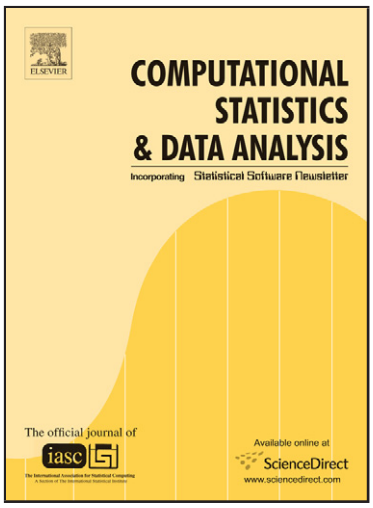

Exact algorithms for computing p-values of statistics-linear combination of 3-nomial variables

Farid Beninel, Gérard Grélaud

PII: S0167-9473(07)00134-X
DOI: doi:10.1016/j.csda.2007.03.019
Reference: COMSTA 3669

To appear in: *Computational Statistics & Data Analysis*

www.elsevier.com/locate/csda

Cite this article as: Farid Beninel and Gérard Grélaud, Exact algorithms for computing p-values of statistics-linear combination of 3-nomial variables, *Computational Statistics & Data Analysis* (2007), doi:10.1016/j.csda.2007.03.019

Exact algorithms for computing p-values of Statistics-Linear Combination of 3-nomial Variables

Farid Beninel [a,*] and Gérard Grélaud [b]

[a] *CREST, ENSAI and UMR CNRS 6086, campus de Ker Lann. 35170 Bruz, France.*
[b] *Université de Poitiers and UMR 6086, UFR de Mathématiques Bd. Marie et Pierre Curie, 86962 Futuroscope, France*

Abstract

Algorithms for computing exact distribution values of statistics-linear combination of some multinomial variables called, 3-nomial variables, are derived. In the general case, there are not explicit formulae of the cumulative distribution function of these statistics that allow the direct computation of p-values. The exact distribution in the independent case and in some dependent ones is studied. For each of the two cases of dependence, an algorithm based on recursive relationships is proposed, giving the number of favorable cases and combinatorics to minimize computation time. For each of these NFC algorithms, concurrent PGF algorithms and some realized comparisons are presented.

Key words: Exact distribution algorithms, generalized linear rank statistics, optimized enumeration, probability generating function.

1. Introduction

The study of some association indexes leads to a linear combination of one order *3-nomial* variables. More precisely, we are concerned with the class of statistics

$$T_{\lambda,a}(Z^{(n)}) := \sum_{k=1}^{n} a_{nk}(Z_{nk}^1 + \lambda Z_{nk}^2). \tag{1}$$

* Corresponding author: CREST, ENSAI, campus de Ker Lann. 35170 Bruz, France.
Tel: +33 (0)2 99 05 32 76
Email addresses: farid.beninel@univ-poitiers.fr (Farid Beninel), grelaud@math.univ-poitiers.fr (Gérard Grélaud).
URL: http://www-math.univ-poitiers.fr/~grelaud/stat/ (Gérard Grélaud).

where $\lambda \in]0,1[$, $a^{(n)} = (a_{n1}, ..., a_{nn}) \in \mathbb{R}^n$ are the deterministic parameters and $Z^{(n)} = (Z_{n1}, ..., Z_{nn})$ is a random vector of one order *3-nomial* variables, *i.e.*, for $k = 1, \ldots, n$, $Z_{nk} = (Z^1_{nk}, Z^2_{nk}, Z^3_{nk}) \sim \mathcal{M}_3(1; p_1, p_\lambda, 1-p_1-p_\lambda)$. The limit cases $T_{0,a}$ and $T_{1,a}$, obtained for $\lambda = 0$ and $\lambda = 1$ correspond respectively to statistics linear-combination of Bernoulli variables $\mathcal{B}(1, p_1)$ and $\mathcal{B}(1, p_1 + p_\lambda)$. Hence, the class of statistics, given by equation (1), could be considered as an extension of statistics linear combinations of Bernoulli variables such as linear rank statistics (see. Capéraà and Van Cutsem (1988), Good (2005), Van der Vaart (1988)). Consequently, the research of appropriated exact distribution methods for the studied class of statistics (1) could extend the existing methodology for linear combination of Bernoulli variables.

More precisely, we are concerned by the exact computation of the CFD value or p-value associated to an observed value of $T_{\lambda,a}$ under a given hypothesis $\mathcal{H}_0$ on the parameters $p_1,\ p_\lambda$. We only consider null hypothesis $\mathcal{H}_0$ leading to the three types of decision rule $Dr1.a,\ Dr1.b,\ Dr2$. These decision rules consist in rejecting the corresponding $\mathcal{H}_0$, *resp.* for small $T_{\lambda,a}$ values, for the large ones and for the extreme ones.

Let us set $\alpha_1 = \Pr(T_{\lambda,a} \leq t|\mathcal{H}_0)$ and $\alpha_2 = \Pr(T_{\lambda,a} \geq t|\mathcal{H}_0)$, the definition of the p-value depends on the adopted decision rule. Here, the p-value is defined as follows

$$p - value(t) = \begin{cases} \alpha_1 & \text{if} \quad Dr1.a, \\ \alpha_2 & \text{if} \quad Dr1.b, \\ 2\min(\alpha_1, \alpha_2) & \text{if} \quad Dr2. \end{cases}$$

Other definitions of the p-value could be found in the paper of Gibbons and Pratt (1975), which is a more complete discussion about this concept. Thus, the computation of the p-value as defined below leads to the computation of the $T_{\lambda,a}$ CDF values.

From a practical point of view, we encounter the statistics-linear combination of such multinomial variables in two distinct situations. In the first situation, the studied statistic is used to measure the level of concordance between assignations to classes of the same set of objects, applying two distinct assignation rules. Although, the statistic given by equation (1) generalizes the well assigned rate (Beninel and Grun Rehomme (2006)). In the second one, the studied statistics are used to measure the level of association between two ringed individuals given *presence-absence* data from a *capture-recapture* design (Cairns and Schwager (1987), Roberts and Evans (1993)). Related to these two situations, we give the following two examples.

