

Echantillonnage préférentiel

Travaux dirigés

Pierre Gloaguen

Contents

1	Évènement rare	1
2	Cas problématique	6
2.1	Premier cas jouet	6
2.2	Encore pire!	6

1 Évènement rare

On veut estimer la probabilité p^* qu'une loi normale centrée réduite dépasse la valeur 3.

1. Pour un effort de Monte Carlo de taille M , proposer un estimateur de Monte Carlo pour p^* .

On peut poser

$$\hat{p} = \frac{1}{M} \sum_{k=1}^M \mathbf{1}_{X_k > 3}$$

où les X_1, \dots, X_M sont des variables aléatoires i.i.d. de loi normale $\mathcal{N}(0, 1)$.

2. Implémenter cet estimateur sur R pour un effort de Monte Carlo de taille $M = 10000$

```
library(tidyverse)
```

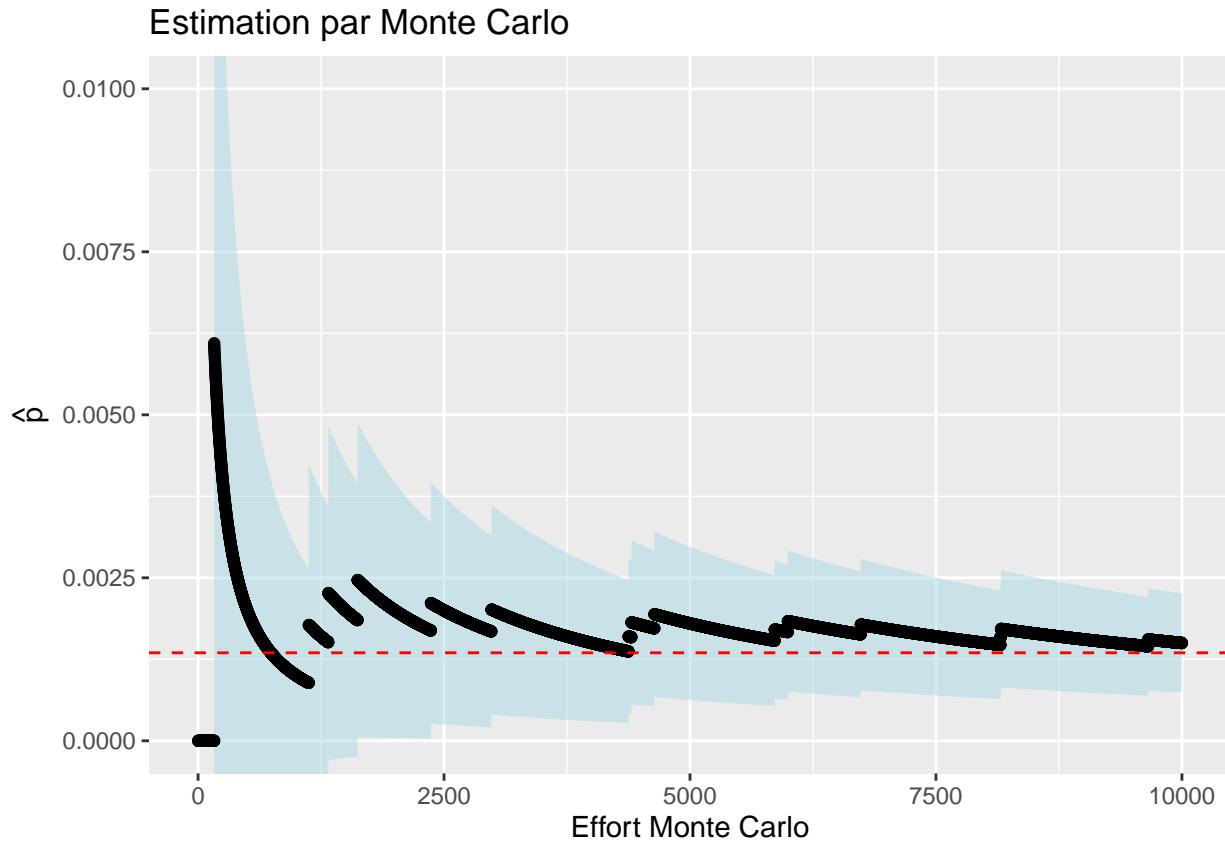
La fonction suit le même squelette que d'habitude

```
get_monte_carlo_estimate <- function(M, threshold = 3){  
  x_sample <- rnorm(M, 0, 1) # Simulation des X  
  above_threshold <- x_sample > threshold  
  z_975 <- qnorm(0.975) # Quantile de la loi normale  
  p_hat <- cumsum(above_threshold) / (1:M) # Estimation de p  
  # Fast way to compute E[f(x)^2] - E[f(x)]^2 on the fly  
  sigma2_p_hat <- cumsum(above_threshold^2) / (1:M) - p_hat^2  
  # Stacking in a data_frame  
  tibble(index = 1:M,  
         phi_x = above_threshold,  
         p_hat = p_hat) %>%  
  mutate(sup_IC_emp = p_hat + z_975 * sqrt(sigma2_p_hat / index), # IC bounds  
        inf_IC_emp = p_hat - z_975 * sqrt(sigma2_p_hat / index))  
}  
  
set.seed(123) # For reproducible results  
my_monte_carlo_estimate <- get_monte_carlo_estimate(1e4)  
  
my_monte_carlo_estimate %>%  
  ggplot(aes(x = index, y = p_hat)) +  
  geom_ribbon(mapping = aes(ymin = inf_IC_emp, ymax = sup_IC_emp),
```

```

        fill = "lightblue", alpha = 0.5) +
coord_cartesian(y = c(0, 0.01)) + # Zoom on the interesting zone
geom_point() +
labs(x = "Effort Monte Carlo", y = expression(hat(p)),
title = "Estimation par Monte Carlo") +
geom_hline(yintercept = 1 - pnorm(3), # True value
linetype = 2, col = "red")

