INTRODUCING UNDERGRADUATES TO PROBABILITY USING THE OPEN-SOURCE PROGRAMMING LANGUAGE R

Jane M. Horgan
Dublin City University, Ireland
jane.horgan@dcu.ie

We outline the role of the open-source statistical programming environment R (R Development Core Team, 2005) in teaching a first course in probability. We show how R, which is freely available and downloadable from the web, may be used not only as a tool for calculation and data analysis, but also to illustrate probability concepts, to simulate distributions, and to explore by experimentation different scenarios in decision making. Recognising that the student best understands definitions, generalisations, and abstractions after seeing their applications, almost all new ideas are introduced and illustrated with real examples, covering a wide range of applications in computer science. While we have addressed in the first instance undergraduate students of computing, the approach outlined could also be adapted for students from other disciplines.

INTRODUCTION

In this paper we demonstrate one way to teach a one-semester introductory course in probability at the most basic level using the statistical programming language R. Taking advantage of its powerful graphical and simulation facilities, we largely replace the mathematics with experimentation, and adopt a hands-on approach to learning probability. We begin in Section 2 by introducing R and its facilities, and go on in Section 3 to show how R can be used to illustrate the basic concepts of probability. After that, in Section 4 we introduce probability distributions, both discrete and continuous. Section 5 shows how to use R to deal with applications in queuing, process control and machine learning.

LEARNING R

R is taught from scratch, with no prior knowledge assumed. First the R package is downloaded from the web (www.r-project.org/). A file consisting of the examination marks from the previous year is given on our website (*http://www.janehorgan.com/)*, and the students are required to summarise and provide a graphical analysis of these as part of their practical work. Realistic data sets are provided in R, and these are also used. The *summary* command which gives quartiles, mean, median, standard deviation and range eliminates the tedium of the arithmetic calculations, and allows the emphasis to be put on understanding their meaning. The data are further analysed graphically with histograms (*hist*), boxplots (*boxplot*), scatter diagrams (*plot*), and stem and leafs (*stem*). R is particularly strong on graphics: further graphical procedures are used as the need arises throughout the course.

LEARNING PROBABILITY

The fundamentals of probability are introduced through simulation using the *sample* function; for example *sample(c("H", "T "), 10, prob = c(0.5, 0.5), replace = TRUE)* simulates 10 tosses of a fair coin. By changing *c("H", "T ")* to *c("d", "g")* and *prob = c(0.5, 0.5)* to *prob = c(0.01, 0.99)* we could, for example, model a manufacturing process with a 1% defective rate.

After this, we quickly move on to real probability problems. R is used to compute the probability of a match for the birthday problem for varying numbers of people. We show how the manufacturers of the Intel chip in 1994 got it wrong when they did not immediately replace a chip that they knew was flawed, believing that the probability of the occurrence of the flaw (1/9billion) was so low that it would occur only "once in 27,000 years" for a typical user. The probabilities of the occurrence of the flaw are calculated for realistic numbers of divisions; for example one billion is not unusual in a multivariate regression problem. Probabilities of this order are easily calculated using R, and, in the Intel case, shown to be non-negligible.

We examine the reliability of a product with components in series and in parallel and a mixture of both, and investigate the improvement in reliability levels when components are backed

In C. Reading (Ed.), *Data and context in statistics education: Towards an evidence-based society. Proceedings of the Eighth International Conference on Teaching Statistics (ICOTS8, July, 2010), Ljubljana, Slovenia*. Voorburg, The Netherlands: International Statistical Institute. www.stat.auckland.ac.nz/~iase/publications.php

up. Figure 1, obtained with *plot(k, 1- .25^k)*, represents the effect of backing up component *k* times, when each component has a 75% chance of functioning.

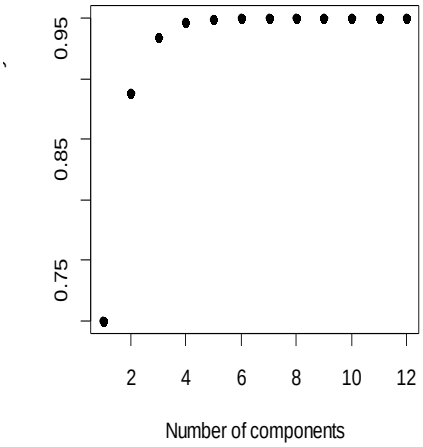

Figure 1. Reliability of Components in Parallel

From Figure 1 we see that four components in parallel appears to be the optimum; any more will not improve the reliability substantially while any fewer will result in a noticeable decrease. This point is often missed in algebraic considerations.

