Fiche d'utilisation du logiciel 

3 – Statistique non paramétrique

D. Chessel & J. Thioulouse

Résumé

La fiche contient le matériel nécessaire pour une séance de travaux dirigés sur R consacrée à la statistique non paramétrique. Les tests classiques directement accessibles sont illustrés par les exemples de P. Dagnélie (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 respectivement $n!$, $\binom{n}{m}$ et p^n éléments, dont les solutions analytiques sontconnues, ont des solutions simulées basée sur la fonction `sample`. Les exemples de rééchantillonnage sont abordées (`jackknife` et `bootstrap`). L'installation des librairies necessaires est décrite.

Plan

1. Bases du raisonnement statistique

De la connaissance d'un espace probabilisé, on peut déduire ce qui va se passer sur un échantillon, du genre si on tire une boule au hasard dans une urne contenant 7 rouges et 3 bleues, 7 fois sur 10 on aura une rouge ! On peut inverser totalement la question : si une urne contient 10 boules et qu'en tirant au hasard on obtient 3 rouges qu'est-ce qu'on peut dire des autres ? C'est ce qu'on appelle l'inférence statistique.

1.1. Vraisemblance d'une hypothèse

La base du raisonnement est dans la fonction de vraisemblance. Prenons un exemple. Il y a dans l'amphi 100 personnes et l'enseignant X veut savoir combien d'étudiants ont une opinion favorable de ce qu'il raconte. Le plus efficace est de les interroger tous un par un. Mais c'est long et pénible. Alors X en prend 5 au hasard et pose la question « Pensez vous que la statistique est intéressante? ». C'est OUI 4 fois et NON 1 fois. On peut dire que cet échantillon représente tout à fait la réalité et que 80% des étudiants s'intéressent à la statistique. On peut dire aussi bien que X a eu beaucoup de chance et que la petite minorité de ceux qui s'intéressent à la statistique est sur-représentée.

Il y a exactement 101 hypothèses *a priori* : dans l'amphi, il y a *m* étudiants qui répondent OUI à la question. Ces hypothèses notées $H_0, H_1, ..., H_{100}$ correspondent aux valeurs 0, 1, 2, ..., 100, valeurs que peut prendre l'inconnue *m* (*a posteriori*, après l'échantillonnage, on peut éliminer *certainement* les cas 100, 99, 98, 97 et 0) mais *m* est inconnu et peut prendre *a priori* les valeurs 0 à 100. Nous allons étudier ces 101 hypothèses.

Soit l'hypothèse H_{30}. Si *m* vaut 30, en tirant au hasard 5 étudiants sur 100 on aura l'observation 0, 1, 2, 3, 4 ou 5 avec les probabilités $P(0), P(1), ..., P(5)$. Il y a $\binom{100}{5}$ façons de choisir 5 étudiants et :

$$P(j) = \frac{\binom{30}{j}\binom{70}{5-j}}{\binom{100}{5}}$$

```
> dhyper(4,30,70,5)
[1] 0.02548
```

La probabilité de trouver 4 (le résultat observé) est alors 0.02548.

```
> ?Hypergeometric

Hypergeometric             package:base             R Documentation

The Hypergeometric Distribution

Description:

     Density, distribution function, quantile function and random
```

```
     generation for the hypergeometric distribution.

Usage:

     dhyper(x, m, n, k, log = FALSE)
     phyper(q, m, n, k, lower.tail = TRUE, log.p = FALSE)
     qhyper(p, m, n, k, lower.tail = TRUE, log.p = FALSE)
     rhyper(nn, m, n, k)

Arguments:

    x, q: vector of quantiles representing the number of white balls
          drawn without replacement from an urn which contains both
          black and white balls.

       m: the number of white balls in the urn.

       n: the number of black balls in the urn.

       k: the number of balls drawn from the urn.

       p: probability, it must be between 0 and 1.

      nn: the number of observations to be generated.

log, log.p: logical; if TRUE, probabilities p are given as log(p).

lower.tail: logical; if TRUE (default), probabilities are P[X <= x],
          otherwise, P[X > x].

Value:

     `dhyper' gives the density, `phyper' gives the distribution
     function, `qhyper' gives the quantile function, and `rhyper'
     generates random deviates.

References:

     Johnson, N. L., Kotz, S., and Kemp, A. W. (1992) Univariate
     Discrete Distributions, Second Edition. New York: Wiley.

> ?Poisson      Voir l'uniformité de la construction
> ?Binomial
```

La probabilité d'observer le résultat sous une hypothèse H arbitraire est la vraisemblance de cette hypothèse pour cette observation. Nous pouvons tracer la valeur de la vraisemblance en fonction de l'hypothèse. Ce n'est pas une loi de probabilité mais une fonction dont les valeurs sont des probabilités pour des lois différentes.

Ce que le help ne dit pas : on peut mettre plusieurs valeurs pour le premier paramètre.

```
> dhyper(4,0:10,1,5)
 [1]    NaN    NaN    NaN    NaN 1.0000 0.8333 0.7143 0.6250 0.5556 0.5000
[11] 0.4545
Warning message:
NaNs produced in: dhyper(x, m, n, k, log)
```

C'est normal, trouver 4 blanches sur 5 tirées quand il n'y en a pas, c'est trop.

```
> dhyper(4,6,1,5)
[1] 0.7143
> dhyper(4,4:10,1,5)
[1] 1.0000 0.8333 0.7143 0.6250 0.5556 0.5000 0.4545
```

Ce que le help ne dit pas : on peut mettre plusieurs valeurs pour le second paramètre.

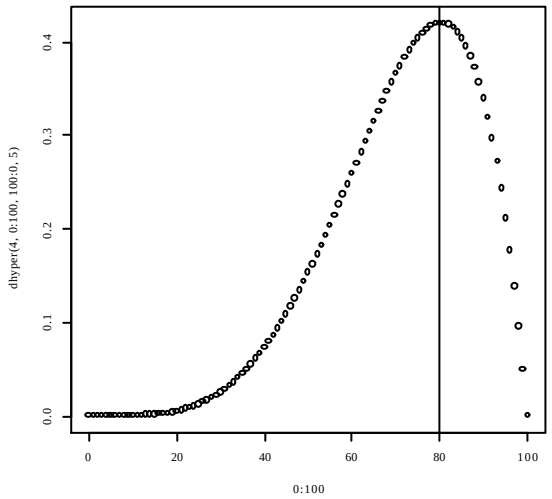
```
> dhyper(4,6,0:6,5)
[1] 0.0000 0.7143 0.5357 0.3571 0.2381 0.1623 0.1136
```

C'est normal, trouver 4 blanches sur 5 tirées quand il n'y a pas de noires, c'est impossible. Ce que le help ne dit pas : on peut mettre plusieurs valeurs pour les deux paramètres !

```
> dhyper(4,3:5,1:2,5)
[1]    NaN 0.3333 0.8333
> dhyper(4,3,1,5)     3 blanches et 1 noire
[1] NaN
> dhyper(4,4,2,5)     4 blanches et 2 noires
[1] 0.3333
> dhyper(4,5,1,5)     5 blanches et 1 noire
[1] 0.8333

> dhyper(4,3:5,1:6,5)
[1]     NaN 0.33333 0.26786 0.00000 0.03968 0.06494
> dhyper(4,3,1,5)     3 blanches et 1 noire
[1] NaN
> dhyper(4,4,2,5)     4 blanches et 2 noires
[1] 0.3333
> dhyper(4,5,3,5)     5 blanches et 3 noires
[1] 0.2679
> dhyper(4,3,4,5)     3 blanches et 4 noires
[1] 0
> dhyper(4,4,5,5)     4 blanches et 5 noires
[1] 0.03968
> dhyper(4,5,6,5)     5 blanches et 6 noires
[1] 0.06494
```

On retrouve un principe propre à R : les deux séries de paramètres sont explorées ensemble et la plus courte est réutilisée en cas de besoin.

```
> plot(0:100,dhyper(4,0:100,100:0,5))
> abline(v=80)
```