Example 1: An independent case

This example is related to the estimation of a generalized concordance rate computed over a finite population of companies. Each company has to answer, at two different dates, the question of the type of its business. The first date is the creation and the second one at a survey. Let $\Omega = \{\omega_1, \ldots, \omega_N\}$ be the population of concerned companies and the random pair $(Z^1, Z^2)(\Omega \longrightarrow \{0,1\}^2)$.

Given a company ω, $(Z^1(\omega), Z^2(\omega)) = (1,0)$ if the company responses are concordant; $(Z^1(\omega), Z^2(\omega)) = (0,1)$ if different but indicate nearest business; $(0,0)$ if discordant. Let $Z(\omega_k) := (Z^1_{Nk}, Z^2_{Nk})$ and $\lambda \in [0,1]$, the generalized concordance rate over the population is $T := \frac{1}{N}\sum_{k=1}^{N}(Z^1_{Nk} + \lambda Z^2_{Nk})$. Let us denote $(Z^1_{nk}, Z^2_{nk})_{k=1,\ldots,n}$ the sampling random pairs, *i.e.*, given an independent sample $\mathcal{S} = \{\omega^*_1, \ldots, \omega^*_n\} \subset \Omega$, $(Z^1_{nk}, Z^2_{nk})(\mathcal{S}) = (Z^1, Z^2)(\omega^*_k) = (Z^1(\omega^*_k), Z^2(\omega^*_k))$. In this case sampling pairs are *i.i.d.* and when values

a_{nk} are survey weights, the statistic given in (1) is the Horvitz-Thompson estimate of the global rate T.
Table 1 gives an example of related data we use to illustrate computation of an observed value of $T_{\lambda,a}(Z^{(n)})$ and the associated p-value.

Table 1
Concordance scores of the 25 individuals from an observed sample

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Z^1_{nk}	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	0
Z^2_{nk}	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
$10^3.a_{nk}$	122	122	73	93	122	41	32	55	58	29	44	31	24	26	28	28	5	7	10	4	0	7	11	12	16

Here $T_{\lambda,a} = 10^{-3}(122+122+73+122+41+29+44+24+26+28+28+7+10+4+0+11+12+\lambda(55+58+31+5+16))$. For $\lambda = 0.5$, we have the observed value $T_{\lambda,a} = 0.7855$. For normalized weights, as here, the minimum and maximum values of the statistic $T_{\lambda,a}(Z^{(n)})$, with respect to the sample $Z^{(n)}$, are $\min(T_{\lambda,a}) = 0$ and $\max(T_{\lambda,a}) = 1$ respectively. Using the `PNFC1` algorithm which we introduce in subsection 3.3.1, we obtain the exact probability related to the p-value, $P(T \leq 0.7855 \mid p_1 = p_\lambda = \frac{1}{3}) = 0.9977$.

Example 2: A dependent case

In some *capture-recapture* experiments, one observes over a set of successive dates, the presence or absence at a given geographical area, ringed individuals from a finite animal population. The problem, we focus on is the measure of the association of two ringed individuals o_X, o_Y. Denote $\mathcal{T}$ the set of dates where the two individuals could be observed. Consider the associated *presence-absence* variables X, Y ($\mathcal{T} \longrightarrow \{0,1\}$), *i.e.*, $X(t) = 1$ (*resp.* $Y(t) = 1$) if o_X (resp. o_Y) is present on the site; $X(t) = 0$ (*resp.* $Y(t) = 0$) otherwise.

Let us define the random pair (Z^1, Z^2) ($\mathcal{T} \rightarrow \{(0,1),(1,0),(0,0)\}$), by $Z^1 = XY$, $Z^2 = (1-X)(1-Y)$. Hence, $Z^1_{nk} = 1$ when the two individuals are simultaneously present at the k^{th} date, $Z^2_{nk} = 1$ when absent simultaneously and $Z^3_{nk} = 1$ when only one is present. We measure the association of the individuals o_1, o_2 by

$$p(o_1, o_2) = P(Z^1 = 1) + \lambda P(Z^2 = 1) = p_1 + \lambda p_\lambda, \tag{2}$$

where $\lambda \in [0,1]$. The λ values depends on how the model of association takes into account the information of simultaneous absences. This definition extends the definition given in (Cairns and Schwager (1987)) for the particular case $\lambda = 0$. In the context of spatial association, Rao and Simple Matching indexes are based on a similar definition of the association (Hubaleck (1982)).
Given an observed sample $\{(X_{n1}, Y_{n1}), \ldots, (X_{nn}, Y_{nn})\}$ of (X, Y) values corresponding to the successive dates $\{t_1, \ldots, t_n\}$ and considering $\{a_{n1}, \ldots, a_{nn}\}$ as the associated weights, $T_{\lambda,a}$ is an estimate of the association $p(o_1, o_2)$.

As an example of such data, we present in table 2 some ecological data related to a *capture-recapture* study (Bretagnolle and Beninel (1994), Delstrade (1999)).

The permutational approach used to study the statistic $T_{\lambda,a}$ considers as possible samples those generated from the observed one, using pairs of permutations, *i.e.*, each

Table 2
Presence-absence process of two ringed individuals and corresponding Z^1, Z^2 values.

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
X_{nk}	1	0	1	0	1	1	0	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	0	1	1
Y_{nk}	0	1	1	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0	1
Z^1_{nk}	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	1
Z^2_{nk}	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
$10^3 a_{nk}$	89	26	51	32	61	45	35	41	26	28	29	25	54	30	50	36	29	42	28	50	41	30	26	34	62

pair of permutations (σ_1, σ_2) of the set $\{1, \ldots, n\}$, corresponds to a possible sample $\{((\sigma_1 X)_{nk}, (\sigma_2 Y)_{nk}) : \ k = 1, \ldots, n\}$.

We have

$$\sum_{k=1}^{n} (\sigma_1 X)_{nk} = \sum_{k=1}^{n} X_{nk} = u_X, \qquad \sum_{k=1}^{n} (\sigma_2 Y)_{nk} = \sum_{k=1}^{n} Y_{nk} = u_Y.$$

Corresponding to data presented in table 2, we have $u_X = 15$, $u_Y = 16$. The associated possible samples $\{(Z^1_{nk}, Z^2_{nk}) : \ k = 1, \ldots, n\}$ are such that $Z^1_{nk} = (\sigma_1 X)_{nk}(\sigma_2 Y)_{nk}$ and $Z^2_{nk} = (1 - (\sigma_1 X)_{nk})(1 - (\sigma_2 Y)_{nk})$. In this case, the random pairs (Z^1_{nk}, Z^2_{nk}) are dependent.

