Projet Poisson: Modèle de réparation imparfaite de Brown-Proschan

Arman Hosseini, Paul Slisse, Guillaume Staub, Jade Roumazeille-Peter

2025-03-14

Contents

1 Présentation du modèle

1.1 Introduction

Dans ce projet, nous allons nous intéresser au processus de Poisson à réparation imparfaite dévellopé par Brown et Proschan [2] en 1983 , retravaillé par Whitaker et Samaniego [1], que nous allons raccourcir par la suite en BP. Dans ce modèle, lorsque le système tombe en panne, celui-ci est réparé "parfaitement" avec une probabilité p: il redevient donc dans un état neuf. Sinon, le système est remis dans un état équivalent à celui précédant la panne.

Pour ce faire, commençons par introduire les notations qui nous seront nécessaires. On note T_i les temps entre chaque panne. On note A_i l'âge du système après la i-ème panne. Ainsi, $A_i = 0$ si l'on a une réparation parfaite après la i-ème panne et $A_i = A_{i-1} + T_i > 0$ sinon. On note F la fonction de répartition de l'âge sur $[0, +\infty[$ et S la fonction de survie associé à F notée $S(t) = P(T > t) = 1 - F(t)$.

Ainsi, les paramètres à estimer de notre modèle sont (p, F).

1.2 Estimation des paramètres (p, F)

Pour le modèle de BP, les paramètres (p, F) ne peuvent pas être estimés avec la seule connaissance de la distribution des T_i. En effet, on peut montrer que pour deux valeurs de p différentes et la même fonction F, il est possible d'obtenir la même distribution des T_i. Par exemple, si F est la fonction de répartition d'une loi exponentielle, alors les T_i suivent la même loi exponentielle qui est indépendante de p. Bien que la fonction F puisse être estimée via les T_i, le problème est que l'on ne peut estimer le paramètre p avec la suite des T_i.

Pour remédier à cela, nous allons introduire une nouvelle suite de variables aléatoires de Bernoulli de paramètre p que l'on note Z_i. A la i-ème panne, si la réparation est parfaite alors $Z_i = 1$ et si elle est imparfaite, $Z_i = 0$. Rajouter cette information sur le type de panne qui est uniquement determiné par le paramètre p rend le problème d'estimation soluble.

A partir de maintenant, nous allons faire les hypothèses suivantes pour notre étude : -$p > 0$ -F est absolument continue (admet une densité f) -$F(0) = 0$ (pas de réparation instantanée)

Nous allons observer le système jusqu'à la m-ème réparation parfaite. On note n le nombre total de pannes et donc $n - m$ est le nombre de réparations imparfaites.

La suite des âges observés (a_i) est entièrement déterminée par les couples (t_i, z_i).

$$L(p, F) = f(t_1)p^{z_1}(1-p)^{1-z_1} \times \frac{f(t_2 + a_1)}{S(a_1)}p^{z_2}(1-p)^{1-z_2} \times ... \times \frac{f(t_n + a_{n-1})}{S(a_{n-1})}p^{z_n}(1-p)^{1-z_n}$$

Pour comprendre cette formulation de la vraisemblance, on considère le cas n = 2 (les n > 2 se comprennent par récurrence).

$$L(p, F|(t_1, t_2), (z_1, z_2)) = f_{T_1,Z_1}(t_1, z_1)f_{T_2,Z_2}((t_2, z_2)|(T_1 = t_1, Z_1 = z_1))$$

Par indépendance des événements "panne" et "la réparation est parfaite", on a :

$$f_{T_1,Z_1}(t_1, z_1) = f(t_1)p^{z_1}(1-p)^{1-z_1}$$

Ensuite, on a deux cas possibles :

Si $z_1 = 0$ (réparation imparfaite), on a $a_1 = t_1$:

$$f_{T_2,Z_2}((t_2, z_2)|(T_1 = t_1, Z_1 = z_1)) = \frac{f(t_2 + a_1)}{S(a_1)}p^{z_2}(1-p)^{1-z_2}$$

Si $z_1 = 1$ (réparation imparfaite), on a $a_1 = 0$ et $S(a_1) = 1$, d'où :

$$f_{T_2,Z_2}((t_2,z_2)|(T_1=t_1,Z_1=z_1)) = f(t_2)p^{z_2}(1-p)^{1-z_2} = \frac{f(t_2+a_1)}{S(a_1)}p^{z_2}(1-p)^{1-z_2}$$

En notant x_i l'âge observé juste avant la i-ème panne : si $z_i = 0$, on a directement que $a_i = x_i$, si $z_i = 1$, $a_i = 0$, $x_i = a_{i-1} + t_i$, mais on peut tout de même écrire que $S(a_i) = S(x_i)^{1-z_i}$. Si dans la formule de la vraisemblance, on remplace les a_i par les x_i et qu'on factorise les p et $(1-p)$ ensemble à l'aide des n et m, on obtient:

$$L(p,F) = p^m(1-p)^{n-m}f(x_1)\frac{f(x_2)}{S(x_1)^{1-z_1}} \times ... \times \frac{f(x_n)}{S(x_{n-1})^{1-z_{n-1}}}$$

Avec cette écriture de la vraisemblance, on trouve le MLE de p: $\hat{p} = \frac{m}{n}$. Ce résultat est cohérent, on estime la probabilité p de réparation parfaite en calculant le ratio de réparations parfaites sur le nombre total de pannes.

Maintenant qu'on a le MLE de p, intéressons nous à F.

Pour trouver un estimateur empirique du MLE de F, on peut maximiser la partie ne dépendant pas de p dans la vraisemblance au dessus. Cela équivaut à déterminer F, qui est une fonction continue par morceaux, maximisant, l'estimation suivante de la vraisemblance:

$$l(F) = \prod_{i=1}^{n} \frac{(F(x_{(i)}) - F(x_{(i)}^-))}{S(x_{(i-1)})^{1-z_{(i-1)}}}$$

où la densité f en x_i est approximée par $F(x_{(i)}) - F(x_{(i)}^-)$,les $x_{(i)}$ sont les âges x_i avant les pannes ordonnés. Les $z_{(i)}$ ne sont pas ordonnés mais sont les z_i réindexés pour que $z_{(i)}$ correspondent au mode de réparation après la panne survenue à l'âge $x_{(i)}$. En particulier notre MLE empirique ne sera pas nécessairement continu à l'inverse de la vraie fonction F.

Malheureusement, le maximum de la fonction $l(F)$ n'est pas toujours atteignable. En effet, il faut que $z_{(n-1)} = 1$ [1].

Plaçons nous dans le cas où $z_{(n-1)} = 1$. Définissons $\phi_i = \frac{S(x_{(i)})}{S(x_{(i-1)})}$ la probabilité de survivre jusqu'à l'âge $x_{(i)}$ sachant que le système n'a pas eu de réparation parfaite depuis une durée $x_{(i-1)}$. Ainsi, notre problème de maximisation revient à maximiser par rapport à ϕ:

$$l(\phi) = \prod_{i=1}^{n}(1-\phi_i)\phi_i^{k_i}$$

où $k_i = \sum_{j=i}^{n-1} z_{(j)}$ est le nombre de réparation parfaites pour des pannes intervenues à un âge supérieur à $x_{(i)}$.

