We commend Professor Agresti for pooling together the tremendous recent developments in the area of exact inference for contingence tables and suggesting additional research for the next decade. We would also like to congratulate Dr. Cyrus Mehta on his successful development of the extremely useful computer software, StatXact, an important contribution to our profession.

ACKNOWLEDGMENTS

This research was supported by grants from the National Institutes of Health.

Comment: An Interdisciplinary Approach to Exact Inference for Contingency Tables

Cyrus R. Mehta

I congratulate Professor Agresti for a masterful survey of the blossoming field of exact inference. This paper would not have been as exciting had it been written a decade earlier. Few algorithms were then available for generating permutational distributions or their tail probabilities. Statisticians urgently needing exact tests relied either on brute force exhaustive enumeration of the reference set, or on Monte Carlo sampling. The personal computer industry was in its infancy, and one had to factor the cost of expensive CPU time on a mainframe computer into the decision to compute an exact p-value. But, today, one can buy a 33 mHz 80486 IBM-PC clone for the same price as I paid for my first IBM-XT, $3,300. Yet the 80486 is a hundred times faster. The trend toward increased computing power, more random access memory and more disk storage space, at reduced prices, continues with no end in sight. Are all these computing resources being fully utilized? Let me draw an analogy from the automobile industry. Manufacturers of sports cars are always on the look out for skilled racing drivers able to push a car to its limit by fully utilizing all the available horse power. Similarly, computer manufacturers eagerly solicit software developers whose products can take full advantage of the phenomenal power inside their new machines. Permutational inference is one of the few fields that can satisfy the appetite of an 80486-based PC, a SUN SPARC 2 or a DECstation 5000, eager to devour hard computational problems. Professor Agresti is to be commended for opening up the field and pointing out so many new research directions, guaranteed to keep us occupied for the next decade.

Exact permutational inference is interesting because of its interdisciplinary nature. It draws on ideas from four disciplines: statistics, discrete mathematics, computer science and operations research. I will illustrate this through an exact treatment of the $2 \times k$ contingency table.

1. STATISTICS: THE LINEAR RANK TESTS

Let x denote a generic $2 \times k$ contingency table of the form:

	Col 1	Col 2	...	Col k	Total
Row 1	x_1	x_2	...	x_k	m_k
Row 2	x'_1	x'_2	...	x'_k	m'_k
Total	n_1	n_2	...	n_k	N

Define the reference set of all such $2 \times k$ contingency tables with fixed row and column margins by

$$\Gamma = \left\{ \mathrm{x}: \sum_{i=1}^{k} x_i = m_k,\ x_i + x'_i = n_i,\ i = 1, 2, \ldots k \right\}.$$

For a rich class of statistical problems, the linear rank tests [see, e.g., Chapter 4 of the StatXact, (1991) manual], one needs the permutational distribution of

$$T = \sum_{i=1}^{k} w_i X_i, \tag{1.1}$$

where the w_i's are arbitrary scores, and for any $\mathrm{x} \in \Gamma$,

$$\Pr(X = x) = \prod_{i=1}^{k} \binom{n_i}{x_i} \Big/ \binom{N}{m_k}. \tag{1.2}$$

Cyrus R. Mehta is Associate Professor of Biostatistics, Harvard School of Public Health and President of Cytel Software Corporation, Cambridge, Massachusetts. His mailing address is 137 Erie Street, Cambridge, Massachusetts 02139.

The support points of this discrete distribution are the elements of the set

$$\Omega(k, m_k) = \bigcup_{x \in \Gamma} \left\{ t: t = \sum_{i=1}^{k} w_i x_i, \sum_{i=1}^{k} x_i = m_k \right\}. \tag{1.3}$$

The probability of each $t \in \Omega(k, m_k)$ is, up to the normalizing constant $N!/m_k!(N - m_k)!$,

$$c(k, m_k, t) = \sum_{x \in \Gamma(k, m_k, t)} \prod_{i=1}^{k} \binom{n_i}{x_i}, \tag{1.4}$$

where

$$\Gamma(k, m_k, t) = \left\{ x: x \in \Gamma, \sum_{i=1}^{k} x_i = m_k, \sum_{i=1}^{k} w_i x_i = t \right\}. \tag{1.5}$$

The goal is to develop an efficient algorithm to compute the distribution

$$\{(t, c(k, m_k, t)): t \in \Omega(k, m_k)\}. \tag{1.6}$$

Often, we will only be interested in values of $[t, c(k, m_k, t)]$ in the range $a \le t \le b$ and will expect the algorithm to furnish this portion of the distribution without having to first generate the entire distribution.

