Projet-RM04

Baptiste Toussaint

2024-05-05

Contents

Exercice 2

```r
# Pour régler la génération aléatoire et conserver les mêmes
# valeurs à la compilation
set.seed(54684)

# Block de code pour effacer le contenu de la mémoire, utile à la compilation
rm(list = ls())
```

On a un système S à n composants iid de loi exponentielle. On a $\lambda = 0.1(jour^{-1})$ en configuration parallèle.

On peut procéder ainsi pour simuler un instant de panne de S unique pour n fixé :

1. On simule n réalisation $T_1, ..., T_n$ aléatoires de la loi exponentielle de paramètre lambda
2. On calcule $T_S = max(T_1, ..., T_n)$ (car la structure est parallèle).

Fonctions utilisées

```r
# Ce bloc de code permet d'écrire des fonctions utiles pour la suite le
# l'exercice.

# Le 'pipe' %>% permet de passer des arguments à des fonctions en R.
# On peut par exemple écrire :
#   10 %>% exp()
# Pour calculer exp(10).

exe2_simu_panne_systeme <- function(n, taux) {

    # Retourne une réalisation de panne du système pour n
    # et pour lambda = taux

    # crée n tirage de loi exponentiel avec lambda = taux, calcule le maximum
    # et retourne le résultat.
    n %>% rexp(taux) %>% max() %>% return()
}

exe2_simu_mttf_system <- function(n,taux,p) {

    # Calcul un MTTF théorique à partir de p réplications pour n composants
    # et un taux.

    # Crée p réplications de date de panne pour le système.
    # Calcule la moyenne de ces p réplications et retourne le résultat.
    replicate(p,exe2_simu_panne_systeme(n,taux)) %>% mean() %>% return()
}

# Calcul théorique du MTTF avec le résultat connu pour ce système
# spécifique.
exe2_theoric_mttf_system <- function(n, taux) {return(sum(1/1:n)/taux)}
```

Explications

La fonction `exe2_simu_panne_systeme` ci-dessus retourne une réalisation de panne du système pour n et λ.

On peut essayer pour les paramètres suivants : $n = 10$ et $\lambda = 0.1$.

```r
exe2_simu_panne_systeme(10,0.1)
```

```
## [1] 32.16055
```

Dans ce cas, le système tombe donc en panne pour 32,16 jours.

Pour rappel, pour un tel système, on a le résultat théorique suivant :

$$MTTF_n = \frac{1}{\lambda}\sum_{i=1}^{n}\frac{1}{i}$$

Si on fait l'application numérique avec $n = 10$ et $\lambda = 0.1$:

```r
exe2_theoric_mttf_system(10,0.1)
```

```
## [1] 29.28968
```

On trouve donc : $MTTF_{n=10}(\lambda = 0.1) \simeq 29.29$.

Ensuite on va chercher à évaluer l'évolution du $MTTF$ en fonction de n et comparer les simulations avec les résultats théoriques.

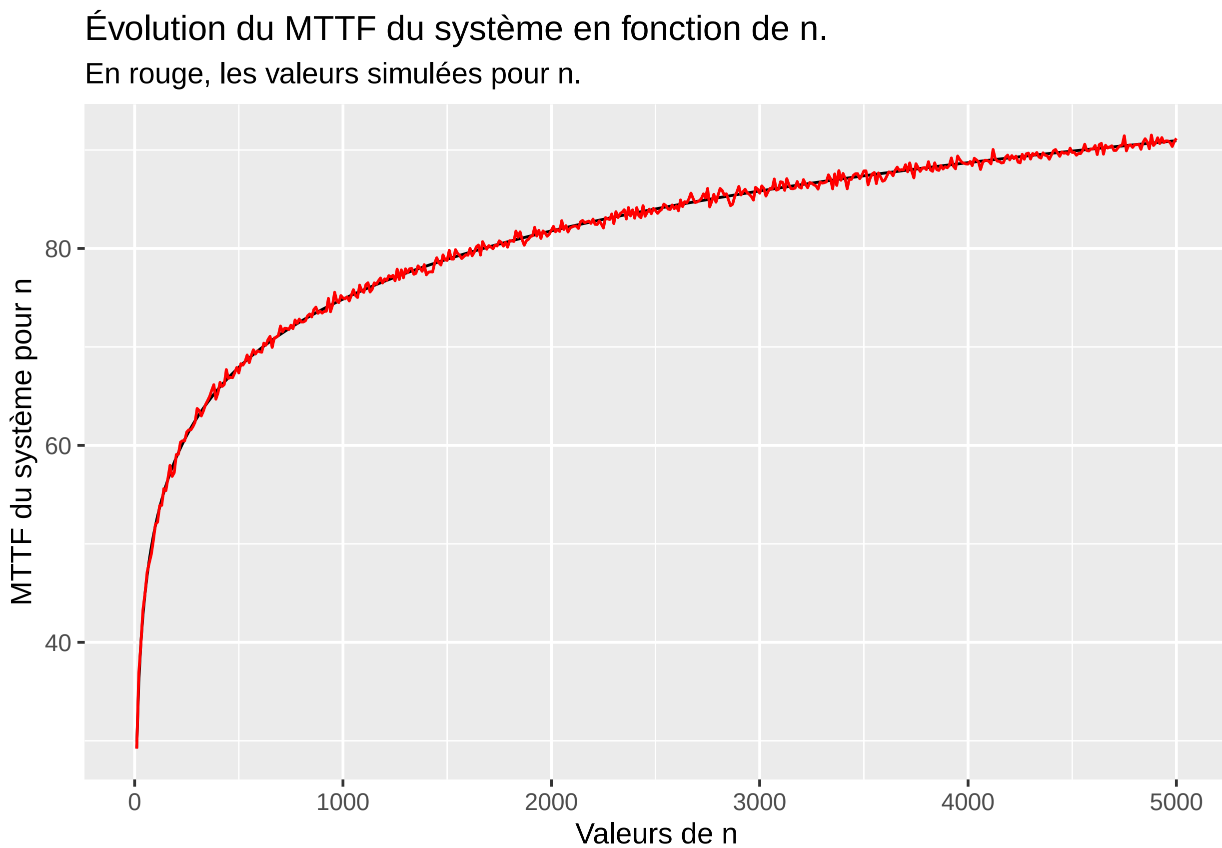
Pour trouver le $MTTF$ par simulation, il suffit de répliquer la fonction : `exe2_simu_panne_systeme` p fois (avec p relativement grand) et de prendre la moyenne des simulations réalisées, ce que fait la fonction `exe2_simu_mttf_system`.

On va donc appeler cette fonction pour plusieurs valeurs de n pour constituer une liste de $MTTF$ simulés en fonction de n.

```r
p = 1000 # nombre de répliques par simulations pour le calcul du MTTF
taux = 0.1
nvals = seq(10,5000,10)

# L'object contient les simulations de MTTF pour n de 10 à 5000
# par pas de 10
# Un peu long à la compilation

exe2_simu_MTTF_n <- nvals %>% mapply(FUN = exe2_simu_mttf_system, taux = taux, p = p)

# L'object contient les valeurs théoriques de MTTF pour n de 10 à 5000
# par pas de 10
# Un peu long à la compilation
exe2_theoric_MTTF_n <- mapply(FUN = exe2_theoric_mttf_system,
                              n = nvals, taux = 0.1)
```

On peut tracer les courbes :

```r
ggplot() +
    geom_line(aes(x = nvals, y = exe2_theoric_MTTF_n)) +
    geom_line(aes(x = nvals, y = exe2_simu_MTTF_n),
              color = "red") +
    labs(x = "Valeurs de n",
         y = "MTTF du système pour n",
         title = "Évolution du MTTF du système en fonction de n.",
         subtitle = "En rouge, les valeurs simulées pour n.")
```



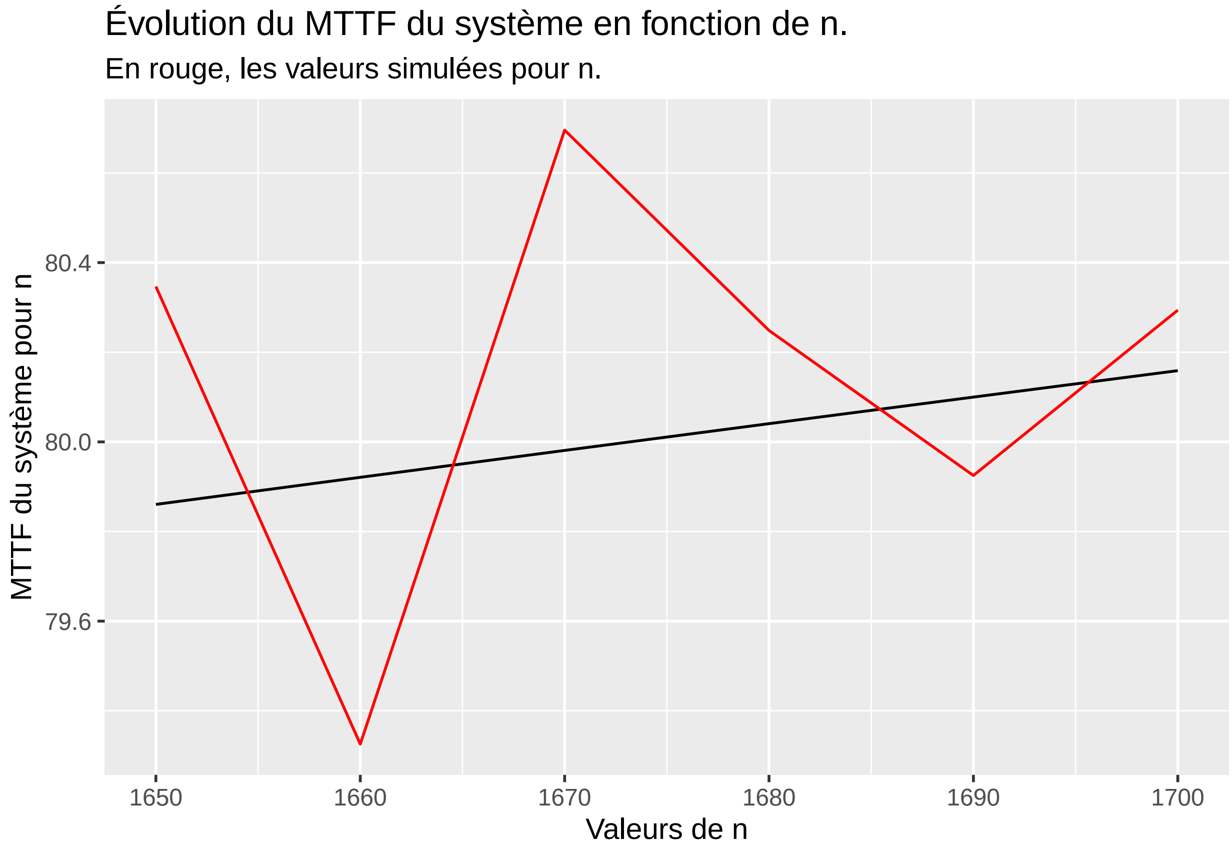
On observe donc que les valeurs simulés sont bel et bien proches des valeurs théoriques.

On remarque que le $MTTF$ croit rapidement mais il semble ensuite prendre une forme logarithmique et donc une croissance très très lente.

Graphiquement, il semble qu'l'on attend un $MTTF$ de 80 jours pour une valeur de n proche de $n = 1700$.

On peut essayer de zoomer sur le grahe pour s'en assurer :

```r
zoom = 165:170
ggplot() +
    geom_line(aes(x = nvals[zoom], y = exe2_theoric_MTTF_n[zoom])) +
    geom_line(aes(x = nvals[zoom], y = exe2_simu_MTTF_n[zoom]),
              color = "red") +
    labs(x = "Valeurs de n",
         y = "MTTF du système pour n",
         title = "Évolution du MTTF du système en fonction de n.",
         subtitle = "En rouge, les valeurs simulées pour n.")
```



On remarque qu'on atteint très précisément un $MTTF$ de 80 pour un n entre 1670 et 1675.

Celà signifie qu'il faudrait environ 1670 composants en série pour s'assurer que le système tombe en panne dans environ 80 jours.

Ce résultat peut être approché par le calcul. En effet on utilise l'approximation :

$$\sum_{i=1}^{n} \frac{1}{i} \simeq log(n) + 0.577$$

Dans ce cas :

$$MTTF_n \simeq \frac{1}{\lambda}(log(n) + 0.577)$$

On peut donc résoudre l'équation pour n^*:

$MTTF_n \simeq \frac{1}{\lambda}(log(n) + 0.577) \simeq 80$

$log(n) \simeq 80\lambda - 0.577$

$n \simeq e^{80\lambda - 0.577}$

```
exp(80*0.1-0.577)
```

```
## [1] 1674.048
```

On retrouve une très bonne approximation qui nous indique que pour avoir un système qui tombe en panne au bout de 80 jours il nous faut environ 1674 composants.

Exercice 3

Partie 1

Dans cette première partie nous allons étudier 6 systèmes différents :

- le système élémentaire simple A/B ;
- le système avec B en redondance uniquement ;
- le système en redondance haut-niveau ;
- le système en redondance bas niveau ;
- le système en redondance passive haut-niveau ;
- le système en pont.

Pour chacun de ces systèmes nous allons :

- calculer l'expression de R_{sys} et $MTTF_{sys}$;
- effectuer l'application numérique pour trouver la valeur du $MTTF_{sys}$;
- simuler une panne du système ;
- simuler p panne du système pour calculer une approximation du $MTTF_{sys}$.

```r
panne_comp_exp <- function(lambda = 0.1) {

    # Simule la date de panne d'un composant de loi exponentielle
    # et de paramètre lambda. date en annnée
    # Revient à renommer la fonctions le base de R pour plus de lisibilité dans
    # le rapport

    return(rexp(1,lambda))
}
```

Système élémentaire simple A/B

```r
# Données
lambda_A = 1 # ans-1
lambda_B = 5 # ans-1

# nombre de réplications
p = 10000

# Pour simplifier l'affichage des simulations plus loin.
message_mttf_simule_a = "MTTF simulé en années : %f"
message_mttf_simule_j = "MTTF simulé en jours : %f"
```

Dans un premier temps a un système simple avec deux composants A et B en série.

Ce cas est simple puisqu'on a :

$$MTTF_{sys} = \frac{1}{\sum_{i \in C} \lambda_i}$$

On a :

- $T_{sys_1} = min(T_A, T_B)$;
- $R_{sys_1} = e^{-(\lambda_A + \lambda_B)t}$;
- $MTTF_{sys_1} = \frac{1}{\lambda_A + \lambda_B}$;
- $[A.N] MTTF_{sys_1} = \frac{1}{6} ans^{-1} \simeq 60.83 j^{-1}$

On peut simuler ce système :

```r
exe3_sys1_panne <- function(rate_A, rate_B) {

    # Retourne la date de panne d'un système en série de deux composants A
    # Et B de loi exponentielle.

    return(min(panne_comp_exp(rate_A),panne_comp_exp(rate_B)))
}
```

Dans ce cas on peut maintenant estimer le $MTTF$ de ce système par simulation :

```r
simulation <- replicate(p,exe3_sys1_panne(lambda_A,lambda_B)) %>% mean()

# Afficher les résultats de simuation
sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.169248"
```

```r
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 61.775435"
```

Donc ce première système tombe en panne en moyenne en 61.76 jours.

Système avec B en redondance uniquement

Cette fois, on a un système série/parallèle avec A puis deux composants B en parallèles. Dans ce cas :

- $T_{sys_2} = min(T_A, max(T_B, T'_B))$
- $R_{sys_2} = e^{-\lambda_A t}(2e^{-\lambda_B t} - e^{-2\lambda_B t}) = 2e^{-(\lambda_A+\lambda_B)t} - e^{-(2\lambda_B+\lambda_A)t}$
- $MTTF_{sys_2} = \frac{2}{\lambda_A+\lambda_B} - \frac{1}{2\lambda_B+\lambda_A}$
- $[A.N] MTTF_{sys_2} \simeq 88.48j$

Si l'on fait l'application numérique on a en théorie :

```r
# MTTF théorique en jours
((2/(5+1))-(1/(2*5+1)))*365
```

```
## [1] 88.48485
```

Le système tombe en moyenne en panne après 88.48 jours.