On montre facilement que le maximum de cette fonction est atteint en :

$$\hat{\phi}_i = \frac{k_i}{k_i+1}, 1 \leq i \leq n-1; \hat{\phi}_n = 0$$

Ainsi, on obtient le MLE empirique de la fonction de survie S:

$$\hat{S}(t) = \begin{cases} 1, & t < x_{(1)} \\ \prod_{j=1}^{i} \hat{\phi}_j, & x_{(i)} < t < x_{(i+1)}, \quad i = 1, \ldots, n-1 \\ 0, & t \geq x_{(n)} \end{cases}$$

Dans le cas où $z_{(n-1)} = 0$, on a toujours le même maximiseur de $l(\phi)$ mais désormais par définition, $k_{n-1} = 0$ et donc $\hat{\phi}_{n-1} = 0$ ce qui implique que $\hat{S}(t) = 0, t \geq x_{(n-1)}$. Ainsi, on voit que notre approximation de la

vraisemblance $l(F)$ n'est pas définie car dans le dernier terme du produit, on a une division par 0. C'est pourquoi dans ce cas précis, il n'existe pas de MLE empirique issu de la maximisation de notre estimation de la vraisemblance. Ce MLE empirique est en réalité un "neighbourhood MLE" de F dans une certaine topologie, on peut tout de même l'utiliser pour $z_{(n-1)} = 0$.

Ce modèle de BP peut-être vu comme une suite de réparations parfaites indépendantes, ainsi, on peut se restreindre aux temps de réparations parfaites pour étudier les propriétés du modèle. On a le résultat suivant (citer BP 1983) qui nous dit que si Y_1 est l'âge du système à la première réparation parfaite, alors la fonction de survie S_y de Y_1 vérifie:

$$S_y(t) = (S(t))^p$$

En effet, en notant r le taux de hasard associé à T, le taux de hasard associé à Y_1 est égal à pr car la probabilité qu'une réparation ait lieu à l'instant t sachant qu'il n'y avait pas précédemment est égale à $r(t)$, et elle est alors parfaite avec une probabilité p, avec indépendance entre les deux phénomènes. On peut ensuite utiliser le lien entre taux de hasard et fonction de survie:

$$S_y(t) = exp(-\int_0^t pr(x)dx) = (S(t))^p$$

En établissant la distribution empirique du processus de Poisson inhomogène obtenu en ne considérant que les durées entre deux réparations parfaites $Y_1, \dots, Y_m$. On obtient ainsi une approximation de S_y, que l'on note $\hat{S}_y$. Or $S_y(t) = (S(t))^p$, donc on obtient via $1 - \hat{S}_y^{n/m}$ un autre estimateur de la fonction de répartition.

On peut également établir la distribution empirique des $T_i|Z_{i-1} = 1$, c'est-à-dire des durées entre chaque réparation parfaite et la réparation suivante. On obtient ainsi une approximation de la densité, et en la cumulant, une nouvelle estimation de la fonction de répartition.

Ces deux derniers estimateurs n'exploitent pas toutes les informations issues des observations de T et Z, contrairement au premier, on peut donc supposer qu'ils seront moins performants.

2 Modelisation numerique

2.1 Simulation d'un processus de Weibull

Les Z_i suivent une loi Binomiale par définition: $\mathbb{P}(Z_i = 1) = p$.

On simule ici le modèle. Pour ce faire, on commence par créer deux fonctions: $simulPPh1$ et $invWeibull$. La fonction $simulPPh1$ permet de simuler un processus de Poisson homogène: on simule k fois une variable aléatoire suivant la loi exponentielle de paramètre 1 ce qui nous donne une liste de nombres $[n_1, n_2, ..., n_k]$. On renvoie ensuite les temps d'arrivée simulés (notés A_k dans la fonction $simulPPIWeibull$) qui représentent la somme cumulée $[n_1, n_1 + n_2, ..., n_1 + ... + n_k]$.

```
simulPPh1 <- function(k)
{
  return(cumsum(rexp(k, 1)))
}
```

La fonction $invWeibull$ renvoie l'intensité cumulée inverse de Weibull: Λ^{-1}. Montrons comment nous l'avons trouvé:

Nous avons prouvé en cours que la fonction d'intensité cumulée est égale à $\Lambda(t) = (\frac{t}{\alpha})^\beta$ avec α, β des paramètres strictement positifs. Calculons l'inverse de cette fonction:

$$y = (\frac{t}{\alpha})^\beta \iff y^{\frac{1}{\beta}} = \frac{t}{\alpha} \iff t = \alpha y^{\frac{1}{\beta}}$$

Ainsi, $\Lambda^{-1}(y) = \alpha y^{\frac{1}{\beta}}$. On l'implémente donc avec comme paramètres par défaut $\beta = 2$ et $\alpha = 1$.

```r
invWeibull <- function(y, alpha=1, beta=2)
{
  return(alpha*y**(1/beta))
}
```

On peut maintenant simuler le modèle BP. Pour cela, on génère une variable aléatoire géométrique de paramètre p, représentant le nombre k de réparations imparfaites nécessaires avant d'atteindre la première réparation parfaite. On simule ensuite le processus homogène comme expliqué précédemment et on retourne les temps $T_i = \Lambda^{-1}(A_i)$ et les valeurs de la fonction de survie de la loi Weibull: $S(t) = exp(-(\frac{t}{\alpha})^{\beta})$ créant ainsi un processus de Poisson inhomogène. On réitère cela m fois où m représente le nombre de réparations parfaites souhaité.

2.2 Première simulation d'un processus de Brown-Proschan

```r
simulPPIWeibull <- function(m, alpha, beta, p)
{
  k <- rgeom(1, p)+1
  A_k = simulPPh1(k)
  T_k = c(invWeibull(A_k, alpha, beta))
  S_k = c(exp(-(head(T_k,-1)/alpha)**beta),1)
  T_succes = c(tail(T_k,1)) #le temps où il ya la rep parfaite
  S_succes = c(tail(S_k,1)) #et sa valeur de fonction de survie associée
  for(i in 1:m-1)
  {
   k <- rgeom(1, p)+1
   A_k = simulPPh1(k)
   T_k_temp = c(invWeibull(A_k, alpha, beta))
   T_k = c(T_k, tail(T_k,1) + T_k_temp)
   S_k = c(S_k,exp(-(T_k_temp[-1]/alpha)**beta),1)
   T_succes = c(T_succes, tail(T_k,1))
   S_succes = c(S_succes, tail(S_k,1))
  }
  return(list(T = T_k,S=S_k, T_succes = T_succes,S_succes=S_succes))
}
```

