Corrigé de l'examen NOISE de janvier 2012 (sujet A)

25 janvier 2012

1 Exercice 1

1.1 Question 1

On utilise comme distribution instrumentale la loi normale $N(0, 3/2)$ (de moyenne 0 et de variance $3/2$). Si l'on note sa densité $g(x)$, avec : $g(x) = e^{-x^2/3}/\sqrt{3\pi}$, il faut trouver un majorant M de $\frac{f(x)}{g(x)}$ sur $\mathbb{R}_+$. Or :

$$\frac{f(x)}{g(x)} = \frac{Cxe^{-2x^2/3}\mathbf{1}_{x\geq 0}}{e^{-x^2/3}/\sqrt{3\pi}} = C\sqrt{3\pi}xe^{-x^2/3}\mathbf{1}_{x\geq 0}.$$

On est donc ramené au problème de trouver un majorant sur $\mathbb{R}_+$ de $\varphi(x) = xe^{-x^2/3}$. En faisant une étude de variation, on trouve facilement que : $\max_{\mathbb{R}_+}\varphi = \sqrt{3/2}e^{-1/2}$, d'où : $M = 3C\sqrt{\pi/2}e^{-1/2}$. Pour générer des réalisations de la densité f, on peut donc procéder comme suit :

1. Tirer n candidats $Y_1, \ldots, Y_n$ dans la loi normale $N(0, 3/2)$, et n réalisations $U_1, \ldots, U_n$ de la loi $\mathcal{U}_{[0,1]}$
2. Ne conserver que les Y_i tels que :

$$U_i \leq \frac{f(Y_i)}{Mg(Y_i)} \Leftrightarrow U_i \leq \sqrt{2/3}Y_i e^{(1/2-Y_i^2/3)}\mathbf{1}_{Y_i\geq 0}.$$

On en déduit le code suivant :

```
> fAR1 = function(n){
+ Y = rnorm(n, sd=sqrt(1.5))
+ U = runif(n)
+ A = U < sqrt(2/3) * Y * exp(.5 - Y^2/3) * (Y >= 0)
+ Y[A]
+ }
```

Barème : 2 points. Résolution partielle : 0.5 point pour le choix de la loi instrumentale, 1 point pour le calcul de la constante (optimisation numérique acceptée) et 0.5 points pour le code final

1.2 Question 2

```
> n = 1E4
> X = fAR1(n)
```

Le code ci-dessus génère un certain nombre n_A de réalisations de la densité f, ce qui permet d'estimer la probabilité d'acceptation P de l'algorithme d'acceptation-rejet par Monte-Carlo, à l'aide du taux d'accceptation $\widehat{P} = n_A/n$, ainsi qu'un intervalle de confiance à 95%, basé sur le théorème central limite :

```
> nA = length(X)
> Pest = nA / n
> alpha = .05
> IC_P = Pest + qnorm(1-alpha/2) * sqrt(Pest*(1-Pest)/n) * c(-1,1)
```

Or, on sait que la constante M utilisée dans la construction de l'algorithme d'acceptation-rejet vérifie : $M = 1/P$, d'où :

$$3C\sqrt{\pi/2}e^{-1/2} = 1/P \Leftrightarrow C = \frac{1}{3P}e^{1/2}\sqrt{2/\pi}.$$

On peut donc estimer C par $\widehat{C} = \dfrac{1}{3\widehat{P}}e^{1/2}\sqrt{2/\pi}$. De même, si $[a; b]$ est un intervalle de confiance à 95% pour P, avec $0 < a < b$, alors $\dfrac{1}{3}e^{1/2}\sqrt{2/\pi}[1/b; 1/a]$ est un intervalle de confiance à 95% pour C. On en déduit le code suivant :

```
> Cest = exp(.5)*sqrt(2/pi)/(3*Pest)
> IC_C = (exp(.5)*sqrt(2/pi)/(3*IC_P))[c(2, 1)]
```

On obtient finalement les valeurs numériques suivantes :

```
> Cest
```

```
[1] 1.339329
```

```
> IC_C
```

```
[1] 1.302733 1.378042
```

Barème : 2 points. Résolution partielle : 0.5 point pour l'écriture de la relation entre C et P, 0.5 points pour l'estimation de C et 1 point pour la construction de l'intervalle de confiance (une autre possibilité est de faire du bootstrap)

1.3 Question 3

$$\begin{aligned} F(t) &= C\int_0^t xe^{-2x^2/3}dx \\ &= C\left[-\frac{3}{4}e^{-2x^2/3}\right]_0^t \\ &= \frac{3C}{4}\left(1 - e^{-2t^2/3}\right). \end{aligned}$$