Here the estimate of the association is $T_{\lambda,a} = 10^{-3}(51 + 61 + 41 + 54 + 50 + 29 + 50 + 62 + \lambda(32 + 36))$. For $\lambda = 0.5$, one obtains $T_{\lambda,a} = 0.432$. To evaluate this observed value, we refer to maximum and minimum possible values of $T_{\lambda,a}$ using formulae in subsection 2.2 obtaining $\max(T_{\lambda,a}) = 0.851$ and $\min(T_{\lambda,a}) = 0.159$. However, the use of the p-value is more informative on how large or how small an observed value is.

Using `PNFC2` algorithm described in subsection 3.3.2, we obtain

$$P(T_{\lambda,a} \leq 0.432 | \mathcal{H}_0) = 0.3862.$$

Here $\mathcal{H}_0$ consists in assessing the uniform distribution for pairs of permutations (σ_1, σ_2). To rely the Z distribution, we establish that under this null hypothesis $p_1 = \dfrac{u_X u_Y}{n^2}$ and $p_\lambda = \dfrac{(n - u_X)(n - u_Y)}{n^2}$.

In the general case, there is no explicit formula for the exact CDF of statistics $T_{\lambda,a}$. For large n values, central limit theorems could be used to derive an approximate CDF, having satisfied some conditions on the deterministic parameters (see. Billingsley (1995), Van der Vaart (1988)).

Exact computation of probabilities $P(T_{\lambda,a} \leq t | \mathcal{H}_0)$ is studied for small n values, *i.e.*, values for which computation is feasible in an acceptable amount of time. For the limit cases $\lambda = 1$ and $\lambda = 0$, different exact computation methods exist. Our aim here is to extend these methods to our class of statistics. The existing methods deal with exact distribution algorithms. The recent development of computation tools implies important advances of these kinds of algorithms. We classify these exact algorithms in three categories:

• A first class covers algorithms which compute some or all coefficients of the probability generating function (we abbreviate PGF). This class of algorithms covers the earlier algorithm of Streiberg and Röhmel (1986) called *shift algorithm*. This algorithm

is based on a recurrence formula giving the probability generating function. Using the same representation of PGF, Hothorn and Hornick (2002) extend the algorithm to statistics incorporating real or rational scores.

For these algorithms the representation of the PGF is in a matrix form. The use of symbolic methods introduced by Di Buchianico (1999) and Van de Wiel (2001) optimizes significantly the time of computation. Another kind of optimization in the context of symbolic methods is proposed by Van de Wiel (2001) (*Split up algorithm*).

• The second class of exact algorithms is based on the use of characteristic functions. The algorithm consists in solving a linear complex equations system giving the probability distribution. The equations system is obtained by applying the inversion theorem in the discrete case (Brigham (1988)). The most famous algorithm of this class is the *FFT algorithm* of Pagano and Trichler (1983).

• The third class of algorithms is based on the number of favorable cases and covers what we call *NFC algorithms*. These algorithms are built using recursions giving the number of favorable cases associated to the probability one wishes to compute. The number of terms to compute recursively is minimized at each step when possible. As examples of such algorithms, we have the algorithms developed in (Castagliola (1996)) and (Beninel and Husson (1999)) for statistics linear combination of Bernoulli variables and the *Network algorithm* of Mehta and Patel (1983), used in the *StatXact* software, for a larger set of statistics.

The p-value is computed using only an *NFC algorithm* or a *PGF algorithm*. These two approaches will be compared in the two real contexts discussed below and leading to an *i.i.d* case and a dependent case.

Next developments are organized as follows. In section 2 we give some combinatoric results to build functions `NFC1` and `NFC2` (computing numbers of favorable cases) called in the *PNFC algorithms* which compute p-values. These results consist in a recursive relationship giving the number of favorable cases and other results to optimize the computation time. In section 3, we present results to build *PGF algorithms* computing p-value in the studied contexts. Section 4 is devoted to the software and section 5 to simulations and examples.

2. NFC Algorithms

Let us set $N_1 = \sum_{k=1}^{n} Z_{nk}^1$ and $N_\lambda = \sum_{k=1}^{n} Z_{nk}^2$. For $(N_1, N_\lambda) = (k, l)$, $T_{\lambda,a}(Z^{(n)})$ values could be written as $\sum_{j=1}^{k} a_{ni_j} + \lambda \sum_{j=1}^{l} a_{ni_{k+j}}$ where $(i_1, ..., i_{k+l}) \subset \{1, \ldots, n\}$. For $j_0, k, l \in \{0, \ldots, n\}$ such that $k + l \leq n - j_0$, consider the set

$$E_{\lambda,a}(k, l, t|j_0) = \{(i_1, \ldots, i_{k+l}) \subset \{j_0 + 1, \ldots, n\} : \sum_{j=1}^{k} a_{ni_j} + \lambda \sum_{j=1}^{l} a_{ni_{k+j}} \leq t\}$$

and denote $K_{\lambda,a}(k, l, t|j_0)$ its cardinal.

