate power empirically

```
 1 library(nlme)
 2 source("rmultnorm.R")
 3
 4 effect <- .3   # effect size
 5 corr <- .4       # intrafamilial correlation
 6 numsim <- 1000
 7
 8 n1fam <- 50 # families with 2 patients and 1 relative
 9 n2fam <- 50 # families with 1 patient and 2 relatives
10
11 vmat <- matrix(c(1,   corr,corr,
12                  corr,1,   corr,
13                  corr,corr,   1),3,3)
14
15 x <- c(rep(1,n1fam),rep(1,n1fam),rep(0,n1fam),
16        rep(1,n2fam),rep(0,n2fam),rep(0,n2fam))
17 id <- c(1:n1fam,1:n1fam,1:n1fam,
18         (n1fam+1:n2fam),(n1fam+1:n2fam),(n1fam+1:n2fam))
19
20 power <- rep(0,numsim)
21
22 for (i in 1:numsim) {
23     cat(i," ")
24     grp1 <- rmultnorm(n1fam, c(effect,effect, 0), vmat)
25     grp2 <- rmultnorm(n2fam, c(effect,0,0), vmat)
26
27     y <- c(grp1[,1],grp1[,2],grp1[,3],
28        grp2[,1],grp2[,2],grp2[,3])
29
30     group <- groupedData(y~x|id)
31     res <- lme(group,random = ~ 1)
32     pval <- summary(res)$tTable[2,5]
33     power[i] <- pval<=0.05
34 }
35
36 cat("\nempirical power for effect size of ",effect,
37     " is ",round(sum(power)/numsim,3),".\n",sep="")
```

normal random variables with the appropriate means and covariances [lines 24-25]. These are strung together into a single vector `y` [lines 27-28] with entries that correspond to the `x` and `id` structures. The `lme` function is called specifying a random intercept model [lines 30-31] and the p-value for the appropriate group comparison is saved [line 32]. If this p-value is less than 0.05 [line 33] then an indicator is set to T (True, or 1). After the loops have run, the sum of these indicators (divided by the number of simulations) is used to estimate the power. For these 1000 simulations, the empirical estimate of power was 0.89. As a comparison, we also calculated the power of a t-test comparing two groups assuming that all observations were independent. A comparison of 150 schizophrenics and 150 relatives with an effect size of 0.3 yields power of 0.74 (see the `power.t.test()` function). In this setting, since we are interested in a comparison where group membership varies within family, accounting for the correlation within families yields smaller standard errors. While analytic formulas to account for correlated outcomes do exist (see for example sections 2.4 and 8.5 of Diggle, Heagerty, Liang & Zeger (2002)), they typically require a number of simplifying assumptions, which may not always be tenable. Use of R for empirical simulation of power is quite handy.

3.6 Bootstrapping of a sample statistic

Bootstrapping is a powerful and versatile approach to parameter estimation (see Efron & Tibshirani (1993) for a readable introduction). The observed data are assumed to be a population, and repeated samples (with replacement) are taken from this population. A summary statistic (e.g. the mean) is calculated for each of these samples, and can be used to estimate the empirical distribution of that statistic. While a number of advanced implementations of bootstrapping are available for R (see for example the `boot` library), it is straightforward to implement this technique directly.

Figure 10 displays the R code and output for a simple simulation to compare $100 * (1 - \alpha)\%$ confidence intervals based on the bootstrap and normal approximations. For this example we take $alpha = 0.10$. Lines 1-7 set up initialization variables needed for the simulation, including a call to the `qnorm()` function to determine the 0.95 quantile of a standard normal random variable (1.6449 in this case). For each of the number of simulations, a set of random exponentials is sampled [line 11], normal approximation 90% confidence intervals are calculated [line 12], and stored in a matrix of dimension $numsim * 2$ [line 13]. Within this loop, another loop runs to generate *numboot* samples with replacement from the sample of exponentials, and the $\alpha/2$ (0.05) and $1-\alpha/2$ (0.95) quantiles are taken from these summary statistics [line 20]. Finally, an empty plot is made to create appropriate axes [lines 22-23], the `matlines()` command is used to add lines [lines 24-25] (a similar command to add points or text exists), and a legend [line 27] is

