TP no 2 - Méthodes de Monte Carlo

Master parcours SSD - UE Statistique Computationnelle

Septembre 2019

1 Méthodes MC pour l'inférence

Exercice 1 (comparaison d'estimateurs)

Dans cet exercice[1] *nous allons évaluer la robustesse des estimateurs décrits dans le cours pour estimer la moyenne d'une loi normale, quand on s'éloigne des hypothèses de normalité. Pour cela, nous allons générer des les observations selon une loi normale* $\mathcal{N}(0, \sigma_1)$ *"contaminée" par seconde loi normale de variance différente* $\mathcal{N}(0, \sigma_2)$ *selon un modèle de mélange :*

$$p\mathcal{N}(0, \sigma_1) + (1 - p)\mathcal{N}(0, \sigma_2),$$

où $p \in [0, 1]$ *contrôle l'ampleur de cette contamination.*

1. *Implémenter une procédure pour simuler des observations selon ce modèle de mélange.*
   La fonction `rnorm` accepte des vecteurs comme argument pour la moyenne et l'écart type. Par conséquent, on peut commencer par tirer la valeurs σ_1 ou σ_2 à considérer pour chaque observation avant d'appeler la fonction `rnorm`. On peut le faire par exemple via la fonction `sample`.

   ```
   > ##############################
   > ##### STARTING EXERCICE 1 ####
   > ##############################
   > sigma1 = 1
   > sigma2 = 10
   > n = 20
   > p = 0.9
   > sigma.smple = sample(c(sigma1,sigma2), size = n, replace = TRUE, prob = c(p,1-p))
   > x = rnorm(n, mean = 0, sd = sigma.smple)
   ```

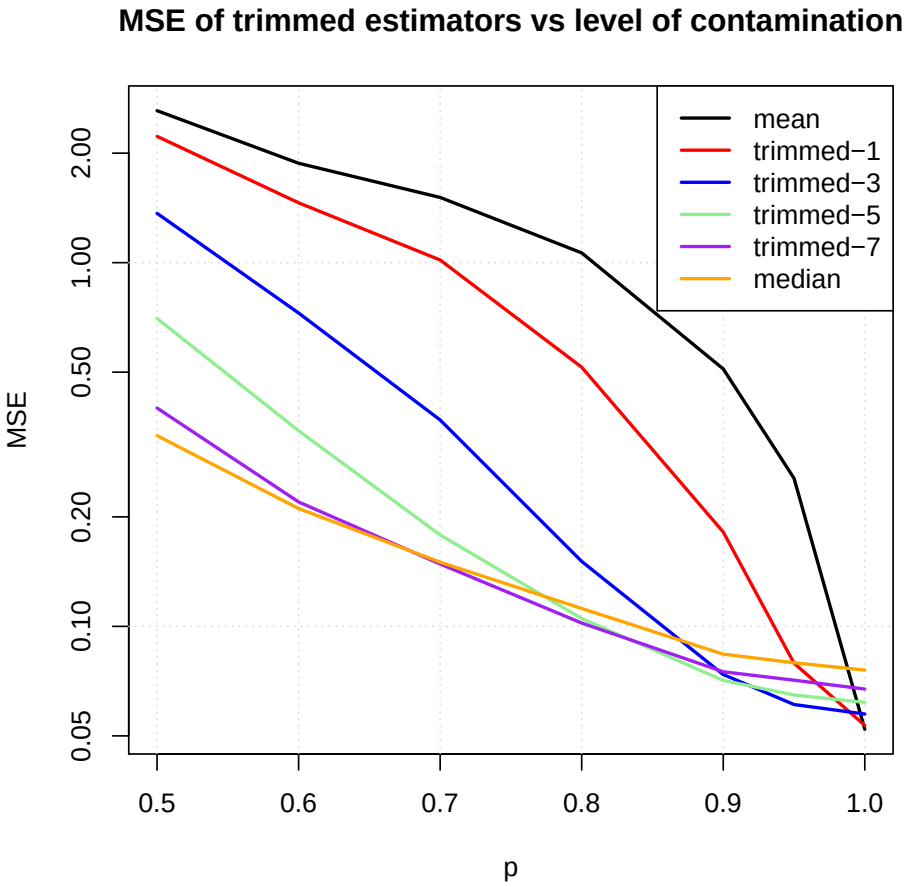
2. *En se basant sur la procédure décrite dans le cours, représenter l'évolution de la précision (quantifiée en terme d'erreur quadratique moyenne) des différents estimateurs de la moyenne proposés (moyenne empirique, moyenne empirique "trimmée" et médiane) pour* $\sigma_1 = 1$, $\sigma_2 = 10$, *et* $p \in \{1; 0.95; 0.9; 0.8; 0.7; 0.6; 0.5\}$. *Comment interpréter ces résultats ?*