The particular set $E_{\lambda,a}(k, l, t|0)$ corresponds to the set of favorable cases $Z^{(n)}$ associated to events $\{T_{\lambda,a}(Z^{(n)}) \leq t|(N_1, N_\lambda) = (k, l)\}$. To calculate the associated probability, it suffices to compute $K_{\lambda,a}(k, l, t|0)$, *i.e.*, and to use the following equation

$$P(T_{\lambda,a} \leq t|(N_1, N_\lambda) = (k, l)) = \frac{K_{\lambda,a}(k, l, t|0)}{\binom{n}{k}\binom{n-k}{l}}. \quad (3)$$

Consequently, to compute the p-value, *i.e.*, the total probability, for the two context described below, we use equation

$$P(T_{\lambda,a} \leq t) = \sum_{k,l} \frac{P((N_1, N_\lambda) = (k,l))}{\binom{n}{k}\binom{n-k}{l}} K_{\lambda,a}(k,l,t|0). \tag{4}$$

Given the distribution of the random pair (N_1, N_λ), computation of $P(T_{\lambda,a} \leq t)$ values leads to computation of $K_{\lambda,a}(k,l,t|0)$ values. As algorithms based on the number of favorable cases, we propose algorithms `PNFC1` and `PNFC2` corresponding *resp.* to the independent and the dependent case defined below. Each, of these two algorithms, uses a recurrence relation to compute $K_{\lambda,a}(k,l,t|0)$ and they differ by the (N_1, N_λ) distribution. These two `NFC` algorithms will be compared respectively to `PPGF1` and `PPGF2` concurrent PGF algorithm we present in section 3.

2.1. *An example of enumeration*

The table 3 shows counted combinations when computing numbers $K_{\lambda,a}(1,2,t|0)$ for $t = 7.0$ or $t = 8.0$, with $\lambda = 0.5$ and $a = \{1,2,3,4,5,6,7\}$.

Table 3
Admissible combinations

a_{i_1}	a_{i_2}	a_{i_3}	a_{i_1}	a_{i_2}	a_{i_3}	a_{i_1}	a_{i_2}	a_{i_3}
1	2	3,4,5,6,7	2	1	3,4,5,6,7	3	1	2,4,5,6,7
	3	4,5,6,7		3	4,5,6,7		2	4,5,6,[7]
	4	5,6,7		4	5,6,[7]		4	[5],[6]
	5	6,7		5	[6],[7]			
	6	[7]						

a_{i_1}	a_{i_2}	a_{i_3}	a_{i_1}	a_{i_2}	a_{i_3}	a_{i_1}	a_{i_2}	a_{i_3}
4	1	2,3,5,[6],[7]	5	1	2,3,[4]	6	1	[2],[3]
	2	3,[5],[6]		2	[3],[4]			
	3	[5]						

$K_{\lambda,a}(1,2,8.0|0)$ is the number of combinations $(a_{i_1}, a_{i_2}, a_{i_3}) \subset \{1,\ldots,7\}$ such that $a_{i_1} + 0.5(a_{i_2} + a_{i_3}) \leq 8$ and it is equal to 56. To obtain the number $K_{\lambda,a}(1,2,7.0|0)$, we subtract the number of combinations $(a_{i_1}, a_{i_2}, [a_{i_3}])$ from $K_{\lambda,a}(1,2,8.0|0)$. We obtain $K_{\lambda,a}(1,2,7.0|0) = 39$.

The second array gives, for the same parameters, numbers $K_{\lambda,a}(1,2,t|j_0)$ for different j_0 values.

Table 4
Numbers of admissible combinations

$j_0 \backslash t$	0	1	2
7.0	39	10	0
8.0	56	21	3

2.2. *Recursive relationships*

Denote $\sum_{\lambda,a}(k,l|j)$ the set of sums corresponding to $T_{\lambda,a}(Z^{(n)})$ values when $(N_1, N_\lambda) = (k,l)$ and call $S^+_{\lambda,a}(k,l)$, $S^-_{\lambda,a}(k,l|j)$ *resp.* the associated maximum and minimum. In what recurs, we suppose without loss of generality, that $a_{n1} \leq a_{n2} \leq ... \leq a_{nn}$. We have

$$S^+_{\lambda,a}(k,l) = \lambda(a_{n,n-k-l+1} + ... + a_{n,n-k}) + a_{n,n-k+1} + ... + a_{n,n},$$

$$S^-_{\lambda,a}(k,l|j) = a_{n,j+1} + ... + a_{n,j+k} + \lambda(a_{n,j+k+1} + ... + a_{n,j+k+l}).$$

Note that, the only maximum is independent from j and this justifies the adopted notations. Computation of $K_{\lambda,a}(k,l,t|j)$ depends on t value.

To begin, let us consider t values outside the interval $[S^-,\ S^+]$ which covers the particular situations at limits. For these t values we have the following equation

$$K_{\lambda,a}(k,l,t|j) = \begin{cases} \binom{n-j}{k}\binom{n-j-k}{l} \text{ if } t \geq S^+_{\lambda,a}(k,l), \\ \\ 0 \qquad\qquad\quad \text{if } t < S^-_{\lambda,a}(k,l|j)\ . \end{cases} \tag{5}$$

For the general case, we derive the following recurrence relation on which we build the enumeration algorithm.

$$K_{\lambda,a}(k,l,t|0) = \sum_{j=1}^{n-k-l+1} K_{\lambda,a}(k-1,l,t-a_{nj}|j) + K_{\lambda,a}(k,l-1,t-\lambda a_{nj}|j)$$

with boundary condition

$$K_{\lambda,a}(0,0,t|j) = \begin{cases} 1 & \text{if } t \geq 0, \\ 0 & \text{elsewhere.} \end{cases}$$

In practice, this relation is implemented in `NFC2` to compute $K_{\lambda,a}(k,l,t|j)$. In the following section, we propose an improvement in order to shorten computational time. The optimization of the algorithm consists in reducing the number of recursions at each step. The simplifications proposed rely on the intrinsic properties of the deterministic parameters, *i.e.*, the sequence $a^{(n)}$ and the parameter λ.

2.3. *Optimization*

Optimization is achieved using, whenever possible, the result given by equation (5) to compute numbers $K_{\lambda,a}(k,l,t|j)$.

At step j, we decompose $K_{\lambda,a}(k,l,t|j)$ as a sum of numbers defined analytically and numbers defined recursively. Let us set

$$\begin{aligned} M_1(k,l,t) &:= \text{Max}\{j:\ S^+_{\lambda,a}(k-1,l) \le t - a_{nj}\},\\ m_1(k,l,t) &:= \min\{j:\ S^-_{\lambda,a}(k-1,l|j) > t - a_{nj}\},\\ M_\lambda(k,l,t) &:= Max\{j:\ S^+_{\lambda,a}(k,l-1) \le t - \lambda a_{nj}\} \end{aligned}$$

and

$$m_\lambda(k,l,t) := \min\{j:\ S^-_{\lambda,a}(k,l-1|j) > t - \lambda a_{nj}\}.$$

We easily establish inequalities $m_\lambda \ge m_1 \ge M_1 + 1 \ge M_\lambda + 1$.

