Statistics and Computing (1994) **4**, 287–302

Gray codes for randomization procedures

PERSI DIACONIS

Department of Mathematics, Harvard University, One Oxford Street, Cambridge, MA 02138, USA

SUSAN HOLMES

INRA, Unité de Biométrie, 2, Place Pierre Viala, 34060 Montpellier, France and Department of Statistics, Sequoia Hall, Stanford University, CA 94305, USA

Received January 1994 and accepted April 1994

We introduce a simple combinatorial scheme for systematically running through a complete enumeration of sample reuse procedures such as the bootstrap, Hartigan's subsets, and various permutation tests. The scheme is based on Gray codes which give 'tours' through various spaces, changing only one or two points at a time. We use updating algorithms to avoid recomputing statistics and achieve substantial speedups. Several practical examples and computer codes are given.

Keywords: Bootstrap, gray-code, low rank update, pesmutation test

1. Introduction

This paper introduces Gray codes as systematic procedures for complete enumeration of sample reuse procedures such as the bootstrap and permutation tests. Gray codes are a well developed part of combinatorial algorithms. They allow a 'tour' of a set of combinatorial objects making minimal changes at each step. Updating algorithms allows us to avoid complete recomputation of statistics of interest.

As an example, consider the law school data used by Efron (1982). The data consist of 15 pairs of numbers (GPA, LSAT) for a sample of American law schools. The correlation coefficient is $\hat{\rho} = 0.776$. The bootstrap gives a perturbation analysis as explained in Section 3. We carried out an exact computation of the bootstrap distribution using the Gray codes. A histogram of the values of ρ is shown in Fig. 1.1. There is a 'corner' in this figure at about 0.9 which does not show up even in large Monte Carlo replications (Fig. 1.2). We discuss this corner further in Section 4. The point for now is that there are features of the distribution picked up by the present analysis not available by routine Monte Carlo or asymptotic analysis.

Section 2 presents the original Gray code for binary n-tuples and gives applications to Fisher's randomization test and Hartigan' subsets. These problems can also be treated exactly using the fast Fourier transform on binary n-tupules. We show that the Gray code provides an improvement on these approaches in many situations.

Section 3 develops Gray codes for the bootstrap. There the combinatorial space consists of compositions of n. We explain the Gray codes, review the updating literature, discuss algorithms for m and n versions of the bootstrap and discuss a variety of uses of these complete enumeration procedures.

Section 4 presents some comments and examples, Section 5 discusses other non-parametric procedures where suitable Gray codes can be applied.

One can only hope to enumerate completely for moderate sample sizes (currently $n \leq 20$). For larger sample sizes partial enumeration through carefully spaced points is discussed in Diaconis et al. (1994a). Another idea is to use a small dose of randomness, not as much as Monte Carlo, by doing a random walk between close points so as to be able to use updating procedures all the same, this is detailed in the case of exploring the tails of a bootstrap distribution in Diaconis and Holmes (1994b).

As will emerge from the examples below, the exact results give a good feel for the usefulness of asymptotics.

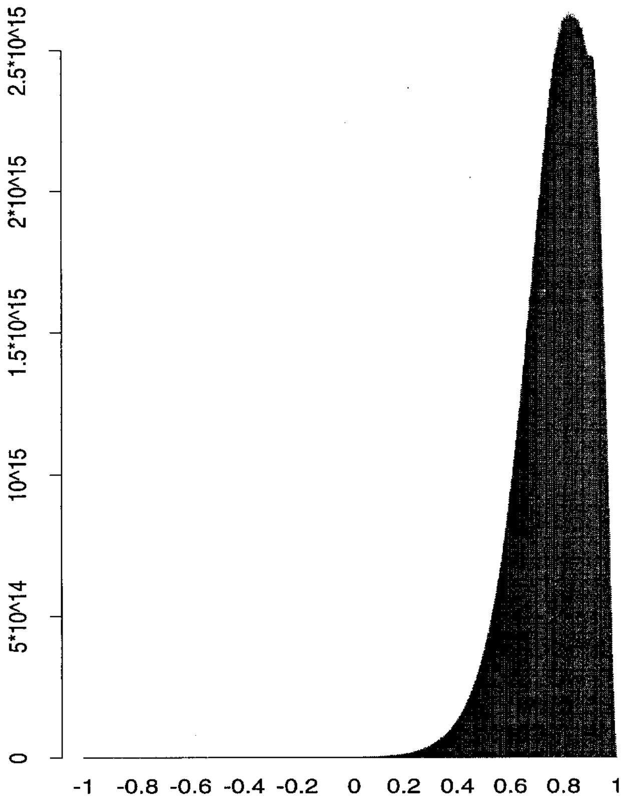

Fig. 1.1. *Exhaustive bootstrap for the correlation coefficient of the law school data*

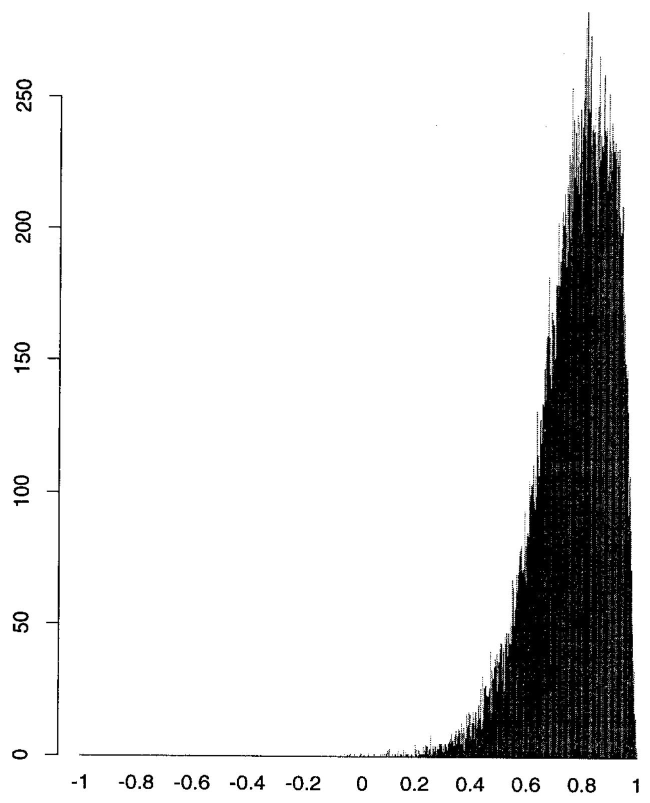

Fig. 1.2. *Monte Carlo bootstrap for the law school data,* $B = 40\,000$

2. The original Gray code, Fisher's test and Hartigan's subsets

2.1. *The original Gray code*

Let $\mathbb{Z}_2^n$ be the set of binary n-tuples. This may be identified with the vertices of the usual n-cube or with the set of all subsets of an n-element set. The original Gray code gives an ordered list of $\mathbb{Z}_2^n$ with the property that successive values differ only in a single place. For example, when $n = 3$ such a list is

$$000, 001, 011, 010, 110, 111, 101, 100.$$

It is easy to give a recursive description of such a list, starting from the list for $n = 1$ (namely 0,1). Give a list L_n of length 2^n, form L_{n+1} by putting a zero before each entry in L_n, and a one before each entry in L_n. Concatenate these two lists by writing down the first followed by the second in reverse order. Thus from 0,1 we get 00,01,11,10 and then the list displayed above for $n = 3$. For $n = 4$ the list becomes:

0000,0001,0011,0010,0110,0101,0100,1100,
1101,1111,1110,1010,1011,1001,1000.

Gray codes were invented by F. Gray (1939) for sending sequences of bits using a frequency transmitting device. If the ordinary integer indexing of the bit sequence is used then a small change in reception, between 15 and 16, for instance, has a large impact on the bit string understood. Gray codes enable a coding that minimizes the effect of such an error. A careful description and literature review can be found in Wilf (1989). One crucial feature: there are non-recursive algorithms for providing the successor to a vector in the sequence in a simple way. This is implemented through keeping track of the divisibility by 2 and of the step number.

One way to express this is as follows: let $m = \Sigma\epsilon_i 2^i$ be the binary representation of the integer m, and let $\ldots e_3e_2e_1e_0$ be the string of rank m in the Gray code list. Then $e_i = \epsilon_i + \epsilon_{i+1}$ (mod 2) $(i = 0, 1, 2\ldots)$ and $\epsilon_i = e_i + e_{i+1} + \cdots$(mod2) $(i = 0, 1, 2\ldots)$. For example, when $n = 4$, the list above shows the string of rank 6 is 0101; now $6 = 0110 = 0\cdot 1 + 1\cdot 2 + 1\cdot 4 + 0\cdot 8$. So $e_0 = 0 + 1 = 1$, $e_1 = 1 + 1 = 0$, $e_2 = 1 + 0 = 1$, $e_3 = 0 + 0 = 0$. Thus from a given string in the Gray code and the rank one can compute the successor. There is a parsimonious implementation of this in the algorithm given in the appendix. Proofs of these results can be found in Wilf (1989).

We study two uses for such Gray codes here: they provide a way of complete enumeration for Fisher's randomization procedure and for Hartigan's subset procedure. Similar procedures can be used for signed rank statistics and the usual two sample (m out of n) tests.