```



On peut constater que l'estimation est très instable, avec un IC asymptotique qui comprend très longtemps des valeurs négatives! La variance empirique de l'estimateur peut être obtenue:

```
var(my_monte_carlo_estimate$phi_x)
```

```
[1] 0.0014979
```

3. On se propose d'utiliser un échantillonnage préférentiel pour estimer cette probabilité. On utilisera comme loi d'échantillonnage une loi exponentielle translatée de 3, de paramètre λ notée $Y \sim t\mathcal{E}(3, \lambda)$, i.e., la variable aléatoire Y telle que $Y - 3 \sim \mathcal{E}(\lambda)$. Calculer les poids d'importance associés, et proposer un estimateur de p^*

On se donne un échantillon Y_1, \dots, Y_M i.i.d. de loi $t\mathcal{E}(3, \lambda)$. On note f la densité d'une loi normale $\mathcal{N}(0, 1)$ et g la densité d'une loi $\mathcal{E}(\lambda)$. On a:

$$g(y) = \lambda e^{-\lambda(y-3)} \mathbf{1}_{y>3}$$

Pour $1 \leq k \leq M$, les poids d'importance sont donnés par

$$W(Y_k) = \frac{f(Y_k)}{g(Y_k)} = \frac{1}{\lambda\sqrt{2\pi}} e^{-\frac{Y_k^2}{2} + \lambda(Y_k - 3)}$$

Un estimateur de p^* est donc donné par:

$$\hat{p}_M^{IS} = \frac{1}{M} \sum_{k=1}^M \frac{f(Y_k)}{g(Y_k)} \mathbf{1}_{Y_k > 3}$$

où les Y_1, \dots, Y_M sont des variables aléatoires i.i.d. selon une distribution exponentielle translatée de 3.

4. Ecrire la variance de l'estimateur comme fonction de λ . Comment doit on choisir λ ?

$$\mathbb{V}[\hat{p}_M^{IS}] = \frac{1}{M} (\mathbb{E}[W(Y)^2 \mathbf{1}_{Y > 3}] - (p^*)^2)$$

Donc, la variance est minimale quand $\mathbb{E}[W(Y)^2]$ est minimal.