Let us set

$$A_1(k,l,t) := \sum_{j=1}^{M_1} \binom{n-j}{k-1}\binom{n-j-k+1}{l},$$

$$A_\lambda(k,l,t) := \sum_{j=1}^{M_\lambda} \binom{n-j}{k}\binom{n-j-k}{l-1},$$

$$R_1(k,l,t) := \sum_{j=1}^{m_1-1} K_{\lambda,a}(k-1,l,t-a_{nj}|j)$$

and

$$R_\lambda(k,l,t) := \sum_{j=1}^{m_\lambda-1} K_{\lambda,a}(k,l-1,t-\lambda a_{nj}|j).$$

Using equation (5), we can derive from (2.2) the following reduced recurrence relation

$$K_{\lambda,a}(k,l,t|0) = A_1(k,l,t) + A_\lambda(k,l,t) + R_1(k,l,t) + R_\lambda(k,l,t). \tag{6}$$

2.4. *Other results*

Let $S = \{s_1, ..., s_n\}$ a sequence of real scores, $j \in \{1, ..., n\}, k \le n - j$ and $t \in \mathbb{R}$. let us set

$$N_S(k,t|j) = \sharp\{(1, ..., k) \subset \{j+1, ..., n\} :\quad \sum_{i=1}^{k} s_i \le t\}. \tag{7}$$

In (Beninel and Husson (1999)) we introduced an algorithm (called, here, `NFC1`) for computation of numbers $N_S(k,t|j)$. This algorithm will be used as a function in `NFC2` devoted to two sequences of scores and computing $K_{\lambda,a}(k,l,t|j)$. The call of `NFC1` in `NFC2` occurs at the step $k = 0$ or $l = 0$. The number of combinations resulting from the two sequences of scores $a^{(n)}$ and $\lambda a^{(n)}$ and the number of combinations among one sequence, *i.e.*, $a^{(n)}$ or $\lambda a^{(n)}$, are linked by the following equations

$$K_{\lambda,a}(k,0,t|j) = N_a(k,t|j) \tag{8}$$

and

$$K_{\lambda,a}(0,k,t|j) = N_{\lambda a}(k,t|j). \tag{9}$$

In order to simplify, we use only one of the N_a or the $N_{\lambda a}$ functions, applying the following equation. For $\lambda \neq 0$

$$N_a(k, \frac{t}{\lambda}|j) = N_{\lambda a}(k, t|j). \tag{10}$$

The algorithm `NFC2` and `NFC1` could be viewed as an adaptation of the *network algorithm* (Mehta and Patel (1983)). The associated network is a rooted tree where a level is defined as the number of nodes from the root. One could define the level as the set of nodes with the same $k+l$ value. At each of these levels, each call of the recursive function defines a step, at each step we have to determine $m_\lambda, m_1, M_1, M_\lambda$ and to compute the value A_1+A_λ and to recall the recursive function as necessary as it appears in number $R_1 + R_\lambda$.

2.5. *Some inequalities*

For $j, k, l \in \mathbb{N}$ such that $k + j + l \leq n$ and $t \in \mathbb{R}$, we have

$$N_a(k+l, t|j) \leq \frac{1}{\binom{k+l}{k}} K_{\lambda,a}(k, l, t|j) \leq N_{\lambda a}(k+l, t|j). \tag{11}$$

These inequalities could be used to derive upper and lower bounds of the p-value. They also allow detection of the extreme values using only an algorithm of computation among a unique sequence of scores.

3. The algorithms

The algorithms we propose differ on the (N_1, N_λ) distribution and the chosen enumeration method (*NFC* or *PGF*).

3.1. (N_1, N_λ) *distribution*

3.1.1. *The independent case*

Here the null hypothesis $\mathcal{H}_0$, concerns p_1, p_λ values. In our simulation we consider the null hypothesis $\mathcal{H}_0 : p_1 = p_\lambda = \frac{1}{3}$. $Z_{n1}, \ldots, Z_{nn}$ are independent and follow a $\mathcal{M}_3(1, p_1, p_\lambda, 1-p_1-p_\lambda)$ distribution. Hence $(N_1, N_\lambda, N-N_1-N_\lambda) \sim \mathcal{M}_3(n, p_1, p_\lambda, 1-p_1-p_\lambda)$. Consequently for k, l such that $0 \leq k+l \leq n$,

$$P((N_1, N_\lambda) = (k, l)) = \binom{n}{k}\binom{n-k}{l} p_1^k p_\lambda^l (1-p_1-p_\lambda)^{n-k-l}. \tag{12}$$

3.1.2. *The dependent case*

Here, the null hypothesis $\mathcal{H}_0$ is that the pairs of permutations are uniformly distributed. Consequently, $Z_{n1}, \ldots, Z_{nn}$ are such that $N_1 \sim \mathcal{H}(n, u_X, u_Y)$ and $N_\lambda = n - u_X - u_Y + N_1$.

Let us set $M_1 = \min(u_X, u_Y)$, $m_1 = \max(0, u_X + u_Y - n)$ and $l(k) = n - u_X - u_Y + k$. For $k \in \{m_1, \ldots, M_1\}$, $l = l(k)$,

$$P((N_1, N_\lambda) = (k, l)) = P(N_1 = k) = \frac{\binom{u_X}{k}\binom{n-u_X}{u_Y-k}}{\binom{n}{u_Y}}. \tag{13}$$

3.2. *PPGF algorithms*

Consider $P(x, y, w)$ an homogeneous polynomial of variables x, y, w and $P(x, y, w)[y^k w^l]$ the polynomial (of variable x) derived from $P(x, y, w)$, as a sum of the coefficients associated to $y^k w^l$.
Denote $G_{T,k,l}$ the PGF associated to the conditional distribution of T with respect to $(N_1, N_\lambda) = (k, l)$

$$G_{T,k,l}(x) := \prod_{i=1}^{n} (1 + yx^{a_i} + wx^{\lambda a_i})[y^k w^l]. \tag{14}$$

The global generating function associated to T is

$$G_T(x) = \sum_{k,l:\ 0 \le k+l \le n} \mu(k, l) G_{T,k,l}(x), \tag{15}$$

where

$$\mu(k, l) = \frac{P((N_1, N_\lambda) = (k, l))}{\binom{n}{k}\binom{n-k}{l}}.$$

In the following we present the PGF corresponding to two distinct situations developed in the introduction. For these situations two distinct PGF algorithms are built.