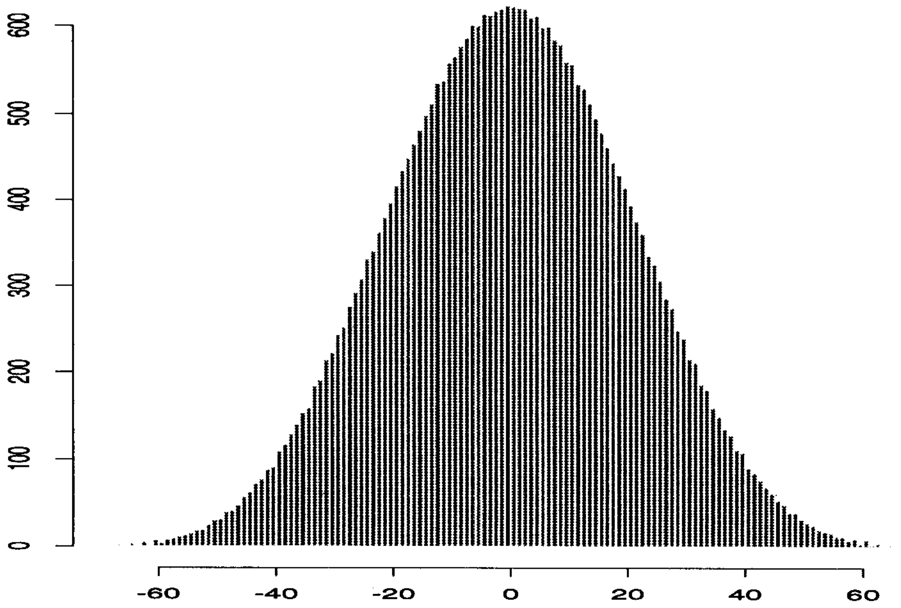

Fig. 2.1. *Randomization distribution of S_n for Zea data*

2.2. *Fisher's randomization test*

Fisher (1935) gave a permutation justification for the usual test for n paired observations $(x_1,y_1),\cdots,(x_n,y_n)$. In his example (Darwin's *Zea* data) x_i and y_i were real numbers representing plant height for treated and untreated plants. Darwin had calculated the mean difference. Fisher gave a way of calibrating this by calculating $d_i = |x_i - y_i|$ and considering all 2^n possible sums $S_n = \epsilon_1 d_1 + \cdots + \epsilon_n d_n$ with $\epsilon_i = \pm 1$.

We observe that a Gray code allows us to run systematically through all 2^n patterns. One need only update the present value of the sum by changing a single value. Furthermore the symmetry of the problem saves a factor 2 of the computations. Figure 2.1 shows a histogram of the randomization distribution based on this procedure applied to the *Zea* data. The original value of the sum S_0 is 39.25 inches. This clearly falls in the tail of the distribution. In fact the exact proportion of sign patterns with a larger absolute sum is 0.0518 (as Fisher also calculated).

For the remainder of this section we use the exact distribution to compare two methods of approximation for the significance level of Fisher's randomization test.

Given $(x_1,y_1),\ldots,(x_n,y_n)$ let $S_{n0} = (x_1 + \cdots + x_n) - (y_1 + \cdots + y_n)$. Let $d_i = |x_i - y_i|$ and let M_n be the number of choices of 2^n sign values such that $S_n = \epsilon_1 d_1 + \epsilon_2 d_2 + \cdots + \epsilon_n d_n$ is strictly larger than S_{n0}. For Fisher's example $M_{15} = 835$ giving a p-value of $M_n/2^n = 0.0518$.

The first approximation to M_n uses the fact that S_n is a sum of independent (non-identically distributed) random variables. Thus with $\sigma_n^2 = d_1^2 + \cdots + d_n^2$,

$$P\{S_n \le x\} = \Phi\left(\frac{x}{\sigma_n}\right). \tag{2.1}$$

Figure 2.2a shows a plot of $\{F_{\text{true}}(x\sigma_n) - \Phi(x)\}$ versus x for the *Zea* data. The normal approximation is quite good; it gives a p-value of 0.0538.

Fisher himself suggested a second approximation: using the studentized statistic

$$T_n = \frac{\sum_i X_i/\sqrt{n}}{\left\{\sum_{i=1}^{n}(X_i^2 \times \bar{X})^2/(n-1)\right\}^{1/2}} \qquad \text{where} \quad X_i = \epsilon_i d_i.$$

This T_n is a one-to-one increasing function of S_n (if $f(s) = \sqrt{(n-1)/n}S\{\Sigma_i d_i^2 - S^2/n\}^{1/2}, T_n = f(S_n)$). So the number of sign patterns such that T_n is strictly larger than the observed T_{n0} equals M_n. Thus Fisher's t-approximation is based on

$$P\{T_n \le x\} \doteq F_{n-1}(x), \tag{2.2}$$

with $F_{n-1}(x)$ the t_{n-1} distribution function.

Figure 2.2b shows the difference between the true distribution and the t-approximation: $\{G_{\text{true}}(x) - F_{n-1}(x)\}$ versus x for the *Zea* data. This approximation is also quite good, perhaps better than the normal approximation for these data. It gives a p-value of 0.0497.

It is natural to ask when one approximation is better than the other. It is possible to give a surprisingly clear answer. The following result was proved in joint work with Tom DiCiccio.

Theorem Let $(x_1,y_1),\cdots,(x_n,y_n)$ be n pairs of real numbers, let $d_i = |x_i - y_i|$ and $\kappa = n\Sigma d_i^4/(\Sigma_i d_i^2)^2$. Then the normal approximation to M_n (based on (2.1)) is more accurate than the t-approximation to M_n (based on (2.2)) to second order if and only if $\kappa \le 3/2$.

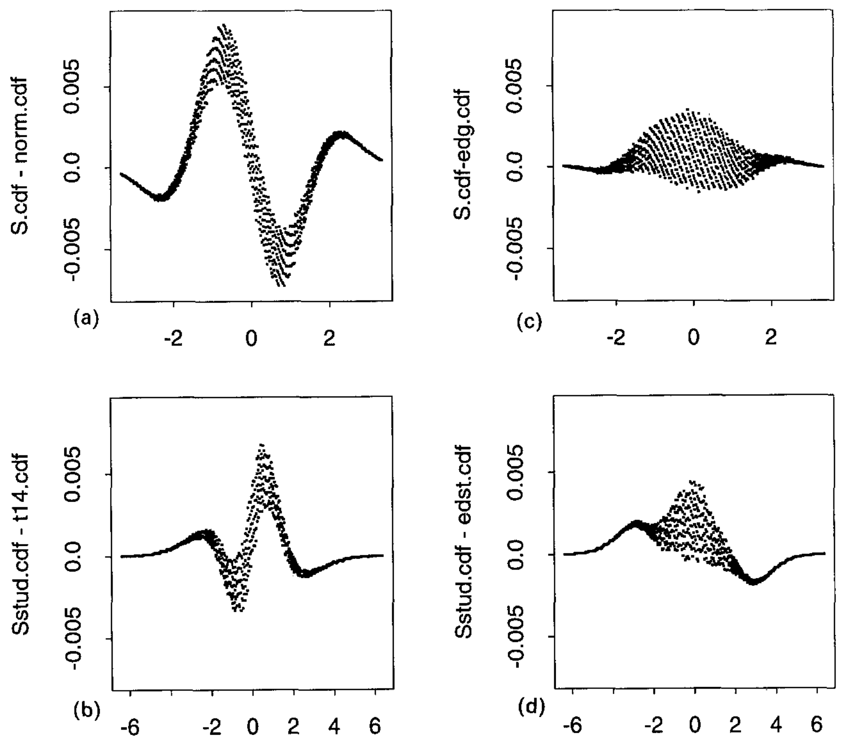

Fig. 2.2. *Comparisons between the true and the approximate distributions*

Proof The argument proceeds by computing Edgeworth expansions to the difference between the two sides of (2.1) and (2.2) and comparing lead terms.

For the normal approximation, let

$$F_e(z) = \Phi(z) + \phi(z)\frac{\kappa}{12n}(z^3 - 3z). \qquad (2.3)$$

Then Albers *et al.* (1976) showed there is a constant A such that

$$\left|P\left\{\frac{S_n}{\sigma_n} \le x\right\} - F_e(x)\right| \le An^{-5/4} \qquad (2.4)$$

uniformly in x. There are mild conditions in their theorem which we detail in the remark below.

For the t-approximation to M_n we can write that

$$\begin{aligned} P\{S_n \le x\} &= P\{f(S_n) \le f(x)\} \\ &= P\{T_n \le f(x)\} \doteq F_{n-1}(f(x)). \qquad (2.5)\end{aligned}$$

The validity of the Edgeworth approximation (2.4) and calculus based on (2.5) gives the following Edgeworth approximation to the permutation distribution of the studentized statistic:

$$\begin{aligned} P\{T_n \le y\} &= \Phi(y) \\ &\quad + \frac{\phi(y)}{12n}\{y^3(\kappa - 6) + y(6 - 3\kappa)\} + O(n^{-5/4}). \end{aligned} \qquad (2.6)$$

To evaluate the quality of the t-approximation in (2.5) we also need the normal approximation to the t-distribution

$$F_n(y) = \Phi(y) - \frac{\phi(y)}{4n}\{y^3 + y\} + O(n^{-2}). \qquad (2.7)$$

This is a consequence of Barndorff-Nielsen and Cox (1989), p. 74. It also follows from (2.6) setting $\kappa = 3$.