$$\begin{aligned} \mathbb{E}[W(Y)^2] &= \int_3^\infty \left(\frac{1}{\lambda\sqrt{2\pi}} e^{-\frac{x^2}{2} + \lambda(x-3)} \right)^2 \lambda e^{-\lambda(x-3)} dx \\ &= \frac{1}{\lambda^2 2\pi} e^{-3\lambda} \int_3^\infty e^{-x^2 + \lambda x} \\ &= \frac{1}{\lambda\sqrt{2\pi}} e^{-3\lambda} \int_3^\infty \frac{1}{\sqrt{2\pi}} e^{-(x-\frac{\lambda}{2})^2 + \frac{\lambda^2}{4}} \\ &= \frac{1}{\sqrt{2}\lambda\sqrt{2\pi}} e^{\frac{\lambda^2}{4} - 3\lambda} \int_3^\infty \frac{\sqrt{2}}{\sqrt{2\pi}} e^{-\frac{(x-\frac{\lambda}{2})^2}{2 \times \frac{1}{2}}} \\ &= \frac{1}{2\lambda\sqrt{\pi}} e^{\frac{\lambda^2}{4} - 3\lambda} (1 - \phi_{\lambda/2, 1/2}(3)) \end{aligned} \quad \text{On fait apparaître la loi normale}$$

où $\phi_{\lambda/2, 1/2}$ est la fonction de répartition d'une loi $\mathcal{N}(\lambda/2, 1/2)$.

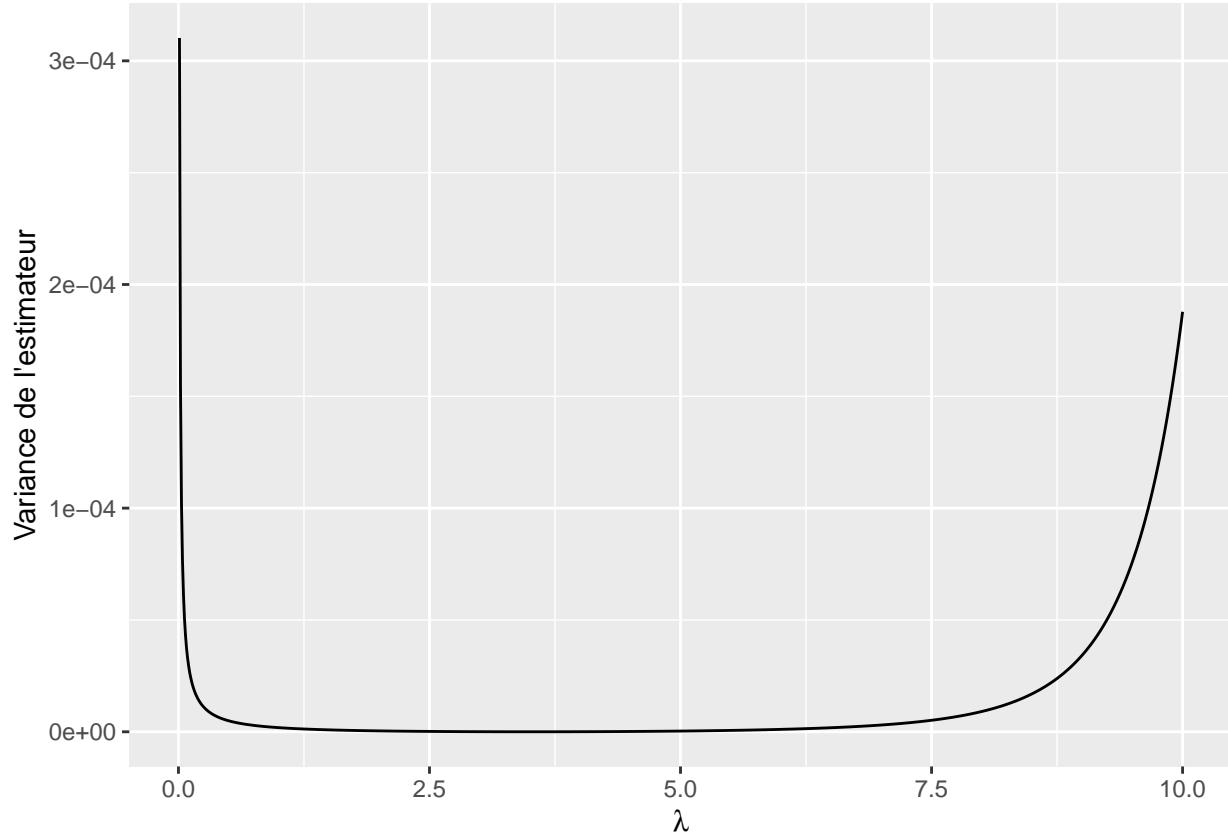
Ainsi, il faut choisir λ qui minimise cette expression.

