Computational Statistics & Data Analysis 3 (1985) 159–185
North-Holland

A survey of algorithms for exact distributions of test statistics in $r \times c$ contingency tables with fixed margins

Albert VERBEEK *

Department of Sociology, University of Utrecht and Netherlands Central Bureau of Statistics, Voorburg, The Netherlands

Pieter M. KROONENBERG

Department of Education, University of Leiden, Leiden, The Netherlands

Received June 1984
Revised May 1985

Abstract: Testing of independence in an $r \times c$ table is generally carried out by using a continuous limiting distribution of the test statistic, rather than the discrete distribution itself. This paper surveys algorithms for the computation of the latter. The central idea is the efficient enumeration of all tables with the same margins, or of a suitable subset thereof. The simplest case is Fisher's exact test for a 2×2 table and a one-side alternative hypothesis. Fisher's method is easily extended to $r \times c$ tables and to arbitrary statistics.

Keywords: Exact conditional distributions, Asymptotic approximations, χ^2-tests, Simulation, Finite sampling distributions, Monte Carlo.

1. Introduction

Testing of independence against various alternatives in contingency tables is generally carried out by using continuous limiting distributions of the test statistics employed rather than the discrete distributions themselves. The prime example is the use of the χ^2-distribution to approximate the null distribution of Pearson's X^2 statistic. This practice is so wide spread that the statistic itself is

* Requests for reprints and copies of the program FISHER should be sent to P.M. Kroonenberg, Dept. of Education, Univ. of Leiden, P.O. Box 9507, 2300 RA Leiden, The Netherlands. The views expressed in this paper are those of the authors and do not necessarily reflect the policies of the Netherlands Central Bureau of Statistics (CBS). Research supported by the Netherlands Orginisation for the Advancement of Pure Science (ZWO) and the Institute for Road Safety Research (SWOR).

generally referred to as (Pearson's) χ^2 by many authors including Pearson himself.

The purpose of this paper is to discuss methods to test this independence hypothesis using finite sampling ditributions of the statistics. Primarily we will discuss variations of the approach of completely enumerating all possible tables with given margins, i.e. the isomarginal family. The statistics and hypergeometric probabilities are calculated along with such enumeration.

The issue of finite sampling distributions of statistics arises in virtually all types of contingency tables: one-dimensional tables, 2×2 tables, $r \times c$ tables with more than one degree of freedom, multidimensional tables. A central concern in many papers is the quality of approximation of the Pearson X^2 statistic to its asymptotic χ^2-distribution, but other statistics have been investigated as well. The outcomes of such investigations depend heavily on the decision of whether to condition on the margins or not. Of course, this decision is also related to the sampling design of the studies in question, i.e. which quantities are known a priori and thus fixed in the design: neither margins nor overall total fixed (Poisson sampling), only the overall total fixed (multinominal sampling), one margin fixed (product–multinominal sampling), or for two-way tables: the two margins fixed (hypergeometric sampling). But even in Poisson sampling one may opt for a decision rule that conditions on all margins. This type of investigation requires very efficient algorithms to make the determination of the finite sampling distribution at all feasible. The present paper is devoted to a survey of such algorithms in $r \times c$ tables with two fixed margins, i.e. with hypergeometric sampling as H_0 distribution. In contrast to many studies we will treat statistics against general alternatives as well as statistics against various ordered alternatives. The computationally trivial case $r = c = 2$ is not discussed separately.

The main variations are complete enumeration, short-cuts avoiding enumeration of certain tables not in the critical region, and generation of a Monte Carlo sample from the isomarginal family. Implementations based on the algorithms discussed here are planned to be included in the NAG library Mark 12 or 13, and in the package CTPACK [48] which is being developed by the authors and others.

2. Fisher's exact test

Sir Ronald Fisher [12; section 21.01] was the first to indicate that it was possible to construct an 'exact' test, i.e. a test which employs the distribution of the statistic itself, rather than an approximation. He only indicated the procedure for testing a one-sided hypothesis in a 2×2 table using the distribution in one tail. In this case all reasonable statistics are equivalent because they are monotonic functions of each other. Fisher was undoubtedly aware that the procedure could be used for any $r \times c$ table and any test statistic (cf. Yates [51], p. 217, 218). Fisher stipulated that it was obligatory to condition on the marginal frequencies, i.e. one need only compare the values of the statistic for the observed table with other values of the statistic arising from tables with the same marginal totals.

> "If it be admitted that these marginal frequencies by themselves supply no information on the point at issue, namely, as to the proportionality of the frequencies in the body of the table, we may recognize the information they supply as wholly ancillary, and therefore recognize that we are concerned only with the relative probabilities of occurrence of the different ways in which the table can be filled in, subject to these marginal frequencies" [13; p. 48].

Because of its lack of clarity, this explanation is quoted frequently; see Yates [52] for further discussion of this issue.

Fisher's procedure is very simple:
- enumerate all possible tables given the fixed marginal totals;
- compute the statistics of each table;
- sum all probabilities of tables as least as extreme as the observed one.

Nothing in this formulation restricts the procedure to 2×2 tables, and it is the fundament to all published algorithms for exact testing in $r \times c$ contingency tables. The first general treatment of this problem was given by Freeman and Halton [14] who, however, lacked the present-day computing power to make a routine or large-scale solution feasible. A possible alternative to enumeration is to compute the characteristic function, and then invert this function (using Fast Fourier Transforms) to obtain exceedance probabilities, analogous to the procedure developed by Pagano and Tritchler [42] for linear rank statistics and scores.

In Fisher's time the enumeration was almost exclusively restricted to 2×2 tables due to the computational burden of performing enumeration for larger tables. The set of all possible tables given the fixed margins, the *isomarginal family*, can be found in 2×2 tables by letting *one* cell, the *free cell*, run through all its range; after all, there is just a single degree of freedom. Fisher's exact test can in fact be computed with a pocket calculator using some 6 memory registers, say for a, u, r, n, $p_a = r!s!u!v!(n!a!b!c!d!)$, and Σp_a, where summation extends from observed table to current a (for notation see Fig. 1). Note that $v = n - u$, $s = n - r$, and

$$\begin{aligned} p_{a+1} &= p_a * b * c/((a+1)*(d+1)) \\ &= p_a * (u-a)*(r-a)/((a+1)*(n+a-u-r+1)). \end{aligned} \tag{1}$$

The existence of special (books of) tables for this problem [20,31] seems at present a bit overdone. Equally unnecessary is the fact that large and widely distributed computer programs and packages like SPSS [45] and NAG [39] still use the χ^2-approximation in 2×2 tables for n larger than 20 and 40 respectively. The quality of the approximation depends on $m = \min(u, v, r, s)$ more than on n

a	c	r
b	d	s
u	v	n

Fig. 1. Notation for a 2×2 table.

(note: $n \geqslant 2m$). However, even for $m = u = v = r = s = 100$. the quality is not very good: $P(X^2 \geqslant 3.84) = 0.066$, while $P(\chi^2 \geqslant 3.84) = 0.05$.

3. Algorithms structures

Given that we would like to use finite sampling distributions rather than asymptotic ones to test for independence in a contingency table, what are the basic structures of the algorithms to either find the complete discrete distribution or the (exact) significance of the input table? Or more precisely:

(a) Given a pair of margins and a statistic S, find the distribution of S under the hypergeometric distribution of the table.

(b) Given a pair of margins, a statistic S, and an observed value of S, S_0, find $p_0 = p(S \geqslant S_0)$ under the null distribution.

Basic algorithms for these two problems are:

1a. *Basic enumeration algorithm for* (a):
- enumerate the isomarginal family (= set of all possible tables with the given margins);
- for each table
  * calculate its probability p and S,
  * write p and S to a file;
- sort file on key S.

1b. *Basic enumeration algorithm for* (b):
- $p_0 = 0$;
- enumerate the isomarginal family;
- for each table
  * calculate S,
  * if $S \geqslant S_0$ calculate the probability p of this table and add p to p_0;
- print p_0.

2b. *Basic Monte Carlo algorithm for problem* (b):
- $\hat{p}_0 = 0$;
- generate M tables from the isomarginal family (independent, weighted sampling);
- for each table
  * compute S,
  * if $S \geqslant S_0$ then $\hat{p}_0 = \hat{p}_0 + 1/M$;
- print $\hat{p}_0$.

Here $\hat{p}_0$ denotes the Monte Carlo estimate for p_0.

3b. *Basic characteristic function algorithm for problem* (b):
- compute the characteristic function;
- invert it via Fast Fourier Transforms to obtain p.

We have developed an algorithm and a FORTRAN 77 program, FISHER, which performs among other things the complete enumeration and the Monte Carlo algorithms based on the principles described; a User's Manual is [49,50], an Installation and Programmer's Manual is [30].

Common to all algorithms for problem b is the test "if $(S \geqslant S_0)$ then..."; the "if $S = S_0$ then..." part of the test is problematic in any implementation in which S and S_0 are real numbers. In particular the results of this test may depend upon:

- size of the mantissa of a machine real number;
- order of computations at machine level;
- the rounding process in the central processor.

Thus a table included in one calculation may be excluded if the calculation is repeated at a different machine, or at the same machine using a different compiler, or at the same machine and compiler but with different levels of optimization or treatment of rounding. That this problem is not an academic one became clear to us when comparing results between the FORTRAN IV and FORTRAN 77 versions of our program. For instance, the IMSL routine CTPR exhibits this behaviour without any warning from the manual [23]. (In addition, it does not specify which statistic is used: the hypergeometric probability itself.) The only practical solution seems to be to provide the user with an upperbound for $p(S = S_0)$, called the probability mass at S_0. This upperbound is calculated as the sum of the probabilities of all tables for which the calculated S differs less from the calculated S_0 than some small threshold ε, which must exceed the possible inaccuracy in the calculation of S. This upperbound to the probability mass informs the user about two points:

- it is a measure of the discreteness of the null distribution of S at S_0;
- it is a pessimistic upperbound to the possible difference between any two calculations of the significance.