Figure 10: R code to calculate a bootstrap confidence interval of the mean

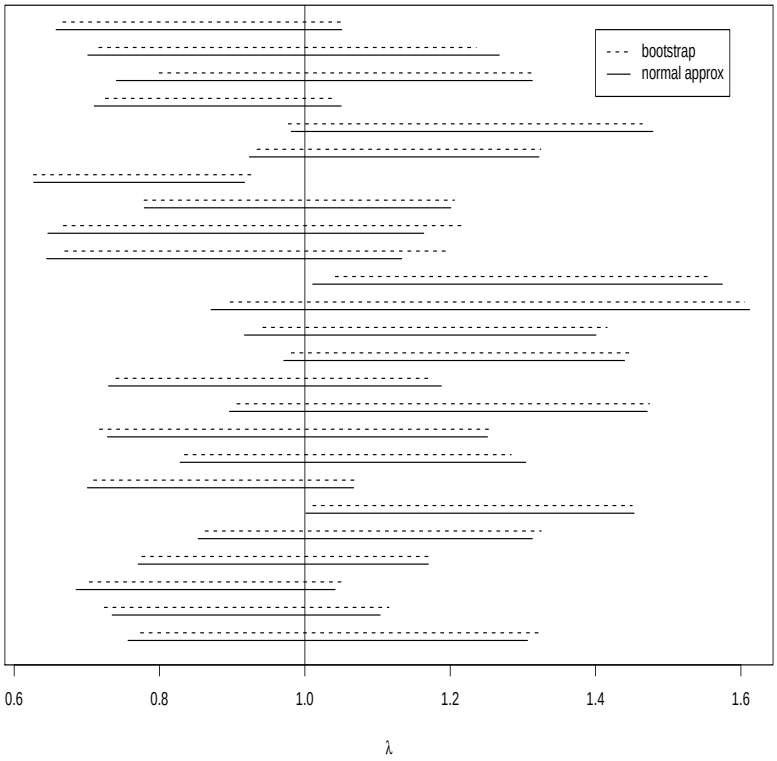
```
alpha <- .10
normval <- qnorm(1-alpha/2)
numsamp <- 50
numsim <- 25
numboot <- 500
bootmat <- matrix(0,nrow=numsim,ncol=2); normmat <- bootmat
y <- 1:numsim; ymat <- rbind(y,y)

for (i in 1:numsim) {
    cat(i," ")
    samp <- rexp(numsamp)    # generate random exponentials
    sampmean <- mean(samp); sampse <- sqrt(var(samp)/numsamp)
    normmat[i,] <- c(sampmean - normval*sampse,sampmean + normval*sampse)

    bootmean <- 0    # initialize this to hold our bootstrap estimates
    for (j in 1:numboot) {
        bootsamp <- sample(samp,numsamp,replace=T)
        bootmean[j] <- mean(bootsamp)
    }
    bootmat[i,] <- quantile(bootmean,c(alpha/2,1-(alpha/2)))
}
matplot(t(bootmat),ymat,pch=" ",yaxt="n",ylab="",
    xlab=expression(paste(lambda))) # empty plot
matlines(t(normmat),ymat,lty=rep(1,numsim),col=1)
matlines(t(bootmat),ymat+.3,lty=rep(2,numsim),col=1)
abline(v=1)
legend(1.4,numsim,legend=c("bootstrap","normal approx"),lty=c(2,1))
```

added. The results from the `matplot()` function are displayed in Figure 11. In this setting,

Figure 11: Normal approximation and bootstrap approximation 90% confidence intervals for exponential ($\lambda = 1, n = 50$)



the performance of the normal approximation and the bootstrap confidence intervals are quite similar.

3.7 Iteration to maximize a likelihood

R is an excellent environment for iterative calculations. Consider, for example, estimation of the unknown parameter θ from a set $X_1, X_2, \ldots, X_n$ of independent and identically distributed realizations from a Weibull density $f(\mathbf{X}|\theta) = \theta\lambda X^{\theta-1} \exp(-\lambda X^{\theta})$, where $\lambda > 0$ is known, and $\theta > 0$. The score function (derivative of the log likelihood with respect to θ) is given by:

$$U(\theta|\mathbf{X}) = \frac{n}{\theta} - \lambda \sum_{i=1}^{n} X_i^{\theta} \ln(X_i) + \sum_{i=1}^{n} \ln(X_i),$$

and observed information (negative of the second derivative of the log likelihood):

$$\hat{I}(\theta|\mathbf{X}) = \frac{n}{\theta^2} - \lambda \sum_{i=1}^{n} X_i^{\theta} \left(\ln(X_i)\right)^2 .$$

The Newton-Raphson algorithm (Thisted 1988) can be used to find the maximum likelihood estimator of θ in this setting, using the iterative procedure:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \frac{U(\hat{\theta_k})}{\hat{I}(\theta_k)},$$

beginning with any $\theta_0 = c > 0$. Figure 12 displays the R code and output for this procedure with a dataset of size 10 [see line 1]. Two functions are defined [lines 4-10], and a `while` loop is used to iterate until convergence is reached.

3.8 ROC curves

A Receiver Operating Characteristic (ROC) curve is a popular tool for assessing the predictive ability of a logistic regression model (Pepe 2000, Murphy 1995). For any cut-off point in the predicted values, one can calculate the sensitivity and specificity associated with that point. For example, assume $Y = 1$ indicates that a subject is diseased, $Y = 0$ indicates no disease, and the expected probability of disease is $E(Y) = p$. If an individual is defined as diseased if their expected probability p is greater than some cut-off value, p_c, then it is straightforward to calculate the sensitivity and specificity associated with p_c. Repeating this for all possible cut-off values provides values for a plot of (1-specificity) against the sensitivity, which is known as the ROC curve.

