TP no 4 - Tests par permutation

Master parcours SSD - UE Statistique Computationnelle

Septembre 2019

1 Exercice 1 - tests de permutation et différence de moyennes

Dans cet exercice nous allons mettre en oeuvre une procédure de permutation pour faire un test de différence de moyennes. Pour cela nous travaillerons sur le jeu "chicken weights" qui vise à comparer l'efficacité de différents aliments sur la prise de poids de poulets.

1. *Charger le jeu de données via la commande* `attach(chickwts)`*, ce qui a pour effet de charger deux variables :* `weight` *et* `feed`*. De combien d'observations dispose t'on pour chaque type de nourriture ? Tracer la distribution du poids en fonction du type de nourriture.*

2. *Nous voulons tester s'il y a une différence de poids significative entre des poulets nourris avec* `soybean` *et* `linseed`*. Au vu du graphique précédent, pensez-vous que cette différence soit significative ? Quelle est la statistique du test de Student et la p-valeur associée ?*

3. *Combien de permutations faudrait-il considérer pour faire un test par permutation exact?*

4. *Implémenter la procédure de permutation "randomisée" décrite en cours pour un nombre* $B = 1000$ *de tirages. Quelle p-valeur obtenez-vous ? Représenter la distribution de la statistique de test obtenue lors des permutations et positionner la valeur observée sur l'échantillon.*

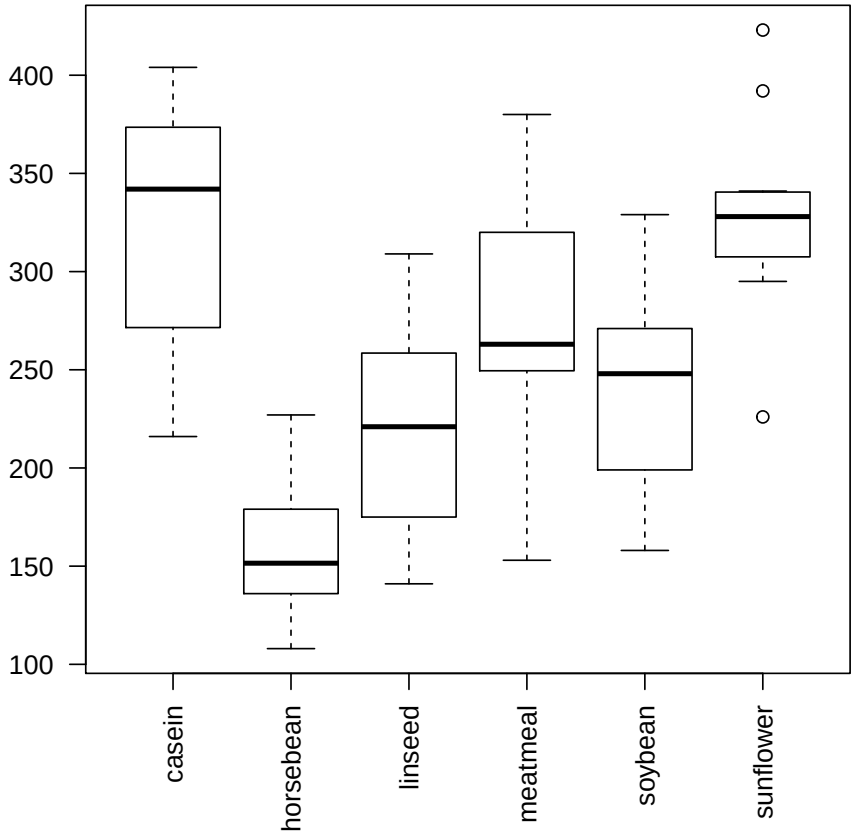
5. *Représenter l'évolution de la p-valeur estimée quand on passe de* $B = 100$ *à* $B = 2000$ *tirages par pas de 100. La procédure converge t'elle rapidement ?*

6. *Faire la même analyse pour comparer l'alimentation par* `linseed` *et* `sunflower` *en considérant* $B = 1000$*.*

On commence par charger le jeu de données et afficher le poids en fonction de l'alimentation.

```
> ########################
> #### STARTING EXO 1 ####
> ########################
> attach(chickwts)
> boxplot(weight ~ feed, las = 2)
> print(table(feed))
```

```
feed
   casein horsebean   linseed  meatmeal   soybean sunflower
       12        10        12        11        14        12
```



Au vu de ce graphique, les alimentations `soybean` et `linseed` semblent comparables. Le test de Student nous le confirme.

```
> x = weight[feed == "soybean"]
> y = weight[feed == "linseed"]
> ttest.res = t.test(x,y)
> t0 = ttest.res$statistic
> p0 = ttest.res$p.value
> cat("*** linseed vs soybean : p.val of standard t.test =", p0, "***\n")
```

```
*** linseed vs soybean : p.val of standard t.test = 0.1979871 ***
```

```
> cat("*** linseed vs soybean : test statistic =", t0, "***\n")
```

```
*** linseed vs soybean : test statistic = 1.324556 ***
```

Le nombre de permutations est égal à C_N^k, oú k est la longeur de x (soit ici 14) et N est la somme des longeurs de x et y (soit ici 26).