4. Representations

The construction of an algorithm for the enumeration of the isomarginal family is very much dependent upon the way the contingency table and the enumeration process is conceived.

Contingency table. The obvious representation of a contingency table is as a table with r rows and c columns with $\delta = (r-1) \times (c-1)$ degrees of freedom or with δ *free cells* in the first $r-1$ rows and the first $c-1$ columns. The $r+c-1$ *border cells*, i.e. those adjacent to the margins, are fixed given the free cells and the marginal constraints, and the last free cell is located in the south-east corner. A less trivial representation of a contingency table is its so-called *vectorized form*: all cells are arranged into a column vector $t = (t_f, t_b)$ with t_f the vector of free cells, and t_b the vector of border cells. The vectorization we use is: $t_{11}, t_{21}, \ldots, t_{12}, t_{22}, \ldots$; i.e. the order used in FORTRAN, rather than in PASCAL. The constraints on

t_{11}	t_{12}	.	1
t_{21}	t_{22}	.	3
.	.	.	6
3	5	2	10

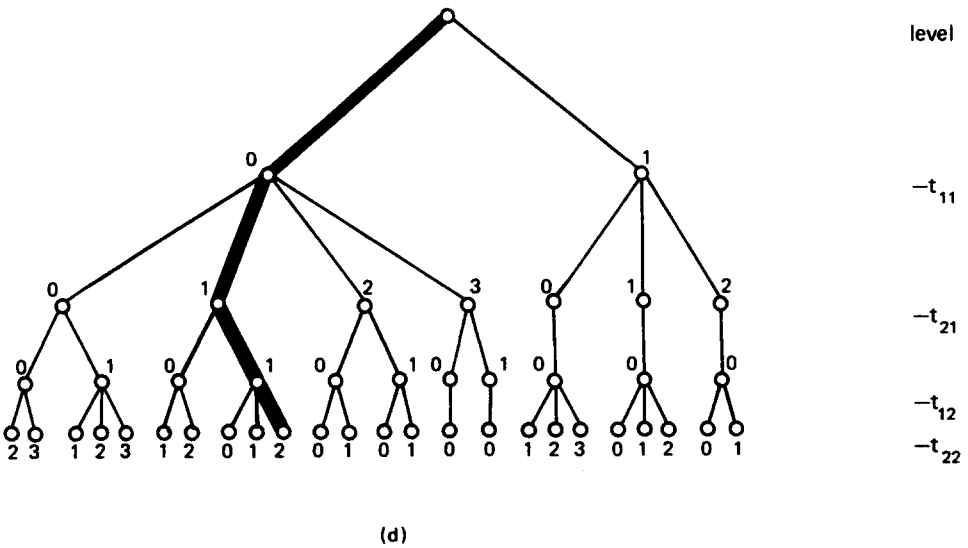
(a)

0	1	0	1
1	2	0	3
2	2	2	6
3	5	2	10

(b)

$$t_{ij} \geqslant 0 \quad (i=1,2;\ j=1,2)$$
$$t_{11} + t_{12} \leqslant 1$$
$$t_{21} + t_{22} \leqslant 3$$
$$t_{11} + t_{21} \leqslant 3$$
$$t_{12} + t_{22} \leqslant 5$$
$$t_{11} + t_{21} + t_{12} + t_{22} \leqslant 2$$

(c)



(d)

Fig. 2. Part (d) is the enumeration tree of the isomarginal family corresponding to part (a). Each table is represented by a leaf of the tree, or equivalently by a path from the root to a leaf. For table (b) this path is marked by heavy lines. Finally (c) lists the restrictions on $(t_{11}, t_{21}, t_{12}, t_{22})$.

the cells due to fixed margins are thought of (more abstractly) as linear constraints on the values of the elements of the vector. We will use both representations of a contingency table, often interchangeably. In both cases for a given element of cell we will refer to *previous cells* as the elements above the given cell in the linearized vector, and to *following cells* as the elements below the given cell.

Isomarginal family. Both the isomarginal family of all tables with the same margins and certain relations between these tables can be represented in various ways. We will discuss a tree, a network, and a convex subset of lattice points in R^d or R^δ with $d = r \times c$.

In the first representation (see Fig. 2) the isomarginal family can be seen as the set of all leaves of a *directed tree*, where each level of the tree corresponds to a (free) cell. At each level we have only a 'partially filled table'; the cells of an initial segment of the table in vectorized form have values, the following cells are 'blank'. At the root the whole table is blank; at the leaves we have completely filled tables. So each leaf corresponds to a generated table, i.e. to a member of the isomarginal family.

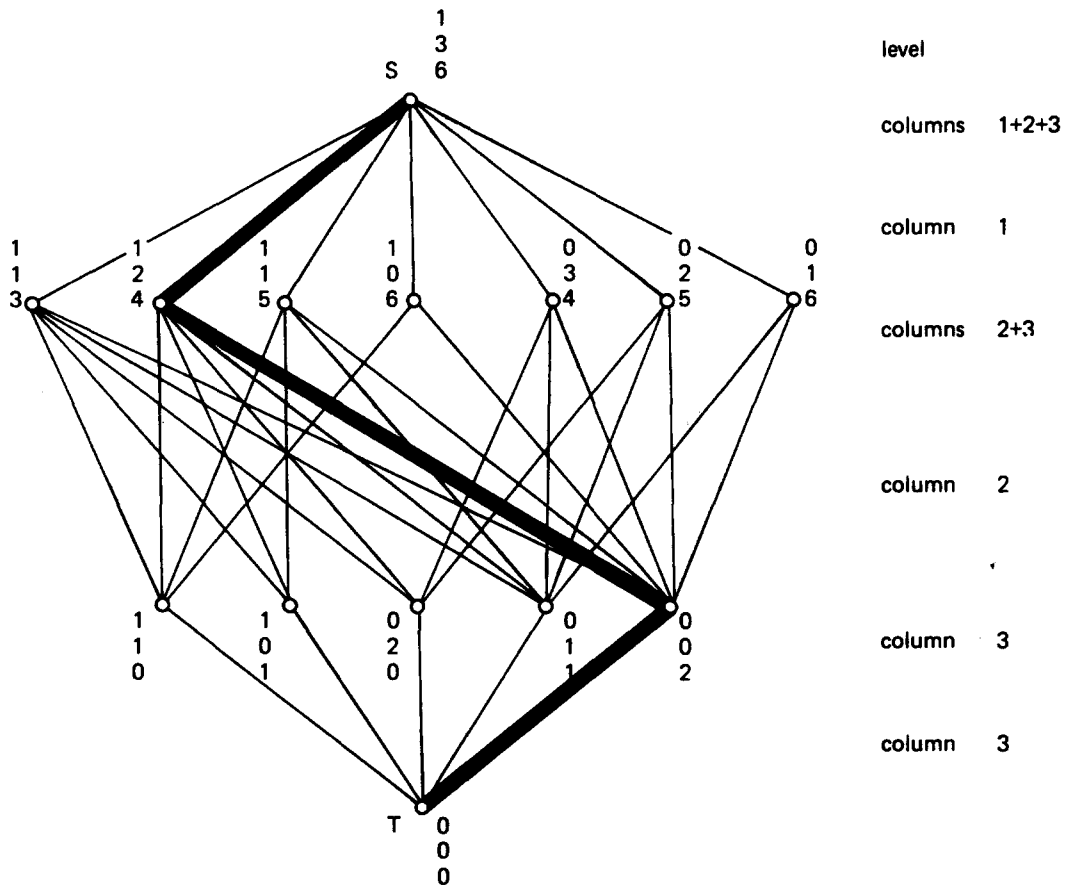


Fig. 3. Representation of the isomarginal family from Figure 2 as an enumeration network. Each table is represented by a path from S to T, and each node represents a sum of columns, from a certain column to the last column. The table from Figure 2(b) is represented by heavy lines.

In the second representation (see Fig. 3), taken from [36], the isomarginal family can be seen as the enumeration of all possible paths through the '*network*' from the initial to the final node. The 'network' is a directed graph with universal source S (representing an empty table) and universal sink U (representing any filled table). This graph contains no (directed) cycles. All maximal (directed) paths start in S, end in T, and have the same length (= number of columns). The isomarginal family is represented by these maximal paths; each of the nodes at distance k from the sink T corresponds one-to-one to the possible pairs of margins of the subtable formed by the last k columns. Each edge from such a node at distance $k-1$ corresponds to a possible $(k-1)$-st column. (But this correspondence between nodes and possible columns is many-to-one.)

In the third representation the isomarginal family is portrayed as a *convex subset* of the lattice $\mathbf{Z}_0^d$ in $\mathbf{R}^d$ with $d = r \times c$ or in $\mathbf{R}^\delta$ with $\delta = (r-1)\times(c-1)$ (see also [5]), where $\{\mathbf{Z}_0 = 0, 1, 2,..\}$ is the set of nonnegative integers. Any member of the isomarginal family corresponds in a one-to-one fashion to a point in $\mathbf{Z}_0^d$. Given the margins m and the vector t_f of free cells, the vector t_b of border cells is a linear function of t_f and m,

$$t_b = -At_f + Bm, \quad \text{say.}$$

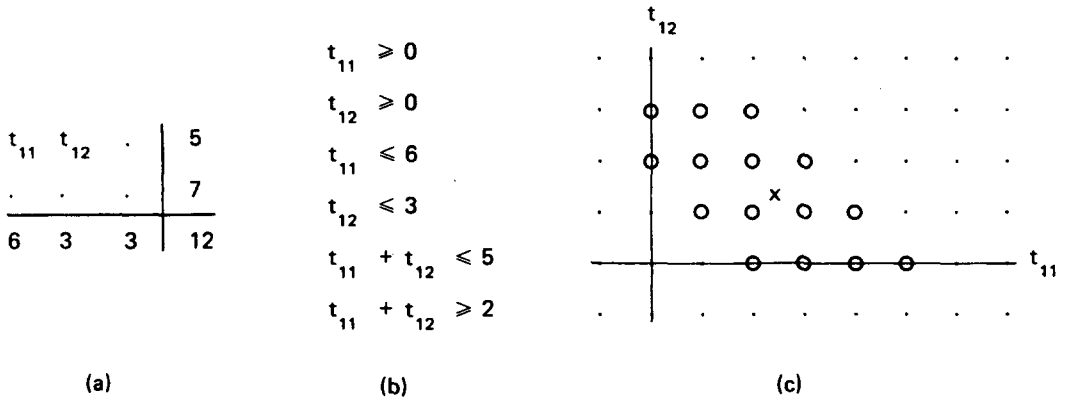


Fig. 4. Part (c) is the convex subset representation of the isomarginal family corresponding to part (a). Part (b) lists the restrictions on (t_{11}, t_{12}).