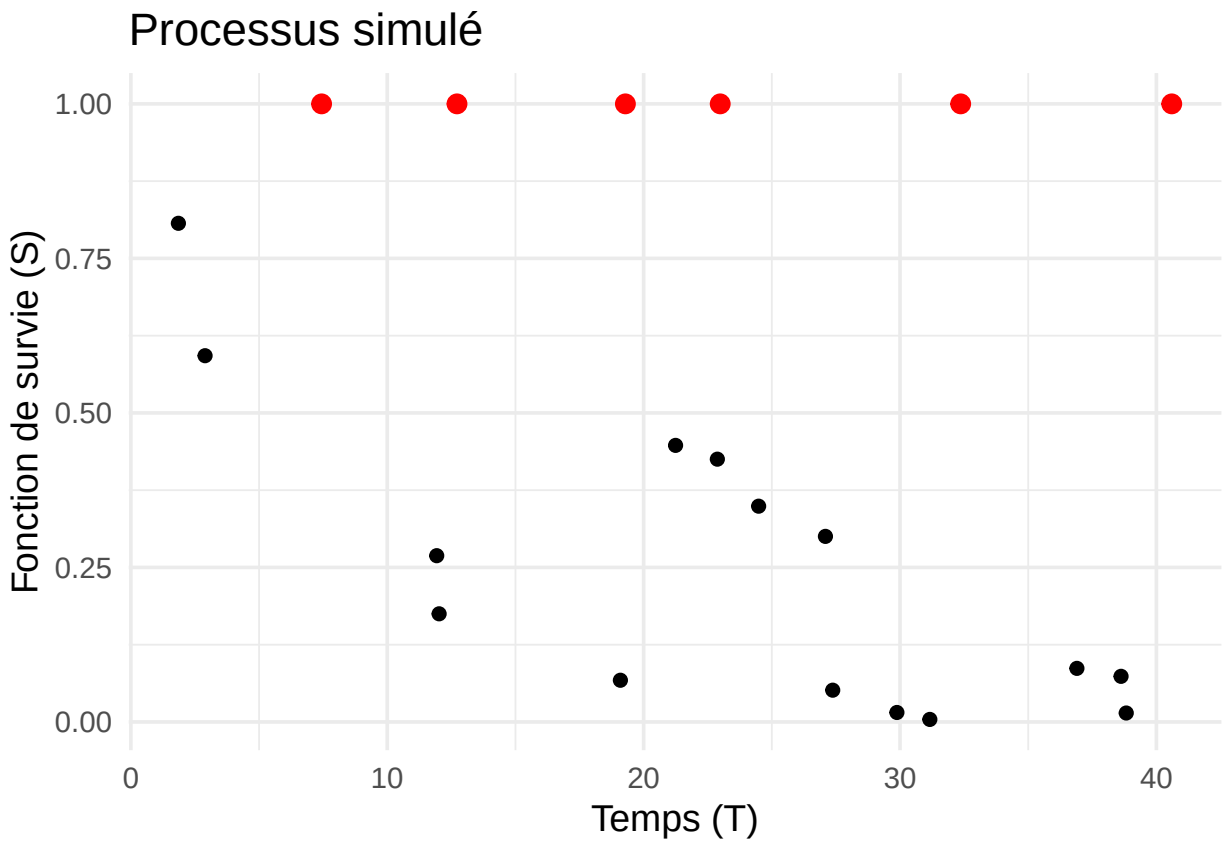
On va désormais afficher les résultats d'une simulation de notre processus de BP pour vérifier que notre code marche bien. Pour cela, on affiche pour chaque temps de panne, la valeur de la fonction de survie théorique (calculable explicitement dans le cas de Weibull).

```r
m <- 5
p <- 0.2
result <- simulPPIWeibull(m, 4, 2, p)
print(result)
```

```
## $T
##  [1]  1.853557  2.893590  7.438877 11.929139 12.022396 12.720042 19.093259
##  [8] 19.287986 21.245498 22.874944 22.987279 24.485143 27.090727 27.374849
## [15] 29.878426 31.162684 32.365639 36.898805 38.620675 38.822827 40.597679
##
## $S
##  [1] 0.806759583 0.592559631 1.000000000 0.269000946 0.174965068 1.000000000
##  [7] 0.067466896 1.000000000 0.447471718 0.425156440 1.000000000 0.349101461
## [13] 0.300239970 0.051406336 0.015339448 0.004098633 1.000000000 0.086696413
## [19] 0.073832570 0.014473936 1.000000000
##
```

```
## $T_succes
## [1]  7.438877 12.720042 19.287986 22.987279 32.365639 40.597679
##
## $S_succes
## [1] 1 1 1 1 1 1
```

```r
df_result <- data.frame(T = result$T, S = result$S)
df_succes <- data.frame(T = result$T_succes, S = result$S_succes)

ggplot(df_result, aes(x = T, y = S)) +
  geom_point(color = "black", size = 2) +
  geom_point(data = df_succes, aes(x = T, y = S), color = "red", size = 3) +
  labs(title = "Processus simulé",
       x = "Temps (T)",
       y = "Fonction de survie (S)") +
  theme_minimal(base_size = 14)
```



On voit donc que le graphe est cohérent d'après les résulats des différents vecteurs. En effet, lorsqu'il n'y a pas de réparations parfaites, l'âge du système ne cesse d'augmenter et la valeur de la fonction de survie tend vers 0 et à l'inverse, lorsqu'une réparation est parfaite, son âge retombe à zéro et la fonction de survie vaut à nouveau 1.

2.3 Estimation du paramètre p par maximum de vraisemblance

On implémente le calcul du MLE trouvé précedemment :$\hat{p} = m/n$.

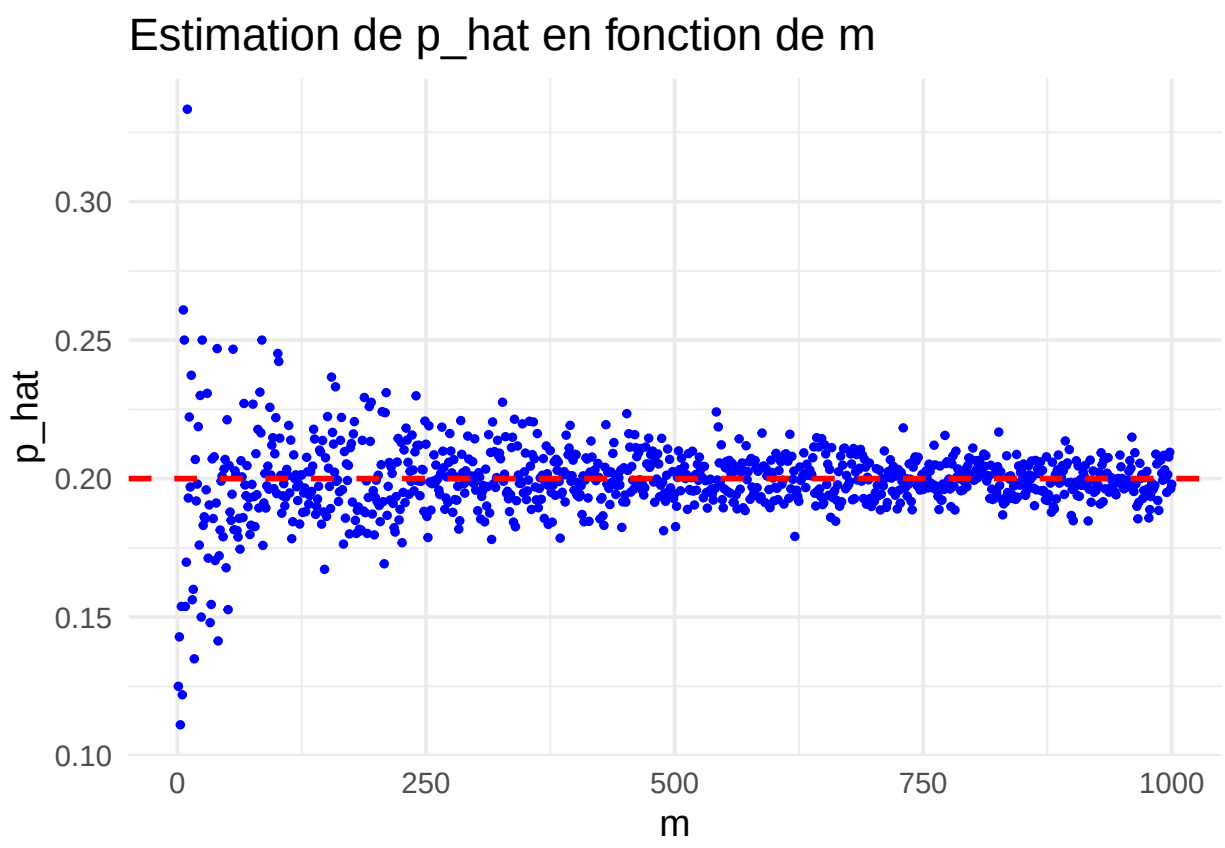
```r
MLE_P <- function(m,n)
{
```

```
  return(m/n)
}
```