$F(\infty) = 1$ entraîne que : $C = 4/3$, d'où : $F(t) = 1 - e^{-2t^2/3}$. L'inverse de F est donc donnée par : $F^{-1}(u) = \sqrt{-\frac{3}{2}\ln(1-u)}$. Pour générer n réalisations de f en utilisant le principe d'inversion générique, il suffit d'appliquer F^{-1} à n réalisations de la loi $\mathcal{U}_{[0,1]}$. On en déduit le code suivant (en utilisant le fait que si $U \sim \mathcal{U}_{[0,1]}$, alors $1 - U \sim \mathcal{U}_{[0,1]}$) :

```
> fAR2 = function(n) sqrt(-1.5*ln(runif(n)))
```

Barème : 1.5 points. Résolution partielle : 0.5 point pour le calcul de $F(t)$ et de $F^{-1}(u)$ chacun, et 0.5 points pour le code

1.4 Question 4

Afin d'obtenir des échantillons de tailles approximativement égales, on simule des réalisations de la densité f à l'aide de la fonction `fAR1()` pour $n/\widehat{P}$ candidats, de manière à obtenir en moyenne n réalisations, et exactement n réalisations à l'aide de la fonction `fAR2()` :

```
> par(mfrow=c(1,2))
> X1 = fAR1(n/Pest)
> hist(X1, sqrt(length(X1)), prob=TRUE)
> curve(4/3 * x * exp(-2*x^2/3), col='red', add=TRUE)
> X2 = fAR2(n)
> hist(X2, sqrt(n), prob=TRUE)
> curve(4/3 * x * exp(-2*x^2/3), col='red', add=TRUE)
```

On obtient alors le graphe illustré dans la figure 1.

Barème : 1 point

1.5 Question 5

Les fonctions `fAR1()` et `fAR2()` génèrent toutes deux des réalisations de la même densité f. On peut donc retenir comme points de comparaison pertinents la complexité de leurs codes, ou leurs temps de calcul respectifs. Dans les deux cas, `fAR2()` apparaît comme la meilleure puisqu'il s'agit à la fois de la plus simple (elle ne nécessite qu'une seule ligne de code) et de la moins coûteuse (elle ne nécessite pas de rejeter une fraction des variables simulées) des deux fonctions.

Barème : 0.5 points.

1.6 Question 6

```
> alpha = .05
> mXest = mean(X2)
> IC_mX = mXest + qnorm(1 - alpha/2)*sd(X2)/sqrt(n)*c(-1,1)
> mX2est = mean(X2^2)
> IC_mX2 = mX2est + qnorm(1 - alpha/2)*sd(X2^2)/sqrt(n)*c(-1,1)
```

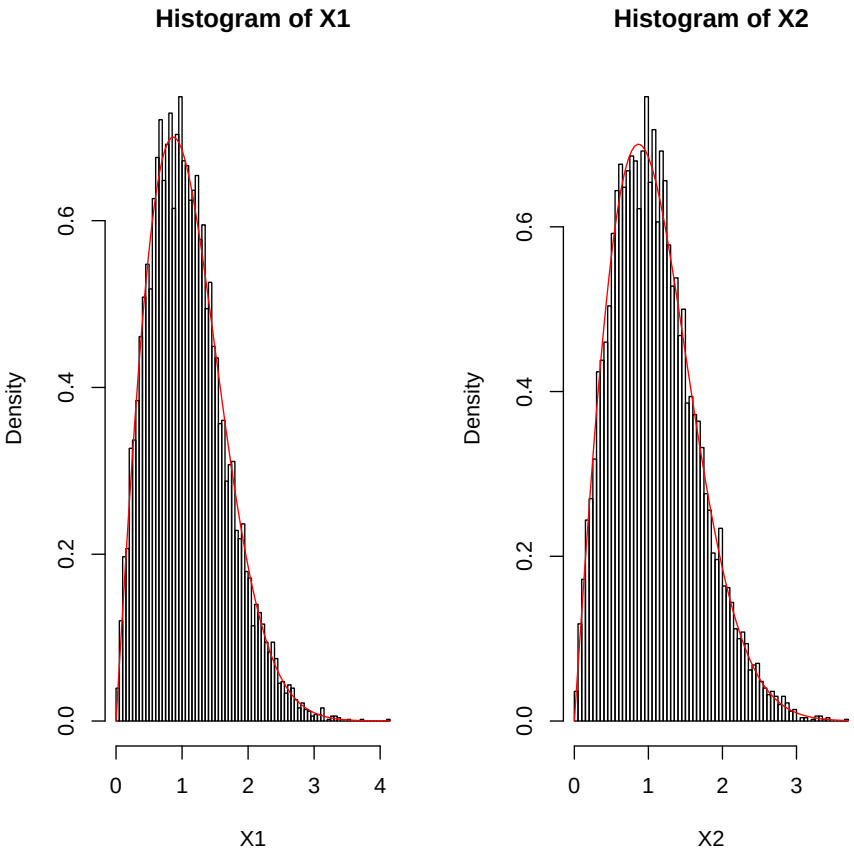
Histogram of X1
Density
0.0
0.2
0.4
0.6
0
1
2
3
4
X1
Histogram of X2
Density
0.0
0.2
0.4
0.6
0
1
2
3
X2