Le nombre de combinaisons peut être obtenu en `R` via la fonction `choose`, et vaut donc $N!/(k!(N-k)!)$.

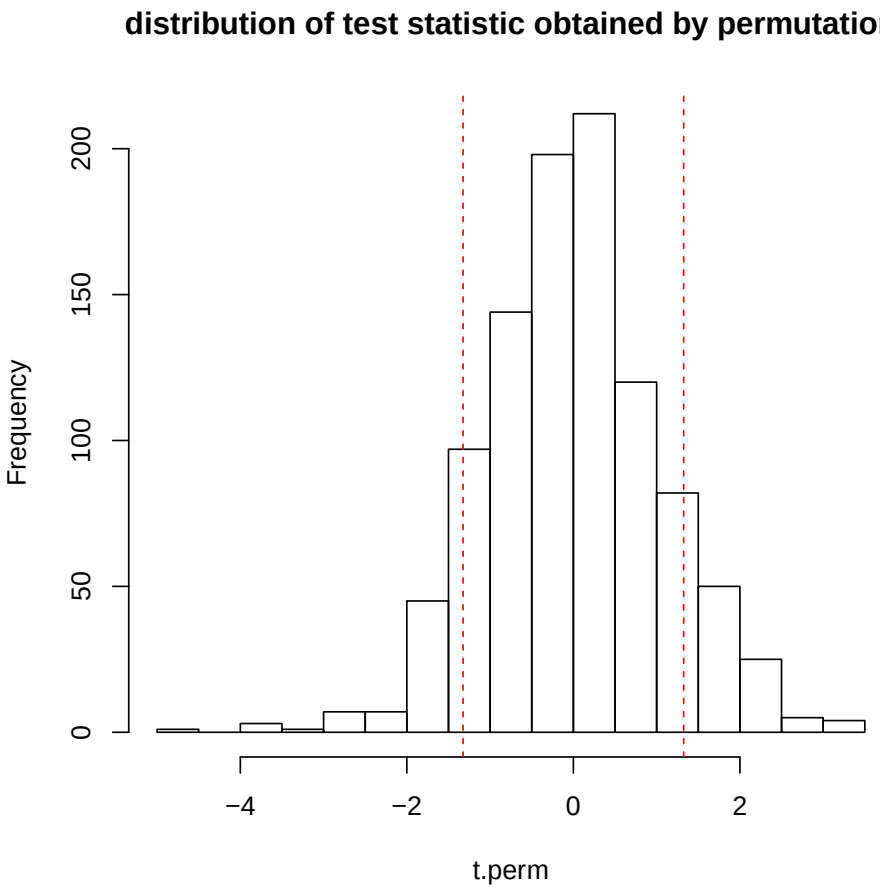
```
> k = length(x)
> N = length(x)+length(y)
> comb = choose(N,k)
> cat("*** total number of permutations =", comb, "***")
```

```
*** total number of permutations = 9657700 ***
```

Le code ci-dessous implémente la procédure de permutation et affiche les résultats. Notons qu'ici on considère une hypothèse alternative bilatétale. Par conséquent, il convient de calculer la proportion de tirages dont la stastique de test est supérieure à l'observée, *en valeur absolue*. Il faut donc sensiblement adapter la procédure décrite dans le cours, qui est définie pour un hypothèse alternative unilatérale supérieure (ou alors spécifier à la fonction `t.test` que notre hypothèse alternative est `"greater"` pour qu'il le prenne en compte dans le calcul de la p-valeur).

```
> z = c(x,y)
> set.seed(27)
> B = 1000
> t.perm = numeric(B)
> t0 = t.test(x, y)$statistic
> for(b in 1:B){
+   z1 = sample(z, replace = F)
+   x1 = z1[1:length(x)]
+   y1 = z1[(length(x)+1):length(z1)]
+   t.perm[b] = t.test(x1, y1)$statistic
+ }
> t.perm = c(t.perm, t0)
> hist(t.perm, breaks = 20, main = "distribution of test statistic obtained by permutation")
> abline(v = t0, col = 2, lty = 2)
> abline(v = -t0, col = 2, lty = 2)
> pval = mean(abs(t.perm) >= abs(t0))
> cat("*** linseed vs soybean : p.val obtained by permutation =", pval, "***\n")
```

