KYBERNETIKA — VOLUME 39 (2003), NUMBER 1, PAGES 55–74

ON THE COMPUTATION OF THE EXACT DISTRIBUTION OF POWER DIVERGENCE TEST STATISTICS

MARCO A. MARHUENDA, YOLANDA MARHUENDA AND DOMINGO MORALES

In this paper we introduce several algorithms to generate all the vectors in the support of a multinomial distribution. Computational studies are carried out to analyze their efficiency with respect to the CPU time and to calculate their efficiency frontiers. The proposed algorithm is used to calculate exact distributions of power divergence test statistics under the hypothesis of uniformity. Finally, several exact power comparisons are done for different divergence statistics and families of alternatives to the uniformity hypothesis.

Keywords: multinomial distribution, algorithms, goodness-of-fit divergence tests, power divergence statistics, chi-squared tests, power comparisons
AMS Subject Classification: 62G10, 62Q05

1. INTRODUCTION

The problems of goodness of fit to a distribution on the real line, $H_0 : F = F_0$, are frequently treated by partitioning the range of data in disjoint intervals and by testing the hypothesis $H_0 : \boldsymbol{p} = \boldsymbol{p}^0$ about the vector of parameters of a multinomial distribution.

Let $\{A_i\}_{i=1,\dots,m}$ be a partition of the real line R into m intervals. Let $\boldsymbol{p} = (p_1, \dots, p_m)$ and $\boldsymbol{p}^0 = (p_1^0, \dots, p_m^0)$ be the true and the hypothetical probabilities of the intervals A_i, $i = 1, \dots, m$; in such a way that $p_i = F(A_i)$ and $p_i^0 = F_0(A_i)$. Let $Y_1, \dots, Y_n$ be a random sample from F and let $N_i = N_i(Y_1, \dots, Y_n) = \sum_{j=1}^n I_{A_i}(Y_j)$ and $\widehat{p}_i = N_i/n$, $i = 1, \dots, m$, be the absolute and relative frequencies of the intervals.

Cressie and Read [3] (see also Read and Cressie [8]) proposed to test $H_0 : \boldsymbol{p} = \boldsymbol{p}^0$ with the power divergence statistics

$$T_{n,m}^{\lambda}(\widehat{\boldsymbol{p}}, \boldsymbol{p}) = = \frac{2n}{\lambda(\lambda+1)} \sum_{i=1}^m \widehat{p}_i \left[\left(\frac{\widehat{p}_i}{p_i} \right)^{\lambda} - 1 \right] = \frac{2}{\lambda(\lambda+1)} \sum_{i=1}^m N_i \left[\left(\frac{N_i}{np_i} \right)^{\lambda} - 1 \right], \qquad (1.1)$$

where $-\infty < \lambda < \infty$, and they recommended $\lambda = 2/3$. In this paper we are mainly interested in $\lambda = -2, -1, -1/2, 0, 2/3, 1$, i. e.

1. $\lambda = -2$ (Neyman's modified test)

$$T_{n,m}^{-2}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = \sum_{i=1}^{m} \frac{(np_i^0 - N_i)^2}{N_i} = n \sum_{i=1}^{m} \frac{(p_i^0 - \widehat{p}_i)^2}{\widehat{p}_i}.$$

2. $\lambda = -1$ $(\lambda \to -1)$ (Loglikelihood ratio modified test)

$$T_{n,m}^{-1}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = 2 \sum_{i=1}^{m} N_i \ln\left(\frac{np_i^0}{N_i}\right) = 2n \sum_{i=1}^{m} p_i^0 \ln\left(\frac{p_i^0}{\widehat{p}_i}\right).$$

3. $\lambda = -\frac{1}{2}$ (Freeman–Tukey's test)

$$T_{n,m}^{-1/2}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = 8n\left(1 - \sum_{i=1}^{m} \sqrt{\frac{p_i^0 N_i}{n}}\right) = 8n\left(1 - \sum_{i=1}^{m} \sqrt{p_i^0 \widehat{p}_i}\right).$$

4. $\lambda = 0$ $(\lambda \to 0)$ (Loglikelihood ratio test)

$$T_{n,m}^{0}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = 2 \sum_{i=1}^{m} N_i \ln\left(\frac{N_i}{np_i^0}\right) = 2n \sum_{i=1}^{m} \widehat{p}_i \ln\left(\frac{\widehat{p}_i}{p_i^0}\right).$$

5. $\lambda = \frac{2}{3}$ (Cressie–Read's test)

$$T_{n,m}^{2/3}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = \frac{9}{5} n \left(-1 + \sum_{i=1}^{m} \widehat{p}_i \left(\frac{\widehat{p}_i}{p_i^0}\right)^{2/3}\right).$$

6. $\lambda = 1$ (Pearson's χ^2 test)

$$T_{n,m}^{1}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = \sum_{i=1}^{m} \frac{(N_i - np_i^0)^2}{np_i^0} = n \sum_{i=1}^{m} \frac{(\widehat{p}_i - p_i^0)^2}{p_i^0}.$$

The continuity criterion is used when $\lambda = -1, -2, 0$, $0 < p_i^0 < 1$ and $\widehat{p}_i = 0$, i.e. the limits $\widehat{p}_i \to 0$ are taken to obtain the following rules:

1. If $\lambda = -2$, then $\frac{(p_i^0 - \hat{p}_i)^2}{\hat{p}_i}$ is substituted by $\lim_{x \to 0^+} \frac{(p_i^0 - x)^2}{x} = +\infty$.

2. If $\lambda = -1$ $(\lambda \to -1)$, then $p_i^0 \ln \frac{p_i^0}{\hat{p}_i}$ is substituted by $\lim_{x \to 0^+} \ln \frac{p_i^0}{x} = +\infty$.

3. If $\lambda = 0$ $(\lambda \to 0)$, then $\widehat{p}_i \ln \frac{\hat{p}_i}{p_i^0}$ is substituted by $\lim_{x \to 0^+} x \ln x = 0$.

A more general family of statistics, containing (1.1) as a particular case, is

$$T_{n,m}^{\phi}(\widehat{\boldsymbol{p}}, \boldsymbol{p}) = \frac{2n}{\phi''(1)} \sum_{i=1}^{m} \widehat{p}_i \phi\left(\frac{\widehat{p}_i}{p_i}\right), \qquad (1.2)$$

where ϕ is a real convex function defined on $[0,\infty)$, twice continuously differentiable in a neighborhood of $u=1$, satisfying $\phi(1)=\phi'(1)=0$, $\phi''(1)>0$, $0\phi(0/0)=0$ and $0\phi(u/0)=\lim_{u\to\infty}\frac{\phi(u)}{u}$. Divergences appearing in (1.2) have been introduced by Csiszár [4] and Ali and Silvey [2] and extensively studied by Liese and Vajda [6].

Cressie and Read [3] proved that $T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) \underset{n\to\infty}{\longrightarrow} \chi^2_{m-1}$ (in law) under $H_0: \boldsymbol{p}=\boldsymbol{p}^0$ for any $\lambda\in\mathbf{R}$. Zografos et al [10] proved that $T^{\phi}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) \underset{n\to\infty}{\longrightarrow} \chi^2_{m-1}$ (in law) under $H_0: \boldsymbol{p}=\boldsymbol{p}^0$ for any ϕ verifying the above cited properties. Therefore if sample sizes are large enough one can use the asymptotic quantile $\chi^2_{m-1,1-\alpha}$, defined by the equation $P(\chi^2_{m-1}\le\chi^2_{m-1,1-\alpha})=1-\alpha$, to establish the decision rule: "reject H_0 if $T^{\phi}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0)>\chi^2_{m-1,1-\alpha}$". However, this approximation is not justified for those values of m and n for which there are algorithms to calculate efficiently the p-value $P_{\boldsymbol{p}^0}\left(T^{\phi}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0)>t\right)$ for any observed t of $T^{\phi}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0)$.

In this paper we introduce several algorithms to generate all the vectors in the support of a multinomial distribution. We compare the algorithms to the one proposed by Kulmann [5], we make computational studies to analyze their efficiency with respect to the CPU time and to the number of generated vectors and we define and calculate efficiency frontiers. To obtain exact distributions of tests, we restrict ourselves to power divergence statistics $T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0)$ in the equiprobable case $\boldsymbol{p}^0=(1/m,\ldots,1/m)$. In the Appendix, we give the critical values $t^{\lambda}_{n,m,1-\alpha}$ for the first kind error $\alpha=0.05$, $m=4,6,8$, $n=1,\ldots,50$ and $\lambda=-1/2,0,2/3,1$. We make several exact power comparisons for different power divergence statistics and families of alternatives to the uniformity hypothesis. Finally, some recommendations about power divergence test statistics are given.