FIGURE 1 –

```
> Pest = mean(X2>2)
> IC_P = Pest + qnorm(1 - alpha/2)* sqrt(Pest*(1-Pest)/n)*c(-1,1)

> mXest

[1] 1.087605

> IC_mX

[1] 1.076500 1.098710

> mX2est

[1] 1.503904

> IC_mX2

[1] 1.474633 1.533174

> Pest

[1] 0.0699

> IC_P

[1] 0.06490251 0.07489749
```

Barème : 3 points (0.5 par estimation et par intervalle de confiance).

2 Exercice 2

2.1 Question 1

La commande :

```
> plot(AirPassengers)
```

affiche le graphe de la figure 2, à gauche. On remarque des pics de voyageurs chaque année durant l'été, ce qui s'explique aisément par le fait qu'il s'agit de la période où le plus de monde part en vacances.

Barème : 1 point (0.5 point pour le graphe, 0.5 point pour l'explication rationnelle)

2.2 Question 2

RAS

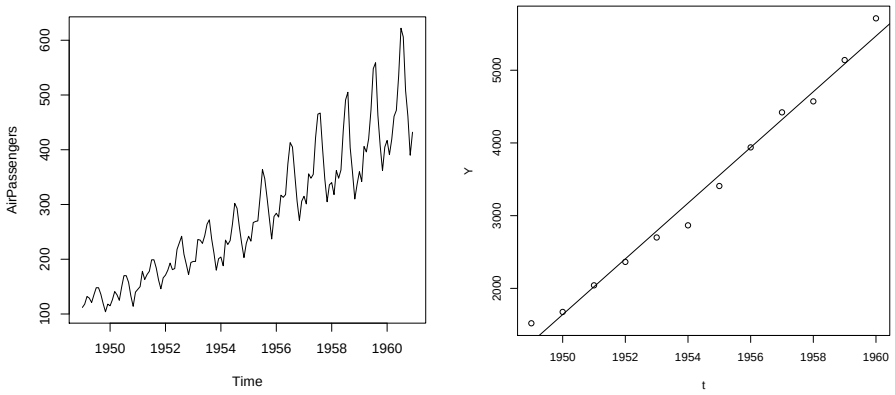

FIGURE 2 –

2.3 Question 3

(a) La ligne de code :

```
> plot(t, Y)
```

génère le nuage de points présenté dans la figure 2, à droite.
Barème : 1 point

(b) Le code ci-dessous ajoute la droite dans la figure 2, à droite.

```
> Ybar = mean(Y)
> tBar = mean(t)
> SYt = mean(Y*t) - Ybar * tBar
> S2t = mean(t^2) - tBar^2
> bHat = SYt / S2t
> aHat = Ybar - best * tBar
> abline(aest, best)
```

Barème : 2 points (1 point pour le calcul, 1 point pour le graphe)

(c)-(d) On utilise la méthode de bootstrap des résidus pour échantillonner la loi des estimateurs $(\widehat{a}, \widehat{b})$, c'est-à-dire que l'on calcule les résidus $R_i = Y_i - \widehat{a} - \widehat{b}t_i$, puis on effectue les opérations suivantes :

- Un échantillon bootstrap de résidus $(R_1^*, \ldots, R_n^*)$ est généré par un tirage uniforme avec remise dans $(R_1, \ldots, R_n)$
- Un échantillon bootstrap d'observations $(Y_1^*, \ldots, Y_n^*)$ est reconstitué selon l'identité : $Y_i^* = R_i^* + \widehat{a} + \widehat{b}t_i$
- Des valeurs bootstraps des paramètres $(\widehat{a}^*, \widehat{b}^*)$ sont estimés à partir de $(Y_1^*, \ldots, Y_n^*)$ par moindre carrés, de la même façon que $(\widehat{a}, \widehat{b})$ est estimé à partir de $(Y_1, \ldots, Y_n)$