We now put the pieces together, neglecting error terms of smaller order than n^{-1}; we have from (2.3), (2.4):

$$P\{S_n \le x\} \doteq \Phi\left(\frac{x}{\sigma_n}\right) + C_1\left(\frac{x}{\sigma_n}\right), \quad C_1(y) = \frac{\delta(y)}{12n}\kappa(y^3 - 3y).$$

From (2.6)

$$P\{T_n \le f(x)\} \doteq \Phi(f(x)) + C_2(f(x))$$

$$C_2(y) = \frac{\phi(y)}{12n}\{y^3(\kappa - 6) + y(6 - 3\kappa)\}$$

From (2.7)

$$F_{n-1}(f(x)) \doteq \Phi(f(x)) + C_3(f(x)) \quad C_3(y) = -\frac{\phi(y)}{4n}(y^3 + y)$$

Thus the lead error term for the normal approximation is $|C_1(x/\sigma_n)|$. The lead error term in the t-approximation is

$$|C_2(f(x)) - C_3(f(x))| = \frac{\phi(f(x))}{12n}|(\kappa - 3)||f(x)^3 - 3f(x)|.$$

In order to compare these expressions $f(x)$ is expanded:

$$f(x) = \frac{x}{\sigma_n} + \frac{1}{2n}\left(\frac{x}{\sigma_n}\right)^3\left(1 + O\left(\frac{1}{n}\right)\right).$$

Straightforward algebra shows the two lead terms as a function of $z = x/\sigma_n$:

$$\text{normal} : \frac{\phi(z)}{12n}\kappa|z^3 - 3z|, \qquad t : \frac{\phi(z)}{12n}|\kappa - 3||z^3 - 3z|.$$

Now $\kappa \leq |\kappa - 3|$ if and only if $\kappa \leq 3/2$.

Remark: To make the theorem complete, we need to specify the conditions on d_i so that Theorem 2.1 of Albers, Bickel and Van Zwet (1976) applies. Let $\gamma(\epsilon)$ be the Lebesgue measure of an ϵ neighbourhood of the d_i's: $\gamma(\epsilon) = \lambda\{x : |x - d_j| \text{ for some } j\}$. Suppose that for some positive c, C, δ we have $\Sigma d_i^2 \leq cN$, $\Sigma d_i^4 \leq CN$ and $\gamma(\epsilon) \leq \delta N\epsilon$ for some $\epsilon \leq N^{-3/2}\log N$. Then (2.4) holds with $A > 0$ depending on c, C, δ and ϵ.

Remarks 1. In Darwin's *Zea* data, $\kappa = 1.93$; Figs 2.2a and 2.2b substantiate the conclusion of Theorem 1: the t-approximation appears slightly better.

2. One can improve over the straightforward normal approximation (2.1) by using the Edgeworth approximation (2.3). Figure 2.2c shows $\{F_{\text{true}}(z) - F_e(z)\}$ versus z for the *Zea* data. This is an improvement over the already good fit of Fig. 2.2a. Similarly Fig. 2.2d shows the Edgeworth corrected approximation to the t-statistic based on (2.6). This is $\{G_{\text{true}}(z) - G_{\text{edst}}(z)\}$ versus z. Again the Edgeworth approximation is an improvement over what is displayed in Fig. 2.2b. All the plots in Fig. 2.2 were computed from complete enumeration by using Gray coding.

We note that there is no reason to use the mean (Pearson, 1937; Efron, 1969). Fisher used it because Darwin had. The Gray code can be useful with any statistic and savings increase if efficient updating for single changes are available. We discuss this in more detail in Section 3.4.

Fisher's test and this data set have been extensively discussed in the statistics literature. Fisher himself avoided complete computation by looking at the tails and coming in carefully. Usual approximations proceed by Monte Carlo; however, Pagano and Tritchler (1983) give a fast Fourier algorithm for complete enumeration. We give a different Fourier implementation and further discussion in Section 2.4. Lehmann (1975) and Tritchler (1984) discuss exact confidence intervals derived from Fisher's randomization test. Manly (1991) carries out the exact computation without using an updating procedure. Basu (1980) and his discussants debate the logic of such 'permutation' tests. Maritz (1981) gives a clear presentation of more general two-sample location problems. The Gray code approach can be used for $n \leq 30$. It also adapts to the vector-valued T^2 statistic studied by Eaton and Efron (1970).

2.3. *Hartigan's subsets*

Hartigan (1969) developed an exact confidence interval for a location parameter. This is explained in Maritz (1979) and Efron (1982). If $\mathscr{X}_n = \{x_1, x_2, \cdots, x_n\}$ is a sample observed from F_θ with $\Pr_\theta(X \in A) = \int_A f(x - \theta)dx$ an M-estimate $\hat{\theta}$ for θ is any solution of $\Sigma\psi(x_i - z) = 0$. Here ψ is antisymmetric and increasing.

There are $2^n - 1$ non-empty subsets of the original n data points. Any of these can be used to generate a new $\hat{\theta}^{(s)}, s = 1, \cdots, 2^n$. Hartigan (1969) showed that the $2^n - 1$ values, $\hat{\theta}^{(s)}$, once ordered, partition the line into 2^n intervals and that each of these has probability 'exactly' $1/2^n$ of containing the true value of θ. This leads to exact confidence intervals for θ.

It is straightforward to run through all subsets via a Gray code. Figures 2.3–2.6 show examples based on subsets on a sample of size $n = 10$ generated by Maritz (1979) from a Cauchy distribution. Three location estimates have been made, the median, the M-estimate based on Huber's psi

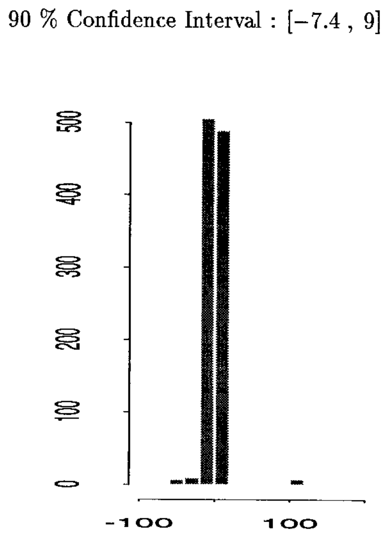

Fig. 2.3. *Subsets for the median*

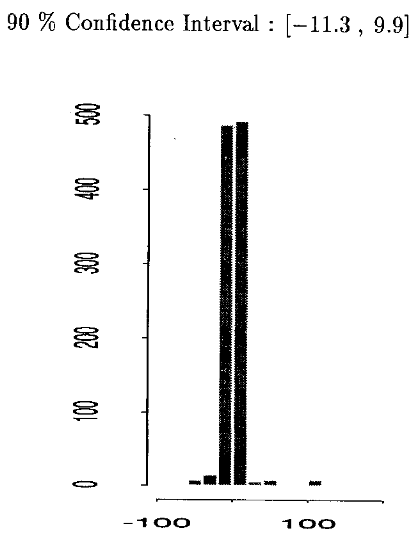

Fig. 2.4. *Subsets with Huber's psi*

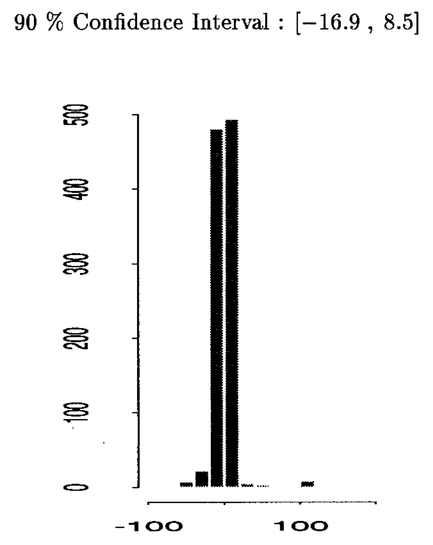

Fig. 2.5. *Subsets with Tukey's biweight*

and the one based on Tukey's biweight. As a comparison we have also provided the bootstrap of the median for this sample (Fig. 2.7). The subsets procedures lead to distributions peaked at a few values. The bootstrap distribution seems slightly smoother.

Hartigan thought of the subset procedure much more broadly. One can induce a distribution for very general statistics using subsets. The subsampling procedures introduced by Hartigan (1969) were studied further by Forsythe and Hartigan (1970) who proved the method asymptotically efficient with respect to Student's t in the case of i.i.d. normal variables.

Gordon (1974a,b) studied the efficiency in the long-tailed case. He also gave convenient combinatorial conditions for classes of subsets to yield efficient procedures. Developing Gray codes for these classes is a challenging open problem.

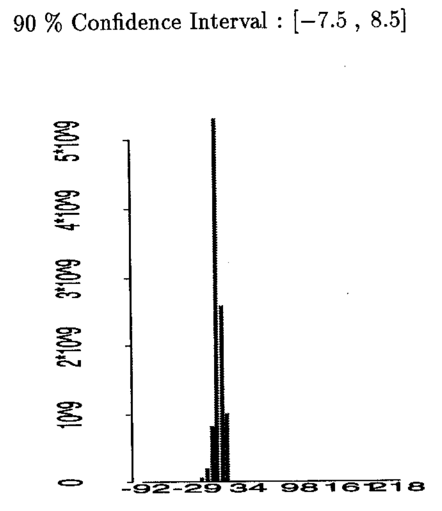

Fig. 2.6. *Bootstrap for the median*

2.4. *A Fourier connection*

This section amplifies a remark in Hartigan (1969). Let $\mathbb{Z}_2^n$ be the binary n-tuples thought of as a group with coordinatewise addition (mod 2). For $x, y \in \mathbb{Z}_2^n$, the inner scalar product (mod 2) is denoted $x \cdot y$.

Let $f : \mathbb{Z}_2^n \to \mathbb{R}$ be a function. The Fourier transform of f is defined by:

$$\hat{f}(y) = \sum_x (-1)^{x \cdot y} f(x).$$

Naive computation of this for a fixed value of y takes $n \times 2^n$ operations. Allowing all values of y will thus take order $n \times 2^{2n}$ operations. The fast Fourier transform (FFT) is a collection of algorithms for computing $\hat{f}(y)$ for all y which only needs $n2^n$ operations. It was originally conceived by Yates for statistical applications and Good (1958) abstracted to the modern FFT. A comprehensive review can be found in Elliott and Rao (1982). Varying a suggestion of Hartigan (1969) we observe that the FFT for this case allows exact computations of all 2^n possible sums in Fisher's randomization test.