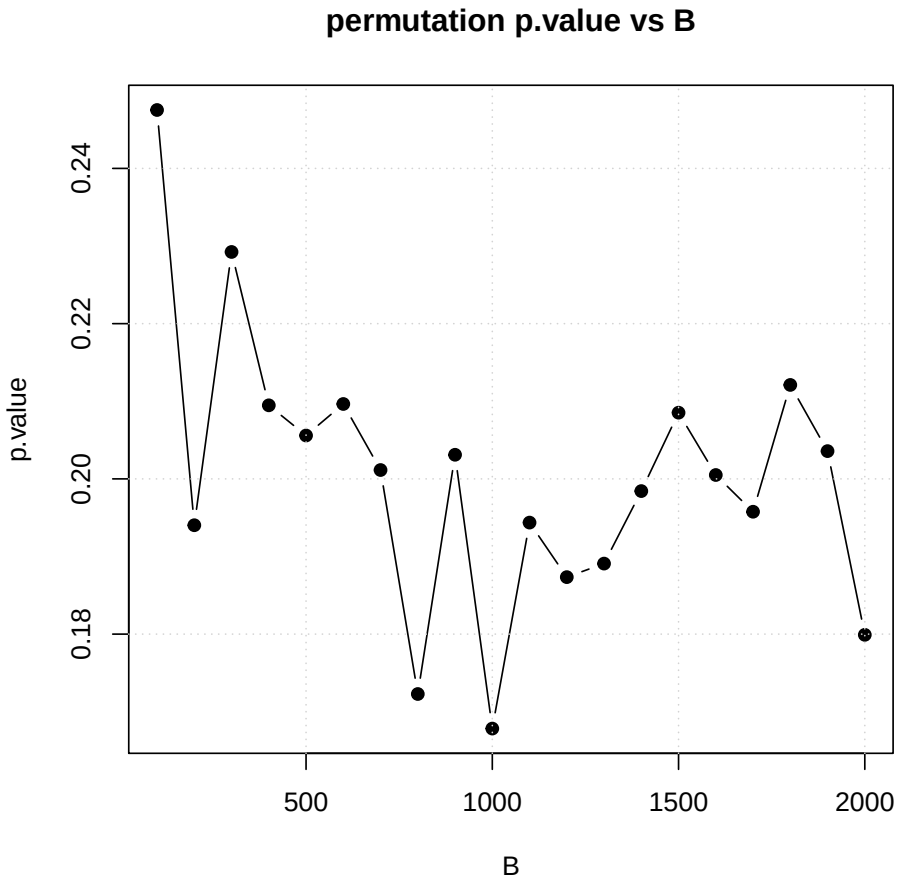
```
*** linseed vs soybean : p.val obtained by permutation = 0.1988012 ***
```



Le code ci-dessous évalue l'impact du nombre de tirages.

```
> B.list = seq(100,2000, by = 100)
> pval.list = c()
> t0 = t.test(x, y)$statistic
> for(B in B.list){
+   t.perm = numeric(B)
+   for(b in 1:B){
+     z1 = sample(z, replace = F)
+     x1 = z1[1:length(x)]
+     y1 = z1[(length(x)+1):length(z1)]
+     t.perm[b] = t.test(x1, y1)$statistic
+   }
+   t.perm = c(t.perm, t0)
+   pval.list = c(pval.list, mean( abs(t.perm) >= abs(t0)))
+ }
> plot(B.list, pval.list, xlab = "B", ylab = "p.value", main = "permutation p.value vs B", t
```

```
> grid()
```