En répétant ces opérations B fois, on obtient un échantillon $(a_j^*, b_j^*)_{1 \leq j \leq B}$ de valeurs bootstrap de paramètres estimés, à l'aide du code suivant :

```
> R = Y - aest - best * t
> B = 1000
> bootaest = rep(0, B)
> bootbest = rep(0, B)
> for (b in 1:B){
+ RB = sample(R, length(R), replace=TRUE)
+ YB = RB + aest + best * t
+ YbarB = mean(YB)
+ SYBt = mean(YB*t) - YbarB * tBar
+ bootbest[b] = SYBt / S2t
+ bootaest[b] = YbarB - bootbest[b] * tBar
+ }
```

On peut utiliser ces échantillons pour calculer le biais de l'estimateur $\widehat{a}$, ainsi qu'un intervalle de confiance pour $\widehat{b}$ par la procédure des percentiles bootstrap (cf. le poly), à l'aide du code suivant :

```
> bias = mean(bootaest) - aest
> alpha = .05
> ICb = best + quantile(best - bootbest, c(alpha / 2, 1 - alpha / 2))
```

On obtient les valeurs suivantes :

```
> bias
[1] 480.5786
> ICb
    2.5%    97.5%
358.4303 409.8688
```

Barème : 4 points (2 points pour le code générant léchantillon de valeurs bootstrap $(a_j^*, b_j^*)_{1\leq j\leq B}$, 1 point pour le calcul du biais de $\widehat{a}$ et 1 point pour le calcul de l'intervalle de confiance de b)

(e) On estime le nombre de passagers en 1962 par : $\widehat{Y}(1962) = \widehat{a} + \widehat{b} \times 1962$, soit :

```
> Yest = aest + best * 1962
> Yest
[1] 6236.739
```

Barème : 2 points (1 point pour la formule, 1 point pour le résultat)

3 Exercice 3

3.1 Question 1

On va utiliser un algorithme d'acceptation-rejet pour simuler des réalisations de la loi de Weibull de densité $g(x; k, \lambda, \theta)$, pour $\theta = 1, k = 3$ et $\lambda = 7$, en utilisant comme distribution instrumentale la loi $N(m, s^2)$, avec $m = \theta + \lambda \left(\frac{k-1}{k}\right)^{1/k}$

et s choisi de manière à maximiser le taux d'acceptation. Notons $h(x;s)$ la densité de cette loi. À s fixé, cela signifie que l'on doit trouver un majorant $M(s)$ de $\frac{g(x;k,\lambda,\theta)}{h(x;s)}$ sur $]\theta,+\infty[$. Il n'y a pas de manière simple de calculer explicitement ce majorant, mais on peut trouver une évaluation numérique du maximum de cette fonction, à l'aide de la fonction `optimize`.

Ayant construit l'algorithme d'acceptation-rejet sur la base de ce majorant, on sait que la probabilité d'acceptation est égale à : $P(s) = 1/M(s)$. Pour maximiser cette probabilité, il faut donc dans un deuxième temps minimiser $M(s)$ en s, ce qui peut également être fait numériquement. Nous pouvons à présent écrire le code suivant :

```
> # Optimisation de la constante d'acceptation-rejet
>
> k = 3; lambda = 7; theta = 1
>
> g = function(x){
+ k/lambda * ((x - theta)/lambda)^(k-1) * exp(-((x-theta)/lambda)^k) * (x > theta)
+ }
>
> m = theta + lambda * ((k-1)/k)^(1/k)
>
> M = function(s){
+ func = function(x) g(x)/dnorm(x, mean=m, sd=s)
+ as.numeric(optimize(func, lower=0.1, upper=100, maximum=TRUE)[2])
+ }
>
> s = as.numeric(optimize(M, lower=0.1, upper=100)[1])
>
> Mopt = M(s)
>
> # Algorithme d'acceptation-rejet
>
> fAR = function(n){
+ Y = rnorm(n, mean=m, sd=s)
+ U = runif(n)
+ A = U < g(Y) / (dnorm(Y, mean=m, sd=s) * Mopt)
+ X = Y[A]
+ tx_accept = length(X) / n
+ list(X, tx_accept)
+ }
```

Notons que la fonction `myrweibull` ci-dessus génère à partir de n candidats un certain nombre n_A (aléatoire) de réalisations de la loi de Weibull voulue, d'espérance $\mathbb{E}[n_A] = n/M(s)$.