2. DISCRETE MATHEMATICS: USE OF RECURSION

One could, of course, enumerate explicitly every $x \in \Gamma$, compute (1.1) and (1.2) each time and thereby generate the desired distribution of T. However, as Professor Agresti has pointed out, the number of elements in Γ grows exponentially so that explicit enumeration soon becomes computationally infeasible. On the other hand, the number of elements in $\Omega(k, m_k)$ is relatively small (for most choices of w_j scores), and this set of support points, along with the corresponding probabilities, $c(k, m_k, t)$ may be built up recursively in $k + 1$ successive stages. Most modern algorithms [e.g., Pagano and Tritchler (1983); Streitberg and Rohmel (1986); Vollset, Hirji and Elashoff (1991); Mehta, Patel and Senchaudhuri (1992)] can be conceptualized in this manner, although there may be differences in their actual implementation. The necessary stagewise recursions are constructed next.

Let $m_j = \sum_{i=1}^{j} x_j$ denote the partial sum of the entries in the first j columns of row 1 of any $x \in \Gamma$. Conditional on m_j, the possible candidates for m_{j+1}, the partial sum up to column $(j + 1)$, are the sets of integers

$$S(j, m_j) = \left\{ m_{j+1}: \max\left(m_j, m_k - \sum_{i=j+1}^{k} n_i \right) \le m_{j+1} \le \min(n_{j+1} - m_j, m_k) \right\}. \tag{2.1}$$

Again, conditional on m_j, the possible candiates for m_{j-1}, the partial sum up to column $(j - 1)$, are the set of integers

$$P(j, m_j) = \left\{ m_{j-1}: m_j - \min(m_j, n_j) \le m_{j+1} \le m_j - \max\left(0, m_j - \sum_{i=1}^{j-1} n_i\right) \right\}. \tag{2.2}$$

Define $\Lambda_0 = \{0\}$ and, for $j = 1, 2, \ldots k$,

$$\Lambda_j = \bigcup_{m_{j-1} \in \Lambda_{j-1}} S(j - 1, m_{j-1}), \tag{2.3}$$

as the set of all possible partial sums, m_j, up to column j. Now, define $\Omega(j, m_j)$ and $c(j, m_j, t)$ as in (1.3) and (1.4), respectively, but with j replacing k. Initialize $\Omega(0, 0) = \{0\}$, and $c(0, 0, 0) = 1$. Then, for $j = 1, 2, \ldots k$, and each $m_j \in \Lambda_j$, form the sets

$$\Omega(j, m_j) = \bigcup_{m_{j-1} \in P(j, m_j)} \{t' + w_j(m_j - m_{j-1}): t' \in \Omega(j - 1, m_{j-1})\}. \tag{2.4}$$

For each $t \in \Omega(j, m_j)$, compute

$$c(j, m_j, t) = \sum_{m_{j-1} \in P(j, m_j)} \binom{n_j}{m_j - m_{j-1}} \cdot c(j - 1, m_{j-1}, t - w_j(m_j - m_{j-1})). \tag{2.5}$$

We end up at stage k with the desired distribution (1.6).

3. COMPUTER SCIENCE: THE DATA STRUCTURES

A crucial step in the implementation of the above recursions is the proper choice of data structures for storing and updating the intermediate information, stage by stage. Particular care must be given to the data structure used to store the sets $\Omega(j, m_j)$. These sets carry all the support points of the distribution at stage j. Equation (2.4) shows that each support point $t' \in \Omega(j - 1, m_{j-1})$ is extended by an amount $w_j(m_j - m_{j-1})$ and then inserted into $\Omega(j, m_j)$. The question to be addressed during this

insertion is whether there already exists a support point in $\Omega(j, m_j)$ that matches the one about to be inserted. Unless the data structure for the elements of $\Omega(j, m_j)$ is appropriately selected, the insertion of each new element will necessitate a costly search for matching elements, over the entire set. This could kill the algorithm outright.

