DEA Analyse et Modélisation des Systèmes Biologiques

Introduction au logiciel S-PLUS©

D. Chessel

2 - Statistique non paramétrique

Résumé

La fiche contient le matériel nécessaire pour une séance de travaux dirigés sur S-PLUS consacrée à la statistique non paramétrique. Les tests classiques (`chisq.test, wilcox.test, kruskal.test, friedman.test`) directement accessibles sont illustrés par les exemples de P. Dagnelie (1975 - *Théories et méthodes statistiques : Analyse statistique à plusieurs variables*, Tome 2. Les presses agronomiques de Gembloux, Gembloux. 1-362). Les tests sur les espaces à repectivement $n!$, $\binom{n}{m}$ et p^n éléments, dont les solutions analytiques sont données dans le cours de Ch. Gautier, ont des solutions simulées basée sur la fonction `sample`. Les exemples de rééchantillonnage du chapitre 1 du cours du module « Introduction à la planification expérimentale » de R. Tomassone sont repris (`jackknife` et `bootstrap`).

Plan

1 - Tests classiques

On prend les illustrations dans l'ouvrage de P. Dagnélie : Dagnelie, P. (1975) *Théories et méthodes statistiques : Analyse statistique à plusieurs variables*, Tome 2. Les presses agronomiques de Gembloux, Gembloux. 1-362.

Implanter au clavier les données (`scan`) :

```
> arbre
     V1 V2
 1 23.4 T1
 2 24.4 T1
 ...
26 27.4 T2
27 28.5 T2
> names(arbre)<-c("hau","type") # Pour changer les noms des variables
> arbre
    hau type
 1 23.4   T1
...
27 28.5   T2
> arbre$hau
 [1] 23.4 24.4 24.6 24.9 25.0 26.2 26.3 26.8 26.8 26.9 27.0 27.6
[13] 27.7 22.5 22.9 23.7 24.0 24.4 24.5 25.3 26.0 26.2 26.4 26.7
[25] 26.9 27.4 28.5
> arbre$type
 [1] T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T2 T2 T2 T2 T2 T2 T2 T2
[22] T2 T2 T2 T2 T2 T2
```

Test de la médiane (op. cit. p. 381)

```
> table(arbre$hau>=quantile(arbre$hau,0.5),arbre$type)
      T1 T2
FALSE  5  8
 TRUE  8  6

> chisq.test(table(arbre$hau>=quantile(arbre$hau,0.5),arbre$type))

  Pearson's chi-square test with Yates' continuity correction

data:  table(arbre$hau >= quantile(arbre$hau, 0.5), arbre$type)
X-square = 0.3426, df = 1, p-value = 0.5584

>
chisq.test(table(arbre$hau>=quantile(arbre$hau,0.5),arbre$type),correct=F
)

  Pearson's chi-square test without Yates' continuity correcti
on

data:  table(arbre$hau >= quantile(arbre$hau, 0.5), arbre$type)
X-square = 0.9423, df = 1, p-value = 0.3317
```

Comparaison avec le test paramétrique (op. cit. p. 132) :

```
> anova(lm(hau~type,data=arbre),test="F") # Voir le modèle linéaire
Analysis of Variance Table

Response: hau

Terms added sequentially (first to last)
          Df Sum of Sq Mean Sq F Value  Pr(F)
     type  1      2.30   2.295  0.9104 0.3491
Residuals 25     63.02   2.521
```

Test de Wilcoxon (op. cit. p. 384)

Les deux réfrences bibliographiques de la documentation de wilcox.test sont incontournables :

```
REFERENCES

Conover, W. J. (1980). Practical Nonparametric Statistics, 2nd ed. Wiley,
New York.

Lehmann, E. L.  (1975). Nonparametrics: Statistical Methods Based on
Ranks. Holden and Day, San Francisco.
```

```
> attach(arbre) # permet l'accès direct aux variables
> wilcox.test(hau[type=="T1"],hau[type=="T2"])

  Wilcoxon rank-sum test

data:  hau[type == "T1"] and hau[type == "T2"]
rank-sum normal statistic with correction Z = 1.068, p-value = 0.2854
alternative hypothesis: true mu is not equal to 0

Warning messages:
  cannot  compute  exact  p-value  with  ties  in:  wil.rank.sum(x,  y,
alternative, exact, correct)

> wilcox.test(hau[type=="T1"],hau[type=="T2"],correct=F)

  Wilcoxon rank-sum test

data:  hau[type == "T1"] and hau[type == "T2"]
rank-sum normal statistic without correction Z = 1.092, p-value = 0.2746
alternative hypothesis: true mu is not equal to 0

Warning messages:
  cannot  compute  exact  p-value  with  ties  in:  wil.rank.sum(x,  y,
alternative, exact, correct)

> detach("arbre") # Important
```