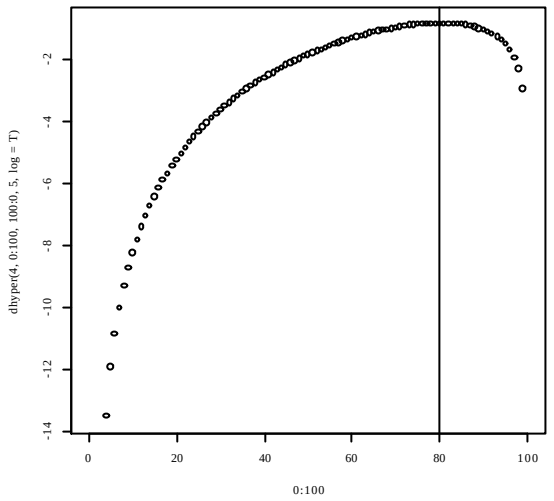


Estimer, c'est choisir une des hypothèses. Estimer au maximum de vraisemblance, c'est choisir l'hypothèse qui donne à l'observation la plus grande vraisemblance. On trouve ici 80. C'est le plus vraisemblable. On note la vraisemblance par :

$$\boxed{L(H) = P_{Hvraie}(\text{Observation})}$$

L renvoie à Likelihood inventé par Sir Ronald Fisher (1890-1962). On parle de statistique fishérienne. Chercher à maximiser L, c'est aussi chercher à maximiser log(L) :

```
> plot(0:100,dhyper(4,0:100,100:0,5,log=T))
> abline(v=80)
```



1.2. Rejet d'une hypothèse

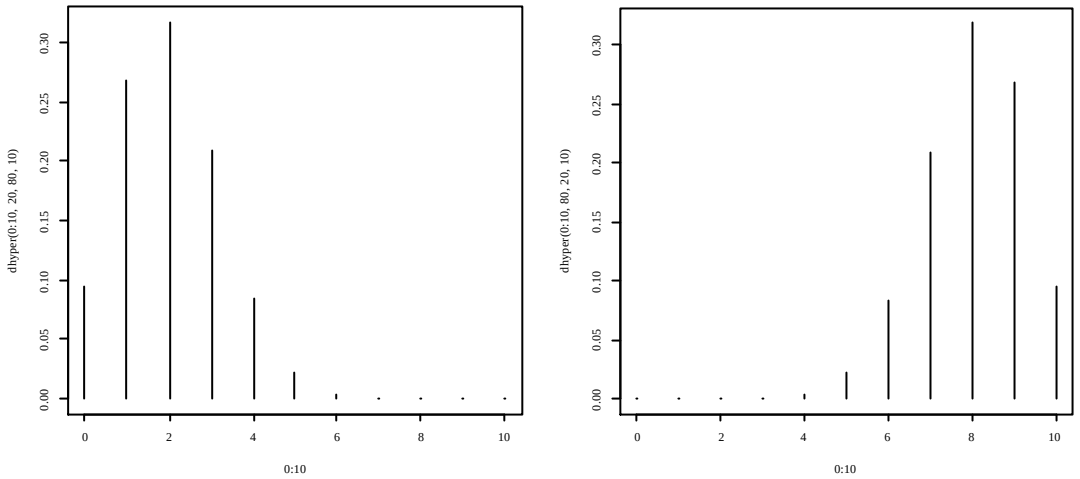
On a choisi une hypothèse : elle a toutes les chances d'être fausse ! Supposons maintenant qu'on interroge 10 personnes et qu'on en trouve 6 réponses favorables. En tout, l'estimation au maximum de vraisemblance donne 60. On ne saura *jamais* si c'est 60. 59, 58 , 61 ou 63 sont presque aussi vraisemblables. Par contre 10 l'est beaucoup moins. Quand peut-on dire que l'hypothèse rend aberrant un résultat ? Peut on être sûr que H_{10} n'est pas acceptable ?

Tester une hypothèse, c'est décider au vu du résultat si elle est vraie ou si elle est fausse. *Décider si elle est vraie est impossible*. X ne pourra jamais savoir si il y a 60 ou 59 ou 61 personnes ayant une opinion positive (sauf à les interroger toutes mais penser aux ours blancs). X est déjà certain qu'il y en a au moins 6 et pas plus de 98. Ce serait bien étonnant qu'il y en ait eu 7, ou 8, ou 9, mais quand doit-on s'arrêter ? S'il y en a 20 ou 80, la loi du résultat est :

```
> dhyper(0:10,80,20,10)
 [1] 1.067e-08 7.762e-07 2.300e-05 3.679e-04 3.541e-03 2.153e-02 8.411e-02
 [8] 2.092e-01 3.182e-01 2.679e-01 9.512e-02
> dhyper(0:10,20,80,10)
 [1] 9.512e-02 2.679e-01 3.182e-01 2.092e-01 8.411e-02 2.153e-02 3.541e-03
 [8] 3.679e-04 2.300e-05 7.762e-07 1.067e-08

> sum(dhyper(0:10,20,80,10)*(0:10))
[1] 2
> sum(dhyper(0:10,80,20,10)*(0:10))
[1] 8
```

Dans l'hypothèse 20, on attend en moyenne 2 positifs (et on en a 6). Dans l'hypothèse 80 on en attend en moyenne 8 (et on en a 6). Dans le premier cas, c'est plutôt faible. Dans le second cas, c'est plutôt fort. Dans les deux cas, on se demande si c'est plutôt loin.

```
> plot(0:10,dhyper(0:10,20,80,10),type = "h")
> plot(0:10,dhyper(0:10,80,20,10),type = "h")
```




```
> sum(dhyper(0:6,80,20,10))
[1] 0.1096
> phyper(6,80,20,10)       Probabilité d'être inférieure ou égale à l'observé
[1] 0.1096
```

Le fait d'en trouver 6 au plus sur 10 tirés quand il y en a 80 sur 100 n'est pas aberrant.

```
> phyper(6,90,10,10)
[1] 0.008225
```

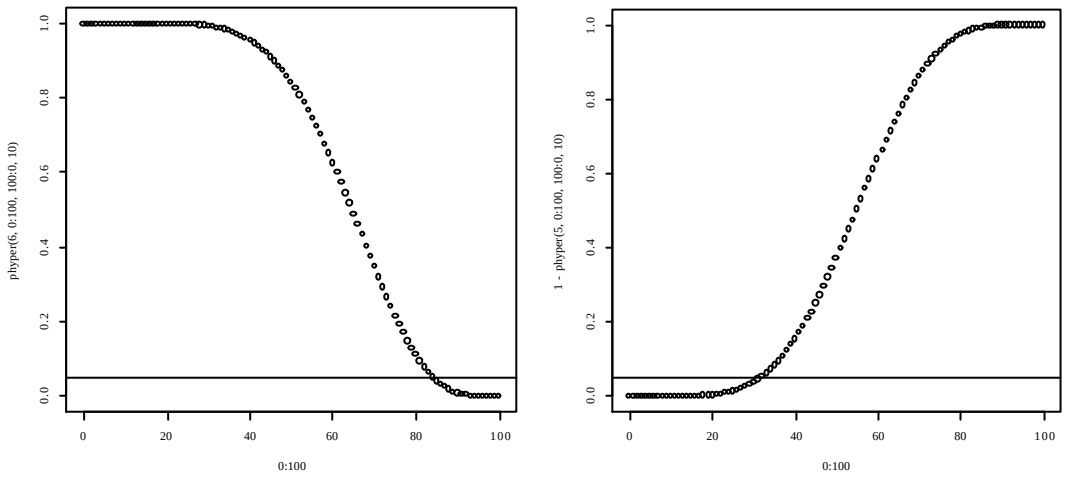
Le fait d'en trouver 6 au plus sur 10 tirés quand il y en a 90 sur 100 est anormal.

```
> plot(0:100,phyper(6,0:100,100:0,10))
> phyper(6,0:100,100:0,10)
  [1] 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
  [8] 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
 [15] 1.000e+00 1.000e+00 9.999e-01 9.999e-01 9.998e-01 9.997e-01 9.996e-01
 [22] 9.994e-01 9.992e-01 9.989e-01 9.985e-01 9.979e-01 9.973e-01 9.965e-01
 [29] 9.954e-01 9.942e-01 9.927e-01 9.909e-01 9.888e-01 9.862e-01 9.833e-01
 [36] 9.799e-01 9.760e-01 9.716e-01 9.665e-01 9.607e-01 9.543e-01 9.471e-01
 [43] 9.391e-01 9.302e-01 9.204e-01 9.097e-01 8.981e-01 8.854e-01 8.717e-01
 [50] 8.569e-01 8.411e-01 8.242e-01 8.062e-01 7.871e-01 7.670e-01 7.458e-01
 [57] 7.236e-01 7.005e-01 6.764e-01 6.515e-01 6.258e-01 5.993e-01 5.723e-01
 [64] 5.447e-01 5.167e-01 4.883e-01 4.598e-01 4.311e-01 4.025e-01 3.741e-01
 [71] 3.460e-01 3.183e-01 2.912e-01 2.648e-01 2.392e-01 2.146e-01 1.911e-01
 [78] 1.687e-01 1.476e-01 1.279e-01 1.096e-01 9.275e-02 7.745e-02 6.370e-02
 [85] 5.149e-02 4.080e-02 3.160e-02 2.384e-02 1.742e-02 1.225e-02 8.225e-03
 [92] 5.204e-03 3.047e-03 1.605e-03 7.248e-04 2.544e-04 5.355e-05 0.000e+00
 [99] 0.000e+00 0.000e+00 0.000e+00
> plot(0:100,phyper(6,0:100,100:0,10))
> abline(h=0.05)
```