(All elements of the matrices A and B are zero or one.) Thus the isomarginal family corresponds to

$$\{t \mid t_f \in Z^\delta \text{ and } t_b = -At_f + Bm \geqslant 0\}.$$

Clearly this is a convex subset of the lattice $\mathbf{Z}_0^d$ (see Fig. 4). Note that independence corresponds to a point in $\mathbf{R}^d$ (not necessarily in $\mathbf{Z}_0^d$) in the convex hull. Tables far from independence have large X^2 but small probabilities ($-\log$(probability) is asymptotically proportional to χ^2). In the probability measures there are 'few' extreme tables, by definition, but in the counting measure there are generally very many of such tables.

5. Complete enumeration

Most published algorithms proposed for calculating exact distributions are based on the enumeration of the isomarginal family. In the present section we will discuss the basic structure of such algorithms.

The basic principle behind the enumeration algorithm is already present in the enumeration of all numbers between, say, 000 and 999 in decimal arithmetic. This is done with three counters: the last counter, representing the units, runs fastest. If it overflows (at 9, the 'hibound'), then the second counter, representing the tens is increased by one, and the last counter is reset to 0 (its 'lobound'), etc., cf. Fig. 5a.

In the enumeration of an isomarginal family, we represent the tables in vectorized form, as δ-vectors with $\delta = (r-1)(c-1)$. Each cell is a counter. Again the last counter runs fastest. The loop incrementing the value in this cell is called the *hot loop*, because it is the loop with 'most friction'. If the last cell hits its hibound, one has to find the nearest cell not yet at its hibound, increment this cell, and start anew from this cell on (see Fig. 5b). The only difference with ordinary counting is that lobounds and hibounds are not constant, neither from cell to cell, nor for one cell during the enumeration process. The lobound and

hibound of any cell depend on the values of all previous cells. The flow chart of Fig. 5b allows for this. Crude bounds are 0 as lobound (giving problems in FORTRAN IV) and min (t_{i+}, t_{+j}) as hibound. Exact bounds are discussed in Section 6. A summary of the core of the FISHER [49] algorithm based on this way of counting is given in Fig. 6. In this description the initial values of several loop indices will sometimes be larger than their end values. When this occurs the loop should be skipped. Decreasing loops should be treated analogously. The labels in Fig. 6 correspond to the labels in Fig. 5b.

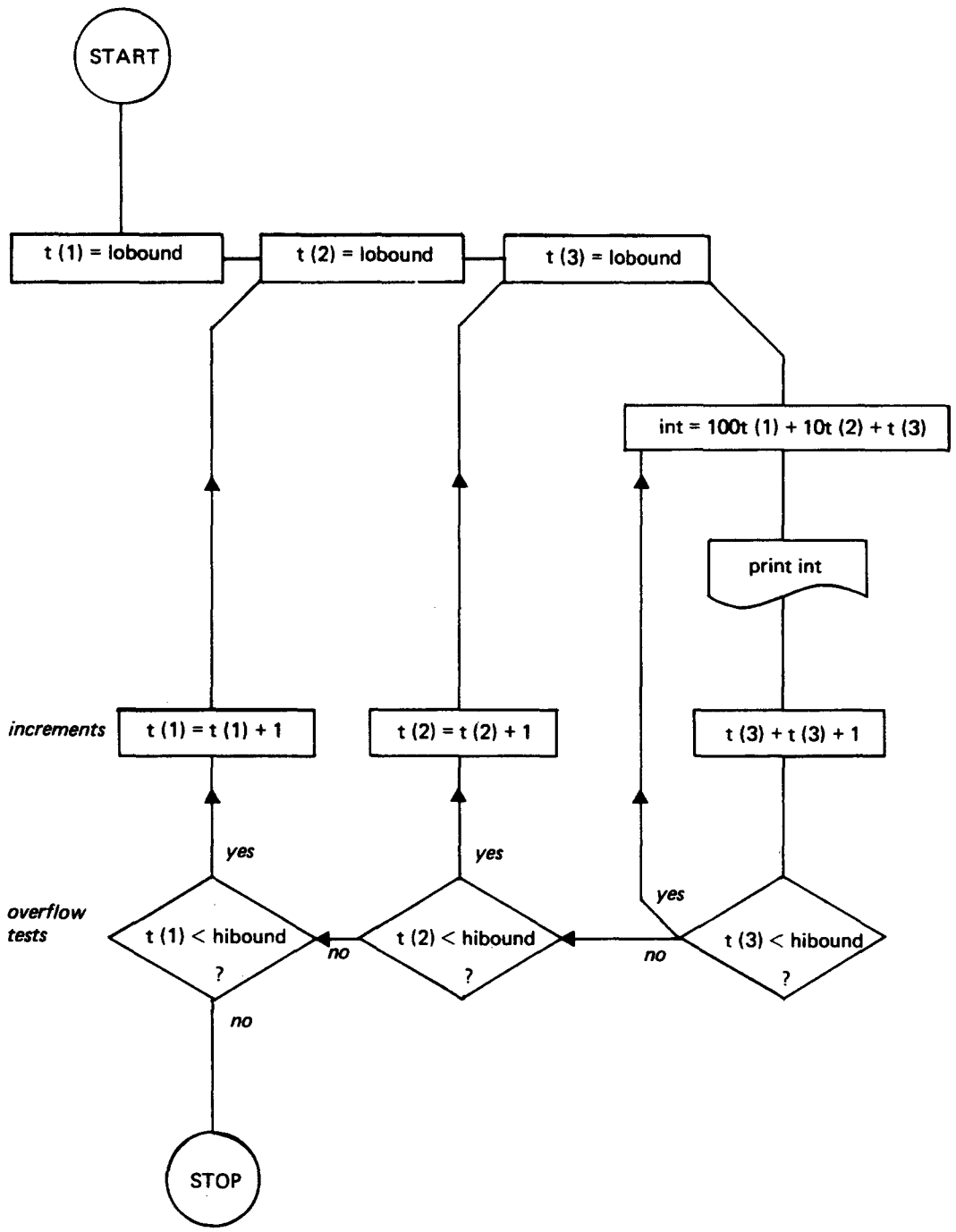


Fig. 5a. How to count from 000 to 999. An algorithm using a fixed number of nested loops.

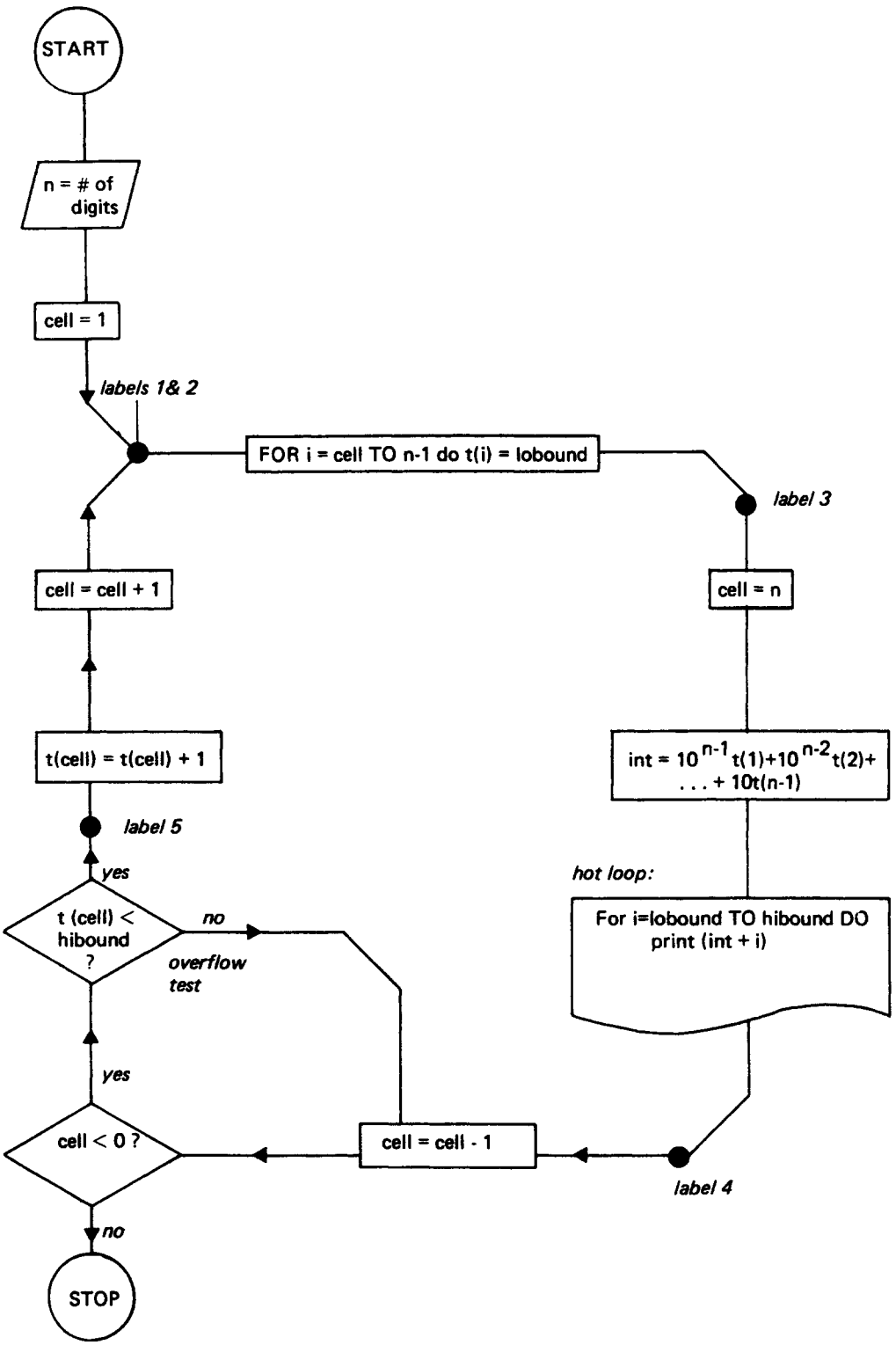


Fig. 5b. Gentleman's generalization, simulating a dynamic number of nested loops. The labels correspond to the labels in Figure 6, which is very similar.