3.2.1. `PPGF1`*: An algorithm using PGF in the i.i.d case*

For this case, the distribution of the random pair (N_1, N_λ) is given by equation (12); substituting in equation (15), we obtain

$$G_T(x) = \sum_{k,l} p_1^k p_\lambda^1 (1 - p_1 - p_\lambda)^{n-k-l} \prod_{i=1}^{n} (1 + yx^{a_i} + wx^{\lambda a_i})[y^k w^l],$$

or, equivalently

$$G_T(x) = \prod_{i=1}^{n} (1 - p_1 - p_\lambda + p_1 x^{a_i} + p_\lambda x^{\lambda a_i}). \tag{16}$$

In what follows, we suppose that the conditions on the scores (a_{nk}) are satisfied, allowing a polynomial representation of the PGF, *i.e.*, $G_T(x) = \sum_k c_k\ x^{d_k}$. Given the scores a_{nk}, the parameter λ and p_1, p_λ the probabilities values under $\mathcal{H}_0$, the algorithm `PPGF1` compute coefficients c_k, the associated degrees d_k and derive at the end the p-value, that is probability $P(T \le t) = \sum\{c_k : d_k \le t\}$. To compute the coefficients c_k under interest, we use directly the polynomial form of the global PGF. As a possible improvement of the algorithm, one could adapt *Split Up algorithm* of Van de Wiel (2001).

3.2.2. `PPGF2`*: The algorithm for a dependent case*

Here we use equation (13) to substitute in equation (15) and we derive the global PGF for this dependent case, *i.e.*,

$$G_T(x) = \sum_{k=m_1}^{M_1} \mu(k) \prod_{i=1}^{n} (1 + yx^{a_i} + wx^{\lambda a_i}))[y^k w^{l(k)}], \tag{17}$$

where

$$\mu(k) = \frac{\binom{u_X}{k}\binom{n-u_X}{u_Y-k}}{\binom{n}{u_Y}\binom{n}{k}\binom{n-k}{n-u_X-u_Y+k}}. \tag{18}$$

Let $G_{T,k,l(k)}(x) = \sum_m c_{k,m}x^{d_m}$ be the polynomial form of the conditional PGF. Given the scores (a_{nk}), the parameter λ and the data(u_X, u_Y, t), to compute the p-value given by $P(T \leq t)$, `PPGF2` works as follow: for each $k \in \{m_1, .., M_1\}$, we compute $S_k(t) = \sum_m\{c_{k,m} : d_m \leq t\}$ and $\mu(k)$, and obtain the researched probability $P(T \leq t) = \sum_{k=m_1}^{M_1} \mu(k)\ S_k(t)$. Quantities $S_k(t)$ are computed via a direct use of the polynomial form of the conditional PGF. An optimization of the computation time could be obtained using an adaptation of *Split Up* algorithm.

3.3. *PNFC algorithms*

Algorithms `PNFC1` and `PNFC2` to compute p-value *resp.* are proposed for the $i.i.d$ case and for dependent one. Computation is based on the equation (4) and uses as a called function `NFC2` the programming of the recurrence relation given by equation (6). These two PNFC algorithms differ only on the used distribution of the counting pair (N_1, N_λ).

3.3.1. *PNFC1*

Here, we use equation (12) to substitute in equation (4). Given $p_1, p_\lambda, \lambda, t$ and the sequence a of scores, PNFC1 returns the probability

$$P(T \leq t) = \sum_{k,l:\ 0\leq k+l\leq n} p_1^k p_\lambda^l (1-p_1-p_\lambda)^{n-k-l} K_{\lambda,a}(k,l,t|0). \tag{19}$$

For the particular case $\mathcal{H}_0 :\ p_1 = p_\lambda = \frac{1}{3}$, we have to compute

$$P(T \leq t|\mathcal{H}_0) = \frac{1}{3^n} \sum_{k,l:\ 0\leq k+l\leq n} K_{\lambda,a}(k,l,t|0).$$

3.3.2. *PNFC2*

In an analogous manner we use equation (13) and substitute in equation (4), `PNFC2` returns

$$P(T \leq t|\mathcal{H}_0) = \sum_{k=m_1}^{M_1} \mu(k) K_{\lambda,a}(k, n-u_X-u_Y+k, t|0). \tag{20}$$

Here $\mu(k)$ is given by equation (18) and m_1, M_1 in subsection 3.1.2.

4. Software

The different programs presented in this paper are available at the web site:

`http://www-math.univ-poitiers.fr/~grelaud/stat/`.

The programs computing p-value, *i.e.*, `PNFC1`, `PNFC2`, `PPGF1`, `PPGF2` are presented in two file formats. A `Maple` format to be loaded with the Maple Software and a `html` format. In the `html` file, we present the source of the program and comments and documentation to improve portability.

The choice of the Maple Software is to benefit from the use of tools devoted to computation related to polynomials. The only functions NFC1 and NFC2 are presented in `Maple` langage and `R` langage.

5. Examples and some simulations

In what concerns **example 1**, the appropriate algorithms to compute p-value are `PNFC1` and `PPGF1`. We have as input $\lambda = 0.5$, $t = 0.7855$, the sequence a_{25} given in table 1. The algorithm `PNFC1` returns $P(T \leq 0.7855) = 0.9977$ after 1177 seconds. Computing using `PPGF1` makes the system run out of memory after 186 seconds.

For the **example 2**, we obtain using `PNFC2` $P(T_{\lambda,a} \leq 0.432) = 0.3862$ and with 105 seconds time length. Here also the PPGF algorithm, `PPGF2` makes the system out of memory.

The realized simulations consist in the testing of the program computing the number of favorable cases. For the programs PNFC and PPGF computing probabilities, tests and simulations are realized. In the *iid* case, simulations allow some comparison of `PPGF1` and `PNFC1`. For the dependent case, we only give some examples; simulations to compare `PPGF2` and `PNFC2` have to be completed.