Pour observer le bon comportement de notre estimateur, on va simuler des processus pour des valeurs de m croissantes et calculer à chaque fois notre $\hat{p}$.

```
p_hat <- c()
for (m in 1:1000) {
  process <- simulPPIWeibull(m, 1, 2, 0.2)
  p_hat <- c(p_hat, m / length(process$T))
}

df <- data.frame(m = 1:1000, p_hat = p_hat)

  ggplot(df, aes(x = m, y = p_hat)) +
    geom_point(color = "blue", size = 1) +
    geom_hline(yintercept = 0.2, color = "red", linetype = "dashed", size = 1) +
    labs(title = "Estimation de p_hat en fonction de m",
         x = "m",
         y = "p_hat") +
    theme_minimal(base_size = 14)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



On observe que plus m augmente, plus notre estimation ponctuelle se rapproche de la vraie valeur de p. On observe également que la variance diminue.

2.4 Estimation de l'intensité du processus par plusieurs méthodes

2.4.1 Estimation par maximum de vraisemblance ou neighbourhood MLE

On rappelle la formule du MLE qui a des garanties théoriques uniquement si $z_{(n-1)} = 1$ (voir première partie du document):

$$\hat{S}(t) = \begin{cases} 1, & t < x_{(1)} \\ \prod_{j=1}^{i} \hat{\phi}_j, & x_{(i)} < t < x_{(i+1)}, \quad i = 1, \dots, n-1 \\ 0, & t \geq x_{(n)} \end{cases}$$

On rappelle également que cet estimateur peut être utilisé pour $z_{(n-1)} = 0$ mais devient un neighbourhood MLE et est nul pour $t \geq x_{(J+1)}$ où J est le dernier indice pour lequel $z_{(J)} = 1$ (en dehors de $z_{(n)}$). On définit une nouvelle fonction $simulPPIWeibull2$, qui s'inspire fortement de la première simulation mais qui ne renvoie pas les mêmes choses. Ici on retourne les instants de pannes, la type de réparation associé à chaque panne ($= Z$) ainsi que les instants de réparations parfaites (sous ensemble de T).

```
simulPPIWeibull2 <- function(m, alpha, beta, p)
{
  k <- rgeom(1, p)+1
  A_k = simulPPh1(k)
  T_k = c(invWeibull(A_k, alpha, beta))
  Z_k = c(rep(0,k-1),1)
  T_succes = tail(T_k,1)
  for(i in 1:m-1)
  {
   k <- rgeom(1, p)+1
   A_k = simulPPh1(k)
   T_k_temp = c(invWeibull(A_k, alpha, beta))
   T_k = c(T_k, tail(T_k,1) + T_k_temp)
   Z_k = c(Z_k,rep(0,k-1),1)
   T_succes = c(T_succes, tail(T_k,1))
  }
  return(list(T = T_k,Z=Z_k, T_succes = T_succes))
}
```

```
res<-simulPPIWeibull2(50,4,2,0.2)
Z<-res$Z
T<-res$T
```

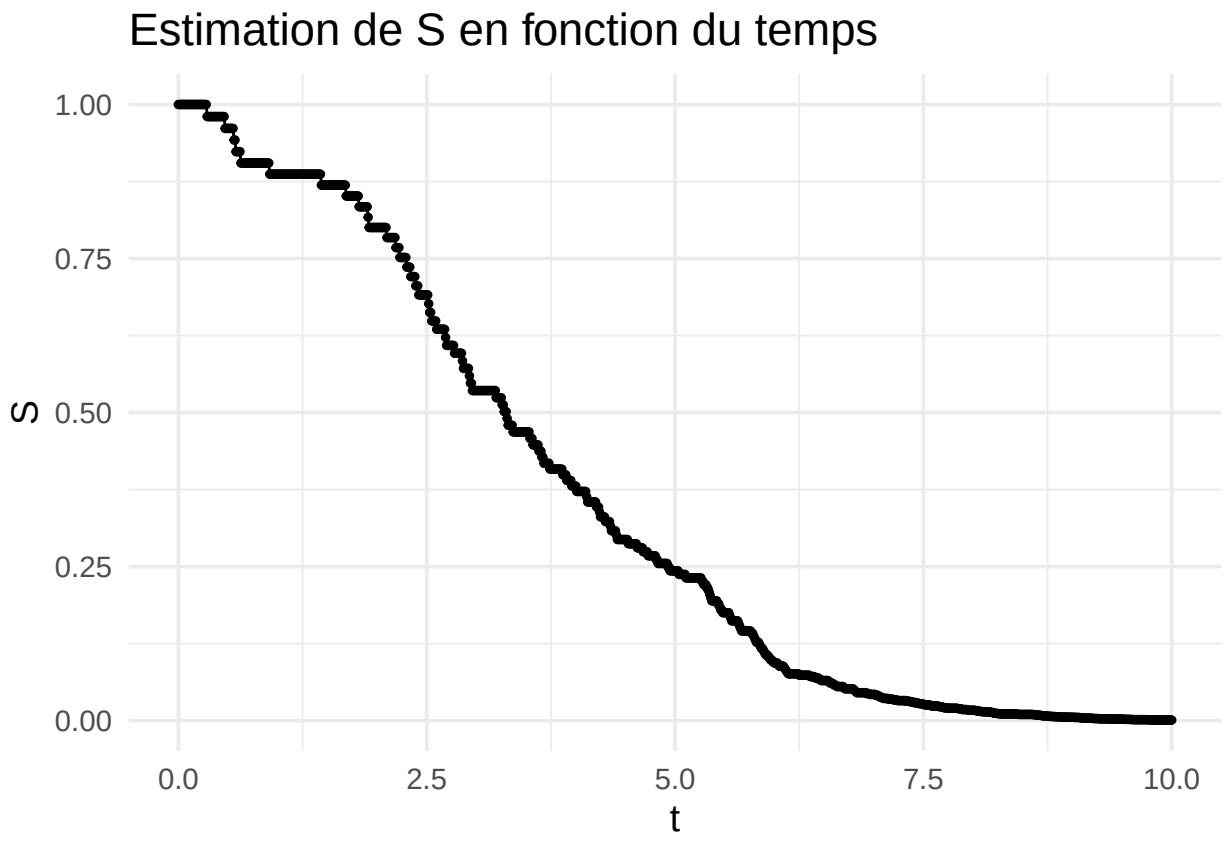
On implémente maintenant le calcul du MLE (ou neighbourhood MLE en fonction des cas) trouvé précedemment. La fonction MLE_S renvoie les vecteurs $x_{()}$ et $\hat{\phi}$ permettant de construire le MLE. On a ensuite une deuxième fonction $plot\_MLE\_S$ qui permet de construire graphiquement le MLE en retournant ses valeurs pour une séquence de temps t.

```
MLE_S <- function(T,Z)
{
  n<-length(Z)
  x <- c()
  last_perfect <- 0
  for (i in 1:n){
    x <- c(x,T[i]-last_perfect)
  if (Z[i]==1){
    last_perfect<-T[i]
```

```
  }
  }
  idx<-order(x)
  x_order<-x[idx]
  z_order<-Z[idx]
  phi_hat<-c()
  for (i in 1:(n-1)){
    k<-sum(z_order[i:(n-1)])
    phi_hat<-c(phi_hat,k/(k+1))
  }
  phi_hat<-c(phi_hat,0)
  return(list(x_order = x_order,phi_hat=phi_hat))
}
plot_MLE_S <-function(t,x_order,phi_hat) {
  x_order<-c(x_order,Inf)
  S_hat<-rep(1,length(t))
  # Inférieur au premier temps de panne
  count<-1
  while (t[count]<x_order[1]){
    count<-count+1
  }
  i<-1
  phi<-phi_hat[1]
  #Entre deux temps de pannes ou supérieur au dernier
  for (j in count:length(t)){
    if (t[j]>x_order[i+1]){
      phi<-phi*phi_hat[i]
      i<-i+1
    }
    S_hat[j]=phi
  }
  return(S_hat)
}
```

On teste le calcul et l'affichage de notre estimateur sur une simulation simple.