PROBABILITY DISTRIBUTIONS

For most discrete and continuous distributions, R provides functions to calculate and illustrate graphically probability density functions (*pdf*) and cumulative distribution functions (*cdf*), to obtain quantiles, and to generate random numbers. Because R treats probability distributions, both discrete and continuous, in a unified manner, it is a natural system for a probability course.

Discrete distributions

We deal with discrete distributions usual in introductory probability courses; the uniform, geometric, binomial, hypergeometric and Poisson; *dbinom* computes the pdf, *pbinom* the cdf, *qbinom* the quantiles and *rbinom* selects random numbers for the binomial. Similarly for the others; *dgeom*, *dhyper*, *dpois*. The Poisson *pdfs* in Figure 2 are obtained using plot *(dpois(x, λ)*), with differing values of *λ*.

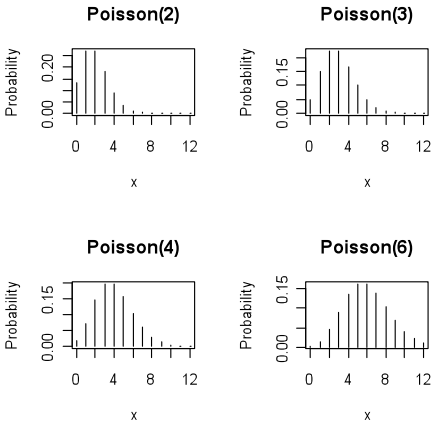

Figure 2. Poisson Distributions

Instead of deriving the mean and variance, we use a simulation approach by generating random numbers (e.g,. *rbinom*, *rgeom*, *rhyper*, *rpois*) and calculating the mean and variance from the simulated data.

Continuous distributions

One stumbling block in moving from discrete to continuous variables is that many students have limited skills in integration. The integrate function provided in R overcomes this problem. Similar to the approach with discrete variables, the functions *rnorm, rexp* are used to simulate values from the normal and exponential distributions, from which *curve (dnorm(x, 50, 10))* plots a normal curve with a mean of 50 and a standard deviation 10. By changing the mean and standard deviation, we investigate how the normal curve changes with differing parameters. R has facilities which allows curves to be superimposed on existing graphs; for example the normal curve is superimposed on the Poisson pdfs in Figure 2 with

curve(dnorm(x, λ, sqrt(λ)), add = TRUE).

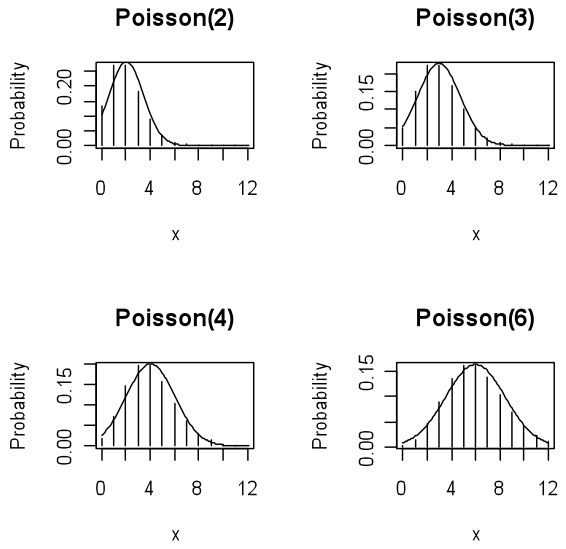

Figure 3. Poisson Distributions with the Normal Superimposed

Notice that the Poisson density is highly skewed when the parameter is 2. The skewness decreases as the value of the parameter increases; at 6 the *pdf* appears symmetrical, bell-shaped and normal-like.

APPLICATIONS

Because of the powerful computational facilities of R, we can deal with topics in probability which would otherwise be too difficult or too tedious in a first course. We illustrate with examples in queuing theory, process control and machine learning, all of which can be made more interesting and accessible by using the plotting and simulation facilities in R.

