The Use of R Language in Mathematics Teaching and Computation

Cheang Wai Kwong
wkcheang@nie.edu.sg
National Institute of Education
Nanyang Technological University
Singapore

Abstract: Computer Algebra Systems (CAS's) like Maple are commonly used in mathematics teaching and computation. The strength of these CAS's lies in their symbolic functionality. However, these CAS's are not free and may not be cost-effective to implement. In this paper, we consider the potential of the R language as a "free" substitute for CAS when symbolic computation is not required. The R language is a powerful environment for data analysis and graphics within which many statistical techniques have been implemented. It is an official part of the Free Software Foundation's GNU Project. Being a statistical software, R is not capable of performing symbolic calculations. The strength of R derives from its powerful computing capability and flexible graphing facility. With its ease of adaptability according to user's need, R has the potential to be an effective teaching and computation tool. We explore the following capabilities of R in mathematics teaching and computation: (i) as a programming language; (ii) as a graphing tool; (iii) as a statistical software.

1. Introduction

Computer Algebra Systems (CAS's) like Mathematica, Matlab and Maple are commonly used in mathematics teaching and computation. Ang and Awyong (1999) described their approach in incorporating Maple into first year undergraduate mathematics. The strength of these CAS's lies in their symbolic functionality. However, these CAS's are not free and may not be cost-effective to implement. In this paper, we consider the potential of the R language as a "free" substitute for CAS when symbolic computation is not required.

The R language is a powerful environment for data analysis and graphics within which many statistical techniques have been implemented. R can be considered as a "free" implementation of the S language which was originally developed at Bell Laboratories (of AT&T and now Lucent Technologies). A commercial implementation of S is S-PLUS from the Insightful Corporation in Seattle. R is an official part of the Free Software Foundation's GNU Project, and is distributed as Free Software under the terms of the GNU General Public License. The development of R is supported by the R Foundation seated in Vienna. R compiles and runs on a wide variety of Unix platforms (including Linux), Windows and MacOS. Precompiled R binaries as well as the R source code can be downloaded from http://www.r-project.org/.

Being a statistical software, R is not capable of performing symbolic calculations like indefinite integration. The strength of R derives from its powerful computing capability and flexible graphing facility. With its ease of adaptability according to user's need, R has the potential to be an effective teaching and computation tool. We explore the following capabilities of R in mathematics teaching and computation: (i) as a programming language; (ii) as a graphing tool; (iii) as a statistical software.

2. R as a programming language

R is a simple and effective programming language which includes conditionals, loops and user-defined functions. So instead of CAS's like Maple, R can be used as a programming tool in a numerical analysis course for students to implement the numerical methods learned. For example, consider the bisection method implemented using Maple in Ang (2002, p. 45) to solve $x^3 - x - 1 = 0$ in the interval [1, 2], accurate to within 0.001. The corresponding R code is given in Appendix A.1, with the following output:

Table 1 The bisection method implemented using R to solve $x^3 - x - 1 = 0$.

```
 n         a       b  c=(a+b)/2   d=|b-c|     f(c)
 1   1.00000 2.00000    1.50000   0.50000  0.87500
 2   1.00000 1.50000    1.25000   0.25000 -0.29688
 3   1.25000 1.50000    1.37500   0.12500  0.22461
...
 8   1.32031 1.32812    1.32422   0.00391 -0.00213
 9   1.32422 1.32812    1.32617   0.00195  0.00621
10   1.32422 1.32617    1.32520   0.00098  0.00204
```

With its capability to perform matrix computation, R can also be used in a linear algebra course. The "**solve**" function in R can solve a linear system or invert a matrix numerically, and other functions like "**eigen**" can be used to verify and investigate results in matrix algebra. For example, we can use the following R code to verify the result that for a $n \times n$ matrix A with eigenvalues $\lambda_1, \ldots, \lambda_n$, $\text{tr}(A) = \sum_{i=1}^{n} \lambda_i$ and $\det(A) = \prod_{i=1}^{n} \lambda_i$:

```
lambda <- eigen(A)$values  # Assign the vector of eigenvalues to lambda
sum(lambda)
sum(diag(A))
prod(lambda)
det(A)
```

In teaching the concept of "limits" in calculus, we can certainly use R to demonstrate numerically standard limits such as $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$. In research, we may encounter some non-standard limits. For example, to find

$$\kappa = \sum_{k=2}^{\infty} \frac{1}{k^2} \sum_{j=1}^{k-1} \frac{1}{j} \quad (\approx 1.201)$$

in Cheang and Reinsel (2003), we can use R to compute an approximation to the series by taking a large partial sum:

```
n <- 10000    # Sum to n terms
kappa <- 0
for (k in 2:n)
{ j <- 1:(k-1)
  kappa <- kappa + sum(1/j)/k^2
}
print(kappa)
```