Par la simulation :

```r
exe3_sys2_panne <- function(rate_A,rate_B) {
    min(panne_comp_exp(rate_A),
        max(panne_comp_exp(rate_B),panne_comp_exp(rate_B))) %>%
        return()
}
```

```r
simulation <- replicate(p,exe3_sys2_panne(lambda_A,lambda_B)) %>% mean()

sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.244062"
```

```r
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 89.082753"
```

On retrouve bien un résultat proche de la valeur théorique.

Système en redondance haut-niveau

Dans ce cas on a un système parallèle/série avec deux séries A-B en paralèlles :

- $T_{sys_3} = max(min(T_A, T_B), min(T'_A, T'_B))$
- $R_{sys_3} = 2e^{-(\lambda_A+\lambda_B)t} - e^{-2(\lambda_A+\lambda_B)t}$
- $MTTF_{sys_3} = \frac{2}{\lambda_A+\lambda_B} - \frac{1}{2(\lambda_A+\lambda_B)}$
- $[A.N] MTTF_{sys_3} = \frac{2}{3} - \frac{1}{12} = 0.25 ans^{-1} = 91.25 j^{-1}$

Dans ce cas on a un $MTTF$ théorique de :

```
(2/(lambda_A+lambda_B))-(1/(2*(lambda_A+lambda_B))) # En ans
```

```
## [1] 0.25
```

```
((2/(lambda_A+lambda_B))-(1/(2*(lambda_A+lambda_B))))*365 # En jours
```

```
## [1] 91.25
```

Par la simulation on trouve :

```
exe3_sys3_panne <- function(rate_A,rate_B) {
    max(
        min(panne_comp_exp(rate_A),panne_comp_exp(rate_B)),
        min(panne_comp_exp(rate_A),panne_comp_exp(rate_B))
    ) %>%
        return()
}
```

```
simulation <- replicate(p,exe3_sys3_panne(lambda_A,lambda_B)) %>% mean()

sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.250382"
```

```
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 91.389605"
```

Système en redondance bas niveau

Dans ce cas on a un système parallèle/série avec :

- $T_{sys_4} = min(max(T_A, T'_A), max(T_B, T'_B))$
- $$\begin{aligned} R_{sys_4}(t) &= (1-(1-R_A(t))^2)(1-(1-R_B(t))^2) \\ &= (R_A(t)^2 - 2R_A(t))(R_B(t)^2 - 2R_B(t)) \\ &= e^{-2(\lambda_A+\lambda_B)t} - 2e^{-(2\lambda_A+\lambda_B)t} - 2e^{-(\lambda_A+2\lambda_B)t} + 4e^{-(\lambda_A+\lambda_B)t} \end{aligned}$$
- $$MTTF_{sys_4} = \frac{4}{\lambda_A+\lambda_B} - \frac{2}{\lambda_A+2\lambda_B} - \frac{2}{2\lambda_A+\lambda_B} + \frac{1}{2(\lambda_A+\lambda_B)}$$
- $$\begin{aligned} [A.N] MTTF_{sys_4} &= \frac{4}{6} - \frac{2}{11} - \frac{2}{7} + \frac{1}{12} \\ &\simeq 0.2824 ans \\ &\simeq 103.10 j \end{aligned}$$

```
(4/(lambda_A+lambda_B))-(2/(lambda_A+2*lambda_B))-(2/(2*lambda_A+lambda_B))+
    (1/(2*(lambda_A+lambda_B))) # En ans
```

```
## [1] 0.2824675
```

```r
((4/(lambda_A+lambda_B))-(2/(lambda_A+2*lambda_B))-(2/(2*lambda_A+lambda_B))+
    (1/(2*(lambda_A+lambda_B)))
)*365 # En jours
```

```
## [1] 103.1006
```

Par la simulation on trouve :

```r
exe3_sys4_panne <- function(rate_A,rate_B) {
    min(
        max(panne_comp_exp(rate_A),panne_comp_exp(rate_A)),
        max(panne_comp_exp(rate_B),panne_comp_exp(rate_B))
    ) %>%
        return()
}
```

```r
simulation <- replicate(p,exe3_sys4_panne(lambda_A,lambda_B)) %>% mean()

sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.280923"
```

```r
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 102.536971"
```

Système en redondance passive haut-niveau

Ce système peut être assimilé à deux système élémentaires A/B (système n°1) mit en redondance passive.

Dans ce cas on a :

- $T_{sys_5} = min(T_A, T_B) + min(T'_A, T'_B)$

Le calcul de R_{sys_5} est plus complexe puisqu'il faut conditionner selon la valeur de panne du premier système.

On pose : $T_{sys_5} = X + Y$ avec $X = min(T_A, T_B)$ et Y = min(T_A',T_B')$. Attention, dans ce cas X et Y ont la même loi, identique à celle du système n°1 vu plus haut. Donc : $R_X = R_Y = R_{sys_1} = e^{-(\lambda_A+\lambda_B)t}$.

On a alors :

$$
\begin{aligned}
P(T_{sys_5} \geq t) = P(X + Y \geq t)& \\
&= P(X \geq t)P(Y \geq t - X | X \geq t) + P(X \leq t)P(Y \leq t - X | X \leq t) \\
&Ona : P(Y \geq t - X | X \geq t) car X \geq t \Rightarrow t - X \leq 0 et P(Y \geq 0) = 1 car \Omega_Y = \mathbb{R}^+ \\
&= R_X(t) + \int_{u \in \Omega_X, u \leq t} R_Y(t-u) f_X(u) du \\
&= e^{-(\lambda_A+\lambda_B)t} + \int_0^t e^{-(\lambda_A+\lambda_B)(t-u)} (\lambda_A + \lambda_B) e^{-(\lambda_A+\lambda_B)u} du \\
&= e^{-(\lambda_A+\lambda_B)t} + \int_0^t (\lambda_A + \lambda_B) e^{-(\lambda_A+\lambda_B)u} du \\
&= e^{-(\lambda_A+\lambda_B)} + (\lambda_A + \lambda_B) e^{-(\lambda_A+\lambda_B)t} t \\
&= e^{-(\lambda_A+\lambda_B)} (1 + (\lambda_A + \lambda_B) t)
\end{aligned}
$$

On peut calculer le $MTTF_{sys_5}$ (On pose $\psi = \lambda_A + \lambda_B$):

$$
\begin{aligned}
MTTF_{sys_5} &= \int_0^{+\infty} e^{-\psi t}(1+\psi t)dt \\
&= \left[(1+\psi t)(-\frac{1}{\psi}e^{-\psi t})\right]_0^{\infty} - \int_0^{\infty} \psi(-\frac{1}{\psi}e^{-\psi t})dt \\
&= -\frac{1}{\psi}\left[e^{-\psi t} + \psi e^{-\psi t} t\right]_0^{\infty} + \int_0^{\infty} e^{-\psi t}dt \\
&= \frac{1}{\psi} - \left[-\frac{1}{\psi}e^{-\psi t}\right]_0^{\infty} \\
&= \frac{1}{\psi} - \frac{1}{\psi}(0-1) \\
&= \frac{2}{\psi} = \frac{2}{\lambda_1+\lambda_B} = 2MTTF_{sys_1}.
\end{aligned}
$$

On a alors :

$[A.N]MTTF_{sys_5} = \quad = \frac{2}{6} = \frac{1}{3}ans^{\simeq}121.67j$

Par la simulation on trouve :

```r
exe3_sys5_panne <- function(rate_A,rate_B) {
    min(panne_comp_exp(rate_A),panne_comp_exp(rate_B)) +
        min(panne_comp_exp(rate_A),panne_comp_exp(rate_B)) %>%
        return()
}
```

```r
simulation <- replicate(p,exe3_sys5_panne(lambda_A,lambda_B)) %>% mean()

sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.336252"
```

```r
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 122.731821"
```

Système en pont avec $\lambda_C = \lambda_B$

Dans ce système en pont, on trouve au centre un composant C de loi identique à B donc $\lambda_C = \lambda_B$.

Système pont

Dans ce cas plusieurs options s'offrent à nous.

Pour calculer la valeur de R_{sys_6} il faut passer par un conditionnement selon la valeur du composant C

- Si C est en marche alors on a $R_{sys_6}(t) = R_{sys_4}(t)$
- Si C est panne alors on a $R_{sys_6}(t) = R_{sys_4}(t)$

On a alors :

$$
\begin{aligned}
R_{sys_6}(t) &= P(T_{sys_6} \geq t) = P(T_C \geq t)P(T_{sys_4} \geq t) + P(T_C \leq t)P(T_{sys_3} \geq t) \\
&= R_C(t)R_{sys_4}(t) + (1-R_C(t))R_{sys_3}(t) \\
&= 2e^{-(\lambda_A+2\lambda_B)t} + 2e^{-(2\lambda_A+3\lambda_B)t} + 2e^{-(\lambda_A+\lambda_B)t} - 2e^{-(\lambda_A+3\lambda_B)t} - 3e^{-(\lambda_A+\lambda_B)t}
\end{aligned}
$$

Dans ce cas on a :

$$
MTTF_{sys_6} = \frac{2}{\lambda_A+2\lambda_B} + \frac{2}{2\lambda_A+3\lambda_B} + \frac{2}{\lambda_A+\lambda_B} - \frac{2}{\lambda_A+3\lambda_B} - \frac{3}{2(\lambda_A+\lambda_B)}
$$

Soit :

$$[A.N] MTTF_{sys_6} = \frac{2}{11} + \frac{2}{17} + \frac{2}{6} - \frac{2}{16} - \frac{3}{12}$$
$$= \simeq 0.2578 ans \simeq 94.10 jours$$

On remarque alors que :

$$MTTF_{sys_5} > MTTF_{sys_4} > MTTF_{sys_6} > MTTF_{sys_3} > MTTF_{sys_2} > MTTF_{sys_1}$$

Si l'on utilise la simulation pour retrouver ce résultat on a alors :

$$T_{sys_6} = max(min(T_A, T_B), min(T_A, T'_B, T_C), min(T'_A, T'_B), min(T'_A, T_B, T_C))$$

```
exe3_sys6_panne <- function(rate_A,rate_B) {

A1 <- panne_comp_exp(rate_A)
A2 <- panne_comp_exp(rate_A)
B1 <- panne_comp_exp(rate_B)
B2 <- panne_comp_exp(rate_B)
C <- panne_comp_exp(rate_B)

  max(
      min(A1,B1),
      min(A1,B2,C),
      min(A2,B2),
      min(A2,B1,C)
  ) %>% return()

}
```

```
simulation <- replicate(p,exe3_sys6_panne(lambda_A,lambda_B)) %>% mean()

sprintf(message_mttf_simule_a, simulation)
```

```
## [1] "MTTF simulé en années : 0.261487"
```

```
sprintf(message_mttf_simule_j, simulation * 365)
```

```
## [1] "MTTF simulé en jours : 95.442743"
```

Partie 2

Dans le cas de ce dernier système en redondance bas niveau, les composants A suivent une loi de Weibull $W(\theta = 100, \beta = 1.5)$ et les composants B suivent un lois normale : $N(\mu = 250, \sigma = 20)$.

Dans ce cas, en reprenant le résultat du système en redondance bas-niveau, on a :

$$R_{sys_7} = (2R_A(t) - R_A(t)^2)(2R_B(t) - R_B(t)^2)$$

En plus d'être pénible, le calcul de la fonction de survie R_B fait intervenir la fonction d'erreur de Gauss $erf(x)$ car la fonction de répartition de la loi normale n'est pas clairement définie.

Il n'est donc pas possible d'obtenir une $MTTF$ théorique par le calcul et l'on passera alors par la simulation.

On a :

$$T_{sys_7} = min(max(T_A, T'_A), max(T_B, T'_B))$$

```r
panne_comp_norm <- function(mu = 0, sigma = 1) {
    return(rnorm(1, mean = mu, sd = sigma))
}

panne_comp_weibull <- function(theta = 0.1, beta = 1) {
    return(rweibull(1, scale = theta, shape = beta))
}

exe3_sys6_panne <- function(param = list()) {

  min(
    max(panne_comp_weibull(param$theta,param$beta),
        panne_comp_weibull(param$theta,param$beta)),
    max(panne_comp_norm(param$mu,param$sigma),
        panne_comp_norm(param$mu,param$sigma))
  ) %>% return()

}
```

On peut alors simuler le $MTTF$:

```r
simulation <- replicate(
  p,exe3_sys6_panne(list(theta = 100, beta = 1.5, mu = 250, sigma = 20))) %>% mean()

sprintf(message_mttf_simule_j, simulation)
```

```
## [1] "MTTF simulé en jours : 122.632615"
```

On a un $MTTF$ égal à 44760 jours.

Exercice 4, Partie 1 : Données simulées

Les données pour cet exercice sont stockées dans le fichier `RM04data.csv`. Commençons par charger ces données.

```
# La fonction read_csv2 est issue du package readr qui permet des imports
# exports facilités dans R
df <- read.csv("../data/RM04data.csv", sep = ',', header = FALSE)
df <- df[1:4,]

columns_names = c("xi_sys1", "xi_sys2", "xi_sys3", "xi_sys4")
observations_labels <- paste('x',as.character(1:200),sep = '')

rownames(df) <- columns_names
colnames(df) <- observations_labels

df <- t(df)

head(df)
```

```
##      xi_sys1  xi_sys2   xi_sys3   xi_sys4
## x1 2.9304179 4.677403 2.2591062 2.9184215
## x2 6.4750017 7.326121 4.9916828 4.2306659
## x3 1.3918117 3.717424 1.0729700 0.2696511
## x4 4.1970742 4.407853 3.2355919 2.8474980
## x5 6.9849789 3.184897 3.6016684 0.0614171
## x6 0.4195876 3.162764 0.3234668 0.6936315
```

Pour vérifier :

```
dim(df)
```

```
## [1] 200   4
```

On a donc un jeu de données à 4 échantillons de 200 éléments (les temps d'observation inter-pannes x_i).

L'objectif de cet exercice est d'identifier les lois les plus adaptées à chaque échantillon.