Note that the algorithm of Fig. 5a is most easily implemented as 3 nested loops. A similar implementation for the algorithm if Fig. 5b is not possible, because the depth of the loops is dynamic: it is the number δ of free cells of the

```
                    FISHER ALGORITHM
      (using nested GOTO's and updating of ln p and x2)

1  p = 0;ntable = 0; lobound(1,1) =    ; hibound(i,c) =    ;
2    hibound (r,j) =  ;row = 1;column = 1.
3  Label 1:
4     FOR j = column TO c-2 DO
5    FOR i = row TO r-1 DO
6      table(i,j) = lobound (i,j)
7      update hibound and lobound for next cells
8      update partial sums of the log-prob. and of S_current
9    OD
10   update partial sums of log-prob. and S_current for the last cell
11   row = 1
12    OD
13 Label 2: /*do the same, but for column c-1 and c simultaneously, uptil row r-2*/
14    FOR i = row TO r-2 DO
15     table (i,c-1) = lobound (i,c-1)
16     table (i,c)  = columnmarg (i) -table (i,c-1)
17     update hibound and lobound for cell (i+1,c-1)
18     update partial sums of log-prob. and S_current for both cells
19    OD
20    d0 = lobound (r-1,c-1)
21    dlast    = hibound (r-1,c-1) /*notational convenience*/
22 Label 3:    /* hot loop */
23    ntable = ntable + dlast -d0+1
24    FOR  d = d0 TO dlast DO
25    compute hypergeometric probability p_current of this table
26    compute the value S_current for this table
27    IF (S_current ≥ S_obs) THEN p = p + p_current
28    OD
29 Label 4:    /* overflow: search for previous cell to be increased by one */
30    last row = r-1
31    FOR j = c-1 DOWN TO 1 DO
32    column = j
33    FOR i = last row DOWN TO 1 DO
34        row = i
35        IF (table(i,j) > hibound(i,j)) THEN GOTO Label 5.
36    OD
37    last row = r-1
38    OD
39    print p
40    STOP  /*all tables have been generated*/
   ****
41 Label 5:                    /* prepare for: GOTO Label 1 */
42                        table(row,column)=table(row,column)+1
43                        IF(column=c-1)THEN table(row,c)=table(row,c)+1
44                        update hibound and lobound for next cells
                                              (as many as possible)
45                        update partial sums of log-prob. and S_current
46                        row = row +1
47                        GOTO Label 1.
```

Fig. 6. The basic algorithm of the program FISHER.

table. Our oldest reference for this simulation of a dynamic number of loops in statistical computations is Gentleman [17]. (See also O'Flaherty and MacKenzie [40], whose generalization is not as general as our Fig. 5b.) The algorithm is certainly of interest in its own right.

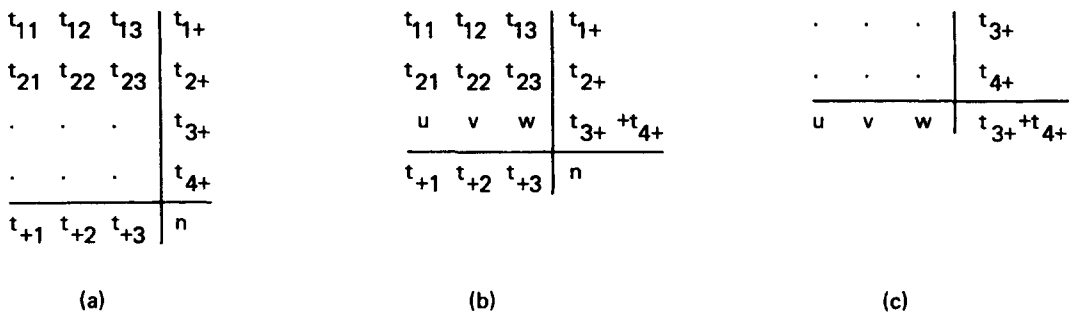


Fig. 7. Generating all 4×3 tables (a) corresponding to the collapsed 3×3 table (b) is equivalent to generating all 2×3 tables (c).

As mentioned above, Boulton and Wallace [5] have designed a truly recursive algorithm, which they have implemented in ALGOL. The basic idea is the following: Given the marginals of an $r \times c$ table $(t_{1+}, \ldots, t_{r+};\ t_{+1}, \ldots, t_{+c})$, collapse the $(r-1)$-st and the r-th row:

$$t_{1+}, \ldots, t_{r-2,+}, (t_{r-1,+} + t_{r+});\quad t_{+1}, \ldots, t_{+c}.$$

To any collapsed or reduced table with these marginas corresponds a non-empty family of unreduced tables with unreduced marginals (see Fig. 7). Generating this family is precisely the problem of generating all $2 \times c$ tables with given marginals (see again Fig. 7). In an analogous way the generation of the $2 \times c$ tables may be recursively reduced to the generation of 2×2 tables. Thus we have reduced the problem of generating all $r \times c$ tables to three smaller problems:
- generating all $(r-1) \times c$ tables with the collapsed margins;
- generating certain $2 \times c$ tables with fixed margins;
- generating certain 2×2 tables with fixed margins.

Despite the mathematical elegance of this method its direct implementation has its drawbacks. First, recursion unfortunately conflicts with the use of FORTRAN, the language most often used for statistical computation. Secondly, this recursive enumeration process requires very many routine calls and exits, leading to a sizeable overhead in the computations. This probably explains why no other authors have literally followed up Boulton and Wallace's work. Instead the efficient complete enumeration algorithms described above all use a variant of the Gentleman [17] procedure.

6. Bounds on cell frequencies and probabilities

General expressions for hibounds and lobounds for cell (i, j) with $i \leqslant r-1$ and $j \leqslant c-1$ can be derived algebraically, but also from the observation that at each cell (i, j) the complete $r \times c$ table may be condensed to a 2×2 table with (i, j) as the free cell (see Fig. 8). In Fig. 8 we use the following notational

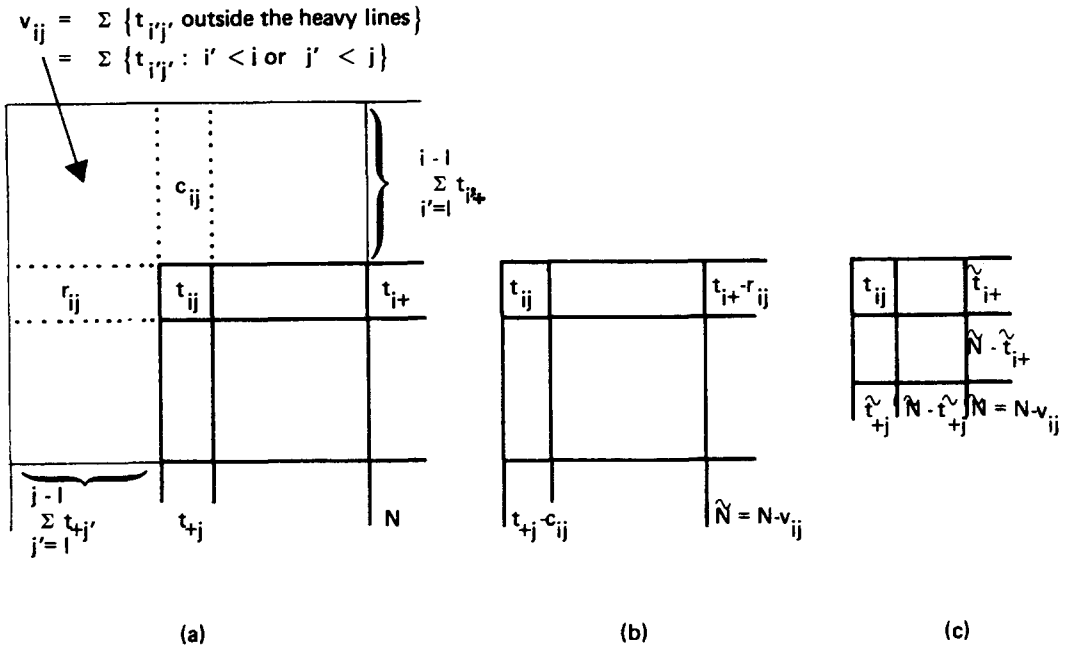


Fig. 8. Stripping and condensation of a table in order to determine the range of t_{ij} given the margins and the previous cells (i.e. the cells $t_{i'j'}$ with $j' < j$ or with $j' = j$ and $i' < i$).

conventions:

$$c_{ij} = \sum_{i'=1}^{i-1} t_{i'j}, \qquad r_{ij} = \sum_{j'=1}^{j-1} t_{ij'},$$

$$v_{ij} = \sum_{i'=1}^{i-1} t_{i'+} + \sum_{j'=1}^{j-1} t_{+j'} - \sum_{i'=1}^{i-1} \sum_{j'=1}^{j-1} t_{ij}, \tag{2}$$

$$\tilde{t}_{+j} = t_{+j} - c_{ij}, \qquad \tilde{t}_{i+} = t_{i+} - r_{ij}, \qquad \tilde{N} = N - v_{ij},$$

using the convention that a summation yields null, if its upper limit is less than its lower limit.