A partir de 85 sur 100 la probabilité de ne pas en trouver plus de 6 est inférieure ou égale à 5%. Les hypothèse 100, 99, 97 sont impossibles (il faut au moins 4 noires pour avoir pas plus de 6 blanches). Les hypothèses 96, 95, 94, ..., 85 sont peu vraisemblables. Dans l'autre sens :

```
> sum(dhyper(6:10,80,20,10))
```

```
[1] 0.9745
> 1-phyper(5,80,20,10)     Probabilité d'être supérieure ou égale à l'observé
[1] 0.9745
> 1-phyper(5,20,80,10)
[1] 0.003933    Quand il y en a 20, en trouver au moins 6 est anormal


> 1-phyper(5,0:100,100:0,10)
  [1] 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 1.762e-07
  [8] 1.188e-06 4.578e-06 1.322e-05 3.182e-05 6.737e-05 1.296e-04 2.314e-04
 [15] 3.893e-04 6.235e-04 9.582e-04 1.422e-03 2.047e-03 2.871e-03 3.933e-03
 [22] 5.279e-03 6.956e-03 9.014e-03 1.151e-02 1.449e-02 1.802e-02 2.216e-02
 [29] 2.696e-02 3.247e-02 3.877e-02 4.588e-02 5.388e-02 6.279e-02 7.267e-02
 [36] 8.356e-02 9.547e-02 1.084e-01 1.225e-01 1.376e-01 1.538e-01 1.711e-01
 [43] 1.895e-01 2.089e-01 2.293e-01 2.507e-01 2.731e-01 2.963e-01 3.202e-01
 [50] 3.450e-01 3.703e-01 3.963e-01 4.227e-01 4.495e-01 4.765e-01 5.037e-01
 [57] 5.310e-01 5.582e-01 5.853e-01 6.121e-01 6.386e-01 6.645e-01 6.899e-01
 [64] 7.146e-01 7.385e-01 7.615e-01 7.837e-01 8.048e-01 8.249e-01 8.438e-01
 [71] 8.616e-01 8.782e-01 8.936e-01 9.078e-01 9.208e-01 9.325e-01 9.431e-01
 [78] 9.526e-01 9.609e-01 9.682e-01 9.745e-01 9.799e-01 9.844e-01 9.882e-01
 [85] 9.913e-01 9.937e-01 9.956e-01 9.970e-01 9.981e-01 9.988e-01 9.993e-01
 [92] 9.996e-01 9.998e-01 9.999e-01 1.000e+00 1.000e+00 1.000e+00 1.000e+00
 [99] 1.000e+00 1.000e+00 1.000e+00
> plot(0:100,1-phyper(5,0:100,100:0,10))
> abline(h=0.05)
```



Les hypothèse 0, 1,..., 5 sont impossibles (il faut au moins 6 blanches pour avoir au moins 6 blanches). Les hypothèses 7, 8, 9, ..., 30 sont peu vraisemblables.

Pour les hypothèse de H_0 jusqu'à H_{35} le résultat est trop favorable au risque de 5% pour qu'on puisse accepter l'hypothèse. A l'inverse, on peut se demander pour quelles hypothèses le résultat est trop défavorable.

1.3. Intervalle de confiance

Les hypothèses H_0 jusqu'à H_{30} et H_{85} jusqu'à H_{100} sont rejetées au risque d'erreur de 5%. Les hypothèses H_{31} jusqu'à H_{84} ne peuvent être exclues. On dit que :

Les valeurs de 31 à 84 forment l'intervalle de confiance [31,84] de l'estimation de *m* au niveau de confiance de 90%.

Trouver dans un échantillon de taille r =10 dans une population de taille n = 100 un résultat de j = 6 opinions positives conduit à penser que dans la population toute entière il y a entre m = 31 et m = 84 personnes d'opinion positive. La maîtrise, pour l'utilisation, des concepts de vraisemblance, test d'hypothèse et estimation est indispensable. A retenir :

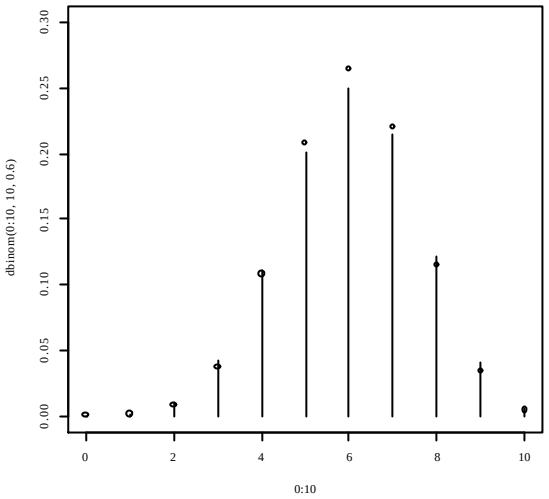
Le calcul des probabilités parle de l'échantillon à partir de la population, la statistique inférentielle parle de la population à partir de l'échantillon.

Appeler la librairie des tests statistiques :

```
> library(ctest)
> prop.test(6,10,conf=0.9)

         1-sample proportions test with continuity correction

data:  6 out of 10, null probability 0.5
X-squared = 0.1, df = 1, p-value = 0.7518
alternative hypothesis: true p is not equal to 0.5
90 percent confidence interval:
 0.3095 0.8405
sample estimates:
  p
0.6
```

Fondamentalement, R est un logiciel de statistique. On peut même dire que c'est un langage de statistique. Quand une urne contient 100 boules, on peut considérer qu'elle est infinie.

```
> dhyper(0:10,60,40,10)
 [1] 4.897e-05 9.478e-04 7.864e-03 3.686e-02 1.081e-01 2.076e-01 2.643e-01
 [8] 2.204e-01 1.153e-01 3.416e-02 4.355e-03
> dbinom(0:10,10,0.6)
 [1] 0.0001049 0.0015729 0.0106168 0.0424673 0.1114767 0.2006581 0.2508227
 [8] 0.2149908 0.1209324 0.0403108 0.0060466
> plot(0:10,dbinom(0:10,10,0.6),type="h",ylim=c(0,0.3))
> points(0:10,dhyper(0:10,60,40,10))
```



La loi hypergéométrique tend vers la loi binomiale quand les nombres de boules tendent vers l'infini, leur proportion étant constante. Le même raisonnement est utilisé sur le p de la loi binomiale. prop.test fait le test de l'hypothèse nulle $p = 0.5$. Ce n'est pas l'objet de son emploi. Ce pourrait l'être. 6 sur 10 n'invalide pas l'hypothèse $p = 0.5$, évidemment. 6 sur 10 invalide au seuil de 5% toutes les hypothèses p avec $p \leq 0.3095$ et $p \geq 0.8405$.