On peut commencer à travailler les données en calculant les T_i : les dates de panne observées.

```
df <- cbind(df,cumsum(df[,1]),cumsum(df[,2]),cumsum(df[,3]),cumsum(df[,4]))
columns_names <- c(columns_names,
                   "ti_sys1", "ti_sys2", "ti_sys3", "ti_sys4")

colnames(df) <- columns_names

head(df)
```

```
##      xi_sys1  xi_sys2   xi_sys3   xi_sys4   ti_sys1   ti_sys2   ti_sys3
## x1 2.9304179 4.677403 2.2591062 2.9184215  2.930418  4.677403  2.259106
## x2 6.4750017 7.326121 4.9916828 4.2306659  9.405420 12.003525  7.250789
## x3 1.3918117 3.717424 1.0729700 0.2696511 10.797231 15.720948  8.323759
## x4 4.1970742 4.407853 3.2355919 2.8474980 14.994306 20.128801 11.559351
## x5 6.9849789 3.184897 3.6016684 0.0614171 21.979284 23.313698 15.161019
## x6 0.4195876 3.162764 0.3234668 0.6936315 22.398872 26.476462 15.484486
##      ti_sys4
## x1  2.918421
## x2  7.149087
## x3  7.418738
```

```
## x4 10.266236
## x5 10.327654
## x6 11.021285
```

On obtient les T_i en faisant la somme cumulée des x_i ce qui donne :

$$\forall i \in [1, n] : T_i = \sum_{k=1}^{i} x_k$$

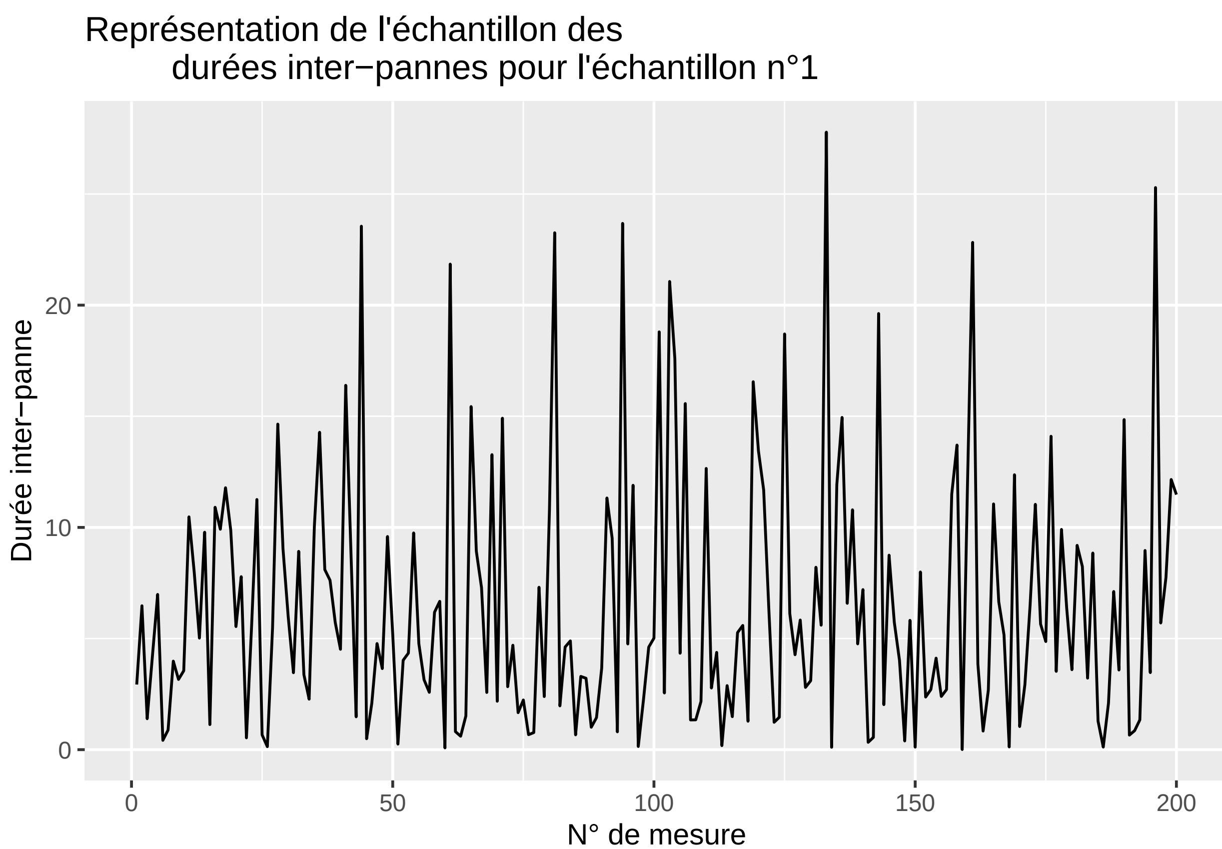
> Attention : durant cet exercice, je n'utilise pas les estimateurs non biaisés.

Échantillon n°1

On peut commencer par tracer les données :

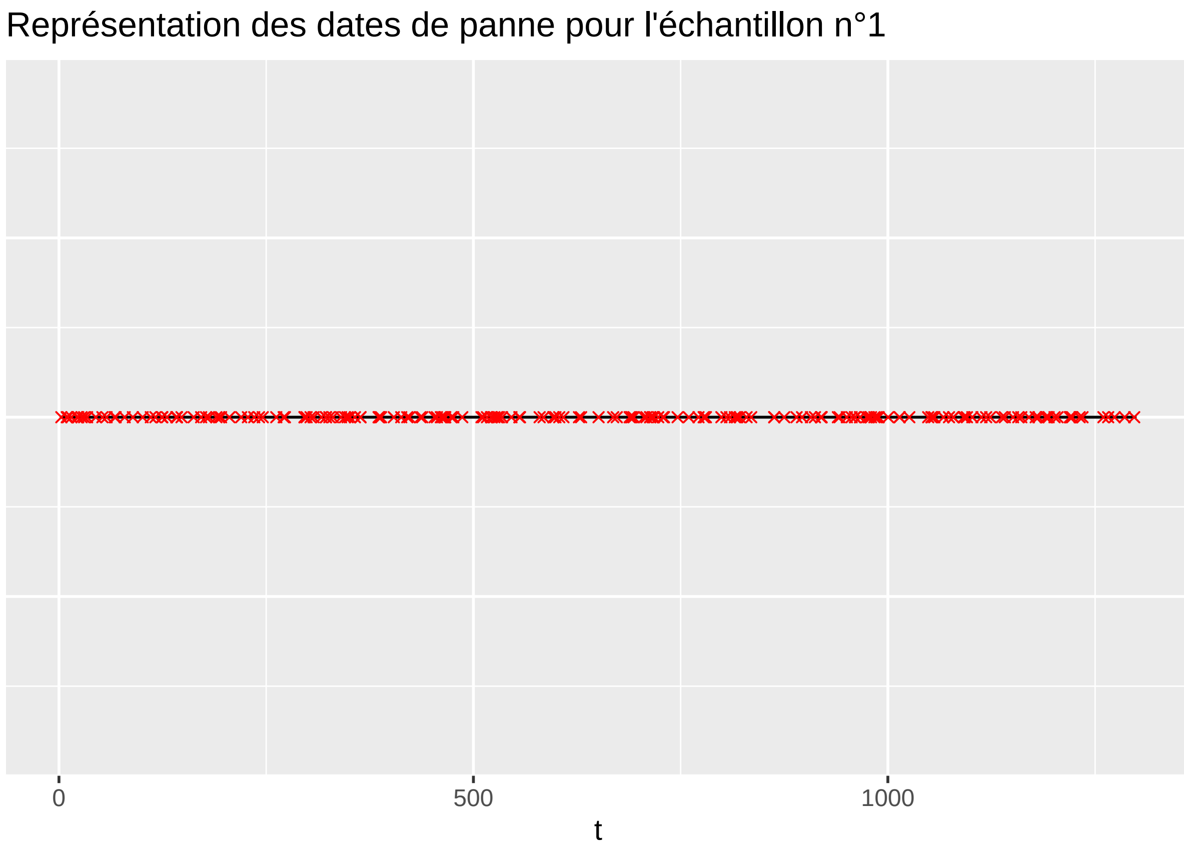
- l'évolution des durées inter-pannes,
- les dates de panne.

```
ggplot() +
    geom_line(aes(x = 1:200, y = df[,"xi_sys1"])) +
    labs(x = "N° de mesure",
         y = "Durée inter-panne",
         title = "Représentation de l'échantillon des
         durées inter-pannes pour l'échantillon n°1")
```




```
ggplot() +
    geom_line(aes(x = 1:1300, y = 1)) +
    geom_point(aes(x = df[,"ti_sys1"], y = 1)
```

```
              , color = "red",
              shape = 4) +
  labs(x = "t",
       y = "",
       title = "Représentation des dates de panne pour l'échantillon n°1") +
  guides(y = "none")
```



D'après les deux graphiques précédent, on peut supposer que la variable aléatoire dont est issu l'échantillon n°1 ne présente pas de tendance : il semble que les dates de pannes (et donc les durées inter-pannes) sont uniformément distribuées.

Tendance

Si les X_i (ou T_i) sont *iid*, l'échantillon ne possède aucune tendance. On va donc tester l'hypothèse selon laquelle notre échantillon est *iid* pour déterminer ou non la présence de tendance.

Nous disposons de deux outils mathématique :

- le test de Laplace,
- le test de Spearman.

Je propose d'utiliser le test de Laplace dans un premier temps.

Dans le cadre de la sûreté de fonctionnement il se présente ainsi :

$$\begin{cases} H_0 : \text{les données sont issues d'un processus de poisson homogène} \\ H_1 : \text{les données ne sont pas issues d'un processus de poisson homogène} \end{cases}$$

Si l'on établit que les données sont issues d'un processus de Poisson Homogène (PPH), alors par définition, on établit que les données ne possèdent pas de tendance.

On a la statistique de test suivante :

$$U = \sqrt{\frac{12}{(n-1)T_n^2}}\left(\sum_{i=1}^{n} T_i - (n+1)\frac{T_n}{2}\right)$$

que l'on compare à au quantile de la loi normale centrée réduite.

$$\begin{cases} |U| > F^{-1}_{N_{(0,1)}}(1-\frac{\alpha}{2}), \text{ alors on rejette } H_0 \\ |U| < F^{-1}_{N_{(0,1)}}(1-\frac{\alpha}{2}), \text{ on ne rejette pas } H_0 \end{cases}$$

Si on conserve H_0 on en conclue que l'échantillon est issus d'un PPH, et donc qu'il n'y a pas de tendance.

On peut écrire une fonction qui calcule la statistique de test en fonction des T_i:

```
compute_laplace_stat <- function(ti){

  # Prend le tableau des Ti en entrée de fonction et retourne U

  n <- length(ti)
  U <- sqrt((12)/((n-1)*ti[n]^2))*(sum(ti)-(n+1)*ti[n]/2)

  U %>% as.numeric %>% return()

}
```

On calcul la statistique de test U :

```
compute_laplace_stat(df[,"ti_sys1"])
```

```
## [1] -0.5647952
```

Au seuil $\alpha = 0.05$ on va comparer la statistique de test avec le quantile de la loi normale centrée réduite de 1.96.

```
alpha = 0.05 # risque
n = 200 # nombre d'observartion
qnorm(1-alpha/2)
```

```
## [1] 1.959964
```

On a bel et bien $|U| < 1.96$, on ne rejette donc pas H_0 et on peut conclure que la série de mesure ne possède pas de tendance.

Adéquation à la loi exponentielle

Le test précédent nous confirme qu'il n'y a pas de tendance. On peut alors tester l'adéquation des données avec une loi exponentielle. On peut utiliser un test de Bartlett que l'on doit programmer. Le test se présente ainsi :

$$\begin{cases} H_0 : \text{les données sont issues d'un processus de poisson homogène} \\ H_1 : \text{les données ne sont pas issues d'un processus de poisson homogène} \end{cases}$$

Avec la statistique de test :

$$B = \frac{2n\left(ln\left(\frac{T_n}{n}\right) - \frac{\sum_{i=1}^{n} ln(X_i)}{n}\right)}{1+\frac{n+1}{6n}}$$

que l'on compare au quantile de la loi du χ^2_{n-1}

$$\begin{cases} B \in \left[F^{-1}_{\chi^2_{n-1}}(\frac{\alpha}{2}); F^{-1}_{\chi^2_{n-1}}(1-\frac{\alpha}{2})\right], \text{ on ne rejette pas } H_0 \\ B \notin \left[F^{-1}_{\chi^2_{n-1}}(\frac{\alpha}{2}); F^{-1}_{\chi^2_{n-1}}(1-\frac{\alpha}{2})\right], \text{ alors on rejette } H_0 \end{cases}$$

```
compute_bartlett_stat <- function(tn, xi) {

  # Fonction qui calcule la statistique de test du test de Bartlett

  n <- length(xi)

  2*n*(log(tn/n)-sum(log(xi))/n)/(1+((n+1)/(6*n))) %>% return()
}
```

```
# on calcule la statistique de test
compute_bartlett_stat(df[n,"ti_sys1"], df[,"xi_sys1"])
```

```
## [1] 183.6263
```

On cherche alors les quantiles de la loi du χ^2 :

```
qchisq(p = 0.025, n-1)
```

```
## [1] 161.8262
```

```
qchisq(p = 0.975, n-1)
```

```
## [1] 239.9597
```

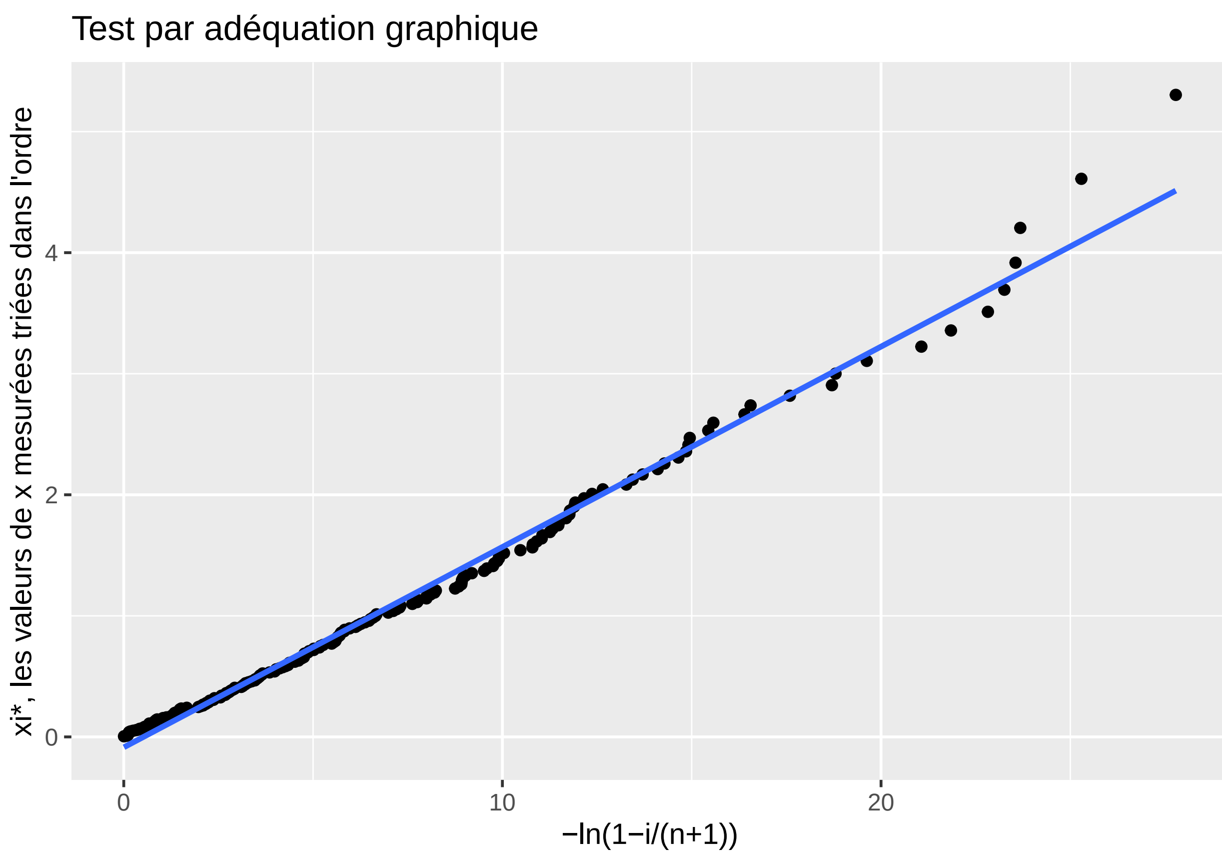
On a : $161.83 < 183.63 < 239.96$. Dans ce cas on ne va pas rejeter H_0 et considérer que l'échantillon est issu d'un processus de poisson homogène (PPH).

Les observations sont donc issus d'une variable aléatoire exponentielle.

On peut compléter l'analyse avec une adéquation graphique.

```
nuage = tibble(-log(1-(1:200)/(n+1)),sort(df[,"xi_sys1"]))
colnames(nuage) <- c('formula','x')

ggplot() +
    geom_point(aes(x = nuage$`x`,y = nuage$formula)) +
    geom_smooth(aes(x = nuage$`x`,y = nuage$formula),
                se = FALSE,
                method = lm) +
    labs(y = "xi*, les valeurs de x mesurées triées dans l'ordre",
         x = "-ln(1-i/(n+1))",
         title = "Test par adéquation graphique")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



La fonction geom_smooth du package ggplot2 permet d'ajuster une droite de régression aux données affichées.

On peut utiliser la fonction lm pour obtenir les informations sur la droite de régression.

