Université des Sciences et Technologies de Lille
U.F.R. de Mathématiques Pures et Appliquées
Bât. M2, F-59655 Villeneuve d'Ascq Cedex

Agrégation externe **Année 2000-2001**

Compléments de documentation Scilab : les fonctions de répartition (cdf)

1 Fonctions de répartition de gaussiennes

Voici ce que l'on peut lire dans l'aide en ligne de Scilab sur la fonction de répartition de la loi normale.

```
cdfnor(1)                    Scilab Function                    cdfnor(1)
NAME
  cdfnor - cumulative distribution function normal distribution

CALLING SEQUENCE
  [P,Q]=cdfnor("PQ",X,Mean,Std)
  [X]=cdfnor("X",Mean,Std,P,Q)
  [Mean]=cdfnor("Mean",Std,P,Q,X)
  [Std]=cdfnor("Std",P,Q,X,Mean)

PARAMETERS

  P,Q,X,Mean,Std
            : six real vectors of the same size.

  P,Q (Q=1-P)
            : The integral from -infinity to X of the normal density.
              Input range: (0,1].

  X         :Upper limit of integration of the normal-density.
             Input range: ( -infinity, +infinity)

  Mean      :  The mean of the normal density.  Input range: (-infinity,
            +infinity)

  Sd        :  Standard Deviation of the normal density.
            Input range: (0,+infinity).

DESCRIPTION
```

```
  Calculates any one parameter of the normal distribution given values for
  the others.

  A slightly modified version of ANORM from Cody, W.D. (1993). "ALGORITHM
  715: SPECFUN - A Portabel FORTRAN Package of Special Function Routines and
  Test Drivers" acm Transactions on Mathematical Software. 19, 22-32.  is
  used to calulate the  cumulative standard normal distribution.

  The rational functions from pages  90-95  of Kennedy and Gentle, Statisti-
  cal  Computing,  Marcel  Dekker, NY,  1980 are  used  as starting values to
  Newton's Iterations which compute the inverse standard normal.  Therefore
  no  searches  are necessary for  any parameter.

  For X < -15, the asymptotic expansion for the normal is used  as the start-
  ing value in finding the inverse standard normal.  This is formula 26.2.12
  of Abramowitz and Stegun.

  The normal density is proportional to exp( - 0.5 * (( X - MEAN)/SD)**2)...
```

Ces « explications » se décryptent comme suit. La première forme :

```
[P,Q]=cdfnor("PQ",X,Mean,Std)
```

correspond à la formule

$$P = \Phi_{m,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\Big(\frac{-(t-m)^2}{2\sigma^2}\Big)\,\mathrm{d}t, \quad Q = 1 - \Phi_{m,\sigma}(x),$$

où $\Phi_{m,\sigma}$ est la fonction de répartition de $N(m,\sigma)$ avec m = Mean et σ = Std (pour *standard deviation*, le nom anglais de l'écart-type). Il importe de respecter le premier paramètre "PQ" et de comprendre que les deux paramètres de sortie peuvent prendre n'importe quel nom : p, q, ou u, v, etc. Si on ne donne pas de nom aux deux paramètres de sortie, la fonction retourne seulement la valeur de $P = \Phi_{m,\sigma}(x)$.

```
-->[u,v]=cdfnor("PQ",1.2,0,1)
 v  =

    0.1150697
 u  =

    0.8849303
-->[u,v]=cdfnor("PQ",-1.2,0,1)
 v  =

    0.8849303
 u  =
```

```
    0.1150697
-->cdfnor("PQ",1.96,0,1)
 ans  =

    0.9750021
-->b=cdfnor("PQ",1.96,0,1)
 b  =

    0.9750021
```

Examinons la deuxième forme d'appel de `cdfnor` :

```
[X]=cdfnor("X",Mean,Std,P,Q)
```

La fonction retourne cette fois la valeur du quantile $x = \Phi^{-1}_{m,\sigma}(P)$ lorsque m, σ et P sont connus (et représentés respectivement dans la formule `Mean`, `Std` et `P`). La valeur du paramètre P doit être dans $]0,1[$ et il faut quand même fournir comme dernier paramètre la valeur $Q = 1 - P$. Par exemple, pour vérifier que $\Phi^{-1}_{0,1}(1/2) = 0$, il suffit de taper :

```
-->[X]=cdfnor("X",0,1,0.5,0.5)
 X  =

  - 5.405E-17
```

ce qui est bien le résultat attendu (n'est-il pas ?). Voici d'autres essais.

```
-->[X]=cdfnor("X",0,1,0.999,0.001)
 X  =

    3.0902323
-->[X]=cdfnor("X",0,1,0.001,0.999)
 X  =

  - 3.0902323
-->[X]=cdfnor("X",3,1,0.999,0.001)
 X  =

    6.0902323
-->[X]=cdfnor("X",3,2,0.999,0.001)
 X  =

    9.1804646
-->z=cdfnor("X",3,2,0.999,0.001)
 z  =
```

```
    9.1804646

-->(z-3)/2
 ans  =

    3.0902323
```