Figure 13 displays the R code to calculate an ROC curve for a given model. Assume that data is collected on N subjects along with disease status $Y_i, i = 1, \ldots, N$ and a vector of two predictors (X_{1i}, X_{2i}). We begin by generating data from a known logistic regression model (with parameters $\beta_0 = 2, \beta_1 = 0.2$ and $\beta_2 = 1$ [lines 2-8]) using the `sample()` function with the appropriate probabilities.

Using this simulated data, we can fit a logistic regression model [line 9], and then calculate the corresponding fitted values for each subject ($\hat{p}_i, i = 1, \ldots, N$ given $X_i, i = 1, \ldots, N$ [line 10]). We estimate the ROC curve by selecting a vector of unique cutoff values, p_c [line 11]. For each element of p_c, we calculate the specificity and sensitivity [lines 14-17] and then plot the resulting sensitivities against the calculated specificities (subtracted from 1) to yield the estimated ROC curve [lines 18-19]. An ROC curve with slope=1 indicates that the model does

Figure 12: R code to use the Newton-Raphson algorithm to maximize a likelihood mean

```
1 x <- c(0.08, 0.2, 0.4, 0.42, 0.51, 0.67, 0.9, 1.5, 2.7, 4.9)
2 lambda <- 1 # fixed
3
4 U <- function(theta,x,lambda) {
5     ret <- length(x)/theta - lambda*sum(x^theta*log(x)) + sum(log(x))
6 }
7
8 I <- function(theta,x,lambda) {
9     ret <- length(x)/theta^2 + lambda*sum(x^theta*(log(x))^2)
10 }
11
12 tol <- .00001
13 oldtheta <- 0
14 theta <- 2
15 iter <- 1
16 while (abs(oldtheta-theta)>tol) {
17     oldtheta <- theta
18     theta <- theta + U(theta,x,lambda)/I(theta,x,lambda)
19     cat("iter=",iter," theta=",theta,"\n")
20     iter <- iter + 1
21 }

OUTPUT:
iter= 1  theta= 1.374588
iter= 2  theta= 0.9461312
iter= 3  theta= 0.8824284
iter= 4  theta= 0.8830516
iter= 5  theta= 0.8830517
```

not discriminate between diseased and non-diseased individuals, and the area under this ROC curve equals 0.5. If the area under the ROC curve is greater than 0.5, the model discriminates diseased from non-diseased. If it is less than 0.5, the model will assign diseased status to non-diseased individuals. For this reason, the area under the ROC curve can be a useful statistic for assessing the predictive ability of a binary regression model. We estimate the area using the trapezoidal rule [line 21]. The estimated ROC curve is displayed in Figure 14; the area under the curve in this example was 0.62.

Figure 13: R code to calculate the area under the curve (AUC) for a receiver operating characteristic (ROC) curve

```
N <- 250
x1 <- runif(N,0,1)
x2 <- sample(c(0,1),N,prob=c(.5,.5),replace=T)
pr <- exp(2-0.2*x1+x2)/(1+exp(2-0.2*x1+x2))
Y <- rep(0,N)
for (i in 1:N) {
    Y[i] <- sample(c(0,1),1,prob=c(1-pr[i],pr[i]))
}
glm1 <- glm(Y ~ x1+x2,family=binomial)
phat <- fitted(glm1)
pc <- sort(unique(phat))
n2 <- length(pc)
m2 <- matrix(0,nrow=n2,ncol=2)
for (i in 1:n2) {
    t1 <- table(factor(phat>pc[i],levels=c(T,F)),Y)
    m2[i,] = c((t1[1,2])/sum(t1[,2]),1-t1[2,1]/sum(t1[,1]))
}
plot(m2[,2],m2[,1], type='l',xlim=c(0,1),ylim=c(0,1), xlab='1-specificity',
    ylab='sensitivity')
abline(0,1)
AUC <- sum((m2[1:(n2-1),1]+m2[2:n2,1])/2*(-m2[2:n2,2]+m2[1:(n2-1),2]))
mytitle <- paste("ROC CURVE (AUC=",paste(round(AUC,3)),")",sep="")
title(mytitle)
```