```
lm(nuage$formula~nuage$x) %>% summary()
```

```
##
## Call:
## lm(formula = nuage$formula ~ nuage$x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17956 -0.04031 -0.01285  0.04897  0.79090
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.086328   0.010183  -8.478 5.24e-15 ***
## nuage$x      0.165511   0.001177 140.603  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09529 on 198 degrees of freedom
## Multiple R-squared:  0.9901, Adjusted R-squared:   0.99
## F-statistic: 1.977e+04 on 1 and 198 DF,  p-value: < 2.2e-16
```

La valeur de la pente de la droite de régression estimée par la fonction lm est de 0.165511, il s'agit d'une estimation de λ que l'on notera $\hat{\lambda}^{sys1}_{gph}$

On a un $R^2 = 0.99$ ce qui permet de renforcer la conclusion selon laquelle l'échantillon est issue d'une loi exponentielle.

Estimation des paramètres

On peut ensuite chercher à estimer les paramètres

```
estim_param_sys1_mm = 1/mean(df[,"xi_sys1"])
estim_param_sys1_mm
```

```
## [1] 0.1541832
```

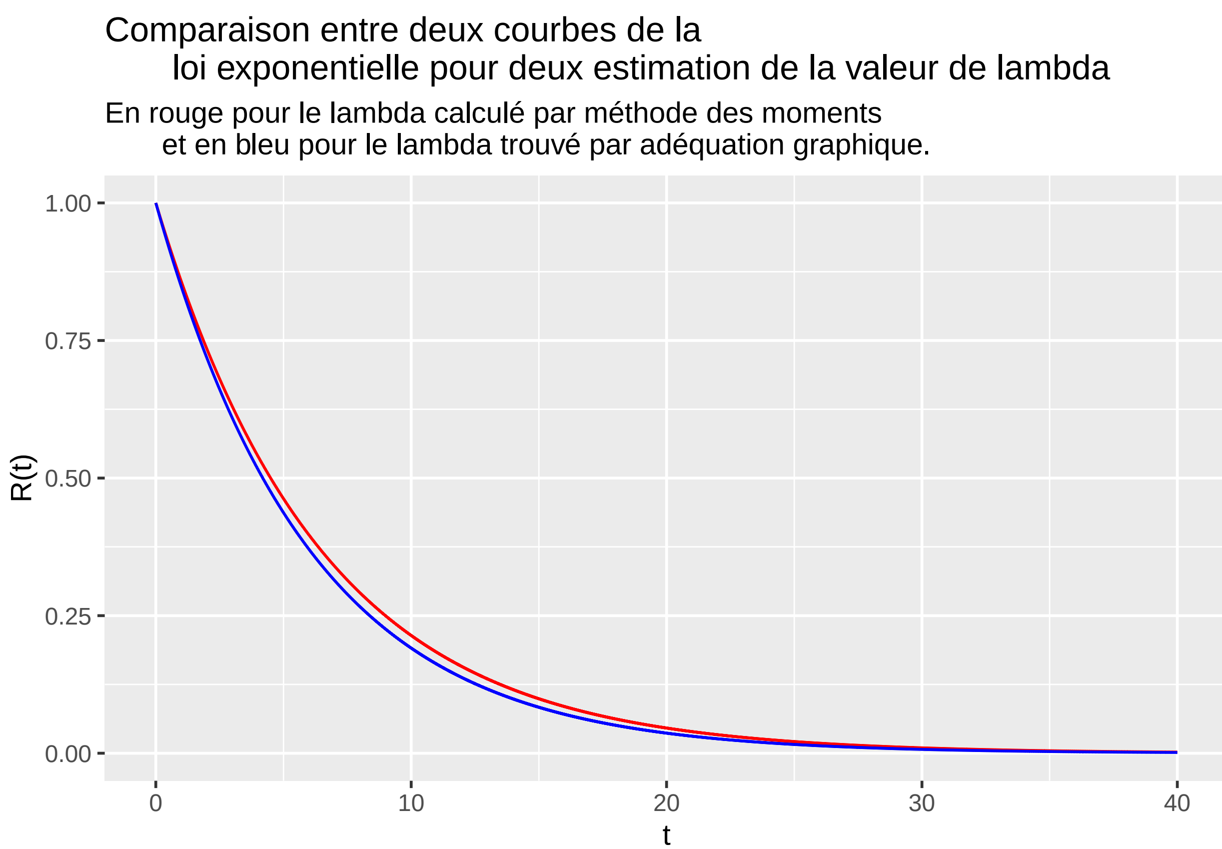
On trouve donc l'estimateur par méthode des moments : $\hat{\lambda}_{mm}^{sys_1} \simeq 0.1541832$.

Nos deux estimateurs : $\hat{\lambda}_{mm}^{sys_1} \simeq 0.1541832$ et $\hat{\lambda}_{gph}^{sys1} \simeq 0.165511$ sont très proches quoi que légèrement différents.

Représentation graphique des deux modèles

On peut tracer les fonctions de survie des deux modèles trouvés.

```r
estim_param_sys1_gph = 0.165511
t = seq(0,40,0.001)

ggplot() +
  geom_line(aes(t, 1-pexp(t,estim_param_sys1_mm)), color = 'red') +
  geom_line(aes(t, 1-pexp(t,estim_param_sys1_gph)), color = 'blue') +
  xlim(0,40) +
  ylim(0,1) +
  labs(x = "t",
       y = "R(t)",
       title = "Comparaison entre deux courbes de la
       loi exponentielle pour deux estimation de la valeur de lambda",
       subtitle = "En rouge pour le lambda calculé par méthode des moments
       et en bleu pour le lambda trouvé par adéquation graphique.")
```



Conclusion

On peut donc conclure que l'échantillon n°1 est issue d'un système suivant une loi exponentielle de paramètre $\lambda \in [0.1541832; 0.165511]$ donc de moyenne (nous de disposons pas des unités) :

```r
1/estim_param_sys1_mm
```

```
## [1] 6.48579
```

```r
1/estim_param_sys1_gph
```

```
## [1] 6.041894
```

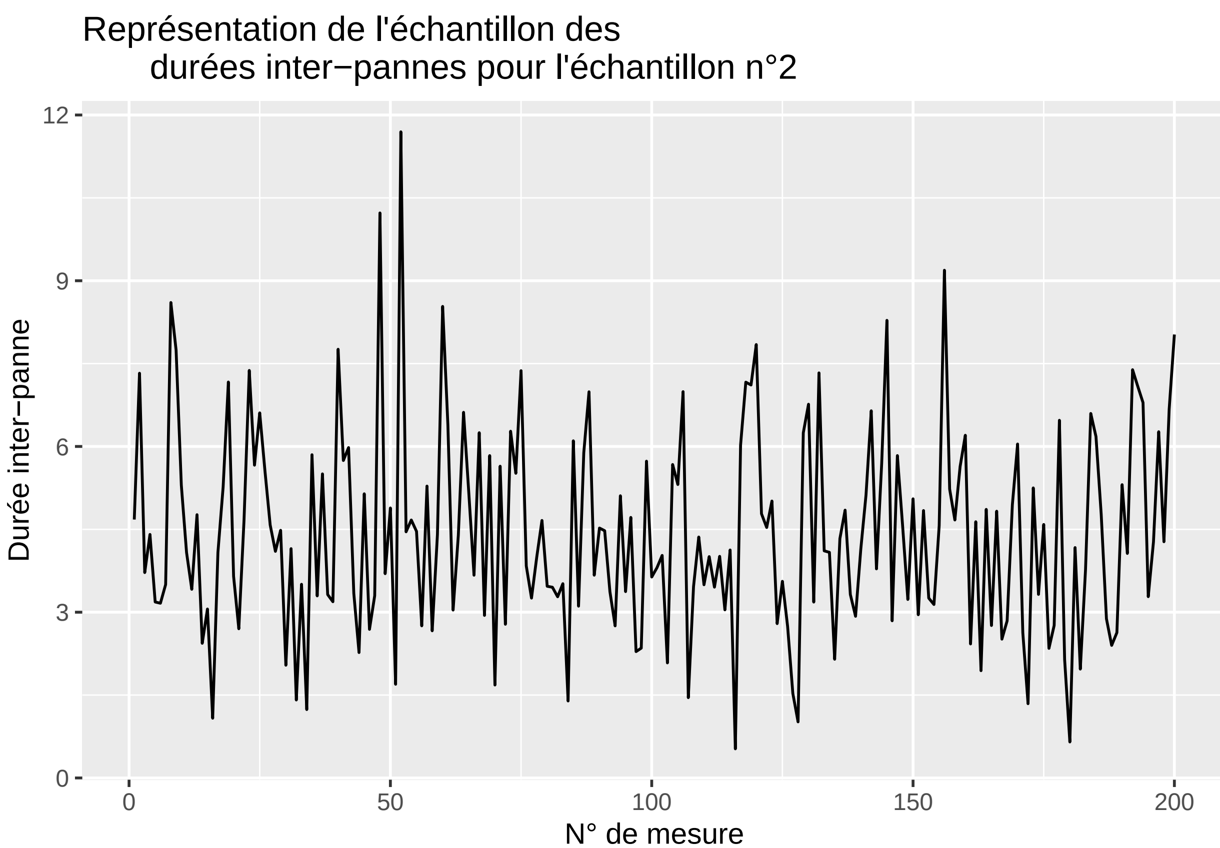
Donc $MTTF_{sys_1} \in [6.041894; 6.485791]$ (attention, il ne s'agit pas ici d'un intervalle de confiance).

Sur un cas réel, il faudrait trancher entre ces deux estimateurs.

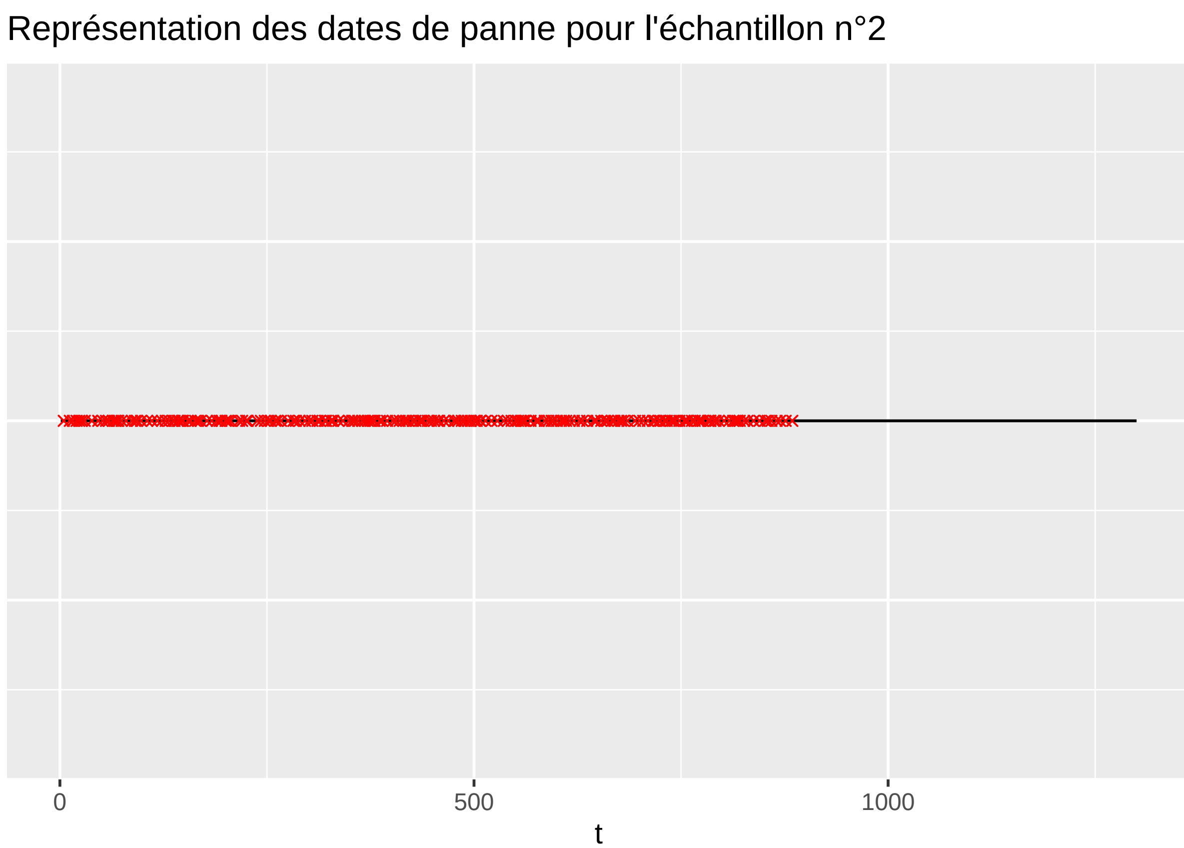
Échantillon n°2

On commence par tracer l'échantillon :

```r
ggplot() +
  geom_line(aes(x = 1:200, y = df[,"xi_sys2"])) +
  labs(x = "N° de mesure",
       y = "Durée inter-panne",
       title = "Représentation de l'échantillon des
       durées inter-pannes pour l'échantillon n°2")
```




```r
ggplot() +
  geom_line(aes(x = 1:1300, y = 1)) +
  geom_point(aes(x = df[,"ti_sys2"], y = 1)
             , color = "red",
             shape = 4) +
  labs(x = "t",
       y = "",
       title = "Représentation des dates de panne pour l'échantillon n°2") +
  guides(y = "none")
```



On peut encore supposer une absence de tendance au vu de la répartition des pannes.

Tendance

Pour observer la tendance on peut utiliser le test de Laplace.

```
compute_laplace_stat(df[,"ti_sys2"])
```

```
## [1] 0.1656895
```

Là encore, on est en dessous de 1.96, on peut conclure à l'absence de tendance.

Adéquation à la loi exponentielle

On réalise un test de Bartlett :

```
compute_bartlett_stat(df[200,"ti_sys2"], df[,"xi_sys2"])
```

```
## [1] 33.92903
```

Dans ce cas on va très clairement rejeter H_0 car nous ne nous trouvons pas entre les quantiles du χ^2 :

```
qchisq(p = 0.025, n-1)
```

```
## [1] 161.8262
```

```
qchisq(p = 0.975, n-1)
```

```
## [1] 239.9597
```

Donc la loi exponentielle ne semble pas adéquate à cet échantillon.

Adéquation à la loi de Weibull par maintenance parfaite

On peut alors tester l'hypothèse selon laquelle la variable aléatoire dont l'échantillon est issue suit une loi de Weibull avec réparation parfaite.

On peut vérifier cette hypothèse en par adéquation graphique.

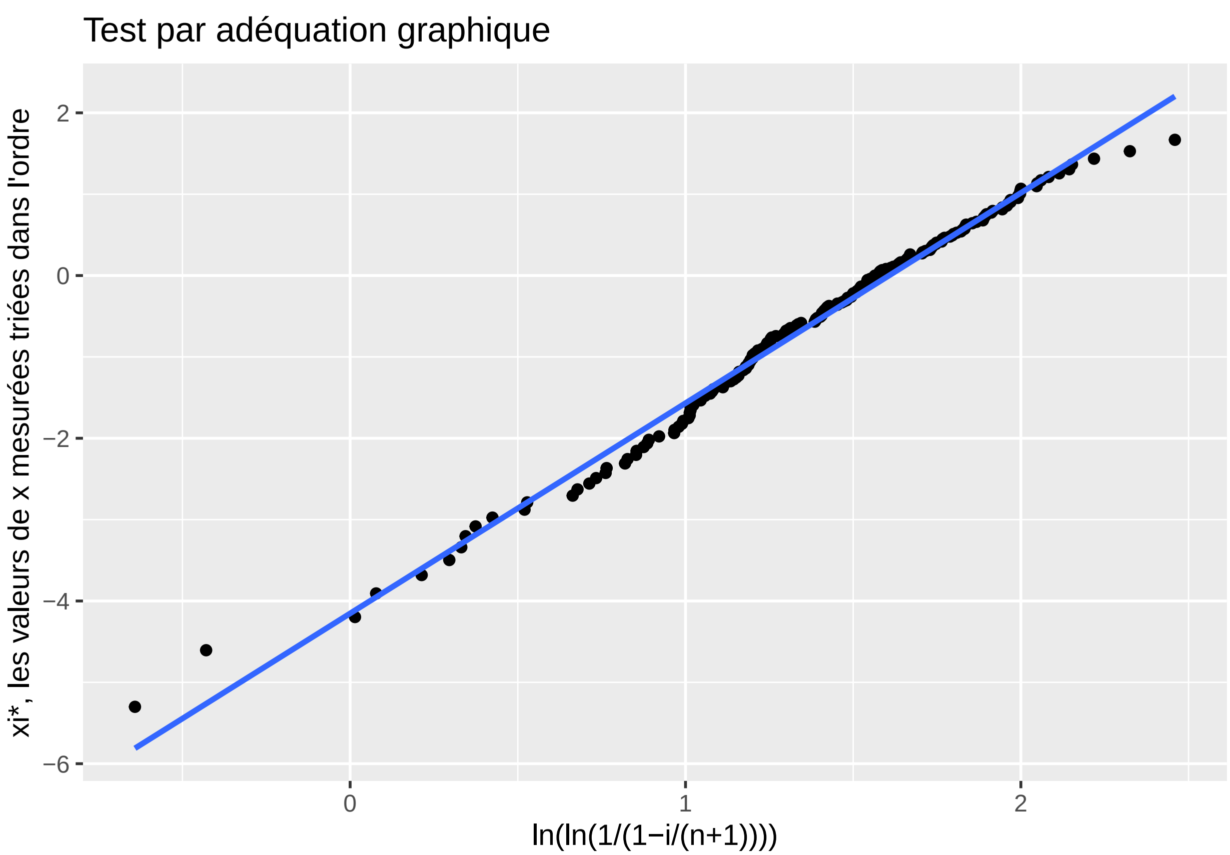
Pour celà on doit tracer le nuage de point :

$$\left(ln(x_i^*), ln\left(ln\left(\frac{1}{1-\frac{i}{n+1}} \right) \right) \right)$$

Avec les x_i^* sont les données mesurées triées dans l'ordre.