2. ALGORITHMS TO GENERATE THE VECTORS IN THE SUPPORT OF A MULTINOMIAL DISTRIBUTION

In this section we propose an algorithm to generate the set of vectors

$$A^n_m=\{\boldsymbol{x}_m=(x_1,\ldots,x_m)\in[N\cup\{0\}]^m \,/\, x_1+\ldots+x_m=n,\ n\in N\},$$

with cardinal (number of elements in the set)

$$Card\,(A^n_m)=CR^n_m=\frac{(m+n-1)(m+n-2)\cdots m}{n!}\,.$$

The proposed algorithm is compared with two recursive algorithms that generate supersets of A^n_m and a recursive algorithm that generates the set A^n_m. The first two algorithms follow the *backtracking* and *branch-and-bound* design techniques respectively. The last algorithm is implemented by making a slight modification to the second algorithm.

The backtracking algorithm generates the set

$$A^n_{m,\text{backtracking}}=\{\boldsymbol{x}_m=(x_1,\ldots,x_m)\in[\{0,\ldots,n\}]^m, n\in N\}$$

with $Card(A^n_{m,\text{backtracking}}) = VR^m_{n+1} = (n+1)^m$. The branch-and-bound algorithm generates the set

$$A^n_{m,\text{branch-and-bound}} = \{x_m = (x_1, \ldots, x_m) \in [N \cup \{0\}]^m \,/\, x_1 + \ldots + x_m \leq n, n \in N\}$$

with $Card(A^n_{m,\text{branch-and-bound}}) = CR^n_{m+1}$.

These two algorithms work similarly. The backtracking algorithm generates recursively the vectors x_m, with components $x_i \in \{0, 1, \ldots, n\}$. This algorithm starts with the generation of $n+1$ vectors by assigning to their first component ($i = 1$) the values $n, n-1, \ldots, 1, 0$, respectively. For each of the $n+1$ vectors generated at step 1, the algorithm generates $n+1$ new vectors and assigns to the second component ($i = 2$) the values $n, n-1, \ldots, 1, 0$. This process stops at step m, i.e. when the m components of all the generated vectors are assigned.

The branch-and-bound algorithm assigns to each x_i a value in $\{0, 1, \ldots, r\}$, where r is the difference between n and the sum of the values of the already assigned components, i.e. $r = n - \sum_{j<i} x_j$. The algorithm starts with the generation of $n+1$ vectors by assigning to their first component ($i = 1$) the values $n, n-1, \ldots, 0$. At the second step, the algorithm calculates r for each of the $n+1$ generated vectors and generates new vectors by assigning to their second component ($i = 2$) the values $r, r-1, \ldots, 0$. The process of generating a vector stops when all its components are assigned or when the sum of its assigned components is equal to n. In the last case, the remaining components of the vector are assigned to 0. The algorithm ends when the m components of all the generated vectors have been assigned.

Note that if we modified the branch-and-bound algorithm by only assigning one value, that is r, to the last component of the vector ($i = m$), we obtain a recursive algorithm which generates the set A^n_m. This algorithm is called efficient branch-and-bound algorithm.

Example. Let $m = 3$ and $n = 4$. At the beginning $r = 4$ and the algorithm generates the vectors $(4, ,), (3, ,), (2, ,), (1, ,), (0, ,)$. For each of the vectors with assigned components not summing up to 4, the algorithm calculates r and assigns values from r to 0 to the component $i = 2$. If $r = 0$, the algorithm assigns 0 to the remaining components.

$(4, ,) \to r = 4 - 4 = 0 \to$ **(4,0,0)**
$(3, ,) \to r = 4 - 3 = 1 \to (3,1,), (3,0,)$
$(2, ,) \to r = 4 - 2 = 2 \to (2,2,), (2,1,), (2,0,)$
$(1, ,) \to r = 4 - 1 = 3 \to (1,3,), (1,2,), (1,1,), (1,0,)$
$(0, ,) \to r = 4 - 0 = 4 \to (0,4,), (0,3,), (0,2,), (0,1,), (0,0,)$.

The process is repeated for $i = 3$. In this case only one value, that is r, is assigned to the actual component since the algorithm is in the last position of the vector ($i = m$).

(3,1,) $\to r = 4-(3+1) = 0 \to$ **(3,1,0)**	(3,0,) $\to r = 4-(3+0) = 1 \to$ **(3,0,1)**
(2,2,) $\to r = 4-(2+2) = 0 \to$ **(2,2,0)**	(2,1,) $\to r = 4-(2+1) = 1 \to$ **(2,1,1)**
(2,0,) $\to r = 4-(2+0) = 2 \to$ **(2,0,2)**	(1,3,) $\to r = 4-(1+3) = 0 \to$ **(1,3,0)**
(1,2,) $\to r = 4-(1+2) = 1 \to$ **(1,2,1)**	(1,1,) $\to r = 4-(1+1) = 2 \to$ **(1,1,2)**
(1,0,) $\to r = 4-(1+0) = 3 \to$ **(1,0,3)**	(0,4,) $\to r = 4-(0+4) = 0 \to$ **(0,4,0)**
(0,3,) $\to r = 4-(0+3) = 1 \to$ **(0,3,1)**	(0,2,) $\to r = 4-(0+2) = 2 \to$ **(0,2,2)**
(0,1,) $\to r = 4-(0+1) = 3 \to$ **(0,1,3)**	(0,0,) $\to r = 4-(0+0) = 4 \to$ **(0,0,4)**.

Finally, we implement an iterative algorithm to generate the set A_m^n. Before describing this algorithm, we introduce several concepts in order to define a total order relationship on the set A_m^n. This is done by means of functions *next* and *previous*, which generate the elements of A_m^n in an ordered way. Proofs of results presented below are straightforward and can be found in Marhuenda et al [7].

Definition 1. Let $\boldsymbol{x}_m$ and $\boldsymbol{y}_m$ be two elements in A_m^n, then

(a) $\boldsymbol{x}_m = \boldsymbol{y}_m \iff x_i = y_i \quad \forall i = 1,2,\ldots,m.$

(b) $\boldsymbol{x}_m \neq \boldsymbol{y}_m \iff \exists i \in \{1,2,\ldots,m\}$ such that $x_i \neq y_i$.

(c) $\boldsymbol{x}_m > \boldsymbol{y}_m \iff \exists i \in \{1,2,\ldots,m\}$ with $x_i > y_i$ and $x_j \geq y_j \; \forall j \in \{1,2,\ldots,i-1\}$.

(d) $\boldsymbol{x}_m \geq \boldsymbol{y}_m \iff \boldsymbol{x}_m > \boldsymbol{y}_m$ or $\boldsymbol{x}_m = \boldsymbol{y}_m$.

(e) $\boldsymbol{x}_m < \boldsymbol{y}_m \iff \exists i \in \{1,2,\ldots,m\}$ with $x_i < y_i$ and $x_j \leq y_j$, $\forall j \in \{1,2,\ldots,i-1\}$.

(f) $\boldsymbol{x}_m \leq \boldsymbol{y}_m \iff \boldsymbol{x}_m < \boldsymbol{y}_m$ or $\boldsymbol{x}_m = \boldsymbol{y}_m$.

Note that $\boldsymbol{x}_m < \boldsymbol{y}_m$ holds when $\boldsymbol{x}_m \geq \boldsymbol{y}_m$ does not hold and vice versa.

Proposition 1. The relation $\geq$ is a good order in A_m^n, i.e. the reflexive, antisymmetric and transitive properties hold, and also

1. $\forall \boldsymbol{x}_m, \boldsymbol{y}_m \in A_m^n$, $\boldsymbol{x}_m \geq \boldsymbol{y}_m$ or $\boldsymbol{y}_m \geq \boldsymbol{x}_m$.

2. $\forall B_m^n \subset A_m^n$, $B_m^n \neq \emptyset$, $\exists \boldsymbol{x}_m \in B_m^n$ such that $\boldsymbol{y}_m \geq \boldsymbol{x}_m \; \forall \boldsymbol{y}_m \in B_m^n$.

The relation $\leq$ is also a good order in A_m^n.

Definition 2. The *first element*, $\boldsymbol{p}_m$, of A_m^n is $\boldsymbol{p}_m = (p_1,\ldots,p_m)$, where $p_1 = \ldots = p_{m-1} = 0$, $p_m = n$.

Definition 3. The *last element*, $\boldsymbol{u}_m$, of A_m^n is $\boldsymbol{u}_m = (u_1,\ldots,u_m)$, where $u_1 = n$, $u_2 = \ldots = u_m = 0$.

Corollary 1. The following statements hold.

1. Let $\boldsymbol{p}_m$ be the first element of A_m^n. If $\boldsymbol{x}_m \in A_m^n$ is such that $\boldsymbol{x}_m \neq \boldsymbol{p}_m$, then $\boldsymbol{p}_m < \boldsymbol{x}_m$.
2. Let $\boldsymbol{u}_m$ be the last element of A_m^n. If $\boldsymbol{x}_m \in A_m^n$ is such that $\boldsymbol{x}_m \neq \boldsymbol{u}_m$, then $\boldsymbol{u}_m > \boldsymbol{x}_m$.
3. The minimum element of the relation $\leq$ is $\boldsymbol{p}_m$.
4. The maximum element of the relation $\leq$ is $\boldsymbol{u}_m$.

