3.1. COMPUTATION OF VARIANCES AND COVARIANCES

MOTIVATION

In chapter 1 : we have seen that calculation of a variance can raise problems (even with 3 observations equal to 1000001, 1000002, 1000003)
Single precision is not sufficient and a more refined algorithm proves to be necessary

ALGORITHM I (definition)

based on the definition : given a series of n observations, x_i, $i = 1, ..., n$, with mean $\bar{x}$:

$$s^2 = \frac{1}{n}\sum (x_i - \bar{x})^2 .$$

```
    S1 ← 0
    S2 ← 0
{Corps}
    Pour i = 1 à n
        Lire $x_i$
        S1 ← S1 + $x_i$
    finpour
    $\bar{x}$ ← S1/n
    Pour i = 1 à n
        Lire $x_i$
        S2 ← S2 + ($x_i$ - $\bar{x}$)^2
    finpour
{Clôture}
    $s^2$ ← S2/n
```

Advantage : simplicity and accuracy
Inconvenience : requires two passes over the data
which implies
either data storage in memory
or second reading of the data

ALGORITHM II (calculator)

This is the algorithm used in pocket calculators and paper and pencil
It is based on the alternative formula for the variance:

$$s^2 = \frac{1}{n}\sum x_i^2 - \bar{x}^2$$

```
    S1 ← 0
    S2 ← 0
{Corps}
    Pour i = 1 à n
        Lire $x_i$
        S1 ← S1 + $x_i$
        S2 ← S2 + $x_i$^2
    finpour
{Clôture}
    $\bar{x}$ ← S1/n
    $s^2$ ← S2/n - $\bar{x}$^2
```

Advantage : fast and requires only one pass (a requirement for calculators with a small memory space)
Inconvenience : often bad results even with double precision numbers
Example : 1000001, 1000002, 1000003:

$$\bar{x} = 1000002$$

$$s^2 = \frac{1}{n}\sum (x_i - \bar{x})^2 = \frac{(-1)^2 + (0)^2 + (1)^2}{3} = \frac{2}{3}$$

whereas

$$s^2 = \frac{(1000001)^2 + (1000002)^2 + (1000003)^2}{3} - (1000002)^2$$

Accuracy depends on the variation coefficient of the data set : $CV = s/\bar{x}$.
Indeed $S2 = ns^2 + n\bar{x}^2$ so dividing both sides by ns^2 gives $1 + (CV)^{-2}$.

If calculations are done with 7 digits and $CV^2 < 10^{-7}$, all the significant digits are lost
The smallest CV, the worst
Accuracy of Algorithm I doesn't depend on CV.
It is said that Algorithm I is numerically stable.
Another stable algorithm is as follows.

ALGORITHM III (definition with correction)
Like Algorithm I : requires 2 passes
Like Algorithm II : correction for the mean
But the mean is that of devations from the mean, equal to zero in principle
Consequence : better numerical stability

```
    S0 ← 0
    S1 ← 0
    S2 ← 0
{Corps}
    Pour i = 1 à n
        Lire $x_i$
        S0 ← S0 + $x_i$
    finpour
    $\bar{x}$ ← S0/n
    Pour i = 1 à n
        Lire $x_i$
        S1 ← S1 + ($x_i$ - $\bar{x}$)
        S2 ← S2 + ($x_i$ - $\bar{x}$)^2
    finpour
{Clôture}
    $s^2$ ← S2/n - (S1/N)^2
```

ALGORITHM IV (recurrence)

Also a compromise between algorithms I and II
All the avantages but no inconveniences.

```
    Lire $x_1$
    S1 ← $x_1$
    S2 ← 0
{Corps}
    Pour i = 2 à n  {Attention! 2 et non 1}
        Lire $x_i$
        S1 ← S1 + $x_i$
        S2 ← S2 + ((i*$x_i$ - S1)^2)/(i*(i - 1))
    finpour
{Clôture}
    $\bar{x}$ ← S1/n
    $s^2$ ← S2/n
```

Proof : by induction.
Correct for $n = 1$.
Suppose it is correct for $n = N$, and show that it is still correct for $n = N + 1$

Advantages :

Only one pass
No storage needed
Same accuracy as Algorithm I
Can even be used with simple precision (provided that the data have no more significant digits than simple precision)
Can be used with on line data
Can be generalised for weighted moments
Can be generalised to the covariance : using two accumulators S1x and S1y, say, and by replacing updating S2 par:

```
S2 ← S2 + ((i*x_i - S1x)*(i*y_i - S1y))/(i*(i - 1)).
```

Remark. Still another method : provisional mean μ_0 hence updating formulae

```
S1 ← S1 + x_i - μ0
S2 ← S2 + (x_i - μ0)^2
```

Cumulants

Remark. Questions of precision aside, let us mention the transition from moments with respect to the origin $m'_j = S_j / n$ and cumulants K_j

$$K_j = m'_j - \sum_{i=1}^{j-1} \binom{j-1}{i-1} m'_{j-i} K_i .$$

In particular

$$K_1 = m'_1 = \overline{x}$$

$$K_2 = m'_2 - m'_1 K_1 = s^2$$

The Fisher asymmetry coefficient = K_3/s^3 and the Fisher kurtosis coefficient = K_4/s^4.
Computation by recurrence available (Spicer [1972])

Use of MATLAB

Let x a matrix whose rows correspond to observations and columns to variables

cov(x)	covariance matrix
corrcoef(x)	correlation matrix
std(x)	vector of standard deviations (corrected for the number of degrees of freedom)

References

T. F. CHAN, G. H. GOLUB, and R. J. LEVEQUE, "Algorithms for computing the sample variance: analysis and recommendations", American Statistician 37, 242-247, 1983.
P. GRIFFITHS and I. D. HILL (editors), «Applied Statistics Algorithms», Ellis Horwood, Chichester, 1985.
HERRAMAN, Algorithm AS12, J. Roy. Statist. Soc. Ser. C Applied Statistics, 17, 1968, 289-292.
W. J. KENNEDY, Jr. and J. E. GENTLE, «Statistical Computing», Marcel Dekker, New York, 1980.
W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, and W. T. VETTERLING, «Numerical Recipes in FORTRAN 77: The Art of Scientific Computing», 2nd edition, Cambridge University Press, Cambridge, 1993.
C. C. SPICER, Algorithm AS52, J. Roy. Statist. Soc. Ser. C Applied Statistics, Reprinted in GRIFFITH and HILL, pp. 98-100.
R. A. THISTED, «Elements of Statistical Computing: Numerical Computation», Chapman and Hall, New York, 1988.

3.2. PROBABILITIES AND QUANTILES

We consider a univariate distribution (probability or density distribution function, empirical distribution)

REMINDERS

Some definitions (Patel *et al.* [1976])

(standard) normal distribution

or $N(0,1)$

$$F_{N(0,1)}(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2} dy.$$

bêta distribution

over [0;1], with parameters (a, b)

$$F_{\beta(a,b)}(x) = I_x(a,b) = \frac{1}{B(a,b)} \int_0^x y^{a-1}(1-y)^{b-1} dy$$

with $\quad B(a,b) = \int_0^1 y^{a-1}(1-y)^{b-1} dy$

and a, $b > 0$ and $0<x<1$.

Fisher-Snedecor *F* distribution

With (n_1, n_2) degrees of freedom

$$F_{F_{n_1,n_2}}(x) = 1 - I_{\frac{n_2}{n_2+n_1 x}}(\frac{n_2}{2}, \frac{n_1}{2})$$

Remark. Student *t* distribution is definied by $t_n^2 = F_{1,n}$.

Gamma distribution

Density :

$$f_{\text{gamma}}(x) = \frac{e^{-\theta x} x^{m-1} \theta^m}{\Gamma(m)} \qquad \text{with} \qquad \Gamma(m) = \int_0^\infty e^{-y} y^{m-1} dy.$$

where $x > 0$, $\theta > 0$, $m > 0$.
The chi squared distribution with *n* degrees of freedom is a special case of the gamma law, with parameters $\theta = 1/2$ and $m = 2n$.

binomial distribution

with exponent *n* and parameter *p*

$$p_x = P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, \ldots, n.$$

In the binomial coefficient $\binom{n}{x} = \frac{n!}{x!(n-x)!}$ and factorials $x! = \Gamma(x+1) = x.(x-1)...2.1$ very large numbers are generally involved.

EVALUATION OF PROBABILITES

The worst way to do the computations is to use the direct way (like nearly always). For example, for n = 13, $n!$ = 6227020800, exceeding simple precision
Recommended procedures.
1. Compute by taking logarithms.
2. Computation by recurrence :

$$p_x = \binom{n}{x} p^x (1-p)^{n-x} = \binom{n}{x-1} p^{x-1} (1-p)^{n-x+1} . \frac{n-x+1}{x} \frac{p}{1-p} = p_{x-1} . \frac{n-x+1}{x} \frac{p}{1-p}$$

(also the fastest but can loose accuracy by propagation of rounding errors)
3. Use Pascal triangle

$$\binom{n}{x} = \binom{n-1}{x-1} + \binom{n-1}{x} \text{ with } \binom{n}{0} = \binom{n}{n} = 1$$

keeping the coefficients on one line from right to left, while erasing (less appropriate than 2. for fixed n).