1.4. Fonction puissance

En général, quand un test n'est pas significatif, on dit :« Le test ne permet pas de rejeter l'hypothèse nulle ». Le risque de première espèce (probabilité de rejeter l'hypothèse quand elle est vraie) est toujours connu. La probabilité d'accepter l'hypothèse quand elle est fausse l'est rarement. Dans les cas simples on peut la calculer. C'est une fonction du caractère plus ou moins faux de l'hypothèse nulle.

Un étudiant A soutient qu'il est plus fort que B aux dames. Supposons que ça soit vrai. A et B décident de tester l'hypothèse H_0 « A et B sont égaux aux dames » à l'aide de 10 parties. Ils se mettent d'accord sur la procédure en supposant que deux parties sont indépendantes. S'ils sont de même force, ils ont une chance sur deux de gagner chaque partie. On dira que A est plus fort s'il gagne un nombre anormal de parties.

```
> dbinom(0:10,10,0.5) # Attention numéros de 1 à 11 pour les valeurs de 0 à 10
 [1] 0.0009766 0.0097656 0.0439453 0.1171875 0.2050781 0.2460938 0.2050781
 [8] 0.1171875 0.0439453 0.0097656 0.0009766
```

La probabilité de gagner 3 parties est $\binom{10}{3}\left(\frac{1}{2}\right)^{10} = \frac{10\times 9\times 8}{3\times 2}\frac{1}{2^{10}} = \frac{720}{6\times 1024} = 0.1171875$

```
> pbinom(0:10,10,0.5)
 [1] 0.0009766 0.0107422 0.0546875 0.1718750 0.3769531 0.6230469 0.8281250
 [8] 0.9453125 0.9892578 0.9990234 1.0000000
```

La probabilité d'en gagner au plus 3 est 0.1718.

```
> 1-pbinom(0:10,10,0.5)
 [1] 0.9990234 0.9892578 0.9453125 0.8281250 0.6230469 0.3769531 0.1718750
```

```
 [8] 0.0546875 0.0107422 0.0009766 0.0000000
```

La probabilité d'en gagner 4 ou plus est de 0.8282. Ils décident de faire un test au risque a=5%. La probabilité d'en gagner 8 ou plus vaut 0.055. Donc si A gagne 8, 9 ou 10 parties, on rejettera l'hypothèse H_0. Sinon ?

Ce qui va se passer dépend de la façon dont A est plus fort que B.

Ceci peut se mesurer par la probabilité p réelle que A a de gagner une partie contre B. Si A est vraiment très fort $p=1$. Ils jouent 10 parties, A gagne 10 fois et le test donne « H_0 est fausse ». Et elle est bien fausse. B ne dit plus rien.

Si A est grandement plus fort que B mais si sa domination n'est pas écrasante, on aura $p=0.9$. Ils jouent 10 fois :

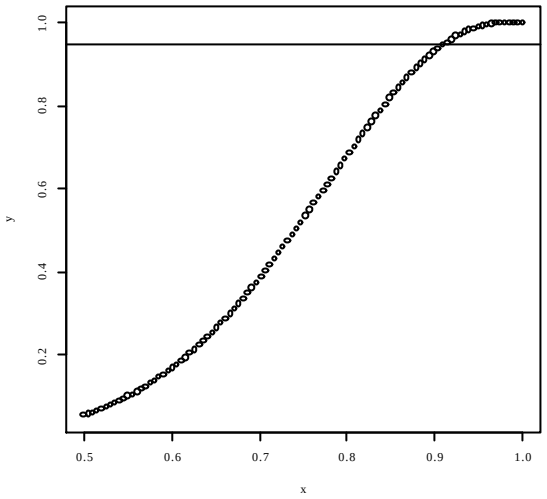
```
> dbinom(0:10,10,0.9)
 [1] 1.000e-010 9.000e-009 3.645e-007 8.748e-006 1.378e-004 1.488e-003 1.116e-002
 [8] 5.740e-002 1.937e-001 3.874e-001 3.487e-001
> pbinom(0:10,10,0.9)
 [1] 1.000e-010 9.100e-009 3.736e-007 9.122e-006 1.469e-004 1.635e-003 1.280e-002
 [8] 7.019e-002 2.639e-001 6.513e-001 1.000e+000
> 1-pbinom(0:10,10,0.9)
 [1] 1.0000 1.0000 1.0000 1.0000 0.9999 0.9984 0.9872 0.9298 0.7361 0.3487 0.0000
```

Dans 35% des cas, A gagne 10 fois et le test est très significatif. Dans 39% des cas, A gagne 9 fois et le test est très significatif. Dans 19% des cas, A gagne 8 fois et le test est significatif. Dans 6 % des cas A, gagne 7 fois et le test n'est pas significatif. Dans 1% des cas, A gagne 6 fois et il l'est encore moins. Quand le test n'est pas significatif, B dit « H_0 est vraie » et se trompe. Quelle probabilité avait-il de se tromper ?

```
pbinom(7,10,0.9)
[1] 0.07019
```

Il peut dire « j'accepte l'hypothèse nulle », j'ai 7 chances sur 100 de me tromper. Dans le cas seulement où $p=0.9$. Dans le cas seulement où le test est à 5%. On appelle puissance du test *la probabilité d'erreur quand on accepte l'hypothèse nulle*. C'est une fonction du risque de première espèce et de la vraie alternative. On peut donc tracer la fonction :

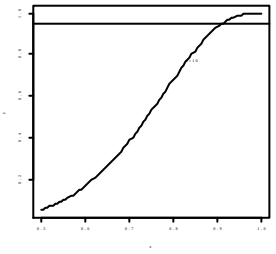
```
> x_seq(0.5,1,le=100)
> y_1-pbinom(7,10,x)
> plot(x,y)
> abline(h=0.95)
```



Ceci veut dire que, si le test est significatif, A peut dire qu'il est le plus fort avec une probabilité de se tromper de α=5% et que, s'il ne l'est pas, B ne peut pas dire qu'ils se valent. Il pourra juste dire que la puissance du test est suffisante pour dire que $p \leq 0.9$ mais pas que $p = 0.5$. Le graphe ci-dessus est celui de la puissance du test « une chance sur 2 » quand *n* vaut 10 et α=5%.

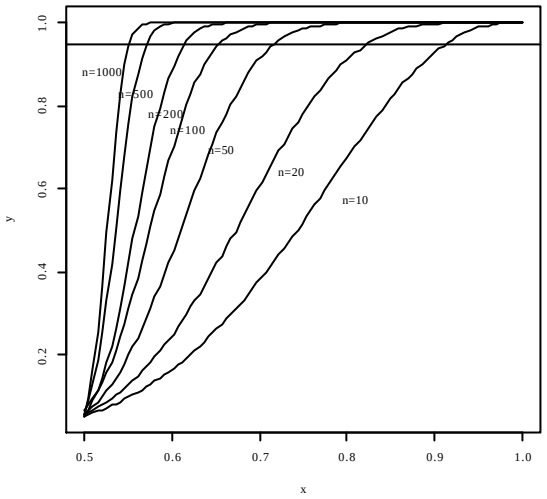
La puissance du test est une fonction de *n*. Il faut recommencer le raisonnement en entier pour $n = 20$. Ceci s'automatise par :

```
> y_1-pbinom(qbinom(0.95,10,0.5)-1,10,x)
> plot(x,y,type="n")
> lines(x,y)
> text(locator(1),"n=10")
> abline(h=0.95)
```



On peut maintenant refaire cette opération pour 20, 50, 100, 200, 500 et 1000.

```
> toto
function ()
{
        x_seq(0.5,1,le=100)
        y_1-pbinom(qbinom(0.95,10,0.5)-1,10,x)
        plot(x,y,type="n")
        lines(x,y)
        text(locator(1),"n=10")
        abline(h=0.95)
        for (n in c(20,50,100,200,500,1000)) {
                y_1-pbinom(qbinom(0.95,n,0.5)-1,n,x)
                lines(x,y)
                text(locator(1),paste("n=",n,sep=""))
        }
}
```



En faisant 20 parties, on pourra mettre en évidence $p < 0.83$; en faisant 100 parties $p < 0.63$; en faisant 1000 parties $p < 0.53$. On n'aura jamais d'argument pour $p = 0.5$.