Lemma Let $z_i, 1 \leq i \leq n$ be n real numbers. Define

$$f : \mathbb{Z}_2^n \to \mathbb{R} \text{ by } \begin{cases} f(e_i) = z_i, & 1 \leq i \leq n \\ f(x) = 0, & \text{otherwise} \end{cases}$$

$$e_i = \begin{cases} e_i = (\delta_{ij})_{j=1,\cdots,n} \\ \quad = (0, \cdots, 0, 1, 0, \cdots, 0) & \text{the } i\text{th basis vector} \end{cases}$$

Then $\{\hat{f}(y)\}_{y \in Z_2^d}$ takes all 2^n values $\epsilon_1 z_1 + \cdots + \epsilon_n z_n$; $\epsilon_i \in \{-1, +1\}$.

Remarks The FFT gives an order $n2^n$ algorithm for these problems. The Gray code gives an order 2^n algorithm. There is another approach to exact computation for this problem developed by Pagano and Tritchler (1983). This uses the FFT on a different group and, in a sense that is developed below, works in polynomial time in n. The idea is to consider the distribution of $\epsilon_1 z_1 + \cdots + \epsilon_n z_n$ as a convolution of n independent non-identically distributed random variables and then compute the law of the sum using the FFT. To implement this, z_i must be taken as integers. We will give a running time analysis of the Pagano–Tritchler algorithm.

Let $N = |z_1| + \cdots + |z_n|$ this is a measure of the size of the input. To do convolutions exactly one must work on Z_{3N} (to avoid wrap-around). Then one must compute n transforms of $\pm|z_i|$ (cost $n(3N)\log(3N)$). These must be multiplied together pointwise (cost $n \cdot 3N$) and then an inverse transform is needed (cost $3N \log 3N$). Thus, this FFT approach works in order $n \cdot (3N)\log(3N)$ operations. If all $|z_i| \leq B$, then $N \leq nB$ and one may say the algorithm works in a polynomial in n.

In practice, much depends on the size of $|z_i|$. As an example, Darwin's data are originally measured in eighths of an inch. Multiplying by 8 gives $N = |z_i| + \cdots + |z_{15}| = 545$.

Then order $15 \cdot (3N)\log(3N) = 181\,470$ steps are needed for this FFT approach.

By contrast, Gray codes require an order of $2^{14} = 16384$ operations. (One update of a sum costs one operation at each step, generation of the Gray code costs at each step). This gives a nice class of examples where an exponential algorithm seems preferable to a quadratic algorithm.

Clearly the FFT on Z_{3N} will perform well when the z_i numbers are small. Pagano and Spino (1991) have shown how to adapt the Z_{3N} ideas to lightly trimmed means. Their approach is limited to examples of statistics that are close to linear and at present are not extensible to medians or general M-estimates.

In summary, for Fisher's randomization test and Hartigan's subsets, Gray coding is a useful procedure for samples up to size 30 or so. It works for general location estimates. For larger sample sizes on linear statistics the convolution approach is better.

3. The bootstrap

3.1. *Introduction*

The bootstrap is a widely used tool for non-parameteric procedures for bias correction, construction of confidence regions and estimation of sampling variability under minimal assumptions on the functional form of estimators or underlying distribution. Efron and Tibshirani (1993) give an accessible account with many examples and extensive references. Hall (1992) gives a theoretical development.

Bickel and Freedman (1981) carried out exact enumeration of the bootstrap distribution for the mean using the FFT, Bailey (1992) used a similar approach for simple functionals of means. Fisher and Hall (1991) suggest exact enumeration procedures that we will compare to the Gray code approach in Section 3.2. below.

Let $\mathscr{X}_n = \{x_1, x_2, \ldots, x_n\}$ be the original data supposed to be independent and identically distributed from an unknown distribution F on a space $\mathscr{X}$. The bootstrap proceeds by supposing that replacement of F by F_n, the empirical distribution, can provide insight on sampling variability problems.

In practice one proceeds by repeatedly choosing from the n points with replacement. This leads to bootstrap replications $\mathscr{X}_n^* = \{x_1^*, \ldots, x_n^*\}$. There are n^n such possible replications, however these are not all different and grouping together replications that generate the same subset we can characterize each resample by its weight vector $(k_1, k_2, \cdots, k_n)$ where k_i is the number of times x_i appears in the replication. Thus $k_1 + \cdots + k_n = n$.

Let the space of compositions of n into at most n parts be

$$C_n = \{\boldsymbol{k} = (k_1, \cdots, k_n), k_1 + \cdots + k_n = n, k_i \geq 0, k_i \text{ integer}\}. \tag{3.1}$$

Thus $|C_n| = \binom{2n-1}{n-1}$. We proceed by running through all compositions in a systematic way. Note that the uniform distribution on $\mathscr{X}_n^n$ induces a multinomial distribution on C_n

$$m_n(\boldsymbol{k}) = \frac{1}{n^n}\binom{n}{k_1 \cdots k_n}. \tag{3.2}$$

To form the exhaustive bootstrap distribution of a statistic $T(\mathscr{X}_n)$ one need only compute each of the $\binom{2n-1}{n-1}$ statistics and associate a weight $m_n(\boldsymbol{k})$ with it. The shift from $\mathscr{X}_n^n$ to C_n gives substantial savings. For the law school data, $n = 15, 15^{15} \approx 4.38 \times 10^{17}$ while $\binom{29}{14} \approx 7.7 \times 10^7$.

Efficient updating avoids multiplying such large numbers by factors of n. This is what makes the computation feasible. Gray codes generate compositions by changing two coordinates of the vector $\boldsymbol{k}$ by 1 up and 1 down. This means that $m_n(\boldsymbol{k})$ can be easily updated by multiplying and dividing by the new coordinates. Similar procedures, discussed in Section 3.3 below allow efficient changes in the statistics of interest.

3.3. *Gray codes for compositions*

Following earlier work by Nijenhuis, Wilf and Knuth, Klingsberg (1982) gave methods of generating Gray codes for compositions. We will discuss this construction briefly here, details can be found in Klingsberg (1982) and Wilf (1989).

For $n = 3$, the algorithm produces the 10 compositions of C_3 in the following order:

300, 210, 120, 030, 021, 111, 201, 102, 012, 003.

The easiest way to understand the generation of such a list is recursive, construction of the n-compositions of n can actually be done through that of the $(n-1)$ compositions of $(n-i), i = 1, \ldots, n$.

For any n, the 2-composition is just provided by the list $n0, (n-1)1, \ldots, 0n$, which is of length $n+1$. So the 2-compositions out of 3 are $L(n, n-1) = L(3,2) = 30, 21, 12, 03$;

the 2-compositions out of 2 are $L(n-1, n-1) = L(2,2) = 20, 11, 02$;

and 2-compositions out of 1 are $L(n-2, n-1) = L(1,2) = 10, 01$.

Finally there is only one 2-composition of 0: $L(0,2) = 00$.

The 3-out of 3 list is obtained by appending a 0 to the $L(3,2)$ list, a 1 to the $L(2,2)$ list, a 2 to the $L(1,2)$ list and a 3 to the $L(0,2)$ list. These four lists are then concatenated by writing the first, the second in reverse order, the third in its original order followed by the fourth in reverse order. This is actually written:

$$\mathscr{L}(3,3) = \mathscr{L}(3,2) \oplus 0, \quad \overline{\mathscr{L}(2,2) \oplus 1},$$
$$\mathscr{L}(1,2) \oplus 2, \quad \overline{\mathscr{L}(0,2) \oplus 3}$$

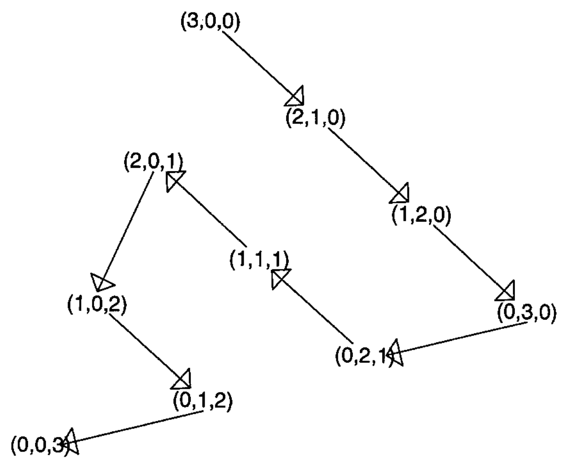

Fig. 3.1. *Gray code path on 3-simplex in perspective*

and more generally

$$\mathscr{L}(n,n) = \mathscr{L}(n,n-1) \oplus 0, \quad \overline{\mathscr{L}(n-1,n-1) \oplus 1}, \quad \mathscr{L}(n-2,n-1) \oplus 2, \cdots.$$

Figure 3.1 gives an idea of the hamiltonian path on the simplex that the method generates.

The same procedure leads to the 35 compositions of $n = 4$ in the following order:

4000, 3100, 2200, 1300, 0400, 0310, 1210,
2110, 3010, 2020, 1120, 0220, 0130, 1030,
0040, 0031, 0121, 1021, 2011, 1111, 0211,
0301, 1201, 2101, 3001, 2002, 1102, 0202,
0112, 1012, 0022, 0013, 0103, 1003, 0004.

The lists generated in this way have the property that two successive compositions differ only by ± 1 in two coordinates.

Klingsberg (1982) provides a simple nonrecursive algorithm that generates the successor of any composition in this Gray code. This is crucial for the implementation in the the present paper. It requires that one keep track of the whereabouts of the first two non-zero elements and an updating counter. Both an S and a C version of the algorithm are provided in the appendix.