Rational approximations

Of a distribution function F :

$$F(x) = \frac{\sum_{j=0}^{k} a_j x^j}{\sum_{l=0}^{m} b_l x^l} + e(x)$$

with an error bounded by: $|e(x)| \leq 10^{-p}$

Remark. Recall Horner rule for evaluating a polynomial of degree n

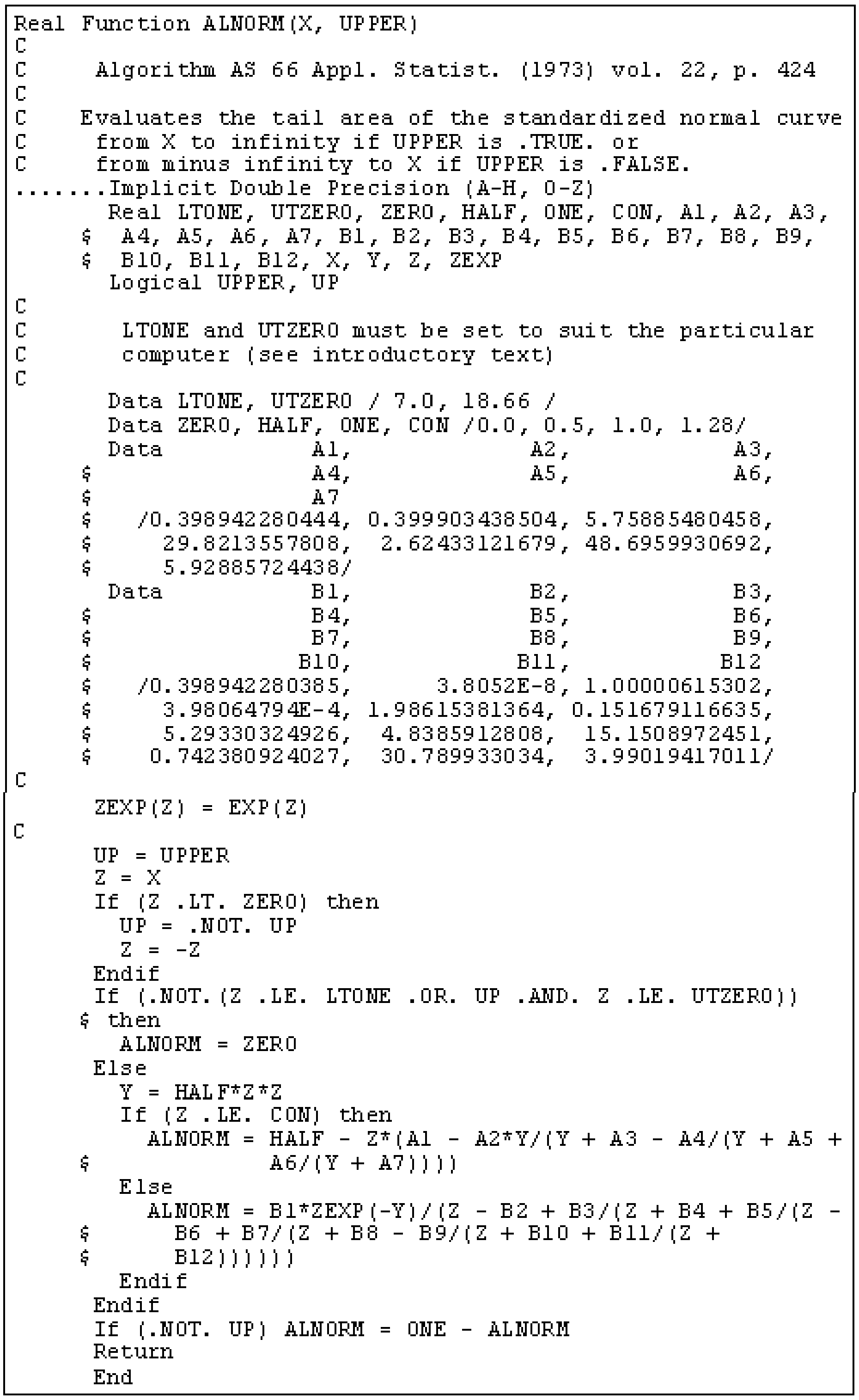
$$\sum_{j=0}^{n} c_j x^j = c_0 + x\{c_1 + x\{c_2 + x\{ \ldots\{ c_{n-1} + xc_n \}\}$$

Only n multiplications, whereas the direct method requires $2n - 1$ multiplications, if care is taken that $x^j = x.x^{j-1}$,
and even $1 + 2 + \ldots + n = (n + 1)n/2$ multiplications, otherwise.
There exists a Horner method for derivatives of polynomials.

Example. For evaluation of $\Phi(x)$, with $|e(x)| \leq 2{,}5.10^{-4}$

```
y ← |x|
v ← 1/((((0,019527y + 0,000344)y
          + 0,115194)y + 0,196854)y + 1)↑4
Si x > 0    alors Φ(x) ← 1 - v/2
            sinon Φ(x) ← v/2
finsi
```

Example. Algorithm AS66, Hill (1985), pour l'évaluation de $\Phi(x)$, with $|e(x)| \leq .10^{-9}$



```
Real Function ALNORM(X, UPPER)
C
C     Algorithm AS 66 Appl. Statist. (1973) vol. 22, p. 424
C
C    Evaluates the tail area of the standardized normal curve
C     from X to infinity if UPPER is .TRUE. or
C     from minus infinity to X if UPPER is .FALSE.
.......Implicit Double Precision (A-H, O-Z)
       Real LTONE, UTZERO, ZERO, HALF, ONE, CON, A1, A2, A3,
     $  A4, A5, A6, A7, B1, B2, B3, B4, B5, B6, B7, B8, B9,
     $  B10, B11, B12, X, Y, Z, ZEXP
       Logical UPPER, UP
C
C       LTONE and UTZERO must be set to suit the particular
C       computer (see introductory text)
C
       Data LTONE, UTZERO / 7.0, 18.66 /
       Data ZERO, HALF, ONE, CON /0.0, 0.5, 1.0, 1.28/
       Data           A1,             A2,             A3,
     $                A4,             A5,             A6,
     $                A7
     $   /0.398942280444, 0.399903438504, 5.75885480458,
     $     29.8213557808,  2.62433121679, 48.6959930692,
     $     5.92885724438/
       Data           B1,             B2,             B3,
     $                B4,             B5,             B6,
     $                B7,             B8,             B9,
     $               B10,            B11,            B12
     $   /0.398942280385,      3.8052E-8, 1.00000615302,
     $     3.98064794E-4, 1.98615381364, 0.151679116635,
     $     5.29330324926,  4.8385912808,  15.1508972451,
     $    0.742380924027,  30.789933034,  3.99019417011/
C
      ZEXP(Z) = EXP(Z)
C
      UP = UPPER
      Z = X
      If (Z .LT. ZERO) then
        UP = .NOT. UP
        Z = -Z
      Endif
      If (.NOT.(Z .LE. LTONE .OR. UP .AND. Z .LE. UTZERO))
     $ then
        ALNORM = ZERO
      Else
        Y = HALF*Z*Z
        If (Z .LE. CON) then
          ALNORM = HALF - Z*(A1 - A2*Y/(Y + A3 - A4/(Y + A5 +
     $             A6/(Y + A7))))
        Else
          ALNORM = B1*ZEXP(-Y)/(Z - B2 + B3/(Z + B4 + B5/(Z -
     $      B6 + B7/(Z + B8 - B9/(Z + B10 + B11/(Z +
     $      B12))))))
        Endif
      Endif
      If (.NOT. UP) ALNORM = ONE - ALNORM
      Return
      End
```

There exist methods for which $|e(x)| \leq 10^{-20}$. Note that $\Phi(x) = (1+\text{erf}(x/2^{1/2}))/2$ if $x \geq 0$, where erf is called the error function.

Approximation by continued fractions

$$\cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \ldots}}}}$$

Convergence criteria do exist. If convergence is valid, an approximant can be used. For example, the order 3 approximant is

$$v_3 = \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3}}} = \cfrac{a_1}{b_1 + \cfrac{a_2 b_3}{b_2 b_3 + a_3}} = \frac{a_1 (b_2 b_3 + a_3)}{b_1 (b_2 b_3 + a_3) + a_2 b_3} =_{def} \frac{A_3}{B_3}$$

The numerator and denominator of the order *k* approximant, respectively A_k et B_k can be obtained by the recurrence relations (due to Wallis in 1655):

$$A_k = b_k A_{k-1} + a_k A_{k-2}$$
$$B_k = b_k B_{k-1} + a_k B_{k-2}$$

with $A_{-1} = B_0 = 1$, $A_0 = B_{-1} = 0$.

For example:

$$A_1 = b_1 A_0 + a_1 A_{-1} = a_1$$
$$A_2 = b_2 A_1 + a_2 A_0 = a_1 b_2$$
$$A_3 = b_3 A_2 + a_3 A_1 = a_1 b_2 b_3 + a_1 a_3 \text{ (see above).}$$

Normal law : if $x > 0$

$$\Phi(x) = 1 - f(x) \cfrac{1}{x + \cfrac{1}{x + \cfrac{2}{x + \cfrac{3}{x + \ldots}}}}$$

The continued fraction is written

$$\frac{1}{x+}\frac{1}{x+}\frac{2}{x+}\frac{3}{x+}\ldots$$

Fraction continuée et fonction de distribution normale

x:	1,96
f(x):	0,058440944
Phi(x):	0,97500210
"VRAI" (Excel)	0,97500217

$$\boxed{\frac{1}{x+}\,\frac{1}{x+}\,\frac{2}{x+}\,\frac{3}{x+}\cdots}$$

$$\boxed{\begin{aligned} A_k &= b_k A_{k-1} + a_k A_{k-2} \\ B_k &= b_k B_{k-1} + a_k B_{k-2} \end{aligned}}$$

A-1 = B0 = 1, A0 = B-1 = 0

ak	bk	Ak	Bk	Ak/Bk	Phi(x)	Erreur
		1	0			
		0	1			
1	1,96	1	1,96	0,510204082	0,97018319	-0,00481898
1	1,96	1,96	4,8416	0,404824851	0,97634165	0,00133948
2	1,96	5,8416	13,409536	0,435630286	0,97454135	-0,00046082
3	1,96	17,329536	40,80749056	0,424665564	0,97518214	0,00017997
4	1,96	57,33229056	133,6208255	0,429067029	0,97492492	-0,00007726
5	1,96	199,0189695	465,9342708	0,427139582	0,97503756	0,00003538

But the approximation is bad when *x* is close to 0 (due to slow convergence)

x	F(x) fract cont.	F(x) Excel	Erreur
	0,97500210	0,97500217	-7,00E-08
-4,0	-0,00003167	0,00003169	-6,34E-05
-3,5	-0,00023263	0,00023267	-4,65E-04
-3,0	-0,00134990	0,00134997	-2,70E-03
-2,5	-0,00620967	0,00620968	-1,24E-02
-2,0	-0,02275013	0,02275006	-4,55E-02
-1,5	-0,06680720	0,06680723	-1,34E-01
-1,0	-0,15865525	0,15865526	-3,17E-01
-0,5	-0,30845553	0,30853753	-6,17E-01
-0,4	-0,34403997	0,34457830	-6,89E-01
-0,3	-0,37856501	0,38208864	-7,61E-01
-0,2	-0,39805470	0,42074031	-8,19E-01
-0,1	-0,32795694	0,46017210	-7,88E-01
0,0	0,50000000	0,50000000	2,18E-10
0,1	0,67204306	0,53982790	1,32E-01
0,2	0,60194530	0,57925969	2,27E-02
0,3	0,62143499	0,61791136	3,52E-03
0,4	0,65596003	0,65542170	5,38E-04
0,5	0,691544468	0,691462467	8,20007E-05
1,0	0,841344753	0,84134474	1,25092E-08
1,5	0,933192799	0,933192771	2,75251E-08
2,0	0,977249868	0,977249938	-6,9912E-08
2,5	0,993790335	0,99379032	1,4533E-08
3,0	0,998650102	0,998650033	6,91916E-08
3,5	0,999767371	0,999767327	4,42946E-08
4,0	0,999968329	0,999968314	1,47928E-08

Numerical integration

Notably the Newton-Cotes methods (including Simpson and Romberg).
Difficulty: when the variable is not bounded
Also Gauss quadrature related to families of orthogonal polynomials (Legendre, Laguerre, Hermite, etc.).