Barème : 3 points (1 point pour le calcul de $M(s)$ - optimisation numérique acceptée - 1 point pour l'optimisation en s et 1 point pour le code)

4 Question 2

La fonction de répartition de la distribution de Weibull généralisée de paramètres (k, λ, θ) s'écrit :

$$G(x; k, \lambda, \theta) = \left(1 - e^{-\left(\frac{x-\theta}{\lambda}\right)^k}\right) \mathbf{1}_{\{x>\theta\}}$$

et son inverse :

$$G^{-1}(u; k, \lambda, \theta) = \theta + \lambda\{-\ln(1-u)\}^{1/k}.$$

Pour générer des réalisations de la distribution de Weibull généralisée en utilisant le principe d'inversion généralisée, il suffit donc d'appliquer G^{-1} à des réalisations de la loi $\mathcal{U}_{[0,1]}$. On en déduit le code suivant :

```
> fIG = function(n) theta + lambda*(-log(runif(n)))^(1/k)
```

Barème : 2 points (1 point pour le calcul de G^{-1}, 1 point pour le code)

4.1 Question 3

Afin d'obtenir des échantillons de taille approximativement égales, on simule des réalisations de la densité g à l'aide de la fonction `fAR()` pour $n * M(s)$ candidats, de manière à obtenir en moyenne n réalisations, et exactement n réalisations à l'aide de la fonction `fIG()` :

```
> n = 1E4
> xAR = fAR(n * Mopt)[[1]]
> xIG = fIG(n)
> par(mfrow=c(1,2))
> hist(xAR, sqrt(length(xAR)), prob=TRUE)
> curve(g, col='red', add=TRUE)
> hist(xIG, sqrt(n), prob=TRUE)
> curve(g, col='red', add=TRUE)
```

On obtient ainsi les graphes présentés en figure 3

Barème : 2 points (1 pour chaque figure)

4.2 Question 4

Le code ci-dessous génère les graphes présentés en figure 4

```
> G = function(x) (1 - exp(-((x - theta)/lambda)^k)) * (x > theta)
> par(mfrow=c(1,2))
> plot(sort(xAR), (1:length(xAR))/length(xAR), type='s')
> curve(G, col='red', add=TRUE)
> plot(sort(xIG), (1:n)/n, type='s')
> curve(G, col='red', add=TRUE)
```