We conclude this subsection by discussing a different algorithm due to Nijenhuis and Wilf (1978, pp. 40–46) which runs through the compositions in lexicographic order (reading from right to left). This algorithm was suggested for bootstrapping by Fisher and Hall (1991).

N–W algorithm to run through compositions C_n

```
1. Set $k_1 = n, k_i = 0, 2 \le i \le n$.
2. Let $h$ = first $i$ with $k_i \neq 0$. Set $t = k_h, k_h = 0, k_i = t - 1$,
   $k_{n+1} = k_{n+1} + 1$.
3. Stop when $k_n = n$.
```

For example, the following list gives all 35 compositions in C_4 in the order produced by the N–W algorithm:

4000, 3100, 2200, 1300, 0400, 3010, 2110,
1210, 0310, 2020, 1120, 0220, 1030, 0130,
0040, 3001, 2101, 1201, 0301, 2011, 1111,
0211, 1021, 0121, 0031, 2002, 1102, 0202,
1012, 0112, 0022, 1003, 0103, 0013, 0004.

The N–W algorithm (and so lexicographic order) changes at most three entries each time. Approximately half the time the first entry is non-zero. Then, two entries are changed, each by 1. On the other hand, there are some large changes. For example, following $(0, \cdots, n, 0)$ is $(n-1, 0, \cdots, 1)$. The following lemma shows that the average move in the N–W algorithm involves switching three things. It is useful to observe that the number of switches in the N–W algorithm is even. It is $2j$ for steps of form $(0, \cdots, 0, j, *, * \cdots, *) \to (j-1, 0, \cdots, 0, *, 1, *, \cdots, *)$ when $j \ge 2$.

Lemma On C_n of (2.1) let $P(2j)$ be the proportion of transitions in the N–W algorithm involving $2j$ switches, $1 \le j \le n$. Then, for fixed j and large n

$$P(2) \sim \frac{3}{4}, \qquad P(2j) \sim \frac{1}{2^{j+1}}. \tag{3.3}$$

Further,

$$\sum_{j=1}^{n} 2jP(2j) \sim 3. \tag{3.4}$$

Proof A step leads to two switches at the start if the step is $(k_1, k_2, \cdots)$ with $k_1 > 0$ or $(0, \cdots, 1, *, * \cdots)$. The proportion of steps with $k_1 > 0$ is

$$\frac{\binom{2n-1}{n-1} - \binom{2n-2}{n-1}}{\binom{2n-1}{n-1}} \to \frac{1}{2}.$$

The proportion of steps of form $(0, 0, \cdots, 1, *, *, \cdots)$ where the initial 1 is in position a, $1 \le a \le n-1$, is

$$\frac{\binom{2n-a-2}{n-2}}{\binom{2n-1}{n-1}} \to \frac{1}{2^{a+2}} \text{ for } a \text{ fixed, } n \text{ large.}$$

Summing in a shows that the proportion of such steps tends to 1/4 which gives $P(2) \sim 3/4$.

The argument for $P(2j) \sim 1/(2^{j+1})$ is similar. Finally

$$2 \cdot \frac{3}{4} + \sum_{j=2}^{\infty} (2j) \frac{1}{2^{j+1}} = 3.$$

That is, the expected number of switches between adjacent lexicographic compositions is 3, rather than a constant 2 linking Gray code compositions.

When the number of compositions is large these additional switches add appreciably to the computational burden. More importantly, the variable size of the lexicographic switching makes it complicated to take advantage of existing up-dating algorithms.

3.3. *Updating*

The Gray code algorithms avoid complete recomputation of a statistic because only one or two values need be changed. In the numerical analysis literature these are often referred to as rank-1 or rank-2 updates. In discussing this we find it useful to introduce a notion of gain as:

gain = (number of floating point operations required for complete recomputation)− (number of operations required for the updated statistic).

For example suppose $X_1, \cdots, X_n$ are real-valued and the statistic is $\bar{X} = (1/n)\{X_1 + X_2 + \cdots + X_n\}$. It takes $n-1$ operations to recompute if the X_i/n are stocked instead of X_i), and it takes 2 operations to delete an X_d/n and to add X_i/n. Thus the gain is $((n-1)-2) = n-3$.

The simplest case of updating is for functions of means (like the correlation coefficient for instance). Here X_i takes values in $\mathbb{R}^p$ and the statistic of interest is $f(X_1 + \cdots X_n)$. The gain is $p(n-3)$. We note that most of the theoretical justification for the bootstrap takes place in this arena so there are many examples.

For regression problems, the 'sweep' procedures in Dempster (1969) allow for up and downdating. A more recent study of rank-2 updating for Choleski decomposition is Bartels and Kaufman (1989). This allows gains for many multivariate methods that call for QR-decompositions, this includes discriminant analysis, multiresponse regression and parts of principal components analysis. For the latter another classical procedure is available, that is the rank-1 update of singular values and eigenvalues described in Bunch and Nielsen (1978) and Bunch *et al.* (1978) that was built on the original idea of Golub and Reinsch (1970). They show that a rank-1 update for a singular value decomposition can be obtained in order p^2 steps for a $(n \times p)$ data matrix. Arbenz and Golub (1988) and Elhay *et al.* (1989) give recent variations. Updating techniques are used in the cross-validation literature, see for example Eastment and Krzanowski (1982) who use Bunch *et al.* (1978).

The problem of updating the median can be shown to provide a gain of at least $n/2$, because if one uses the 'efficient' algorithm for searching for the median as described, p. 459 of Press *et al.* (1986) in their 'Numerical recipes', then the median can be seen as the solution of

$$x_{\text{med}} = \frac{\sum_{i=1}^{n} \frac{x_i}{|x_i - x_{\text{med}}|}}{\sum_{i=1}^{n} \frac{1}{|x_i - x_{\text{med}}|}}.$$

Half of the time the increasing and decreasing index are on the same side of $n/2$, for instance $x_i < x_{\text{med}}$ and $x_d < x_{\text{med}}$, replacement of x_d by x_i will not change the median and two comparisons are sufficient to see this.

Even when i and d are each side of $n/2$, the equation will only need one new passage to be done through the data to update the sums. Updating quantiles needs further study.

3.4. *Uses for exhaustive bootstrap distributions*

In modern developments of the bootstrap the original data are re-used in complex ways to get more accurate estimates and intervals and indeed to assess the variability of bootstrap intervals themselves. Techniques like double bootstrapping and jackknife-after-bootstrap are being actively explored. In principle, all of this information is contained in the exhaustive bootstrap. In the present section we spell this out in a simple case: the jackknife after bootstrap as developed in Efron (1992).

The object is to assess the sampling variability of a functional of the bootstrap distribution such as the length of a confidence interval. Efron used the jackknife idea, considering separately each of the histograms $H_i, 1 \leq i \leq n$, where H_i used bootstrap replications which were missing the ith of the original observations. The functional of interest can be calculated using the usual jackknife weights. Efron (1992, Section 3) contains details and examples.

Efron based his analysis on B bootstrap resamples ($B =$ 1000). The same ideas apply for the exhaustive bootstrap.

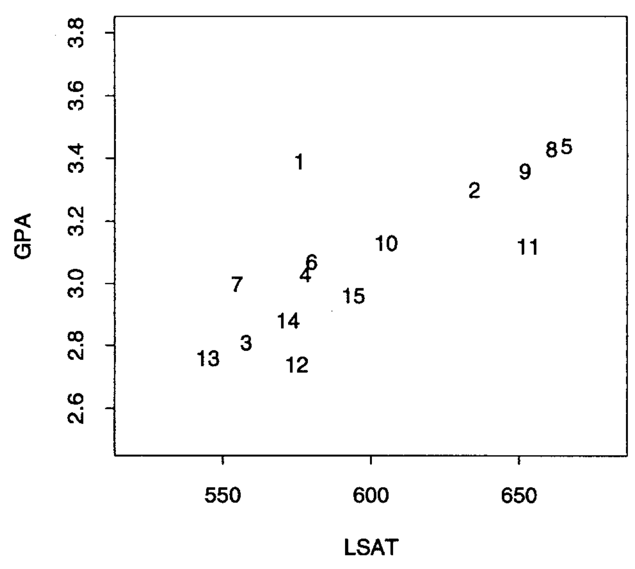

Fig. 4.1. *Scatterplot of law school data*

Here, while running through all compositions, one simply distributes the associated function value (and associated weight) into the n distribution being built up. Of course if a given composition is missing more than one value then it gets used once for each missing value.

4. Examples of the exhaustive bootstrap

4.1. *Correlation coefficient for the law-school data*

The 15 data points are as follows:

LSAT:	576	635	558	578	666	580	555	661
	652	605	653	575	545	572	594	
GPA:	3.39	3.30	2.81	3.03	3.44	3.07	3.00	3.43
		3.36	3.13	3.12	2.74	2.76	2.88	2.96

Observation 1 is a potential outlier. To investigate we carried out the exhaustive bootstrap without observation 1. Figure 4.3 shows that the mode is definitely larger (around 0.92) when this outlier is left out; so that the bump occurs because of the translation of the model class towards 1. We also investigated the impact of point 11. Figure 4.2 shows the exhaustive bootstrap without either the points 1 or 11. This gives further evidence that point 1 is responsible for the corner.

We carried out an exhaustive bootstrap analysis on the first 10 data points from the list above. Here, the effect of the first data point is enormous, creating a hole that is visible in both the exhaustive and the Monte Carlo bootstrap (Figs 4.5 and 4.6). A final remark that we have to make about the corner is that nothing corresponding to it is visible if we replace the correlation coefficient by its variance-stabilized Fisher transform ($z = \text{arctanh}(\rho)$), as suggested by Hall *et al.* (1989) (see Fig. 4.7).

4.2. *Bioequivalence data*

As described in Efron and Tibshirani (1993) or Efron (1992), these are data on 8 patients for which we have measurements on placebo, an approved treatment and a new drug patch.