Taylor series expansion

And term by term integration
Example:

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \int_0^x e^{-y^2/2} dy \text{ and } e^{-y^2/2} = \sum_{n=0}^{\infty} \frac{(-1)^n y^{2n}}{2^n n!}$$

hence

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2^n (2n+1) n!}$$

has terms alternating in sign, producing roundoff errors. On the contrary the following series expansion has positive coefficients

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} e^{-x^2/2} g(x) \text{ where } g(x) = \sum_{n=0}^{\infty} c_n x^{2n+1} \text{ and } c_n = 1, \quad c_n = \frac{1}{2n+1} c_{n-1}$$

Ad hoc methods

Example. Simple algorithm for evaluating the probabilities of a chi squared distribution (Hill and Pike [1967], Abramowitz and Stegun [1964])

Based on the recurrence relation:

$$1 - F_{\chi_n}(x) = 1 - F_{\chi_{n-2}}(x) + \frac{(x/2)^{(n-2)/2}\exp(-x/2)}{\Gamma(n/2)}$$

with initial conditions

$$1 - F_{\chi_2}(x) = \exp(-x/2) \qquad 1 - F_{\chi_1}(x) = (2/\sqrt{2\pi})\int_{\sqrt{x}}^{\infty} \exp(-y^2/2)dy$$

Program :

```
      FUNCTION PROB (CHI2,N)
* taken from the ULB-VUB computer center library
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PROB=0.
      IF (N.LE.0) RETURN
      IF(CHI2.GT.FLOAT(100*N))RETURN
      IF (CHI2.LT.0.) RETURN
      EMYO2=EXP(-0.5*CHI2)
      SUM=1.
      TERM=1.
      M=N/2
      IF (2*M .NE.N) GO TO 1
C--                ENTRY IF N IS EVEN
      IF (M.EQ.1) GO TO 11
    4    DO 10 I=2,M
      FI=I-1
      TERM=0.5*TERM*CHI2/FI
   10 SUM=SUM+TERM
   11 PROB=EMYO2*SUM
      RETURN
C--                ENTRY IF N IS ODD
    1 SRTY=SQRT (CHI2)
      SRT2=SQRT(CHI2/2.)
C     VALUE=2.*(1.-FREQ (SRTY))
      VALUE=1.-ERF(SRT2)
*      VALUE= 1.0
      IF (N.NE.1) GO TO 2
      PROB=VALUE
      RETURN
    2 CONST=0.7978846*SRTY*EMYO2
      IF (N.EQ.3) GO TO 21
      K=M-1
         DO 20 I=1,K
      FI =I
      TERM=TERM*CHI2/(2.*FI+1.)
   20 SUM=SUM+TERM
   21 PROB=CONST*SUM+VALUE
  100 RETURN
      END
```

APPROXIMATED COMPUTATION OF QUANTILES

Let x_p be the order p quantile of the distribution function $F(x)$: the (or one) solution of equation $F(x) = p$, or $f(x) =_{\text{def}} F(x) - p = 0$.

Iteratives methods

Classical methods for solving an algebraic equation:

- methods which make use of derivatives
- methods which don't use them (but well the fact that the distribution function is monotone non decreasing).

The iterative methods require an initial value and a convergence criterion.
We consider iteration $(i + 1)$.

Method using a derivative

Based on a Taylor series expansion of $f(x)$,

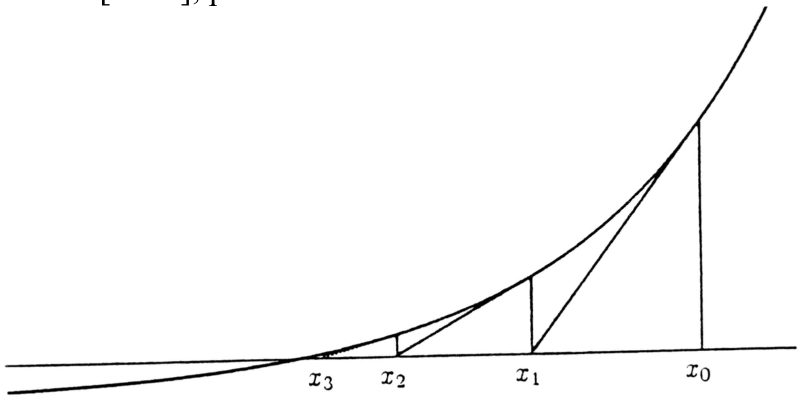
$$f(x) = f(x_i) + (x - x_i)\,f'(x_i) + \text{remaining}$$

Newton-Raphson algorithm

Point at iteration (i + 1): intersection of the tangent of $f(x)$ at point x_i with the x axis:

$$x_{i+1} = x_i - f(x_i)\,/\,f'(x_i)$$

Excerpt from Thisted [1988], p. 165.



Secant algorithm

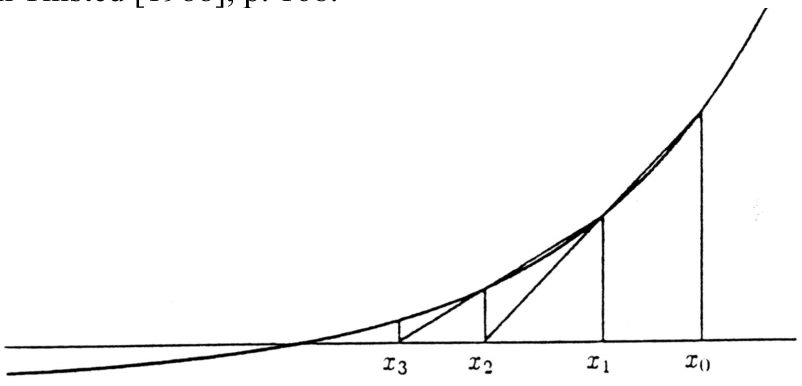
Point at iteration (i + 1): intersection with the x axis of the secant of $f(x)$ between the points with abscissea x_i and x_{i-1}:

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i)$$

where the fraction is an approximation of $1/f'(x_i)$.

Note. The method requires two initial values.

Excerpt from Thisted [1988], p. 168.



Bisection method

Suppose the points x_i and x_{i-1} are such that $f(x_i)\ f(x_{i-1}) < 0$ i. e.

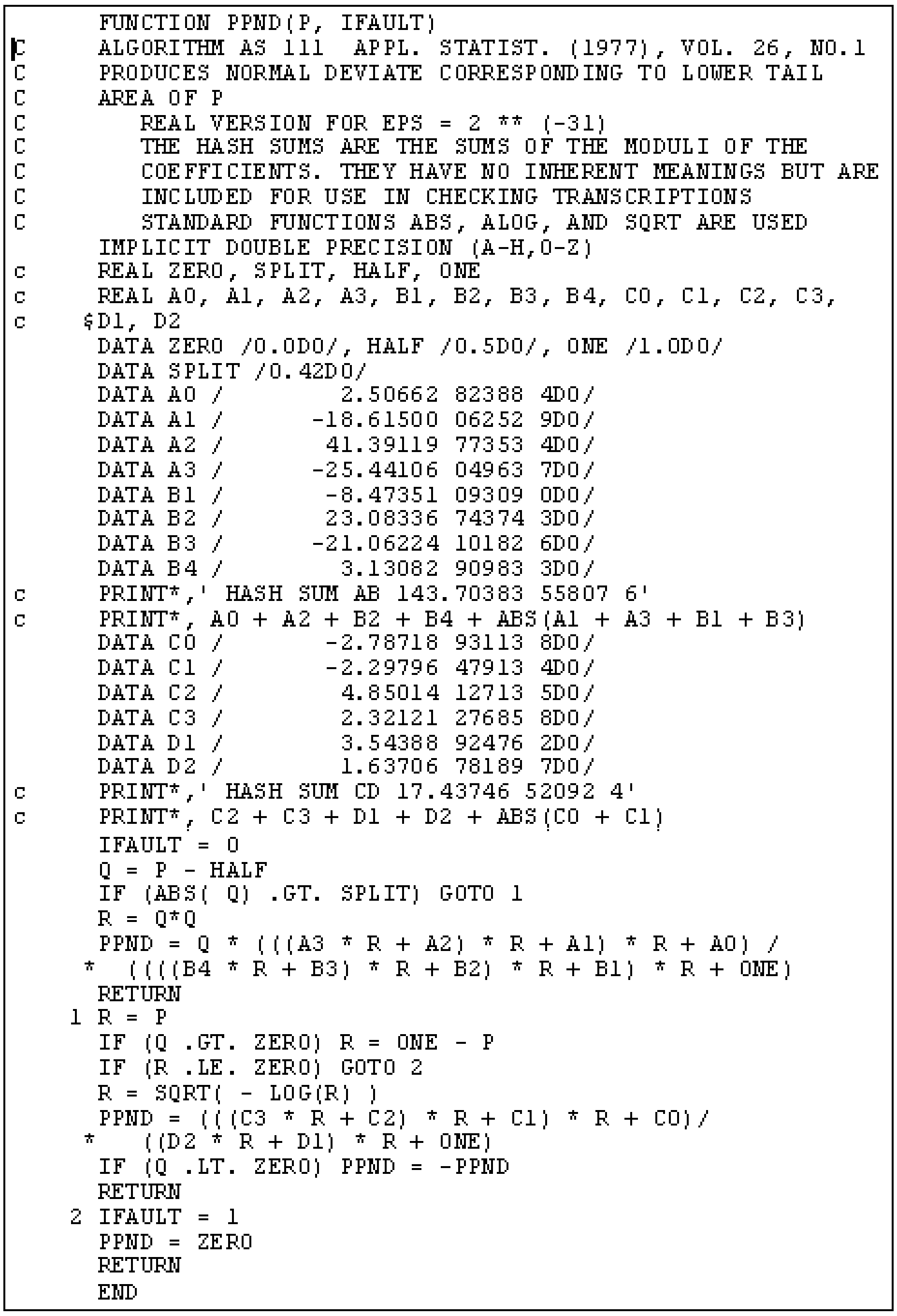
$$x_{i+1} = \frac{x_i + x_{i-1}}{2}.$$

The method works provided a bracket has been found.
Advantage: certain convergence by a factor of 2 at each iteration.