Test des paires de Wilcoxon (op. cit. p. 387)

```
> debout
 [1] 20.4 25.4 25.6 25.6 26.6 28.6 28.7 29.0 29.8 30.5 30.9 31.1
> abattu
 [1] 21.7 26.3 26.8 28.1 26.2 27.3 29.5 32.0 30.9 32.3 32.3 31.7
> wilcox.test(debout,abattu,paired=T,correct=F)

  Wilcoxon signed-rank test

data:  debout and abattu
signed-rank normal statistic without correction Z = -2.394, p-value =
0.0167
alternative hypothesis: true mu is not equal to 0

Warning messages:
  cannot  compute  exact  p-value  with  ties  in:  wil.sign.rank(dff,
alternative, ex
act, correct)
```

Test de Kurskal et Wallis (op. cit. p. 392)

Les données sont étendues (il y a plusieurs manières de le faire). Par exemple :

```
> provi<-scan()
1: 19.9
2: 21.1
3: 21.2
4: 22.1
5: 22.5
6: 23.6
7: 24.5
8: 24.6
9: 26.2
10: 26.7
11:
> provi
 [1] 19.9 21.1 21.2 22.1 22.5 23.6 24.5 24.6 26.2 26.7
> provi1<-cbind.data.frame(provi,rep("T3",le=10)) # Coller deux tableaux
ayant les mêms lignes.
> names(provi1)<-c("hau","type")
> arbre<-rbind.data.frame(arbre,provi1)

> attach(arbre)
> hau
    1    2    3    4  5    6    7    8    9   10 11   12   13   14
 23.4 24.4 24.6 24.9 25 26.2 26.3 26.8 26.8 26.9 27 27.6 27.7 22.5
   15   16 17   18   19   20 21   22   23   24   25   26   27   28
 22.9 23.7 24 24.4 24.5 25.3 26 26.2 26.4 26.7 26.9 27.4 28.5 18.9
   29   30   31   32   33   34   35   36   37
 21.1 21.2 22.1 22.5 23.6 24.5 24.6 26.2 26.7
> type
 [1] T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T2 T2 T2 T2 T2 T2 T2 T2
[22] T2 T2 T2 T2 T2 T2 T3 T3 T3 T3 T3 T3 T3 T3 T3 T3

> kruskal.test(hau,type)

  Kruskal-Wallis rank sum test

data:  hau and type
Kruskal-Wallis chi-square = 9.334, df = 2, p-value = 0.0094
alternative hypothesis: two.sided
```

Matrix, col, row

```
> pomme<-matrix(0,nrow=5,ncol=4)
> pomme[1,]_c(527,604,606,533) # les signes <- ou _ sont équivalents
> pomme[2,]_c(633,600,650,567)
> pomme[3,]_c(642,708,662,504)
> pomme[4,]_c(623,550,562,667)
> pomme[5,]_c(377,408,500,333)
> pomme
     [,1] [,2] [,3] [,4]
[1,]  527  604  606  533
[2,]  633  600  650  567
[3,]  642  708  662  504
[4,]  623  550  562  667
[5,]  377  408  500  333
> col(pomme)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
[4,]    1    2    3    4
[5,]    1    2    3    4
> row(pomme)
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    2    2    2
[3,]    3    3    3    3
[4,]    4    4    4    4
[5,]    5    5    5    5
```

Test de Friedman (op. cit. p. 394)

```
> as.vector(pomme)
 [1] 527 633 642 623 377 604 600 708 550 408 606 650 662 562 500 533
[17] 567 504 667 333
> pomme.vec<-as.vector(pomme)
> pomme.vec
 [1] 527 633 642 623 377 604 600 708 550 408 606 650 662 562 500 533 567
504 667 333
> traitement<-as.factor(row(pomme))
> traitement
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> bloc<-as.factor(col(pomme))
> bloc
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4

> friedman.test(pomme.vec,traitement,bloc)

   Friedman rank sum test

data:  pomme.vec and traitement and bloc
Friedman chi-square = 9.8, df = 4, p-value = 0.0439
alternative hypothesis: two.sided
```