> En statistique, on n'accepte JAMAIS une hypothèse. On discute au mieux de celles qu'on peut refuser avec un risque donné.

2. 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`) ou lire les données :

```
> read.table("arbre.txt")
     V1 V2
1  23.4 T1
2  24.4 T1
3  24.6 T1
...
25 26.9 T2
26 27.4 T2
27 28.5 T2
> arbre_read.table("arbre.txt")
> names(arbre)<-c("hau","type")
> names(arbre)<-c("hau","type") Pour changer les noms des variables
> arbre
    hau type
1  23.4   T1
2  24.4   T1
...
26 27.4   T2
27 28.5   T2
> arbre$hau
 [1] 23.4 24.4 24.6 24.9 25.0 26.2 26.3 26.8 26.8 26.9 27.0 27.6 27.7 22.5 22.9
[16] 23.7 24.0 24.4 24.5 25.3 26.0 26.2 26.4 26.7 26.9 27.4 28.5
> arbre$type
 [1] T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T1 T2 T2 T2 T2 T2 T2 T2 T2 T2 T2 T2 T2
[26] T2 T2
```

```
Levels:  T1 T2 c'est un facteur
```

2.1. 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 correction

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
          Df Sum Sq Mean Sq F value Pr(>F)
type       1    2.3     2.3    0.91   0.35
Residuals 25   63.0     2.5
```

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

La référence bibliographique de la documentation de wilcox.test est incontournable :

```
References:

     Myles Hollander & Douglas A. Wolfe (1973), Nonparametric
     statistical inference. New York: John Wiley & Sons. Pages 27-33
     (one-sample), 68-75 (two-sample).
```

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

         Wilcoxon rank sum test with continuity correction

data:  hau[type == "T1"] and hau[type == "T2"]
W = 113.5, p-value = 0.2854
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test(hau[type == "T1"],
hau[type == "T2"])

> detach("arbre") Important
```

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

```
> debout_c(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_c(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
V = 8.5, p-value = 0.01669
alternative hypothesis: true mu is not equal to 0

Warning message:
Cannot compute exact p-value with ties in: wilcox.test(debout, abattu, paired =
T, correct = F)
```

2.3. 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_c(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)  Coller deux tableaux ayant les mêms
colonnes.

> attach(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 27.7 22.5 22.9
[16] 23.7 24.0 24.4 24.5 25.3 26.0 26.2 26.4 26.7 26.9 27.4 28.5 19.9 21.1 21.2
[31] 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 T2 T2 T2 T2
[26] T2 T2 T3 T3 T3 T3 T3 T3 T3 T3 T3 T3
Levels:  T1 T2 T3

> kruskal.test(hau,type)

        Kruskal-Wallis rank sum test

data:  hau and type
Kruskal-Wallis chi-square = 9.334, df = 2, p-value = 0.009402
```

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
```

2.4. 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
Levels:  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
Levels:  1 2 3 4

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

         Friedman rank sum test

data:  pomme.vec, traitement and bloc
Friedman chi-square = 9.8, df = 4, p-value = 0.04393
```

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
           Df Sum Sq Mean Sq F value Pr(>F)
traitement  4 133419   33355    9.58 0.0010 **
bloc        3  14987    4996    1.43 0.2814
Residuals  12  41797    3483
---
```

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

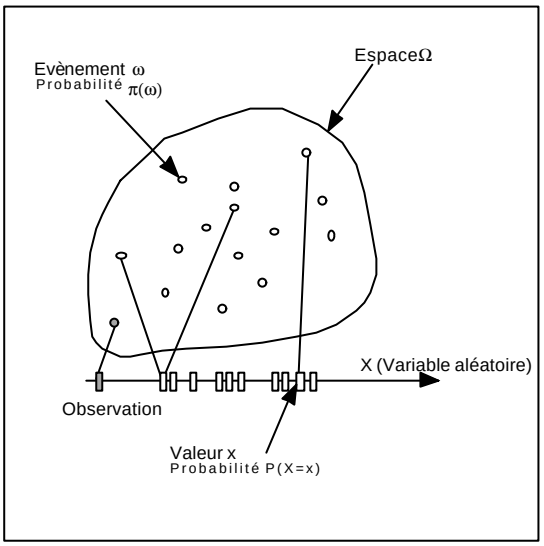
2.5. Test du Chi2 (op. cit. p. 84)

```
> 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

data:  fongi
X-squared = 53.72, df = 6, p-value = 8.402e-10
```

3. 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, à la psychologie rugueuse, 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 :

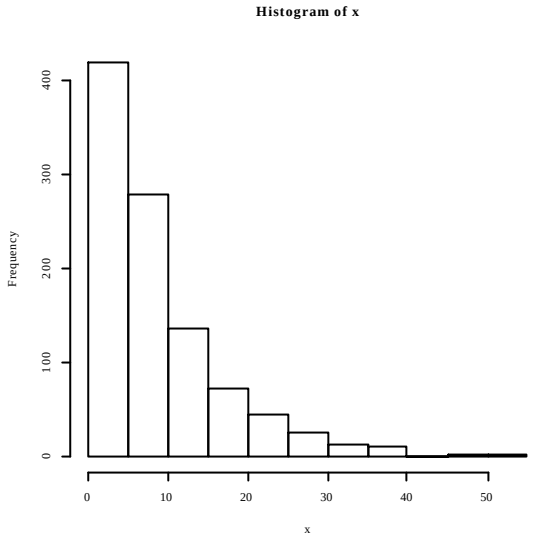
```
> sample(5,8,replace=T)
[1] 1 3 3 5 1 5 2 2
```

3.1. Quelle est la loi du plus petit ?

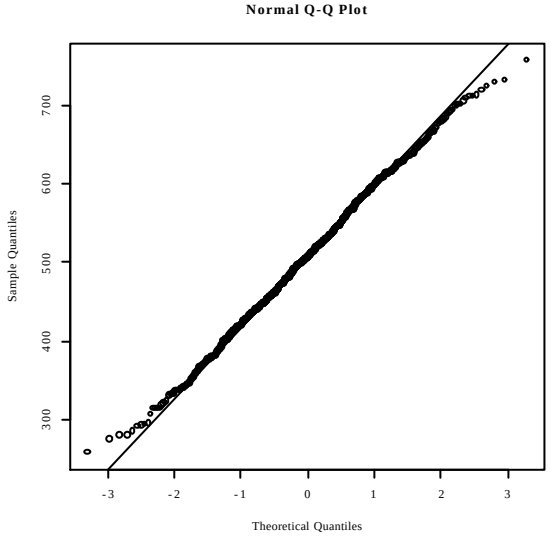
Quelle est la loi du plus petit, de la somme, de la variance, du plus grand, de 10 entiers tirés au hasard parmi 100 sans remise ?
Implanter la fonction :

```
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
> echa(min,10)
 [1] 12 18  2  4  1  3  4 19 12  3
> x<-echa(min,1000)
> hist(x,nclass=10)
> hist(x,nclass=10,plot=F)
$breaks
 [1]  0  5 10 15 20 25 30 35 40 45 50 55

$counts
 [1] 418 279 136  73  45  24  12  10   0   2   1

$intensities
```

```
 [1] 0.0836 0.0558 0.0272 0.0146 0.0090 0.0048 0.0024 0.0020 0.0000 0.0004
[11] 0.0002

$mids
 [1]  2.5  7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5
```