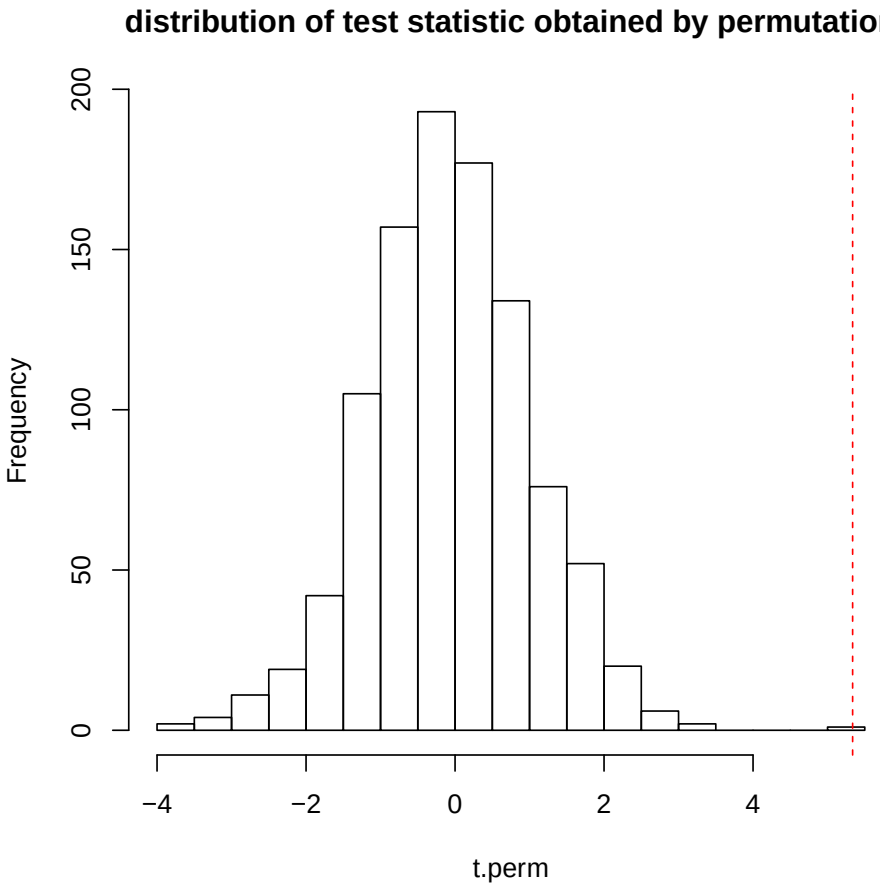
On constate que la procédure converge assez vite.

Enfin, le code ci-dessous reproduit la même analyse pour comparer les alimentations `linseed` et `sunflower`, pour lesquelles on suspecte une différence.

```
> x = weight[feed == "sunflower"]
> y = weight[feed == "linseed"]
> ttest.res = t.test(x,y)
> t0 = ttest.res$statistic
> p0 = ttest.res$p.value
> cat("*** linseed vs sunflower : p.val of standard t.test =", p0, "***\n")

*** linseed vs sunflower : p.val of standard t.test = 2.374104e-05 ***

> z = c(x,y)
> set.seed(27)
```

```
> B = 1000
> t.perm = numeric(B)
> t0 = t.test(x, y)$statistic
> for(b in 1:B){
+   z1 = sample(z, replace = F)
+   x1 = z1[1:length(x)]
+   y1 = z1[(length(x)+1):length(z1)]
+   t.perm[b] = t.test(x1, y1)$statistic
+ }
> t.perm = c(t.perm, t0)
> hist(t.perm, breaks = 20, main = "distribution of test statistic obtained by permutation")
> abline(v = t0, col = 2, lty = 2)
> pval = mean(abs(t.perm) >= abs(t0))
> cat("*** linseed vs sunflower : permutation p.val =", pval, "***\n")
```

```
*** linseed vs sunflower : permutation p.val = 0.000999001 ***
```



2 Exercice 2 - tests de permutation et dé pipé

A l'issue de multiples lancés d'un même dé, on a obtenu les résultats suivants :

face	*1*	*2*	*3*	*4*	*5*	*6*
tirages	*8*	*9*	*19*	*6*	*8*	*10*

On souhaite tester si ce dé est pipé, i.e., s'il conduit à obtenir certaine(s) face(s) plus fréquemment qu'attendu.

1. *Quel test paramétrique peut-on utiliser et avec quelle hypothèse nulle ? Donner une définition intuitive de la statistique de test.*
2. *Quelle p-valeur obtient-on en appliquant la fonction* `R` *correspondante ?*
3. *Appliquer une procédure par permutation (randomisée) pour* $B = 1000$ *tirages.*

On commence par créer notre vecteur d'observations.

```
> ########################
> #### STARTING EXO 2 ####
> ########################
> rm(list = ls())
> x = c(8,9,19,6,8,10)
```

Le test à utiliser est typiquement un test du χ^2, qu'on peut réaliser en `R` via la fonction `chisq.test`. L'hypothèse nulle ici est que la distribution multinomiale est uniforme, i.e., que chaque face du dé a une probabilité 1/6 de sortir.