R can also be used as a programming tool to evaluate special functions. For example, consider the hypergeometric function $F(a,b;c;z)$ defined by

$$F(a,b;c;z) = 1 + \frac{ab}{c\cdot 1}z + \frac{a(a+1)b(b+1)}{c(c+1)\cdot 1\cdot 2}z^2 + \cdots = \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)}\sum_{k=0}^{\infty}\frac{\Gamma(a+k)\Gamma(b+k)}{\Gamma(c+k)k!}z^k.$$

Based on the integral representation [e.g., Gradshteyn and Ryzhik (1994, p. 1066)]

$$F(a,b;c;z) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int_0^1 x^{b-1}(1-x)^{c-b-1}(1-zx)^{-a}\,dx, \qquad \mathrm{Re}(c) > \mathrm{Re}(b) > 0,$$

where $\Gamma(\cdot)$ is the gamma function, we can use the numerical integration function "**integrate**" in R to evaluate $F(a,b;c;z)$ if the arguments are real. The user's defined R functions to evaluate

$$F(a,1;b;z) = (b-1)\int_0^1 (1-x)^{b-2}(1-zx)^{-a}\,dx, \qquad a,b,z \text{ are real and } b>1,$$

are as follows:

```
HGM1 <- function(x,a0,b0,z0)
{ (((1-x)/(1-z0*x))^(b0-2))*((1-z0*x)^(b0-2-a0))     # Integrand
}
HGMF1 <- function(a,b,z)
{ h <- integrate(HGM1,lower=0,upper=1,a0=a,b0=b,z0=z)
  h <- (b-1)*h$value
  h
}
```

3. R as a graphical tool

One attractive feature of R is its flexible graphing facility and the ease with which publication-quality plots can be produced. In teaching functions and graphs, R is a useful graphing tool for plotting elementary functions such as polynomials and exponential, and also for solving equations using graphical method. The R code to solve $x^2 = e^x$ using graphical method is given in Appendix A.2. We vary the value of the estimated root x_0 in Figure 1, and observe how the values of $f(x_0) = x_0^2$ and $g(x_0) = e^{x_0}$ and the vertical line $x = x_0$ are changing.

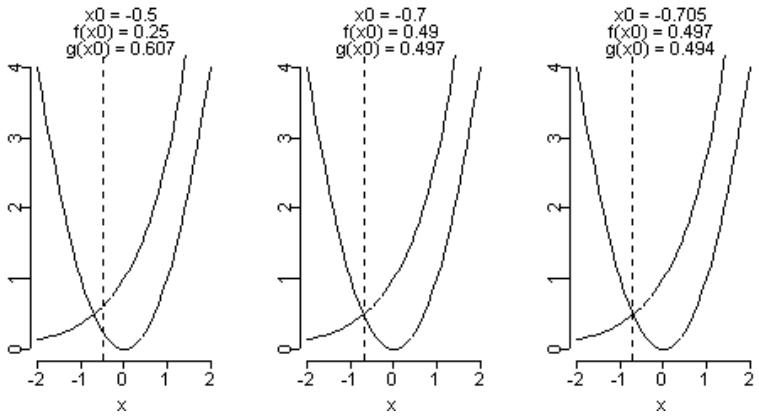

Figure 1 Solving $x^2 = e^x$ using graphical method

We can exploit the graphics capabilities of R in the teaching of statistics. In understanding the parameters of a probability density function (pdf), R can be used to display the behaviour of the pdf when one of its parameters changes. For example, consider the exponential pdf with parameter β, $f(x) = \frac{1}{\beta}\exp(-x/\beta)$, for $x > 0$. The dependence of f on β, and the statistical interpretation of β as both mean and standard deviation, can readily be understood graphically as presented in Figure 2. R is capable of plotting the density function one-at-a-time as β varies.

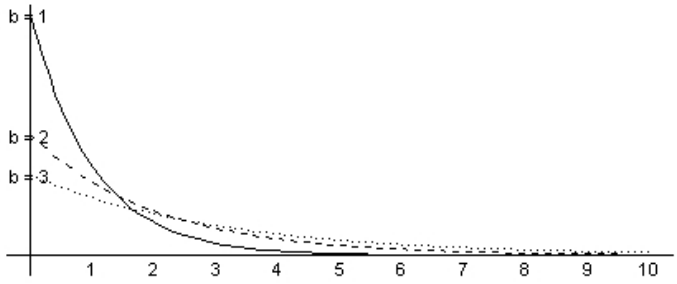

Figure 2 The exponential probability density function with varying parameter

In R, we can simulate data from standard distributions like the binomial and normal, and demonstrate graphically the normal distribution as an approximation to the binomial, for example. We can also demonstrate the Central Limit Theorem by displaying graphically that the sampling distribution of the sample mean is closely approximated by the normal distribution for large sample size n. To demonstrate this fundamental theorem in statistics, Wild and Seber (2000, p. 284-287) consider four distributions that do not look like normal. For each distribution, 5000 samples of size n (n = 1, 2, 4, 10, 25) are simulated and the density histogram of the 5000 sample means is plotted for each n. The R code to perform such demonstration for random samples from an exponential distribution with mean β = 0.5 is given in Appendix A.3. Figure 3 shows that the density histogram of sample means from an exponential distribution approaches a bell-shaped (normal) density as n increases.