1. Cet exercice est tiré directement du livre "Statistical Computing with R" de Maria L. Rizzo.

— *Question subsidiaire : que se passe-t-il si on fait varier p de 0 à 1 ?*

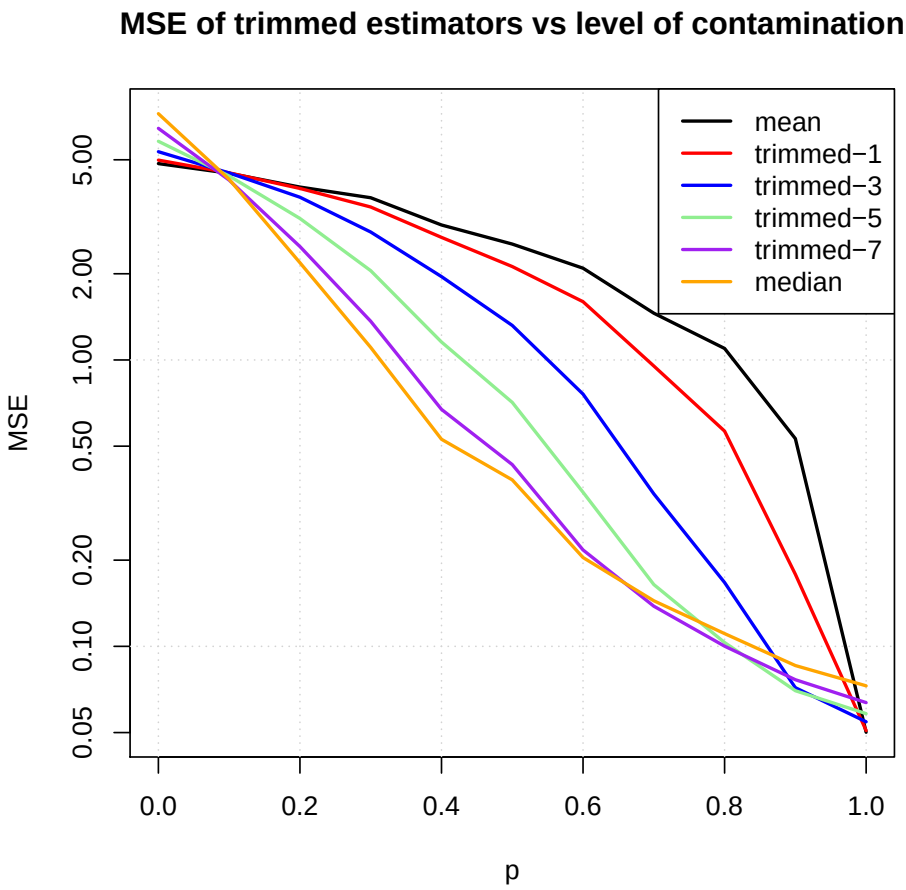
Le code ci-dessous réalise cette analyse pour différentes valeurs de k (définissant la moyenne empirique "trimmée").

```
> n = 20
> m = 1000
> p.list = c(1,0.95,0.9,0.8,0.7,0.6,0.5)
> k.list = c(1,3,5,7)
> MSE = c()
> SE = c()
> for(p in p.list){
+     tmp = replicate(m, expr = {
+       # generate sd to use
+       sigma = sample(c(1,10), size = n, replace = TRUE, prob = c(p,1-p))
+       # draw samples
+       x = sort(rnorm(n, mean = 0, sd = sigma))
+       # compute estimators
+       res = c(mean(x))
+       for(k in k.list){
+         res = c(res,mean(x[(k+1):(n-k)]))
+       }
+       res = c(res, median(x))
+       return(res)
+     })
+
+     mse = apply(tmp, 1, function(x){ mean(x^2) })
+     se = apply(tmp, 1, function(x){ sqrt( sum( (x - mean(x))^2 ) / m ) })
+
+     MSE = cbind(MSE, mse)
+     SE = cbind(SE, se)
+ }
> cols = c("black","red","blue","palegreen2", "purple","orange")
> plot(p.list, MSE[1,],  log = "y", ylim = range(MSE), col = cols[1], type = "l", xlab = "p'
> grid()
> for(i in 1:nrow(MSE)){
+   lines(p.list, MSE[i,], type = "l", col = cols[i], lwd = 2)
+ }
> legend("topright", c("mean",paste("trimmed-",k.list,sep=""),"median"), col = cols, lwd = 2
```



On note que quand le taux de contamination augmente (i.e., p décroit), l'efficacité de la moyenne empirique décroit au profit de la moyenne trimmée. On note également que la valeur de k optimale dépend du degré de contamination.