The hibound and lobound for t_{ij} can easily be expressed in these quantities:

$$\text{lobound}(i, j) = \max\left(0, \tilde{t}_{i+} + \tilde{t}_{+j} - \tilde{N}\right),$$

$$\text{hibound}(i, j) = \min\left(\tilde{t}_{i+}, \tilde{t}_{+j}\right).$$

Note that $\tilde{t}_{i+}$, $\tilde{t}_{+j}$, and $\tilde{N}$ are all expressed in terms of previous cells.

Algorithms using $\min(t_{i+}, t_{+j})$ and 0 as crude hibounds and lobounds are *overcomplete* in the sense that they may generate 'impossible' tables, which contain cells with negative frequencies; see, for instance, proposals by Baker [3] and Cox (unpublished, cf. Cox and Plackett [10]). The reason is that 0 is not a true lower bound for each cell. For example, in the 2×3 table

0	b	c	5
2	e	f	5
2	4	4	10

the lowest value for b is not 0 but 1. The overcompleteness of these algorithms makes them unnecessarily inefficient. *Complete* algorithms continuously update the bounds during enumeration and avoid thereby impossible tables.

The hypergeometric probability of a table $t = \{t_{ij}\}$ is

$$P(t) = \left\{\prod_{j=1}^{c} t_{+j}!\right\}\left\{\prod_{i=1}^{r} t_{i+}!\right\} \Big/ \left(N! \prod_{i=1}^{r}\prod_{j=1}^{c} t_{ij}!\right), \tag{3}$$

which may be computed directly for each generated table, or which may be built up via conditional probabilities given the previous cells of a cell (i, j). This conditional probability π_{ij} is

$$\begin{aligned}\pi_{ij} &= \binom{t_{+j} - c_{ij}}{t_{ij}} \binom{(N - v_{ij}) - (t_{+j} - c_{ij})}{(t_{i+} - r_{ij}) - t_{ij}} \Big/ \binom{N - v_{ij}}{t_{i+} - r_{ij}} \\ &= \binom{\tilde{t}_{+j}}{t_{ij}} \binom{\tilde{N} - \tilde{t}_{+j}}{\tilde{t}_{i+} - t_{ij}} \Big/ \binom{\tilde{N}}{\tilde{t}_{i+}}\end{aligned} \tag{4}$$

where c_{ij}, r_{ij}, v_{ij}, $\tilde{t}_{i+}$, $\tilde{t}_{+j}$, and $\tilde{N}$ are defined as in (2).

Note that π_{ij} is the hypergeometric probability of a 2×2 table collapsed from a subtable the original $r \times c$ table (see also Fig. 7). The π_{ij} can be obtained via recursion from the previous cell (cf. [43]). From the computation of the hibounds and lobounds of cell (i, j), c_{ij}, r_{ij} and v_{ij} are already available.

In the computation of probabilities most authors employ log-factorials rather than the factorials themselves in order to prevent integer or real overflow. For larger factorials (say $n > 20$) an extension of Stirling's approximation may be used, as its error of approximation can easily be made smaller than say 10^{-20}, which is often smaller than the error caused by rounding during the computation of the probabilities themselves. Recursive calculation of log-factorials through $\log (n+1)! = \log n! + \log(n+1)$ has the disadvantage of accumulating a large rounding error. It is recommended only if the process is done in higher precision than the calculations using the log-factorials in the enumeration process. Working with logarithms has the additional advantages of adding over multiplication. Generally, it is efficient to compute the log-factorials before the enumeration process itself, and to store them in an array.

7. Computation of statistics

In this section we will discuss the computation of test statistics for which exact distributions are to be calculated (for some examples see Table 1). First we will introduce the necessary terminology.

The variable giving rise to row classification or *row variable* will be denoted by X, and the *column variable* by Y. If X and/or Y are ordinal, then many tests depend upon rank statistics. In a contingency table ranks have many ties. A

Table 1
Test statistics

Alternative hypothesis	Values of row/column variables	Statistic [a]	Transformation	Limit distribution	Degrees of freedom	Reference
1. general	–	X^2 LR EPT FT		χ^2	$(r-1)(c-1)$	[38]
2. row variable ordered; columns are groups; one-way anova	row variable: (mid)ranks	K	$(n-1)$	χ^2	$c-1$	[32; p. 396, example 3]
	row variable scores	η^2	$(n-1)\eta^2$			
3. both variables are ordered	both variables: (mid)ranks	τ_b	$\tau_b/\text{st.dev.}(\tau_b)$	standard normal	–	[26; sect. 49, 56]
	both variables: (mid)ranks	r_s	$r_s\sqrt{(n-2)}/(1-r_s^2)$	student's t (or standard normal)	$n-2$	[27; sect. 31.19)
	both variables: scores	r	$r\sqrt{(n-2)}/(1-r^2)$			

[a] X^2 = Pearson's X^2; LR = -2 log-likelihood-ratio; EPT = exact probability test; FT = Freeman & Tukey's statistic; K = Kruskal–Wallis' statistic; η^2 = correlation ratio; τ_b = Kendall's τ_b; r_s = Spearman's rank correlation; r = Pearson's product–moment correlation.

standard way to deal with ties is the use of *midranks*, i.e. the average of the ranks of the observations in a row (column). Note that the use of midranks is not necessarily the only or the best way to deal with ties (see e.g. Lehmann [32], section 1.4, p. 18). Alternatives to the use of midranks are: the use of 1, 2, 3,..; the use of values assigned by the researcher, or the estimation of 'optimal' values of optimizing some measure of association. If X and/or Y are numerical measurements each row (column) has a natural 'value', which, however, may be transformed if appropriate. Midranks may be interpretated as just one scale of values (or transformation of values). When X and Y have numerical values, Pearson's r and the correlation ratio η^2 may be used as measures of association; if the values are midranks, they reduce to Spearman's r_s and Kruskal–Wallis' K, respectively.

Some authors associate exact testing with the use of the hypergeometric probability P as test statistic, e.g. Freeman and Halton [14], Tate and Hyer [46] and IMSL [23], routine CTPR). A low value of P leads to rejection. Under independence P and X^2 are asymptotically equivalent, because $-2 \ln P$ and X^2 are asymptotically equal. The use of $-2 \ln P$ as a test statistic will be referred to as 'exact probability test' (EPT). Typically this choice of P as a test statistic is not explicitly mentioned, let alone motivated. Of course, this choice leads to fast algorithms, since P has to be computed anyway. Also incomplete enumeration (see Section 9) is simplified. On the other hand, the approximation of the distribution of P by χ^2 for small and moderately sized samples is far inferior to the χ^2-approximation to the distribution of X^2. Cochran's [8,9] well-known rule of thumb (df > 1, 80% of expected values $\geqslant 5$, all expected values $\geqslant 1$) applies to the 95% point of the distribution of X^2, but not to P (as e.g. Tate and Hyer seem to believe). Therefore, it seems inconsistent to test with X^2 when Cochran's rule is satisfied, and to perform an exact test with P when the rule is not satisfied.

Given enumeration procedures to generate all possible tables with fixed margins the exact distributions of any test statistic can be computed. For a well-structured algorithm the inclusion of the computation of the statistics is relatively straightforward as can be seen from Fig. 6 (lines 8, 10, 18, 26, 45) (see for an example Molenaar [37]).

Most calculations for statistics and probabilities can be reduced to the accumulation of sums, indexed either over columns or over cells. Two succesively generated tables of an isomarginal family have a frequently nonempty initial segment of cell frequencies, or set of previous cells, in common. Therefore the partial sums over the initial segment of the earlier table can be used as a starting point for the summations of the next table. As an example consider Pearson's X^2. This statistic is the sum of contributions $X_{ij}^2 = (\text{obs}_{ij} - \text{exp}_{ij})^2/\text{exp}_{ij}$ for each cell (i, j). For a given table, let s_{ij} be the partial sum

$$s_{ij} = \sum_{(i', j') < (i, j)} X_{i', j'}^2$$

for each i and j. Let the successor table be equal to its predecessor table up to (i_0, j_0). The X_{ij}^2 and s_{ij} needs to be redefined only for $(i, j) \geqslant (i_0, j_0)$.

Computations will be simplified for a statistic using (mid)ranks or arbitrary

values by suitable standardization. For example, standardization of X and Y to z_x and z_y will simplify the calculations for r to $r = \sum z_x z_y$. Similarly care should be taken that the most efficient form of a statistic is used for computation. For example,

$$X^2 = \sum_{ij} (\text{obs}_{ij} - \text{exp}_{ij})^2 / \text{exp}_{ij}$$
$$= \sum_{ij} (\text{obs}_{ij}^2 / \text{exp}_{ij}) - N.$$

Obviously, the second expression is more efficient.

For some statistics using ranks, e.g. Kendall's τ and r_s the formulas can be written in such a form that integer arithmetic is possible, thereby avoiding problems of rounding errors, and difficulties in comparing whether two values of a statistic are the same or not (see also Section 3). As a nontrivial example we will look at r_s. In a contingency table with fixed margins r_s takes the form

$$r_s = \frac{\text{covariance}(X, Y)}{\text{st.dev}(X)\text{st.dev}(Y)} = \frac{\sum_{i,j} t_{ij}(X_i - \bar{X})(Y_j - \bar{Y})}{\sqrt{\sum_i t_{i+}(X_i - \bar{X})^2}\sqrt{\sum_j t_{+j}(Y_j - \bar{Y})^2}}$$

where X_i denotes the row midrank, Y_j the column midrank, $\bar{X}$ the average of X_i and $\bar{Y}$ the average of Y_i. The denominator only depends on the fixed margins and the fixed midranks, thus it is a constant which may be computed outside the enumeration proper. When we rewrite r_s as

$$r_s = \left\{ \sum_{i,j} t_{ij} X_i Y_j - \tfrac{1}{2} N^2 (N+1) \right\} / \text{constant}$$

we see that also the second term in the numerator is a constant. Moreover, t_{ij}, $2X_i$, and $2Y_i$ are integers for each i and j. If, therefore, the first term multiplied by 4 is computed and updated during enumeration, the variable part of r_s may be computed in integer arithmetic. Details on the computation of other statistics mentioned in Table 1 may be found in [30].

