

BST 140.778 Assignment 4

February 14, 2005

Fine print All computing assignments must be completed in the R statistical programming language. For non-computing assignments please appropriately typeset using \LaTeX . Bundle your R functions and \LaTeX files for each assignment in a zip or tar.gz file and email them to bcarvalh@jhsph.edu. Include *specific* instructions on how to run the code to answer the assignments in a README file. All code should be readable, formatted well, commented and clearly indicate the author and date. The general rule is: the more thought that has to go into understanding how to implement your code, the worse your grade will be. Please feel free to give each other small hints, but otherwise students must complete assignments individually.

1. In the class, we went over Gaussian quadrature. This is a useful technique for integrating with respect to common densities. Two other useful methods are Monte Carlo integration and Romberg integration. Monte Carlo is particularly useful for integrating high dimensional functions and we will discuss it at length. Romberg integration uses the Euler-Maclaurin formula to reduce the error of composite Newton-Cotes integration rules such as the trapezoidal rule. If you haven't heard of Newton-Cotes integration, these algorithms approximate the integrand with a polynomial and then integrate that polynomial (which is easier). The trapezoidal rule approximates the integrand with a line connecting the two endpoints. That is

$$\int_a^b f(t)dt \approx \frac{(b-a)}{2} \{f(a) + f(b)\}. \quad (1)$$

This approximation can be improved by breaking the interval $[a, b]$ into n subintervals and applying the trapezoidal rule to the subintervals. That is we write

$$\int_a^b f(t)dt = \sum_{k=0}^{n-1} \int_{a+kh}^{a+(k+1)h} f(t)dt$$

for h equal $(b-a)/n$ and approximate interval k with

$$\int_{a+kh}^{a+(k+1)h} f(t)dt \approx \frac{h}{2} \{f(a+kh) + f(a+(k+1)h)\}.$$

Breaking a Newton-Cotes rule up like this results in a composite Newton-Cotes rule. Following the text (page 211), let T_{m0} be the approximation to (1) based on breaking $[a, b]$ up into $n = 2^m$ subintervals. Define

$$T_{mi} = T_{m,i-1} - \frac{1}{4^i - 1} [T_{m-1,i-1} - T_{m,i-1}].$$

Romberg integration uses the T_{mi} instead of the T_{m0} . Provided i is less than the number of continuous derivatives of f , Romberg integration will be much more accurate than the underlying composite Newton-Cotes rule. Since often we don't know how many continuous derivatives f has, we report a table of the T_{mi} . The homework exercise is to read the section in the text on Romberg integration (Chapter 16 Section 3) and reproduce Table 16.2 on page 212.

2. I'm going to use R's `runif` function to generate (pseudo) random numbers. However, I want to make sure my random numbers are reasonably random. There are quite a few tests that single various aspects of randomness and uniformity. We will consider only one. We will use the following result.

If (n_1, \dots, n_k) is multinomial with success probabilities $p = (p_1, \dots, p_k)$ and N total trials then

$$X = \sum_{i=1}^k \frac{(n_i - Np_i)^2}{Np_i}$$

follows a Chi-squared distribution with $k - 1$ degrees of freedom as $N \rightarrow \infty$.

Consider the "Runs Test" (see Knuth, *The Art of Programming Volume 2*). Suppose we take a random sequence and keep counting until $U_j > U_{j+1}$. For example, the following illustrates a run of length three: $U_0 = .1, U_1 = .2, U_2 = .9, U_3 = .8$. If we throw away the element that immediately follows a run, then the run lengths are independent. Thus when $U_j > U_{j+1}$ we discard U_{j+1} and start counting again at U_{j+2} . Notice, theoretically, we can easily derive the expected proportions of the various run lengths and perform a Chi-squared test. Derive these probabilities and perform the runs test for a very large sample. (Note, there is no bound on the largest possible run length. Therefore the last multinomial category should be runs of length x or larger, for some value of x .)