Definition 4. (*next* function) Let $\boldsymbol{x}_m = (x_1, \ldots, x_m) \in A_m^n$ such that $x_i \neq 0$ for some $i \in \{1, \ldots, m\}$ and $x_j = 0 \ \forall j \in \{i+1, \ldots, m\}$. Suppose that $\boldsymbol{x}_m \neq \boldsymbol{u}_m$ (last element). We distinguish the following two cases in order to define $\boldsymbol{y}_m = next(\boldsymbol{x}_m) = (y_1, \ldots, y_m)$:

1. If $i < m$, then

$$y_k = \begin{cases} x_k & \text{if} \quad 1 \leq k \leq i-2 \\ x_{i-1}+1 & \text{if} \quad k = i-1 \\ 0 & \text{if} \quad i \leq k \leq m-1 \\ x_i - 1 & \text{if} \quad k = m. \end{cases}$$

2. If $i = m$, then

$$y_k = \begin{cases} x_k & \text{if} \quad 1 \leq k \leq m-2 \\ x_{m-1}+1 & \text{if} \quad k = m-1 \\ x_m - 1 & \text{if} \quad k = m. \end{cases}$$

Definition 5. (*previous* function) Let $\boldsymbol{x}_m = (x_1, \ldots, x_m) \in A_m^n$ such that $x_i \neq 0$ for some $i \in \{1, \ldots, m\}$ and $x_j = 0 \ \forall j \in \{i+1, \ldots, m\}$. Suppose that $\boldsymbol{x}_m \neq \boldsymbol{p}_m$ (first element). We distinguish several cases in order to define the components of the previous element of $\boldsymbol{x}_m$, $\boldsymbol{y}_m = previous(\boldsymbol{x}_m) = (y_1, \ldots, y_m)$:

1. If $i < m-1$, then

$$y_k = \begin{cases} x_k & \text{if} \quad 1 \leq k \leq i-1 \\ x_i - 1 & \text{if} \quad k = i \\ 1 & \text{if} \quad k = i+1 \\ 0 & \text{if} \quad i+2 \leq k \leq m. \end{cases}$$

2. If $i = m-1$, then

$$y_k = \begin{cases} x_k & \text{if} \quad 1 \leq k \leq m-2 \\ x_{m-1} - 1 & \text{if} \quad k = m-1 \\ 1 & \text{if} \quad k = m. \end{cases}$$

3. If $i = m$, $\boldsymbol{x}_m \neq \boldsymbol{p}_m$, then $\exists j \in \{1, \ldots, m-1\}$ such that $x_j \neq 0$ and $x_\ell = 0 \ \forall \ell \in \{j+1, \ldots, m-1\}$. We consider two cases

(a) If $j < m - 1$, then

$$y_k = \begin{cases} x_k & \text{if } 1 \le k \le j-1 \\ x_j - 1 & \text{if } k = j \\ x_m + 1 & \text{if } k = j+1 \\ 0 & \text{if } j+2 \le k \le m. \end{cases}$$

(b) If $j = m - 1$, then

$$y_k = \begin{cases} x_k & \text{if } 1 \le k \le m-2 \\ x_{m-1} - 1 & \text{if } k = m-1 \\ x_m + 1 & \text{if } k = m. \end{cases}$$

Corollary 2. The following statements hold.

1. If $\boldsymbol{x}_m \in A^n_m$, $\boldsymbol{x}_m \neq \boldsymbol{u}_m$ and $\boldsymbol{y}_m = \mathit{next}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m \in A^n_m$.

2. If $\boldsymbol{x}_m \in A^n_m$, $\boldsymbol{x}_m \neq \boldsymbol{p}_m$ and $\boldsymbol{y}_m = \mathit{previous}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m \in A^n_m$.

3. If $\boldsymbol{y}_m = \mathit{next}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m > \boldsymbol{x}_m$.

4. If $\boldsymbol{y}_m = \mathit{previous}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m < \boldsymbol{x}_m$.

5. Let $\boldsymbol{x}_m, \boldsymbol{y}_m \in A^n_m$, such that $\boldsymbol{y}_m = \mathit{next}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m$ is the immediate successor of $\boldsymbol{x}_m$, that is, $\boldsymbol{y}_m > \boldsymbol{x}_m$ and $\nexists \boldsymbol{z}_m \in A^n_m$ such that $\boldsymbol{y}_m > \boldsymbol{z}_m$ and $\boldsymbol{z}_m > \boldsymbol{x}_m$.

6. Let $\boldsymbol{x}_m, \boldsymbol{y}_m \in A^n_m$, such that $\boldsymbol{y}_m = \mathit{previous}(\boldsymbol{x}_m)$, then $\boldsymbol{y}_m$ is the immediate predecessor of $\boldsymbol{x}_m$, that is, $\boldsymbol{y}_m < \boldsymbol{x}_m$ and $\nexists \boldsymbol{z}_m \in A^n_m$ such that $\boldsymbol{y}_m < \boldsymbol{z}_m$ and $\boldsymbol{z}_m < \boldsymbol{x}_m$.

7. Let $\boldsymbol{x}_m, \boldsymbol{y}_m \in A^n_m$, then $\boldsymbol{y}_m = \mathit{next}(\boldsymbol{x}_m) \Longleftrightarrow \boldsymbol{x}_m = \mathit{previous}(\boldsymbol{y}_m)$.

We now describe the iterative algorithm. This algorithm starts with the first element $\boldsymbol{p}_m$ of A^n_m and generates the remaining elements in A^n_m by applying the *next* function to the last generated element. This process continues until the last element $\boldsymbol{u}_m$ is generated.

Algorithm can also be applied in a descending order. In this case, the algorithm begins with the last element $\boldsymbol{u}_m$, it applies the *previous* function to the last generated element and stops when the first element $\boldsymbol{p}_m$ is generated.

Example. Let $m = 3$ and $n = 4$. We use the *next* function to generate the set A^4_3. We begin with the first element $\boldsymbol{p}_m = (0, 0, 4)$ and apply the *next* function to the last generated element. The process ends when this function generates the last element $\boldsymbol{u}_m = (4, 0, 0)$.

$$p_m = (0, 0, 4)$$

$next((0,0,4)) = (0,1,3)$	$next((1,2,1)) = (1,3,0)$
$next((0,1,3)) = (0,2,2)$	$next((1,3,0)) = (2,0,2)$
$next((0,2,2)) = (0,3,1)$	$next((2,0,2)) = (2,1,1)$
$next((0,3,1)) = (0,4,0)$	$next((2,1,1)) = (2,2,0)$
$next((0,4,0)) = (1,0,3)$	$next((2,2,0)) = (3,0,1)$
$next((1,0,3)) = (1,1,2)$	$next((3,0,1)) = (3,1,0)$
$next((1,1,2)) = (1,2,1)$	$next((3,1,0)) = (4,0,0) = u_m.$

In Figure 1 the flow diagrams corresponding to the iterative algorithm using the ascending and descending order are presented. Algorithms have been written in standard C and can be found in Marhuenda et al [7].

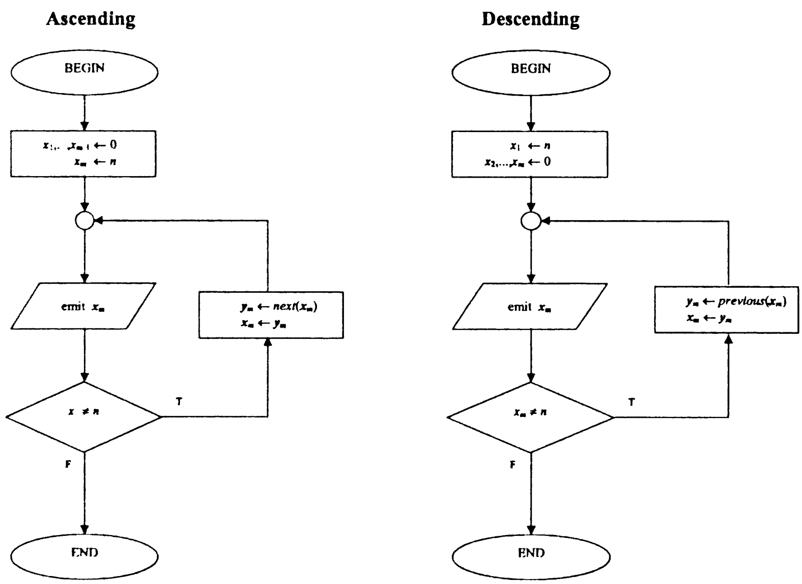

Fig. 1. Flow diagrams corresponding to the iterative algorithm with ascending and descending order.

3. COMPARISONS BETWEEN ALGORITHMS

In this section, we analyze the efficiency of the algorithms described in the previous section in relation to the CPU time that each algorithm uses to generate the set A_m^n.

We calculate the efficiency frontier for the iterative algorithm.

The four algorithms have been implemented in C and run on a Pentium II 350MHz biprocessor workstation with 512MB RAM, under the LINUX operating system.

The CPU time depends on many factors, such as, the programming language, the compilation options and the hardware. Due to the fact that LINUX uses multitasking and supports multiple users, the CPU time is the sum of the user and system times which have been obtained by using the *time* command. In addition, the algorithms have been run 25 times for each m and n and the average of CPU time calculated. Figure 2 shows the results for $m = 5, 6$ and $n = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$. The CPU time values obtained for the backtracking and branch-and-bound algorithms are not represented because they are greater than the values obtained for the others algorithms. For instance, the CPU times obtained for $n = 30$, $m = 5$ are 4.04 and 0.08 seconds, respectively.