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



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

Pour comprendre les qqplot (graphes quantiles-quantiles), il faut savoir qu'une loi de probabilités définit la probabilité de ne pas dépasser une valeur donnée. Pour la loi normale (de moyenne 0 et de variance 1), on a 2.5 chances sur 100 de ne pas 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 de n'être pas dépassée est le quantile d'ordre p :

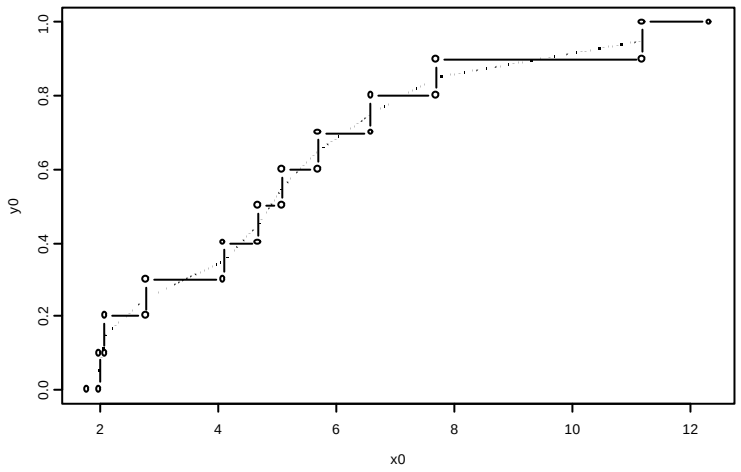
```
> qnorm(0.025)
[1] -1.96
> qnorm(0.975)
[1] 1.96
```

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 :

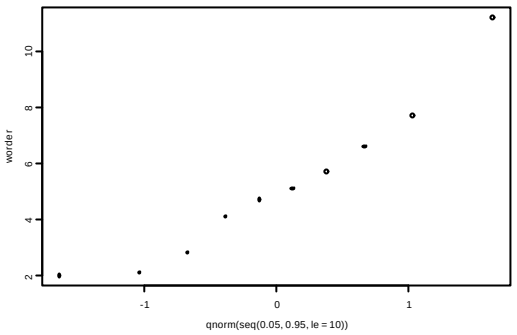
```
> worder<-sort(w)
> worder
 [1]  2.0  2.1  2.8  4.1  4.7  5.1  5.7  6.6  7.7 11.2

> 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

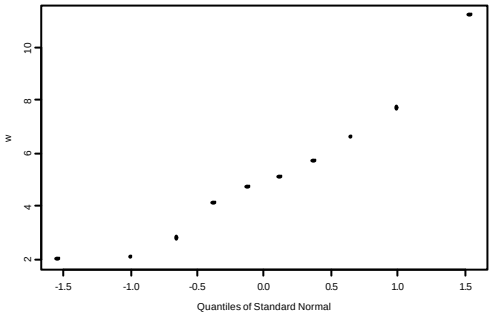
points sont sur une droite. C'est l'équivalent du tracé de la droite de Henry sur papier gaussien de nos ancètres (remarque : il y en a encore dans l'armoire).

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



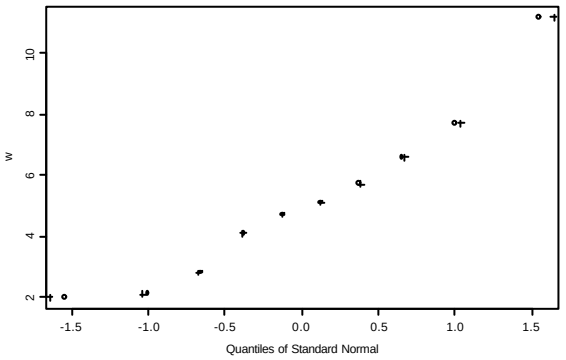
Ceci est obtenu directement par :

```
> 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.

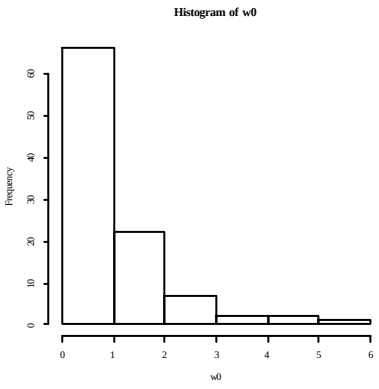
REFERENCES

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
```

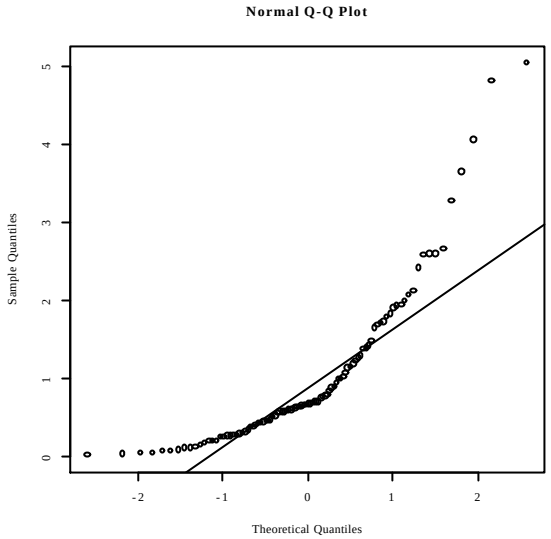
Ouf ! La documentation est transparente. Il faut retenir de œtte expérience les capacités considérables que R propose pour l'étude des distributions. Par exemple, générer un échantillon d'une loi exponentielle :

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



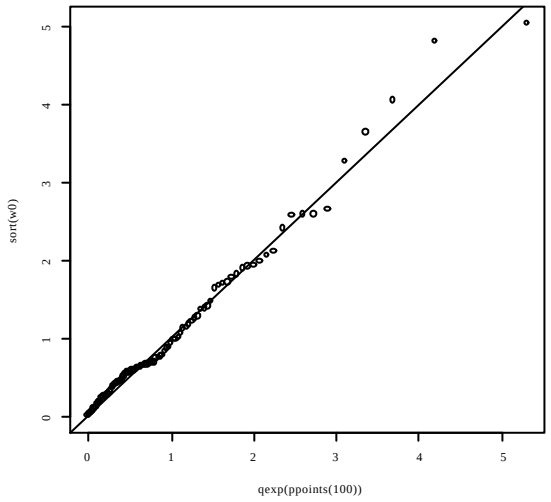
La distribution n'est pas gaussienne (et pour cause) :

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



Elle est exponentielle :

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



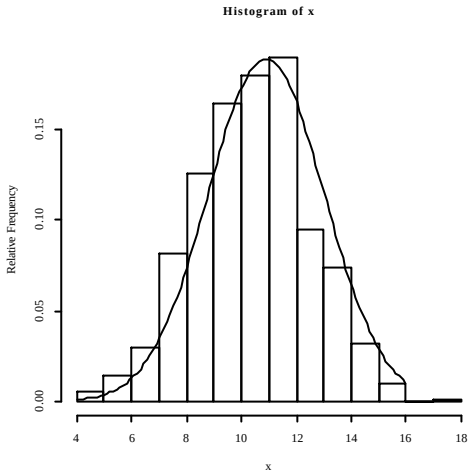
3.2. Le nombre de suites est gaussien pour N>=10 et N-M>=10 ?

```
> 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) {
```

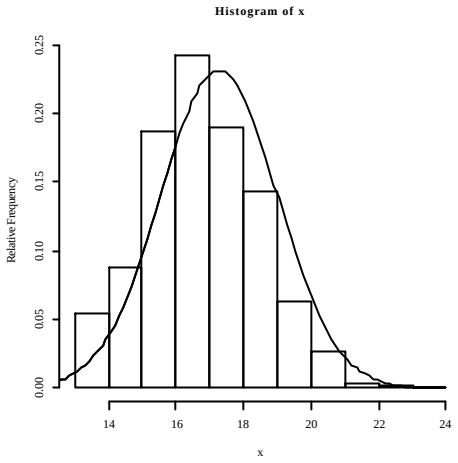
```
            y<-rep(0,le=n)
            z<-sample(1:n, m, replace = F)
            y[z]<-1
            x[i]<-(sum(abs(diff(y))))+1
      }
      return (x)
}

> 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.