Queues

Queuing theory is relevant and important in many areas of applications, but is not usually included in a basic probability course. In courses where it is included, it is dealt with either in a theoretical way, an approach which is out of reach of most except the mathematically inclined, or by applying the formulas without proof, which is unsatisfactory from the pedagogical point of view. Our approach here is to omit the theory, and to concentrate completely on experimentation. The familiar M/M/1 queue is introduced using *rpois* to simulate the arrivals and services with various arrival and service rates. Of interest in the study of queues, among other things, is the length of the queue and hence the likely waiting time. We investigate the length of queues for (i) when the arrival rate is greater than the service rate, (ii) when both the arrival and service rates are equal, and (iii) when the arrival rate is less that the service rate. The traffic intensity (I), is the ratio of the arrival rate to the service rate. In (i) when the arrival rate is greater than the service rate, the traffic intensity $I > 1$, in (ii) $I = 1$ and in (iii) $I < 1$. Using *rpois* we investigate each of these three

scenarios. Figure 4 is an example of the simulated queue length with the three different traffic intensities.

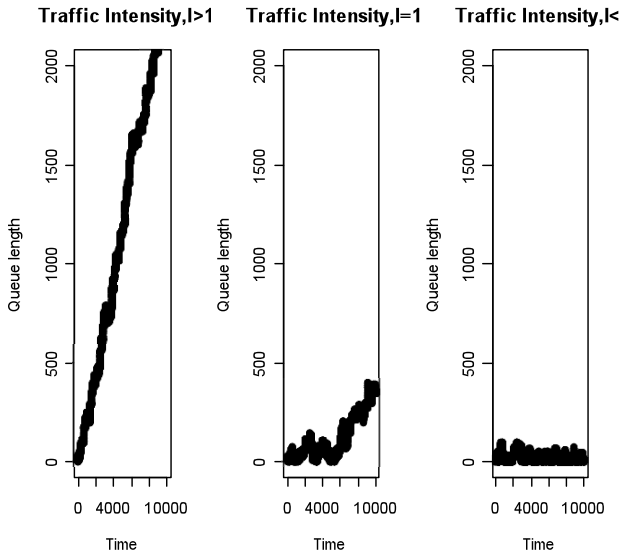

Figure 4. Queue Lengths

Here we see the severe problem when the arrival rate is greater than the service rate (I > 1), the length of the queue is increasing steeply. With arrival and service rates equal (I = 1), the problem is not as severe, but, contrary to what most people believe, it does exist, and, in the long run, it will become serious. The only tenable solution to the queuing problem is to keep the arrival rate less than the service rate (I < 1). It makes sense, in this case, to calculate the average queue length, *mean(queue)*, We can also estimate the worst case scenario, the longest wait that is likely, *max(queue)*, and the best case, the shortest wait, *min(queue)*. By varying the service and arrival rates, we also investigate the optimum traffic intensity, i.e., by how much greater than the arrival rate does the service rate need to be for a satisfactory throughput?

Process Control

Though not intellectually challenging, process control can be interminably boring, because of what seems to be the endless repeated calculations that need to be made. With R, these calculations can be done effortlessly, allowing the student to concentrate more on the interpretation and implications of the results.

The most commonly used technique for detecting changes in a manufacturing process is the control chart: this is handled in R using *plot.* An alternative is to use what is known as a cusum chart which accumulates the deviations from the target of the individual observations. In R the function *cumsum* obtains the cumulative partial sums. Figure 5 gives an example of a control chart and a cusum chart based on the same set of data.

From this diagram, it is easy to see that, with the cusum chart, the change in the pattern begins to happen after day 20: from there on the cumulative sums increase steadily. This pattern is not so obvious with the control chart, because each measurement is independently compared, and so, small changes in the target are often obscured by the residual variation. The cusum chart is more sensitive to changes in the process than the control chart; it combines successive results, and picks up on changes in the process more quickly.

Machine Learning

Machine learning attempts to improve classification accuracy. Individual decisions of an ensemble of classifiers are often combined to classify new examples, and a majority decision is taken. With R it is easy to investigate the change in accuracy when we know the probability that an

individual classifier will be in error, *p* say. Figure 6 gives the probability of incorrect decisions based on a majority result of 21 classifiers, and is obtained by plotting *p* against *1-pbinom(10, 21, p)*, the probability of more than 10 of the classifiers are in error. The line, superimposed on the graph with *lines(p, p)*, illustrates the point at which a majority decision error is equal to an individual decision error.