```
r<-MLE_S(T,Z)
t<-seq(0,10,0.01)
S_hatBP <- plot_MLE_S(t,r$x_order,r$phi_hat)
df <- data.frame(t = t, S_hat = S_hatBP)
ggplot(df, aes(x = t, y = S_hatBP)) +
  geom_point(color = "black", size = 1) +
  geom_line()+
  labs(title = "Estimation de S en fonction du temps",
       x = "t",
       y = "S") +
  theme_minimal(base_size = 14)
```



2.4.2 Estimation de l'intensité du processus en utilisant les temps de pannes à la suite d'une réparation parfaite

On va maintenant utiliser la distribution empirique des $T_i|Z_{i-1} = 1$, c'est-à-dire des durées entre chaque réparation parfaite et la réparation suivante pour obtenir une approximation de la densité. En la cumulant, on obtient une estimation de la fonction de répartition. On définit comme avant 2 fonctions, une nous retournant tout le nécessaire pour construire l'estimateur et une seconde pour construire l'estimateur et retourner les valeurs en une séquence de temps t donnée.

```
MLE_emp_1 <- function(T,Z)
{
  n<-length(Z)
  delta_t <- c()
  last_perfect <- 0
  for (i in 1:(n-1)){
    if(Z[i]==1){
      delta_t <- c(delta_t,T[i+1]-T[i])
    }
  }
  idd<-order(delta_t)
  delta_t_ordered <- delta_t[idd]
  return(delta_t_ordered)
}

plot_MLE_emp_1 <- function(t, x)
{
```

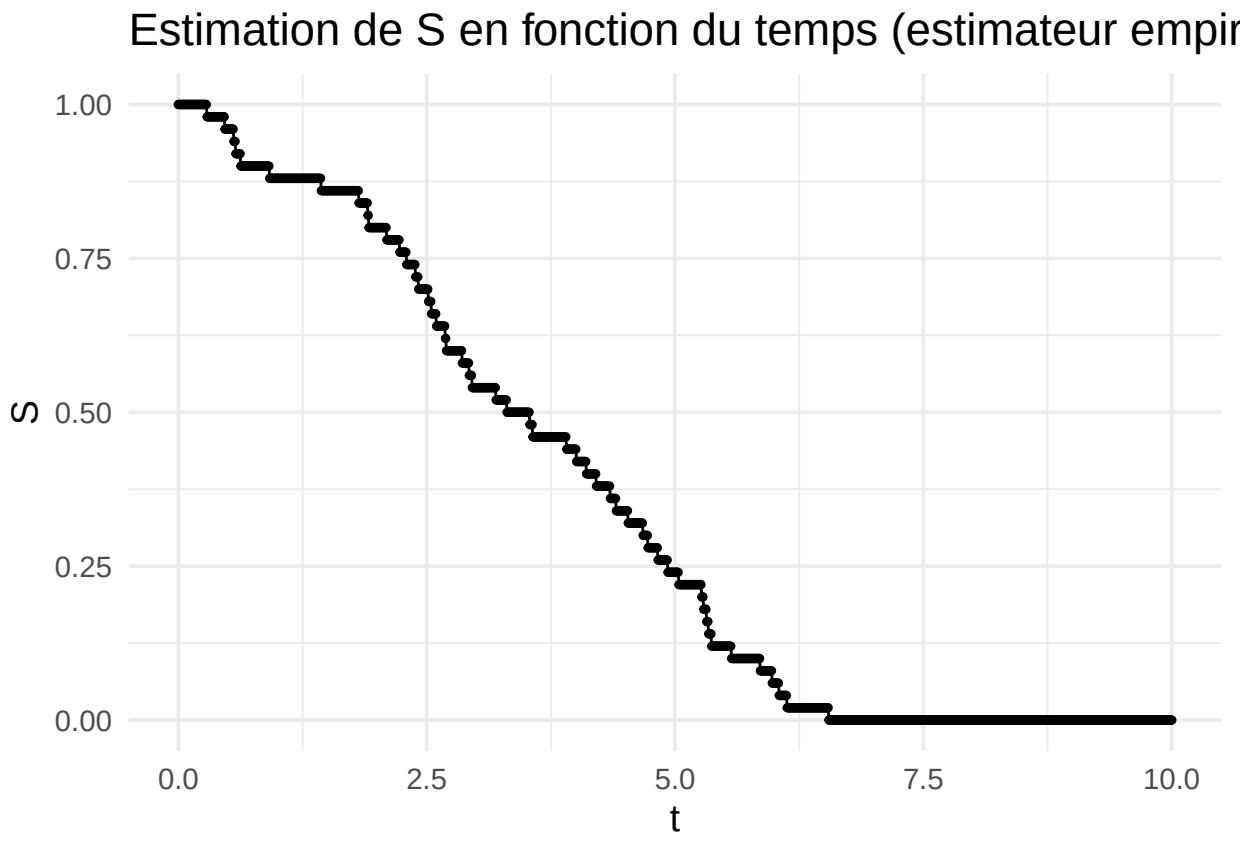
```
  num_perfect <- length(x)
  x<-c(x,Inf)
  S_hat<-rep(1,length(t))
  # Inférieur au premier temps de panne
  count<-1
  while (t[count]<x[1]){
    S_hat[count] <- 1
    count<-count+1
  }
  i<-1
  S_hat[count]<-(num_perfect - 1)/num_perfect
  #Entre deux temps de pannes ou supérieur au dernier
  for (j in count:length(t)){
    if (t[j]>x[i+1]){
      i<-i+1
    }
    S_hat[j]=(num_perfect - i)/num_perfect
  }
  return(S_hat)
}
```

On teste l'affichage de l'estimateur sur la même simulation que l'estimateur précédent.

```
idd<-MLE_emp_1(T,Z)
idd
```

```
##  [1] 0.2892069 0.4647671 0.5536490 0.5722335 0.6208866 0.9122589 1.4337080
##  [8] 1.8128988 1.9009104 1.9022039 2.0988061 2.2209289 2.2970509 2.3806662
## [15] 2.4107576 2.5141777 2.5473267 2.5959086 2.6813598 2.6893403 2.8543719
## [22] 2.9231403 2.9579725 3.1918126 3.3060178 3.5375719 3.5657135 3.9015647
## [29] 4.0004368 4.1077089 4.2009949 4.3468425 4.4030827 4.5256253 4.6757087
## [36] 4.7296936 4.8262141 4.9236650 5.0369153 5.2698873 5.2839027 5.3138321
## [43] 5.3396305 5.3665156 5.5619336 5.8521385 5.9720842 6.0476597 6.1281931
## [50] 6.5422389
```

```
S_hat_e1 <- plot_MLE_emp_1(t,idd)
df <- data.frame(t = t, S_hat_e1 =S_hat_e1)
ggplot(df, aes(x = t, y =S_hat_e1)) +
  geom_point(color = "black", size = 1) +
  geom_line()+
  labs(title = "Estimation de S en fonction du temps (estimateur empirique 1)",
       x = "t",
       y = "S") +
  theme_minimal(base_size = 14)
```



2.4.3 Estimation de l'intensité du processus avec uniquement les temps de réparations parfaites

On va maintenant utiliser la distribution empirique du processus de Poisson inhomogène obtenu en ne considérant que les durées entre deux réparations parfaites $Y_1, \dots, Y_m$. On obtient ainsi une approximation de S_y, que l'on note $\hat{S}_y$. Or $S_y(t) = (S(t))^p$, donc on obtient via $1 - \hat{S}_y^{n/m}$ notre estimateur de la fonction de répartition. On reprend les mêmes conventions de codes que pour les deux estimateurs précédent.