Ad hoc methods

Like for probabilities, there exist particular methods for some classical laws.
Let us consider the normal law.

Example. Algorithm of approximation by a rational fraction for quantiles of a normal law (Beasley and Springer [1985]). Precision: 7 digits if $|z| < 3{,}5$, 6 if $|z| = 4$, 5 if $|z| = 5{,}5$, and 4 si $|z| = 9{,}5$.



```
      FUNCTION PPND(P, IFAULT)
C     ALGORITHM AS 111  APPL. STATIST. (1977), VOL. 26, NO.1
C     PRODUCES NORMAL DEVIATE CORRESPONDING TO LOWER TAIL
C     AREA OF P
C        REAL VERSION FOR EPS = 2 ** (-31)
C        THE HASH SUMS ARE THE SUMS OF THE MODULI OF THE
C        COEFFICIENTS. THEY HAVE NO INHERENT MEANINGS BUT ARE
C        INCLUDED FOR USE IN CHECKING TRANSCRIPTIONS
C        STANDARD FUNCTIONS ABS, ALOG, AND SQRT ARE USED
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
c     REAL ZERO, SPLIT, HALF, ONE
c     REAL A0, A1, A2, A3, B1, B2, B3, B4, C0, C1, C2, C3,
c    $D1, D2
      DATA ZERO /0.0D0/, HALF /0.5D0/, ONE /1.0D0/
      DATA SPLIT /0.42D0/
      DATA A0 /        2.50662 82388 4D0/
      DATA A1 /      -18.61500 06252 9D0/
      DATA A2 /       41.39119 77353 4D0/
      DATA A3 /      -25.44106 04963 7D0/
      DATA B1 /       -8.47351 09309 0D0/
      DATA B2 /       23.08336 74374 3D0/
      DATA B3 /      -21.06224 10182 6D0/
      DATA B4 /        3.13082 90983 3D0/
c     PRINT*,' HASH SUM AB 143.70383 55807 6'
c     PRINT*, A0 + A2 + B2 + B4 + ABS(A1 + A3 + B1 + B3)
      DATA C0 /       -2.78718 93113 8D0/
      DATA C1 /       -2.29796 47913 4D0/
      DATA C2 /        4.85014 12713 5D0/
      DATA C3 /        2.32121 27685 8D0/
      DATA D1 /        3.54388 92476 2D0/
      DATA D2 /        1.63706 78189 7D0/
c     PRINT*,' HASH SUM CD 17.43746 52092 4'
c     PRINT*, C2 + C3 + D1 + D2 + ABS(C0 + C1)
      IFAULT = 0
      Q = P - HALF
      IF (ABS( Q) .GT. SPLIT) GOTO 1
      R = Q*Q
      PPND = Q * (((A3 * R + A2) * R + A1) * R + A0) /
     *  ((((B4 * R + B3) * R + B2) * R + B1) * R + ONE)
      RETURN
    1 R = P
      IF (Q .GT. ZERO) R = ONE - P
      IF (R .LE. ZERO) GOTO 2
      R = SQRT( - LOG(R) )
      PPND = (((C3 * R + C2) * R + C1) * R + C0)/
     *   ((D2 * R + D1) * R + ONE)
      IF (Q .LT. ZERO) PPND = -PPND
      RETURN
    2 IFAULT = 1
      PPND = ZERO
      RETURN
      END
```

Example. Simple algorithme for quantiles of a normal law (Wichura [1987], see Thisted [1988, p. 332])

```
v ← -2 log(p)
x ← log(2*π*v)
θ ← x/v + (2 - x)/v^2 + (-14 + 6x - x^2)/(2*v^3)
xp ← (v*(1 - θ))^(1/2)
```

with only a 2-digit precision for p < 0,1 and 4 digits for *p* < 0,025.

Summary of the practical methods

Distribution	Probabilities	Quantiles
normale	many (Taylor, continued fractions)	Rational approximations
Student *t*	several approaches	Several approximations
bêta	series, continued fractions	Numerical solving
Fisher *F*	series, rational functions	Numerical solving
chi carré	récurrence, fraction continuée	série de Taylor

FREQUENCIES OF AN EMPIRICAL DISTRIBUTION

A priori, no problem since this is a simple counting procedure.
But what for very large number of observations? For example: plot of distribution function, quantiles.
Not feasible to evaluate for each real number but at all point $x = i\ 10^{-d}$ where $d > 0$.
If finite support, then finite number of points. Otherwise, truncate the support.

Example. Correlation coefficient r known to be between -1 and 1.
Suppose an accuracy of 3 digits => $d = 3$.
We consider $i\ 10^{-3}$, $i = -999, ..., 999$
corresponding to an interval $[(i - 0{,}5)\ 10^{-3} ; (i + 0{,}5)\ 10^{-3}[$,
except for $i = -1000$ and $i = 1000$ (with intervals of width $0{,}5\ 10^{-3}$).
The following algorithm performs counting using 2001 counters (the sign(r) function gives +1 or -1, according to $r > 0$ or $r < 0$):

```
Tableau C(-1000:1000): numérique
Variable r: numérique
Effectuer CalculCorrélation(r)
i ← ⌊r*1000 + 0,5*signe(r)⌋
C(i) ← C(i) + 1
```

QUANTILES OF AN EMPIRICAL DISTRIBUTION

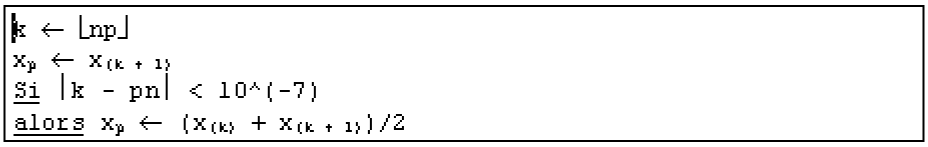
Let the data x_i, $i = 1, ..., n$,
and p, $0 < p < 1$, the order of the requested quantile.
Denote $x_{(i)}$ the *i*th observation ranked in increasing order.
We want to determine the order p quantile, $x_{(p)}$.

The distribution function is a step function. Let integer $k = \lfloor np \rfloor$.
Let us distinguish two cases:

$k/n < p < (k + 1)/n : x_p = x_{(k + 1)}$.

$p = k/n$. Several conventions (see procedure SAS Univariate),. e.g. the middle of the interval between $x_{(k)}$ and $x_{(k + 1)}$.

Algorithm:



```
$k \leftarrow \lfloor np \rfloor$
$x_p \leftarrow x_{(k + 1)}$
Si $|k - pn| < 10$^(-7)
alors $x_p \leftarrow (x_{(k)} + x_{(k + 1)})/2$
```

Sorting

Highest cost : sorting.
Simplest sorting algorithms require a number of operations of order $k.n^2$, where k is a constant.
Operations: comparisons (a =? b) and exchanges ($a \leftrightarrow b$, i. e. the three assignments, strictly in that order $c \leftarrow a$, $a \leftarrow b$, $b \leftarrow c$, where c is a temporary variable)
More complex algorithms (Heapsort, Quicksort) require a number of operations $k.n\log(n)$ (Knuth [1973]).

When any quantile can be requested, a sorting algorithm is faster

But when only a few quantiles are requested: better not to sort the whole data set but to carry out what is called a partition.

Partitioning

A partition compared to a pivot *X* consists in subdividing, by permutation, the observations in two subsets

- on the left, those which are smaller than X, and,
- on the right, those which are greater than X.

To determine the requested quantile: carry out a partition with respect to the ordered observation $x_{(k)}$ which is confused with that quantile.

One starts from an initial pivot (e.g. the *k*-st observation).

Then reiterate the partition on the left *X* if the position of *X* is greater than *k* and on the right of *X* if the position of *X* is smaller than *k*

When pivot *X* arrives in position *k*, the quantile has been found.

The algorithm is as follows:

```
$L \leftarrow 1$
$R \leftarrow n$
Répéter
    $X \leftarrow x_k$                                  {pivot}
    $i \leftarrow L$
    $j \leftarrow R$
    Itérer
        Tant que  $x_i < X$          {$\geq$ pivot à gauche}
        faire $i \leftarrow i + 1$
        fintantque
        Tant que  $x_j > X$          {$\leq$ pivot à droite}
        faire $j \leftarrow j - 1$
        fintantque
    sortir si    $i > j$
        $x_i \leftrightarrow x_j$                   {échanger}
        $i \leftarrow i + 1$
        $j \leftarrow j - 1$
    finitérer                      {fin partition}
    Si $j < k$ alors $L \leftarrow i$ finsi {plus à droite}
    Si $k < i$ alors $R \leftarrow j$ finsi {plus à gauche}
jusqu'à $L \geq R$
$x_{(k)} \leftarrow x_k$
```

Example

16	12	99	95	18	87	10

for determing $x_{(4)}$. Start from $X = x_4 = 95$.

16	12	99	95	18	87	10
L = 1		i = 3				j = R= 7

Exchange 99 and 10.

16	12	10	95	18	87	99
L = 1			i = 4		j = 6	R = 7

Exchange 95 and 87.

16	12	10	87	18	95	99
L = 1				i = j = 5		R = 7

Nothing to exchange.