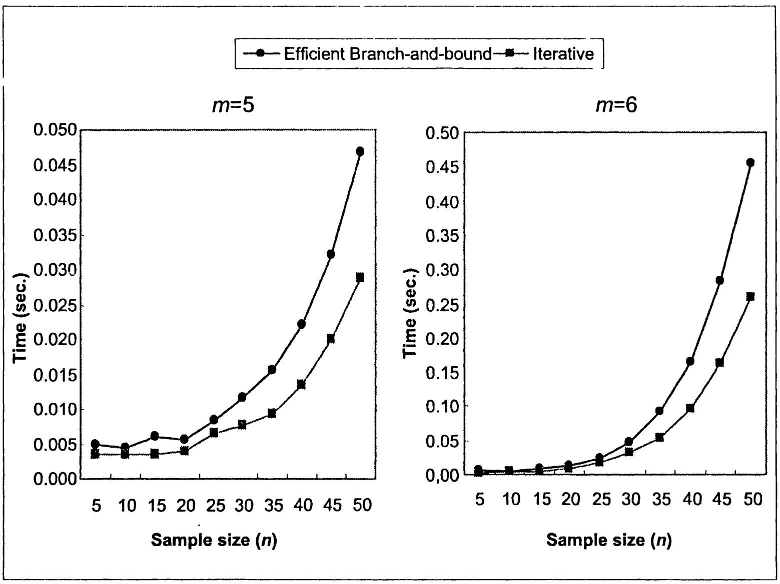

Fig. 2. CPU time for the efficient branch-and-bound and iterative algorithms for $m = 5, 6$ cells.

Let $t_{m,n}$ be the CPU time that a given algorithm uses to generate all the elements in A_m^n. At a level of t_0 seconds, its *efficiency frontier* is defined by the set $\{(m, n_{m,t_0}) : m = 2, 3, \ldots\}$, where

$$n_{m,t_0} = \max\{n \in N : t_{m,n} \leq t_0\}.$$

In Figure 3 the efficiency frontier of the iterative algorithm for 1 second of CPU time

($t_0 = 1$) and the hardware and operating system described above is presented.

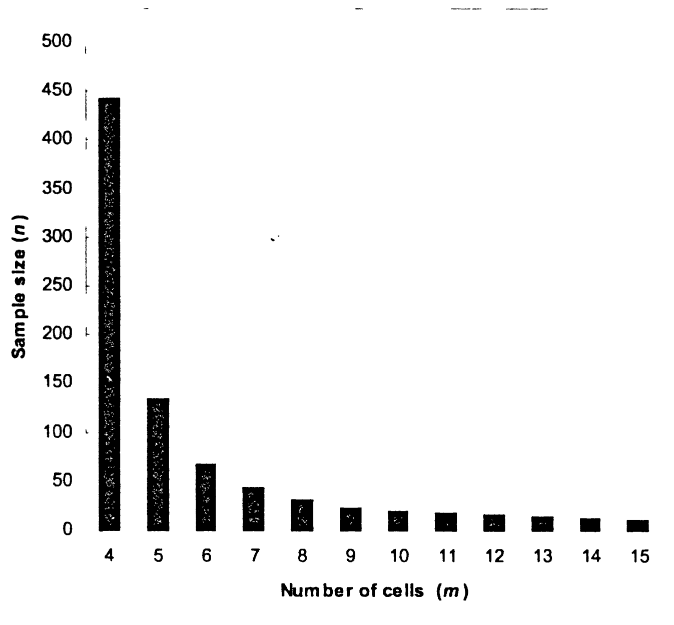

Fig. 3. Efficiency frontier of the iterative algorithm for 1 second of CPU time.

It is interesting to observe that for $m_1 < m_2$

$$0 < t_{m_1,n} - t_{m_1,n-1} < t_{m_2,n} - t_{m_2,n-1}.$$

For instance, the CPU time difference between ($m = 14, n = 10$) and ($m = 14, n = 11$) is 0.183, whereas between ($m = 15, n = 10$) and ($m = 15, n = 11$) is 0.350 seconds.

4. UNFORMITY TESTS WITH EXACT DISTRIBUTIONS

In this section, power divergence test statistics $T^{\lambda}_{n,m}(\widehat{p}, p^0)$ are used to test the hypothesis $H_0 : p = p^0$. Exact quantiles $t^{\lambda}_{n,m,1-\alpha}$ are calculated for the probability of first kind error $\alpha = 0.05$ and for $\lambda = -2, -1, -1/2, 0, 2/3, 1$. The continuity criterion is used when $\lambda = -1, -2, 0$ and $\widehat{p}_i = 0$, i. e. we take limits $\widehat{p}_i \to 0$ in order to evaluate the test statistics.

The distribution function of $T^{\lambda}_{n,m} = T^{\lambda}_{n,m}(\widehat{p}, p^0)$ under the null hypothesis $H_0 : p = p^0$ is

$$F_{T^{\lambda}_{n,m}(\widehat{p},p^0)}(t) = P_{p^0}\left(T^{\lambda}_{n,m}(\widehat{p}, p^0) \le t\right) = 1 - P_{p^0}\left(T^{\lambda}_{n,m}(\widehat{p}, p^0) > t\right),$$

where

$$P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) > t\right) = \sum_{(x_1,\ldots,x_m)\in A^n_{m,t}} P_{\boldsymbol{p}^0}\left(N_1 = x_1,\ldots,N_m = x_m\right),$$

$$A^n_{m,t} = \left\{(x_1,\ldots,x_m) \in [N\cup\{0\}]^m / x_1+\ldots+x_m = n, T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) > t\right\}$$

and

$$P_{\boldsymbol{p}^0}\left(N_1 = x_1,\ldots,N_m = x_m\right) = \frac{n!}{x_1!\ldots x_m!}(p^0_1)^{x_1}\ldots(p^0_m)^{x_m}.$$

The set of upper tail probabilities of $T^{\lambda}_{n,m}$ is

$$\mathcal{U}^{\lambda}_{n,m} = \left\{\alpha\in(0,1) : \exists t>0 \text{ with } P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) > t\right) = \alpha\right\}.$$

Quantiles $t^{\lambda}_{n,m,1-\alpha}$ of $T^{\lambda}_{n,m}$ are obtained for any $\alpha\in\mathcal{U}^{\lambda}_{n,m}$ through the equation

$$\alpha = P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) > t^{\lambda}_{n,m,1-\alpha}\right).$$

If $\alpha\in(0,1)-\mathcal{U}^{\lambda}_{n,m}$, we consider

$$\alpha_1 = \alpha(n,m,\lambda,\alpha) = \max\left\{\alpha_0\in(0,\alpha] : \exists t>0 \text{ with } P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}},\boldsymbol{p}^0) > t\right) = \alpha_0\right\},$$

so that $t^{\lambda}_{n,m,1-\alpha_1}$ is defined as the approximate quantile of order α. We calculate the approximate quantiles for $\alpha = 0.05$, $m = 2,\ldots,10$, $n = 1,\ldots,50$ and the above specified λ. This process can be divided into four steps:

Step 1. Generate all the elements $\boldsymbol{x}_m = (x_1,\ldots,x_m)$ of A^n_m by using the iterative algorithm and calculate the corresponding probabilities $P_{\boldsymbol{p}^0}(x_1,\ldots,x_m)$.

Step 2. For each $\boldsymbol{x}_m\in A^n_m$, calculate the test statistics $T^{\lambda}_{n,m}$ with the special considerations for $\lambda = -2,-1,0$ and $\widehat{p}_i = 0$.

Step 3. Put $T^{\lambda}_{n,m}$ and $P_{\boldsymbol{p}^0}(x_1,\ldots,x_m)$ in increasing order with respect to the values of $T^{\lambda}_{n,m}$.

We have used internal and external classification in this step. In the internal classification the ordination takes place in the main memory of the computer, where it is possible to use random access to the data. In this case, the values of the test statistic and the probability of each $\boldsymbol{x}_m$ are stored in the main memory. We have implemented the quicksort algorithm specified in Aho, Hopcroft and Ullman [1] to order the data. This algorithm is recursive and has a complexity in the average case of $O(k\log_2 k)$, where $k = Card(A^n_m)$.

The external classification is used when there is not enough main memory available to store the data and secondary storage devices are needed. We have implemented the files intercalation algorithm specified in Aho, Hopcroft and Ullman [1]. This algorithm needs $\lceil\log_2(k/\ell)\rceil$ repetitions, where $k = Card(A^n_m)$ is the number of elements to be ordered and ℓ is the initial size of an ordered block of data which depends on the computer main memory capacity. The complexity of the algorithms in the better, worse and average cases has been investigated by Aho, Hopcroft and Ullman [1] and Weiss [9].

Step 4. Calculate the approximate quantile $t^{\lambda}_{n,m,1-\alpha}$ of order $\alpha = 0.05$.