An important general principle in the computation of statistics and probabilities is that computations should be done as much as possible in the outer loops, or, if possible, outside the enumeration proper. For example, the storage of log-factorials may be handled in this way, as pointed out at the end of Section 6. The usefulness of such procedures depends both on the time necessary for recomputing as compared to the time required for retrieving an array element, and on the time gain compared to the storage requirements.

A facility for a user-defined statistic can easily be accommodated, if not very efficiently. Each time a new table is generated, its frequencies are passed on to a user supplied function, which returns the value of the user defined statistic. This routing call within the hot loop slows down the enumeration considerably for two reasons. First a routine call inside a loop makes loop optimization difficult, if not impossible for most FORTRAN compilers. Second, one has to compute each value

of the statistic from scratch, and one cannot store intermediate results, as discussed in the last paragraph. Nevertheless, this facility is very powerful in its wide applicability, if only as a prototype. For example, De Leeuw and Van der Burg [11] used this facility to compute the exact significance of canonical correlations in 3×3 and 4×3 tables.

If the user supplied function also returns a user defined probability, then the user can replace the hypergeometric distribution by another one. This, too, is very easy to implement.

8. Fine tuning

In this section we will discuss a variety of details which could be termed 'algorithmic tricks', in that they are not essential for understanding the algorithm, but their influence on its performance can be substantial.

The value of a statistic and the probability of a newly generated table is often only computed after the table has been generated in its entirety. A more efficient procedure is to update the statistics and probability during enumeration, i.e. to use partial sums containing the contributions to a statistic and the probability of all previous cells. This involves reserving separate arrays for the partial sums of each statistic and probability (see also lines 8, 10, 18 and 45 of Fig. 6). Generally the gain in computing time far outweighs the increase of storage requirements and retrieval time.

Against non-ordered alternative hypotheses sorting marginal totals to increasing sizes towards the south-east corner of a table will ensure that the range of the last loop element (i.e. the hot loop) will be as large as possible, and we assume that this will increase the efficiency of the enumeration. Similarly, given our vectorization $(t_{11}, t_{21}, \ldots)$ the contingency table should be given the form $r \geqslant c$, as in that way loop overhead by going from one column to the next is minimized. For consider the choice between two alternatives to nesting of loops:

```
(A) FOR i = 1 TO k DO
      FOR j = 1 TO l DO
        work1
      OD
      work2
    OD
```

and

```
(B) FOR j = 1 TO l DO
      FOR i = 1 TO k DO
        work1
      OD
      work3
    OD
```

The success of both optimizations depends on the following:

- The efficiency of retrieval of array elements in work1 with respect to their

prescribed order. The efficiency depends on the language (PASCAL behaves differently from FORTRAN), and machine and operating system architecture.
- Loop overhead: if $k < l$ then (A) incurs less overhead.
- The amount of work in k * work2 compared to l * work3. If work2 and work3 are nearly equal in workload, and $k < l$ then, again (A) is preferable.

Above we mentioned that as a general principal computations should be done as much as possible in outer loops or even outside the enumeration proper. This argument is valid a fortiori with respect to the hot loop, as the program spends more time in this part of the algorithm than anywhere else. Therefore, simplification in this particular loop will have a strong effect on the efficiency of the algorithm. In discussing such simplifications it will be useful to view the last free element or cell as the single free cell of a 2×2 table. The range of this cell is thus the range of the hot loop. We will utilize the notation as given in Fig. 1, i.e. $a = t_{r-1,c-1}$, $b = t_{r-1,c}$, $c = t_{r,c-1}$, and $d = t_{r,c}$.

For optimizing computations for the probabilities in the hot loop use may be made of simple recurrence relations. If a, b, c, d are the values of the 2×2 table before entry into the hot loop and p^* is the partial sum of the log-probabilities of the previous cells upto cell $(r-1, c-1)$, then for any value of the loop index i,

$$p_i = \exp(p^* - \ln(a+i)! - \ln(d+i)! - \ln(b-i)! - \ln(c-i)!).$$

Rather than performing exponentiations inside the hot loop, we should evaluate p_i for the first value of i before loop entry, and inside the loop one may use

$$p_{i+1} = p_i(b-i)(c-i)/(a+i+1)(d+i+1).$$

But one should beware of possible underflow problems. Note that first p_i increases and after reaching a maximum 'in the middle' decreases monotonically. The first p_i may be very small, while 'in the middle', the values of p_i may be substantial. Values of p_i smaller than, say, 10^{-20} may safely be neglected. One will never enumerate as many as 10^{10} tables, so these neglected tables will never contribute more than 10^{-10} to the significance, which is practically nothing. Similarly, one may stop at the other tail if p_i decreases below 10^{-20}.

Analogously to the efficient updating of the probabilities in the innermost loop as described above, most statistics may be updated through simple recursion as well, for the k-th order polynomial functions of the loop index may be done by differencing up to order k, which can be updated by k additions. X^2, K, and η^2 are quadratic in the index, while τ, r_s, and r are linear. This implies that the first group can be updated linearly in the first cell, and the latter group by a constant, which may be computed beforehand.

As an example, X^2 is quadratic in a (for notation see Fig. 1), and can be computed recursively as follows:

$$\begin{aligned} X^2(a+1, b-1, c-1, d+1) &= X^2(a, b, c, d) + (2a+1)/\exp a + (-2b+1)/\exp b \\ &\quad + (-2c+1)/\exp c + (2d+1)/\exp d \\ &= X^2(a, b, c, d) + B + C, \end{aligned}$$

where exp a is the constant expected value for the free cell a, and the other expected values are defined analogously, $C = 1/\exp a + \cdots + 1/\exp d$, and $B = 2(a/\exp a - b/\exp b - c/\exp c + d/\exp d)$. Note that C depends only on the (fixed) expected values and may be computed before enumeration, and B needs to be computed only once before the hot loop starts. So after computing initial values for $X^2(a,b,c,d)$, B, and C, the updating becomes

$$X^2(a+i,\ b-i,\ c-i,\ d+i)$$
$$= X^2(a+(i-1),\ b-(i-1),\ c-(i-1),\ d+(i-1)) + B + iC.$$

The updating of r_s is even simpler:

$$r_s(a+1,\ b-1,\ c-1,\ d+1)$$
$$= r_s(a,\ b,\ c,\ d) + aX_{nr-1}Y_{nc-1} - bX_{nr-1}Y_{nc} - cX_{nr}Y_{nc-1} + dX_{nr}Y_{nc}$$
$$= r_s(a,\ b,\ c,\ d) + C,$$

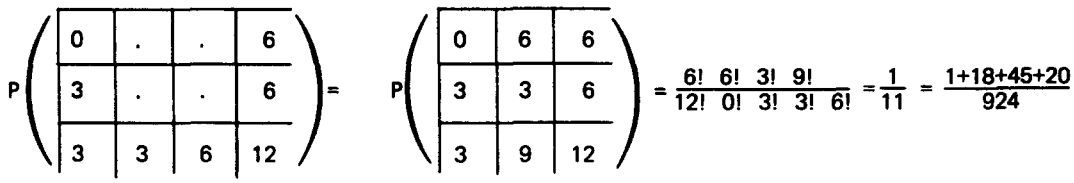
where C may be computed outside the hot loop.

9. Incomplete enumeration

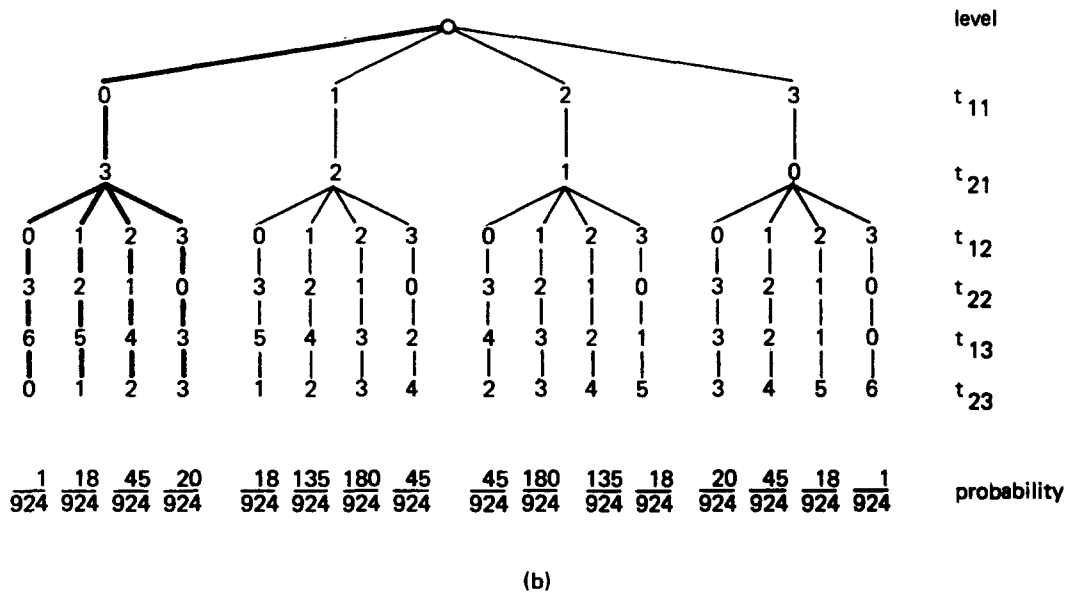
In the basic enumeration algorithm for the calculation of the entire distribution of a given statistic S (basic algorithm 1a; Section 3), one cannot avoid complete enumeration of the isomarginal family. But for the calculation of significances (basic algorithm 1b; Section 3) one only has to enumerate the critical region (or its complement). And even in the enumeration of points in the critical region shortcuts are possible by methods which allow computing the sum of the probabilities of certain subsets of the critical region more efficiently than by complete enumeration.