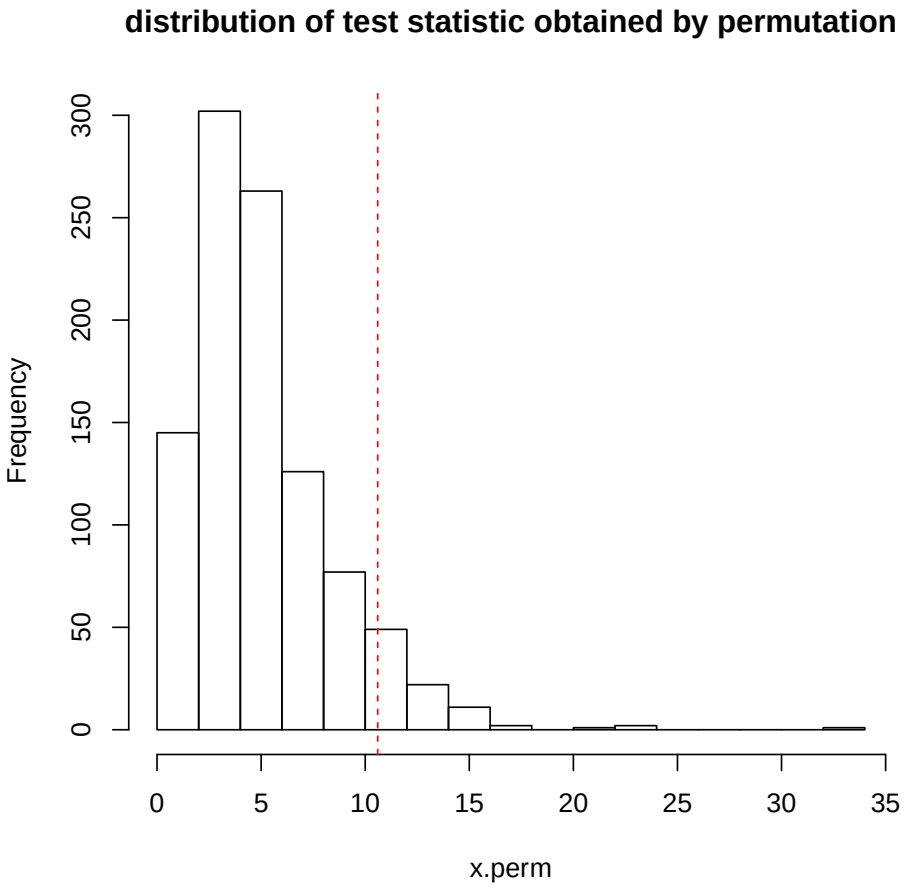
```
> chitest = chisq.test(x)
> p0 = chitest$p.value
> x0 = chitest$statistic
> cat("*** pval obtained by standard test =", p0, "***\n")
```

```
*** pval obtained by standard test = 0.05991363 ***
```

La procédure ci-dessous réalise le test par permutation. Attention, il ne faut pas simplement permuter le vecteur x (auquel cas la statistique de test reste inchangée), mais bien simuler 60 lancés de dé (60 étant la somme des valeurs contenues dans le vecteur x) avec une loi multinomiale uniforme.

```
> B = 1000
> x.perm = numeric(B)
> for(b in 1:B){
+   x1 = table( sample(c(1:6), sum(x), prob = rep(1/6,6), replace = T) )
+   x.perm[b] = chisq.test(x1)$statistic
+ }
```

```
> x.perm = c(x.perm, x0)
> hist(x.perm, breaks = 20, main = "distribution of test statistic obtained by permutation")
> abline(v = x0, col = 2, lty = 2)
> pval = mean(x.perm >= x0)
> cat("*** pval obtained by permutation procedure =", pval, "***\n")

*** pval obtained by permutation procedure = 0.07092907 ***
```



3 Exercice 3 - tests de permutation et corrélation

Reprendre les données `LSAT` *et* `GPA` *du TP précédent*[1]. *On s'intéressera de nouveau à la corrélation entre ces deux variables.*

1. *Quelle corrélation observe t'on sur ce jeu de données entre ces deux variables ?*

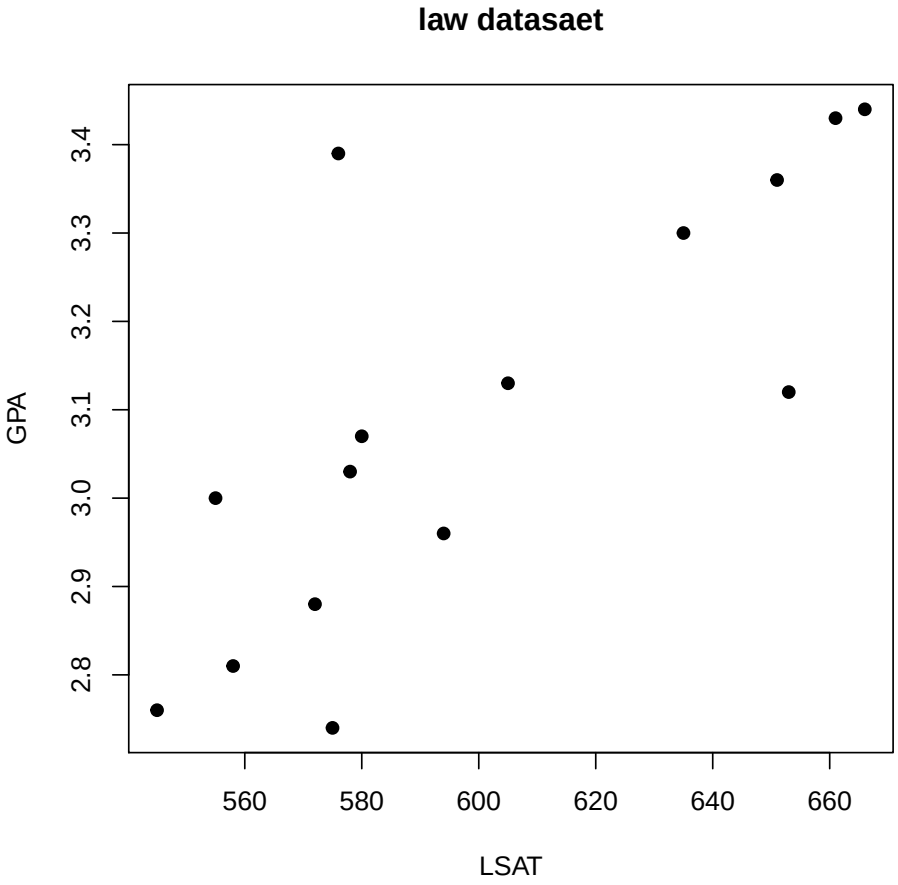
[1] On les retrouve en chargeant le package `bootstrap` et en venant chercher les variables du même nom dans la data.frame `law`.

