Data Analysis Activities and Problems for the Computer Science Major in a Post-calculus Introductory Statistics Course*

Juana Sanchez
Department of Statistics, UCLA

KEY WORDS: Teaching, Computer Science majors, Data sets, Internet, log data, algorithm, queueing models, students

1 Introduction

The material presented here is a very small subset of problems currently being prepared for a larger instructional improvement project funded by the Office of Instructional Development (OID) at UCLA. The objective of the project is to create a manual with data sets and contextual problems for Computer Science majors that will complement the textbooks used in the calculus-based upper-division Applied Statistics course. More than one third of the students in this course are from Computer Science, while the remaining students come from Engineering and Applied Math, with a very few majoring in fields like Economics, Biology or Genetics. The course is a prerequisite for another one taught by the Computer Science Department on advanced Probability Models for Computer Science, which many majors taking Applied Statistics never take.

The need for developing material tailored to the needs of Computer Science majors in a calculus-based Applied Statistics course is a natural consequence of the recent trend toward making Statistics more data oriented and contextual-based than it has been in the past. The data sets and problems fill a gap that exists today in most textbooks for this kind of courses, which, although adhering to the trend, concentrate almost exclusively on Engineering and other Science, neglecting to a large extent the Computer Science student.

There exist a few books on Probability and Statistics that were written with Computer Science majors in mind exclusively. The book of Allen(1990) has the standard statistics topic areas: Probability and Statistical Inference. In addition to these, the book covers stochastic processes and queing theory, two areas of utmost importance in Computer Science. Throughout the book, the author uses contextual problems that have some part of a computer system as a theme. But there are no data sets coming with the book. And many contemporary problems such as Internet data analysis is not covered. Ross(2000) is exclusively focused on Probability for Computer Science, lacking any discussion of data analysis or statistical inference. Plenty of examples are presented of a different nature than those in Allen's book. The level of Ross's book is a little bit too high for the probability part of the course that occupies this paper. But the two books mentioned above, and a few others, as well as numerous recent journal articles and data sets publicly available are very good resources for inspiration and starting points. New problems and data set sets can be adapted from them to create contextual problems and data analysis projects accessible to the students in the course at hand.

In this paper, I present three basic exercises that I have adapted from different resources to illustrate the nature of the hands-on activities that comprise the whole project. Before moving on to the examples, I give an overview of the nature of randomness in Computer Science. I conclude with some information on the remaining steps in this project, and the web site where the problems and data sets can be found as the project advances.

2 Randomness and Data in Computer Science contexts

In most areas of Computer Science we deal not with deterministic phenomena, but rather with probabilistic phenomena. The time it takes to write and check a computer program, the time it takes to run a program (measured from the time it is submitted to a batch system or invoked via an online-system), the time it takes to retrieve information from a storage device, the number of jobs awaiting execution on a computer system, the number of terminals in use, how long a user stays in a web page, how many web pages a user browses, among others, are all very basic examples of probabilistic or random phenomena. Data on those or many other variables may come from log files that all computer systems keep, or, in some cases like some web usage data, from

*Supported by UCLA'S Office of Instructional Development Grant IIP-03-20. Paper presented at the Joint Statistical Meetings, Session 338, August 6, 2003, 8:30-10:20

survey data. The log data allow us to estimate the parameters of probabilistic models assumed for those variables and conduct inference. Alternatively, we can use simulation of probabilistic models with given parameters to generate data, particularly for those variables for which we have the analytic solutions of their behavior obtained through probability theory. The simulation data can be used to determine whether the information coming from log files really conforms to those models, or to predict the performance of the system under certain conditions. Either way, the point is that it is feasible to have data (real or created) to prepare exercises in descriptive Statistics and Statistical Inference.

Due to the probabilistic nature of many Computer Science phenomena and to the large probabilistic component of the Applied Statistics course assumed in this paper, the project described in this paper has also a large probability theory component. Thus the manual of exercises and activities for the computer science major in that course would not be complete without probability problems that reflect the current modeling trends in computer science while keeping the material at the level of the beginning student. Ideally, the same topics discussed during the descriptive and probability parts of the course, will be brought up during the inference part.