3.3. 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)
> sum(x<=14)/1000
[1] 0.041
```



3.4. Trois œufs dans cinq cocons

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

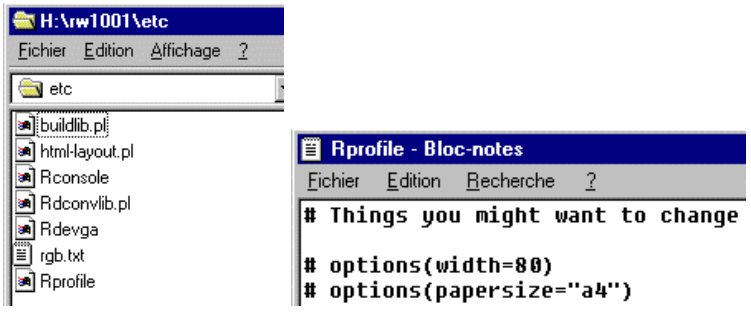
On a observé 19,31,20 : c'est donc clair !

4. Les techniques de ré échantillonnage

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 domaine des méthodes de rééchantillonnage en exercices imposés.

4.1. Installer une librairie

Il faut d'abord étendre la version de base de R. Noter d'abord l'opération suivante. On voudrait qu'en lançant R, le programme édite le nom du dossier de travail, qu'il utilise l'options d'édition à quatre chiffres significatifs, qu'il branche la librairie ctest, ... selon ses propres intentions. Pour ce faire, quitter R et dans le dossier d'installation ouvrir le dossier etc, puis le fichier Rprofile avec un éditeur :



C’est tout prêt. Rajouter ses desiderata :

```
# options(winhelp=TRUE)

options(editor="C:\\WINDOWS\\WORDPAD.EXE")
options (digits=4)
options("editor")
options ("digits")
paste("Dossier de travail =",getwd(),sep=" ")
library(ctest)
library(mva)
```

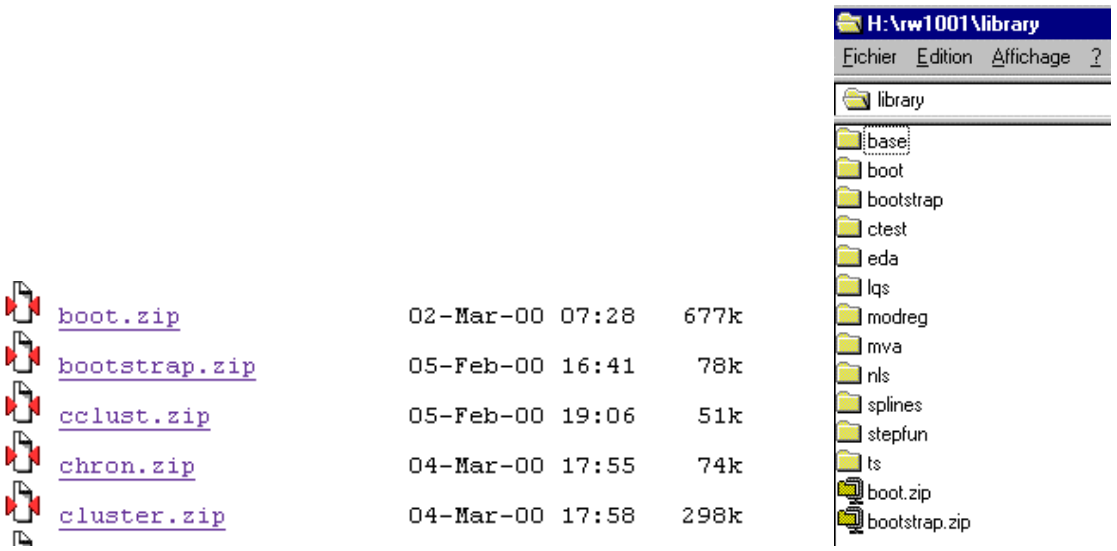
Fermer, sauvegarder et relancer R :

```
Version 1.0.1  (April 14, 2000)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type    "?license" or "?licence" for distribution details.

R is a collaborative project with many contributors.
Type    "?contributors" for a list.

Type    "demo()" for some demos, "help()" for on-line help, or
        "help.start()" for a HTML browser interface to help.
Type    "q()" to quit R.

[Previously saved workspace restored]

$editor
[1] "C:\\WINDOWS\\WORDPAD.EXE"

$digits
[1] 4

[1] "Dossier de travail = E:\\Rprojet"
> ?wilcox.test vérifie que la librairie ctest est disponible
```

Avec un navigateur, aller à

http://www.stat.math.ethz.ch/R-CRAN/bin/windows/windows-NT/contrib/

Chercher boot et bootstrap :



Charger les fichiers boot.zip et bootstrap.zip (et tout ce que vous voulez). Placer les fichiers dans le dossier \library du dossier R et décompacter dans le même dossier en gardant la structure de dossiers. Les librairies sont disponibles.

Aller ensuite à :

http://www.ci.tuwien.ac.at/R/src/contrib/PACKAGES.html

boot

functions and datasets for bootstrapping from the book "Bootstrap Methods and Their Applications" by A.C. Davison and D.V. Hinkley (1997, CUP).

Version: 1.1-5

Depends: R (>= 0.99)

Author: S original by Angelo Canty. R port by Brian Ripley <ripley@stats.ox.ac.uk>.

License: Unlimited distribution.

Index of Contents (Text)
Reference Manual (PDF)

bootstrap

Software (bootstrap, cross-validation, jackknife), data and errata for the book "An Introduction to the Bootstrap" by B. Efron and R. Tibshirani, 1993, Chapman and Hall.

Version: 1.0-5

Author: S original by Rob Tibshirani <tibs@utstat.toronto.edu>. R port by Fritz Leisch <Friedrich.Leisch@ci.tuwien.ac.at>.

License: ??? (S original from StatLib)

Index of Contents (Text)
Reference Manual (PDF)

Charger les manuels en PDF .

```
> library(bootstrap)
> 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)
```

4.2. bootstrap

```
>? bootstrap
bootstrap              package:bootstrap              R Documentation
```

```
Non-Parametric Bootstrapping

Usage:

     bootstrap(x,nboot,theta,..., func=NULL)

Arguments:

       x: a vector containing the data. To bootstrap more complex data
          structures (e.g. bivariate data) see the last example below.

   nboot: The number of bootstrap samples desired.

   theta: function to be bootstrapped. Takes `x' as an argument, and
          may take additional arguments (see below and last example).

     ...: any additional arguments to be passed to `theta'

    func: (optional) argument specifying the functional the
          distribution of thetahat that is desired.  If func is
          specified, the jackknife after-bootstrap estimate of its
          standard error is also returned. See example below.

References:

     Efron, B. and   Tibshirani, R. (1986).  The bootstrap method for
     standard errors, confidence intervals, and other measures of
     statistical accuracy. Statistical Science, Vol 1., No. 1, pp 1-35.