Barème : 1 point

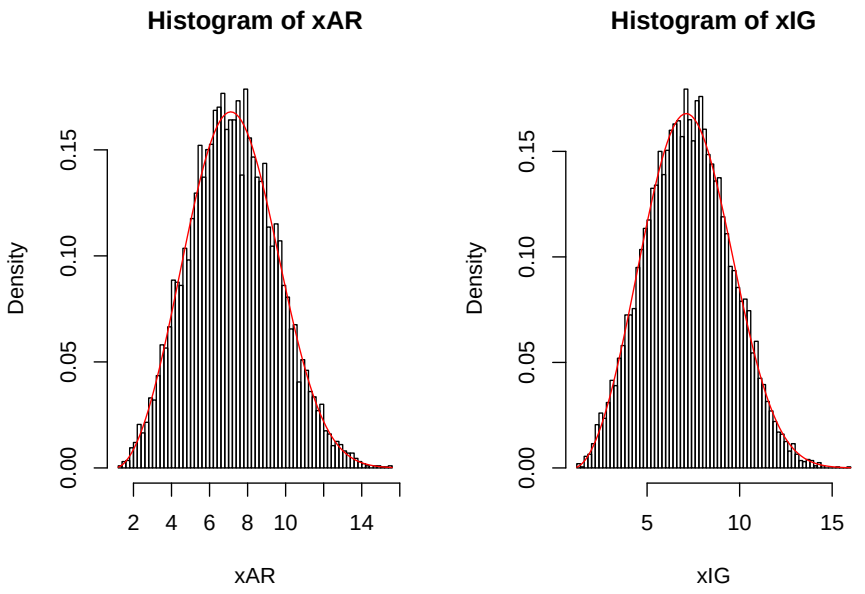

FIGURE 3 –

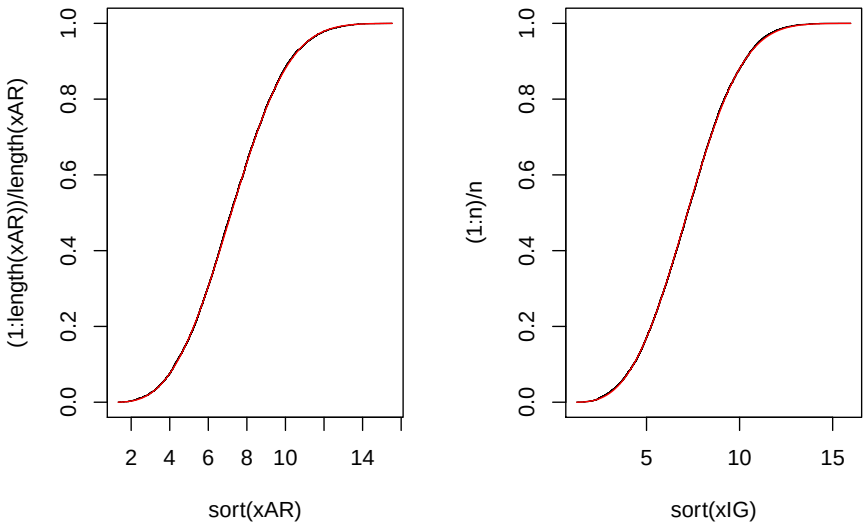

FIGURE 4 –

4.3 Question 5

Les deux algorithmes de simulation donnent des résultats équivalents, en revanche celui codé dans `fIG()` est plus simple et plus rapide, puisqu'il ne comporte pas de rejet.

```
> Mest = mean(xIG)
> Vest = mean(xIG^2) - Mest^2
>
> Mest

[1] 7.258179

> Vest

[1] 5.244181
```

Barème : 2 points (1 pour le choix de l'algorithme et 0.5 pour chaque estimation de Monte-Carlo)

5 Exercice 4

5.1 Question 1

(a) X est centrée, d'où : $\mathbb{E}[X] = 0$. Donc :

$$\sigma^2 = \mathbb{V}[X] = \mathbb{E}[X^2] = \frac{1}{2}\int_{-1}^{1} t^2 dt = \frac{1}{2}\left[\frac{t^3}{3}\right]_{-1}^{1} = \frac{1}{3}.$$

Enfin :

$$\beta_1 = \mathbb{E}\left[\left(\frac{X - \mathbb{E}[X]}{\sigma}\right)^4\right] = \frac{\mathbb{E}[X^4]}{\sigma^4} = \frac{9}{2}\int_{-1}^{1} t^4 dt = \frac{9}{2}\left[\frac{t^5}{5}\right]_{-1}^{1} = \frac{9}{5}.$$

Barème : 1 point

(b) Pour simplifier le code, on peut créer une fonction `kurtosis()` qui calcule l'estimateur des moments de β_1 pour un échantillon donné. On obtient par exemple pour $n = 100$:

```
> kurtosis = function(X) mean(((X - mean(X))/sd(X))^4)
> n = 100
> X = runif(n, -1, 1)
> B1 = kurtosis(X)
> B1

[1] 1.671663
```

On peut alors estimer le biais de l'estimateur B_1 par bootstrap, à l'aide du code suivant :

```
> MCB1 = function(n){
+ BB1 = rep(0, B)
+ for (b in 1:B) BB1[b] = kurtosis(runif(n, -1, 1))
+ BB1
+ }
>
> B = 10000
> n = 10
> bias = mean(MCB1(n)) - 9/5
> bias
[1] -0.1957881
> n = 100
> bias = mean(MCB1(n)) - 9/5
> bias
[1] -0.009922961
> n = 1000
> bias = mean(MCB1(n)) - 9/5
> bias
[1] -0.000993309
```

Barème : 1 point

5.2 Question 2

$$\beta_1 = \mathbb{E}[X^4] = 3$$

Barème : 1 point

5.3 Question 3

(a) La ligne de code ci-dessous génère le graphe présenté dans la figure 5 :

```
> hist(X, sqrt(length(X)), prob=TRUE)
```

Barème : 1 point

(b) On estime β_1 par l'estimateur des moments B_1 :

```
> B1 = kurtosis(X)
> B1
[1] 1.488395
```

On trouve une valeur inférieure à la valeur théorique sous hypothèse gaussienne, ce qui suggère que la distribution des données est moins "aplatie" qu'une gaussienne. Ceci s'explique clairement par son caractère bimodal.
Barème : 2 points (1 point pour le calcul et 1 point pour l'interprétation)