En 'R', on peut ainsi coder une fonction pour la variance en fonction de λ :

```
get_IS_variance <- function(lambda, threshold = 3){
  # Note that we add the true -p^2 term, which is not important
  # as it is a constant
  0.5 * exp(0.25 * lambda^2 - threshold * lambda) / (lambda * sqrt(pi)) *
    (1 - pnorm(threshold, 0.5 * lambda, sqrt(0.5))) -
    (1 - pnorm(threshold))^2 # p^2
}
```

On peut représenter la courbe graphiquement:

```
tibble(lambda = seq(0.01, 10, by = 0.01)) %>% # initial lambda
  mutate(variance = get_IS_variance(lambda)) %>% # Related variance
  ggplot(mapping = aes(x = lambda, y = variance)) + # then plot it
  geom_line() +
  labs(x = expression(lambda), y = "Variance de l'estimateur")
```



On peut montrer par optimisation numérique que cette valeur optimale est légèrement supérieure à $\lambda = 3.5$. La variance est de l'ordre de $7 \cdot 10^{-9}$.

```
# Numerical optimization in R
optimize(f = get_IS_variance, interval = c(0.01, 10))
```

```
$minimum
[1] 3.512659
```

```
$objective
[1] 7.056104e-09
```

On a énormément gagné par rapport à l'estimateur Monte Carlo où la variance était de l'ordre de $10^{-3}!$

5. Donner un estimateur empirique de cette variance, et donner l'expression de l'intervalle de confiance à 95% pour l'estimateur de p^* .

Avec les mêmes notations que pour \hat{p}^{IS} , un estimateur empirique de la variance est donné par:

$$\hat{\sigma}_{M,IS}^2 = \frac{1}{M} \sum_{k=1}^M \left(\frac{f(Y_k)}{g(Y_k)} \mathbf{1}_{Y_k > 3} - \hat{p}_M^{IS} \right)^2$$

Un intervalle de confiance asymptotique de niveau 95

$$\left[\hat{p}_M^{IS} - 1.96 \sqrt{\frac{\hat{\sigma}_{M,IS}^2}{M}}, \hat{p}_M^{IS} + 1.96 \sqrt{\frac{\hat{\sigma}_{M,IS}^2}{M}} \right]$$

6. Implémenter cet estimateur sur R avec $\lambda = 3.5$ et le comparer à celui de Monte Carlo.

Le squelette de la fonction est le même que celui d'un estimateur de Monte Carlo usuel. Il faut faire attention à bien simuler les échantillons selon la nouvelle loi et à calculer les poids en conséquence.