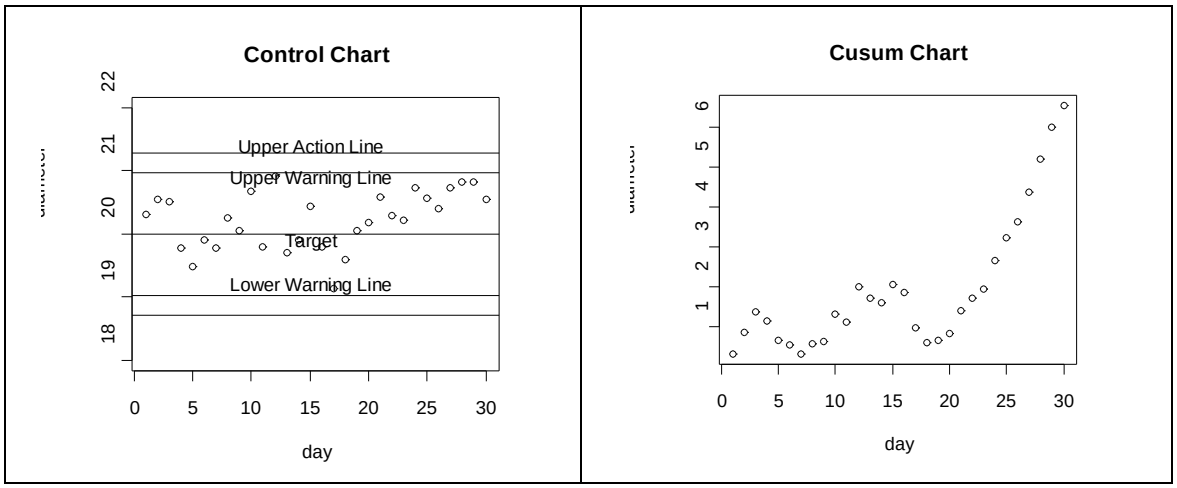

Figure 5. Process Control

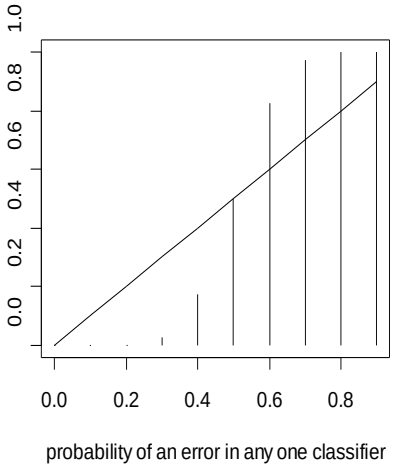

Figure 6. Machine Learning Accuracies

We see in Figure 6 that, when $p < 0.5$, the majority probability is below this line, which means that the probability of making an incorrect classification is less than the error probability of an individual. When $p = 0.5$ the majority and individual decision error probabilities are the same, but when $p > 0.5$, the probability of making an error based on a majority decision is above the line, which means that it is greater than the error probability of an individual. So combining classifiers and taking a majority decision is the wrong thing to do when individual classifiers have error rates greater than 0.5 as it increases the probability of classifying incorrectly.

SUMMARY

We have given a flavour of our R-led approach to teaching a first course in probability. We have shown how to use R to get the students to adopt a hands-on approach rather than memorising formulas and carrying out tedious calculations. More details and further examples are given in our book "Probability with R", together with the R code. Interactive tutorials are provided on our

website (*http://www.janehorgan.com/*), and the R manual is downloadable from the web (Horgan, 2009).

REFERENCES

R Development Core Team (2005). *R: A language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. *http://www.r-project.org*.

Horgan J. M. (2009). *Probability with R: An Introduction with Computer Science Applications*, Hoboken: Wiley. *http://www.janehorgan.com/*.

Venables, W. N., Smith, D. M., & the R Development Core Team (2005). *An Introduction to R: A Programming Environment for Data Analysis and Graphics. http://www.r-project.org/*.