La troisième forme

```
[Mean]=cdfnor("Mean",Std,P,Q,X)
```

sert à calculer l'espérance inconnue $\mathbb{E}\,Z = m$ d'une variable aléatoire Z de loi $N(m, \sigma)$ lorsque σ est connue et Z telle que $\mathbf{P}(Z \leq x) = P$ avec x et P connus. Exemples :

```
-->[Mean]=cdfnor("Mean",2,0.999,0.001,9.1804646)
 Mean  =

    3.
-->m=cdfnor("Mean",1,0.975,0.025,1.96)
 m  =

    0.0000360
```

Est-il vraiment nécessaire de préciser que la quatrième forme

```
[Std]=cdfnor("Std",P,Q,X,Mean)
```

résout le problème analogue avec σ inconnu et m connu ? Signalons quand même le piège stupide : l'ordre des paramètres n'est pas celui de la troisième forme en remplaçant `Std` par `Mean` !

```
-->s=cdfnor("Std",0,0.975,0.025,1.96)
input parameter P is out of range
      bound exceeded: 0.000000
                                    !--error   999
s=cdfnor("Std",0,0.975,0.025,1.96)
-->s=cdfnor("Std",0.975,0.025,1.96,0)
 s  =

    1.0000184
```

Résumé : La fonction `cdfnor` a toujours 5 paramètres. Le premier est une chaîne de caractères indiquant celui des 5 qui sera retourné comme valeur de sortie en fonction des 4 autres. Les paramètres P et Q sont toujours liés par $Q = 1 - P$ et doivent être fournis tous les deux lorsqu'ils sont en entrée. La fonction s'applique aussi avec des paramètres vectoriels, pourvu qu'ils soient de même taille. On peut ainsi tracer le graphe de $\Phi_{m,\sigma}$, $1 - \Phi_{m,\sigma}$, $\Phi^{-1}_{m,\sigma}$... Attention, la moyenne m et l'écart type σ doivent être aussi sous forme vectorielle. Essayez

```
-->x=(-4:.01:4)';m=zeros(x);s=ones(x);

-->[p,q]=cdfnor("PQ",x,m,s);plot2d([x x],[p q])
```

2 Fonctions de répartitions de binomiales

Encore un coup d'oeil sur la documentation en ligne :

```
cdfbin(1)                        Scilab Function                  cdfbin(1)
NAME
  cdfbin - cumulative distribution function Binomial distribution

CALLING SEQUENCE
  [P,Q]=cdfbin("PQ",S,Xn,Pr,Ompr)
  [S]=cdfbin("S",Xn,Pr,Ompr,P,Q)
  [Xn]=cdfbin("Xn",Pr,Ompr,P,Q,S)
  [Pr,Ompr]=cdfbin("PrOmpr",P,Q,S,Xn)

PARAMETERS

  P,Q,S,Xn,Pr,Ompr
          : six real vectors of the same size.

  P,Q (Q=1-P)
          : The cumulation from 0 to S of the binomial distribution.
            (Probablility of S or fewer successes in XN trials each with
            probability of success PR.) Input range: [0,1].

  S       : The number of successes observed. Input range: [0, XN] Search
            range: [0, XN]

  Xn      : The number of binomial trials.  Input range: (0, +infinity).
            Search range: [1E-300, 1E300]

  Pr,Ompr (Ompr=1-Pr)
          : The probability of success in each binomial trial.  Input
            range: [0,1].  Search range: [0,1]

DESCRIPTION
  Calculates any one parameter of the binomial distribution given values
  for the others.

  Formula  26.5.24    of   Abramowitz  and  Stegun,  Handbook  of Mathe-
  matical  Functions (1966) is used to reduce the  binomial distribution
```

```
  to  the  cumulative incomplete   beta distribution.

  Computation of other parameters involve a seach for a value that
  produces the desired  value  of P.  The search relies  on  the
  monotinicity of P with the other parameter....
```