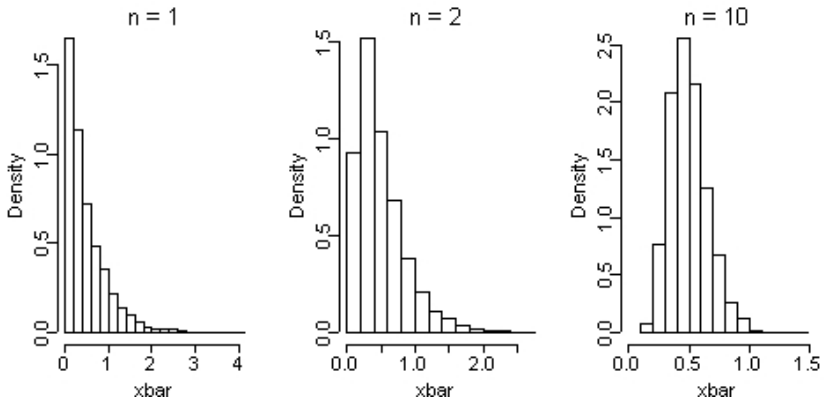

Figure 3 Central Limit Theorem for random samples from exponential distribution

4. R as a statistical software

Finally, we must not forget that R is intrinsically a statistical software, and we certainly want to take full advantage of the packages available. Many standard statistical procedures such as regression and time series have been implemented in R. An inter-disciplinary school project often requires some data analysis, and R is a good choice of statistical software. A useful reference of how to use R in practical data analysis is Venables and Ripley (2002).

As an example, we consider the winning times in Olympic sprints (100-m, 200-m, 400-m) for men and women from 1896 to 1996 (Wild and Seber, p. 548). We can use the linear model package "**lm**" in R to analyse the downward trends in the winning times. Figure 4 compares graphically the winning times, as well as the least squares regression lines, for men and women in the 100-m race. Table 2 shows the "**lm**" output for fitting a simple regression model to the men 100-m data. Can we say that trends are linear? Is there any sign of a leveling-off in the winning times over recent years? To answer such questions, we can easily construct in R any additional regressors needed, such as quadratic term or indicator variable.

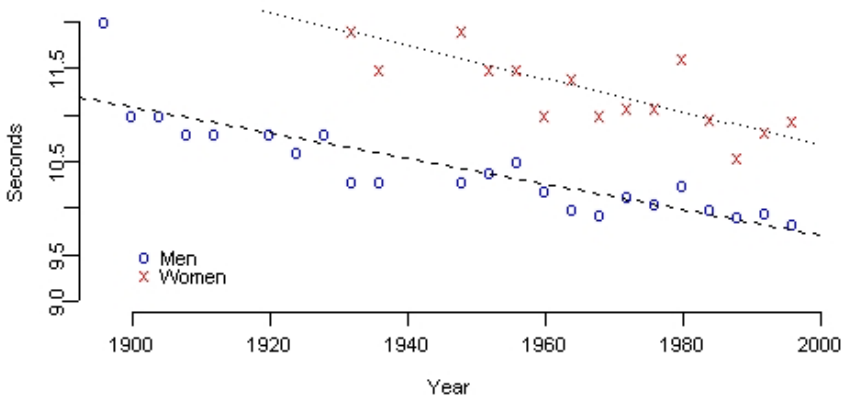

Figure 4 Winning times in Olympic 100-m sprint for men and women

Table 2 Simple regression model for winning times in Olympic 100-m for men

```
lm(formula = m100 ~ year)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.172069   3.200625  11.614 1.33e-10 ***
year        -0.013730   0.001643  -8.356 4.07e-08 ***

Residual standard error: 0.2459 on 21 degrees of freedom
Multiple R-Squared: 0.7688,     Adjusted R-squared: 0.7578
F-statistic: 69.82 on 1 and 21 DF,  p-value: 4.069e-08
```

Suppose we are only interested in computing the sample correlation coefficient r for a simple regression model,

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)\left(\frac{y_i-\bar{y}}{s_y}\right),$$

where s_x and s_y are the sample standard deviations. The R code below to find r will help students understand the "black box" which generate the regression output "**Multiple R-Squared**" in Table 2. In turn, this will encourage students to become more critical consumers of computer output.

```
xbar <- mean(x)
ybar <- mean(y)
sx <- sqrt(var(x))
sy <- sqrt(var(y))
r <- (1/(n-1))*sum((x - xbar)*(y - ybar))/(sx*sy)
```

Being a command-based language, R allows students to explore other "black boxes" associated with the software. For example, statistical functions such as mean ("**mean**") and variance ("**var**") are inbuilt in R. In the process of using these functions to find r, students would naturally raise questions on the formulae used for s_x and s_y: Does the software use n or $n - 1$ as the divisor when computing the sample variance of n observations?