La figure suivante trace le mêmes indicateurs quand p varie de 0 à 1. On note qu'à mesure qu'on se rapproche de $p = 0$, l'écart entre les différents estimateurs se réduit. A $p = 0$ l'estimateur de la moyenne empirique redevient le plus performant, ce qui est attendu car on est de nouveau en présence d'une distribution "pure". Le fait que l'erreur d'estimation soit plus forte que quand $p = 1$ reflète le fait qu'à n fixé il est plus difficile d'estimer la moyenne d'une loi normale de variance



importante.

Exercice 2 (Estimation d'un niveau de confiance)

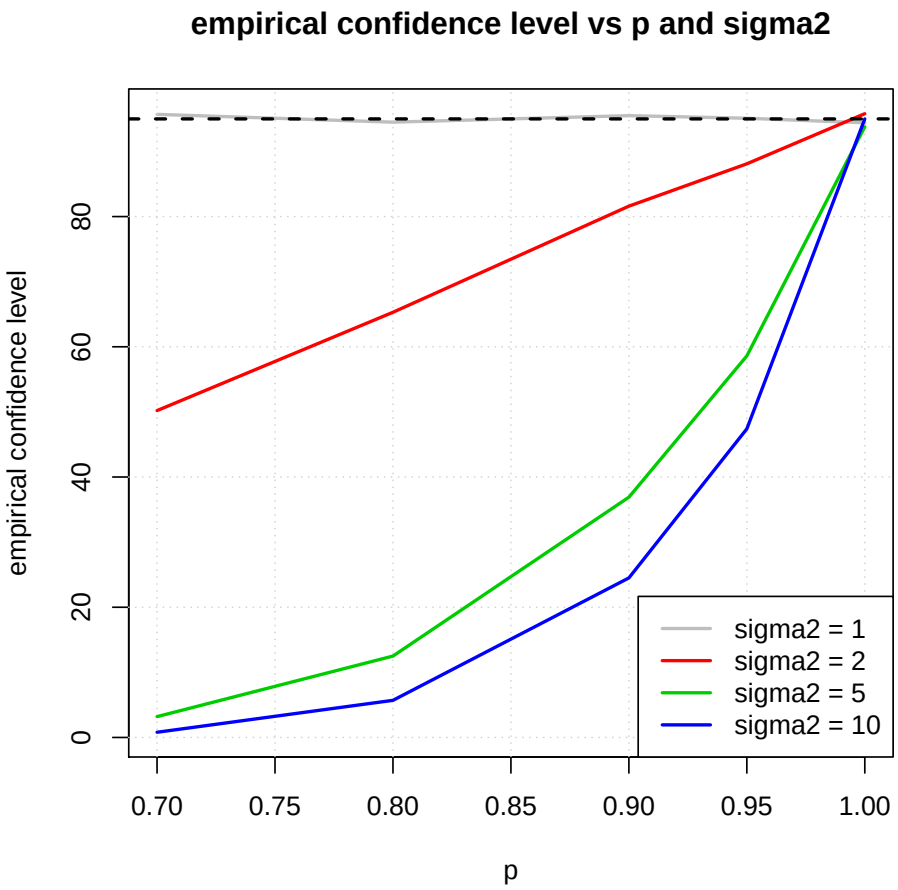
Dans cet exercice, nous allons évaluer empiriquement la couverture de l'intervalle de confiance associé á l'estimation de la variance d'une loi normale, quand on s'éloigne des hypothèses de normalité. Pour cela nous allons considérer le modèle de loi contaminée introduit dans l'exercice précédent.