16	12	10	87	18	95	99
L = 1				j = 5	i = 6	R = 7

Partition with respect to 95 finished but $95 = x_{(6)}$ too large. Look on the left. Pivot is X = 87.

16	12	10	87	18	95	99
L = 1			i = 4	j = R = 5		

Exchange 87 and 18

16	12	10	18	87	95	99
L = 1			j = 4	i = R = 5		

Partition with respect to 87 finished but $87 = x_{(5)}$ too large. Look on the left. Pivot is X = 18.

16	12	10	18	87	95	99
L = 1			R = 4			

Nothing to exchange. Partition with respect to $18 = x_{(4)}$ is finished. Algorithm is terminated .

Remark. The sorting algorithm called Quicksort is based on a recursive use of partitioning in each subsets of a partition at each stage.

Remark. The procedure given above is not practical when *n* is extremely large.

Start then from the empirical distribution of cumulated frequencies.

Let us continue the example related to the correlation coefficient *r* rounded up to 3 decimal digits, $r = i\ 10^{-3}$.

Example. Suppose the counts $C(i)$, i = -1000, ..., 1000. The algorithm is as follows.

```
Tableau C(-1000:1000): numérique
Variable r: numérique
i ← -1000
cum ← 0.0
Tant que cum < p
faire      cum ← cum + C(i)/n
           i ← i + 1
fintantque
$x_p$ ← (i - 1)/1000
```

Use of MATLAB

Law	Probabilities/ Density function		Cumulated probabilities/ Distribution function		quantiles	
	MATLAB	GKSLIB	MATLAB	GKSLIB	MATLAB	GKSLIB
Bêta	betapdf		betacdf		betainv	
Binomial	binopdf		binocdf	binp	binoinv	
chi square	chi2pdf		chi2cdf	chisqp	chi2inv	chisqq
Exponential	exppdf		expcdf		expinv	
Fisher *F*	fpdf		fcdf	fdistp	finv	fdistq
Gamma	gampdf	gammad	gamcdf		gaminv	
Géometric	geopdf		geocdf		geoinv	
Hypergeometric	hygepdf		hygecdf		hygeinv	
Normal	normpdf		normcdf	normp	norminv	normq
Poisson	poisspdf		poisscdf	poissp	poissinv	poisq
Student *t*	tpdf		tcdf		tinv	
Discrete uniform	unidpdf		unidcdf		unidinv	
Continuous uniform	unifpdf		unifcdf		unifinv	
Weibull	weibpdf		weibcdf		weibinv	

Let x a matrix whre rows correspond to observations and columns, to variables :

iqr(x)	interquartile interval
prctile(x,p)	order p/100 quantile

Also:

fzero('function', val)	root of function(x) close to val
median(x)	row vector with medians of the columns
sort(x)	sort in increasing order of each column
	f.ex. [y,i] = sort(x) sends in i the permutation such that y = x(i)

References

M. ABRAMOWITZ and I. A. STEGUN, «Handbook of Mathematical Functions», National Bureau of Standards Applied Mathematics Series, Number 55, U. S. Government Printing Office, Washington, 1964.
J. D. BEASLEY and S. G. SPRINGER, Algorithm AS111, J. Roy. Statist. Soc. Ser. C Applied Statistics, Reprinted in GRIFFITH and HILL, pp. 188-191.
P. GRIFFITHS and I. D. HILL (editors), «Applied Statistics Algorithms», Ellis Horwood, Chichester, 1985.
I. D. HILL, Algorithm AS66, J. Roy. Statist. Soc. Ser. C Applied Statistics, Reprinted in GRIFFITH and HILL, pp. 126-129.
I. D. HILL and A. C. PIKE, Algorithm 299: Chi-Squared Integral, Communications of the Association for Compututing Machinery 10, 243-244.
W. J. KENNEDY, Jr. and J. E. GENTLE, «Statistical Computing», Marcel Dekker, New York, 1980.
D. E. KNUTH, «The Art of Computer Programming, 3. Sorting and Searching». Addison-Wesley, Reading, 1973.
J. K. PATEL, C. H. KAPADIA and D. B. OWEN, «Handbook of statistical distributions», Marcel Dekker, New York, 1976.
W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, and W. T. VETTERLING, «Numerical Recipes in FORTRAN 77: The Art of Scientific Computing», 2nd edition, Cambridge University Press, Cambridge, 1993.
R. A. THISTED, «Elements of Statistical Computing: Numerical Computation», Chapman and Hall, New York, 1988.

3.3. GENERATION DE PSEUDO-RANDOM NUMBERS AND VARIABLES

GENERATORS OF PSEUDO-RANDOM NUMBERS

Pseudo-random numbers are used in various methods of simulation, for the study of complex stochastic models,

- in industry (management of a production chain),
- in the building industry (fitting of a car park)
- in telecommunications (developement of protocols of transmission or network architecture).

In statistics

- to compose a random sample in surveys
- for the randomization of an experimental designs
- for the study of methods of estimation and tests
  distribution of a complex statistic in the case of a finite sample
  precision of an estimator or the power of a test when the assumptions are not filled
- jacknife and bootstrap methods

A long time ago, random numbers come from lotteries or other games of chance
They were made available in tables (those of Rand Corporation are most famous)
Nowadays the needs are such as one must resort to a deterministic process to produce them : using algorithms.
Therefore the quality of the pseudo-random numbers which are produced must be investigated

One of the first methods: taking a number of 2 (originally 4) decimal digits, square it and take the 2 (4) central figures. The left-hand column of the following table shows how this process can be bad (accumulation in 0).

Génération de valeurs nulles	**Périodicité très courte**
$X_0 = 44$	a = 2, b = 3, m = 10, $X_0 = 0$
$(44)^2 = 1936 \rightarrow X_1 = 93$	$X_1 = (2 \times 0 + 3) \bmod 10 = 3$
$(93)^2 = 8649 \rightarrow X_2 = 64$	$X_2 = (2 \times 3 + 3) \bmod 10 = 9$
$(64)^2 = 4096 \rightarrow X_3 = 09$	$X_3 = (2 \times 9 + 3) \bmod 10 = 1$
$(09)^2 = 0081 \rightarrow X_4 = 08$	$X_4 = (2 \times 1 + 3) \bmod 10 = 5$
$(08)^2 = 0064 \rightarrow X_5 = 06$	$X_5 = (2 \times 5 + 3) \bmod 10 = 3$
$(06)^2 = 0036 \rightarrow X_6 = 03$	$X_6 = (2 \times 3 + 3) \bmod 10 = 9$
$(03)^2 = 0009 \rightarrow X_7 = 00$	$X_7 = (2 \times 9 + 3) \bmod 10 = 1$
$(00)^2 = 0000 \rightarrow X_8 = 00$	$X_8 = (2 \times 1 + 3) \bmod 10 = 5$

(example is drawn from U. W. POOCH and J. A. WALL, «Discrete Event Simulation: A Practical Approach», CRC Press, Boca Raton, p. 149 et p. 150.)

The most successful methods rest on an algorithm of linear congruence.
The algorithm depends on three parameters *a*, *b* et *m*.
One starts from a seed $X_0 = v(0)$.
At stage $(n + 1)$,

```
v(n + 1) ← (a v(n) + b) mod m
u(n + 1) ← v(n + 1)/m
```

The numbers $X_n = v(n)$ obtained are located between 0 and m - 1.
Consequently, the numbers $u(n)$ obtained are located between 0 and $(m - 1)/m$.
To exploit the binary character of the computer, $m = 2^k$ is taken.
We need to study the properties of the sequence of numbers $\{u(n)\}$.
The column on the right of the above table shows a bad choice of the parameters *a*, *b* and *m* (periodicity of period 4).

Génération de valeurs nulles	**Périodicité très courte**
$X_0 = 44$	a = 2, b = 3, m = 10, $X_0 = 0$
$(44)^2 = 1936 \rightarrow X_1 = 93$	$X_1 = (2 \times 0 + 3) \bmod 10 = 3$
$(93)^2 = 8649 \rightarrow X_2 = 64$	$X_2 = (2 \times 3 + 3) \bmod 10 = 9$
$(64)^2 = 4096 \rightarrow X_3 = 09$	$X_3 = (2 \times 9 + 3) \bmod 10 = 1$
$(09)^2 = 0081 \rightarrow X_4 = 08$	$X_4 = (2 \times 1 + 3) \bmod 10 = 5$
$(08)^2 = 0064 \rightarrow X_5 = 06$	$X_5 = (2 \times 5 + 3) \bmod 10 = 3$
$(06)^2 = 0036 \rightarrow X_6 = 03$	$X_6 = (2 \times 3 + 3) \bmod 10 = 9$
$(03)^2 = 0009 \rightarrow X_7 = 00$	$X_7 = (2 \times 9 + 3) \bmod 10 = 1$
$(00)^2 = 0000 \rightarrow X_8 = 00$	$X_8 = (2 \times 1 + 3) \bmod 10 = 5$

(example is drawn from U. W. POOCH and J. A. WALL, «Discrete Event Simulation: A Practical Approach», CRC Press, Boca Raton, p. 149 et p. 150.)

Choice of a generator

The choice of the parameters a, b and m is limited by theorems such as the following:

Theorem
If $m = 2^k$, b is odd and if $(a - 1)$ is a multiple of 4, then the period of the sequence $\{u(n)\}$, that is to say the smallest integer number k such that $u(n + k) = u(n)$ for all n, is m.

When results of simulation are reported, it is essential to describe the generator used completely, including the seed, so as to guarantee the reproductiveness of the results. Indeed, when one generates several sequences starting from the same seed, one finds the same sequence.
Conversely, to have a different sequence, another seed should be taken.