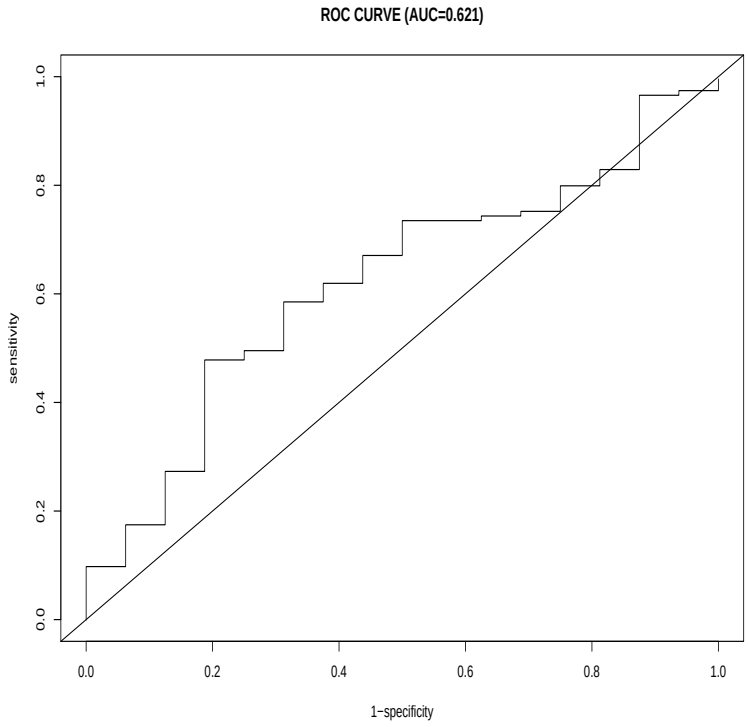
It is straightforward to use the bootstrap to estimate the standard error of the AUC; the code to implement this is included on the examples website (see Appendix).

3.9 EM algorithm

The EM (Expectation-Maximization) algorithm (Dempster, Laird & Rubin 1977, McLachlan & Krishnan 1997) is a powerful and elegant framework for estimation of incomplete data problems. While this approach is typically not taught in an introductory mathematical statistics course, it is included in some texts on the subject (e.g. Casella & Berger (2002)). The general technique is to calculate the expected values for missing objects, given current parameter estimates (Expectation step), then to use those expected values to find new parameter estimates (Maximization step). These steps are then repeated until convergence.

We consider the application of this algorithm to estimation of the parameters of a mixture

Figure 14: Plot of estimated ROC curve



distribution of two normals (Titterington, Smith & Makov 1985):

$$f(y|\boldsymbol{\theta}) = \pi f_1(y|\mu_1, \sigma_1) + (1 - \pi) f_0(y|\mu_0, \sigma_0),$$

where $\boldsymbol{\theta} = (\mu_1, \mu_0, \sigma_1, \sigma_0, \pi)$, and $f_1()$ and $f_0()$ are univariate normals. While problems involving mixtures of distributions may not at first blush appear to be a "missing-data" problem, the EM algorithm is quite natural. If one knew the parameters of the two distributions, and the mixture probability, it is straightforward to calculate the expected values. Given the expected values, it is easy to find new parameter estimates.

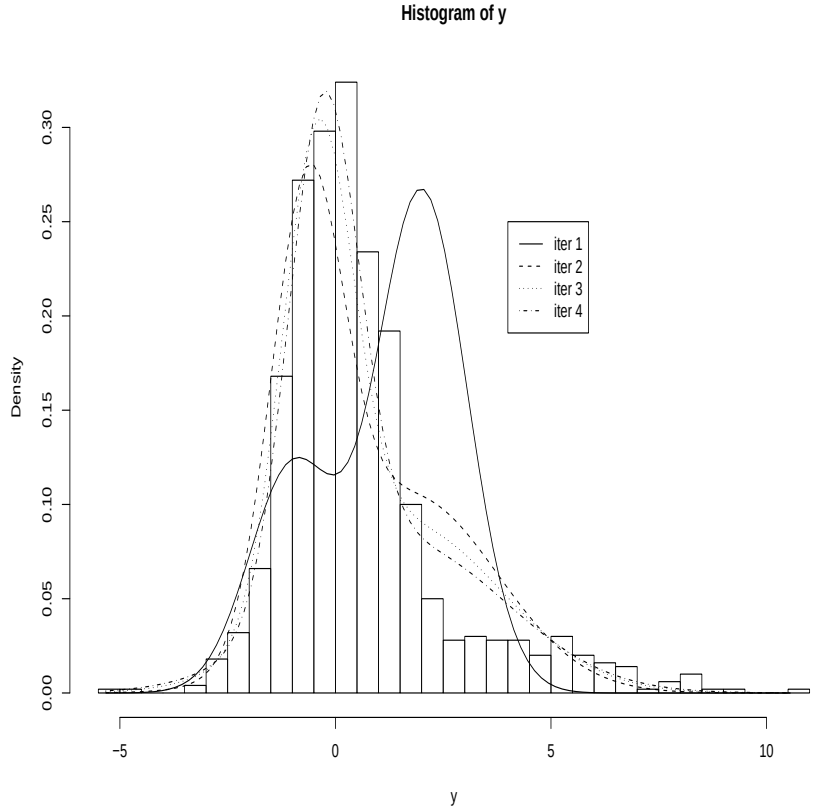
Figure 15 displays the R code that implements a new function `mixnorm()`, which takes as arguments the observed values from the mixture distribution, and a set of starting values. After some initialization [lines 3-8], the main routine is repeatedly called (until convergence, which is defined on line 9). The expectation step is implemented in lines 14-16, while the maximization