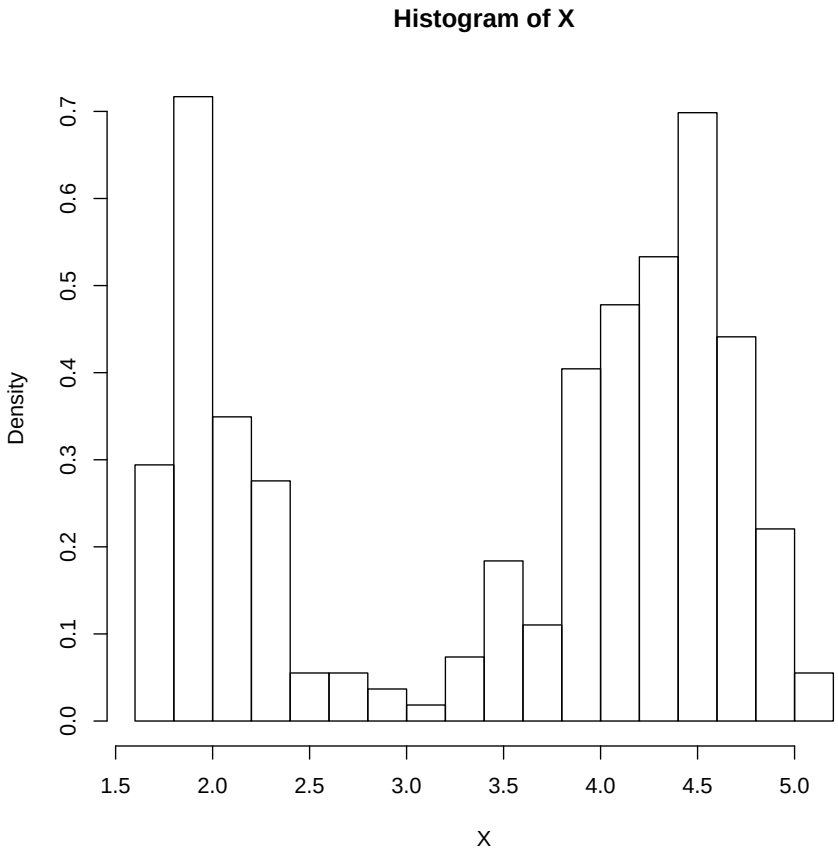
Histogram of X
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
X

FIGURE 5 –

(c) Le code ci-dessous définit une fonction `bootstrap_kurtosis()` qui génère un échantillon de valeurs bootstrap de l'estimateur B_1 à partir d'un échantillon de valeur donné :

```
> bootstrap_kurtosis = function(H, B){
+ bootB1 = rep(0, B)
+ for (b in 1:B) bootB1[b] = kurtosis(sample(H, length(H), replace=TRUE))
+ bootB1
+ }
```

On en déduit l'intervalle de confiance pour B_1 basé sur les données complètes :

```
> bootB1 = bootstrap_kurtosis(X, B)
> alpha = .05
> IC = B1 + quantile(B1 - bootB1, c(alpha / 2, 1 - alpha / 2))
> IC
    2.5%    97.5%
1.345158 1.584136
```

Cet intervalle de confiance ne contient pas 3. On peut donc rejeter au niveau 5% l'hypothèse que les données `faithful$eruptions` sont gaussiennes (ce qu'on pouvait prévoir vu leur caractère bimodal).
Barème : 2 points (1 point pour le calcul et 1 point pour l'interprétation)

(d)
```
> Y = X[X>3]
> B1 = kurtosis(Y)
> bootB1 = bootstrap_kurtosis(Y, B)
> alpha = .05
> IC = B1 + quantile(B1 - bootB1, c(alpha / 2, 1 - alpha / 2))
> IC
    2.5%    97.5%
2.149178 3.227942
```

L'intervalle de confiance pour la kurtosis de la distribution des durées d'éruption supérieures à 3 *mn* contient 3 ; on ne peut donc pas rejeter au niveau 5% l'hypothèse de normalité de ces éruptions. Autrement dit, la loi normale paraît pertinente pour modéliser ces données.
Barème : 2 points (1 point pour le calcul et 1 point pour l'interprétation)

6 Exercice 5

6.1 Question a

```
> med = median(x)
> med

[1] 4
```

Barème : 1 point

6.2 Question b

On approche la distribution de $\hat{\theta}(x) - \theta$ par la distribution empirique de $\hat{\theta}(x^*) - \hat{\theta}(x)$:

```
> B = 500
> bootmed = rep(0, B)
> for (b in 1:B) bootmed[b] = median(sample(x, length(x), replace=TRUE))
> alpha = .05
> ICbias = quantile(bootmed, c(alpha/2, 1 - alpha/2)) - med
> ICbias

      2.5%      97.5%
-0.1670000  0.1044625
```

Barème : 3 points (2 points pour le calcul de l'échantillon bootstrap et 1 point pour le calcul de l'intervalle de confiance)

6.3 Question c

On approche la distribution de $\hat{\theta}(x) - \theta$ par la loi normale $N(m, \sigma^2)$ où m est la moyenne empirique de l'échantillon de valeurs bootstrap de $\hat{\theta}(x^*) - \hat{\theta}(x)$ et σ son écart-type.

```
ICbias_TLC = mean(bootmed) + qt(1-alpha/2,B-1)*c(-1,1)*sd(bootmed)/sqrt(B) - med
> ICbias_TLC

[1] -0.021528646 -0.007731354
```

On remarque que l'hypothèse d'une absence de biais est rejetée si l'on suppose que le biais empirique est distribué normalement, alors qu'on ne pouvait conclure dans la question précédente, ou le calcul n'était pas basé sur cette hypothèse.