We use randomized tests in order to decide with probability $\gamma^{\lambda}_{n,m,\alpha}$ the rejection of the hypothesis H_0 when the test statistic takes the value $t^{\lambda}_{n,m,1-\alpha_1}$. Let $\phi(T^{\lambda}_{n,m})$ be a function giving the probability of rejecting H_0 when $T^{\lambda}_{n,m}$ is observed. This function is defined by the formula

$$\phi(T^{\lambda}_{n,m}) = \begin{cases} 1 & \text{if } \; T^{\lambda}_{n,m} > t^{\lambda}_{n,m,1-\alpha_1} \\ \gamma^{\lambda}_{n,m,\alpha} & \text{if } \; T^{\lambda}_{n,m} = t^{\lambda}_{n,m,1-\alpha_1} \\ 0 & \text{if } \; T^{\lambda}_{n,m} < t^{\lambda}_{n,m,1-\alpha_1} \end{cases} \tag{4.1}$$

$$\alpha = E_{\boldsymbol{p}^0}\left(\phi(T^{\lambda}_{n,m})\right) = 1 \cdot P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m} > t^{\lambda}_{n,m,1-\alpha_1}\right) + \gamma^{\lambda}_{n,m,\alpha} \cdot P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m} = t^{\lambda}_{n,m,1-\alpha_1}\right)$$

$$\gamma^{\lambda}_{n,m,\alpha} = \frac{\alpha - P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m} > t^{\lambda}_{n,m,1-\alpha_1}\right)}{P_{\boldsymbol{p}^0}\left(T^{\lambda}_{n,m} = t^{\lambda}_{n,m,1-\alpha_1}\right)}.$$

Using the previous process, approximate quantiles $t^{\lambda}_{n,m,1-\alpha_1}$ and probabilities $\gamma^{\lambda}_{n,m,\alpha}$ are calculated for the uniform distribution $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$, with $\alpha = 0.05$, $n = 1, \ldots, 50$, $m = 2, \ldots, 10$ and $\lambda = -2, -1, -1/2, 0, 2/3, 1$. In Tables 1–4 of the Appendix, computed values for the Freeman–Tukey ($\lambda = -1/2$), loglikelihood ($\lambda = 0$), Cressie–Read ($\lambda = 2/3$) and Pearson's χ^2 ($\lambda = 1$) test statistics and $m = 4, 6, 8$ are presented. The rest of the computed values can be found in Marhuenda et al [7].

In addition to the memory limitations in Step 3, there are limitations related to the maximum size of a file. The operating systems that we have used, SUSE Linux 6.0 with kernel 2.2.7 and Linux Mandrake with kernel 2.2.13.7, allow a maximum size of 2GB (2,147,483,648 bytes) for a file. This value is insufficient to store all the values of the test statistics and probabilities calculated for each $\boldsymbol{x}_m \in A^n_m$ when m and n are large. For example, for $m = 10, n = 30$, the number of elements of the set A^n_m, $Card(A^n_m)$, is 211,915,132. The implemented program stores the value of the test statistic as a float data type with 4 bytes and the probability as a double data type with 8 bytes, so we would need an ordered file of $211,915,132 \times (4+8) = 2,542,981,584$ bytes > 2GB. For that reason, Steps 1–3 have been slightly modified. If $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$, the function $g(\widehat{\boldsymbol{p}}) = T^{\lambda}_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0)$ is not one to one, i. e. there are sets $\{\widehat{\boldsymbol{p}}_1, \ldots, \widehat{\boldsymbol{p}}_s\}$ of probability vectors such that $g(\widehat{\boldsymbol{p}}_1) = \ldots = g(\widehat{\boldsymbol{p}}_s)$. In this case, we only store the values of the test statistics which are different, and their corresponding total probabilities.

Although we have calculated the quantiles $t^{\lambda}_{n,m,1-\alpha}$ and the probabilities $\gamma^{\lambda}_{n,m,\alpha}$ for the equiprobable distribution, the program is able to calculate quantiles and probabilities for nonequiprobable distributions since the whole set A^n_m is generated. This fact is relevant when calculating exact powers in Section 5. The algorithm introduced by Kulmann [5] only calculates the different partitions of a number n in a vector of m positive natural numbers so that their sum equals to n and considers that two partitions are equal if they differ only in the order of the numbers.

This assumption reduces significantly the operations, but it can be only applied to equiprobable distributions.

5. EXACT POWERS OF TESTS

Let $\boldsymbol{p} = (p_1, \ldots, p_m)$ be a probability vector. The exact power function of test $\phi(T_{n,m}^{\lambda})$, defined in (4.1), is

$$\beta_{n,m}^{\lambda}(\boldsymbol{p}) = E_{\boldsymbol{p}}\left(\phi(T_{n,m}^{\lambda})\right) = 1 \cdot P_{\boldsymbol{p}}\left(T_{n,m}^{\lambda} > t_{n,m,1-\alpha_1}^{\lambda}\right) + \gamma_{n,m,\alpha}^{\lambda} \cdot P_{\boldsymbol{p}}\left(T_{n,m}^{\lambda} = t_{n,m,1-\alpha_1}^{\lambda}\right).$$

In this section, we calculate the exact powers of the tests (4.1) and the inefficiencies for different families of alternatives to the uniformity hypothesis H_0 : $\boldsymbol{p} = \boldsymbol{p}^0$, with $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$. The power divergence statistics for $\lambda = -2, -1, -1/2, 0, 2/3, 1, 2$, are considered for $m = 6$, $n = 30, 42$, $\alpha = 0.05$ and five families of alternatives.

The first family is

$$p_i^{1,\delta} = \begin{cases} \frac{m-1-\delta}{m(m-1)} & \text{if} \quad i = 1, \ldots, m-1 \\ \frac{1+\delta}{m} & \text{if} \quad i = m, \end{cases} \tag{5.1}$$

where $-1 \leq \delta \leq m-1$. Probability vectors $\boldsymbol{p}^{1,\delta}$ of this family are calculated by adding $\frac{\delta}{m}$ to $p_m^0 = \frac{1}{m}$, while the rest are adjusted so that they still sum to one. The following values of δ are considered: $\delta = -1.00, -0.98, -0.97, -0.95, -0.90, -0.80,$ $-0.60, -0.30,$ 0.00, 0.50, 1.00, 1.50, 2.00, 2.25, 2.50, 2.75, 3.00.

The second family is

$$p_i^{2,\delta} = \begin{cases} \frac{m-2-2\delta}{m(m-2)} & \text{if} \quad i = 1, \ldots, m-2 \\ \frac{1+\delta}{m} & \text{if} \quad i = m-1, m, \end{cases} \tag{5.2}$$

where $-1 \leq \delta \leq \frac{m-2}{2}$. Probability vectors $\boldsymbol{p}^{2,\delta}$ of this family are calculated by adding $\frac{\delta}{m}$ to $p_m^0 = p_{m-1}^0 = \frac{1}{m}$, while the rest are adjusted so that they still sum to one. The following values of δ are considered: $\delta = -1.00, -0.98, -0.97, -0.95, -0.90, -0.80,$ $-0.60, -0.30,$ 0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00.

The third family is

$$p_i^{3,\delta} = \begin{cases} \frac{1}{m} - \frac{2i\delta}{m^2(m-1)} & \text{if} \quad i = 1, \ldots, m-1 \\ \frac{1+\delta}{m} & \text{if} \quad i = m, \end{cases} \tag{5.3}$$

where $-1 \leq \delta \leq m/2$. Probability vectors $\boldsymbol{p}^{3,\delta}$ of this family are calculated by adding $\frac{\delta}{m}$ to $p_m^0 = \frac{1}{m}$ and $a\frac{i}{m}$ to p_i^0, $i = 1, \ldots, m-1$, and calculating a so that they still sum to one. The following values of δ are considered: $\delta = -1.00, -0.98, -0.97, -0.95, -0.90, -0.80,$ $-0.60, -0.30,$ 0.00, 0.50, 1.00, 1.50, 2.00, 2.25, 2.50, 2.75, 3.00.

The fourth family is

$$p_i^{4,\delta} = \begin{cases} \frac{1}{m} - \frac{4i\delta}{m(m-1)(m-2)} & \text{if} \quad i = 1, \dots, m-2 \\ \frac{1+\delta}{m} & \text{if} \quad i = m-1, m, \end{cases} \tag{5.4}$$

where $-1 \leq \delta \leq \frac{m-1}{4}$. Probability vectors $p^{4,\delta}$ of this family are calculated by adding $\frac{\delta}{m}$ to $p_m^0 = p_{m-1}^0 = \frac{1}{m}$ and $a\frac{i}{m}$ to p_i^0, $i = 1, \dots, m-2$, where a is selected so that $\sum_{i=1}^m p_i^{4,\delta} = 1$. The following values of δ are considered: $\delta = -1.00, -0.98, -0.97, -0.95, -0.90, -0.80, -0.60, -0.30, 0.00, 0.25, 0.50, 0.75, 0.90, 1.00, 1.10, 1.20, 1.25$.