5.1. *Computing the number of favorable cases*

Here, to test the performance of the program `NFC2`, simulations are made using an *Intel Pentium 4 Cpu AT* compatible. These simulations consist in generating randomly score sequences $a^{(n)}$ and in computing $K_{\lambda,a}(k, l, t \mid 0)$ the number of combinations of interest. Performance in the sense of computing time depends upon the length of the score sequences, the position of the t-values and the integer k, l values.

We undertook three small simulation studies to test the performance of the proposed algorithm. Such a performance is evaluated by the mean time of computation obtained from $N = 100$ sequences of scores (generated randomly from the uniform distribution).

5.1.1. *On the position of the t value*

For fixed $\lambda, a^{(n)}$ we considered different values between the maximum S^+ and the minimum S^-. Computing time appears as an increasing function of absolute deviation of t value from the extreme values.

5.1.2. *On the scores sequence*

The computation time depends on the dispersion of the score sequence. This dispersion determines relatively the position of the t value to the extreme S^+ and S^-. In general, the algorithm is fast for $n \leq 30$ and is more efficient for extreme values.

5.1.3. *On the (k, l) values*

Computation time increases with $(k+l)$ and decreases with $|k-l|$ value. Here t is not fixed as an absolute value but only by its relative value on the interval $[S^-, S^+]$.

Table 5
The behavior of the computation time in seconds (and the number K of admissible combinations) with respect to t-value (here $\lambda = 0.5$ and $n = 15$).

t	min	$+0.2$	$+0.4$	$+0.6$	$+0.8$	$+1.0$	$+1.2$	$+1.4$	$+1.6$	$+1.8$	$+2.0$	$+2.2$
K	1	51	490	2019	6099	15055	31292	58194	95313	143645	197346	253741
Mean time	0.00	0.02	0.03	0.09	0.17	0.30	0.54	0.72	1.02	1.22	1.42	1.44

$+2.4$	$+2.6$	$+2.8$	5.063	$+3.0$	$+3.8$	$+4.0$	$+4.2$	$+4.4$	max
304328	347341	378344	399451	411064	417095	419463	420266	420413	420420
1.31	1.08	0.86	0.57	0.34	0.19	0.08	0.01	0.00	0.00

Table 6
Computation time with respect to $(k - l)$ value (here $n = 15$, $k + l = 11$, $t = 4.4$ and $\lambda = 0.5$).

$k - l$	11	9	7	5	3	1	-1	-3	-5	-7	-9	-11
time	~ 0	~ 0	0.05	0.28	0.98	2.02	2.52	1.76	0.64	0.12	~ 0	~ 0

5.2. *Comparing `PNFC1` and `PPGF1`*

In the beginning we present examples of results when processing the two algorithms on a unique sequence of scores (a_{nk}). These results allow some comparison of the two algorithms from a computational time point of view and could be used to verify the exactness of computation. The scores sequences are generated randomly; we use the *Rand* function of *Maple* and must be ordered in an increasing order to be considered in input and $\lambda = 0.5$ (the default value).

Example 3: $n = 10$

Consider the generated sequence $a = \{21, 29, 37, 41, 42, 47, 56, 70, 76, 82\}$. The following table returns for different t values, the associated probability $P(T \leq t)$ and the computation time when using *resp.* `PNFC1`, `PPGF1`.

Table 7
Computation time in seconds and probability with `PPGF1` and `PNFC1`.

t	30	80	130	180	230	280	330	380	430	480	530
PNFC1	0.141	0.344	0.781	1.545	1.985	2.375	3.078	3.719	4.20	4.38	4.75
PPGF1	0.	0.	0.094	0.203	0.360	0.625	0.844	1.141	1.250	1.391	1.281
Prob	0.0002	0.0054	0.041	0.160	0.388	0.66	0.87	0.99	0.996	0.9999	1

Example 4: $n = 15$

The generated sequence here is $a = \{4, 9, 10, 29, 66, 75, 104, 113, 119, 125, 134, 138, 144, 148, 150\}$.

In a second step we repeat simulations to compare in a more consistent manner `PNFC1` and `PGF1`. We organize simulations as follows: We fix the n and x values. For each pair (n, x) we generate randomly from the uniform distribution N sequences of scores $a^{(n)}$

Table 8
Computation time in seconds and probability with `PPGF1` and `PNFC1`. ** : System out of memory.

t	70	120	170	220	270	320	370	420	470
PNFC1	4	6.8	10.162	14.39	20.79	30.64	43.25	65.46	83.57
PPGF1	0.	0.109	0.235	0.593	1.5	3.59	7.98	15.56	94.031
Prob	0.00002	0.00012	0.00056	0.0019	0.0054	0.0134	0.0294	0.057	0.101

520	570	600	650	700	750	800	850	900	950	1000	1050	1100
111.079	135.359	122	168	203	227	250	272	298	319	342	361	383
644.43	3826	**	**	**	**	**	**	**	**	**	**	**
0.166	0.251	0.31	0.42	0.53	0.65	0.75	0.83	0.90	0.94	0.97	0.98	0.99

($N = 10$ for $n = 15$ and $N = 5$ for $n = 20$). For each sequence of scores, we compute $P(T \leq x \max(T))$. The following table gives for each pair (n, x) the mean time to compute associated probability.

Table 9
Computation time in seconds; for each double line, the first one gives performance of `PNFC1` and the second one for `PPGF1`.

$n\backslash x$	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
$n = 15$	1.95	3.4	5.15	6.9	8.7	10.2	11.0	11.1	12.7
	0.59	1.13	1.20	3.25	7.63	14.62	77.73	**	**
$n = 20$	17.5	31.12	47.30	64.9	85	106	132	160	192
	0.17	5.4	5.93	19.5	148.7	**	**	**	**

5.3. *Comparing* `PNFC2` *and* `PPGF2`

A more complete simulation, taking into account of all the parameters, *i.e.*, $(\lambda, x, n, u_X, u_Y, a, \ldots)$, is to organize. Here, we only present some examples results from tests to give an idea about the computation time performance of these two algorithms.