A convenient representation of the isomarginal family for studying the 'critical region' CR is the third representation mentioned in Section 4: the isomarginal family is portrayed as a convex subset C of the lattice $\mathbf{Z}_0^d$. The critical region CR is the subset of C consisting of tables with values of S at least as large as the observed value S_0. Thus the size of C depends very much on the value S_0. For χ^2-statistics $C \setminus$ CR is approximately ellipsoidal, and for moderate values of S will contain far fewer tables than CR (cf. last remark of Section 3). But for a linear statistic like τ or r, CR is an intersection of C with a half space.

The basic tool to shortcut complete enumeration is the following [41,36]: Consider the first representation (Fig. 2) in Section 3. At any node of the tree we know the values in all cells of an initial segment of the table in vectorized form. From these it is often easy to determine the largest and/or smallest possible values of S for all tables with this initial segment (for all leaves sprouting from the node). Consider, for example, $X^2 = \Sigma(O-E)^2/E$. At any node the sum $\Sigma(O-E)^2/E$ over all cells in the initial segment up to the node clearly is a lower bound to the value of X^2 of any corresponding complete table. So if this partial sum already exceeds the observed value, then all tables with this initial segment



(a)



(b)

Fig. 9. Sum of the probabilities of all tables with a fixed initial segment. Here $r = 2$, $c = 3$, and initial segment is $t_{11} = 0$, $t_{21} = 3$. Part (a) gives the margins, and shows the reductions to a table in which all cells are known; part (b) gives the enumeration tree, where the initial segment is shown by heavy lines.

are in CR. The sum of the probabilities of all tables with a given initial segment is easily computed from (4). Especially if the initial segment precisely consists of, say, the first c' columns, this sum of probabilities is the hypergeometric probability of the $r \times (c' + 1)$ table obtained by collapsing columns $c' + 1$ up to c to one column. See Fig. 9.

In the implementation of this shortcut one has to decide on the following two points. First, how often does one want to test whether a shortcut is feasible; e.g. at each free cell or at each completed column. In FISHER we opted for the last solution on rather intuitive grounds. Secondly, one has to decide whether to test for the minimum of S, for the maximum of S or for both. For instance, for $S = X^2$ it hardly seems worthwhile to test whether a branch is completely in the complement of CR, i.e. to test for the maximum X^2, firstly because this maximum is rather costly to compute, secondly because relative few branches have this property in a large isomarginal family.

In Section 8 we mentioned another way to skip certain tables, viz. sets of tables with a total probability less than, say, 10^{-20}. In Section 8 this has been applied to the hot-loop only, but the argument can also be applied to all tables corresponding to a given initial segment of the vectorized table. The total probability of such a set is very easy to compute.

Saunders [44] indicates a way to skip enumerating a number of tables in the special case that two or more rows (columns) are equal. This situation is likely to occur in designed experiments, but in any case its testing can be done at virtually no cost. Generalization to arbitrary patterns of equal row and column marginal is, however, not easy.

Yet another idea comes from Goodall [19]. He introduces the following equivalence relation on an isomarginal family. Two tables t and s with the same margins are equivalent if and only if the vector (t_{ij}) is a permutation of the vector (s_{ij}). Obviously equivalent tables have the same hypergeometric probability. For the EPT it, therefore, suffices to enumerate the equivalence classes and determine their sizes. However, these two subproblems still seem to be open. Moreover, other test statistics such as X^2 and the likelihood ratio statistic (LR) are not constant on these equivalence classes.

10. Monte Carlo methods

Table generation by simulating draws from a hypergeometric distribution, is described by Patefield [43]; an earlier, less efficient algorithm is due to Boyett [6]. Essentially the cells from the table are filled one by one, where each cell frequency is drawn according to the distribution described by (3) or (4). After all, this is the distribution of a cell conditional on the distribution of its predecessors. Two successively generated tables will in general have an empty or very short equal initial segment, unlike two successive tables generated by a basic enumeration algorithm. Hence it is not possible to store partial sums of statistics, as was outlined in Section 7, which would have yielded important savings in subsequent computations. In Monte Carlo simulation we know of no substantial improvements to the calculation from scratch of the statistic for each newly generated table. The generation process and the calculation of the statistic per table are much more time consuming than the generation of the next table in the complete enumeration process. It turns out that the former process typically generates 20–100 times less tables per second than the latter. But it has the advantage that the number of tables generated is fixed in advance, and the computing time does not depend greatly on N, as opposed to the N^N-behaviour of the complete enumeration process.

11. Power and null distributions differing from the hypergeometric

For power calculations one has to replace the hypergeometric null distribution of the tables in the isomarginal family by the distribution under the specified

alternative. Also one has to reconsider whether or not one wants to compute the power conditional on the observed margins or not. If power calculations are performed before the observations are collected, then this is obviously impossible. But this paper only discusses distributions conditional on the margins because of the algorithmic similarity of their algorithms.

Consider the unconditional alternative distribution

$$\text{multinominal}(N, \pi_{11}, \pi_{21}, \ldots, \pi_{rc})$$

where for each of the N independent observations π_{ij} is a probability that the observation will fall in cell (i, j). Then the conditional probability of observing $t = (t_{11}, t_{12}, \ldots, t_{rc})$ is

$$P(t) = c \prod_{i=1}^{r} \prod_{j=1}^{c} \pi_{ij}^{t_{ij}} / t_{ij}!$$

where

$$c = 1 \Big/ \left(\sum_{\{\tau_{ij}\}} \prod_{i=1}^{r} \prod_{j=1}^{c} \pi_{ij}^{\tau_{ij}} / \tau_{ij}! \right)$$

where the summation is over all tables $\tau = \{\tau_{ij}\}$ in the isomarginal family of t. For power calculations we know of no way to avoid or shortcut the enumeration of all tables in order to compute c. During the enumeration one works with $P(\cdot)/c$ or $\log P(\cdot) - \log c$, rather than with the probability $P(\cdot)$ itself, and only upon completion of the enumeration one can multiply all probabilities with c. In particular we are not aware of any efficient Monte Carlo algorithm for power calculations.

For null distributions differing from the hypergeometric the situation is very similar. If the probabilities have an explicit representation allowing easy computation, then this can be implemented in the same way as the calculation of the hypergeometric probabilities and of the statistics. If only unconditional probabilities are readily available one has to enumerate all tables in the isomarginal family, to obtain the factor c, just as above.

12. History

Since Klotz [28] and Goodall [19], many authors have published algorithms for treating the enumeration of the isomarginal family, and for calculating its size. The latter problem has essentially the same structure, be it that only the number of tables is counted, and that no probabilities or statistics have to be computed.

A striking aspect of this literature is that so many individuals have attempted to solve the problem, often with different ultimate aims, and secondly, that nearly each of them introduced some minor or major improvement by proposing reordering of the input table, using flexible lobounds, performing certain calcula-

Table 2
Overview of algorithms for calculating distributions in contingency tables with fixed margins mentioned in the literature (df > 2)

A. Enumeration Algorithms	
Structure	
– straightforward filling of table	Freeman & Halton [14], March [34], Hancock [21], Cantor [7], Howell & Gordon [22], Kannemann [24,25]
– simulating a dynamic number ($r \times c$) of nested 'for-loops'	Agresti & Wackerly [2], Mehta & Patel [35,36], Pagano & Taylor-Halvorson [41], Verbeek, Kroonenberg & Kroonenberg [49,50], Klotz [28], Klotz & Teng [29], Baker [3], Cox & Plackett [10]
– true recursion	Boulton [4]
Completeness	
– overcomplete (also impossible tables)	March [34], Baker [3], Cox & Plackett [10],
– complete (only possible tables)	Hancock [21], Cantor [7], Howell & Gordon [22], Klotz [28], Klotz & Teng [29], Agresti & Wackler [7], Verbeek, Kroonenberg & Kroonenberg [49], Kannemann [24,25]
– incomplete (sufficient subset)	Mehta & Patel [35,36], Pagano & Taylor-Halvorson [41], Saunders [44], Verbeek, Kroonenberg & Kroonenberg [50]
Statistics	
– None, only number of tables	Abrahamson & Moser [1], Good [18], Gail & Mantel [16], Klotz & Teng [29], Boulton & Wallace [5],
– Exact Probability Test (EPT)	Mehta & Patel [35,36], Pagano & Taylor-Halvorson [41], IMSL-CTPR [23], Boulton [4], Howell & Gordon [22], March [34], Cantor [7]
– User supplied function	Baker [3], Cox & Plackett [10], Kannemann [24,25]
– Various statistics	Agresti & Wacker [2] (Kendall's τ, Kruskal & Goodman's τ, EPT, X^2) Klotz [28] (Wilcoxon); Klotz & Teng [29] (K); Verbeek, Kroonenberg & Kroonenberg [49,50] (EPT, X^2, G^2, FT, K, r_s, r, τ, η^2; user supplied function)
B. Monte Carlo Algorithms	Boyett [6; AS 144], Patefield [43; AS 159]

tions (like computing factorials, or rather log-factorials) outside the main enumeration process, updating of statistics and probabilities during enumeration, rather than computing them at one place in the algorithm, etc.

One of the objectives of the present paper, and of constructing our program FISHER [49,50] was to bring all these different improvements, which were scattered through the literature and algorithms, into one framework, or to quote Tom Lehrer [33; p. 29]: "*Every chapter I stole from somewhere else*": Many authors published some tricks, we publish all. In Table 2 we given an overview of the published algorithms we are aware of, with their major characteristics.