The fifth family is

$$p_i^{5,\delta} = \begin{cases} \frac{1}{m} - \frac{2\delta}{m} & \text{if} \quad i = 1, \dots, \frac{m}{2} \\ \frac{1}{m} + \frac{2\delta}{m} & \text{if} \quad i = \frac{m}{2} + 1, \dots, m, \end{cases} \tag{5.5}$$

where $-1/2 \leq \delta \leq 1/2$. Probability vectors $p^{5,\delta}$ of this family are calculated by splitting the set of cells in two and by adding or subtracting $2\delta/m$ to the $p_i^{5,\delta}$'s of first or second subset respectively. Here, $\delta = 0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50$ are the values under consideration.

The maximum power of the family f in the alternative δ is

$$\beta_{\max}(n, m, p^{f,\delta}) = \max_{\lambda} \left\{ \beta_{n,m}^{\lambda}(p^{f,\delta}) \right\}.$$

The inefficiency of the test $T_{n,m}^{\lambda}$ for the family f in the alternative δ is

$$i_{n,m}(p^{f,\delta}, \lambda) = \beta_{\max}(n, m, p^{f,\delta}) - \beta_{n,m}^{\lambda}(p^{f,\delta}).$$

The maximum inefficiency of the test $T_{n,m}^{\lambda}$ for the family f is

$$i_{\max}(n, m, f, \lambda) = \max_{\delta} \left\{ i_{n,m}(p^{f,\delta}, \lambda) \right\}.$$

In Table 1, we present, for the five considered families, the number of times that each statistic can be recommended. These quantities are obtained by counting the three smallest $i_{\max}(n, m, f, \lambda)$ for $m = 6$ and $n = 30, 42$. The intermediate tables with the computed values of the powers and inefficiencies can be found in Marhuenda et al [7].

From Table 1, we can give the following recommendations on which power divergence tests one should use for $f = 1, \dots, 5$, $m = 6$ and $n = 30, 42$:

- $\lambda = -1, -1/2, 0$ for the families (5.1),(5.3),
- $\lambda = 0, 2/3, 1$ for the families (5.2),(5.5),
- $\lambda = -1/2, 0, 2/3, 1$ for the family (5.4),

so that $\lambda = -1/2, 0, 2/3, 1$ are the most frequently recommended values.

In Table 2, we present the sum of inefficiencies $\sum_{n=30,42} \sum_{f=1}^{5} i_{\max}(n, m, f, \lambda)$, for $m = 6$ and each considered λ. Best results are obtained for $\lambda = -1, -1/2, 0, 2/3$. Finally, we observe that power divergence statistics with $\lambda = -1/2, 0, 2/3$ are recommended with both criteria.

Table 1. Number of times that we recommend each λ for $f = 1, \ldots, 5$, $m = 6$ and $n = 30, 42$.

	Family					
λ	(5.1)	(5.2)	(5.3)	(5.4)	(5.5)	Total
-2						0
-1	2		2			4
$-1/2$	2		2	1		5
0	2	2	2	2	2	10
2/3		2		2	2	6
1		2		1	2	5
2						0

Table 2. Sums $\sum_{n=30,42} \sum_{f=1}^{5} i_{\max}(n, m, f, \lambda)$ for $m = 6$ and each λ.

$\lambda = -2$	$\lambda = -1$	$\lambda = -1/2$	$\lambda = 0$	$\lambda = 2/3$	$\lambda = 1$	$\lambda = 2$
1.76186	1.25244	0.93686	0.61648	1.59248	2.12815	3.37238

(Received November 12, 2001.)

APPENDIX

Tables with $t = t^{\lambda}_{n,m,0.95}$, $q = q^{\lambda}_{n,m,t}$, $\gamma = \gamma^{\lambda}_{n,m,0.05}$

Table 1. Freeman–Tukey's test ($\lambda = -1/2$) for $\alpha = 0.05$ and $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$

$$T^{-1/2}_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}) = 8n\left(1 - \sum_{i=1}^{m} \sqrt{p_i \widehat{p}_i}\right), \quad q^{-1/2}_{n,m,t} = P_{\boldsymbol{p}^0}(T^{-1/2}_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) > t).$$

m	4			6			8		
n	t	q	γ	t	q	γ	t	q	γ
1	...	...		...	...		...	...	
2	...	...		...	...		...	...	
3	...	...		10.343145	.027778	.053333	12.172817	.015625	.104762
4	10.143594	.015625	.183333	14.154319	.004630	.490000	16.545187	.001953	.878571
5	13.167184	.003906	.786667	17.022934	.020062	.776000	16.396439	.025879	.235238
6	14.547675	.018555	.715556	16.000000	.020062	.776000	19.273838	.025879	.313651
7	11.169670	.046387	.117460	17.856834	.015775	.760381	18.583426	.049911	.003156
8	11.388070	.033569	.267063	19.396900	.036280	.914476	20.686291	.025378	.585438
9	12.382575	.038055	.258862	16.638773	.039781	.378413	22.544155	.033965	.677815
10	13.768396	.034348	.542751	18.032267	.041989	.854273	23.717913	.046451	.360089
11	13.763933	.040442	.361573	18.866249	.029978	.727918	21.729162	.045246	.292310
12	14 750144	.032385	.888446	19.835539	.046651	.292241	23.355839	.041045	.880924
13	14.848555	.043923	.282925	20.223293	.044391	.564673	24.165394	.046944	.367907
14	15.950030	.035229	.982410	18.567375	.049695	.276108	25.105490	.047232	.718274
15	16.427786	.036366	.967240	17.417574	.047839	.783221	22.654919	.049159	.387955
16	9.372583	.049207	.118098	17.293797	.048288	.310222	22.467934	.049831	.023331
17	9.138135	.045019	.582028	17.457148	.049193	.619598	23.459345	.045388	.800825
18	8.776540	.044549	.636988	17.909870	.042137	.718459	23.880388	.042413	.878349
19	8.919504	.045445	.522889	18.338568	.047260	.885368	24.365887	.049974	.026494
20	8.850932	.043494	.768186	18.533375	.048647	.281070	24.652370	.047747	.527067
21	9.251921	049871	.286302	18.729479	.048421	.491944	24.587753	.049153	.532693
22	9.153990	.047480	.641225	18.765167	.048293	.483542	23.009394	.049814	.526356
23	9.265980	.049365	.173985	19.034439	.047189	.830866	21.582781	.049109	.527600
24	9.590574	.046720	.979581	19.255266	.047752	.797261	21.621170	.048502	.788311
25	8 722542	.047371	.658219	19.055866	.049291	.582151	21.455919	.048973	.327600
26	8.511540	.049047	.462688	18.633085	.049721	.117575	21.612207	.049767	.144514
27	8.676909	048903	.477882	15.056806	.049860	.528413	21.842281	.049012	.297856
28	8.523068	.047114	.592814	14.050842	.049538	.534529	21.999659	.049629	.878186
29	8.724597	.049079	.574114	13.343666	.049696	.698018	22.028124	.049859	.087458
30	8.658084	.047548	.518684	12.916054	.049753	.301181	22.131367	.049104	.312091
31	8.771932	.049280	.613369	12.868749	.049012	.769778	22.310320	.049691	.330227
32	8.513114	.049170	.368207	12.692142	.049930	.228703	22.285065	.049875	.240574
33	8.494273	.048209	.606776	12.647284	.049977	.093743	22.200830	.049423	.991222
34	8 391171	.049888	035789	12.550234	.049974	.059112	22.145418	.049934	.072150
35	8.471379	.048041	.951717	12.572437	.049726	.665249	22.134418	.049137	.841801
36	8.298555	049701	.085663	12.575457	.049146	.670702	21.937300	.049996	.839647
37	8.492036	.048903	.339488	12.536434	.049497	.384135	21.668766	.048973	.728959
38	8.531345	.048074	.878246	12.592931	.049684	.360248	21.015249	.049723	.425235
39	8 186401	.048587	.576924	12.654531	.049865	.956881	19.140820	.049880	.797557
40	8.111301	.047801	.987364	12.565123	.049758	.941215	17.918610	.049907	.839036
41	8 384149	.048022	.866767	12.544221	.049535	.655761	17.310171	.049990	.229544
42	8 312694	.048512	.605419	12.536637	.049805	.249698	16.983387	.049564	.901004
43	8 282926	.049330	.458033	12.359401	.049960	.271974	16.595566	.049969	.123265
44	8 466223	.049047	.807623	12.266216	.049982	.110829	16.408812	.049821	.878964
45	8 353590	.048886	.641447	12.176671	.049799	.542256	16.239330	.049951	.219711
46	8 185480	.049404	.440935	12.169446	.049700	.468714	16.175310	.049975	.153847
47	8 251279	.049434	.427856	12.093030	.049895	.695503	16.056660	.049690	.516541
48	8.323591	.049463	.946299	12.058262	.049937	.189933	16.021381	.049928	.292185
49	8 230300	049105	.627887	12.063498	.049800	.450911	15.960017	.049742	.653673
50	8 404796	.049593	.731914	12.096867	.049971	.176057	15.875415	.050000	.001003