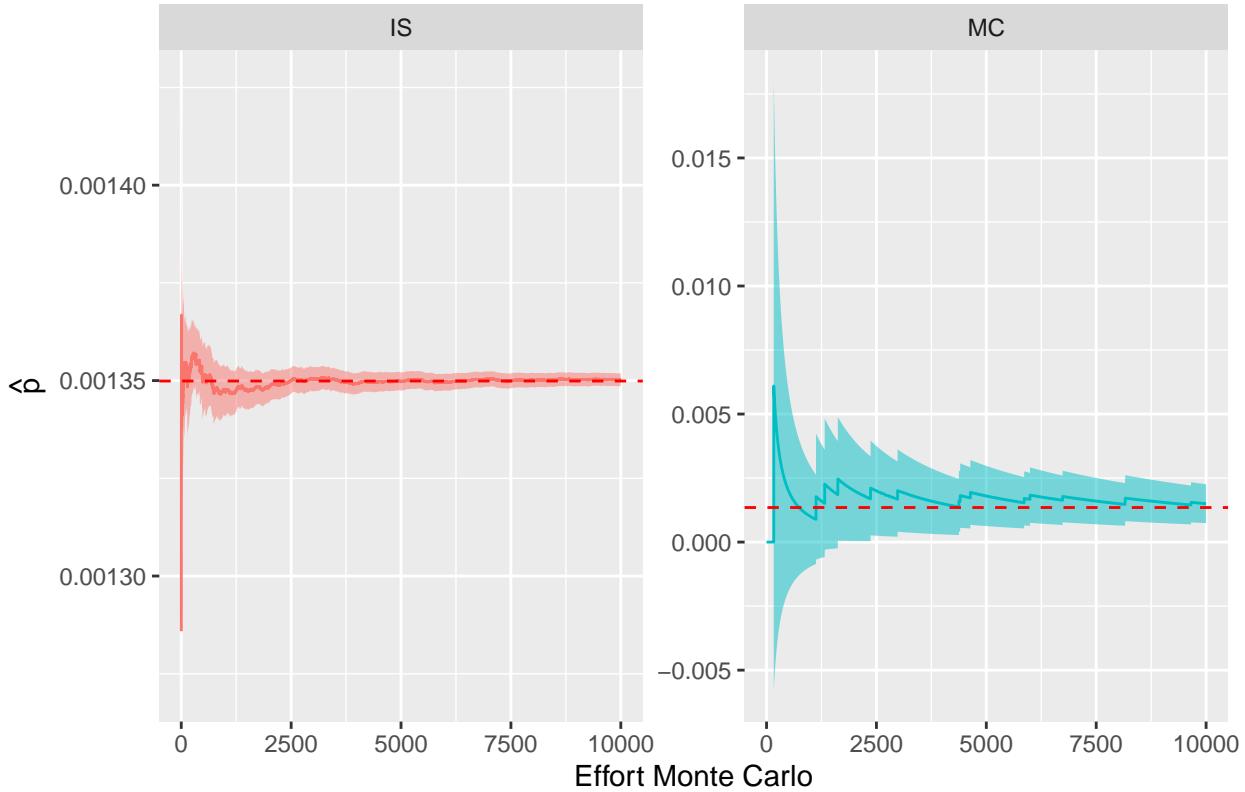
```
get_IS_estimate <- function(M, lambda, threshold = 3){
  y_sample <- threshold + rexp(M, lambda) # Simulation Y
  above_threshold <- y_sample > threshold
  z_975 <- qnorm(0.975) # Quantile de la loi normale
  weights <- dnorm(y_sample, 0, 1) / dexp(y_sample - threshold, lambda)
  p_hat <- cumsum(above_threshold * weights) / (1:M)
  sigma2_p_hat = cumsum((above_threshold * weights)^2) / (1:M) - p_hat^2
  tibble(index = 1:M,
         phi_x = above_threshold,
         p_hat = p_hat) %>%
  mutate(sup_IC_emp = p_hat + z_975 * sqrt(sigma2_p_hat / index), # IC bounds
        inf_IC_emp = p_hat - z_975 * sqrt(sigma2_p_hat / index))
}

my_IS_estimate <- get_IS_estimate(1e4, lambda = 3.5)
```

On peut comparer l'estimation avec la précédente. On aggrège les deux tableaux (en rajoutant une colonne Estimateur). Vous remarquerez la différence dans les échelles!

```
bind_rows(mutate(my_monte_carlo_estimate,
                 Estimateur = "MC"),
          mutate(my_IS_estimate,
                 Estimateur = "IS")) %>%
  ggplot(aes(x = index, y = p_hat)) +
  geom_ribbon(mapping = aes(ymin = inf_IC_emp, ymax = sup_IC_emp,
                            fill = Estimateur),
              alpha = 0.5) +
  geom_line(aes(color = Estimateur)) +
  labs(x = "Effort Monte Carlo", y = expression(hat(p)),
       title = "Estimation") +
  facet_wrap(~Estimateur, scales = "free_y") +
  geom_hline(yintercept = 1 - pnorm(3), # True value
             linetype = 2, col = "red") +
  theme(legend.position = "none")
```

Estimation



2 Cas problématique

2.1 Premier cas jouet

Afin de montrer qu'un mauvais choix de loi d'échantillonnage peut augmenter drastiquement la variance de l'estimateur, on peut prendre un exemple très simple.

Supposons qu'on veuille estimer par méthode de Monte Carlo $\mathbb{E}[X]$ où $X \sim \mathcal{N}(0, 1)$. On se propose de le faire par méthode de Monte Carlo standard et par échantillonnage préférentiel en tirant dans une loi $Y \sim \mathcal{N}(\mu, 1)$ où μ est choisi par l'utilisateur.

1. Quelle est la variance de l'estimateur de Monte Carlo standard?

Pour un Monte Carlo classique, on simule X_1, \dots, X_M i.i.d. de loi $X \sim \mathcal{N}(0, 1)$ et on prend la moyenne empirique. La variance de cet estimateur est $\frac{1}{M}$.