13. Conclusions

With today's computing power the calculation of exact significances in $r \times c$ tables has become feasible. In fact, it is only a small job for a wide class of tables from samples too small to rely with blind faith upon asymptotic approximations, but yet large enough to allow statistical inference. Nevertheless, one should not use methods like these routinely, if asymptotic methods are also appropriate. We stated that the job is small in many cases, but of course it is unwieldy for larger samples.

The two basic algorithms discussed here are complete or incomplete enumeration and Monte Carlo sampling. Each is efficient for a different set of tables. Generally speaking, Monte Carlo sampling is called for, if enumeration becomes infeasible because of the large sample size. The computational work of Monte Carlo sampling depends only little on sample size, but the number of tables generated per second is much smaller than for enumeration, typically by a factor $d = r \times c$.

The enumeration algorithm is interesting in its own right, especially the simulation of a dynamic number of nested FOR loops with dynamic bounds, which essentially goes back to Gentleman [17]. But it is not very complex. Yet its implementation requires careful analysis of several interesting numerical problems: overflow due to factorials like $N!$; underflow of probabilities used in a recurrence relation; testing "IF ($S \geqslant S_0$) THEN" for reals S and S_0; numerical accuracy of statistics and of probabilities. These problems have not been discussed in other publications on contingency tables. The implementation is also interesting because of the gains in efficiency that are possible by several nontrivial enhancements. The nontriviality is illustrated by the large number of authors contributing to the state of the art surveyed here, as can be seen from Table 2.

References

[1] M. Abrahamson and W.O.I. Moser, Arrays with fixed row and column sums, *Discrete Mathematics* **6** (1973) 1–14.

[2] A. Agresti and D. Wackerly, Some exact conditional tests of independence for $R \times C$ cross-classification tables, *Psychometrika* **42** (1977) 111–125.

[3] R.J. Baker, Algorithm AS 112. Exact distributions derived from two-way tables, *Appl. Statist.* **26** (1977) 199–206; Corr. **27** (1978) 109; Corr. **30** (1981) 106–107.

[4] D.M. Boulton, Remark on algorithm 434, *Commun. ACM* **17** (1974) 326.

[5] D.M. Boulton and C.S. Wallace, Occupancy of a rectangular array, *Computer J.* **16** (1973) 57–63.

[6] J.M. Boyett, Algorithms AS 144. Random R×C tables with given row and columns totals, *Appl. Statist* **28** (1979) 329–332.

[7] A.B. Cantor, A computer algorithm for testing significance in M×K contingency tables, *Fifth Proceedings of the Statistical Computing Section of the American Statistical Association* (ASA, Washington, DC, 1979).

[8] W.G. Cochran, The χ^2 test of goodness of fit, *Ann. Math. Statist.* **23** (1952) 315–345.

[9] W.G. Cochran, Some methods for strengthening the common χ^2 tests, *Biometrics* **10** (1954) 417–451.

[10] M.A.A. Cox and R.L. Plackett, Small samples in contingency tables, *Biometrika* **67** (1980) 1–13.
[11] J. de Leeuw and E. van der Burg, The permutational limit distribution of generalized canonical correlations, Technical Report RR-85-04, Department of Data Theory, University of Leiden (1985).
[12] R.A. Fisher, *Statistical Methods for Research Workers*, 5th Edition (Oliver and Boyd, Edinburgh, 1934).
[13] R.A. Fisher, The logic of inductive inference (with discussion), *J. Roy. Statist. Soc.* **98** (1935) 39–54.
[14] G.H. Freeman and J.H. Halton, Note on an exact treatment of contingency, goodness of fit, and other problems of significance, *Biometrika* **38** (1951) 141–149.
[15] M.F. Freeman and J.W. Tukey, Transformations related to the angular and the square root, *Ann. Math. Statist.* **21** (1950) 607–611.
[16] M. Gail and N. Mantel, Counting the number of $r \times c$ contingency tables, *J. Amer. Statist. Assoc.* **72** (1977) 859–862.
[17] J.F. Gentleman, Algorithm AS 88. Generation of all ${}_N C_R$ combinations by simulating nested FORTRAN DO loops, *Appl. Statist.* **24** (1975) 374–376.
[18] I.J. Good, On the application of symmetric Dirichlet distributions and their mixtures to contingency tables, *Ann. Statist.* **4** (1976) 1159–1189.
[19] D.W. Goodall, Contingency tables and computers, *Biometrie-Praximetrie* **9** (1968) 113–119.
[20] D. Goyette and M. Mickey, Chi-square probabilities for 2×2 tables, BMDP Technical Report no. 15., Dept. of Biomathematics, UCLA (1975).
[21] T.W. Hancock, Remark on algorithm 434, *Commun. ACM* **18** (1975) 117–119.
[22] D.C. Howell and L.R. Gordon, Computing the exact probability of an r by c contingency table with fixed marginal totals, *Behavior Res. Meth. Instrum.* **8** (1976) 317.
[23] IMSL, Edition 8 (IMSL Inc., Houston, 1980).
[24] K. Kannemann, The exact evaluation of 2-way cross-classifications: An algorithmic solution, *Biom. J.* **24** (1982) 157–169.
[25] K. Kannemann, The exact evaluation of 2-way cross-classifications: Sequel. A fugal algorithm, *Biom. J.* **24** (1982) 679–684.
[26] M.G. Kendall, *Rank Correlation Methods* (Griffin, London UK, 1948, 1975).
[27] M.G. Kendall and A. Stuart, *The Advanced Theory of Statistics*, Vol. 3, 3rd Edition (Griffin, London, 1973).
[28] J.H. Klotz, The Wilcoxon, ties, and the computer, *J. Amer. Statist. Assoc.* **61** (1966) 772–787; Corr. **62** (1967) 1520–1521.
[29] J.H. Klotz and J. Teng, One-way layout for counts and the exact enumeration of the Kruskal-Wallis H distribution with ties, *J. Amer. Statist. Assoc.* **72** (1977) 165–169.
[30] S. Kroonenberg and A. Verbeek, Programmer's and Installation Manual to FISHER. Technical Report, Sociological Institute, University of Utrecht, The Netherlands (in preparation).
[31] H.-P. Krüger, W. Lehmacher, and K.-D. Wall, *Statistische Tafeln für Sozial- und Biowissenschaftler. Die Vierfelder-Tafel* (Gustav Fischer, Stuttgart FRG, 1980).
[32] E.L. Lehmann, *Nonparametrics: Statistical Methods Based on Ranks* (Holden-Day, San Francisco, CA, 1975).
[33] T. Lehrer, *Too many songs by Tom Lehrer with not enough drawings by Ronald Searle* (Eyre Methuen, London U.K., 1981).
[34] D.L. March, Algorithm 434. Exact probabilities for $R \times C$ contingency tables. *Commun. ACM* **15** (1972) 991–992.
[35] C.R. Mehta and N.R. Patel, A network algorithm for the exact treatment of the $2 \times k$ contingency table, *Commun. Statist. Simul. Comput.* **9** (1980) 649–664.
[36] C.R. Metha and N.R. Patel, A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables, *J. Amer. Statist. Ass.* **78** (1983) 427–434.
[37] I.W. Molenaar, Mokken scaling revisited, *Kwantitatieve Methoden* **3**(8) (1982) 145–164.
[38] A.M. Mood, F.A. Graybill, and D.C. Boes, *Introduction to the Theory of Statistics*, 3rd Edition (McGraw-Hill, New York, 1974).

[39] *NAG FORTRAN Library Manual*, Mark 9 (Numerical Algorithm Group, Oxford, UK, 1982).
[40] M. O'Flaherty and G. McKenzie, Algorithm AS 172, Direct simulation of nested FORTRAN DO-loops, *Appl. Statist.* **31** (1982) 71–74.
[41] M. Pagano and K. Taylor-Halvorson, An algorithm for finding the exact significance levels of r×c contingency tables, *J. Amer. Statist. Assoc.* **76** (1981) 931–934.
[42] M. Pagano and D. Tritchler, On obtaining permutation distributions in polynomial time, *J. Amer. Statist. Assoc.* **78** (1983) 435–440.
[43] W.M. Patefield, Algorithm AS 159. An efficient method of generating random $R \times C$ tables with given row and column totals, *Appl. Statist.* **30** (1981) 91–97.
[44] I.W. Saunders, Algorithm AS 205. Enumeration of $R \times C$ tables with repeated row totals, *Appl. Statist.* **33** (1984) 340–352.
[45] SPSS, Inc., *SPSS-X User's Guide* (McGraw-Hill, New York, 1983).
[46] M.W. Tate and L.A. Hyer, Inaccuracy of the χ^2-test of goodness of fit when expected frequencies are small, *J. Amer. Statist. Assoc.* 68 (1973) 836–841.
[47] A. Verbeek and P.M. Kroonenberg, Exact χ^2 tests of independence in contingency tables with small numbers, Preprint no. 122, Dept. of Mathematics, University of Utrecht, The Netherlands; presented at 12th EMS, Varna, Bulgaria (1979).
[48] A. Verbeek, D. Denteneer, P.M. Kroonenberg, S. Kroonenberg and J. Weesie, CTPACK, a library of subroutines and programs for the analysis of contingency tables, Technical Report, Sociological Institute, University of Utrecht, The Netherlands (1984).
[49] A. Verbeek, P.M. Kroonenberg and S. Kroonenberg, User's Manual to FISHER. A program to compute exact distributions and significance levels of statistics used for testing independence in $r \times c$ contingency tables with fixed marginal totals, Technical Report, Sociological Institute, University of Utrecht, The Netherlands (1983).
[50] A. Verbeek, P.M. Kroonenberg and S. Kroonenberg, *User's Manual to FISHER*, Version 2 (in prep.)
[51] F. Yates, Contingency tables involving small numbers and the χ^2 test. *J. Roy. Statist. Soc. Suppl.* **1** (1934) 217–235.
[52] F. Yates, Tests of significance for 2×2 contingency tables (with discussion). *J. Roy. Statist. Soc. A* **147** (1984) 426–463.