Table 2. Loglikelihood ratio test ($\lambda = 0$) for $\alpha = 0.05$ and $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$

$$T^0_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}) = 2n \sum_{i=1}^{m} \widehat{p}_i \ln\left(\frac{\hat{p}_i}{p_i}\right), \quad q^0_{n,m,t} = P_{\boldsymbol{p}^0}(T^0_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) > t).$$

m	4			6			8		
n	t	q	γ	t	q	γ	t	q	γ
1	...	...		...	...		...	...	
2	...	...		...	...		...	...	
3	...	...		6.931472	.027778	.053333	8.657564	.015625	.104762
4	6.591674	.015625	.183333	9.835395	.004630	.490000	12.136851	.001953	.878571
5	8.858919	.003906	.786667	11.187478	.020062	.776000	11.291710	.025879	.235238
6	8.997362	.018555	.715556	11.090355	.020062	.776000	12.816447	.025879	.313651
7	8.259758	.046387	.117460	11.704834	.015775	.760381	12.959794	.049911	.003156
8	7.776613	.033569	.267063	12.032619	.036280	.914476	13.862944	.025378	.585438
9	8.089309	.038055	.258862	11.568596	.039781	.378413	14.515438	.033965	.677815
10	8.858919	.037094	.447513	11.964197	.041989	.854273	14.404097	.047765	.141723
11	8.610156	.041386	.325859	12.123654	.030633	.704108	14.633574	.045246	.292310
12	8.997362	.036948	.658288	12.136851	.049434	.049384	15.222136	.041278	.858067
13	9.217709	.045457	.986983	12.626974	.043871	.822739	15.469584	.047935	.937536
14	9.109803	.044268	.381219	12.824656	.038820	.964788	15.334577	.046927	.797243
15	9.091181	.049638	.089885	12.374017	.048706	.549137	15.459301	.046646	.742621
16	9.051566	.049207	.118098	12.295287	.047739	.702443	15.414783	.049895	.362658
17	8.349522	.047872	.248695	12.287928	.041578	.808005	15.419436	.048069	.335368
18	8.089309	.047758	.261988	11.966402	.048958	.166544	15.534832	.046317	.426350
19	7.895985	.048494	.172889	12.213305	.049541	.111306	15.677502	.045746	.565481
20	7.960445	.049140	.236878	12.158443	.049479	.116521	15.714861	.048914	.357362
21	8.237560	.045246	.748400	12.222190	.049573	.665637	15.790620	.048528	.655990
22	7.878248	.046792	.459174	12.146267	.049426	.623931	15.632663	.049125	.638428
23	8.223836	.047881	.421989	12.390699	.046786	.712544	15.558018	.048271	.805148
24	8.317766	.045291	.658410	12.191055	.049103	.715715	15.557030	.048612	.730240
25	8.376768	.047225	.694686	12.188744	.048399	.817770	15.585126	.049496	.495564
26	8.413756	.046502	.942946	12.070373	.049868	.717257	15.488575	.048953	.697575
27	8.038909	.049144	.205143	12.055545	.048791	.582495	15.283076	.049716	.550158
28	7.924864	.048117	.902464	12.014827	.049649	.181110	15.254371	.049843	.198520
29	8.026070	.049714	.075732	12.018681	.049577	.541749	15.324442	.049567	.488014
30	8.030598	.048446	.457810	11.895776	.049423	.552595	15.409772	.049929	.174156
31	8.120552	.048904	.352578	11.858018	.049782	.921822	15.344414	.049849	.974906
32	8.043770	.047571	.745938	11.796326	.048740	.657929	15.378973	.049308	.730041
33	8.003992	.049294	.239276	11.734163	.049311	.558682	15.248160	.049816	.680258
34	7.986678	.049329	.240171	11.527147	.049952	.081610	15.272035	.049832	.542859
35	7.974287	.048392	.586010	11.573995	.049596	.712986	15.250461	.049480	.255770
36	7.983791	.049131	.690909	11.505684	.049692	.604079	15.216360	.049971	.097273
37	8.012158	.048076	.808538	11.439818	.049416	.391334	15.184706	.049914	.194920
38	7.973851	.049901	.043146	11.452411	.049840	.332598	15.128876	.049764	.580898
39	8.091228	.047968	.979074	11.465799	.049947	.743498	15.086462	.049928	.183207
40	8.084802	.048452	.753071	11.418536	.049968	.084939	15.033144	.049736	.746625
41	7.911872	.048590	.669389	11.461734	.049848	.382713	15.063690	.049426	.922672
42	8.035414	.048800	.638322	11.467292	.049902	.262389	14.971709	.049751	.799506
43	8.009272	.049770	.247840	11.448422	.049630	.519989	14.944547	.049941	.175378
44	7.964378	.049258	.392860	11.480868	.049859	.528743	14.884252	.049768	.817939
45	8.030094	.048745	.709273	11.455256	.049862	.498734	14.875436	.049809	.781321
46	7.977850	.048940	.641756	11.430017	.049722	.530468	14.815397	.049976	.154592
47	7.989554	.049566	.527617	11.435036	.049904	.835209	14.793161	.049984	.277725
48	7.867796	.049089	.539786	11.404029	.049997	.006885	14.738991	.049765	.803814
49	7.903203	.049127	.565515	11.403136	.049811	.828221	14.724584	.049951	.491432
50	7.942816	.049965	.023293	11.394898	.049600	.928820	14.668654	.049994	.074359

Table 3. Cressie–Read's test ($\lambda = 2/3$) for $\alpha = 0.05$ and $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$

$$T^{2/3}_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) = \tfrac{9}{5} n \left(-1 + \sum_{i=1}^{m} \widehat{p}_i \left(\frac{\hat{p}_i}{p_i^0} \right)^{2/3} \right), \quad q^{2/3}_{n,m,t} = P_{\boldsymbol{p}^0}(T^{2/3}_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) > t).$$

m	4			6			8		
n	t	q	γ	t	q	γ	t	q	γ
1	...	...		...	...		...	...	
2	...	...		...	...		...	...	
3	...	...		6.528753	.027778	.053333	9.050655	.015625	.104762
4	5.832453	.015625	.183333	9.877344	004630	.490000	13.487729	.001953	.878571
5	8.186238	.003906	.786667	10.137394	.020062	.776000	11.290510	.025879	.235238
6	7.406685	.018555	.715556	10.942863	.020062	.776000	11.910490	.025879	.313651
7	8 000628	.020752	.950794	10.551720	.015775	.760381	13.134787	.033089	.603156
8	6.912071	.033569	.267063	10.011272	.048282	.114476	13.057507	.025378	.585438
9	6.716119	.045746	.092196	10.578651	.048449	.103365	12.802979	.040273	.411148
10	7.564440	.037094	.895026	10.373855	.044490	.587606	13.790803	.034512	.785600
11	7.931182	044785	.657624	10.283494	.039801	.370775	13.636891	.036541	.827495
12	7.013424	.048278	.086859	10.332763	.046640	.244328	13.199595	.045509	.441770
13	7.523954	.044076	.643491	10.181168	.048518	.716146	13.337181	.046960	.230065
14	7.778836	.049638	.056178	10.678773	.043648	.548121	13.565905	.043765	.808825
15	7.493391	045611	.290636	10.653409	.049530	.438282	13.723774	.039507	.968004
16	7.484338	.048715	.106351	10.766731	.046698	.448725	13.684767	.045520	.619943
17	7.808631	.044258	.670917	10.850686	.046429	.342589	13.540236	.048670	.779485
18	7.752148	.045904	.478655	10.823076	.048405	.408138	13.495740	.048295	.493338
19	7.785254	.044719	.519619	10.687827	.049376	.252176	13.587867	.049690	.543163
20	7.648801	.049140	.236878	10.799223	.049810	.069042	13.529696	.049395	.177055
21	7.878604	.049652	.130390	10.826941	.049241	.315334	13.775753	.048778	.762274
22	7.682781	.048570	.292326	10.728228	.049041	.760598	13.635578	.049392	.394127
23	7.656865	.048200	.281563	10.908513	.049858	.363168	13.715158	.047438	.808774
24	7.618271	.049795	.037369	10.888792	.049747	.269601	13.695527	.048946	.443755
25	7 534925	.048119	.282562	10.942719	.048991	.751788	13.822061	.048455	.936420
26	7.892993	.048350	.800463	10.801972	.049391	.538145	13.693143	.049320	.905970
27	7 592125	.049678	.077167	10.829724	.049581	.824008	13.771963	.049309	.668310
28	7.721592	.048605	.343734	10.899413	.048316	.709148	13.850690	.049184	.600990
29	7.781187	.048130	.635831	10.849152	.049961	.035938	13.779480	.049884	.875878
30	7.841252	.046916	.706769	10.918189	.049882	.325992	13.821280	.049144	.596520
31	7.701287	046909	.994414	10.938743	.049187	.644592	13.793759	.049738	.880784
32	7.730276	.049353	.200826	10.932748	.049936	.214083	13.829432	.049969	.136098
33	7.804358	.047167	.914029	10.925865	.049586	.894991	13.827413	.049789	.322664
34	7.719558	.049049	.340171	10.983096	.049130	.587357	13.865377	.049846	.957221
35	7.747585	.048803	.791066	10.884257	.049963	.066188	13.873819	.049643	.527057
36	7.633397	.048405	.608779	10.896913	.049518	.557844	13.852202	.049898	.943450
37	7.924823	.048617	.518706	10.929515	.049677	.338922	13.854171	.049934	.726282
38	7 773402	049638	.113486	10.966085	.049690	.686060	13.865868	.049851	.209291
39	7.873756	.047541	.947969	10.938794	.049745	.278796	13.904302	.049869	.481266
40	7.665894	.048432	.604644	10.921140	.049869	.542582	13.892484	.049846	.678665
41	7.720559	.049119	.418084	10.935411	.049933	.158188	13.896460	.049717	.731081
42	7.703407	.049215	.371144	10.966731	.049607	.623283	13.903949	.049691	.780230
43	7.581584	.049638	.218957	10.911161	.049919	.143260	13.930799	.049687	.935567
44	7.830615	048318	.917993	11.005397	.049880	.238871	13.895043	.049964	.184122
45	7.665408	.049341	.328786	10.970819	.049995	.182353	13.922244	.049944	.153918
46	7.708618	.049825	.186054	10.986230	.049683	.825477	13.910903	.049928	.436694
47	7.767457	.049327	.812295	10.997293	.049834	.793914	13.910409	.049999	.003250
48	7.772820	.048273	.989889	10.977602	.049913	.691843	13.901153	.049894	.600499
49	7.843343	049928	.103639	10.963706	.049888	.422450	13.921712	.049957	.489680
50	7.807927	.049970	.015698	10.987567	.049968	.466902	13.943146	.049899	.717789