```
nuage2 = tibble(log(log(1/(1-((1:200)/(n+1))))),log(sort(df[,"xi_sys2"])))
colnames(nuage2) <- c('formula','x')

ggplot() +
    geom_point(aes(x = nuage2$`x`,y = nuage2$formula)) +
    geom_smooth(aes(x = nuage2$`x`,y = nuage2$formula),
                se = FALSE,
                method = lm) +
    labs(y = "xi*, les valeurs de x mesurées triées dans l'ordre",
         x = "ln(ln(1/(1-i/(n+1))))",
         title = "Test par adéquation graphique")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Par adéquation graphique on trouve une très bonne droite.

On peut effectuer la régression pour conclure sur l'adéquation.

```
lm(nuage2$formula~nuage2$x) %>% summary()
```

```
##
## Call:
## lm(formula = nuage2$formula ~ nuage2$x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53439 -0.04541  0.02479  0.07192  0.65719
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.15267    0.02674  -155.3   <2e-16 ***
## nuage2$x     2.58428    0.01823   141.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1227 on 198 degrees of freedom
## Multiple R-squared:  0.9902, Adjusted R-squared:  0.9902
## F-statistic: 2.01e+04 on 1 and 198 DF,  p-value: < 2.2e-16
```

Avec une $p_{value} = 2.2e-16$ et un $R^2 = 0.9902$ on conclure à la pertinence d'effectuer une droite de regression sur ces données.

On peut donc conclure que l'échantillon n°2 est issus d'un Processus de Poisson Homogène (PPH) de type Weibull avec maintenance parfaite.

On peut effectuer deux estimations des paramètres.

Estimation des paramètres

Graphiquement on a :

$$\text{Pente de la droite de regression} = \hat{\beta}_{gph}\text{Ordonnée à l'origine} = -\hat{\beta}_{gph}ln(\hat{\theta}_{gph})\hat{\theta}_{gph} = e^{-\frac{OaO}{\hat{\beta}_{gph}}}$$

Donc on a :

```
beta = 2.58428
theta = exp(-(-4.15267)/(beta))
estim_param_sys2_gph = list(beta = beta, theta = theta)
print(estim_param_sys2_gph)
```

```
## $beta
## [1] 2.58428
##
## $theta
## [1] 4.987308
```

Par estimation par maximum de vraisemblance :

```
# Focntion du package EWGoF
estim_param_sys2_mv <- MLEst(df[,"xi_sys2"])
estim_param_sys2_mv$eta
```

```
## [1] 4.980278
```

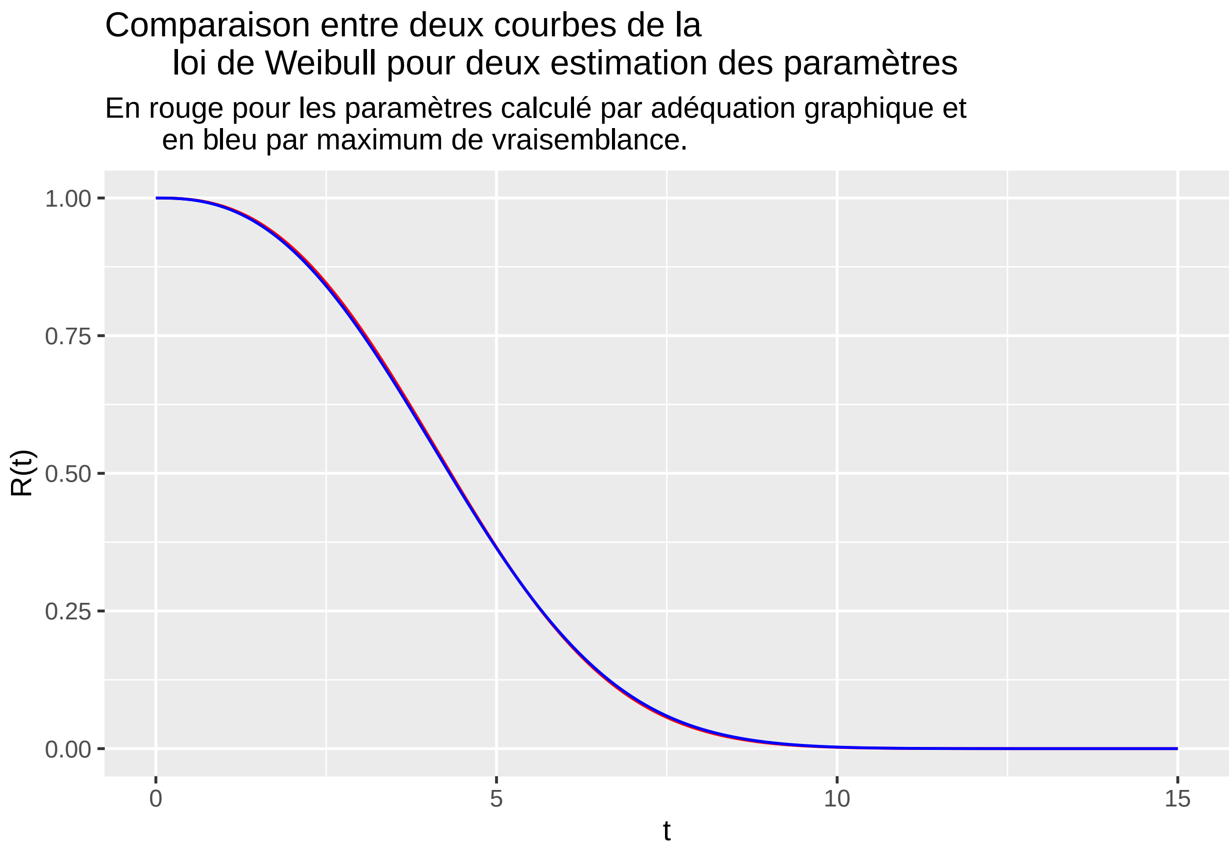
```
estim_param_sys2_mv$beta
```

```
## [1] 2.535939
```

Les deux estimations sont très proches.

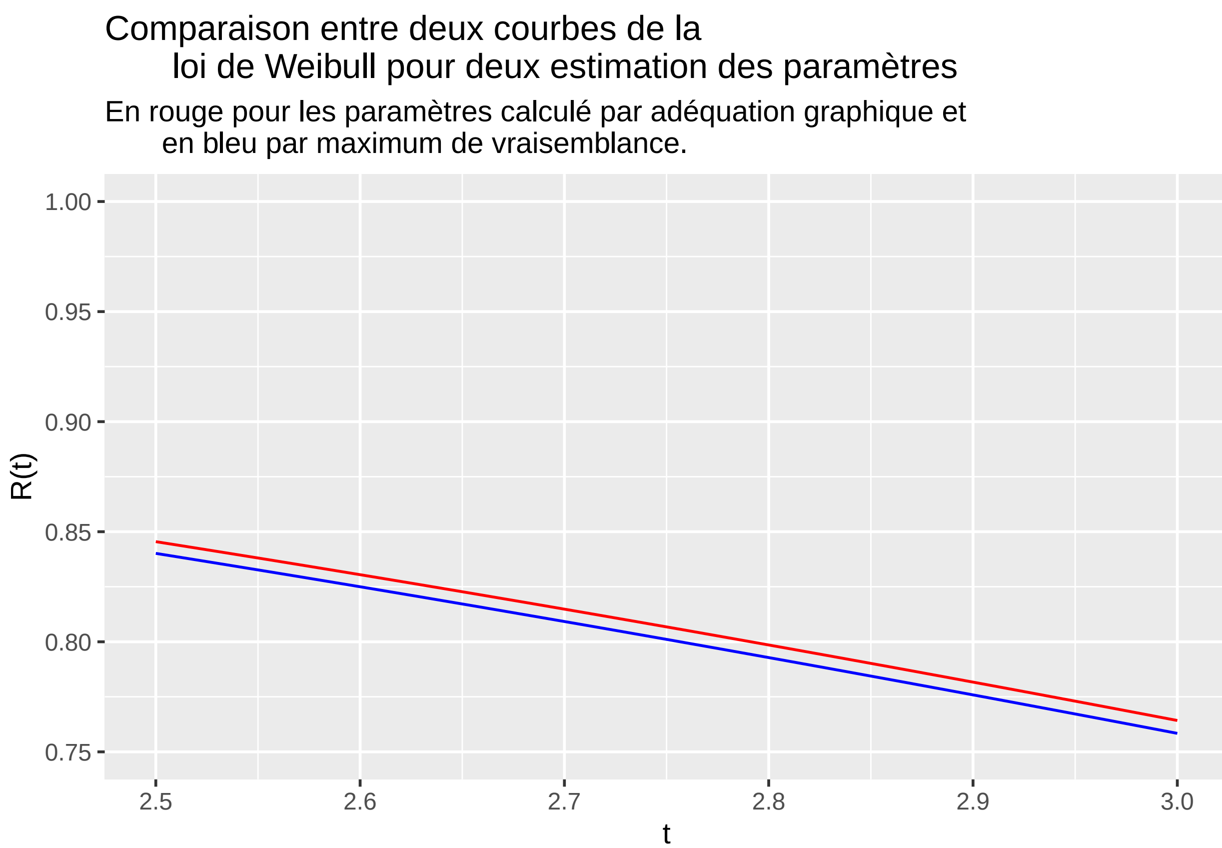
Représentation graphique des deux modèles

```
t = seq(0,15,0.001)
ggplot() +
  geom_line(
    aes(t, 1-pweibull(t,
                shape = estim_param_sys2_gph$beta,
                scale = estim_param_sys2_gph$theta)),
    color = 'red') +
  geom_line(
    aes(t, 1-pweibull(t,
                shape = estim_param_sys2_mv$beta,
                scale = estim_param_sys2_mv$eta)),
    color = 'blue') +
  labs(x = "t",
      y = "R(t)",
      title = "Comparaison entre deux courbes de la
      loi de Weibull pour deux estimation des paramètres",
      subtitle = "En rouge pour les paramètres calculé par adéquation graphique et
      en bleu par maximum de vraisemblance.")
```



```r
t = seq(0,15,0.001)
ggplot() +
  geom_line(
    aes(t, 1-pweibull(t,
                      shape = estim_param_sys2_gph$beta,
                      scale = estim_param_sys2_gph$theta)),
    color = 'red') +
  geom_line(
    aes(t, 1-pweibull(t,
                      shape = estim_param_sys2_mv$beta,
                      scale = estim_param_sys2_mv$eta)),
    color = 'blue') +
  labs(x = "t",
       y = "R(t)",
       title = "Comparaison entre deux courbes de la
       loi de Weibull pour deux estimation des paramètres",
       subtitle = "En rouge pour les paramètres calculé par adéquation graphique et
       en bleu par maximum de vraisemblance.") +
  xlim(2.5,3) +
  ylim(0.75,1)
```

```
## Warning: Removed 14500 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 14500 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Les deux solutions offrent des résultats similaires, la différence est au centième.

Conclusion

l'échantillon n°2 est issue d'une variable aléatoire suivant une loi de Weibull. Nous disposons de deux estimations de paramètres :

- $\beta = 2.58428$ et $\theta = 4.987308$ pour adéquation graphique.
- $\beta = 2.535939$ et $\theta = 4.980278$ par estimation du maximum de vraisemblance.

On ne peut pas trancher simplement en effectuant une moyenne des deux paramètres. On peut choisir l'une ou l'autre ou procéder à d'avantage d'analyses.

Je propose de conserver l'estimation par maximum de vraisemblance (la lecture graphique induit une erreur supplémentaire lors de la régression).

Dans ce cas on a : $MMTF_{sys_2} = \theta\Gamma(1 + \frac{1}{\beta})$

La fonction `gamma()` de R permet de calculer $\Gamma(x)$.