The FDA bioequivalence requirement is that a 0.90 central confidence interval for

$$\rho = \frac{\mathcal{E}(\text{new} - \text{approved})}{\mathcal{E}(\text{approved} - \text{placebo})} \text{ to lie within the range } [-0.2, 0.2]$$

The data points are:

App−Pla:	8406	2342	8187	8459
	4795	3516	4796	10238
New−App:	−1200	2601	−2705	1982
	−1290	351	−638	−2719

The exhaustive bootstrap was computed for these data points: Figure 4.9 presents the histogram of the 6438 weighted ratios. Here the exhaustive picture is easier to look at but the conclusions drawn by Efron and Tibshirani (1993) do not change appreciably.

One can note here that the computation of the exhaustive bootstrap distribution is actually cheaper than doing an ordinary bootstrap with B = 4000 as in Efron and Tibshirani (1993).

This is due to the fact that here we have a function of two means: a complete computation of the statistic has a cost of around $2n - 1$ 'flops' (floating point operations), so for B = 4000 the total cost would be 60 000 flops, an update of the statistic costs 5 'flops', and that of the weights 2 flops, so an exhaustive bootstrap costs 45 066 flops in this case.

5. Other Gray codes

In this section we give a summary of other problems in statistics where Gray codes are available. We treat permutation tests, subgroups of the permutation group, and permutations of multisets (and so m out of n bootstrap replications). Finally, we give pointers to the available Gray code technology.

5.1. *Permutation tests*

Let S_n be the group of $n!$ permutations on n letters. Pitman (1937) first investigated the permutation justification of tests like the t-test. There are any number of modern variants. Among our favourites are Friedman and Rafsky's (1979) generalized measure of association: there one observes $(x_1, y_1), \cdots, (x_n, y_n)$ with x_i and y_j taking values in metric spaces X and Y. Friedman and Rafsky proposed building nearest-neighbour graphs such as the minimal spanning tree using the x_i in x space and again using the y_i values in y space. This gives two graphs on the index set $\{1, 2, \cdots, n\}$. Let the number of pairs $\{i,j\}$ which appear as an edge in both graphs. If T is large, points close in X tend to be close in Y. One can calibrate the observed value of T using the permutation distribution. There are many other variations: Witztum *et al.* (1994) give a fascinating application to finding patterns in the Bible!

Classically, Monte Carlo and asymptotics are used to approximate these distributions. There are also other approaches in the work of Pagano and his co-author for linear rank statistics and in the work of Mehta and Patil. For n small (e.g. $n \leq 10$ or 15) Gray codes are also a possibility. Nijenhuis and Wilf (1978) give the standard Gray code for S_n using transpositions. There are also many variants available, see for example Miller (1970), Chapter 3 of Wilf (1989) and the references to permutation generation in the bibliography of Good (1994).

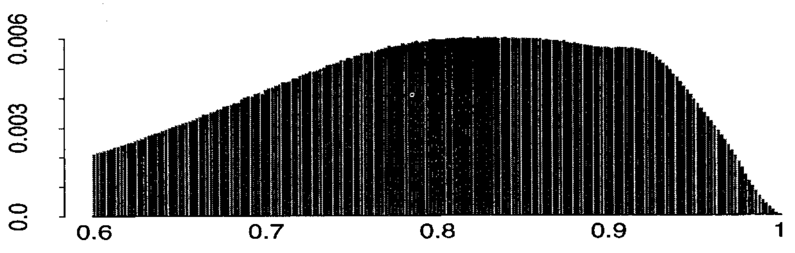

Fig. 4.2. *Exhaustive bootstrap with all n = 15 points (right tail)*

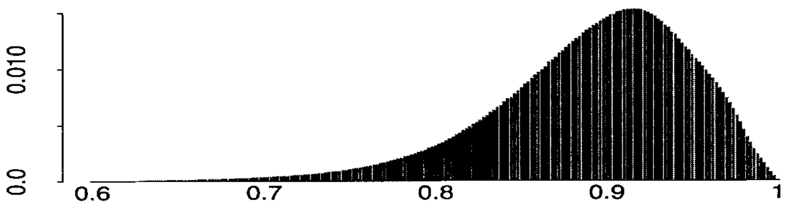

Fig. 4.3. *Exhaustive bootstrap without point 1 (right tail)*

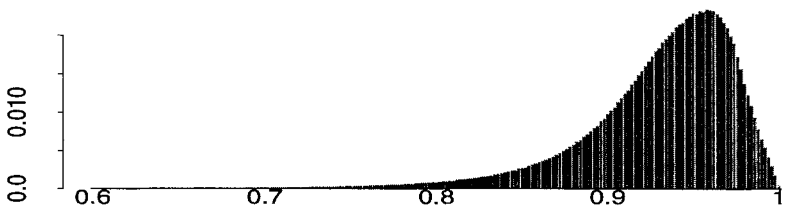

Fig. 4.4. *Exhaustive bootstrap without point 1 or 11 (right tail)*

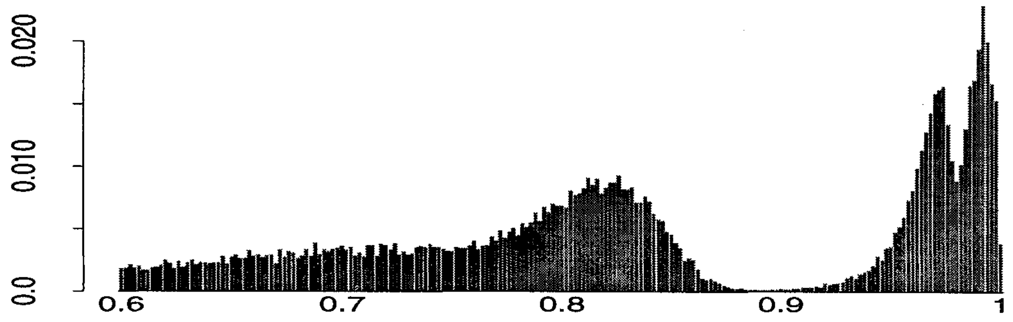

Fig. 4.5. *Histogram for exhaustive bootstrap: first 10 law school points*

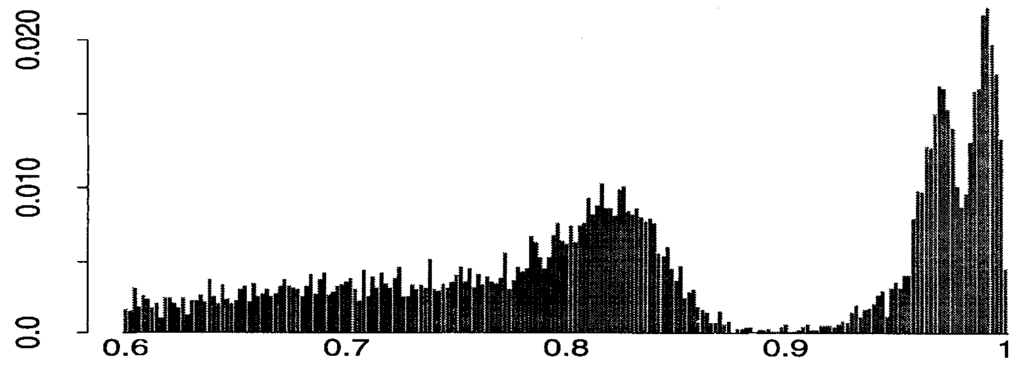

Fig. 4.6. *Histogram for Monte Carlo bootstrap: first 10 law school points (B = 10 000)*

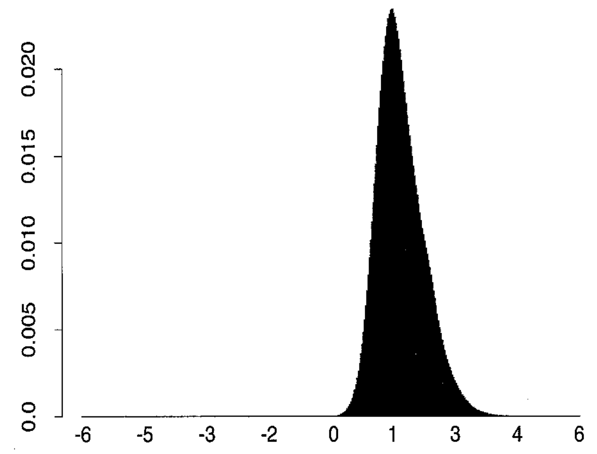

Fig. 4.7. *Histogram of arctanh* $(\hat{\rho})$

5.2. *Permutation groups*

Let G be a finite group and S a symmetric set of generators of G. Form a graph with vertex set G and an edge from t to t' if $t^{-1}t' \in S$. This is called the Cayley graph of (G, S). A *Hamiltonian tour* of (G, S) is a sequence $t_1, t_2, \cdots, t_{|G|}$, with no repeated values and $t_i^{-1}t_{i+1} \in S$ for each i. This gives a Gray coding of G with S making up the basic steps. As an example, if G is the symmetric group and S the set of transpositions, one gets the problem discussed in Section 5.1 above.

Gray codes for more general groups arise in several natural problems. Here are three examples: first, one has the sign change group $B_n = S_n \times \mathbb{Z}_2^n$. This is the group of $n \times n$ signed permutation matrices. Second, one has the symmetry groups of designed experiments such as analysis of variance. Third, one may consider subgroups of the symmetric group S_n as a way of getting a systematic set of points which are well distributed in S_n. These would be useful when n is too large to permit complete enumeration.

