Carolyn Cotterman
Stat 244
May 13, 2011

Assignment 4

1 Introduction

For this assignment, I implemented two different binary tree algorithms to order and count vectors of random numbers. I then explored the general factorial analysis of variance using the reduction technique. My code for this assignment is contained in "Stat244_hw4_main.c," "Stat244_hw4_header1.h," "btrees.c," "trees.c," "btrees.h," and "trees.h."' Corresponding output is in, "Stat244_hw4_OUT.txt"

2 Binary Trees

This section explores strategies for determining the number of levels (i.e., unique values) contained in a covariate vector, and for putting these values in order.

The basic idea is that as I process my data, element by element, I create a separate "struct node" to represent each unique value, and link these nodes together in a tree formation. A tree formation is acheived by giving each node two pointers – one for pointing to a child node to its left (which will represent a lower value as compared to its parent node), and one for pointing to a child node to the right (which will represent a higher value as compared to its parent node). Organizing information using linked nodes can improve run-time, as doing so allows me to re-order elements without having to re-allocate memory (in contrast to re-ordering the elements of an array). Organizing data into trees (as opposed to linear linked lists) further improves run-time by enabling me to more efficiently locate the position at which a new observation belongs (i.e., this is the difference between running a sequential search versus a binary search).

Using binary trees, I was able to order and count a list of 1,000 random numbers. I associated multiple peices of information with each node by linking the nodes to"structs." In each struct, I stored the following peices of information: (1) the variable value that the node corresponds to, (2) the number of times that this value is realized in the data, and (3) the order or "level" of the value (where level 0 indicates that the value is the variable's minimium, level 1 indicates that the value is the second lowest, and so on).

Using regular binary trees to order and count data can be slow when the data is pre-sorted. At the extreme, if the data is perfectly sorted, then the binary tree algorithm would essentially result in a linked list, with each node pointing to only one other node. Balanced binary trees, on the other hand, are able to quickly organize data whether the data is pre-sorted or not. After each iteration, balanced binary trees re-set their pointers such that the

length of the left subtree never differs from the lenght of the right subtree by more than one in absoluate value.

The following table summarizes the time it took for my regular and balanced tree algorithms to execute 10,000 times for sorted and unsorted data:

	Unsorted Data	Sorted Data
Regular Binary Tree	2.12	7.88
Balanced Binary Tree	2.15	1.51

While the run times for the regular and balanced binary tree methods are similar for unsorted data, I find that the regular binary tree method takes much longer than the balanced binary tree method when the data is pre-sorted.

3 Analysis of Variance

Analysis of Variance (aka ANOVA) partitions the observed variance of an outcome variable of interest into components attributable to different sources of variation. For this assignment, I focus on a factorial Analysis of Variance, which involves estimating not only covariate main effects, but also the interaction effects of covariates. Since my data is unbalanced (cell sizes are unequal), I must decide on the type of sums of squares (SS) to use, realizing that my resulting coefficients and their interpretations depend on my choice. I decide to use Type III SS, which means that I will adjust each SS for all other effects in the model.

For the remainder of this report, I will use the following terms and notation: R(e) will represent the regresion SS for a model containing only e, defined as $R(e) = SS_{total} - SS_{residual}$. Similarly, $R(e_1|e_1, e_2) = R(e_2, e_1) - R(e_2)$, where $R(e_2, e_1)$ is the SS of a model containing both e_1 and e_2, and $R(e_2)$ is the SS of a model containing only e_2.

Factorial ANOVA can be implemented by following these steps:

1. Determine the number of observed levels for each factor using balanced binary trees (as discussed in previous section)
2. Create a (n x p) design matrix, X, that codes categorical variables as follows:
   - A categorical variable with k levels will be contained within k-1 dummy variables, each of which gets its own column.
   - Let i represent a particular level of a categorical variable, where i=1,...k. For each level except the kth, set the ith dummy equal to 1 if the value of the observation is the ith level of that variable, and equal to 0 otherwise.
   - For the kth level, set all k-1 dummy variables to -1.
3. Calculate $[X:Y]^1[X:Y]$, where [X:Y] is interpreted as the matrix generated by placing the design matrix side-by-side with the vector of outcomes, Y. Note that

$$[X : Y]^1[X : Y] = \begin{bmatrix} X'X & X'Y \\ Y'X & Y'Y \end{bmatrix} \quad (1)$$

4. Sweep columns 1 thru p of the above matrix. This gives us:

$$\begin{bmatrix} (X'X)^{-1} & (X'X)^{-1}X'Y \\ -Y'X(X'X)^{-1} & Y(I - X(X'X)^{-1}X')Y \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma^2}V(\hat{\beta}) & \hat{\beta} \\ -\hat{\beta}' & RSS_0 \end{bmatrix} \quad (2)$$

   where RSS_0 is the full model's overall residual sum of squares.

5. Find the effect of a particular factor of interest by re-sweeping just the factor of interest's column(s). Call the new element located in the lower right-hand corner of the resulting matrix $RSS_{exclusion}$. This is the RSS that would have been obtained if the original model had been run without the factors represented by the re-swept columns. Note that this RSS value will be higher than the RSS from the full model (RSS_0). The factor's effect on the sum of squares is thus:

$$SS_{effect} = RSS_{exclusion} - RSS_0$$

6. Calculate related quantities of interest. These include:

   - df_{effect} = the number of re-swept columns (i.e., the number of columns belonging to the factor of interest.)
   - $MSE_{effect} = SS_{effect}/df_{effect}$
   - F-ratio$_{effect} = \frac{SS_{effect}/df_{effect}}{RSS_0/(n-p+1)}$ = an indication of how good the factor is at explaining the variation in the outcome.

I tried to implement the above steps using a dataset on birthweights.

4 Conclusion

This assignment introduced me to the concept of a struct, and how structs can be useful. I learned that structs can be used with pointers to organize data without actually needing to re-allocate memory. I also learned about the inner workings of ANOVA, and got to re-visit my old friend from Stat243: the Beaton Sweep algorithm.