2. Quelle est la variance de l'estimateur par échantillonnage préférentiel?

Un simple calcul montre que cette variance est de $\frac{\sigma^2}{M}$!

2.2 Encore pire!

Soit X une variable aléatoire de densité $f_X(x) = 3x^{-4}\mathbf{1}_{x \geq 1}$ (on dit que X suit une loi de Pareto de paramètres $(1, 3)$).

1. Calculer $p = \mathbb{P}(X > 10)$

$$\mathbb{P}(X > 10) = \int_{10}^{\infty} 3x^{-4} \mathbf{1}_{x \geq 1} dx = \frac{1}{10^3}$$

2. Supposons qu'on veuille estimer p par méthode de Monte Carlo. Ecrire un estimateur de Monte Carlo standard utilisant un échantillon d'une loi de Pareto.

Soit X_1, \dots, X_n un échantillon de variables aléatoires i.i.d., de loi de Pareto de paramètre 1 et 3, pour approcher p , on propose l'estimateur:

$$\hat{p}_M^{MC} = \frac{1}{M} \sum_{k=1}^M \mathbf{1}_{X_k > 10}$$

3. Représenter graphiquement les performances empiriques de cet estimateur pour un effort Monte Carlo de $M = 10^4$. On utilisera la fonction `rpareto` du package `EnvStats`.

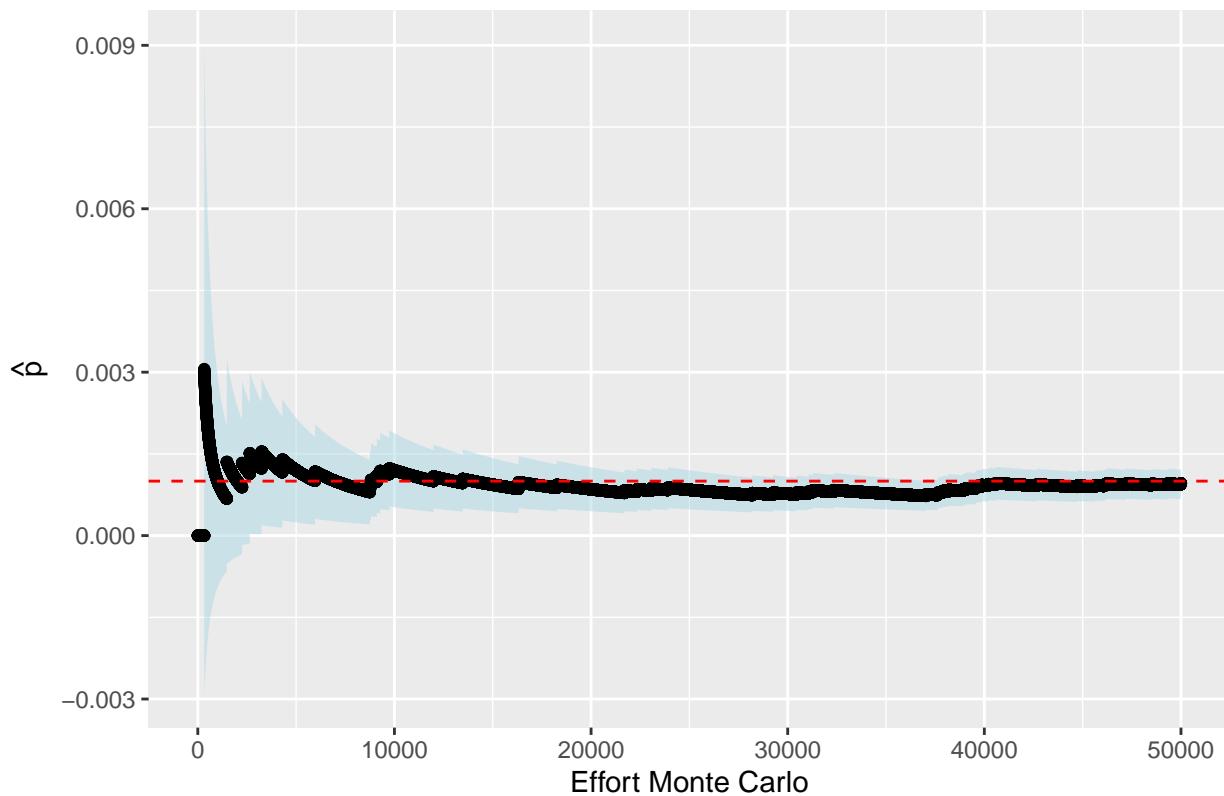
On conserve le squelette habituel.

```
get_MC_estimate_pareto <- function(M, threshold = 10){
  x_sample <- EnvStats::rpareto(M, 1, 3) # Simulation des X
  above_threshold <- x_sample > threshold
  z_975 <- qnorm(0.975) # Quantile de la loi normale
  p_hat <- cumsum(above_threshold) / (1:M) # Estimation de p
  # Fast way to compute  $E[f(x)^2] - E[f(x)]^2$  on the fly
  sigma2_p_hat <- cumsum(above_threshold^2) / (1:M) - p_hat^2
  # Stacking in a data_frame
  tibble(index = 1:M,
         phi_x = above_threshold,
         p_hat = p_hat) %>%
  mutate(sup_IC_emp = p_hat + z_975 * sqrt(sigma2_p_hat / index), # IC bounds
        inf_IC_emp = p_hat - z_975 * sqrt(sigma2_p_hat / index))
}

set.seed(123) # For reproducible results
my_MC_estimate_pareto <- get_MC_estimate_pareto(5e4)

my_MC_estimate_pareto %>%
  ggplot(aes(x = index, y = p_hat)) +
  geom_ribbon(mapping = aes(ymin = inf_IC_emp, ymax = sup_IC_emp),
              fill = "lightblue", alpha = 0.5) +
  geom_point() +
  labs(x = "Effort Monte Carlo", y = expression(hat(p)),
       title = "Estimation par Monte Carlo") +
  geom_hline(yintercept = 1 - EnvStats::ppareto(10, 1, 3), # True value
             linetype = 2, col = "red")
```