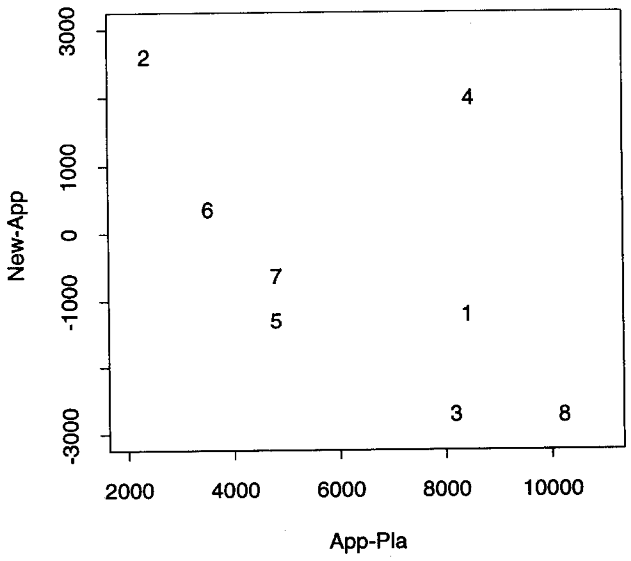

Fig. 4.8. *Scatterplot of bioequivalence data*

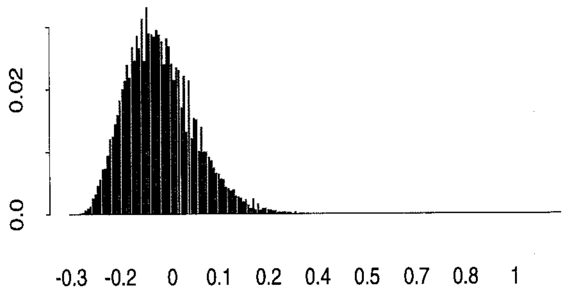

Fig. 4.9. *Exhaustive bootstrap: bioequivalence data*

One would hope to avoid the 'holes' of Monte Carlo sampling. As an example, the group $\mathrm{PSL}_3(F_p)$ of 3×3 matrices mod p, modulo its centre operates transitively and faithfully on the lines of the vector space, $\mathbb{Z}_p^3$. There are $n = p^2 + p$ such lines so one gets a subgroup of S_n of size $(p^3 - 1)(p^2 - 1)p^3/gcd(3, P - 1)$. The classical combinatorial design literature presents many further examples. Investigating how regularly such examples sit in the permutation group seems like a fascinating combinatorial problem.

For any of the examples above one may ask for a Gray code in a natural set of generators which hopefully do not involve too large a change in the permutation representation. While we have not actively tried this out, there is a literature on Hamiltonian paths for Cayley graphs. Entry may be had through Alspach and Godsil (1985) Conway et al (1989), and Ruskey (1994) and the references cited there.

5.3. *Permutations of multisets*

Let X be a multiset with k values labelled $1, k_2$ values labelled $2, \ldots, k_m$ values labelled m. Thus $|X| = k_1 + \cdots + k_m = n$. There are $\binom{n}{k_1 \ldots k_m}$ distinct arrangements of X. Such arrangements arise in analysis of variance (for a one-way layout). When $m = 2$, an arrangement of X may be regarded as subsets of size k_1 out of n. These arise in the basic two-sample test of Fisher (1936). They also arise from the m out of n bootstraps. See Politis and Romano (1992) for applications and history. Of course when $k_1 = n - 1, k_2 = n$ one has the usual bootstrap of Section 3.

Joan Miller (1970) has given an early Gray code for these cases using the set of transpositions. Ko and Ruskey (1992)

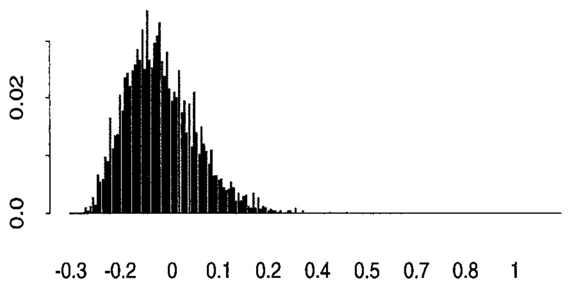

Fig. 4.10. *Monte Carlo bootstrap: bioequivalence data, B = 4000*

discuss this problem and give pointers to the computer science literature.

5.4. *General theory*

It is an unfortunate fact of life that while Gray codes usually exist, there is no general recipe for finding them. Indeed, the problem of finding Hamiltonian cycles in general graphs is NP complete. Nonetheless, with special neat cases or small *n*, one can often find suitable codes. The graph theory approach is usefully laid out in Nijenhuis and Wilf (1978). A survey of a rich variety of special cases can be found in Wilf and in the forthcoming monograph of Ruskey (1994). The cases where such Gray codes are useful in statistical problems involve small *n* and may be solved by brute force.

6. Conclusion

We have seen that a careful 'stepping through' all bootstrap or randomized samples allows for use of updating procedures and thus enables considerable computational gain; especially when the statistics are reused for correction procedures. However as sample sizes grow, the number of resamples or permutations *does* grow exponentially so another scheme has to be introduced to use updating while still studying the randomization distributions, the idea of only allowing randomness to seep in the changes in the sample through low rank updates leads to doing random walks on the resamples, this is studied in Diaconis and Holmes (1994b).

Acknowledgements

We thank Brad Efron for several inspiring conversations, Tom DiCiccio, Steve Stern and Michael Martin for helpful remarks, Frank Ruskey, Gene Golub and Ron Graham for bibliographical pointers and Rosemary Bailey Erich Lehmann, Cliff Lunneborg, Sandy Zabell and a referee for their remarks on the manuscript.

We particularly acknowledge extensive programming help from Brigitte Charnomordic without which nothing would have been possible.

Appendix 1: S functions for Gray codes

A.1 Gray codes for subsets

```
grays_function (n = 5)
{
#Generates the vector of elements whose signs are to change
#An example of a randomization test using this follows
        vec <- 0
        for (i in (1:(n+1))) {
                d <- i
                p <- 1
                while (!as.logical(d %% 2)) {
                        d <- d %/% 2
                        p <- p + 1
                }
                vec <- c(vec, p)
        }
        return(vec[-1])
}
```

A. 2 Randomization test

```
pair.gr_function(dif = abs(diffe), fn = sum, grayc)
{
#
               n <- length(dif)
               s <- rep(0, 2^n)
               if(missing(grayc))
                     grayc <- grays(2^(n - 1)- 1)
               signs <- rep(1, n)
               S[1] <- fn(dif)
               for(i in (2:(2^(n - 1)))) {
                       gc <- grayc[i]
                       signs[gc] <- ( - signs[gc])
                       S[i] <- S[i - 1] + 2 * signs[gc]
                                                     * dif[gc]
                       cat("S :", S[i], "\n")
               }
             S[((2^(n-1)) + 1):(2^n)] <- ( - S[1:(2^(n -1))])
             return(S)
     }
```

A.3 Gray codes for combinations

```
gray <- function(x = 0, n = sum(x), k = length(x),
                                              point = c(0, 0, 0))
{
########################
#x is the current pi
#point is the (d,i,p) vector of three pointers
########################
        if(is.list(x)) {
                point <- x$point
                x <- x$x
         }
         final <- F    ###################
# 1st tour
        if(sum(x) == 0) {
                x <- c(n, rep(0, k - 1))
                point <- c(1, 2, 1)
                return(x, point, final)
        }
# finished
       if(x[k] == n)
               stop("already last") #
       if(point[3] == 1) {
              b <- (1:k)[x > 0] [2] # position of second non
                                                        zero elt
              if(is.na(b)) b <- 2   # instead of 0
              if(b == 2) {
```

```
#careful : do not change d if d not equal to 1
                         if((point[1] == 1) && (x[1] == 1))
                                         point <- c(1, 2, 2)
        # don't forget i
                }
                else {
                         if((n - x[1]) %% 2) == 0)
                                point <- c(1, 2, 2)
                         else if((x[b] %% 2) ==1)
                                  point <- c(1, b, b)
                         else point[1:2] <- c(b, 1)
                }
# end of case b==2
        }
        else {
# end of case p==1
                if(((n - x[point[3]]) %% 2) == 1) {
                       point[1:2] <- c(point[3], point[3]-1)
                       if(x[point[3]] %% 2 == 0)
                              point[2] <- 1
                       point[3] <- point[2]
                 }
                 else {
                        if((x[point[3] + 1] %% 2) == 0) {
                                point[1:2] <- c(point[3],
                                                point[3] +1)
                                if(x[point[3]] == 1)
                                  point[3] <- point[3] + 1
                        }
                        else point [1:2] <- c(point[3] + 1,
                                                  point [3])
        }
      }
      x[point[2]] <- x[point[2]] + 1
      x[point[1]] <- x[point[1]] - 1
      if(x[1] > 0)
              point[3] <- 1
      if(x[k] == n)
         final <- T
      return(x, point, final)
}
callgray <- function(n = 5, k = 5, point = c(0, 0, 0), x = 0,
                                                pasapas = F)
{
         bi <- matrix(0, nrow = right(n), ncol = n)
         final <- F
         i <- 1
         while(!final) {
                 x <- gray(x, n, k, point)
                 bi[i, ] <- x$x
                 i <- i + 1
                 final <- x$final
                 if(pasapas) {
                        rep <- scan(n = 1)
                        if(length(rep) > 0)
                                break
                  }
         }
         cat(``finished...'')
         return(bi)
}
```