If all the support points, $t \in \Omega(j, m_j)$ are equally spaced (as they will be when the w_i scores are equally spaced), a simple data structure suffices. At each stage j, we may store the distinct values of t in a linear array, one below the other. Now, each new value of t to be inserted into $\Omega(j, m_j)$ has a unique address on the linear array, and matching elements are instantly identified without any search. However, if the w_i scores are arbitrary, the support points will not be equally spaced. In that case, one of the most practical techniques for dynamic search and insertion of new elements into $\Omega(j, m_j)$ is hashing. The elements of $\Omega(j, m_j)$ are transformed through a hash function into an address in a hash table. Identical elements will hash to the same address, thus avoiding the costly search. There may be collisions, however, whereby different elements hash to the same address. A good hash function keeps these to a minimum. See Horowitz and Sahni (1983, page 456) for a good discussion of hashing, and Mehta and Patel (1986) for a description of the data structures used to store intermediate support points for Fisher's exact test on $r \times c$ contingency tables.

4. OPERATIONS RESEARCH: INTEGER PROGRAMMING

As stated previously, we might be interested in generating the distribution (16) only in the range $a \le t \le b$. It is then wasteful to generate the entire distribution and extract from it the values in the desired range. One can, however, weed out values of $t = t' + w_j(m_j - m_{j-1})$ prior to their insertion into $\Omega(j, m_j)$ as stipulated by (2.4). To accomplish this, it is necessary to hypothetically extend t, from its current value at stage j to the range of possible values it could assume at stage k. If we can establish at stage j itself that the extended value of t can never fall inside the interval $[a, b]$, this support point may be dropped from further consideration, and the number of elements in $\Omega(j, m_j)$ is correspondingly reduced. This is a valuable saving, for elements of $\Omega(j, m_j)$ tend to proliferate in subsequent stages, using up valuable storage space and consuming large quantities of CPU time needlessly.

Now, the hypothetical extension of the present support point, from stage j to stage k, is an exercise in integer programming. In the context of the $2 \times k$ contingency table, the following minimization (maximization) problem must be solved:

$$\text{Minimize } SP(j, m_j)(\text{Maximize } LP(j, m_j)) = \sum_{i=j+1}^{k} w_i x_i,$$

$$\text{subject to } \sum_{i=j+1}^{k} x_i = m_k - m_j,$$

$$x_i \text{ integer and } 0 \le x_i \le n_i, \forall i = j+1, \ldots, k.$$

The special structure of the problem can frequently be exploited to optimize the above objective function subject to the linear integer constraints. For this problem, a "greedy algorithm" that sorts the w_i's is ascending order and assigns as large an x_{j+1} as is compatible with the linear constraints to w_{j+1}, rapidly. Sometimes these optimization problems require more work. See, for example, the optimization problems connected with the linear-by-linear association test (Agresti, Mehta and Patel, 1990) or Fisher's exact test (Joe, 1988). Backward induction on a network (e.g., Mehta, Patel and Senchaudhuri, 1992) is yet another approach to these optimization problems.

Thus, there may be in general several operations research techniques available to solve the same optimization problems. The point is to find one that solves them quickly, for it will be required repeatedly throughout the algorithm. Each time the optimization problem is solved, one verifies that either $t + SP(j, m_j) < a$ or $t + LP(j, m_j) > b$. Either condition disqualifies the current value of t from insertion into $\Omega(j, m_j)$.

5. THE NETWORK PARADIGM

In Section 7.1 of his paper, Professor Agresti was kind enough to say that the network algorithms developed by Nitin Patel and me, along with various coworkers, provide a "general and versatile" approach to exact permutational inference. He then went on to give a network representation for the reference set of $r \times c$ tables with fixed margins. It may, therefore, appear odd that in this discussion, I have made no mention of the network algorithm. The omission was deliberate. I wanted to demonstrate that the recursions (2.4) and (2.5) can be written down without any reference to networks. However, the process by which they were derived did depend on a network. It was the representation of Γ as a network of nodes and arcs that gave us the necessary insight to write down and implement the recursions (2.4) and (2.5) and to improve their efficiency through integer programming. To make

this clear, I will express the same recursions in network terms.