Portable generators

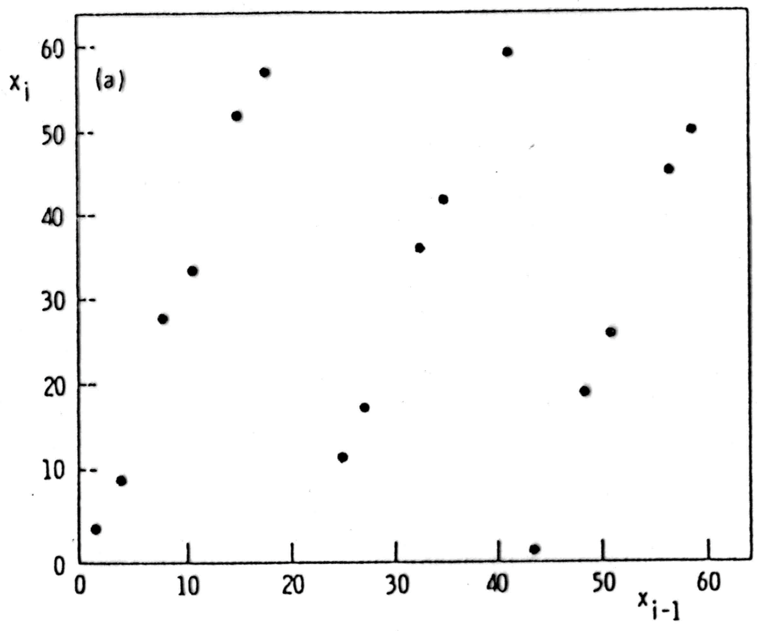
For reasons of effectiveness, the parameters of the generators are often selected according to the processor of the computer.
There are also portable generators, like that of Wichmann and Hill [1982] which employs a combination of three generators.

```
      FUNCTION RANDOM(L)
C
C        ALGORITHM AS 183   APPL. STATIST.
C        (1982) VOL. 31, NO. 2
C        RETURNS A PSEUDO-RANDOM NUMBER
C        RECTANGULARLY DISTRIBUTED
C        BETWEEN 0 AND 1.
C        IX, IY, AND IZ SHOULD BE SET TO
C        INTEGER VALUES BETWEEN 1 AND 30
C        1 AND 30000 BEFORE FIRST ENTRY
C   INTEGER ARITHMETIC UP TO 30323 IS REQUIRED
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RAND/ IX, IY, IZ
      IX = 171*MOD(IX, 177) - 2*(IX/177)
      IY = 172*MOD(IY, 176) - 35*(IY/176)
      IZ = 170*MOD(IZ, 178) - 63*(IZ/178)
C
      IF (IX .LT. 0) IX = IX + 30269
      IF (IY .LT. 0) IY = IY + 30307
      IF (IZ .LT. 0) IZ = IZ + 30323
C
      RANDOM = AMOD(FLOAT(IX) / 30269.0 +
     *         FLOAT(IY) / 30307.0 +
     *         FLOAT(IZ) / 30323.0, 1.0)
      RETURN
      END
```

Bad generators

Generators can be **very bad**.
We should never select a generator at random.
For example, here is the diagram $\{u(n), u(n+1)\}$ that can be obtained for a bad generator (a = 3, b = 0, m = 64, v(0) = 25):

Excerpt from Kennedy and Gentle [1980], p. 148.



Tests of random number generators

One can distinguish

- theoretical tests which use the procedure of congruence itself (in particular by deriving the autocorrelation function)
- empirical tests which consist in passing from long realisations of pseudo-random numbers in a statistical series of tests the purpose of which is to detect a structure or a nonuniform distribution.

GENERATORS OF PSEUDO-RANDOM VARIABLES

Up to now we considered only pseudo-random numbers distributed uniformly between 0 and 1.
In practice, one must frequently generate random samples according to laws of probability given or even realisations of a random process.
We consider here how to produce a realisation of a random variable whose distribution function is $F(x)$.
Several approaches are available but we will stress the first, conceptually simplest and often the fastest.

Inverse distribution function

Let us determine the quantile of order *u* where *u* is a realisation of a generator *U* of random numbers between 0 and 1.
This is based on the following theorem:

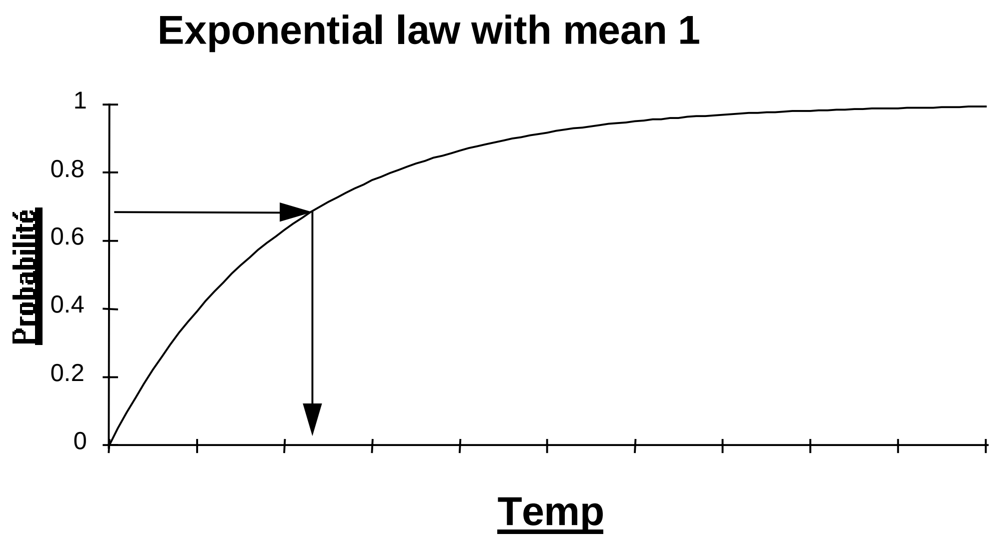
Theorem
If *U* is a random variable with a uniform distribution over [0; 1[, and if $F(x)$ is a continuous and increasing distribution function, then $X = F^{-1}(U)$ has a law with distribution function $F(x)$.

Consequently, if *u* is a pseudo-random number between 0 and 1, we take $x = F^{-1}(u)$.
The advantage of that method is that it provides correlated realisations when the same sequence of pseudo-random numbers are used for several laws.

Examples. For the normal law, we can use function PPND given above (Beasley et Spinger [1985]).
Two other symmetric law frequenly used in non-parametric statistics are the logistic law and the double exponential law (or Laplace law) with respective densities (Patel *et al.* [1976]):

$$f(x) = \frac{e^{-x/\sigma}}{\sigma(1+e^{-x/\sigma})^2} \qquad\qquad f(x) = \frac{1}{2\sigma} e^{-|x|/\sigma}.$$

Examples. Exponential law.



Ad-hoc methods

We first consider two methods for the standard normal law.
2.1 based on a version of the central limit theorem.

If the U_i are independent random variables with a uniform distribution over [0; 1]

$$X_{(n)} = \frac{\sum_{i=1}^{n} U_i - \frac{n}{2}}{\sqrt{n/12}} \underset{n\to\infty}{\overset{L}{\to}} N(0;1).$$

Note that the mean of the sum of the U_i is $n/2$ and its variance is $n/12$.
In particular, the choice of $n = 12$ is practical, because it suffices to subtract 6 from the sum of 12 pseudo-random numbers and the approximation of the normal law so obtained is not bad.

2.2 Box-Muller transformation based on the following theorem:

Theorem

If U_1 and U_2 are two independent variables with a uniform distribution over [0; 1],

then $X_1 = \sqrt{-2\log U_1}\cos(2pU_2)$ and $X_2 = \sqrt{-2\log U_1}\sin(2pU_2)$ are two independent variables with a normal *N*(0, 1) distribution.

Other ad-hoc procedures are deduced for distributions based on the normal distribution. For example, a chi square law with 4 degrees of freedom can be obtained from four values generated from a N(0; 1) using the formula

$c_4^2 = X_1^2 + X_2^2 + X_3^2 + X_4^2$.

Remark. There exist other methods with a less general appeal : acceptance-reject methods and sampling methods in arrays (for discrete laws).

GENERALISATIONS

Multivariate laws

A multivariate normal variable X with k dimensions, and covariance matrix $\Sigma = LL^T$, can be generated by starting with a vector Y of k independent normal variables with variance 1, and using the following theorem.

Theorem

If $\Sigma = LL^T$, then the covariance matrix of $X = LY$ is Σ.

Random processes

A time series $y_1, y_2, ...,$ can be generated according to an autoregressive process of order 1, with equation $Y_t = \phi\ Y_{t-1} + X_t$, where the X_t are independent random variables with mean 0 and variance σ^2.
Note that $\mathrm{var}(Y_t) = \sigma^2/(1 - \phi^2)$.
The realisations of X_t are obtained as above.
To start the recurrences, one could have supposed that $Y_0 = 0$, so that $\mathrm{var}(Y_1) = \sigma^2$. But then, a part of the series should be abandoned at the beginning in order to reduce the effect of that initial condition.
Instead of that, it can be prefered to generate y_1 from a law of variance $\sigma^2/(1 - \phi^2)$ and use the recurrence for $y_2, ...$

Sampling

Let us extract a random sample of size n in a finite population, where the individuals are indexed by i, $i = 1, ..., N$.
For a sample with replacement, one can consider n realisations u_k of a generator U of pseudo-random numbers between 0 and 1, and take individual $i_k = Nu_k + 1$.
For a sample without replacement, we should examine if the individual has not yet been selected. In the affirmative, that pseudo-random number should be rejected.
A faster procedure but which doesn't give exactly the requested size, consists in generating N realisations u_i of a generator U of pseudo-random de numbers between 0 and 1, and select individual i if $u_i < n/N$.
A more accurate procedure consists in (1) generate N realisations u_i of the same generator U, (2) sort them while keeping the indices i, (3) select the first n indices i.
The algorithm of McLeod and Bellhouse (1983) consists in taking the first n elements of the population, then, for $i = n + 1, \ldots, N$, we generate a realisation u_i of the same generator U and we compute $k \leftarrow 1 + \lfloor iu_i \rfloor$ and, if $k \leq n$, we replace the element k of the sample by the element i of the population.