Appendix 2: C program for Gray codes for combinations

```
#cgray.cc
#include <stream.h>
#include <String.h>
#include <stdlib.h>
#include <math.h>

void initab(int *, int, int);
int cgray(int *, int *, int *, int *)
int pos2(int *, int);
int even(int);
void fixepoint(int *, int,int,int);
void affichetab(int *, int);
int sometab(int *, int);

// call of function cgray from Splus
int cgray(int *x, int *point, int *n, int *k)
{
// calls :
// pos2, fixepoint, affichage,initab, sommetab,even
//

         int fini,b;
         fini=0;
//first time round call with x=c(0,0,...0)
         if(sommetab(x,k)==0)
         {
                initab(x,k,0); //init of table to zero
                x[0]=n;
                //cout <<"\nfirst vector\t";
               affichetab(x,k);
               fixepoint(point,0,1,0); //fixes first (d,i,p)
               return(0);
         }
         if(x[k-1]==n)
                fini=1;
         if(point[2]==0)
         {
                b=pos2(x,k); //index of second non zero element
                if(b<=0)
                {
                        //cout <<"erreur";
                        return(-1);
                 }
                 if(b==1)
                 {      if((point[0]==0)&&(x[0]==1))
                                      fixepoint(point,0,1,1);
                  }
                  else
                  {
                        if(even(n-x[0])) fixepoint
                                           (point,0,1,1);
                        else if(!even(x[b])) fixepoint
                                           (point,0,b,b);
                                else fixepoint(point,b,0,
                                                  point[2]);
                 }
         } //end p==0
```

```
        else
        {
                if(!even(n-x[point[2]]))
                {
                        fixepoint(point,point[2],point[2]-1,
                                                point[2]);
                        if(even(x[point[2]])) point[1]=0;
                        point[2]=point[1];
                }
                else
                {
                        if(even(x[point[2]+1]))
                        {
                                fixepoint(point,point[2],
                                        point[2]+1,point[2]);
                                if(x[point[2]]==1) point[2]++;
                        }
                        else fixepoint(point,point[2]+1,
                                        point[2],point[2]);
                }
        }
// end of work
x[point[1]]++;
x[point[0]]--;
if(x[0]>0) point[2]=0;
if(x[k-1]==n)
        fini=1;
affichetab(x,k); //shows result
return(fini);
}
```

References

Albers, W., Bickel, P. J. and van Zwet, W. R. (1976) Asymptotic expansions for the power of distribution-free tests in the one-sample problem. *Annals of Statistics* **4**, 108–156.

Arbenz, P. and Golub, G. (1988) On the spectral decomposition of hermitian matrices modified by low rank perturbations with applications. *SIAM Journal of Matrix Analysis and its Applications* **9**, 40–58.

Bailey, W. A. (1992) *Exploring the Limits of the Bootstrap*, ed. R. LePage and L. Billard, pp. 309–318. Wiley, New York.

Barndorff-Nielsen, O. and Cox, D. (1989) *Asymptotic Methods in Statistics*. Chapman and Hall, London.

Basu, D. (1980) Randomization analysis of experimental data: the Fisher randomization test. *Journal of the American Statistical Association*, **75**, 575–581.

Bickel, P. J. and Freedman, D. A. (1981) Some asymptotic theory for the bootstrap. *Annals of Statistics* **9**, 1196–1217.

Bunch, J. and Nielsen, C. (1978) Updating the singular value decomposition. *Numerische Mathematik,* 111–129.

Bunch, J., Nielsen, C. and Sorensen, D. C. (1978) Rank one modification of the symmetric eigenvalue problem. *Numerische Mathematik*, **31**, 31–48.

Conway, N. H., Sloane, N. J. A. and Wilks, A. R. (1989) Gray codes for reflection groups. *Graphs and Combinatorics* **5**, 315–325.

Dempster, A. P. (1969) *Elements of Continuous Multivariate Analysis*. Addison-Wesley, Reading, MA.

Diaconis, P., Holmes, S., Janson, S., Lalley, S. and Pemantle R. (1994) Metrics on Compositions and Coincidences Among Renewal Sequences. To appear in the IMA: Monte Carlo Markov Chain Workshop.

Diaconis, P. and Holmes S. (1994) Three examples of Monte Carlo Markov Chains: at the interface between statistical computing, computer science and statistical mechanics. To appear in the IMA volume on Monte Carlo Markov chains.

Eastment, H. T. and Krzanowski, W. J. (1982) Cross-validatory choice of the number of components from a principal component analysis. *Technometrics*, **24**, 73–77.

Eaton, M. L. and Efron, B. (1970) Hotelling's T^2 test under symmetry conditions. *Journal of the American Statistical Association*, **65**, 702–711.

Efron, B. (1969) Student's *t*-test under symmetry conditions. *Journal of the American Statistical Association*, **64**, 1278–1302.

Efron, B. (1979) Bootstrap methods : another look at the jackknife. *Annals of Statistics*, **7**, 1–26.

Efron, B. (1982) *The Jackknife, the Bootstrap and other Resampling Plans*. CBMS-NSF. SIAM, Philadelphia.

Efron, B. (1992) Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society*, **54**, 83–127.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman and Hall, London.

Elhay, S., Golub, G. and Kautsky, J. (1989) *Updating and Downdating of Orthonomial polynomials with data fitting applications*. volume **70**, 149–172. Nato Adv. Sci. Ser. F : Comput. Systems Sci.

Elliott, D. and Rao, K. (1982) *Fast Fourier Transforms, Analyses, Applications*. Academic Press, New York.

Fisher, N. and Hall, P.G. (1991) Bootstrap algorithms for small samples. *Journal of Statistical Planning and Inference*, **27**, 157–169.

Fisher, R. A. (1935) *The Design of Experiments*. Oliver and Boyd, London.

Fisher, R. A. (1936) The coefficient of racial likeness and the future of craniometry. *Journal of the Royal Anthropological Institute*, **66**, 57–63.

Forsythe, A. and Hartigan, J. A. (1970) Efficiency of confidence intervals generated by repeated subsample calculations. *Biometrika*, **57**, 629–639.

Friedman, J. H. and Rafsky, L. C. (1979) Multivariate generalizations of the Wald-Wolfowitz and smirnov two-sample tests. *Annals of Statistics*, **7**, 697–717.

Golub, G. and Reinsch, C. (1970) Singular value decomposition and least squares solutions. *Numerische Mathematik*, **14**, 403–420.

Good, I. (1958) The interaction algorithm and practical Fourier analysis. *Journal of the Royal Statistical Society B*, **20**, 361–372.

Good, P. I. (1994) *Permutation Tests*. Springer-Verlag, New York.

Gordon, L. (1974a) Completely separating groups in subsampling. *Annals of Statistics* **2**, 572–578.

Gordon, L. (1974b) Efficiency in subsampling. *Annals of Statistics*, **2**, 739–750.

Graham, R., Hinkley, D., John, P. and Shi, S. (1990) Balanced design of bootstrap simulations. *Journal of the Royal Statistical Society*, **52**, 185–202.

Gray, F. (1939) Coding for data transmission. Technical report, *Bell System Technical Journal.*

Hall, P. (1992) *The Bootstrap and Edgeworth Expansions*. Springer-Verlag, New York.

Hall, P. and Martin, M. (1988) On bootstrap resampling and iteration. *Biometrika*, **75**, 661–671.

Hall, P. Martin, M. A. and Schucany, W. R. (1989) Better non-parametric bootstrap confidence intervals for the correlation coefficient. *Journal of Statistical Computing and Simulation*, **33**, 161–172.

Hartigan, J. (1969) Using subsample values as typical values. *Journal of the American Statistical Association*, **64**, 1303–1317.

Klingsberg, P. (1982) A gray code for compositions. *Journal of Algorithms*, **3**, 41–44.

Ko, C. and Ruskey, F. (1992) Generating permutations of a bag by interchanges. *Information Processing Letters*, **41**, 263–269.

Lehmann, E. (1975) *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, San Francisco.

Manly, B. (1991) *Randomization and Monte Carlo Methods in Biology*. Chapman and Hall, London.

Maritz, J. (1979) A note on exact robust confidence intervals for location. *Biometrika*, **66**, 163–166.

Maritz, J. (1981) *Distribution-Free Statistical Methods*. Chapman and Hall, London.

Miller, J. E. (1970) Transmission of analog signals over a gaussian channel by permutation modulation coding. PhD thesis, Columbia University.

Nijenhuis, A. and Wilf, H. S. (1978) *Combinatorial Algorithms for Computers and Calculators*. Academic Press, New York.

Pagano, M. and Spino, C. (1991) Efficient calculation of the permutation distribution of trimmed means. *Journal of the American Statistical Association*, **86**, 729–737.

Pagano, M. and Tritchler, D. (1983) On obtaining permutation distributions in polynomial time. *Journal of the American Statistical Association*, **78**, 435–440.

Pearson, E. (1937) Some aspects of the problem of randomization. *Biometrika*, **29**, 53–64.

Phua, P. and Chew, S. (1992) *Symmetric Rank-one Update and Quasi-newton methods*. World Scientific Publishing, River Edge, N.J.

Pitman, E. (1937) Significance tests which may be applied to samples from any population. *Journal of the Royal Statistical Society*, **4**, 119–130.

Politis, D. and Romano, J. (to appear) A general theory for large sample confidence regions based on subsamples under minimal assumptions. Technical report, *Annals of Statistics*.

Press, W., Flannery, B. P., Teulosky, S. and Vetterling, W. (1986) *Numerical Recipes*. Cambridge University Press.

Ruskey, F. (1994) *Combinatorial Generation*. Forthcoming.

Tritchler, D. (1984) On inverting permutation tests. *Journal of the American Statistical Association*, **79**, 200–207.

Wilf, H. S. (1989) *Combinatorial Algorithms: an Update*. SIAM, Philadelphia.

Witle, D. and Gallian, J. (1984) A survey; hamiltonian cycles in cayley graphs. *Discrete Mathematics* **51**, 293–304.

Witztum, D., Rips, E. and Rosenburg, Y. (1994) Equidistant letter sequences in the Book of Genesis. Unpublished.