Barème : 3 points (2 points pour le calcul de l'intervalle de confiance, 1 point pour le commentaire)

6.4 Question d

```
> ks.test(jitter(bootmed), pnorm(x, mean(bootmed), sd(bootmed)) )

        Two-sample Kolmogorov-Smirnov test

data:  jitter(bootmed) and pnorm(x, mean(bootmed), sd(bootmed))
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

L'hypothèse de nomalité du biais est rejettée par le test de Kolmogorov-Smirnov

Barème : 1.5 points (1 point pour le calcul et 0.5 point pour le commentaire)

6.5 Question d

```
> ks.test(jitter(bootmed), pnorm(x, 0, sqrt(mean(bootmed^2))) )

        Two-sample Kolmogorov-Smirnov test

data:  jitter(bootmed) and pnorm(x, med, sd = sqrt(mean(bootmed^2)))
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

L'hypothèse de nomalité du biais est toujours rejettée par le test de Kolmogorov-Smirnov lorsqu'on fait de plus l'hypothèse que la distribution du biais est centrée en 0.

Barème : 1.5 points (1 point pour le calcul et 0.5 point pour le commentaire)

7 Exercice 6

7.1 Question a

```
> n = 50
> N = 1000
> X = matrix(rnorm(n * N), nrow=N)
> P = rep(0, N)
> for (i in 1:N) P[i] = as.numeric(ks.test(X[i,], "pnorm")[2])
```

Barème : 2 points

7.2 Question b

Le code ci-dessous génère le graphe présenté en figure 6, à gauche :

```
> hist(P, sqrt(N), prob=TRUE)
> ks.test(P, "punif")

        One-sample Kolmogorov-Smirnov test

data:  P
D = 0.0244, p-value = 0.5902
alternative hypothesis: two-sided
```

On ne rejette effectivement pas l'hypothèse d'adéquation de la loi uniforme aux p-values.

Barème : 3 points

7.3 Question c

```
> P2 = rep(0, N)
> for (i in 1:N) P2[i] = as.numeric(ks.test(X[i,], "pnorm", mean=mean(x), sd=sd(x))[2])
```

Barème : 2 points

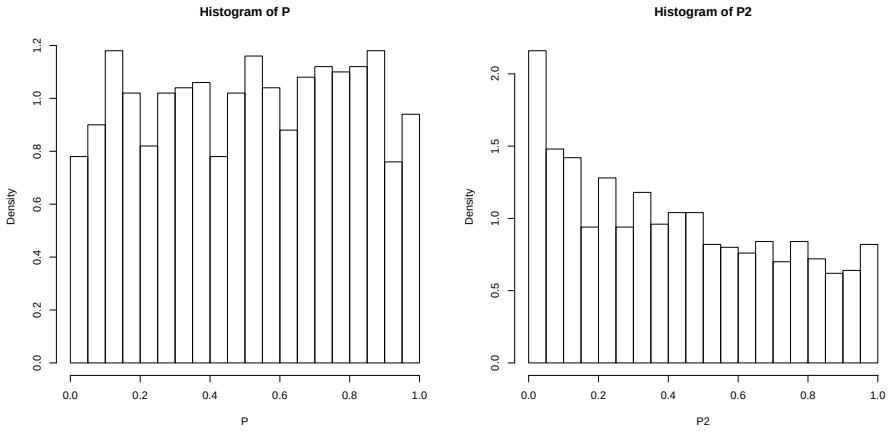

FIGURE 6 –

7.4 Question d

Le code ci-dessous génère le graphe présenté en figure 6, à droite :

```
> hist(P2, sqrt(N), prob=TRUE)
> ks.test(P2, "punif")

        One-sample Kolmogorov-Smirnov test

data:  P2
D = 0.1257, p-value = 3.852e-14
alternative hypothesis: two-sided
```

On rejette cette fois-ci l'hypothèse de distribution uniforme sur $[0, 1]$ des p-values, ce qui prouve que celles-ci ne sont pas valides, puisque les données sont bien sous hypothèse nulle et que dans ce cas les p-values devraient nécessairement être uniformément distribuées sur $[0, 1]$.

Barème : 3 points