```
MLE_emp_2_c <- function(T,Z)
{
  n<-length(Z)
  perfects <- c()
  last_perfect <- 0
  for (i in 1:n){
    if(Z[i]==1){
      perfects <- c(perfects,T[i])
    }
  }
  inter_perfects <- c()
  inter_perfects <- c(perfects[1])
  for (i in 1:length(perfects)-1){
    inter_perfects <- c(inter_perfects, perfects[i+1] - perfects[i])
  }
  idd<-order(inter_perfects)
  inter_perfects_ordered <- inter_perfects[idd]
  return(inter_perfects_ordered)
}
```

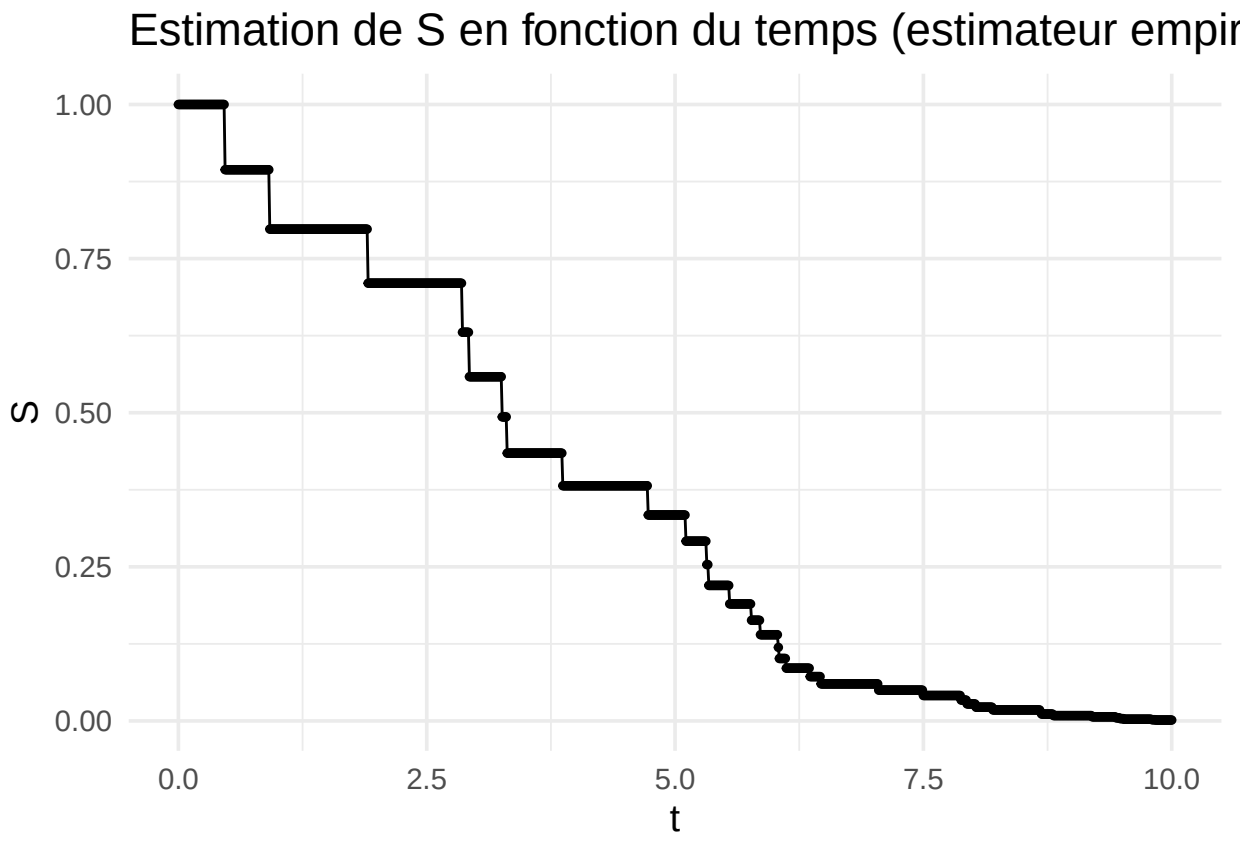
```
plot_MLE_emp_2 <- function(t, ipo, Z)
{
  num_repairs <- length(Z)
  num_perfect <- length(ipo)
  ipo<-c(ipo,Inf)
  S_hat_y<-rep(1,length(t))
  # Inférieur au premier temps de panne
  count<-1
  while (t[count]<ipo[1]){
    S_hat_y[count] <- 1
    count<-count+1
  }
  i<-1
  S_hat_y[count]<-(num_perfect - 1)/num_perfect
  #Entre deux temps de pannes ou supérieur au dernier
  for (j in (count+1):length(t)){
    if (t[j]>ipo[i+1]){
      i<-i+1
    }
    S_hat_y[j]=(num_perfect - i)/num_perfect
  }

  S_hat <- (S_hat_y)**(num_repairs / num_perfect)

  return(S_hat)
}
```

On teste l'affichage de l'estimateur sur la même simulation que pour les estimateurs précédents.

```
ipo<-MLE_emp_2_c(T,Z)
ipo
```

```
##  [1]  0.4647671  0.9122589  1.9022039  2.8501046  2.9231403  3.2513694
##  [7]  3.3060178  3.8610829  4.7296936  5.1015176  5.3138321  5.3396305
## [13]  5.5432250  5.7664666  5.8521385  6.0327295  6.0476597  6.1173296
## [19]  6.3553964  6.4686292  7.0445482  7.4976436  7.8789321  7.9310878
## [25]  8.0218592  8.1946354  8.6708302  8.6736175  8.8073864  9.1909594
## [31]  9.4312037  9.4742356  9.5143975  9.7992094  9.8314732 10.1222066
## [37] 10.1315789 10.3556952 10.4802966 10.5315072 10.5949788 10.6921700
## [43] 10.7362946 11.4339469 11.4768135 12.0343413 13.5443545 14.5466661
## [49] 15.9550520 19.2211546 22.3001296
```

```
S_hat_e2 <- plot_MLE_emp_2(t,ipo, Z)
df <- data.frame(t = t, S_hat_e2 =S_hat_e2)
ggplot(df, aes(x = t, y =S_hat_e2)) +
  geom_point(color = "black", size = 1) +
  geom_line()+
  labs(title = "Estimation de S en fonction du temps (estimateur empirique 2)",
       x = "t",
       y = "S") +
  theme_minimal(base_size = 14)
```



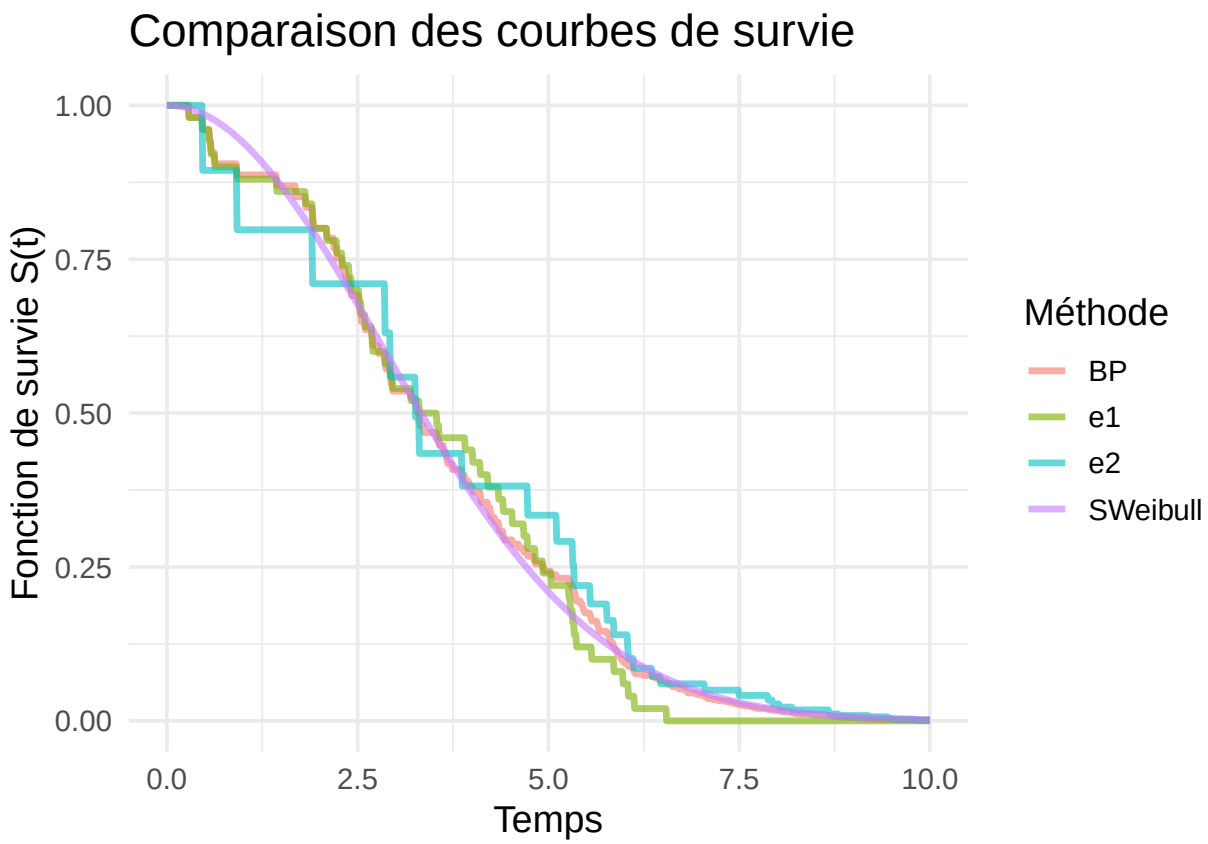
2.4.4 Comparaison des estimateurs

Pour comparer nos estimateurs de la fonction de survie S, on va évaluer la fonction de survie théorique d'un processus de Weibull, que l'on peut calculer, en une séquence de temps positifs. On va ensuite afficher sur le même graphe cette courbe exacte et celles obtenues avec chacun des estimateurs.