Example 5:
Consider the generated scores sequence $a = \{3, 4, 5, 5, 9, 10, 23, 33, 44, 44, 47, 62, 67, 70, 72\}$ and fix $t = 24$ for example. Table 10 presents computation time and $P(T \leq 24)$ values with respect to (u_X, u_Y) values.

For the realized tests, the two algorithms work when $\min(u_X, u_Y) \leq 20$ and `PNFC2` is more rapid . Computation time increases highly when the x value increases and depends on the u_X, u_Y values more than on the n value.

Example 6:
Here $a = \{5, 6, 8, 21, 26, 30, 32, 46, 48, 88, 88, 92, 93, 98, 113, 115, 120, 128, 135, 135, 147, 173, 176, 194, 199\}$. Table 11 presents computational time and probability for different fractil

Table 10
Probability and computation mean time with `PPGF2` and `PNFC2` when $U_X = 4$ and $U_Y = 9$ to 15

U_Y	9	10	11	12	13	14	15
$P(T \leq 24)$	0.0056	0.0087	0.0100	0.0118	0.012	0.0119	0.0659
PPGF2	137	101.2	58	25	8.2	1.9	0.2
PNFC2	0.24	0.26	0.16	0.14	0.02	0.01	~ 0

x and (u_X, u_Y) values. We use the only `PNFC2` program; `PPGF2` fails with '*out of memory*' error message.

Table 11
Computational time in seconds and probability using `PNFC2`

$x \setminus (u_X, u_Y)$	(5, 7)	(5, 10)	(5, 12)	(10,10)
0.1	7.8	29	31	4.4
	0.098	0.07	0.12	0.18
0.2	16.6	39	33	128.6
	0.2	0.2	0.28	0.34
0.5	31	43	40.5	151
	0.52	0.5	0.59	0.53
0.7	50	78	67	186
	0.75	0.74	0.71	0.71
0.9	72	88	76	460
	0.9	0.88	0.86	0.93
0.95	87	99	92	723
	0.97	0.96	0.95	0.99

6. Conclusion

The simulations have to be completed. However, we give some concluding remarks, based on the realized simulations and tests. In general cases, algorithms based on recursions giving the number of favorable cases need time for computation. However, the *NFC algorithms* could be optimized to shorten the computation time by the reduction of some recursive computing. In cases where the probability generating function is available in a direct polynomial form, as in the *i.i.d* context, *PGF algorithms* are faster for small sequences of scores. For great n values, depending on the used computer, PGF algorithms generate problems of memory occupancy. In the case when the polynomial form of the PGF is obtained using some supplementary computing, as in the dependent case, *NFC algorithms* could be more performant.

As a perspective, the *PGF algorithms*, we propose here, will be optimized, using an adaptation of the *Split up algorithm*. Another kind of optimization of all these proposed algorithms should be to use cutting rules when given the maximum level of significance

or to propose a minimal p-value using inequalities proposed in subsection 2.5 when overlapping a fixed computation time.

Acknowledgement

The authors would like to thank the three anonymous referees and associated editor Professor Cristian Gatu for their many helpful comments and suggestions.

References

Beninel, F., Grun Rehomme, M. (2006). Evaluation of an allocation rule under some cost constraints. *Data Science and Classification, Springer verlag* , 67–73.

Beninel, F., Husson, F. (1999). An optimized algorithm to determine the values of the exact C.D.F. of some discrete statistics. *Computational Statistics 14, 2*, 251–261.

Billingsley, P. (1995). *Probability and Measure*. Wiley series in probability and mathematical statistics (third edition).

Bretagnolle, V., Beninel, F. (1994). Un indice d'association basé sur des variables temporelles de présence-absence : application à l'étude des structures sociales. *Ecology and Statistical methods, INIST-CNRS, Cote INIST : Y 30598*, 67-70

Brigham, E.O. (1988). *The Fast Fourier Transform and Its Applications*. Prentice-Hall.

Capéraà, Ph., Van Cutsem, B. (1988). *Méthodes et Modèles en statistique non paramétrique*. Presses de l'université Laval, Dunod, Montreal.

Cairns, S.J., Schwager, S.J. (1987). A comparison of association indices. *Anim. Behav. 35*, 1454–1469.

Castagliola, P. (1996). An optimized algorithm for computing Wilcoxon's statistic when N is small. *Computational Statistics 11, 1*, 1–10.

Delstrade, A. (1999). Foraging strategy in a social bird, the alpine chough: effect of variation in quantity and distribution of food. *Animal Behaviour 57*, 299–305.

Di Buchianico, A. (1999). Combinatorics, computer Algebra and the Wilcoxon-Mann-Whitney test. *Journal of Statistical Planing and Inference 79*, 349–364.

Gibbons, J.D., Pratt, J.W., (1975). P-values : interpretation and methodology. *American Statistician 29*, 20–25.

Good, P. (2005). *Permutation, parametric and Bootstrap tests of hypotheses*. Springer-Verlag, New York.

Hothorn, T., Hornick, K. (2002). Exact nonparametric inference in R. *Compstat 2002. Physica Verlag 14, 2*, 355–361.

Hubaleck, Z. (1982). Coefficients of association and similarity based on binary (presence-absence) data : An evaluation. *Biological Review 57*, 669–689.

Mehta, C.R., Patel, R., (1983). A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables. *JASA 78*, 427–434.

Pagano, M., Trichler, D. (1983). On obtaining permutations distributions in polynomial time. *JASA 78*, 435–440.

Roberts, G., Evans, P. R. (1993). A method for detection of non random association. *Behavioral Ecology and Sociobiology 15*, 349–354.

Seber, G.A.F (1982). *Estimating of animal abundance and related parameters*. Charles Griffin and Company., London.

Streiberg, B., Röhmel, J. (1986). Exact permutation and rank tests : An introduction to some recently published algorithms. *Statistical Software Newsletter 12*, 10–17.

Van der Vaart, A. W., (1998). *Asymptotic statistics*. Cambridge series in statistical and probabilistic mathematics.

Van de Wiel, M.A. (2001). The split up algorithm : a fast symbolic method for computing p-values of rank statistics. *Computational statistics 16*, 519–538.