3 Activities for the Classroom

In the first activity presented in Section 3.1 below, the data are simulated log data resulting from application of a sorting algorithm. I generated this log data by running the bubble sort algorithm many times and keeping track of the number of comparisons needed. With this data set, students can do descriptive data analysis, estimation and hypothesis testing. At the same time, they will be doing comparisons of algorithmic behavior, a very common task in Computer Science. The example would fit in the Statistical inference part of the course as well as in the descriptive data analysis part.

In the second example, Section 3.2, I use a very basic Queueing Theory model of a computer system in which all the parameters are known, and use it as the scenario for a problem where students will have to make probabilistic predictions of the number of jobs in a queue of e-mail messages. This example would be very suitable for several stages of the Probability part of the course. Not only does it make the exponential and Poisson models more interesting by using them in the context of a well known model for a computer system but it allows the student to relate their performance through the model.

In the third example, Section 3.3, students analyze some of the MSNBC Anonymous Web Data from the UCI KDD Archive.This server log data describes the page visits of users who visited msnbc.com on September 28, 1999. The variable of interest in this activity is the number of page categories that the user visited or "depth" of the browsing. With this data set, students can also do descriptive data analysis and estimation and testing. They can also determine the suitability of the sample for generalizing the results to all days, and assumptions needed for that.

All these activities are written below as they would be given to the students. I also describe briefly what is assumed known or previously introduced in class when those activities are handed out to the students. For the first two activities, I give in an Appendix some additional background, work that went into their preparation, and a small answer key. Complete solutions and suggestions for discussion will be provided in the Solutions Manual that will be made available to instructors when the project is completed.

The activities can be done either in a lecture context, as examples, or in a discussion/review session where students can spend more time learning the background, or simply be given as homework. I prefer the discussion/review session because there is more time to focus on the context of the problem.

3.1 Sorting Algorithm Behavior

In computer science, the algorithm is a very important concept and can be seen as a model of the operations done by a computer to solve some problem, for example, sorting. A very important variable of interest in analyzing a sorting algorithm's behavior is the number of comparisons that need to be made before a list is sorted. This is a random variable that could be introduced in class when talking about random variables. Mathematical expressions for the expected number of comparisons can be obtained and could be used as examples when explaining the Expected Value in class, too.

For those unfamiliar with sorting algorithms, Appendix A reviews the workings of two very popular ones considered in this activity: the quicksort and the bubble sort. The data in this activity come from the bubble sort. The Appendix also provides the program to obtain the data set and suggests some basic analysis of the data.

3.1.1 The problem given to students

One thousand random permutations of the list 2, 70, 11, 47, 75, 100, 84, 32, 42, 43, 34, 22 were sorted by 1000 users using the same computer, and X = the number of pairwise comparisons needed to sort the list were recorded. We don't know what kind of sorting algorithm was used by the computer to

do the sorting, but we know that it was either the bubble sort or the quicksort algorithms discussed earlier in class when talking about the Expected Value. Use the records provided in the data set "sort" to determine which of the sorting algorithms was used by the computer. Support your answer with a thorough descriptive and inferential data analysis. Note: The data set with the records is called *sort* and can be found at: `http://www.stat.ucla.edu/ jsanchez/ oid03/index.html`

3.2 Queuing Theory in Computer Science

This very important modeling technique represents a computer system as a network of service centers, each of which is treated as a queueing system. That is, each service center has an associated queue or waiting line where customers who cannot be served immediately queue (wait) for service. The customers are, of course, part of the queueing network. Customer is a generic word used to describe workload requests such as CPU service, I/O service requests, requests for main memory, etc. All these arrive at random to the service facility. Queueing theory models are often used to determine the effects of changes in the configuration of a computer system.