Figure 15: R code to use the EM algorithm to find MLE's of the parameters for a mixture of two normal distributions

```
1    mixnorm <- function(y, mu0, mu1, var0, var1, prob, eps = 1/100000) {
2    # y are assumed to be a mixture of two normal distributions
3      new.params <- c(mu0, mu1, var0, var1, prob)
4      err <- 1
5      iter <- 1
6      maxiter <- 4
7      hist(y,probability=T,nclass=30)
8      xvals <- seq(from=min(y),to=max(y),length=100)
9      while(err > eps) {
10       if (iter <= maxiter) {
11          lines(xvals,prob*dnorm(xvals,mu1,sqrt(var1))+
12             (1-prob)*dnorm(xvals,mu0,sqrt(var0)),lty=iter)
13       }
14       bayes <- (prob * dnorm(y, mu1, sqrt(var1)))/((prob * dnorm(y,
15           mu1, sqrt(var1))) + ((1 - prob) * dnorm(y, mu0, sqrt(
16           var0))))
17       mu1 <- sum(bayes * y)/sum(bayes)
18       mu0 <- sum((1 - bayes) * y)/sum((1 - bayes))
19       var1 <- sum(bayes * (y - mu1)^2)/sum(bayes)
20       var0 <- sum((1 - bayes) * (y - mu0)^2)/sum((1 - bayes))
21       prob <- mean(bayes)
22       old.params <- new.params
23       new.params <- c(mu0, mu1, var0, var1, prob)
24       err <- max(abs((old.params - new.params)/new.params))
25       iter <- iter + 1
26     }
27   legend(4,.25,legend=c("iter 1","iter 2","iter 3","iter 4"),lty=1:4)
28   return(list(mu = c(mu0, mu1), v = c(var0, var1), p = prob))
29   }
30
31   y <- c(rnorm(800,0,1),rnorm(200,3,3))
32
33   vals <- mixnorm(y,-1,2,1,1.1,.7)
```

(given those conditional expectations) is found on lines 17-21. For this problem, we set $\pi = 0.8$, $\boldsymbol{\mu} = (0, 3)$, and $\boldsymbol{\sigma} = (1, 3)$, and simulated 1000 normals. Figure 16 displays the (sometimes) slow convergence of the algorithm as it proceeds from the arbitrary starting values ($\boldsymbol{\mu} = (-1, 2)$, $\boldsymbol{\sigma} = (1, 1.1)$ and $\pi = .7$ [line 33]). The `mixnorm()` function returns a list of values, the names of

Figure 16: Convergence of the EM algorithm



which can be found by use of a call to `names(vals)`.

3.10 Bayesian inference

Our catalog of examples would be incomplete without a mention of Bayesian inference (Gelman, Carlin, Stern & Rubin 1995), a topic that is now often included in a modern mathematical statistics course (see for example Casella & Berger (2002) or Rice (1995)). While this example is by far our most complicated, it is also amongst the most powerful, and allows students (as well as instructors!) to visualize the relative contributions of various priors and the likelihood

to help clarify the Bayesian approach.

We consider the binomial likelihood with a beta prior on the probability p. This is often used as an example (see Rice (1995) section 15.3.2) because this is a family of conjugate priors and an analytic form for the posterior distribution exists. Numerical integration and/or Gibbs sampling is needed in situations where conjugate priors do not exist, and we describe a simple setting and provide R code on our website.

Figure 17 displays the R code to explore the posterior of a binomial likelihood with a beta prior on the probability, p. We consider a sample of size 50 [line 1], and set the pseudo-random number seed to a fixed value (Adams 1980) to allow results to be replicated. The function `postbetbin()` [lines 4-6] defines the posterior distribution as the product of the binomial likelihood and the beta prior. The functions `lbinom()` and `dbetas2()` [lines 7-8] redefine the binomial and beta probability functions as functions of their parameters instead of functions of the data. Lines 12-16 define four sets of parameters for four beta distributions. The first prior [line 13] is non-informative and puts equal weight on all values of p, line 14 reflects a strong prior belief that $p = 0.5$, line 15 is a moderate prior that $p = 0.5$, and line 16 reflects a belief that less than half the population has the trait $p < 0.5$. Lines 20-35 plot the resulting posteriors for the four priors along with the priors and the likelihood. These are plotted on two different scales so that the features in each density function are more obvious to the viewer. The result is shown in Figure 18.

This example, which addressing a very simple setting, can easily be extended by students to explore the effect of other prior distributions on the posterior, such as a mixture of betas or a truncated normal or gamma. It can also be used to explore the effect of the prior distribution on the posterior distribution when the sample size changes.