2. *Comment mettre en oeuvre une stratégie de permutation pour tester l'hypothèse (nulle) que cette corrélation est égale à zéro ?*

3. *Peut-on considérer qu'elle est significativement différente de zéro selon ce test par permutation ?*

On commence par charger et représenter nos données.

```
> ########################
> #### STARTING EXO 3 ####
> ########################
> library(bootstrap)
> x = law$LSAT
> y = law$GPA
> plot(x, y, pch = 19, xlab = "LSAT", ylab = "GPA", main = "law datasaet")
```



On mesure la corrélation suivant.

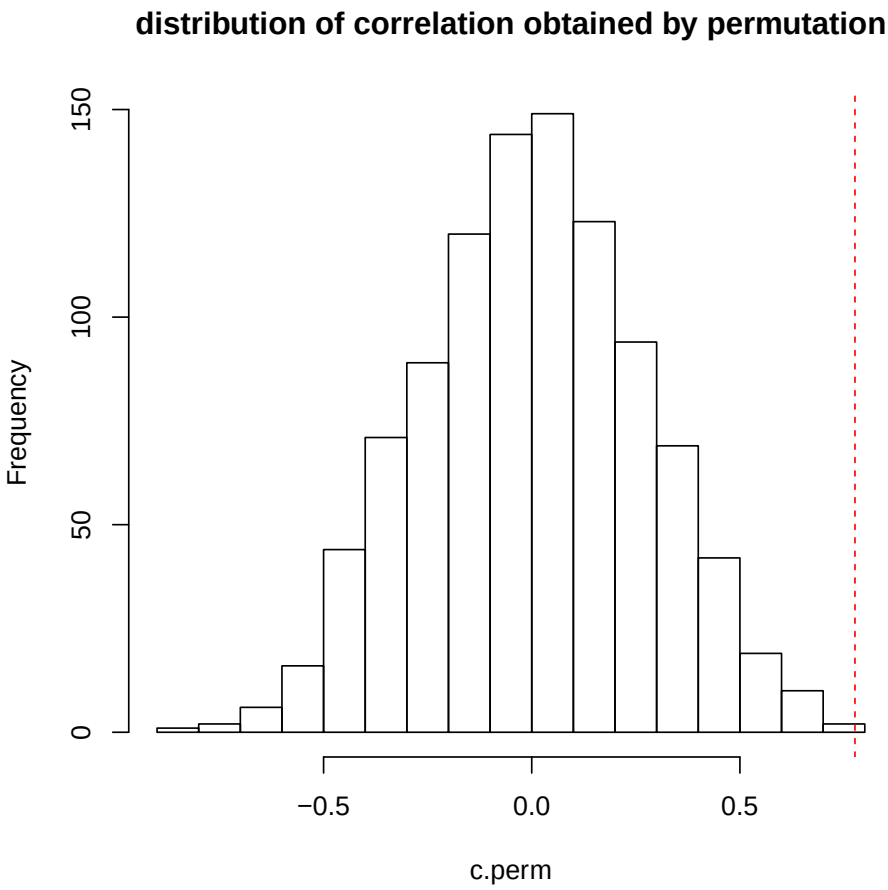
```
> c0 = cor(x,y)
> cat("*** correlation observed =", c0, "***\n")
```

```
*** correlation observed = 0.7763745 ***
```