Comparaison avec l'anova (op. cit. p. 183-184) :

```
>   anova(lm(pomme.vec~traitement+bloc),test="F")   #   Voir   le   modèle
linéaire.
Analysis of Variance Table

Response: pomme.vec

Terms added sequentially (first to last)
           Df Sum of Sq Mean Sq F Value  Pr(F)
traitement  4    133419   33355   9.576 0.0010
      bloc  3     14987    4996   1.434 0.2814
 Residuals 12     41797    3483
```

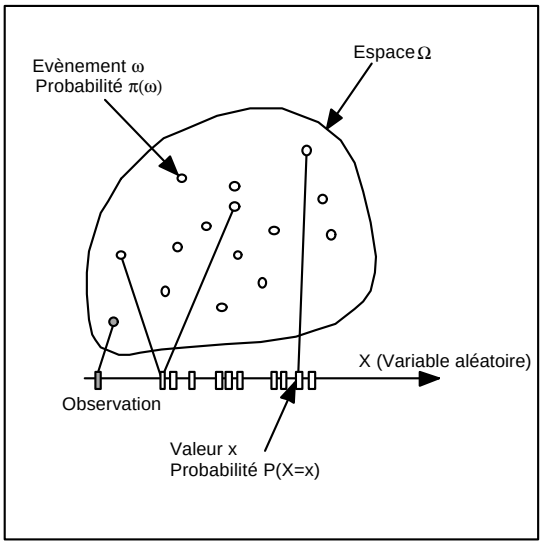
La précision de l'ouvrage de P. Dagnélie est légendaire.

Test du Chi2 (op. cit. p. 84)

```
> fongi<-matrix(c(203,150,6,266,112,1,258,126,2,196,168,17),
       byrow=T,nrow=4,ncol=3)
> fongi
     [,1] [,2] [,3]
[1,]  203  150    6
[2,]  266  112    1
[3,]  258  126    2
[4,]  196  168   17

> chisq.test(fongi)

   Pearson's chi-square test without Yates' continuity correcti
on

data:  fongi
X-square = 53.72, df = 6, p-value = 0
```

2 - Simulations dans les espaces non paramétriques

En statistique, on a toujours ce qu'on appelle un espace probabilisé symbolisé par :



A chaque événement, on associe une valeur numérique par le biais d'une fonction (variable aléatoire). Chacune des valeurs de la variable a une certaine probabilité d'être réalisée fonction de la probabilité des événements qui conduisent à cette valeur. Dans l'ensemble des valeurs de la variable aléatoire on regarde où se trouve la valeur associée à l'observation (qui s'appelle une statistique). Si la position de la statistique est anormale pour la loi induite sur la variable aléatoire par la loi de probabilité sur l'espace d'événements on pense que le modèle est invalide parce qu'on a observé quelque chose qui avait une très faible probabilité d'être observée. On ne raconte pas cette histoire dans chaque article mais on y pense chaque fois qu'on fait un test.

Ce principe de base peut être entièrement mathématisé (statistique mathématique) ou entièrement simulé sur ordinateur (computer-intensive method, citer par exemple Potvin, C. & Roff, D.A. (1993) Distribution-free and robust statistical methods: viable alternatives to parametric statistics ? *Ecology* : 74, 1617-1628).

Ce raisonnement est assez subtile comme en témoigne l'histoire qui suit. Un papa statisticien veut apprendre les rudiments à son fils. Il place 99 pièces de 1 F et 1 pièce de 5 F sur une table et dit « Prend une pièce au hasard. Si tu l'as prise au hasard tu la gardes, sinon je te donne une claque ». Le petit prend la pièce de 5 F et une claque. « Bon, tu n'as pas tout compris. Je te bande les yeux. Prend une pièce au hasard. Si tu l'as prise au hasard tu la gardes, sinon je te donne une claque ». Le petit, qui se méfie, tire un peu sur le bandeau, voit la pièce de 5 F et en prend une autre. « Très bien, tu vois que quand tu ne vois pas, tu tires au hasard ». Comme quoi, il y a plusieurs façon de se faire avoir.

Les plus simples des espaces probabilisés sont discrets : ils comptent un nombre fini d'évènements. Quand ce nombre est grand, on connait beaucoup de choses sur l'espace en tirant au hasard des éléments. On peut ainsi avoir une idée, plus ou moins précise, des lois des variables aléatoires associées à cet espace.