4 Discussion

In this paper, we have described how the R environment can be used as a testbed for exploration of concepts in mathematical statistics education. This exploration can help students lay the groundwork for future use of R as a foundation for more complex computational statistical research and development, as well as the more standard use of R as an analysis package. Hands-on activities are an important component of experiential education, and the flexibility of this environment, combined with the rich variety of lower-level routines, facilitates this type of learning. In addition to providing insights into advanced concepts in mathematical statistics, experiences with R provide a basis for later implementation of novel methodologies. We believe

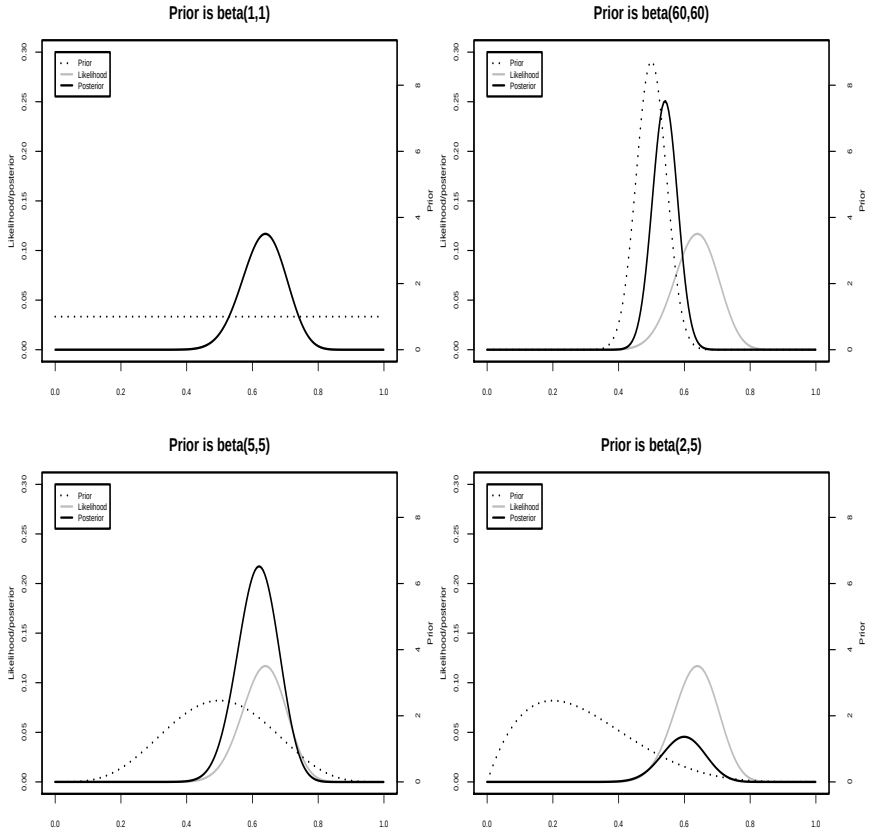
Figure 17: R code to explore posterior distribution of a binomial likelihood with a beta prior on the probability

```
N <- 50
set.seed(42)
Y <- sample(c(0,1),N,pr=c(.2,.8),replace=T)
postbetbin <- function(p, Y, N, alpha, beta) {
    return(dbinom(sum(Y),N,p)*dbeta(p,alpha,beta))
}
lbinom <- function(p,Y,N) dbinom(Y,N,p)
dbeta2 <- function(ab,p) return(unlist(lapply(p,dbeta,shape1=ab[1],shape2=ab[2])))
lines2 <- function(y,x,...) lines(x,y[-1],lty=y[1],...)

x=seq(0,1,l=200)
alphabeta=matrix(0,nrow=4,ncol=2)
alphabeta[1,]=c(1,1)
alphabeta[2,]=c(60,60)
alphabeta[3,]=c(5,5)
alphabeta[4,]=c(2,5)
labs <- c("beta(1,1)", "beta(60,60)", "beta(5,5)", "beta(2,5)")
priors <- apply(alphabeta, 1, dbeta2, p=x)

par(mfrow=c(2,2),lwd=2,mar=rep(3,4) ,cex.axis=.6)
for(j in 1:4){
  plot(x,unlist(lapply(x,lbinom,Y=sum(Y),N=N)), type="l",
     xlab="p", col="gray",ylab="", main=paste("Prior is",labs[j]),
     ylim=c(0,.3))
  lines(x, unlist(lapply(x,postbetbin,Y=sum(Y),N=N,
      alpha=alphabeta[j,1],beta=alphabeta[j,2])),lty=1)
  par(new=T)
  plot(x,dbeta(x,alphabeta[j,1],alphabeta[j,2]),lty=3,axes=F,
     type='l', xlab="", ylab="",ylim=c(0,9))
  axis(4)
  legend(0, 9, legend=c("Prior", "Likelihood", "Posterior"),
       lty=c(3, 1, 1), col=c( "black", "gray", "black"),cex=.6)
  mtext("Prior", side=4, outer=F, line=2, cex=.6)
  mtext("Likelihood/posterior", side=2, outer=F, line=2, cex=.6)
}
```

Figure 18: Posterior distributions for a set of 4 different prior distributions



that this type of teaching environment helps students to *address and solve problems that he or she might encounter in the real world, whether or not they are isomorphic to problems seen in class (page 46)* (Thisted & Velleman 1992).