Table 4. Pearson's χ^2 test ($\lambda = 1$) for $\alpha = 0.05$ and $\boldsymbol{p}^0 = (1/m, \ldots, 1/m)$

$$T^1_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}) = n \sum_{i=1}^{m} \frac{(\hat{p}_i - p_i)^2}{p_i}, \quad q^1_{n,m,t} = P_{\boldsymbol{p}^0}(T^1_{n,m}(\widehat{\boldsymbol{p}}, \boldsymbol{p}^0) > t).$$

m	4			6			8		
n	t	q	γ	t	q	γ	t	q	γ
1	...	...		...	...		...	...	
2	...	...		...	...		...	...	
3	...	...		7.000000	.027778	.053333	10.333333	.015625	.104762
4	6.000000	.015625	.183333	11.000000	.004630	.490000	16.000000	.001953	.878571
5	8.600000	.003906	.786667	10.600000	.020062	.776000	12.600000	.025879	.235238
6	7.333333	.018555	.715556	12.000000	.013632	.808000	12.666667	.025879	.313651
7	8.428572	.020752	.950794	11.000000	.015775	.760381	14.714286	.021873	.716540
8	7.000000	.033569	.267063	10.000000	.048282	.114476	14.000000	.019771	.634210
9	6.555555	.045746	.092196	10.333333	.048449	.064603	13.222222	.040273	.411148
10	7.600000	.037094	.895026	10.400000	.040322	.268453	14.000000	.034512	.633549
11	7.545455	.044785	.657624	10.272727	.040718	.337442	14.454545	.027506	.889104
12	7.333333	.048278	.101336	11.000000	.032887	.622164	13.333333	.042121	.264240
13	7.615385	.037940	.786095	10.538462	.038585	.383066	13.461538	.043485	.334827
14	7.714286	.045879	.289590	10.857142	.038461	.574466	13.428572	.045307	.187331
15	7.666667	.037556	.823969	11.000000	.041804	.663786	13.266666	.049275	.032838
16	7.500000	.043345	.381320	11.000000	.035861	.794477	14.000000	.037668	.673599
17	7.705883	.039979	.501822	10.882353	.040657	.553865	13.588235	.045595	.295591
18	7.777778	.038935	.940321	10.666667	.046096	.230468	14.000000	.039982	.642410
19	7.736842	.045058	.547072	11.000000	.038573	.734003	13.421053	.047959	.111137
20	7.600000	.041518	.453763	10.600000	.047928	.150266	13.600000	.045306	.278618
21	7.761905	.047384	.980730	10.714286	.048079	.167210	13.666667	.045908	.282981
22	7.454545	.047901	.093075	10.727273	.046125	.318871	13.636364	.047399	.205792
23	7.782609	.038623	.951941	10.652174	.049686	.037391	13.521739	.049324	.049585
24	7.666667	.046440	.419270	11.000000	.043883	.628879	14.000000	.042218	.826297
25	7.800000	.040700	.850255	10.760000	.049569	.062358	13.720000	.047581	.229415
26	7.846154	.043324	.955879	10.923077	.044137	.640989	14.000000	.043383	.733981
27	7.814815	.045704	.832790	11.000000	.044427	.768729	13.592592	.049740	.024260
28	7.714286	.042224	.557166	11.000000	.043956	.685813	13.714286	.047896	.224410
29	7.827586	.043801	.893513	10.931034	.046545	.553397	13.758620	.047252	.275095
30	7.866667	.043373	.764272	10.800000	.048479	.201324	13.733334	.047841	.203995
31	7.838710	.044059	.839473	11.000000	.045483	.791099	13.645162	.049426	.055669
32	7.750000	.045698	.404564	11.125000	.042300	.927666	14.000000	.043735	.680280
33	7.606061	.047574	.212773	10.818182	.048569	.249217	13.787879	.047860	.237338
34	7.647059	.046833	.300731	10.823529	.047924	.272691	14.000000	.043997	.638764
35	7.857143	.046292	.789319	10.771428	.049294	.084597	13.685715	.049785	.023796
36	7.777778	.046170	.805722	11.000000	.046182	.748835	13.777778	.047956	.222905
37	7.864865	.046941	.940596	10.837838	.047228	.315433	13.810811	.048097	.249641
38	7.894737	.045984	.900966	10.947369	.046488	.535362	13.789474	.049127	.124379
39	7.666667	.047339	.310966	11.000000	.045951	.756871	14.128205	.043100	.966376
40	7.600000	.049007	.475304	11.000000	.046673	.615011	14.000000	.045844	.666702
41	7.682927	.048318	.298748	10.951220	.046159	.517527	13.829268	.048323	.241678
42	7.714286	.047691	.287899	10.857142	.048515	.233332	14.000000	.046072	.702845
43	7.697674	.048747	.423898	11.000000	.046058	.611585	14.116279	.044133	.988541
44	7.818182	.045114	.588394	10.818182	.049650	.061359	13.818182	.049413	.095956
45	7.711111	.048036	.453403	10.866667	.048249	.224987	13.844444	.048509	.227613
46	7.739130	.045524	.466421	10.869565	.049427	.103503	13.826087	.048577	.198444
47	7.893617	.044238	.953176	11.085107	.044938	.827899	14.106383	.044566	.979756
48	7.833333	.044888	.876154	11.000000	.046089	.714979	14.000000	.046042	.634219
49	7.734694	.048767	.668371	11.122449	.044512	.998418	13.857142	.048931	.186157
50	7.920000	.047329	.681044	10.960000	.047832	.450042	14.000000	.046267	.621153

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman: Data Structures and Algorithms. Addison-Wesley, Massachusetts 1983.
[2] S. M. Ali and S. D. Silvey: A general class of coefficient of divergence of one distribution from another. J. Roy. Statist. Soc. Ser. B *286* (1966), 131–142.
[3] N. A. C. Cressie and T. R. C. Read: Multinomial goodness of fit tests. J. Roy. Statist. Soc. Ser. B *46* (1984), 440–464.
[4] I. Csiszár: Eine Informationstheoretische Ungleichung und ihre Anwendung auf den Beweis der Ergodizität von Markoffschen Ketten. Publ. Math. Inst. Hungarian Academy of Sciences, Series A, *8* (1963), 85–108.
[5] H. Kulmann: Notes on the computation of the exact distribution function of the χ^2 and related tests statistics in the equiprobable case. Comput. Stat. Data Anal., The Statistical Software Newsletter *4* (1996), 707–710.
[6] F. Liese and I. Vajda: Convex Statistical Distances. Teubner, Leipzig 1987.
[7] M. A. Marhuenda, Y. Marhuenda, and D. Morales: Algorithms to calculate the exact distribution function of power divergence statistics. Technical Report of the Operational Research Center, Miguel Hernández University of Elche 2001.
[8] T. R. C. Read and N. A. C. Cressie: Goodness-of-fit Statistics for Discrete Multivariate Data. Springer-Verlag, New York 1988.
[9] M. A. Weiss: Data Structures and Algorithm Analysis. Benjamin/Cummings Publishing Company, Redwood City, CA 1992.
[10] K. Zografos, K. Ferentinos, and T. Papaioannou: ϕ-divergence statistics: sampling properties, multinomial goodness of fit and divergence tests. Comm. Statist. A – Theory Methods *19* (1990), 1785–1802.

Dr. Marco Antonio Marhuenda, Dr. Yolanda Marhuenda and Prof. Dr. Domingo Morales, Operation Research Center, Miguel Hernández University of Elche, Avenida del Ferrocaril s/n, 03202 Elche. Spain.
e-mails: marco@umh.es, y.marhuenda@umh.es, d.morales@umh.es