Pour appliquer une procédure de permutation, on peut garder un des deux vecteurs originaux, et permuter le second. Ainsi, on "casse" la corrĺation (éventuelle) présente au sein du jeu de donnés. On s'attend donc à obtenir une distribution centrée sur zéro lors des permutations. La code suivant met en oeuvre une procédure par permutation et confirme qu'elle est significativement différente de zéro.

```
> set.seed(27)
> B = 1000
> c.perm = numeric(B)
> for(b in 1:B){
+   y1 = sample(y, replace = F)
+   c.perm[b] = cor(x, y1)
+ }
> c.perm = c(c.perm, c0)
> hist(c.perm, breaks = 20, main = "distribution of correlation obtained by permutation")
> abline(v = c0, col = 2, lty = 2)
> pval = mean(c.perm >= c0)
> cat("**** pval obtained by permutation =", pval, "***\n")
```

```
**** pval obtained by permutation = 0.000999001 ***
```



4 Exercice 4 - hypothèse d'échangeabilité

1. Reproduire l'expérience illustrant l'impact de l'hypothèse d'échangeabilité donnée dans le cours.
2. Tracer l'évolution de la puisssance estimée en fonction des valeurs de σ.
3. Comparer ces résultats avec le t-test. Ce test paramétrique s'en sort-il mieux ?

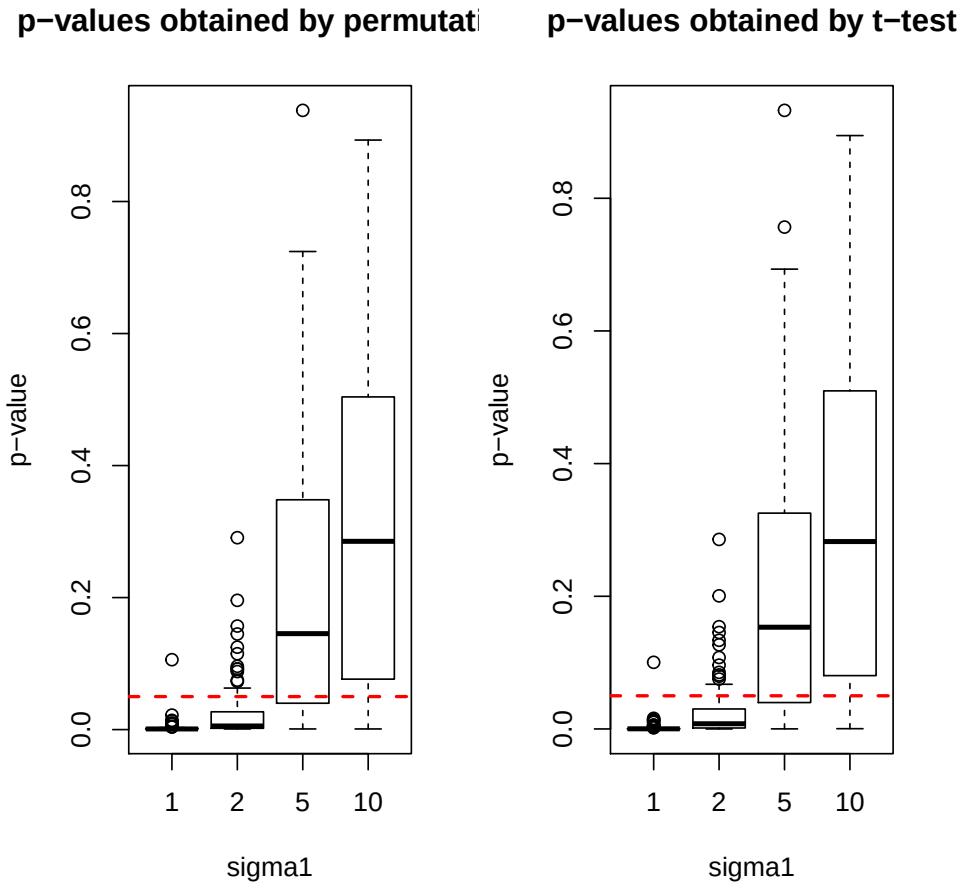
Le code ci-dessous réalise cette analyse. On boucle sur l'ensemble des configurations, en stockant à la fois les p-valeurs obtenues puis on trace (1) les distributions de p-valeurs et (2) les puissances obtenues.

On constate que le problème n'est pas spécifique au test par permutation.