The examples that we present are not intended to provide a gentle introduction to the use of R, though such tutorials exist (see for example the excellent Appendix A of *An introduction to R* or chapters 1 and 2 of Venables & Ripley (2002)). Instead, we hope to present the possibilities of its use on a series of interesting problems.

We have focused on R rather than S-plus, since most code written for one system works on the other, and R has the advantage of being freely redistributable. While there are aspects of S-plus that are quite attractive for students (in particular, the graphical user interface), this is less

pertinent for the application we discuss.

Symbolic mathematics packages such as Maple (Baglivo 2005) or Mathematica also provide a reasonable exploration environment. Other platforms have the capability to be used in a similar fashion. For example, Matlab, Octave, Gauss and SAS/IML (see Appendix) are relatively high-level languages intended for numerical computations. Moler (2004) is an example of a text which shares our philosophy of providing examples (in his case for a numerical methods course implemented using Matlab) that students can explore, modify and extend. While these programs offer a wide variety of routines, they are more general purpose numerical engines than R, and tend to have less support for certain statistical functions.

The flexible programming environment of R is both a plus and a minus as a teaching tool. Students with no prior experience with computer programming may find the environment uncomfortable at first. The provision of sample code (the median length of our examples was 23 lines) combined with the existence of built-in functions and routines minimize the programming requirements and anxiety-level while still providing a rich toolset for experimentation.

5 Appendix

The following list of websites, while not comprehensive, is intended to provide a useful starting point for exploring R and related tools.

5.1 R and S-plus resources

http://www.r-project.org The home of R: this is the main location for distributions, packages, and documentation.

http://math.smith.edu/~nhorton/R This link by the authors provides a repository for sample code and activities useful in teaching mathematical statistics concepts using R.

http://www.insightful.com/products This link is the home of S-plus, a commercial version of S sold by Insightful Corporation. A number of very useful add-on modules provide support for missing data modeling, wavelets, spatial statistics and design of experiments.

http://www.stat.umn.edu/~galin/teaching/Rstuff The site provides a general introduction to R. One attractive feature for those exploring use of R is that it provides a web interface to run R on their server.

http://www.stats.ox.ac.uk/pub/MASS4 This link contains resource materials (example code, answers to exercises, etc.) for Venables and Ripley's classic book *Modern applied statistics with S*, now in its fourth edition. It provides an introduction to S/R as well as a course in modern statistical methods.

http://www.stats.ox.ac.uk/pub/MASS3/Sprog The site is a source for background materials for Venables and Ripley's text *S Programming* which provides a comprehensive introduction to programming in S and R. This book is particularly well-suited for those who like to spelunk in the innards of complex computing environments.

http://socserv.mcmaster.ca/jfox/Books/Companion This link connects to the companion website to Fox's *R and S-plus companion to applied regression* book.

http://stat-www.berkeley.edu/users/statlabs Materials related to the book *Stat labs: mathematical statistics through applications* (Nolan & Speed 1999), which teaches the theory of statistics through a collection of extended case studies using R, are found at this link.

5.2 Other packages and procedures

http://www.stat.sc.edu/rsrch/gasp This site is the home of the GASP (globally accessible statistical procedures initiative). It features a number of Java applets and web pages related to data analysis and statistical education.

http://www.keypress.com/fathom Fathom is a statistical education tool that allows students to understand concepts by dynamic manipulation and simulation.

http://www.gnu.org This is the home of the GNU (Gnu's Not Unix) project and the FSF (Free Software Foundation). The site includes a large number of free software packages, including R and Octave, that are available for distribution under the GNU GPL (general public license). As of July, 2004, 3,316 packages were indexed on the site.

http://www.mathworks.com/products/matlab Matlab is a commercial high-level technical computing environment.

http://www.octave.org This is the home of the GNU Octave language for numerical computations that is mostly compatible with Matlab.

http://www.sas.com/technologies/analytics/statistics/iml The site provides information on the commercial SAS/IML interactive matrix programming environment.

http://www.aptech.com The official site of the GAUSS system, a data analysis, mathematical and statistical matrix environment.

References

Adams, D. (1980). *The hitchhiker's guide to the galaxy*, Harmony Books.

American Statistical Association (2004). Curriculum guidelines for undergraduate programs in statistical science, *http://www.amstat.org/education/index.cfm?fuseaction=Curriculum_Guidelines, date accessed 6/01/2004* .