Sample

Choisir m parmi n objets :

```
> sample(c("a","b","c","d"),3,replace=F)
[1] "b" "d" "c"
```

```
> sample(c("a","b","c","d"),3,replace=F)
[1] "d" "c" "b"
```

Permuter n objets :

```
> sample(10,10)
 [1]  7  8  9  4  6  5 10  1  3  2
```

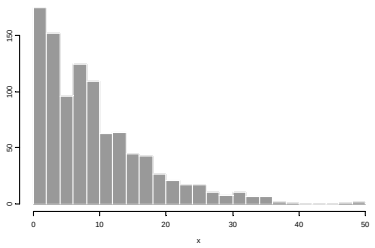
Placer p objets dans n cases :

```
> sample(5,8,replace=T)
[1] 1 3 3 5 1 5 2 2
```

2.1 - Quelle est la loi du plus petit (de la somme, de la variance, du plus grand, de ...) de 10 entiers tirés au hasard parmi 100 sans remise ?

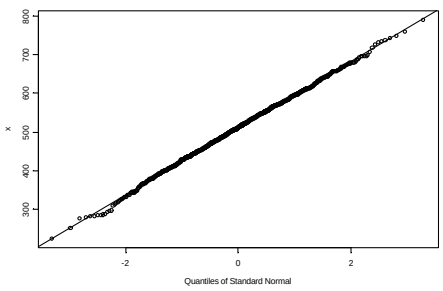
Implanter la fonction :

```
echa <-function (FUN,n=1000) {
   x<-seq(0,le=n)
   for (i in 1:n) x[i]<-FUN(sample(1:100, 10, replace = F))
   return(x)
}
> echa(min,10)
 [1]  8  6  8  8  4 13  7  2  2  5
> x<-echa(min,1000)
```




```
> hist(x,nclass=10,plot=F)
$breaks:
 [1]  0  5 10 15 20 25 30 35 40 45 50
$counts:
 [1] 384 271 155  87  48  25  22   5   0   3
```

```
> x<-echa(sum,1000)
> qqnorm(x)
> qqline(x)
```



La somme de 10 entiers choisis au hasard parmi 100 suit une loi normale.

Pour comprendre les qqplot (graphes quantiles-quantiles), il faut savoir qu'une loi de probabibilités définit la probabilité de dépasser une valeur donnée. Pour la loi normale (de moyenne 0 et de variance 1), on a 2.5 chances sur 100 de dépasser -1.96, et on a 95 chances sur 100 d'être entre -1.96 et 1.96 :

```
> pnorm(-1.96)
[1] 0.025
> pnorm(1.96)
[1] 0.975
```

Pour la même loi, la valeur qui a une probabilité donnée *p* d'être dépassée est le quantile d'ordre *p* :

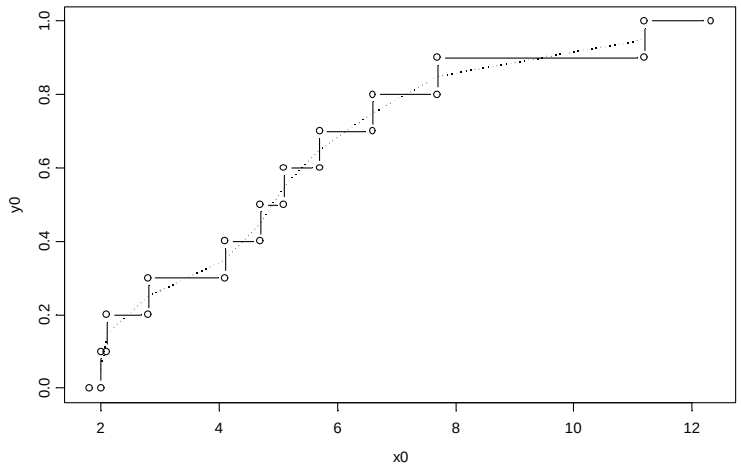
```
> qnorm(0.025)
[1] -1.96
> qnorm(0.975)
[1] 1.96
```

Considérons alors un échantillon d'observations :

```
> w
 [1]  5.7  4.1  2.1  5.1  7.7  4.7  2.0  2.8 11.2  6.6
```

Rangeons ces valeurs par ordre croissant :