```
print("MTTF pour le système 2 :")
```

```
## [1] "MTTF pour le système 2 :"
```

```
estim_param_sys2_mv$eta*gamma(1+1/estim_param_sys2_mv$beta)
```

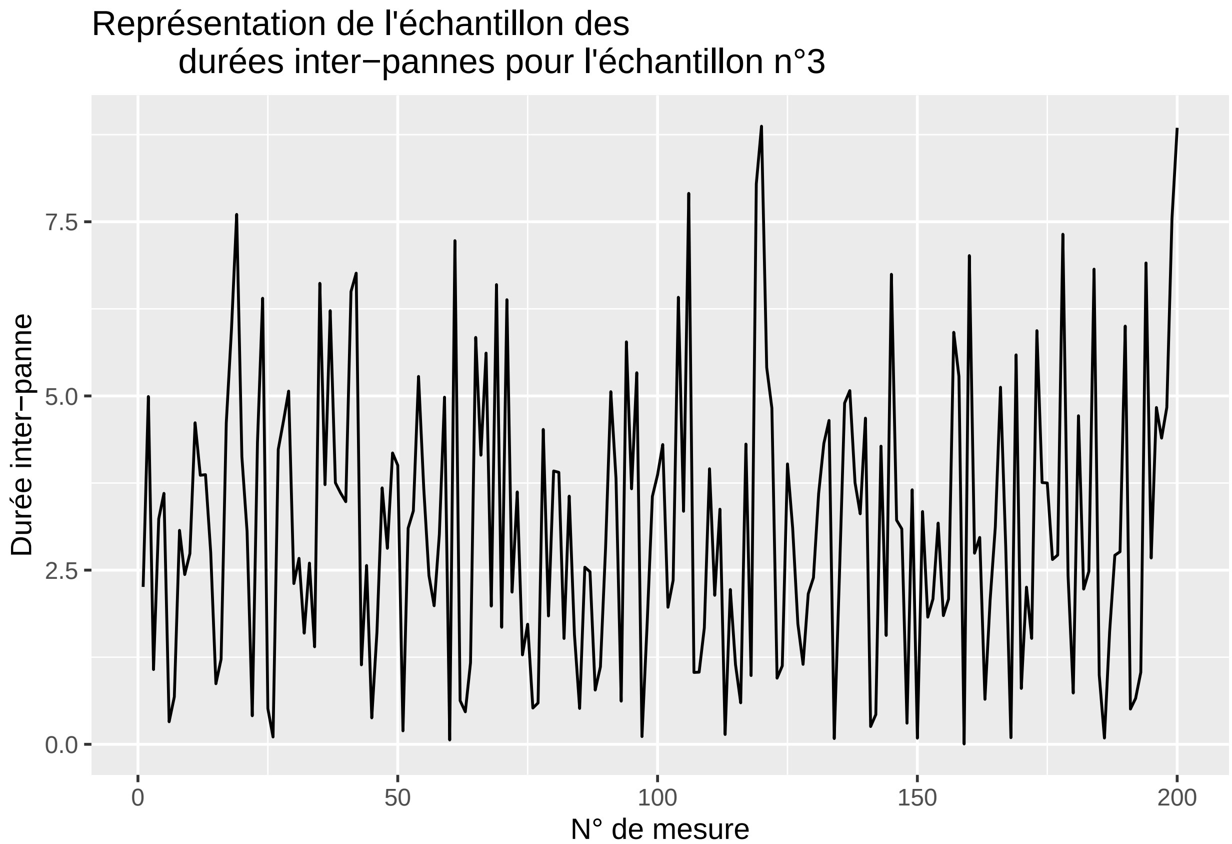
```
## [1] 4.420431
```

Échantillon n°3

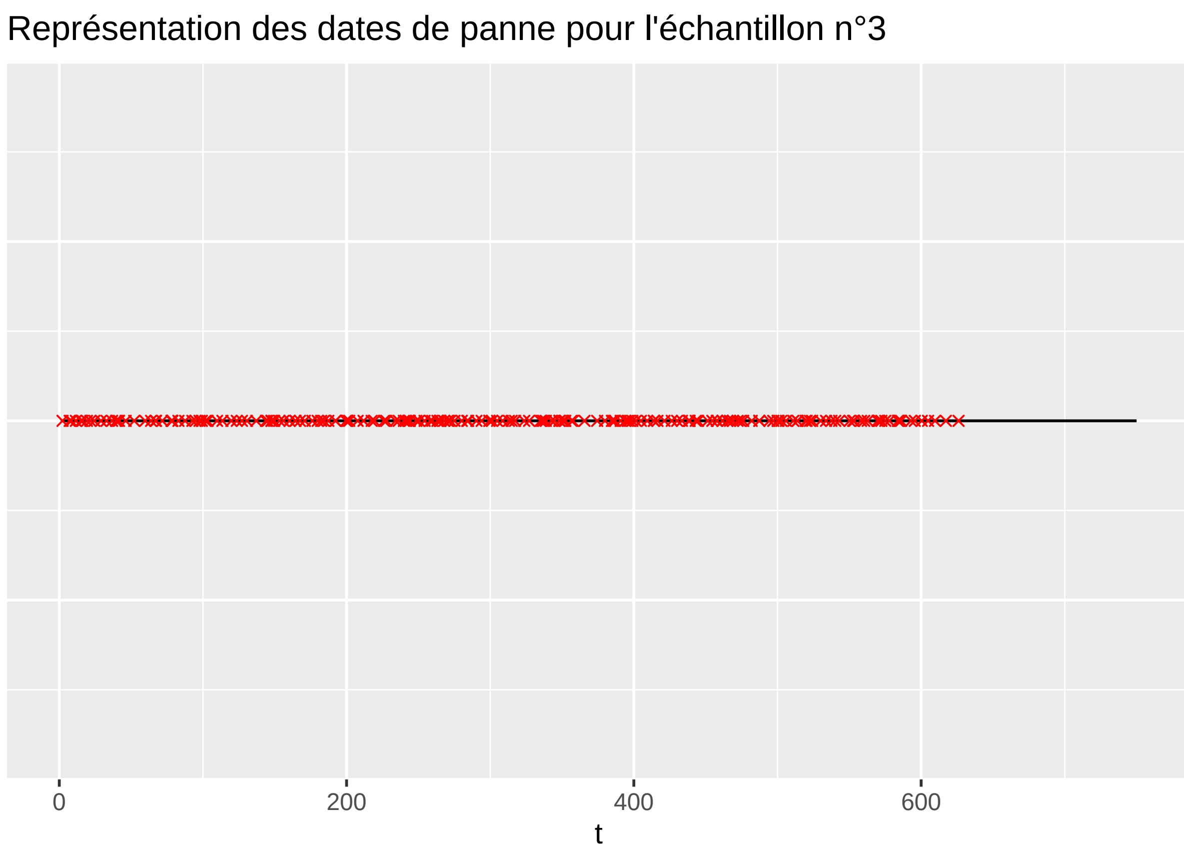
On peut commencer par tracer les données :

- l'évolution des durées inter-pannes,
- les dates de panne.

```
ggplot() +
    geom_line(aes(x = 1:200, y = df[,"xi_sys3"])) +
    labs(x = "N° de mesure",
         y = "Durée inter-panne",
         title = "Représentation de l'échantillon des
         durées inter-pannes pour l'échantillon n°3")
```




```
ggplot() +
    geom_line(aes(x = 1:750, y = 1)) +
    geom_point(aes(x = df[,"ti_sys3"], y = 1)
               , color = "red",
               shape = 4) +
    labs(x = "t",
         y = "",
         title = "Représentation des dates de panne pour l'échantillon n°3") +
    guides(y = "none")
```



A première vue il n'y a pas de tendance dans le jeu de données.

Tendance

Pour observer la tendance on peut utiliser le test de Laplace.

```
compute_laplace_stat(df[,"ti_sys3"])
```

```
## [1] -0.2057661
```

Là encore, on est en dessous de 1.96, on peut conclure à l'absence de tendance.

Adéquation à la loi exponentielle

On réalise un test de Bartlett :

```
compute_bartlett_stat(df[200,"ti_sys3"], df[,"xi_sys3"])
```

```
## [1] 125.6443
```

Dans ce cas on va rejeter H_0 car nous ne nous trouvons pas entre les quantiles du χ^2 :

```
qchisq(p = 0.025, n-1)
```

```
## [1] 161.8262
```

```
qchisq(p = 0.975, n-1)
```

```
## [1] 239.9597
```

Cependant on rejette H_0 d'assez peu il faut donc rester critique sur les résultats obtenus par la suite.

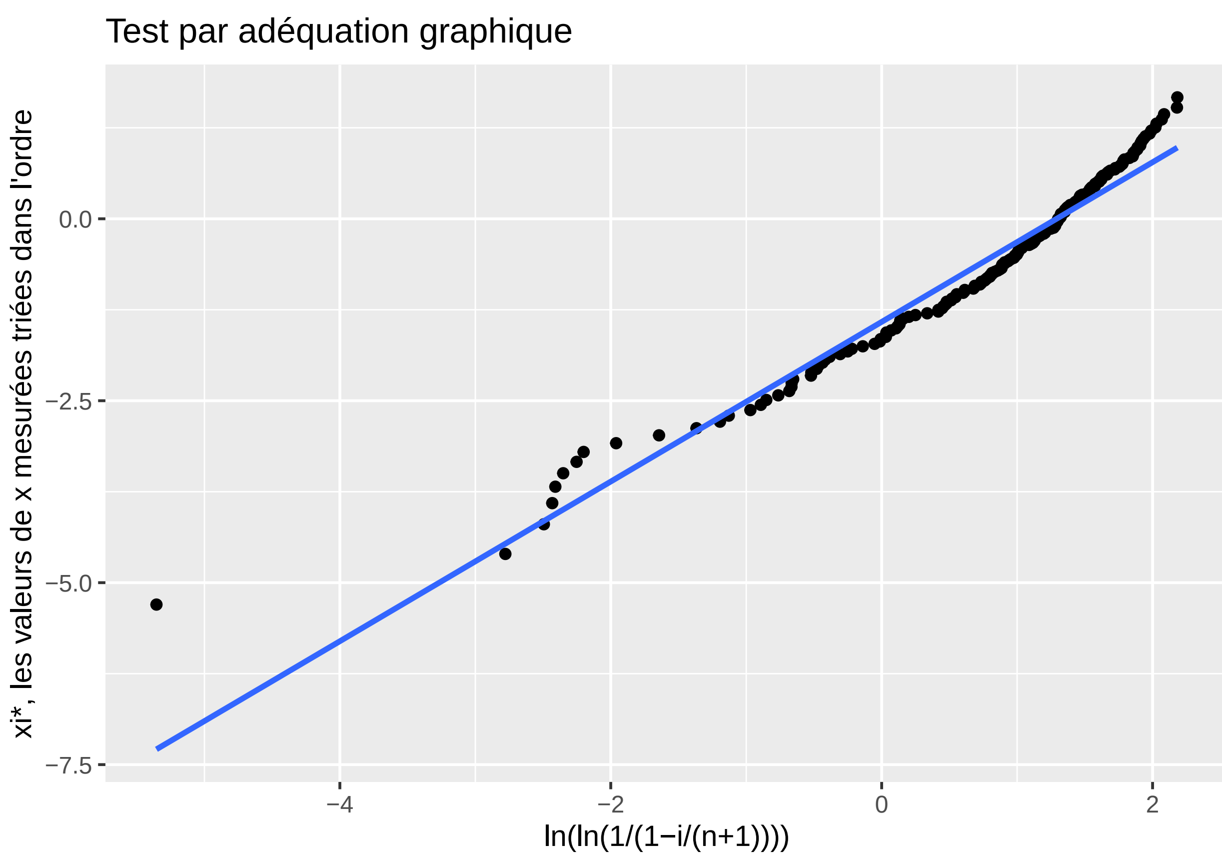
On conclue donc que la loi exponentielle n'est pas adaptée pour modéliser ces données.

Adéquation à la loi de Weibull par maintenance parfaite

On réutilise la même idée que pour le second échantillon : on passe par l' adéquation graphique.

```
nuage3 = tibble(log(log(1/(1-((1:200)/(n+1))))),log(sort(df[,"xi_sys3"])))
colnames(nuage3) <- c('formula','x')

ggplot() +
    geom_point(aes(x = nuage3$`x`,y = nuage3$formula)) +
    geom_smooth(aes(x = nuage3$`x`,y = nuage3$formula),
                se = FALSE,
                method = lm) +
    labs(y = "xi*, les valeurs de x mesurées triées dans l'ordre",
         x = "ln(ln(1/(1-i/(n+1))))",
         title = "Test par adéquation graphique")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



La droite ne pas parfaitement avec les données mais reste plutôt bonne. On peut ajouter la régression linéaire à l'analyse :

```
lm(nuage3$formula~nuage3$x) %>% summary()
```

```
##
## Call:
## lm(formula = nuage3$formula ~ nuage3$x)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3172 -0.1839 -0.0856  0.1433  1.9889
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.41662    0.02293  -61.79   <2e-16 ***
## nuage3$x     1.09692    0.01703   64.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2652 on 198 degrees of freedom
## Multiple R-squared:  0.9544, Adjusted R-squared:  0.9542
## F-statistic:  4147 on 1 and 198 DF,  p-value: < 2.2e-16
```

On peut considérer au vu du R^2 et de la p_{value} que la loi de Weibull est adaptée à modéliser ces données.

On peut donc conclure que l'échantillon n°2 est issus d'un Processus de Poisson Homogène (PPH) de type Weibull avec maintenance parfaite.

Estimation des paramètres

```
beta = 1.09692
theta = exp(-(-1.41662)/(beta))
estim_param_sys3_gph = list(beta = beta, theta = theta)
print("Graphiquement")
```

```
## [1] "Graphiquement"
```

```
print(estim_param_sys3_gph)
```

```
## $beta
## [1] 1.09692
##
## $theta
## [1] 3.638067
```

```
# Focntion du package EWGoF
print("Maximum de vraissemblance")
```

```
## [1] "Maximum de vraissemblance"
```

```
estim_param_sys3_mv <- MLEst(df[,"xi_sys3"])
print("beta")
```

```
## [1] "beta"
```

```
estim_param_sys3_mv$beta
```

```
## [1] 1.410595
```

```
print("theta")
```

```
## [1] "theta"
```

```
estim_param_sys3_mv$eta
```

```
## [1] 3.406006
```

Dans ce cas, on a une incohérence entre les deux estimations pour la valeur de β. Il faut donc rejeter l'hypothèse selon laquelle ces données seraient issues d'une loi de Weibull.

Conclusion

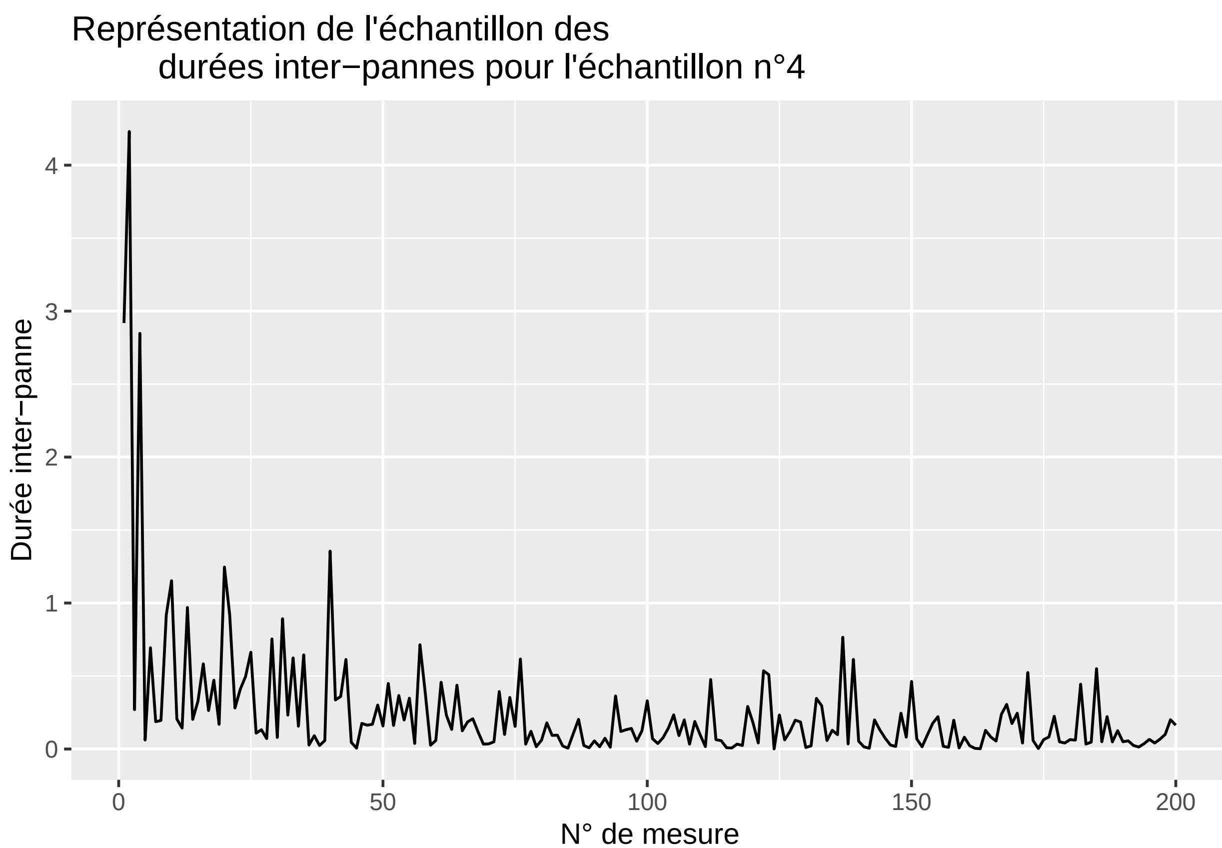
Plus d'analyse devraient être menées pour conclure sur ce jeu de données.