     Efron, B. (1992) Jackknife-after-bootstrap standard errors and
     influence functions. J. Roy. Stat. Soc. B, vol 54, pages 83-127

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

```
> theta <- function(x,xdata){ cor(xdata[x,1],xdata[x,2]) }
> bootstrap(1:10,20,theta,tabac)
$thetastar
 [1] 0.9707 0.6347 0.8606 0.8419 0.9290 0.9329 0.9223 0.9144 0.9077 0.9686
[11] 0.8928 0.9739 0.8592 0.9198 0.7516 0.9533 0.8957 0.9374 0.5487 0.9278

$call
bootstrap(x = 1:10, nboot = 20, theta = theta, tabac)
```

Il s'est passé la procédure suivante. 20 fois un échantillon de 1:10 de longueur 10 a été sélectionné avec remise comme dans :

```
> sample(1:10,10,replace=T)
 [1]  1  7  1  2 10  5  1  7  7  9
```

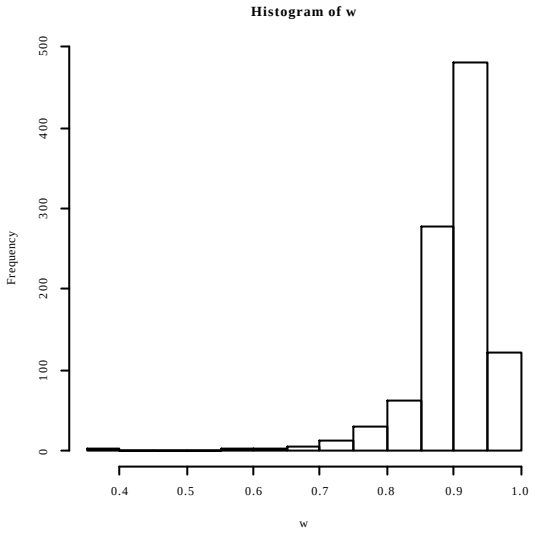
En général, certains points figurent plusieurs fois alors que d'autres sont absents.

```
> s0_sample(1:10,10,replace=T)
> s0
 [1] 9 3 3 1 2 6 4 7 3 4
```

Pour un tirage la statistique est calculée sur la sélection :

```
> tabac[s0,]
      goudron nicotine
 [1,]    0.55      8.5
 [2,]    1.07     14.0
 [3,]    1.07     14.0
 [4,]    0.45     11.0
 [5,]    0.77     13.0
```

```
 [6,]    1.15     14.5
 [7,]    1.03     15.0
 [8,]    1.14     18.0
 [9,]    1.07     14.0
[10,]    1.03     15.0
> cor(tabac[s0,1],tabac[s0,2])
[1] 0.845

> w_bootstrap(1:10,1000,theta,tabac)$thetastar
> hist(w)
```

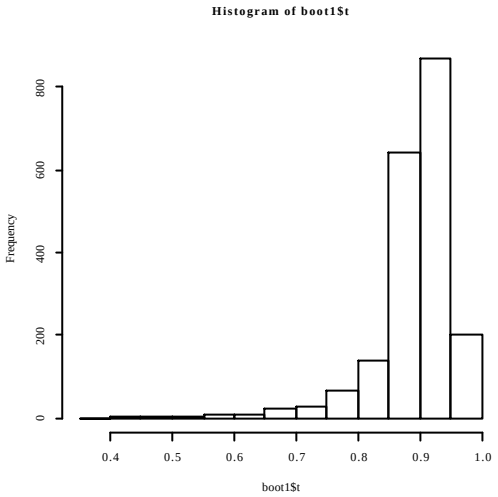



```
> mean(w)
[1] 0.8979
> range(w)
[1] 0.3727 0.9954
```

Le principe du bootstrap est de baser la mesure de l'erreur d'échantillonnage sur le seul échantillon.

```
> library(boot)

boot                  package:boot                  R Documentation

Bootstrap Resampling

Description:

     Generate `R' bootstrap replicates of a statistic applied to data.
     Both parametric and nonparametric resampling are possible.  For
     the nonparametric bootstrap, possible resampling methods are the
     ordinary bootstrap, the  balanced bootstrap, antithetic
     resampling, and permutation. For nonparametric multi-sample
     problems stratified resampling is used.   This is specified by
     including a vector of strata in the call to boot. Importance
     resampling weights may be specified.

Usage:

     boot(data, statistic, R, sim="ordinary", stype="i",
          strata=rep(1,n), L=NULL, m=0, weights=NULL,
          ran.gen=function(d, p) d, mle=NULL, ...)

> f1_function(x,n){cor(x[n,1],x[n,2])}
> boot1_boot(tabac,f1,2000)
> boot1

ORDINARY NONPARAMETRIC BOOTSTRAP
```

```
Call:
boot(data = tabac, statistic = f1, R = 1000)


Bootstrap Statistics :
    original   bias    std. error
t1*   0.8902 0.003731     0.06673
> summary(boot1)
          Length Class  Mode
t0           1   -none- numeric
t         1000   -none- numeric
...
> hist(boot1$t)
```




```
> wilcox.test(w,boot1$t)

         Wilcoxon rank sum test with continuity correction

data:  w and boot1$t
W = 515443, p-value = 0.2318
alternative hypothesis: true mu is not equal to 0
```

Ce sont bien des échantillons compatibles. On a aussi :

```
> boot.ci(boot1, type="bca")
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot1, type = "bca")

Intervals :
Level       BCa
95%   ( 0.5060,  0.9474 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
```

Dans le help de boot.ci

```
References:

     Davison, A.C. and Hinkley, D.V. (1997)  Bootstrap Methods and
     Their Application, Chapter 5. Cambridge University Press.

     DiCiccio, T.J. and Efron  B. (1996) Bootstrap confidence intervals
     (with  Discussion). Statistical Science, 11, 189-228.

     Efron, B. (1987) Better bootstrap confidence intervals (with
```

```
     Discussion). Journal of the American Statistical Association, 82,
     171-200.
```

Dans le help de boot

```
References:

     There are many references explaining the bootstrap and its
     variations. Among them are :

     Booth, J.G., Hall, P. and Wood, A.T.A. (1993) Balanced importance
     resampling  for the bootstrap. Annals of Statistics, 21, 286-298.

     Davison, A.C. and Hinkley, D.V. (1997)  Bootstrap Methods and
     Their Application. Cambridge University Press.

     Davison, A.C., Hinkley, D.V. and Schechtman, E. (1986) Efficient
     bootstrap  simulation. Biometrika, 73, 555-566.

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

     Gleason, J.R. (1988) Algorithms for balanced bootstrap
     simulations. American Statistician, 42, 263-266.

     Hall, P. (1989) Antithetic resampling for the bootstrap.
     Biometrika, 73, 713-724.

     Hinkley, D.V. (1988) Bootstrap methods (with Discussion).  Journal
     of the  Royal Statistical Society, B, 50, 312-337, 355-370.

     Hinkley, D.V. and Shi, S. (1989) Importance sampling and the
     nested bootstrap. Biometrika, 76, 435-446.

     Johns M.V. (1988) Importance sampling for bootstrap confidence
     intervals. Journal of the American Statistical Association, 83,
     709-714.

     Noreen, E.W. (1989) Computer Intensive Methods for Testing
     Hypotheses.  John Wiley & Sons.
```

Bon courage.

4.3. jackknife

```
> ? jackknife

jackknife             package:bootstrap             R Documentation

Jackknife Estimation

Usage:

     jackknife(x, theta, ...)

Arguments:

       x: a vector containing the data. To jackknife  more complex data
          structures (e.g. bivariate data) see the last example below.

   theta: function to be jackknifed. Takes `x' as an argument, and may
          take additional arguments (see below and last example).

     ...: any additional arguments to be passed to `theta'
```

```
> jackknife(1:10,theta,tabac)
$jack.se
[1] 0.03848

$jack.bias
[1] 0.01916

$jack.val
 [1] 0.9013 0.8907 0.9011 0.8892 0.8763 0.9059 0.9146 0.8896 0.8710 0.8834

$call
jackknife(x = 1:10, theta = theta, tabac)


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

Le principe du jackknife est d'énumérer tous les sous-échantillons de l'échantillon observé obtenus par élimination d'un seul point. Si l'échantillon de départ contient *n* points, il y a *n* sous-échantillons de taille *n* - 1. Cela sert à voir si la suppression d'un point influe beaucoup sur la statistique étudiée.

Il y a plusieurs dizaines de librairies.

```
> library(help=bootstrap)
> library(help=boot)
```

La commande ouvre la documentation globale de la librairie. De quoi s'instruire.