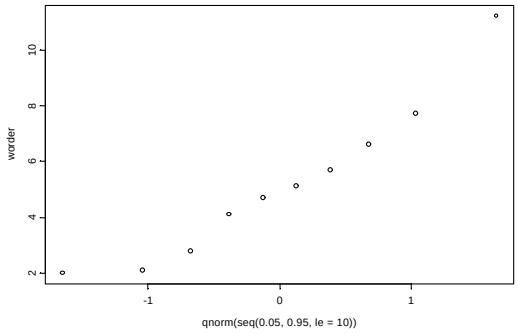
```
> worder<-sort(w)
> worder
 [1]  2.0  2.1  2.8  4.1  4.7  5.1  5.7  6.6  7.7 11.2

> x0<-rep(worder,rep(2,10))
> x0<-c(0.9*worder[1],x0,1.1*worder[10])
> x0
 [1]  1.80  2.00  2.00  2.10  2.10  2.80  2.80  4.10  4.10  4.70
[11]  4.70  5.10  5.10  5.70  5.70  6.60  6.60  7.70  7.70 11.20
[21] 11.20 12.32
> y0<-rep(seq(0,1,le=11),rep(2,11))
> plot(x0,y0,type="b")
> lines(worder,seq(0.05,0.95,le=10),lty=2)
```



La fonction en escalier qui monte de 1/10 à chaque valeur rencontrée est la fonction de répartition empirique. Quand on relie le milieu des « marches » on a le polygone des fréquences cumulées. Ceci montre que la première valeur estime grossièrement la quantile 0.05, la seconde le quantile 0.15, ... En général, les données sont rangées par ordre

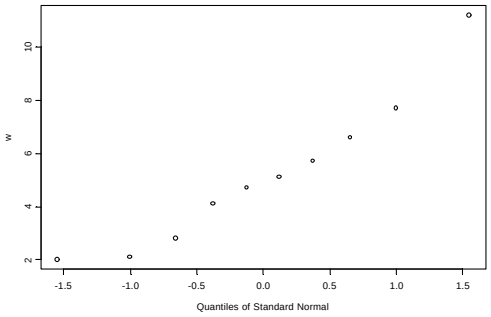
croissant et notées $y_{(1)}, y_{(2)}, \ldots, y_{(n)}$ (statistiques d'ordre). $y_{(i)}$ est le quantile empirique pour $p_i = (i - 0.5)/n$ (explications dans Chambers, J. M., Cleveland, W. S., Kleiner, B. and Tukey, P. A. (1983). *Graphical Methods for Data Analysis*. Wadsworth, Belmont, California, 395 p. voir p. 193 ou dans Hoaglin, D. C., Mosteller, F. and Tukey, J. W., editors (1983). *Understanding Robust and Exploratory Data Analysis*. Wiley, New York.). Le graphique quantile-quantile normal repésente les couples de points $\left(x_i, y_{(i)}\right)$ où x_i est le quantile de la loi normale pour $p_i = (i - 0.5)/n$. Si l'échantillon suit une loi normale, ces points sont sur une droite. C'est l'équivalent du tracé de la droite de Henry sur papier gaussien de nos ancètres (remarque : il y en a encore dans l'armoire).

```
> plot(qnorm(seq(0.05,0.95,le=10)),worder)
```



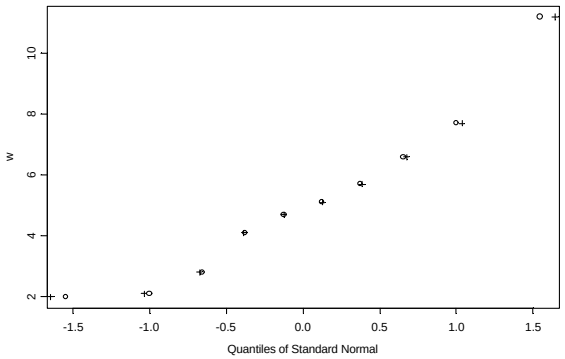
Ceci est obtenu directement par :

```
> qqnorm(w)
```



Mais pas tout à fait !

```
> qqnorm(w)
> points(qnorm(seq(0.05,0.95,le=10)),worder,pch=3)
```



En fait les $p_i = (i - 0.5)/n$:

```
> seq(0.05,0.95,le=10)
 [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
```

ont été remplacés par :

```
> ppoints(10)
 [1] 0.06098 0.15854 0.25610 0.35366 0.45122 0.54878 0.64634 0.74390
 [9] 0.84146 0.93902
```

DESCRIPTION

Returns a vector of probabilities suitable to produce a QQplot.

USAGE

ppoints(n, a=<<see below>>)

REQUIRED ARGUMENTS

n sample size for which plotting points desired (if length(n)==1), or data against which the plot is to be made.