The simplest queueing theory model is the `M/M/1` model, which assumes: (a) that customers arrive in accordance with a Poisson process with average rate λ and thus the interarrival times are exponentially distributed with mean $\frac{1}{\lambda}$. (b) Service time by the server is assumed exponential with parameter μ. Expected service time is then $\frac{1}{\mu}$. (c) The customers are served one at a time by a single server. If the server is busy upon the customer's arrival, then the customer waits in the queue. The activity presented below assumes that the random variables that comprise this model were used as examples when talking about random variables in class, and when talking about the exponential and the Poisson random variables. The set of random variables involved is summarized in the diagram displayed below, which is an adaptation of a diagram in Allen(1990), p. 251.

For these models we are usually interested in determining, among other things, the average number of customers in the system (or in the queue) and the average amount of time a customer spends in the system.

From the assumptions above, it can be proved (Allen 1990) that the number of customers in the system has a geometric distribution with parameter $p = 1 - \frac{\lambda}{\mu}$ and $q = \frac{\lambda}{\mu}$. So the expected number of customers is $\frac{\frac{\lambda}{\mu}}{1-\frac{\lambda}{\mu}}$. And the probability

$$\begin{aligned} P(N < n) &= P(N = 0) + \cdots + P(N = n - 1) \\ &= 1 - (\frac{\lambda}{\mu})^n \end{aligned}$$

Consequently,

$$P(N \geq n) = 1 - P(N < n) = (\frac{\lambda}{\mu})^n.$$

The following problem, inspired by Allen(1990) p. 267, makes use of this result. See Appendix B for a brief discussion of its solution. More detailed solutions will appear in the final Solutions Manual.

3.2.1 The Problem given to students

Traffic to an e-mail server arrives in a random pattern (i,e, exponential interarrival time) at a rate of 240 e-mails per minute. The server has a transmission rate of 800 characters per second. The message length distribution (including control characters) is approximately exponential with an average length of 176 characters. Assume a `M/M/1` queueing system like that often used in class when talking about random variables (i.e., exponential arrival times, exponential service time, and 1 server). What is the probability that 10 or more messages are waiting to be transmitted? Support your answer showing work and explaining the implications of the assumptions.

3.3 Uses of the Internet

Computer Scientists are being called more and more frequently to provide computer log data that can be used to find out how users interact with the Internet (see, for example, Sen and Hansen (2003), Cadez et al. (forthcoming), Huberman et al (1998)). In addition to that, the number of Internet user surveys is growing at an amazing speed (see, for example, Wellman and Haythornthwaite (2002)). The following problem uses a data set published in the UCI KDD Archive (Kederman, 2003) to obtain data on the number of different pages visited by users who entered the msnbc.com page on September 28, 1999. The transformed data set can be found at: `http://www.stat.ucla.edu/ jsanchez/ oid03/index.html`.

It is assumed that during the Data Description and Estimation part of the course, students will have heard some examples that were based on internet data, probably an example based on the same data set that is used for the problem below. Also assumed is that during the Probability part of the course students heard about the random variable "depth of browsing."

3.3.1 The Problem given to students

There is a data set that describes the page visits of users who visited msnbc.com on September 28, 1999. Visits were originally recorded at the level of URL category and were recorded in time order. But the data set that you will use summarizes the number of different categories visited per user. Thus there is only one random variable X = the number of page visits and one record for each user who visited msnbc.com on September 28, 1999. Assuming that this is a typical day, (a) describe what the data are saying about the number of page visits and (b) find the maximum likelihood estimate of the average number of page visits per user. Support your answers with graphs, summary statistics and inference results.

The data set with the processed records is called *visits* and can be found at: `http://www.stat.ucla.edu/ jsanchez/ oid03/index.html`.

4 Conclusions

As mentioned in the Introduction to this paper, the example activities presented above are just a small sample of all the activities that are part of an Instructional Improvement project. This project involves several research steps: (a) adaptation of problems in Computer Science literature to the undergraduate level. (b) Preparation of lab projects with guided questionnaires using publicly available log or survey data. (c) Development of problems based on simulated data sets. (d) Compilation of a set of articles that can be read by undergraduate students. (e) Design of methods to assess the effect of using these problems in the classroom.