Reprendre la procédure décrite dans le cours pour estimer un intervalle à 95% de la variance d'intérêt σ_1^2 *quand* $\sigma_2 \in \{2; 5; 10\}$ *et* $p \in \{1; 0.95; 0.9; 0.8; 0.7\}$. *On basera les estimations sur des n-échantillons de taille 20 et un schéma MC de* $m = 1000$ *répétitions.*

Le code ci-dessous réalise cette analyse. A noter que nous avons également considéré le cas $\sigma_2 = 1$, ce qui permet de vérifier que la procédure est correcte en absence de contamination.

```
> ##############################
> #### STARTING  EXERCICE 2 ####
> ##############################
> n = 20
```

```
> m = 1000
> p.list = c(1,0.95,0.9,0.8,0.7)
> sigma1 = 1
> sigma2.list = c(1,2,5,10)
> alpha = 0.05
> # initialize output matrix
> CONF = matrix(0, nrow = length(sigma2.list), ncol = length(p.list))
> rownames(CONF) = paste("sigma2-",sigma2.list,sep="")
> colnames(CONF) = paste("p-",p.list,sep="")
> # process each configuration
> for(i in 1:length(p.list)){
+   p = p.list[i]
+   for(j in 1:length(sigma2.list)){
+     sigma2 = sigma2.list[j]
+     tmp = replicate(m, expr = {
+       # generate sd to use
+       sigma.smple = sample(c(sigma1,sigma2), size = n, replace = TRUE, prob = c(p,1-p))
+       # draw samples
+       x = rnorm(n, mean = 0, sd = sigma.smple)
+       # compute upper limit of CI
+       Ilow  = (n-1) * var(x) /qchisq(1-alpha/2, df = n-1)
+       Ihigh  = (n-1) * var(x) /qchisq(alpha/2, df = n-1)
+       # return
+       return(c(Ilow,Ihigh))
+      })
+
+     CONF[j,i] = mean(sigma1^2 > tmp[1,] & sigma1^2 < tmp[2,])
+   }
+ }
> CONF = round(100*CONF, digits = 1)
> # plot
> plot(p.list, CONF[1,], ylim = range(CONF), xlab = "p", ylab = "empirical confidence level"
> title("empirical confidence level vs p and sigma2")
> grid()
> for(i in 2:nrow(CONF)){
+   lines(p.list, CONF[i,], col = i, lwd = 2)
+ }
> abline(h = 95, lty = 2, lwd = 2)
> legend("bottomright", paste("sigma2 =", sigma2.list), col = c("gray",seq(2,nrow(CONF))), l
```



2 Méthodes MC et tests d'hypothèses

Exercice 3 (risque de première espèce empirique)

On suppose que X suit une loi normale $\mathcal{N}(\mu, \sigma)$ de moyenne et variance inconnues. On veut tester l'hypothèse $H_0 : \mu = \mu_0 = 500$ contre l'hypothèse alternative $H_1 : \mu = \mu_1 > 500$ à un niveau de significativité $\alpha = 0.05$.

1. *Quelle statistique de test convient-il d'utiliser ?*

   Pour estimer μ nous allons utiliser la moyenne empirique. Quand on ne connait pas la variance de la loi, alors $\frac{\bar{X}_n - \mu_0}{S_n/\sqrt{n}}$, où S_n est la variance empirique, suit une loi de Student à $n-1$ degrés de liberté.

2. *Vérifier empiriquement que le risque de première espèce est bien celui attendu, en considérant des échantillons de taille $n = 20$ et $m = 10000$ itérations de la procédure de MC décrite dans le cours. Pour les simulations, nous prendrons (arbitrairement) $\sigma = 100$.*

On peut donc vérifier que le risque de première espèce empirique est bien celui attendu par la procédure suivante :

```
> ##############################
> #### STARTING  EXERCICE 3 ####
> ##############################
> mu0 = 500
> alpha = 0.05
> m = 10000
> n = 20
> sigma = 100
> p.val = numeric(m)
> for(i in 1:m){
+     x = rnorm(n, mu0, sigma)
+     tt = t.test(x, alternative="greater", mu = mu0)
+     p.val[i] = tt$p.value
+ }
> alpha.hat = mean(p.val < alpha)
> cat("*** empirical Type-I error =", alpha.hat, "(expected =", alpha, ")***\n")


*** empirical Type-I error = 0.0461 (expected = 0.05 )***
```

Notons l'utilisation de la fonction `t.test` qui réalise le test de Student selon l'hypothèse nulle (via l'option μ_0) et qui permet d'avoir accès à la p-value du test. Notons de plus que dans notre cas le test est unilatéral, avec une hypothèse alternative de moyenne plus importante.