OPTIONAL ARGUMENTS

a parameter that controls the precise placement of the plotting points, 0<=a<=1. The default is .5 if m>10 or .375 if m<=10
. where m is n if length(n)==1 and length(n) otherwise.

VALUE

the vector of probabilities, p, such that qdist(p) plotted against sort(y) gives a probability (QQ) plot of y against the distribution of which qdist is the quantile function. The result satisfies p[i]==(i-a)/(m+1-2*a).

DETAILS

Returns a vector of probabilities, p, such that qdist(p) plotted against sort(y) gives a probability (QQ) plot of y against the distribution of which qdist is the quantile function.

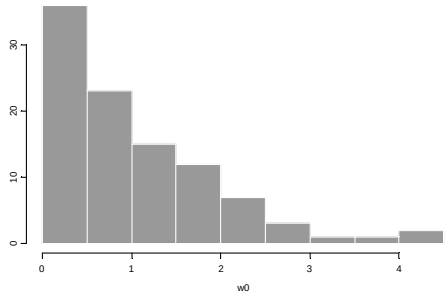
REFERENCES

Blom, G. (1958). Statistical Estimates and Transformed Beta Variables. Wiley, New York.

```
> ppoints(10)
 [1] 0.06098 0.15854 0.25610 0.35366 0.45122 0.54878 0.64634 0.74390
 [9] 0.84146 0.93902
> ((1:10)-0.375)/(10+1-2*0.375)
 [1] 0.06098 0.15854 0.25610 0.35366 0.45122 0.54878 0.64634 0.74390
 [9] 0.84146 0.93902
```

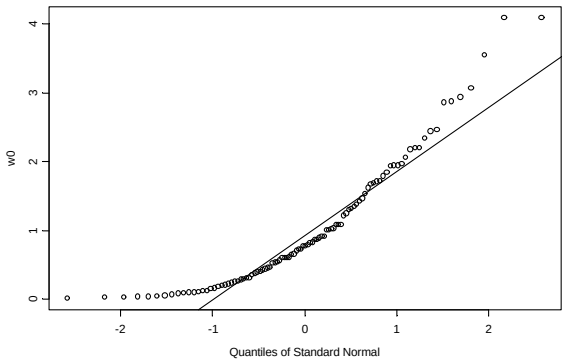
Ouf ! La documentation est transparente. Il faut retenir de cette expérience les capacités considérables que S-PLUS propose pour l'étude des distributions. Par exemple, générer un échantillon d'une loi exponentielle :

```
> w0<-rexp(100)
> hist(w0)
```



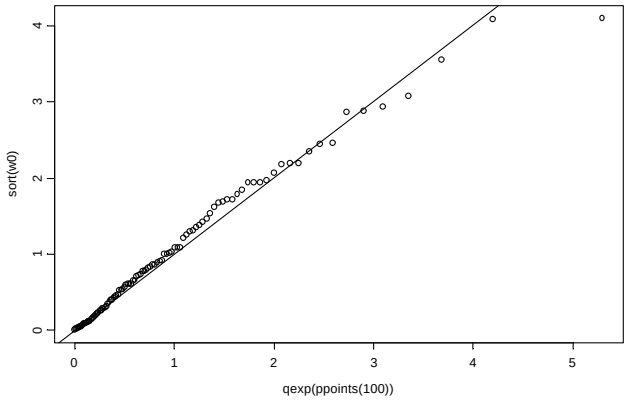
La distribution n'est pas gaussienne (et pour cause) :

```
> qqnorm(w0)
> qqline(w0)
```



Elle est exponentielle :

```
> qqplot(qexp(ppoints(100)),sort(w0))
> abline(0,1)
```

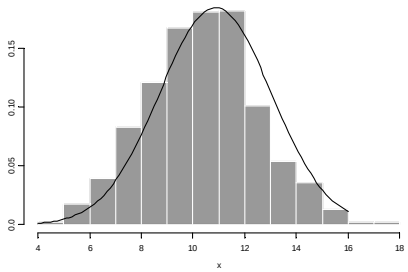


2.2 - Le nombre de suites est gaussien pour N>=10 et N-M>=10 ?

```
> y<-rep(0,10)
> y
 [1] 0 0 0 0 0 0 0 0 0 0
> z<-c(1,2,4,6,7,8)
> y[z]<-1
> y
 [1] 1 1 0 1 0 1 1 1 0 0
```