Échantillon n°4

On peut commencer par tracer les données :

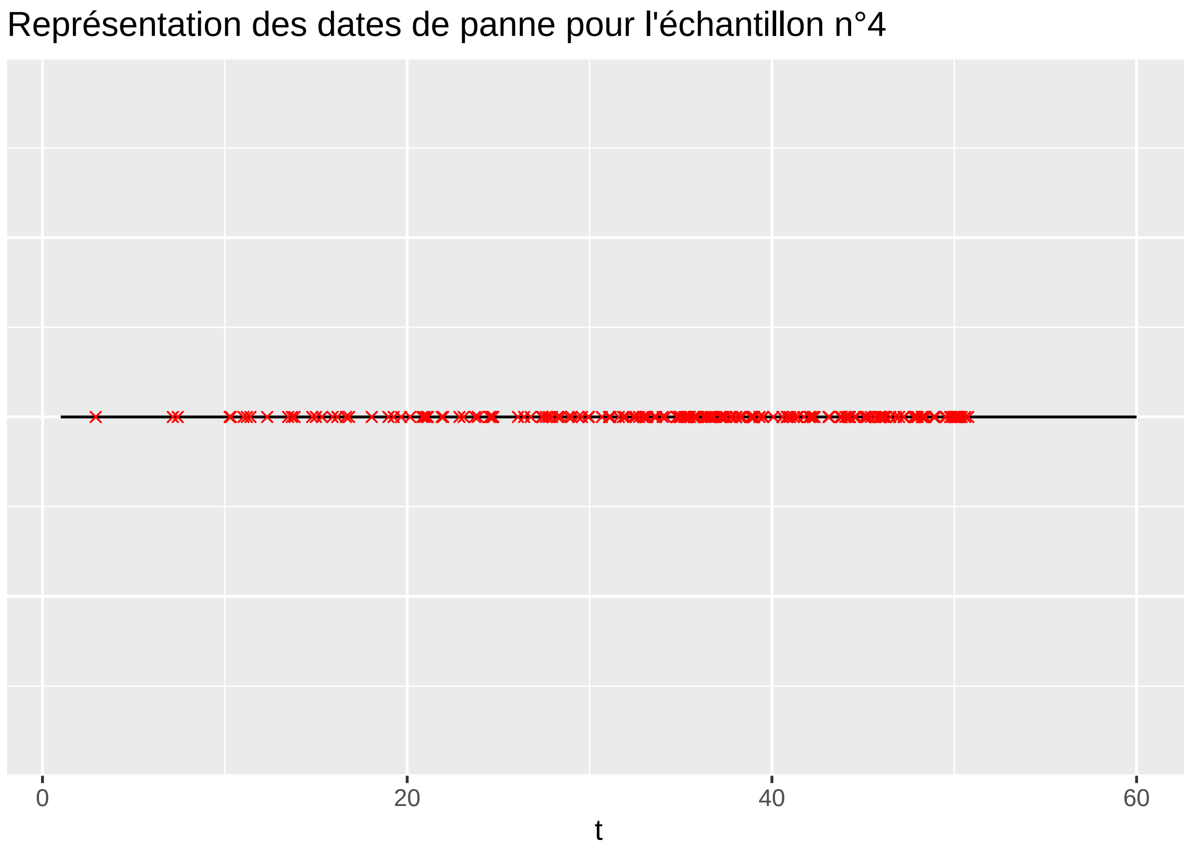
- l'évolution des durées inter-pannes,
- les dates de panne.

```
ggplot() +
    geom_line(aes(x = 1:200, y = df[,"xi_sys4"])) +
    labs(x = "N° de mesure",
         y = "Durée inter-panne",
         title = "Représentation de l'échantillon des
         durées inter-pannes pour l'échantillon n°4")
```




```
ggplot() +
    geom_line(aes(x = 1:60, y = 1)) +
    geom_point(aes(x = df[,"ti_sys4"], y = 1)
              , color = "red",
              shape = 4) +
    labs(x = "t",
         y = "",
```

```
    title = "Représentation des dates de panne pour l'échantillon n°4") +
  guides(y = "none")
```



Il semble presque *évident* que ces données sont issues d'une loi avec tendance.

Tendance

Pour conclure sur la tendance on peut utiliser le test de Laplace.

```
compute_laplace_stat(df[,"ti_sys4"])
```

```
## [1] 9.690126
```

On est bien au dessus de la valeur critique 1.96, on ne rejette donc pas l'hypothèse H_0 et on doit conclure à la présence de tendance.

On peut préciser avec le test de place le type de tendance :

```
compute_laplace_stat(df[,"ti_sys4"]) < qnorm(p = 0.05, mean = 0, sd = 1)
```

```
## [1] FALSE
```

```
compute_laplace_stat(df[,"ti_sys4"]) > qnorm(p = 1-0.05, mean = 0, sd = 1)
```

```
## [1] TRUE
```

On est donc dans un système avec décroissance de fiabilité car la statistique de test U est plus grande que le quantile de niveau $1 - 0.05 = 0.95$.

Adéquation à la loi de Weibull avec maintenance minimale

Dans le cas d'une tendance de fiabilité décroissante on peut d'abord tester une l'adéquation à une loi de Weibull avec maintenance minimale.

On appelle ce processus un processus de poisson non homogène (PPNH) ou *power law process (PLP)*.

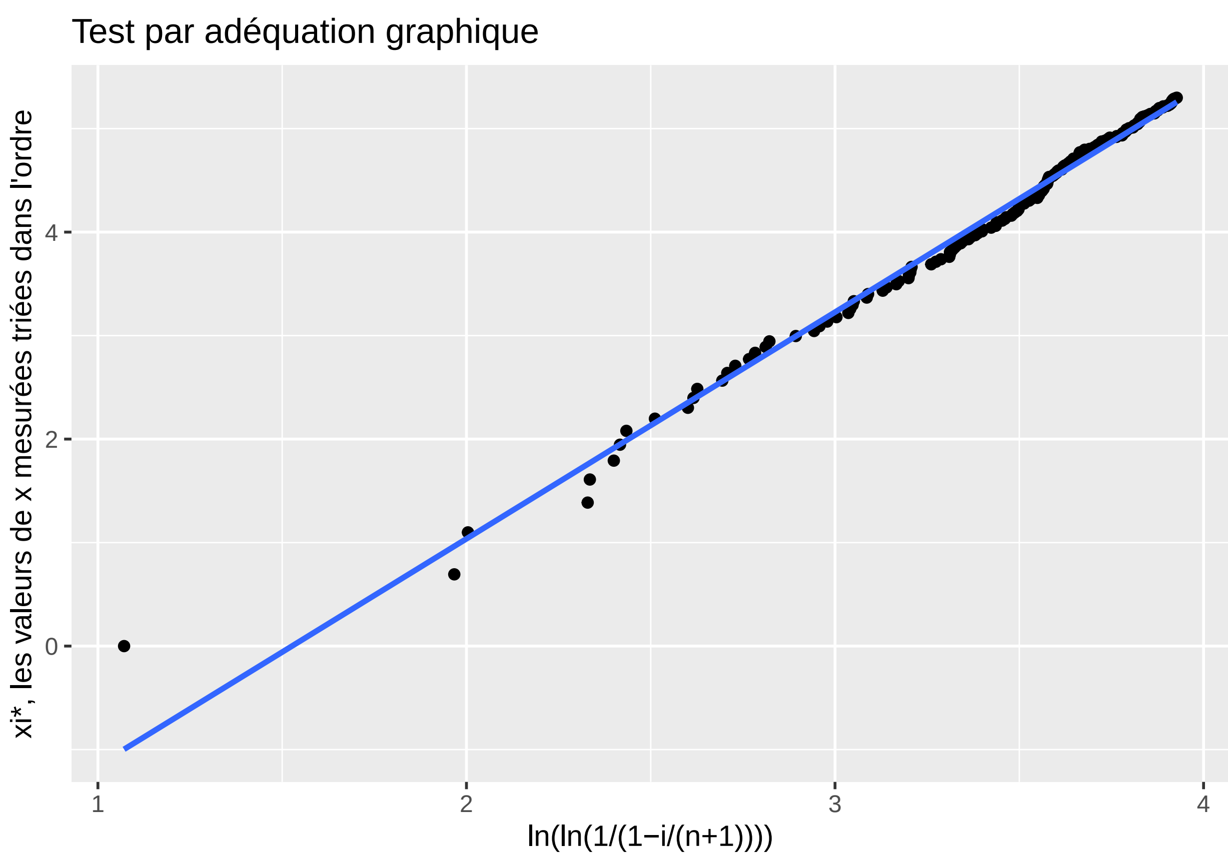
Dans ce cas on parle de **loi de gamma-Weibull**.

Graphiquement on va regarder l'adéquation du nuage de point suivant :

$$(ln(t_i), ln(i))$$

```
nuage4 = tibble(log(df[,"ti_sys4"]),log(1:200))
colnames(nuage4) <- c('x','formula')

ggplot() +
    geom_point(aes(x = nuage4$`x`,y = nuage4$formula)) +
    geom_smooth(aes(x = nuage4$`x`,y = nuage4$formula),
                se = FALSE,
                method = lm) +
    labs(y = "xi*, les valeurs de x mesurées triées dans l'ordre",
         x = "ln(ln(1/(1-i/(n+1))))",
         title = "Test par adéquation graphique")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



L'adéquation graphique est plutôt bonne. On vérifie avec la fonction `lm` :

```r
lm(nuage4$formula~nuage4$x) %>% summary()
```

```
##
## Call:
## lm(formula = nuage4$formula ~ nuage4$x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37061 -0.06391  0.02346  0.04567  0.99743
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.34278    0.05749  -58.14   <2e-16 ***
## nuage4$x     2.18977    0.01631  134.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09976 on 198 degrees of freedom
## Multiple R-squared:  0.9891, Adjusted R-squared:  0.9891
## F-statistic: 1.802e+04 on 1 and 198 DF,  p-value: < 2.2e-16
```

On a une faible p_{value} et un $R^2 = 0.9891$. Le modèle choisit semble pertinent.

Estimation des paramètres

On a donc :

$$\hat{\beta}_{gph} = \text{Pente de la droiteOaO} = ln(\alpha)\alpha = exp(OaO)$$

```r
beta = 2.18977
alpha = exp(-3.34278)
estim_param_sys4_gph = list(beta = beta, alpha = alpha)
print(estim_param_sys4_gph)
```

```
## $beta
## [1] 2.18977
##
## $alpha
## [1] 0.03533858
```

On peut aussi utiliser l'estimateur du maximum de vraisemblance.

On a :

$$\hat{\alpha} = \frac{n}{t_n^{\hat{\beta}}}; \hat{\beta} = \frac{n}{\sum_{i=1}^{n} ln\left(\frac{t_n}{t_i}\right)}$$

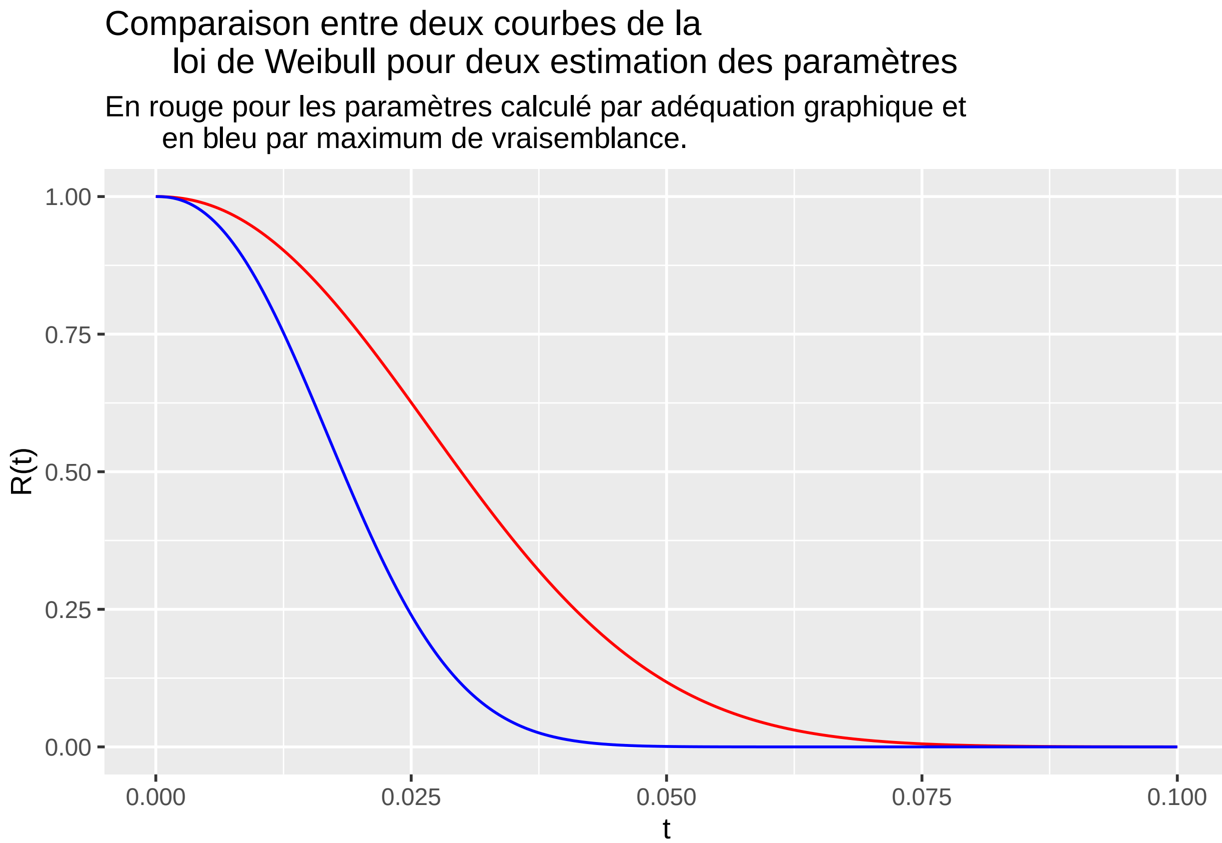
```r
beta = n/sum(log(df[n,"ti_sys4"]/df[,"ti_sys4"]))
alpha = n/(df[n,"ti_sys4"]^beta)
estim_param_sys4_mv = list(
  alpha = alpha,
  beta = beta
)
estim_param_sys4_mv
```

```
## $alpha
## [1] 0.02145103
##
## $beta
## [1] 2.327401
```

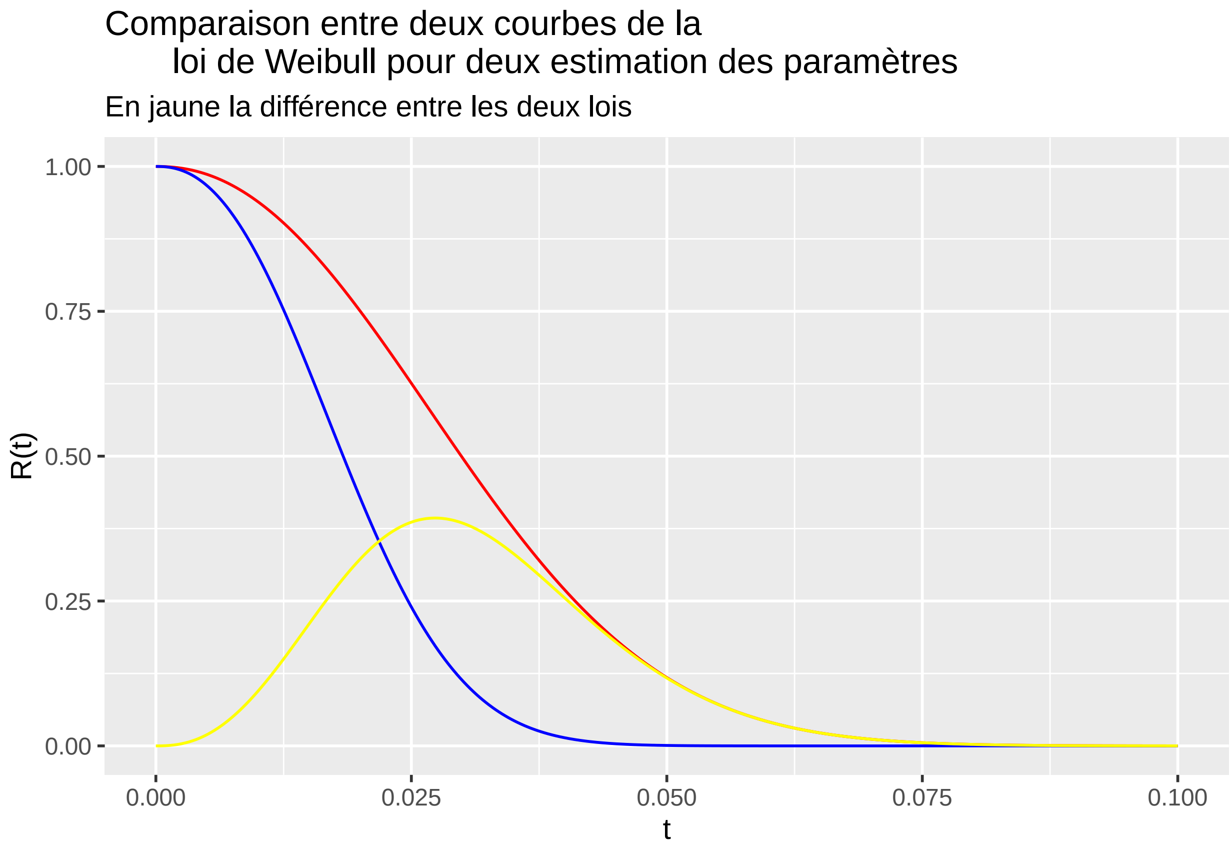
Nos deux approximations sont assez proches mais avec une loi comme la loi de Weibull, la moindre variation du paramètre (notamment pour β) peut avoir une incidence certaine sur les performances dur modèle choisit.

Représentation graphique des deux modèles