Estimation par Monte Carlo



L'estimateur est relativement instable

- On propose maintenant un estimateur par échantillonnage préférentiel comme dans l'exercice précédent. On choisit comme densité d'échantillonnage celle d'une loi exponentielle translatée de 10, de paramètre $\lambda = 1$. Donnez l'estimateur associé et mettez le en place sous R. Que constatez vous? Comment l'expliquez vous. Vous pourrez coder vous même la densité d'une loi de Pareto, ou utiliser la fonction `dpareto` du package EnvStats.

On note g la densité d'une loi exponentielle de paramètre $\lambda = 1$, translatée de 10.

L'estimateur d'échantillonnage préférentiel est donc:

$$\hat{p}_n^{IS} = \frac{1}{M} \sum_{k=1}^M \frac{f_X(Y_k)}{g(Y_k)}$$

où Y_1, \dots, Y_M est un échantillon i.i.d. de variables aléatoires de loi exponentielle translatée de 10.

Le squelette de la fonction est strictement identique à celui de l'exercice précédent

```
get_IS_estimate_pareto <- function(M, lambda, threshold = 10){
  y_sample <- threshold + rexp(M, rate = lambda) # Simulation Y
  above_threshold <- y_sample > threshold
  z_975 <- qnorm(0.975) # Quantile de la loi normale
  weights <- EnvStats::dpareto(y_sample, 1, 3) / dexp(y_sample - threshold, lambda)
  p_hat <- cumsum(above_threshold * weights) / (1:M)
  sigma2_p_hat = cumsum((above_threshold * weights)^2) / (1:M) - p_hat^2
  tibble(index = 1:M,