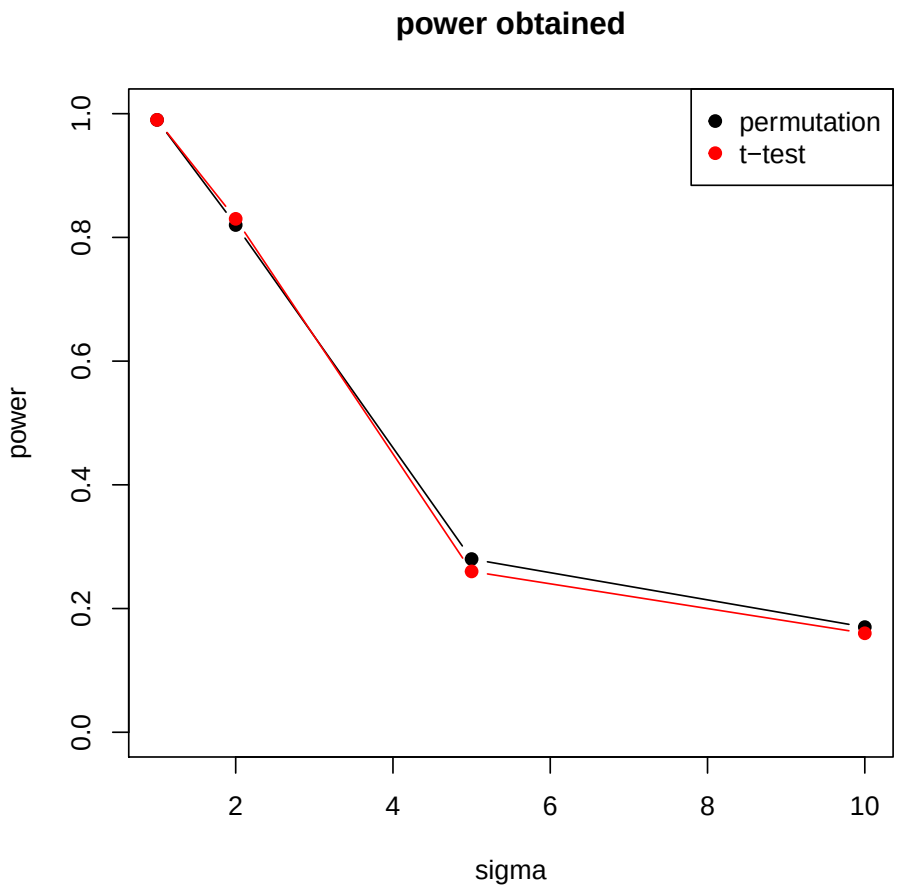
```
> ########################
> #### STARTING EXO 4 ####
> ########################
```

```
> mu0 = 0
> sigma0 = 1
> mu1 = 2
> sigma1 = 1
> B = 1000
> n = 10
> nrep = 100
> set.seed(27)
> PVAL = list()
> PVAL.T = list()
> sigma.list = c(1,2,5,10)
> for(sigma1 in sigma.list){
+   cat("*** processing sigma1 =", sigma1, "***\n")
+   pval = numeric(nrep)
+   pval.t = numeric(nrep)
+   for(r in 1:nrep){
+     x = rnorm(n, mu0, sigma0)
+     y = rnorm(n, mu1, sigma1)
+     z = c(x,y)
+     t.perm = numeric(B)
+     t0 = t.test(x, y)$statistic
+     for(b in 1:B){
+       ind = sample(length(z), n)
+       x1 = z[ind]
+       y1 = z[-ind]
+       t.perm[b] = t.test(x1,y1)$statistic
+     }
+     t.perm = c(t.perm,t0)
+     pval[r] = mean(t.perm <= t0)  # NB : here, we know that x is smaller
+     pval.t[r] = t.test(x, y, alternative = "less")$p.value # same --> use one-side
+   }
+   PVAL[[as.character(sigma1)]] = pval
+   PVAL.T[[as.character(sigma1)]] = pval.t
+ }


*** processing sigma1 = 1 ***
*** processing sigma1 = 2 ***
*** processing sigma1 = 5 ***
*** processing sigma1 = 10 ***


> # show pvalues
> par(mfrow = c(1,2))
> boxplot(PVAL, xlab = "sigma1", ylab = "p-value", main = "p-values obtained by permutation"
```

```
> abline(h = 0.05, col = 2, lty = 2, lwd = 2)
> boxplot(PVAL.T, xlab = "sigma1", ylab = "p-value", main = "p-values obtained by t-test")
> abline(h = 0.05, col = 2, lty = 2, lwd = 2)
> par(mfrow = c(1,1))
```




```
> # show power
> po.perm = sapply(PVAL, function(x){mean(x<0.05)})
> po.t = sapply(PVAL.T, function(x){mean(x<0.05)})
> plot(sigma.list, po.perm, ylim = c(0,1), type = "b", col = 1, pch = 19, xlab = "sigma", yl
> lines(sigma.list, po.t, type = "b", col = 2, pch = 19)
> legend("topright", c("permutation","t-test"), col = c(1,2), pch = 19)
```



References

T. Hastie, R. Tibshirani, and J.. Friedman. *The Elements of Statistical Learning*. Springer, 2001.