Permutations

The following algorithm, due to Page (1967, with a correction) allows to generate a random permutation of the integer numbers 1 to *n*, ranged in a vector RANK.
Note: the version given by Kennedy and Gentle (1980) contains an error.

```
      SUBROUTINE PERMAL( RANK, N )
C-----GENERATES A RANDOM PERMUTATION OF THE FIRST N INTEGERS
C     IN RANK
C-----ALGORITHM OF PAGE (1967) (with a correction: the last
C-----  line before CONTINUE is missing in the paper)
C-----
      INTEGER RANK(N), TEMP, TRANS
      DO 40 NNEW = N , 2 , -1
        TEMP = RANK(NNEW)
        TRANS = INT((NNEW)*(RANDOM(L))) + 1
        RANK(NNEW) = RANK(TRANS)
        RANK(TRANS) = TEMP
   40 CONTINUE
      RETURN
      END
```

(the 4th line from the end is missing in Page (1967))
Generating all the permutations of a *n*uple of numbers, x_i, $i = 1, ..., n$ can be done using the following Fortran program (Langdon [1967]) run *n*! times.

```
      SUBROUTINE PERMUT( RANK, N )
C-----GENERATES ALL THE PERMUTATIONS OF THE FIRST N INTEGERS
C     IN RANK
C-----ALGORITHM OF LANGDON G. G., CACM 10, No 5, 1967, 298.
C-----PARTIALLY BASED ON AN IMPLEMENTATION OF R. DEVILLERS
      INTEGER RANK(N), TEMP
      DO 40 N1 = N, 2, -1
        TEMP = RANK(1)
        DO 20 I = 1, N1 - 1
          RANK(I) = RANK(I + 1)
   20   CONTINUE
        RANK(N1) = TEMP
      IF (RANK(N1) .NE. N1) RETURN
   40 CONTINUE
      END
```

Use of MATLAB

Pseudo random numbers over (0,1) can be obtained by means of function rand. Two variants can be used

rand(n)	for a matrix of size (n, n)
rand(m,n)	for a matrix of size (m,n).
rand('seed')	in order to get the current value of the seed of the generator
rand('seed', n)	in order to set the seed to value n.

Remark. The algorithm makes use of the linear congruence with
$\text{seed} = (7^7 * \text{seed}) \bmod (2^{31} - 1).$

It is also possible to generate realisations of a normal law with mean 0 and variance 1, using randn with the same use as rand.

The following functions of the statistical toolbox of MATLAB and the librairy GKSLIB (Gordon K. Smith, available on server StatLib; it is required to use rand in order to generate the arguments of the functions) are also related to this section

Law	quantiles	
	MATLAB	GKS
Bêta	betarnd	
Binomial	binornd	
chi square	chi2rnd	chisqq
Exponential	exprnd	
Fisher F	frnd	fdistq
Gamma	gamrnd	
Géométric	geornd	
Hypergeometric	hygernd	
Normal	normrnd	normq
Poisson	poissrnd	poisq
Student t	trnd	
Discrete uniform	unidrnd	
Continuous uniform	unifrnd	
Weibull	weibrnd	

A random permutation of intergers 1 to n can be produced by:

randperm(n)

References

P. BRATLEY, B. L. FOX, L. E. SCHRAGE, «A Guide to Simulation », Spinger-Verlag, New York, 1987 (2nd edition).
A. M. LAW and W. D. KELTON, «Simulation modeling and analysis», McGraw-Hill, New York, 1991 (2nd edition).
J. D. BEASLEY and S. G. SPRINGER, Algorithm AS111, J. Roy. Statist. Soc. Ser. C Applied Statistics, Reprinted in GRIFFITH and HILL, pp. 188-191.
J. E. GENTLE, «Random Number Generation and Monte Carlo Methods», Springer-Verlag, 1998.
P. GRIFFITHS and I. D. HILL (editors), «Applied Statistics Algorithms», Ellis Horwood, Chichester, 1985.
W. J. KENNEDY, Jr. and J. E. GENTLE, «Statistical Computing», Marcel Dekker, New York, 1980.
D. E. KNUTH, «The Art of Computer Programming, 2. Seminumerical Algorithms ». Addison-Wesley, Reading, 1969.
C. G. LANGDON, An algorithm for generating permutations, Communications of the Association for Compututing Machinery 10, 298-299.
A. I. McLEOD and D. R. BELLHOUSE, «A convenient algorithm for drawing a simple random sample», J. Roy. Statist. Soc. Ser. C Appl. Statist. 32, 1983, 182-184.
E. S. PAGE, A note on generating random permutations, J. Roy. Statist. Soc. Ser. C Appl. Statist. 16, 273-274.
J. K. PATEL, C. H. KAPADIA and D. B. OWEN, «Handbook of statistical distributions», Marcel Dekker, New York, 1976.
W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, and W. T. VETTERLING, «Numerical Recipes in FORTRAN 77: The Art of Scientific Computing», 2nd edition, Cambridge University Press, Cambridge, 1993.
R. A. THISTED, «Elements of Statistical Computing: Numerical Computation», Chapman and Hall, New York, 1988.
B. A. WICHMANN and I. D. HILL, An efficient and portable pseudo-random number generator, Appl. Statist. 31, 1982, 188-190 (correction: 33, 1984, 123).

3.4. MONTE CARLO METHOD

Example of the Lilliefors test

The Lilliefors test consists in the application of the Kolmogorov-Smirnov test for testing normality in the case of a normal law which is not specified (which means that the parameters of the normal law are estimated using the sample sample as the one used in the test).
Lilliefors [1967] has presented one of the first tables of critical values obtained using the Monte Carlo method.
The Kolmogorov-Smirnov test is a test of a null hypothesis related to the distribution function $F(x)$.
The null hypotheis which is tested is that $F(x) = F_0(x)$ where $F_0(x)$ is the distribution function of a *fully specified law*.
The statistic used in the test is the *supremum of the difference in absolute value* between the empirical distribution function $\hat{F}_n(x)$ and $F_0(x)$:

$$D_n = \sup_X \left|\hat{F}_n(x) - F_0(x)\right|$$

Excerpt from Law and Kelton [1991], p. 389.

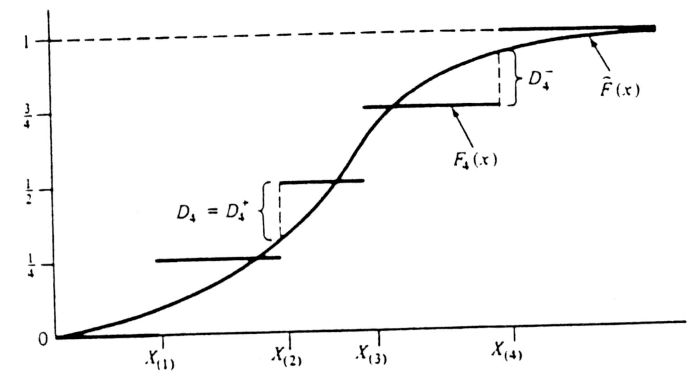


FIGURE 6.39
Geometric meaning of the K-S test statistic D_n for $n = 4$.

This expression is **not** equivalent to (Law et Kelton [1991])

$$\max_{i=1,\ldots,n} \left| \hat{F}_n(x_{(i)}) - F_0(x_{(i)}) \right|$$

but, taking care of continuity to the right of $\hat{F}_n(x)$, to

$$\max\left\{ \max_{i=1,\ldots,n} \hat{F}_n(x_{(i)}) - F_0(x_{(i)}); \max_{i=1,\ldots,n} F_0(x_{(i)}) - \hat{F}_n(x_{(i-1)}) \right\},$$

with $x_{(0)} = -\infty$.

Excerpt from Law and Kelton [1991], p. 389.

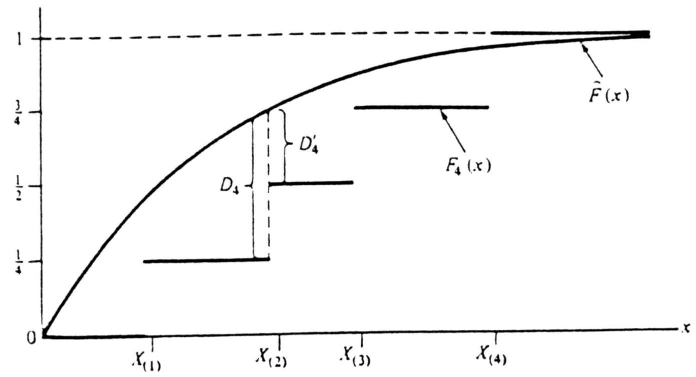


FIGURE 6.40
An example where the K-S test statistic D_n is not equal to D_n'.

The distribution of the statistic under the null hypothesis is known (although of a rather complicated form).
The Lilliefors test differs from the Kolmogorov-Smirnov test by the fact that the distribution function of $F_0(x)$ is that of a $N(m; \sigma)$ law where m and σ are estimated from the same sample of size n used for computing the test statistic.
It is therefore not entirely a priori specified. $F_0(x) = \Phi(\frac{x - \hat{m}}{\hat{\sigma}})$ is taken where $\hat{m}$ is the mean of the sample and $\hat{\sigma}^2$ is the unbiased estimator of the population variance.
The distribution under the null hypothesis is *not* the same and, as a matter of fact, differs even strongly asymptotically.
For example, the quantile of order 0,95 for large n is obtained by the formula $0{,}886/\sqrt{n}$ instead of by the formula $1{,}36/\sqrt{n}$.

It is possible (as Lilliefors did) to investigate the distribution under the null hypothesis by proceeding as follows.
Let us first remark that the test is invariant with respect to m and σ and that the result doesn't therefore depend on m and σ.
Hence, we can take simply $m = 0$ and $\sigma = 1$.
Denote by N the number of realisations of the sample of size n.
(We explain later how to choose N, although this is a similar problem to determining the size of a sample in statistics).
The algorithm is as follows:

```
Tableau Dn(N), x(n): numérique
Variable i, r, XBAR, VARI, quantile : numérique
Pour r = 1 à N
faire     Pour i = 1 à n
          faire      Effectuer GénérationNormale( x(i) )
          finpour
          Effectuer CalculMoyenneVariance(x, n, XBAR, VARI)
          Effectuer CalculDn( x, n, XBAR, VARI, Dn(r) )
finpour
Effectuer RechercheQuantile( Dn, quantile)
```

That algorithm makes use of the algorithms for computing the variance (see chapter 1) and for determining a quantile (see chapter 2, section 2).

Let us mention an increasingly useful presentation of critical values of tests in function, for example, of the sample size, in the form of modified statistics, obtained by a nonlinear regression.
So the critical values obtained by Lilliefors [1967] were supplemented by Stephens [1974] who provides the following table.