```
SWeibull <- function(y, alpha=1, beta=2)
{
  return(exp(-(y/alpha)**beta))
}
```

```
df <- data.frame(
  t = t,
  SWeibull = SWeibull(t, 4, 2),
  BP = S_hatBP,
  e1 = S_hat_e1,
  e2 = S_hat_e2
)

df_long <- df %>%
  pivot_longer(cols = -t, names_to = "Méthode", values_to = "Survie")

# Plot
ggplot(df_long, aes(x = t, y = Survie, color = Méthode)) +
  geom_line(size = 1.2,alpha = 0.6) +
  labs(title = "Comparaison des courbes de survie",
       x = "Temps",
```

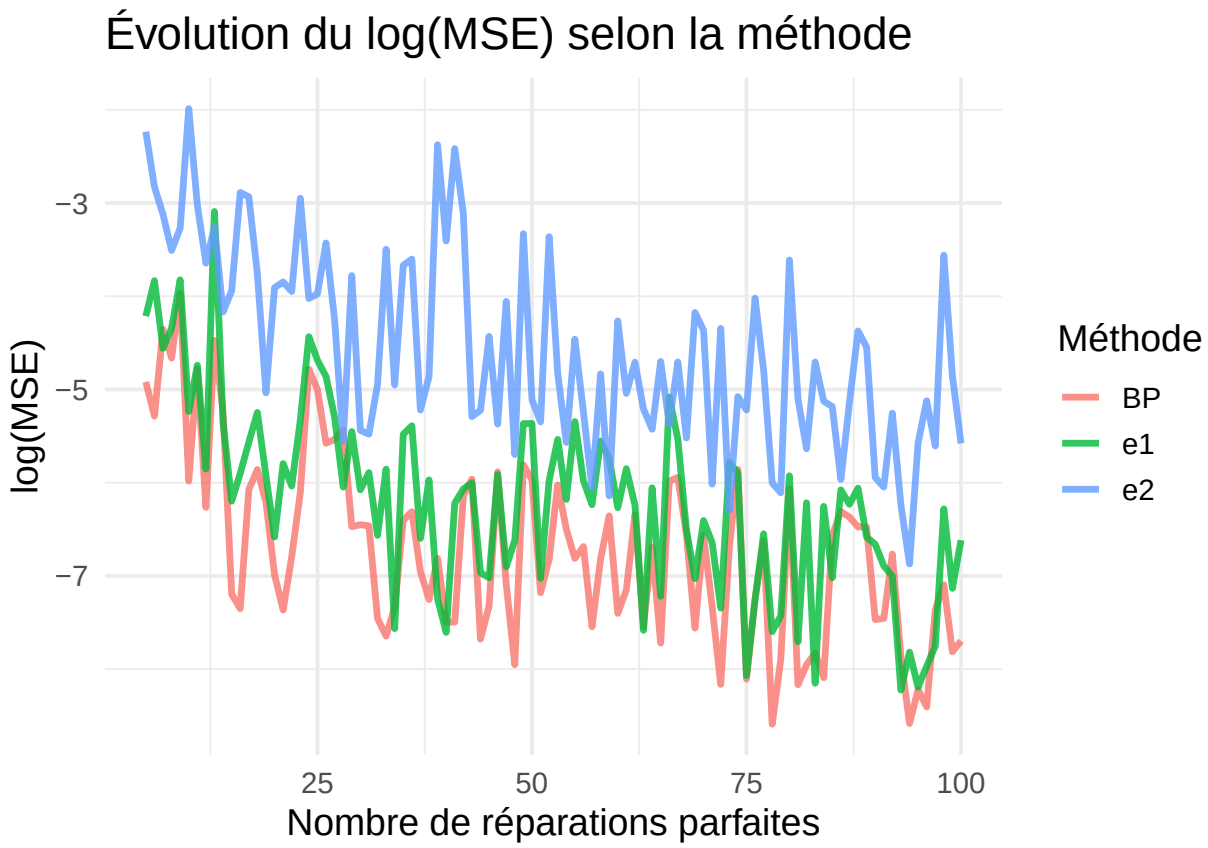
```
      y = "Fonction de survie S(t)",
      color = "Méthode") +
  theme_minimal(base_size = 14)
```



En plus de ce critère graphique qui permet une identification visuelle de la performance, on va implémenter 2 critères numériques, la MSE et la variance en fonction du nombre de réparations parfaites m.

```
nrpm <- 100 # Nombre de réparations parfaites max
mse_BP <- c()
mse_e1 <- c()
mse_e2 <- c()
for (m in 5:nrpm){
  res<-simulPPIWeibull2(m,4,2,0.1)
  Z<-res$Z
  T<-res$T
  t<-seq(0,10,0.1*1/m)
  S_t <- SWeibull(t, 4,2)
  r<-MLE_S(T,Z)
  S_hatBP <- plot_MLE_S(t,r$x_order,r$phi_hat)
  idd<-MLE_emp_1(T,Z)
  S_hat_e1 <- plot_MLE_emp_1(t,idd)
  ipo<-MLE_emp_2_c(T,Z)
  S_hat_e2 <- plot_MLE_emp_2(t,ipo, Z)
  mse_BP <- c(mse_BP, sum((S_t - S_hatBP)**2)/length(S_t))
  mse_e1 <- c(mse_e1, sum((S_t - S_hat_e1)**2)/length(S_t))
  mse_e2 <- c(mse_e2, sum((S_t - S_hat_e2)**2)/length(S_t))
}
```