Baglivo, J. (1995). Computer algebra systems: Maple and Mathematica, *The American Statistician* **49**(1): 86–92.

Baglivo, J. (2005). *Mathematical laboratories for mathematical statistics: emphasizing simulation and computer intensive methods*, ASA-SIAM Series on Statistics and Applied Probability 14.

Casella, G. & Berger, R. L. (2002). *Statistical Inference, 2nd edition*, Duxbury.

Cobb, G. (1991). Teaching statistics: More data, less lecturing, *UME Trends* pp. 3–7.

Dalgaard, P. (2002). *Introductory Statistics with R*, Springer.

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, Series B* **39**(1): 1–22.

Diggle, P. J., Heagerty, P., Liang, K. Y. & Zeger, S. L. (2002). *Analysis of Longitudinal Data, second edition*, Clarendon Press.

Eddy, W. F., Jones, A. C., Kass, R. E. & Schervish, M. J. (1987). Graduate education in computational statistics, *The American Statistician* **41**: 60–64.

Efron, B. & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*, Chapman & Hall, New York.

Fox, J. (2002). *An R and S-Plus Companion to Applied Regression*, Sage Publications, Thousand Oaks, CA, USA.

Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (1995). *Bayesian data analysis*, Chapman and Hall.

Gentle, J. E. (2004). Courses in statistical computing and computational statistics, *The American Statistician* **58**(1): 2–5.

Goodnight, J. H. (1979). A tutorial on the SWEEP operator, *The American Statistician* **33**: 149–158.

Harville, D. (1997). *Matrix algebra from a statistician's perspective*, Springer, New York.

Hornik, K. (2004). The R FAQ, *http://www.ci.tuwien.ac.at/∼hornik/R/, accessed June 2, 2004* .

Ihaka, R. & Gentleman, R. (1996). R: A language for data analysis and graphics, *Journal of Computational and Graphical Statistics* **5**(3): 299–314.

Laird, N. M. & Ware, J. H. (1982). Random-effects models for longitudinal data, *Biometrics* **38**: 963–974.

Lange, K. (2004). Computational statistics and optimization theory at UCLA, *The American Statistician* **58**(1): 9–11.

Maindonald, J. & Braun, J. (2003). *Data analysis and graphics using R: an example-based approach*, Cambridge University Press.

McLachlan, G. J. & Krishnan, T. (1997). *The EM Algorithm and Extensions*, Wiley-Interscience.

Moler, C. B. (2004). *Numerical computing with Matlab*, SIAM.

Monahan, J. (2004). Teaching statistical computing at North Carolina State University, *The American Statistician* **58**(1): 6–8.

Moore, T. L. (2000). *Teaching statistics: resources for undergraduate instructors (MAA Notes #52)*, Mathematical Association of America and the American Statistical Association.

Murphy, J. M. (1995). Diagnostic schedules and rating scales in adult psychiatry, *Textbook in Psychiatric Epidemiology*, John Wiley & Sons, pp. 253–271.

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

Nolan, D. & Speed, T. P. (1999). Teaching statistics theory through applications, *The American Statistician* **53**: 370–375.

Pepe, M. S. (2000). Receiver operating characteristic methodology, *Journal of the American Statistical Association* **95**(449): 308–311.

Pinheiro, J. C. & Bates, D. M. (2000). *Mixed-effects models in S and S-PLUS*, Springer-Verlag Inc.

R Project (2004). What is R?, *http://www.r-project.org/about.html/, accessed June 2, 2004* .

Rice, J. A. (1995). *Mathematical statistics and data analysis*, Duxbury.

Romano, J. P. & Siegel, A. F. (1986). *Counterexamples in probability and statistics*, Duxbury Press.

Seidman, L. J., Pantelis, C., Keshavan, M. S., Faraone, S. V., Goldstein, J. M., Horton, N. J., Makris, N., Falkai, P., Caviness, V. S. & Tsuang, M. T. (2003). A review and new report of medial temporal lobe dysfunction as a vulnerability indicator for schizophrenia: a magnetic resonance imaging morphometric family study of the parahippocampal gyrus, *Schizophrenia Bulletin* **29**(4): 803–830.

Thisted, R. A. (1988). *Elements of statistical computing*, Chapman & Hall Ltd.

Thisted, R. A. & Velleman, P. F. (1992). *Computers and modern statistics*, Mathematical Association of America, MAA Notes number 21, pp. 41–53.

Titterington, D. M., Smith, A. F. M. & Makov, U. E. (1985). *Statistical analysis of finite mixture distributions*, John Wiley & Sons.

Venables, W. N. & Ripley, B. D. (2000). *S Programming*, Springer.

Venables, W. N. & Ripley, B. D. (2002). *Modern Applied Statistics with S. Fourth Edition*, Springer.