The network consists of nodes and arcs over $k + 1$ stages labeled $0, 1, \ldots, k$. The nodes at stage j are ordered pairs of the form (j, m_j), where $m_j \in \Lambda_j$. The set $S(j, m_j)$ defines the successor nodes to the node (j, m_j), while the set $P(j, m_j)$ defines its parent nodes. If we start with an initial node $(0, 0)$ at stage 0, and apply (2.1) systematically to all the nodes created at each stage, we end up with a single terminal node (k, m_k) at stage k. Each successor to node (j, m_j) is a node of the form $(j + 1, m_{j+1})$. It is connected to (j, m_j) by an arc of length, $w_{j+1}(m_{j+1} - m_j)$ and probability $n_{j+1}!/(m_{j+1} - m_j)!(n_{j+1} - m_{j+1} + m_j)!$. A path through the network is a sequency of directed arcs connecting the initial node $(0, 0)$ to the terminal node (k, m_k). Its length is the sum of lengths, and its probability the product of probabilities, of the arcs constituting the path. Through this specification, each path through the network represents one and only one table $x \in \Gamma$. Its length is given by (1.1) and its probability is given by (1.2). The problem of generating the distribution (1.6) is now equivalent to generating the distribution of the lengths of all paths through the network. The set $\Omega(j, m_j)$ represents the distribution of the lengths of all the paths from node $(0, 0)$ to node (j, m_j). The recursions (2.4) and (2.5) amount to expressing the distribution of lengths at node (j, m_j) in terms of the distributions of lengths at its parent nodes $P(j, m_j)$. Also, computing $SP(j, m_j)$ and $LP(j, m_j)$ amounts to computing the lengths of the shortest and longest paths, respectively, from node (j, m_j) to the terminal node (k, m_k). These may be obtained by backward induction on the network (Mehta, Patel and Senchaudhuri, 1992) or by more formal integer programming theorems (Joe, 1988; Agresti, Mehta and Patel, 1991).

In our research papers, although not in this discussion, the network representation of a computational problem has always preceded its algebraic representation. It is certainly elegant to express the computational problem directly in terms of recursions like (2.4) and (2.5). However, it is not so easy to gain the necessary insight to write out the recursions in the first place. Nor is it clear how one implements them on a computer once they are written down. We regard the network approach as a general technique for deriving these recursions, guiding us in selecting appropriate data structures for computer implementation, and solving the necessary integer programming problems. We have used this approach for $2 \times k$ tables, stratified $2 \times k$ tables, $r \times c$ tables and logistic regression.

In summary, this discussion has attempted to show, through a detailed dissection of the $2 \times k$ problem, that the basic ingredients of an efficient numerical algorithm for permutational inference comprise of, recursive generation of the distribution of the test statistic, good data structures for storing intermediate distributions through all stages of the recursion and the use of integer programming to generate a truncated distribution. The network paradigm is a useful aid for carrying out these steps. In particular, forward processing of the network is a general way to conceptualize and implement complicated recursions, whereas backward induction on the network is a general way to solve the integer programming problem.

Comment

Samy Suissa

Professor Agresti must be congratulated for this long-awaited review of the principal issues and methodology surrounding exact inference in contingency tables. Since Fisher proposed his exact method of analysis for the 2×2 table in 1934, the amount of literature produced on the subject and the resulting debates have reached immeasurable proportions. (Yes, this pun intended!) Whether dealing with the accuracy of various asymptotic techniques in small sample situations, the diverse possible factors of correction for continuity, or the conditional, unconditional and Bayesian alternatives, the ensuing research has definitely contributed to our increased knowledge of the situation and has motivated imaginative developments in computing algorithms. Professor Agresti pre-

Samy Suissa is an Associate Professor, Departments of Epidemiology and Biostatistics and Medicine, McGill University, and Research Scholar in the Division of Clinical Epidemiology at the Montreal General Hospital. Mailing address: Purvis Hall, 1020 Pine Avenue West, Montreal, Quebec H3A 1A2, Canada.