Implanter la fonction :

```
echaNS <-function (necha=1000,n=100,m=10) {
   x<-seq(0,le=necha)
   for (i in 1:necha) {
        y<-rep(0,le=n)
        z<-sample(1:n, m, replace = F)
        y[z]<-1
        x[i]<-(sum(abs(diff(y))))+1
   }
   return (x)
}

> echaNS(necha=5,n=10,m=6)
[1] 6 4 5 6 6
> x<-echaNS(1000,20,10)
> hist(x,proba=T)
> lines(seq(4,16,le=100),dnorm(seq(4,16,le=100),mean(x),sqrt(var(x))))
```

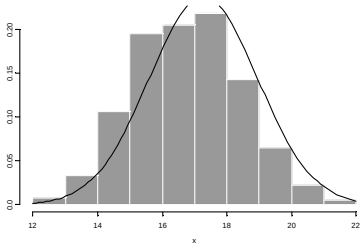


La prudence s'impose.

2.3 - Nombre de cases vides

```
echaCV <-function (necha=1000,n=36,p=26) {
```

```
   x<-seq(0,le=necha)
   for (i in 1:necha) {
        y<-rep(1,le=n)
        z<-sample(1:n, p, replace = T)
        y[z]<-0
        x[i]<-sum(y)
   }
   return (x)
}
> x<-echaCV(1000)
> hist(x)
> sum(x<=14)/1000
[1] 0.041
```



2.4 - Trois œufs dans cinq cocons :

```
> table(echaCV(1000,5,3))
   2   3  4
 488 485 27
```

On a observé 19,31,20 : c'est donc clair !

3 - Les techniques de rééchantillonnage

On reprend ici le chapitre 1 du polycopié de Tomassone R., Charles-Bajard S. & Bellanger L. (1998) Introduction à la planification expérimentale, DEA « Analyse et modélisation des systèmes biologiques », 1-13 pour refaire les calculs. On aborde ainsi le domainedes méthodes de rééchantillonnage en exercices imposés.

```
> goudron<-scan()
1: 0.45
2: 0.77
3: 1.07
4: 1.03
5: 1.34
6: 1.14
7: 1.15
8: 0.90
9: 0.55
10: 1.15
11:
> nicotine<-scan()
1: 11
2: 13
3: 14
4: 15
5: 17
6: 18
7: 14.5
8: 13.5
```

```
9: 8.5
10: 16.5
11:
> tabac<-cbind(goudron,nicotine)
```

bootstrap *(cyrano)*

```
>? bootstrap
```

DESCRIPTION

Performs bootstrap resampling of observations from specified data. Calculates bootstrap statistics for parameters of interest and produces an object of class bootstrap.

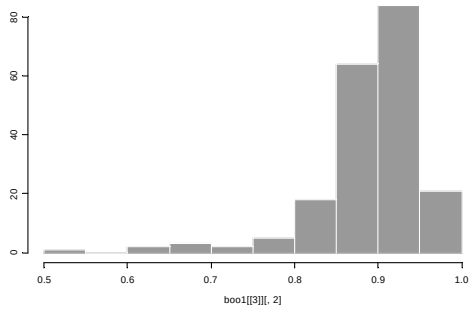
USAGE

```
bootstrap(data, statistic, B=1000, args.stat=NULL,
          group=NULL, sampler=samp.boot.mc, seed=.Random.seed,
          sampler.setup, sampler.wrapup, block.size=min(100,B),
          trace=T, assign.frame1=F, save.indices=F,
          statistic.is.random, seed.statistic=500)
```

```
> boo1<-bootstrap(tabac,cor,B=200)
Forming replications  1  to  100
Forming replications  101  to  200

> boo1
Call:
bootstrap(data = tabac, statistic = cor, B = 200)

Number of Replications: 200

Summary Statistics:
            Observed     Bias   Mean      SE
godrn.godrn   1.0000 0.000000 1.0000 0.00000
nictn.godrn   0.8902 0.005156 0.8953 0.05362
godrn.nictn   0.8902 0.005156 0.8953 0.05362
nictn.nictn   1.0000 0.000000 1.0000 0.00000
```