Forts de notre expérience de la doc Scilab pour les f.d.r. gaussiennes, on se dit que ce texte est parfaitement limpide. La première forme

```
[P,Q]=cdfbin("PQ",S,Xn,Pr,Ompr)
```

correspond au calcul de $P = \mathbf{P}(Z \leq x)$ et $Q = \mathbf{P}(Z > x)$, où la variable aléatoire Z suit la loi binomiale $\mathrm{Bin}(n, p)$ avec $x =$ `S`, $n =$ `Xn`, $p =$ `Pr`, $q = 1 - p =$ `Ompr`. Par acquis de conscience, on vérifie avec $n = 4$ et $p = 1/2$. D'abord « à la main »

```
-->x=0:4;y=cumsum([1 4 6 4 1]./16);[x;y]
 ans  =

!   0.        1.        2.        3.        4. !
!   0.0625    0.3125    0.6875    0.9375    1. !
```

puis avec cdfbin (avec paramètres vectoriels) :

```
-->n=4.*ones(x);p=0.5.*ones(x);q=1-p;[y,z]=cdfbin("PQ",x,n,p,q);

-->[x;y;z]
 ans  =

!   0.        1.        2.        3.        4. !
!   0.0625    0.3125    0.6875    0.9375    1. !
!   0.9375    0.6875    0.3125    0.0625    0. !
```

Visiblement tout va bien. Par curiosité on essaie une valeur non entière pour le paramètre $x =$ `S`. On s'attend soit à un message d'erreur, soit à trouver la même valeur que pour $[x]$.

```
-->cdfbin("PQ",2,4,0.5,0.5)
 ans  =

    0.6875
-->cdfbin("PQ",2.1,4,0.5,0.5)
 ans  =

    0.7216601
```

Raté ! Passée la première déception, on se dit qu'« ils » ont du prendre la fonction de répartition polygonale qui interpole linéairement entre les points d'abscisse entière au lieu de la vraie f.d.r. qui est en escaliers.

```
-->cdfbin("PQ",2.5,4,0.5,0.5)
 ans  =

    0.8395305

-->cdfbin("PQ",2.5,4,0.5,0.5)== (0.6875+0.9375)/2
 ans  =

  F
```

Encore raté ! Pour en avoir le coeur net, on tape

```
-->x=0:.1:4;n=4*ones(x);p=0.5*ones(x);q=1-p;

-->y=cdfbin("PQ",x,n,p,q);plot(x,y)
```

et on obtient une magnifique courbe bien lisse au lieu d'une ligne polygonale. Si on veut tracer la représentation graphique de la vraie f.d.r., il faut taper (par exemple)

```
-->x=0:4;n=4*ones(x);p=0.5*ones(x);q=1-p;y=cdfbin("PQ",x,n,p,q);
-->plot2d2("gnn",x',y')
```

Ce n'est pas encore tout à fait celà, car on ne peut prendre de valeur de x en dehors de $[0, n]$:

```
-->x=0:5;n=4*ones(x);p=0.5*ones(x);q=1-p;y=cdfbin("PQ",x,n,p,q); input
parameter Pr is out of range bound exceeded: 4.000000
-->x=-1:4;n=4*ones(x);p=0.5*ones(x);q=1-p;y=cdfbin("PQ",x,n,p,q);
input parameter Pr is out of range bound exceeded: 0.000000
```

Noter au passage que le message d'erreur semble confondre le paramètre `S` avec `Pr`. Finalement, pour avoir la représentation graphique correcte, il suffit de faire :

```
-->x=0:4;n=4*ones(x);p=0.5*ones(x);q=1-p;y=cdfbin("PQ",x,n,p,q);

-->x=[-1 x 5];y=[0 y 1];plot2d2("gnn",x',y')
```

Passons au calcul des quantiles. On prépare le terrain avec :

```
-->k=0:4;n=4*ones(k);p=0.3*ones(k);q=1-p
 q  =

!   0.7    0.7    0.7    0.7    0.7 !

-->[F,G]=cdfbin("PQ",k,n,p,q)
 G  =
```

```
!   0.7599    0.3483    0.0837    0.0081    0. !
 F  =

!   0.2401    0.6517    0.9163    0.9919    1. !
```

Puis on lance sans crainte

```
-->cdfbin("S",n,p,q,F,1-F)
```

Et là, la machine réagit violemment en tuant Scilab !

à suivre...