5. Conclusion

R is an "evolving" language in the sense that every user is also a developer. As a free software with additional support provided by the R Foundation, the R language provides a platform for mathematics educators and researchers to "freely" explore how technology can be applied into their teaching and research. It is an "open source" route to the development of better teaching strategies and innovations in statistical and computational research.

References

[1] Ang, K. C. (2002). *Computational Methods with Maple*, Prentice Hall, Singapore.

[2] Ang, K. C., and Awyong P. W. (1999). The use of Maple in first year undergraduate mathematics, *The Mathematics Educator* **4**: 87-96.

[3] Cheang, W.-K., and Reinsel, G. C. (2003). Finite sample properties of ML and REML estimators in time series regression models with long memory noise, *Journal of Statistical Computation and Simulation* **73**: 233-259.

[4] Gradshteyn, I. S., and Ryzhik, I. M. (1994). *Table of Integrals, Series, and Products*, *5th ed.*, Academic Press, Boston.

[5] Venables, W. N., and Ripley, B. D. (2002). *Modern Applied Statistics with S*, *4th ed.*, Springer, New York.

[6] Wild, C. J., and Seber, G. A. F. (2000). *Chance Encounters: A First Course in Data Analysis and Inference*, John Wiley, New York.

Appendix: R codes

In this appendix, we give the R codes for some of the examples discussed. A good introduction to the R language is *The R Manual* edited by the R Development Core Team and downloadable from http://www.r-project.org/.

A.1 Bisection method to solve $x^3 - x - 1 = 0$

```
f <- function(x) {x^3 - x - 1}    # Define the function

eps <- 0.001                      # Set tolerance
nmax <- 10                        # Set max. no. of iterations
a <- 1                            # Initialise starting interval
b <- 2
m <- (a+b)/2                      # Find midpoint
d <- abs(b-m)
count <- 1                        # Initialise iteration count
out <- c(count,a,b,m,d,f(m))

while ((d>eps) && (count<nmax))
{ if (f(a)*f(m) < 0) {b <- m}
  else {a <- m}
  m <- (a+b)/2                    # Find new midpoint
  d <- abs(b-m)
  count <- count + 1              # Count increment
  out <- rbind(out,c(count,a,b,m,d,f(m)))
}
dimnames(out) <- list(rep("",nrow(out)),c("n","a","b"," c=(a+b)/2","  d=|b-
c|","f(c)"))
print(round(out,5))
```

A.2 Graphical method to solve $x^2 = e^x$

```
x0 <- -0.7    # Input x0 one-at-a-time

f <- function(x) {x^2}
g <- function(x) {exp(x)}
x <- seq(-2,2,0.1)
y1 <- f(x)
y2 <- g(x)

for (i in 1:length(x0))
{ plot(x,y1,type="l",ylim=c(min(y1,y2),max(y1)),ylab="",mgp=c(2,0.5,0),cex=0.7)
  lines(x,y2,lty=1)
  abline(v=x0[i],lty=2)
  mtext(side=3,line=1,outer=F,paste("x0 =",x0[i]),cex=0.7)
  mtext(side=3,line=0,outer=F,paste("f(x0) =",round(f(x0[i]),3)),cex=0.7)
  mtext(side=3,line=-1,outer=F,paste("g(x0) =",round(g(x0[i]),3)),cex=0.7)
}
```

A.3 Central Limit Theorem for exponential distribution

```
par(mfrow=c(1,3),oma=c(0,0,0,0),mar=c(3,3,3,2),bty="n")
set.seed(321)

n <- c(1,2,10)      # n = vector of increasing sample sizes
r <- 5000           # r = no. of replications of a given sample size
beta <- 0.5         # Exponential(beta)

for (i in 1:length(n))
{ xbar <- rep(NA,r)
  for (j in 1:r)
  { x <- rexp(n[i],1/beta)   # Generation of exponential random sample of size n
    xbar[j] <- mean(x)
  }

  hist(xbar,breaks=15,prob=T,right=T,main="",xlim=c(0,max(xbar)),mgp=c(2,0.5,0),
  cex=0.7)
  mtext(paste("n =",n[i]),cex=0.8)
}
```