My main hypothesis is that the Computer Science major in the post-calculus Introductory Statistics course will benefir immensely from these activities tailored to them. But I also assume that all majors in the course will benefit, as web access, usage and sociodemographic survey data and log data will soon be the standard data to use for research. Finally, I anticipate that instructors will find this complement to textbooks valuable.

The complete set of activities and the data sets that accompany them will be appearing in a web site as they are being developed. This web site is `http://www.stat.ucla.edu/ jsanchez/ oid03/index.html`.

Appendix A. Sorting Algorithms

A.1. The Bubble Sort Algorithm

Ross (2000) p. 8, describes the Bubble Sort Algorithm as follows: Suppose we are given a set of r distinct values $y_1, y_2, ..., y_r$ that we desire to put in increasing order or, as is commonly called, to sort them. The *bubble sort* is an algorithm that can be used. Starting with any initial ordering, it sequentially passes through the elements of this ordering, interchanging any pair that it finds out of order. That is, the first and second values are compared, and interchanged if the second is smaller; then the new value in second position is compared with the value in the third position, and these values are interchanged if the latter is smaller than the former; then the new value in the third position is compared with the value in the fourth position, and so on until a comparison is made with the final value in the sequence, and an interchange, if necessary, is made. At this point the first pass through the list is said to have occurred. This process is then repeated for the new ordering, and this continues until the values are sorted. For instance, if the initial ordering of values is

2,70,11,47,75,100,84,32,42,43,34,22

then the successive orderings in the first pass through are as follows:

2,70,11,47,75,100,84,32,42,43,34,22
2,11,70,47,75,100,84,32,42,43,34,22
2,11,47,70,75,100,84,32,42,43,34,22
2,11,47,70,75,100,84,32,42,43,34,22
2,11,47,70,75,100,84,32,42,43,34,22
2,11,47,70,75,84,100,32,42,43,34,22
2,11,47,70,75,84,32,100,42,43,34,22
2,11,47,70,75,84,32,42,100,43,34,22
2,11,47,70,75,84,32,42,43,100,34,22
2,11,47,70,75,84,32,42,43,34,100,22
2,11,47,70,75,84,32,42,43,34,22,100

There are r-1 (or 11) comparisons in the first pass.

Let X denote the number of comparisons needed by bubble sort and consider $E(X)$ the expected value of X. Then

$$\frac{r(r-1)}{4} \leq E(X) \leq \frac{r(r-1)}{2}.$$

See Ross(2000) for a proof. If $r = 12$, then $33 \leq E(X) \leq 66$.

A.2. The quicksort algorithm

Suppose that we want to sort a given set of r distinct values $y_1, y_2, ..., y_r$. A more efficient algorithm than buble sort for doing so is the *quicksort algorithm*, which is recursively defined in Ross(200) p. 55., as follows. When $r = 2$, the algorithm compares the two values and

puts them in the appropriate order. When $r > 2$, one of the values is chosen, say it is y_i, and then all of the other values are compared with y_i. Those smaller than y_i are put in a bracket to the left of y_i, and those larger than y_i are put on a bracket to the right of y_i. The algorithm then repeats itself on these brackets, continuing until all values have been sorted. For instance, suppose that we desire to sort the following 10 distinct values:

2,70,11,47,75,100,84,32,42,43,34,22

One of these values is now chosen, say it is 75. We then compare each of the other values to 75, putting those less than 75 in a bracket to the left of 75 and putting those greater than 75 in a bracket to the right of 75. This gives

$\{2, 7, 11, 47, 32, 42, 43, 34, 22\}$, 75, $\{84, 100\}$

We now focus on a bracketed set that contains more than a single value –say the one on the left of the preceding –and choose one of its values –say 11 is chosen. Comparing each of the values in this bracket with 11 and putting the smaller ones in a bracket to the left of 11 and the larger ones in a bracket to the right of 11 gives

$\{2\}$, 11, $\{70, 47, 32, 42, 43, 34, 22\}$, 75 $\{84, 100\}$.

This continues until there is no bracketed set that contains more than a single value.

The Expected number of comparisons is

$$E(X) = 2(r+1)log(r) - 2(r-1).$$

If r= 12, then $E(X) = 42.60$. See Ross(2000) for a proof.