Exercice 4 (puissance empirique)

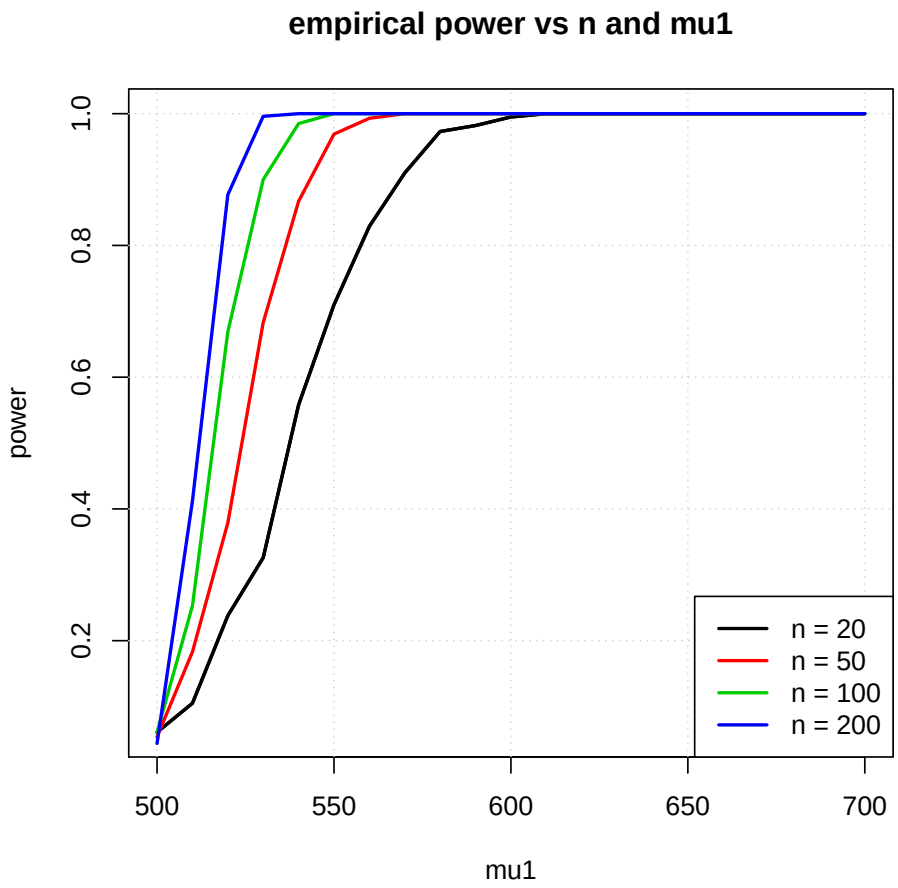
Reprendre la configuration de l'exercice 3 et estimer la puissance de détecter l'hypothèse alternative quand μ_1 varie de 500 à 700 (par pas de 10) en utilisant la procédure décrite dans le cours.

Reproduire cette analyse pour des valeurs croissantes de n (50, 100 et 200).

Le code ci-dessous réalise cette analyse.

```
> ##############################
> #### STARTING  EXERCICE 4 ####
> ##############################
> mu0 = 500
> alpha = 0.05
> m = 1000
> sigma = 100
> # define  mu1 and n values to consider
> mu1.list = seq(500, 700, by = 10)
> n.list = c(20, 50, 100, 200)
```

```
> #initialize output
> P = matrix(0, nrow = length(n.list), ncol = length(mu1.list))
> rownames(P) = n.list
> colnames(P) = mu1.list
> # process each configuration
> for(i in seq(length(n.list))){
+     n = n.list[i]
+     for(j in seq(length(mu1.list))){
+         mu1 = mu1.list[j]
+         # initialize vector of p.values
+         p.val = numeric(m)
+         # simulater according to mu1 and compute p.value
+         for(k in 1:m){
+           x = rnorm(n, mu1, sigma)
+           tt = t.test(x, alternative="greater", mu = mu0)
+           p.val[k] = tt$p.value
+         }
+         # compute power
+         P[i,j] = mean(p.val < alpha)
+   }
+ }
> # plot
> plot(mu1.list, P[1,], type = "l", lwd = 2, xlab = "mu1", ylab = "power", main  = "empirica
> grid()
> for(i in 1:nrow(P)){
+   lines(mu1.list, P[i,], type = "l", col = i, lwd = 2)
+ }
> legend("bottomright", paste("n =", n.list), col = seq(length(n.list)), lwd = 2, bg = "whit
>
>
```


empirical power vs n and mu1
power
0.2
0.4
0.6
0.8
1.0
500
550
600
650
700
mu1
n = 20
n = 50
n = 100
n = 200