		$1 - \alpha$				
Case	Modified statistic	0,850	0,900	0,950	0,975	0,990
Kolm.-Smirn.	$(\sqrt{n} + 0{,}12 + 0{,}11/\sqrt{n})D_n$	1,138	1,224	1,358	1,480	1,628
Normal	$(\sqrt{n} - 0{,}01 + 0{,}85/\sqrt{n})D_n$	0,775	0,819	0,895	0,955	1,035

Other applications of the Monte Carlo method

We have seen the application of the Monte Carlo method to study the distribution of test statistics under the null hypothesis. Also :

- compare several tests under the null assumption (in particular when the quality of the approximation, for *n* large, by a classical law (normal, chi square) is to be evaluated
- when the distribution under the null assumption is unknown
- when one is interested in the exact distribution for small samples
- analyzing the level of the test when the underlying assumptions are not valid (normality, symmetry, independence of the observations).

However, it is mainly for the analysis of *power of tests* that the Monte Carlo method can prove to be most necessary.
Indeed, except for some very particular cases, the distribution of test statistics under the alternative assumption is *never* known.
One can simulate artificial observations in many situations of the alternative assumption, when the conditions for application are valid or even when they are not.

In all the cases which precede, the conclusions are valid only if a sufficient number of realisations are produced, this number being dependent on the desired precision.

Finally, artificial samples can be used to illustrate a procedure : an example. One can then limit oneself to only one realization or a very small number. The range of the conclusions obviously is very limited.

Statistical analysis of the results

The most interesting results are primarily as follows:

- empirical frequency of rejections of the hypothesis when the level of probability of the test is fixed and we
  - either want to analyze the quality of the theoretical distribution
  - or want to study the power of the test;
- critical values of the statistics when the level of probability of the test is fixed and we
  - either are interested in a table of the critical values
  - or want to deduce an approximate formula making it possible to find them (for use inside a program);
- obtaining an approximation of the whole empirical distribution making it possible to evaluate
  - the probability of significance
  - a confidence interval
- obtaining the parameters of the distribution of an estimator :
  - average, median
  - standard deviation (to be compared with the estimate possibly obtained by the theorem of Cramér- Rao)
  - coefficients of asymmetry and flatness
- goodness of fit test of the distribution of the test statistic or the estimator (normality, ...
- goodness of fit test compared to a uniform distribution on [0; 1] of the distribution of the probabilities of significance produced by the test.

Statistical methods can be used on the results of Monte-Carlo analysis like as with any experimental design.
It is thus desirable that the results of the study are in a form which lends itself to further analyses, in particular with an aim to investigate new properties not suspected at the beginning.
It is thus recommended to produce the results in the form of a data file which can be used in a statistical software package, rather than to calculate the synthetis results inside the simulation program.

Let us consider the choice of number N of realisations of the simulations. It must be based on the desired precision.
Example. Let us consider the case of the estimator of a parameter. The true value of the parameter is m. The simulations based on N realisations provide an average $\hat{m}$ with a standard deviation $\hat{\sigma}$. Provided that N is not too small, one can obviously determine a confidence interval with coverage probability of 95% for m in the form
$\hat{m} \pm 1{,}96\hat{\sigma}/\sqrt{N}$.
If we know an a priori approximation of $\hat{\sigma}$ and we set the desired precision, for example 0,5 10^{-3}, one can determine N by equating $1{,}96\hat{\sigma}/\sqrt{N} = 0{,}5.10^{-3}$.

Example. Let us consider the case of the quantile of order p, x_p, of a population of density function $f(x)$. For example, if one can call upon a normal asymptotic distribution as a first approximation, one can consider that $f(x)$ is a normal density. It is known (Cramér [1946]) that the error-type of the asymptotic distribution of this quantile of order p is

$$\frac{1}{f(x_p)}\sqrt{\frac{p(1-p)}{N}}$$

providing a possibility for determining N. Suppose that the statistic wre normalised so that it has mean 0 and standard deviation 1, that $p = 0{,}975$ and that the requested precision be $\delta = 0{,}005$. We obtain then x_p close to 2 and thus $f(x_p) \approx (2\pi)^{-1/2} e^{-2} \approx 0{,}054$ implying that

$$N = \frac{p(1-p)}{\delta^2 f^2(x_p)} \approx \frac{0{,}024}{0{,}25.10^{-4}.2{,}9.10^{-3}} = 3{,}3.10^6.$$

Planning sampling plans

Not only the methods of statistical analysis apply to the results of Monte Carlo simulations but experimental design can and even should be used to improve the precision of the results and to reduce the number of realisations.
When a Monte Carlo analysis is prepared, it is necessary to describe with precision the experimental design which will be followed and to adopt the same procedures (identification of the factors, randomization, etc.) that those adopted for traditional experiments (see Hoaglin and Andrews [1975]).

Variance reduction procedures

Several variance reduction procedures are suggested in order to reduce the size of the study with a constant precision for the results. The simplest of these procedures consist in using the same sequence of pseudo-random numbers in appropriate circomstances, as as follows.

Example. Two estimators of the same parameter β are proposed.
The question bears on a difference of bias.
Consider a first experiment where N_1 réalisations of the first estimation method give a mean equal to $\overline{\beta}^{(1)}$ and a standard deviation $\hat{\sigma}(\beta^{(1)})$, and a second experiment where N_2 realisations of the second estimation method have mean $\overline{\beta}^{(2)}$ and standard deviation $\hat{\sigma}(\beta^{(2)})$.
Consequently, provided that N_1 and N_2 are large enough, we have

$$\frac{\overline{\beta}^{(1)} - \overline{\beta}^{(2)}}{\sqrt{\frac{\hat{\sigma}^2(\beta^{(1)})}{N_1} + \frac{\hat{\sigma}^2(\beta^{(2)})}{N_2}}} \approx N(0;1).$$

With the same sequence of realisations of size $N = N_1 = N_2$, we compute the difference between the estimations of mean $\overline{\Delta\beta} = \overline{\beta}^{(1)} - \overline{\beta}^{(2)}$ and standard deviation $\hat{\sigma}(\beta^{(1)} - \beta^{(2)})$.
Consequently

$$\frac{\overline{\beta}^{(1)} - \overline{\beta}^{(2)}}{\hat{\sigma}(\beta^{(1)} - \beta^{(2)})/\sqrt{N}} \approx N(0;1).$$

But $\hat{\sigma}^2(\beta^{(1)} - \beta^{(2)}) = \hat{\sigma}^2(\beta^{(1)}) + \hat{\sigma}^2(\beta^{(2)}) - 2\,\hat{\text{cov}}(\beta^{(1)}, \beta^{(2)})$ and often $\hat{\text{cov}}(\beta^{(1)}, \beta^{(2)}) > 0$ so that it may be prefered to use the same sequence of pseudo-random numbers.
For example, if the correlation between two estimators is 0,75, the covariance equals 0,75 times the product of the standard deviations and, supposing those being equal, $\hat{\sigma}^2(\beta^{(1)} - \beta^{(2)})$ equals 2 - 2.0,75 = 0,5 time one of the variances instead of 2 times, hence a variance reduction with 4 as factor.
This can be translated into a number of realisations two times smaller.

Among the other procedures of variance reduction, let us mention the stratified sampling methods (well known within sampling methods). See also Snijders (1984), McGeoch (1992).

Use of MATLAB

Besides generation of pseudo-random numbers and variables in MATLAB (functions rand, nrand) and of its statistical toolbox,.MATLAB contains a reduced number of statistical functions which can be used in connection with the methods of this section:

max	Maximum
min	Minimum
cumsum	cumulated sum

Histograms can be obtained by the following procedures

hist(x)	histogram based on x, with 10 classes
hist(x, k)	histogram based on x, with k classes
hist(x, y)	histogram based on x, with classes based on y

The following functions produce absolute frequencies and class limits without making a plot:

[n,y] = hist(x)
[n,y] = hist(x, k)

The statistical toolbox offers:

anova1	analysis of variance with one factor
anova2	analysis of variance with two factors
ztest	test for the mean of a large sample
ttest	test for the mean of a small sample
ttest2	test of comparison of two means
boxplot	box plots
normplot	diagram for the Henri straight line
qqplot	quantile-quantile diagram

References

P. BRATLEY, B. L. FOX, L. E. SCHRAGE, «A Guide to Simulation », Spinger-Verlag, New York, 1987 (2nd edition).
A. M. LAW and W. D. KELTON, «Simulation modeling and analysis», McGraw-Hill, New York, 1991 (2nd edition).
H. CRAMER, «Mathematical Methods of Statistics», Princeton University Press, Princeton.
P. GRIFFITHS and I. D. HILL (editors), «Applied Statistics Algorithms», Ellis Horwood, Chichester, 1985.
G. FISHMAN, «Monte Carlo: Concepts, Algorithms and Applications», Springer-Verlag, 1996.
J. E. GENTLE, «Random Number Generation and Monte Carlo Methods», Springer-Verlag, 1998.
D. C. HOAGLIN and D. F. ANDREWS, The reporting of computation-based results in statistics, American Statistician 29, 1975, 122-126.
W. J. KENNEDY, Jr. and J. E. GENTLE, «Statistical Computing», Marcel Dekker, New York, 1980.
D. E. KNUTH, «The Art of Computer Programming, 2. Seminumerical Algorithms ». Addison-Wesley, Reading, 1969.
H. W. LILLIEFORS, On the Kolmogorov-Smirnov test for normality with mean and variance unknown, Journal of the American Statistical Association, 62, 1967, 399-402.
C. McGEOCH, "Analysing algorithms by simulation: variance reduction techniques and simulation speedups", ACM Computing Surveys 24, 195-212, 1992.
H. NIEDERREITER, P. HELLEKALEK, G. LARCHER, P. ZINTERHOF (Eds.) : (1998), Monte Carlo and Quasi-Monte Carlo Methods, Springer-Verlag, 1998.
H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, and W. T. VETTERLING, «Numerical Recipes in FORTRAN 77: The Art of Scientific Computing», 2nd edition, Cambridge University Press, Cambridge, 1993.
T. A. B. SNIJDERS, "Antithetic variates for Monte Carlo estimation of probabilities, Statistica Neerlandica 38, 55-73, 1984.
M. A. STEPHENS, EDF statistics for goodness of fit and some comparisons, Journal of the American Statistical Association, 69, 1974, 730-737.
R. A. THISTED, «Elements of Statistical Computing: Numerical Computation», Chapman and Hall, New York, 1988.