A.3. R Program to generate the data

```
bubble <- function(l)
{
f <- 0.01
change <- matrix(0,11,1)
x <- matrix(0,l,1)
compare1 <- matrix(0,12,2)
compare <- 10
tlist <-matrix(0,11,1)
slist <-matrix(0,11,1)

perm <- c(2,70,11,47,75,100,84,
32,42,43,34,22)
for(j in 1:l)
{
 slist <- sample(perm)
 tlist <- slist

  for(k in 1:12)
    {
      for(i in 1:11)
      {
   if(slist[i] > slist[i+1])
    { tlist[i] = slist[i+1]
     tlist[i+1]=slist[i]
     change[i] <- 1
   slist[i] <- tlist[i]
   slist[i+1]<- tlist[i+1]
      }
  else change[i] <- 0
             }


   compare <- sum(change)

    if(compare > 0)
   {compare1[k,1] <-  12-k
   compare1[k,2] <- compare }

    else
   {compare1[k,1] <- 0
    compare1[k,2] <- compare }

        }

x[j] <- sum(compare1[,1])
}
    x
}

x <- bubble(1000)
```

Suggested Data analysis and inference

From the descriptive statistics, the histogram and the box plot, one can conclude that the bubble algorithm is the one used to sort the lists. We know from theory that the Expected number of comparisons with the Bubble algorithm is somewhere between 33 and 66 whereas the expected number of comparisons is 42.60 with the quicksort algorithm. If the quicksort algorithm had been used, we would like to see more area of the histogram around 42.60 or a number close to it. The histogram we obtained has mean and median around 60, which does not contradict the theoretical expected value if the bubble algorithm is used. Most of the data have values between 50 and 66, and the histogram is skewed left suggesting that this algorithm takes a large number of comparisons most of the time.

The mean is 58.8 (t=78.46, p=0.000; 95% CI 58.48, 59.29). We reject the null hypothesis that the mean is 42.60.

Students should be able to provide in an Appendix both the summary statistics and the graphs.

Appendix B. Queueing theory

The average service time (s) is the average time required to transmit a message or

$$\begin{aligned} E(s) &= \frac{\text{average message length}}{\text{line speed}} \\ &= \frac{176 \text{ char}}{800 \text{ char/sec}} = 0.22 \text{ sec.} \end{aligned}$$

Hence, since the average arrival rate is $\lambda = 240$ messages /minute $= 4$ messages/second, the server utilization or probability that the server is busy is

$$\rho = \lambda E(s) = 4 \times 0.22 = 0.88$$

that is, the server is transmitting outgoing messages 88% of the time.

The number of messages in the system, N, is a geometric random variable with parameter $p = 1 - \rho$ and $q = \rho$. So the average number of messages in the system is

$$E(N) = \frac{\rho}{1 - \rho} = 7.33 \text{messages.}$$

Since 10 or more messages are queueing if and only if 11 or more messages are in the system, the required probability is $\rho^{11} = 0.245$.

References

[1] Allen, A.O.(1990). Probability, Statistics, and Queueing Theory with Computer Science Applications, 2nd edition. Academic Press.

[2] Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S. (2003). Model-based clustering and visualization of navigation patterns on a Web site. Accepted for publication on Journal of Data Mining and Knowledge Discovery, 7(4).

[3] Hansen, M.H. and Sen, R.(2003). Predicting Web User's next access based on log data. Journal of Computational and Graphical Statistics, Vol 12, No. 1, March, p. 143-155.

[4] Heckerman, D. The UCI KDD Archive

`http://kdd.ics.uci.edu`

Irvine, CA: University of California, Department of Information and Computer Science.

[5] Huberman, B.A., Pirolli, P.L.T., Pitkow, J.E., Lukose, R.M. (1998). Strong Regularities in World Wide Web Surfing. Science, Vol. 280, 3 April.

[6] Ross, S.M.(2002). "Probability Models for Computer Science." Harcourt Academic Press.

[7] Wellman, B. and Haythornthwaite edts.(2002). The Internet in Everyday Life. Blackwell Publishing.