Objet bootstrap

```
> class(boo1)
[1] "bootstrap" "resamp"
> is.list(boo1)
[1] T

> boo1[[1]]
bootstrap(data = tabac, statistic = cor, B = 200)
> boo1[[2]]
 godrn.godrn nictn.godrn godrn.nictn nictn.nictn
           1      0.8902      0.8902           1
> boo1[[3]]
       godrn.godrn nictn.godrn godrn.nictn nictn.nictn
  [1,]           1      0.9072      0.9072           1
  [2,]           1      0.9761      0.9761           1
  [3,]           1      0.9258      0.9258           1
...
> range(boo1[[3]][,2])
[1] 0.5317 0.9838

> hist(boo1[[3]][,2])
```




```
> boo1[[5]]
[1] 200
> boo1[[5]]
[1] 200
> boo1[[6]]
[1] 10
> boo1[[7]]
[1] 2 2
> boo1[[8]]
NULL
> names(boo1)
 [1] "call"       "observed"   "replicates" "estimate"   "B"
 [6] "n"          "dim.obs"    "group"      "seed.start" "seed.end"
>
```

Ecrire une fonction

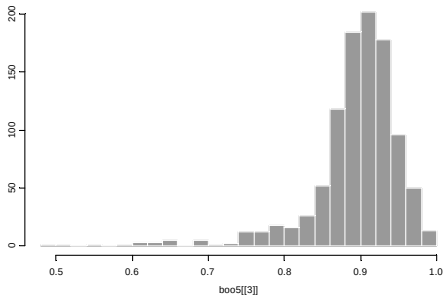
```
cor1<-function(X) {
  a<-cor(X)
  return(a[1,2])
}

> cor1(tabac)
[1] 0.8902
>

> boo5<-bootstrap(tabac,cor1,B=200)
Forming replications  1  to  100
Forming replications  101  to  200

> boo5
Call:
bootstrap(data = tabac, statistic = cor1, B = 200)
Number of Replications: 200

Summary Statistics:
     Observed      Bias   Mean      SE
cor1   0.8902 -0.006713 0.8835 0.07736

> limits.bca(boo5)
       2.5%     5%    95% 97.5%
cor1 0.5199 0.6297 0.9428 0.953
```

OPTIONAL ARGUMENTS

B number of bootstrap resamples to be drawn. We recommend at least 250 to estimate standard errors and 1000 to estimate percentiles.

```
> boo5<-bootstrap(tabac,cor1,B=1000)
Forming replications  1  to  100
...
Forming replications  901  to  1000

> limits.bca(boo5)
```

```
       2.5%     5%    95% 97.5%
cor1 0.6126 0.6447 0.9385 0.949

> hist(boo5[[3]],nclass=20)
```



jackknife *(eustachage)*

```
> ? jackknife
```

DESCRIPTION

Performs delete-one jackknifing of observations from specified data. Calculates jackknife statistics for parameters of interest and produces an object of class jackknife.

USAGE

```
jackknife(data, statistic, args.stat=NULL, seed=0, group.size=1,
          assign.frame1=F)
```

REQUIRED ARGUMENTS

data data to be jackknifed. May be a vector, matrix, or data frame.

```
> jackk1<-jackknife(tabac,cor1)
> names(jackk1)
[1] "call"       "observed"   "replicates" "estimate"   "B"
[6] "n"          "dim.obs"    "seed.start"

> jackk1[[3]]
        cor1
 [1,] 0.9013
 [2,] 0.8907
 [3,] 0.9011
 [4,] 0.8892
 [5,] 0.8763
 [6,] 0.9146
 [7,] 0.9059
 [8,] 0.8896
 [9,] 0.8710
[10,] 0.8834


> cor1(tabac[1:9,])
[1] 0.8834
> cor1(tabac[2:10,])
[1] 0.9013

> cor(tabac[1:9,])
         goudron nicotine
 goudron  1.0000   0.8834
nicotine  0.8834   1.0000
> cor1(tabac[-5,])
[1] 0.8763
```

L’eustachage est bien le jackknife.

```
> summary(jackk1)
Call:
jackknife(data = tabac, statistic = cor1)

Number of Replications: 10

Summary Statistics:
     Observed    Bias   Mean      SE
cor1   0.8902 0.01916 0.8923 0.03848

Empirical Percentiles:
       2.5%     5%    95%  97.5%
cor1 0.8722 0.8734 0.9107 0.9126
> ?quantile
> quantile(jackk1[[3]])
    0%    25%    50%    75%   100%
 0.871 0.8848 0.8902 0.9012 0.9146
> quantile(jackk1[[3]],probs=0.025)
   2.5%
 0.8722
```

Il y a plus de 2000 fonctions ! Premières impressions ?