         phi_x = above_threshold,
```

```

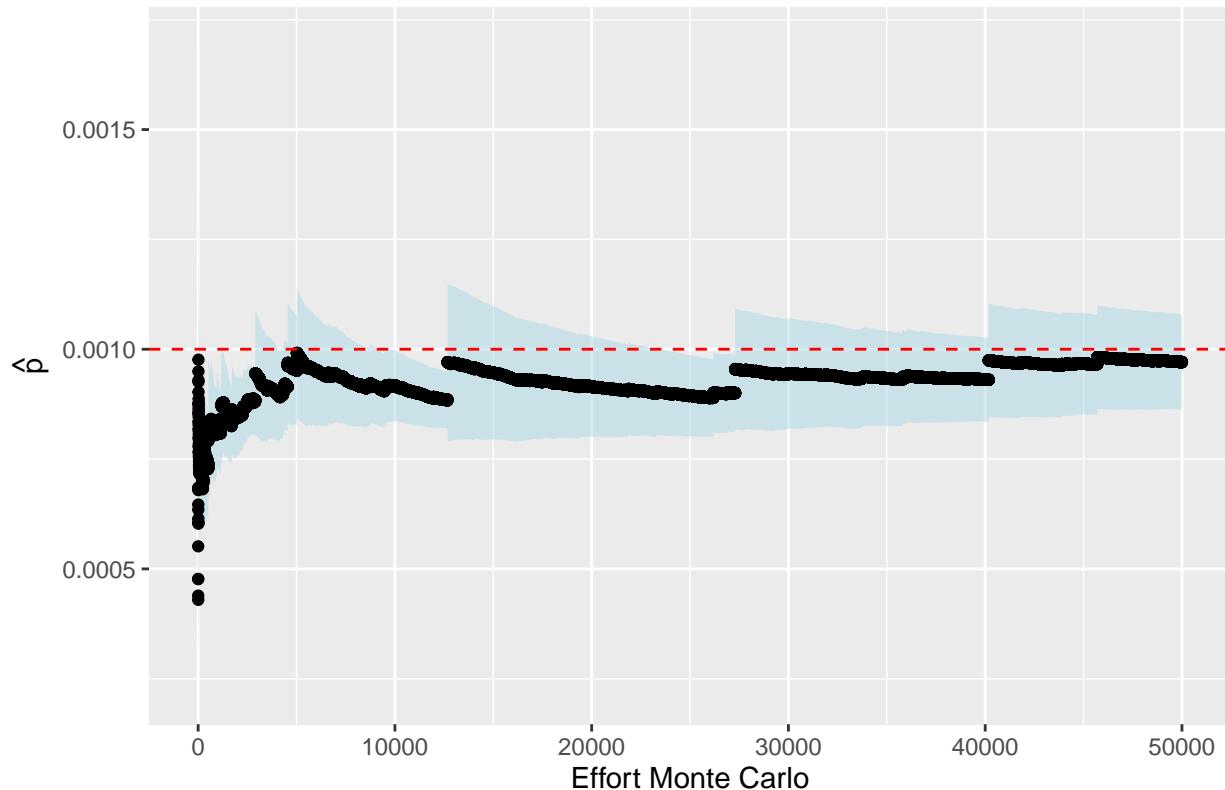
    p_hat = p_hat) %>%
  mutate(sup_IC_emp = p_hat + z_975 * sqrt(sigma2_p_hat / index), # IC bounds
        inf_IC_emp = p_hat - z_975 * sqrt(sigma2_p_hat / index))
}

set.seed(1)
my_IS_estimate_pareto <- get_IS_estimate_pareto(5e4, lambda = 1)

my_IS_estimate_pareto %>%
  ggplot(aes(x = index, y = p_hat)) +
  geom_ribbon(mapping = aes(ymin = inf_IC_emp, ymax = sup_IC_emp),
              fill = "lightblue", alpha = 0.5) +
  geom_point() +
  labs(x = "Effort Monte Carlo", y = expression(hat(p)),
       title = "Estimation par Monte Carlo") +
  geom_hline(yintercept = 1 - EnvStats::ppareto(10, 1, 3), # True value
             linetype = 2, col = "red")

```

Estimation par Monte Carlo



L'estimateur est encore plus instable! La variance ne semble pas nécessairement décroître avec M .

5. Que pouvez vous dire de la variance de cet estimateur?

Regardons la variance de \hat{p}_1^{IS}

$$\begin{aligned}
 \mathbb{V}[\hat{p}_1^{IS}] &= \mathbb{E}[(\hat{p}_1^{IS})^2] - p^2 \\
 &= \mathbb{E}\left[\left(\frac{f_X(Y)}{g(Y)}\right)^2\right] - p^2 \\
 &= \int_{10}^{\infty} \frac{f_X(z)^2}{g(z)^2} g(z) dz - p^2 \\
 &= \int_{10}^{\infty} 9z^{-8} e^{z-10} dz - p^2
 \end{aligned}$$

La variance est donc infinie! Notre estimateur est donc sans biais, mais de variance infinie (entre autres conséquences, le TCL ne s'applique pas).