```r
df_mse <- data.frame(
  m = seq(5,nrpm),
  BP = log(mse_BP),
  e1 = log(mse_e1),
  e2 = log(mse_e2)
)

df_mse_long <- df_mse %>%
  pivot_longer(cols = -m, names_to = "Méthode", values_to = "log_MSE")

ggplot(df_mse_long, aes(x = m, y = log_MSE, color = Méthode)) +
  geom_line(size = 1.2, alpha = 0.8) +
  labs(title = "Évolution du log(MSE) selon la méthode",
       x = "Nombre de réparations parfaites",
       y = "log(MSE)",
       color = "Méthode") +
  theme_minimal(base_size = 14)
```



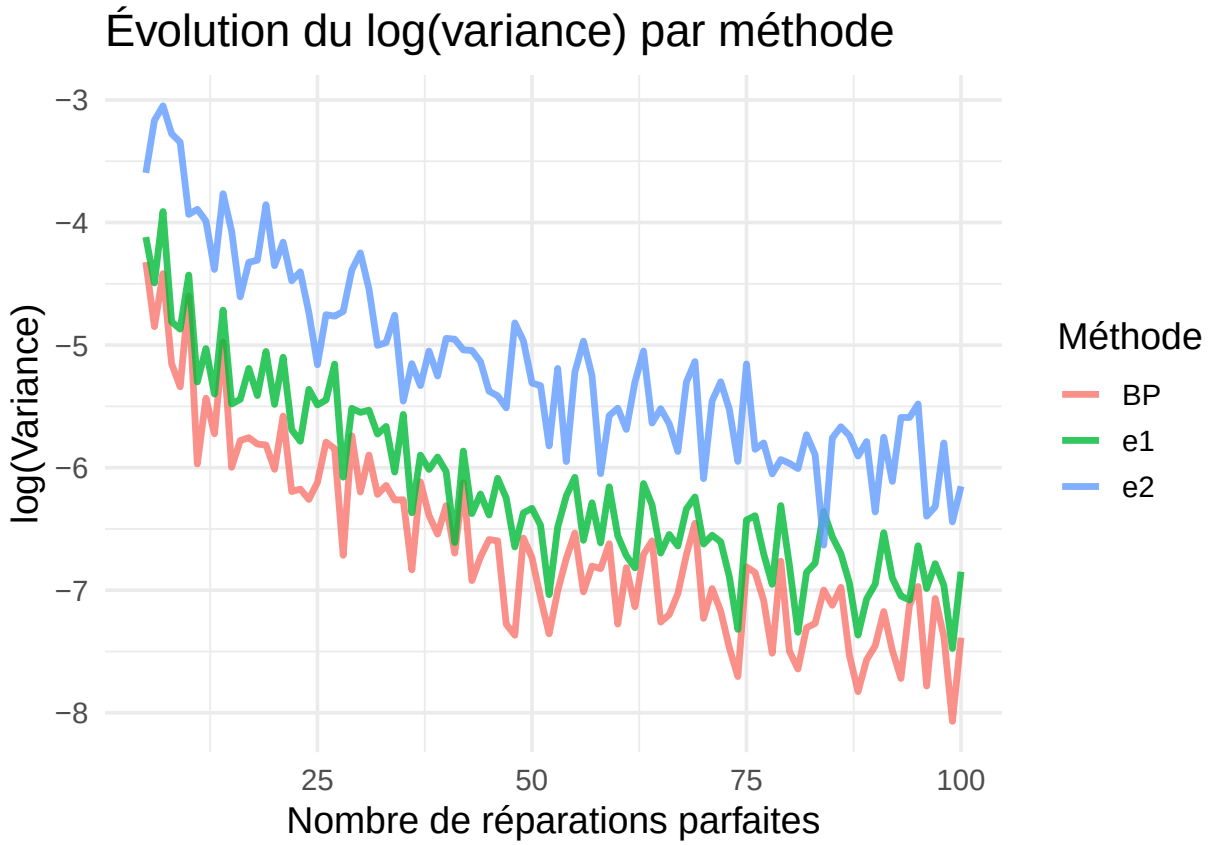
On voit que l'estimateur du MLE non paramétrique, qui est celui qui utilise le plus d'informations, mais qui ne possède pas de garanties théoriques en l'absence de $z_{n-1} = z_n = 1$, est celui qui possède tout de même les meilleures performances pour la majorité des simulations. C'est un résultat particulièrement visible lorsque les réparations parfaites sont assez rares comparées aux réparations imparfaites. On observe également que l'erreur des estimateurs a tendance à diminuer lorsqu'on augmente le nombre de réparations (via l'augmentation du nombre de réparations parfaites).

```r
nrpm <- 100 # Nombre de réparations parfaites max
niter <- 10
var_BP <- rep(0,nrpm)
var_e1 <- rep(0,nrpm)
var_e2 <- c(0,nrpm)
for (m in 5:nrpm){
  t<-seq(0,10,0.1*1/m)
  moy_BP <- rep(0,length(t))
  moy_e1 <- rep(0,length(t))
  moy_e2 <- rep(0,length(t))
  moy_BPs <- rep(0,length(t))
  moy_e1s <- rep(0,length(t))
  moy_e2s <- rep(0,length(t))
  for (i in 1:niter){
    res<-simulPPIWeibull2(m,4,2,0.2)
    Z<-res$Z
    T<-res$T
    r<-MLE_S(T,Z)
    S_hatBP <- plot_MLE_S(t,r$x_order,r$phi_hat)
    idd<-MLE_emp_1(T,Z)
    S_hat_e1 <- plot_MLE_emp_1(t,idd)
    ipo<-MLE_emp_2_c(T,Z)
    S_hat_e2 <- plot_MLE_emp_2(t,ipo, Z)
    moy_BP <- moy_BP + S_hatBP
    moy_e1 <- moy_e1 + S_hat_e1
    moy_e2 <- moy_e2 + S_hat_e2
    moy_BPs <- moy_BPs + S_hatBP**2
    moy_e1s <- moy_e1s + S_hat_e1**2
    moy_e2s <- moy_e2s + S_hat_e2**2
  }
  moy_BP <- moy_BP / niter
  moy_e1 <- moy_e1 / niter
  moy_e2 <- moy_e2 / niter
  moy_BPs <- moy_BPs / niter
  moy_e1s <- moy_e1s / niter
  moy_e2s <- moy_e2s / niter
  var_BP[m] <- sum(moy_BPs - moy_BP**2)/length(t)
  var_e1[m] <- sum(moy_e1s - moy_e1**2)/length(t)
  var_e2[m] <- sum(moy_e2s - moy_e2**2)/length(t)
}
```

Ci-dessus nous calculons la variance moyenne des fonctions de survie sur `niter` simulations en fonction du nombre de réparations parfaites.

```r
df_var <- data.frame(
  m =  seq(5,nrpm),
  BP = log(var_BP[-(1:4)]),
  e1 = log(var_e1[-(1:4)]),
  e2 = log(var_e2[-(1:4)])
)

df_var_long <- df_var %>%
  pivot_longer(cols = -m, names_to = "Méthode", values_to = "logVar")
```

```r
ggplot(df_var_long, aes(x = m, y = logVar, color = Méthode)) +
  geom_line(size = 1.2, alpha = 0.8) +
  labs(title = "Évolution du log(variance) par méthode",
       x = "Nombre de réparations parfaites",
       y = "log(Variance)",
       color = "Méthode") +
  theme_minimal(base_size = 14)
```



On remarque que le MLE non paramétrique est uniformément plus robuste que les estimateurs empiriques, sa plus faible variance indiquant sa robustesse. Ainsi, le MLE non paramétrique est à la fois plus robuste et plus consistant que les deux estimateurs empiriques. De plus, on remarque que la variance diminue avec l'augmentation du nombre de réparations (via l'augmentation du nombre de réparations parfaites).

2.5 Conclusion

2.6 Références

References

[1] Francisco J. Samaniego Lyn R. Whitaker. "Estimating the Reliability of Systems Subject to Imperfect Repair". In: *Journal of the American Statistical Association* 84.405 (1989), pp. 301–309.

[2] Frank Proschan Mark Brown. "Imperfect Repair". In: *Journal of Applied Probability* 20.4 (1983), pp. 851–859.