```r
t = seq(0,0.10,0.0001)
p <- ggplot() +
  geom_line(
    aes(t, 1-pweibull(t,
                      shape = estim_param_sys4_gph$beta,
                      scale = estim_param_sys4_gph$alpha)),
    color = 'red') +
  geom_line(
    aes(t, 1-pweibull(t,
                      shape = estim_param_sys4_mv$beta,
                      scale = estim_param_sys4_mv$alpha)),
    color = 'blue') +
  labs(x = "t",
       y = "R(t)",
       title = "Comparaison entre deux courbes de la
       loi de Weibull pour deux estimation des paramètres",
       subtitle = "En rouge pour les paramètres calculé par adéquation graphique et
       en bleu par maximum de vraisemblance.")

p
```




```r
a <- (
  (1-pweibull(t,
          shape = estim_param_sys4_gph$beta,
          scale = estim_param_sys4_gph$alpha)) -
    (1-pweibull(t,
          shape = estim_param_sys4_mv$beta,
          scale = estim_param_sys4_mv$alpha))
)

p + geom_line(aes(x = t, y = a), color = 'yellow') +
  labs(subtitle = "En jaune la différence entre les deux lois")
```



Il y a entre les deux lois une différence de presque 40% au plus (pour des valeurs de durée inter-panne proches de 0.025).

Conclusion

Il semble que les données proviennent d'une loi de Weibull avec tendance.

Le système suit donc un processus de poisson non homogène.

Partie 2 : Données réelles

Dans cette partie on étudie un jeu de données issu de l'industrie. Les deux jeux de données sont issus de l'ouvrage : Stochastic Ageing and Dependence for Reliability, par : Chin-Diew Lai et Min Xie.

Les données sont :

Jeu n°1

Jeu n°2

On va donc étudier les deux jeux l'un après l'autre.

Jeu n°1 : Nombre de cycle avant panne pour 60 appareils électriques.

Première approche

On commence par créer le vecteur contenant les données :

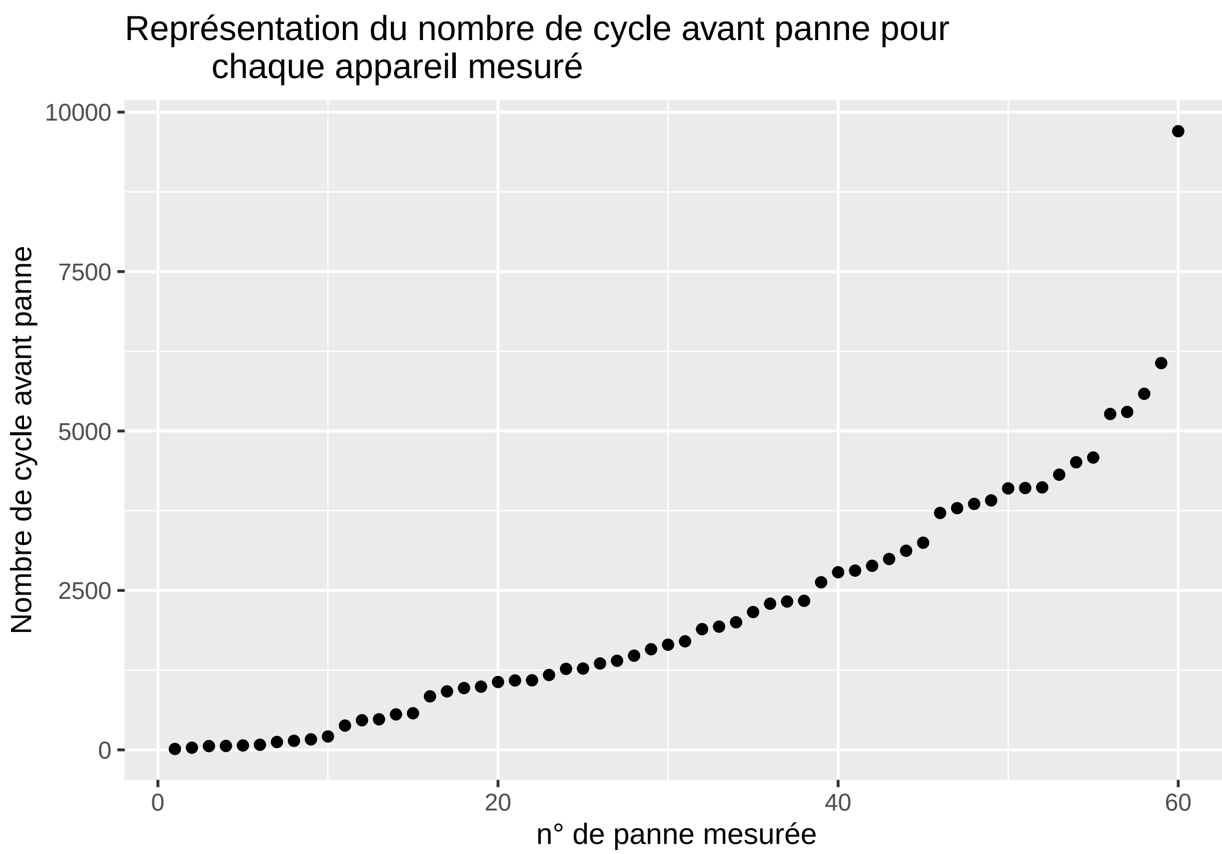
```r
nbr_cycle_avant_panne_1 <- c(14,34,59,61,69,80,123,142,165,210,381,464,479,556,574,
                            839,917,969,991,1064,1088,1091,1174,1270,1275,1355,
                            1397,1477,1578,1649,1702,1893,1932,2001,2161,2292,
                            2326,2337,2628,2785,2811,2886,2993,3122,3248,3715,
                            3790,3857,3912,4100,4106,4116,4315,4510,4584,5267,
                            5299,5583,6065,9701)
nbr_cycle_avant_panne_1
```

```
##  [1]   14   34   59   61   69   80  123  142  165  210  381  464  479  556  574
## [16]  839  917  969  991 1064 1088 1091 1174 1270 1275 1355 1397 1477 1578 1649
## [31] 1702 1893 1932 2001 2161 2292 2326 2337 2628 2785 2811 2886 2993 3122 3248
## [46] 3715 3790 3857 3912 4100 4106 4116 4315 4510 4584 5267 5299 5583 6065 9701
```

Les données ont été rangées dans l'ordre croissant et proviennent de 60 appareils différents que l'on peut supposer indépendants et identiquement distribués.

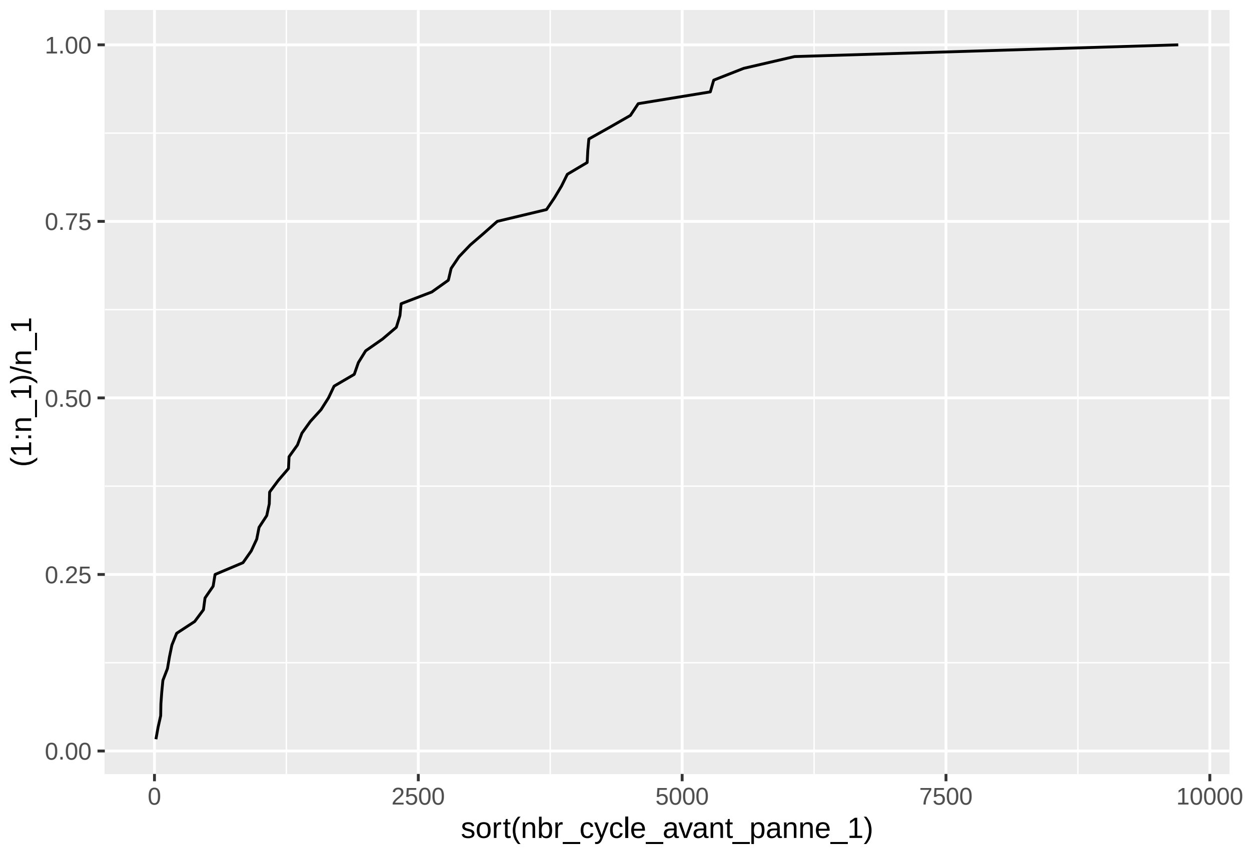
On peut commencer par tracer les données :

```r
ggplot() +
    geom_point(aes(x = 1:60, y = nbr_cycle_avant_panne_1)) +
    labs(x = "n° de panne mesurée",
         y = "Nombre de cycle avant panne",
         title = "Représentation du nombre de cycle avant panne pour
         chaque appareil mesuré")
```



On peut aussi tracer la fonction de répartition empirique :

```
# taille de l'échantillon
n_1 = length(nbr_cycle_avant_panne_1)

plot_repartion_empirique_1 <- geom_line(aes(x = sort(nbr_cycle_avant_panne_1),
                                            y = (1:n_1)/n_1))

ggplot() + plot_repartion_empirique_1
```



Dans ce cas on ne peut pas tester absence de tendance car nous de disposons pas de mesures répétées pour un seul et même système mais de 60 mesures de 60 systèmes différents.

On peut ensuite calculer les estimateurs de la moyenne et de la variance de notre échantillon :

$$\hat{m} = \frac{1}{n}\sum_{k=1}^{n} x_k ; \hat{S}_n^2 = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \hat{m})^2$$

```
# estimateur sans biais de l'espérance
estim_mean_1 <- sum(nbr_cycle_avant_panne_1)/n_1

# estimateur sans biais de la variance
estim_var_1 <- sum((nbr_cycle_avant_panne_1 - estim_mean_1)^2)/(n_1 - 1)

sprintf("Estimateur de la moyenne empirique : %f", estim_mean_1)
```

```
## [1] "Estimateur de la moyenne empirique : 2193.033333"
```

```
sprintf("Estimateur de la variance empirique : %f", estim_var_1)
```

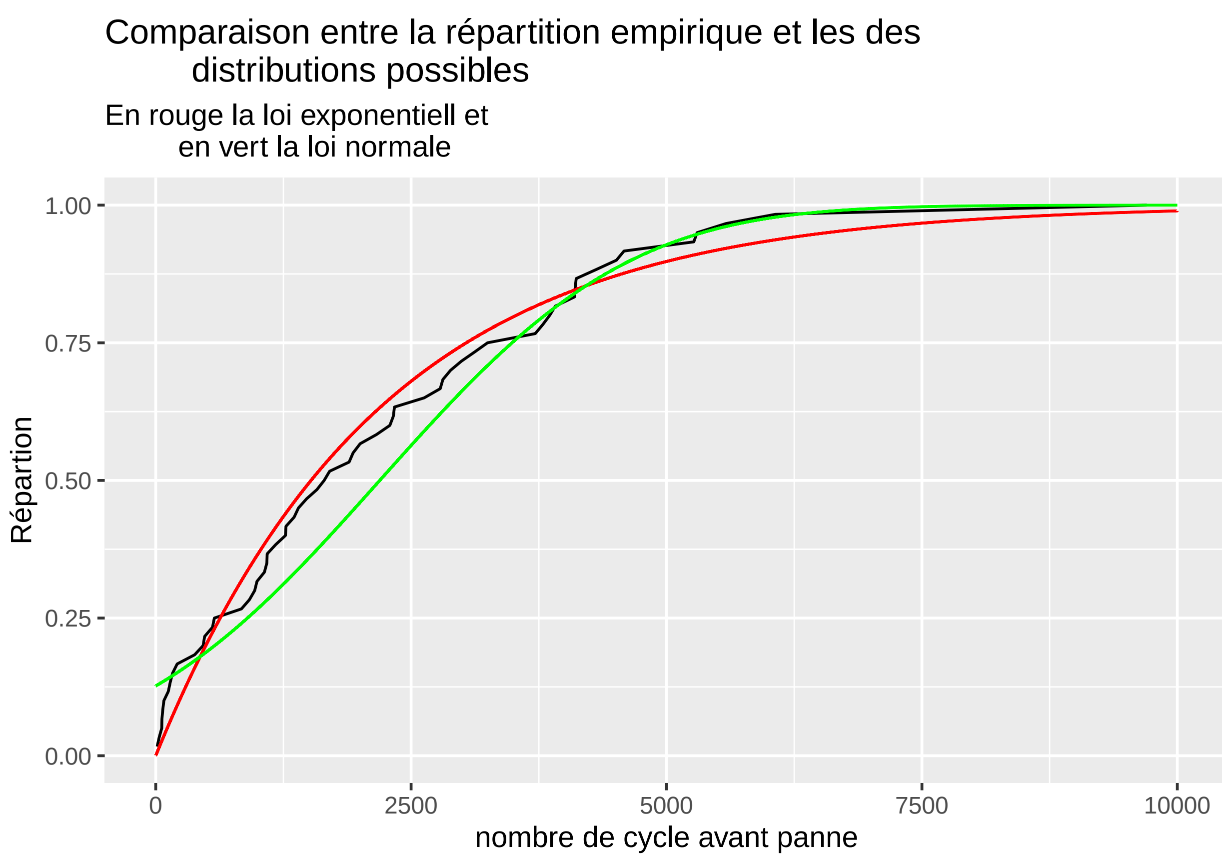
```
## [1] "Estimateur de la variance empirique : 3686963.049718"
```

Pour la loi

On peut comparer la répartition empirique avec la répartition de plusieurs loi connues comme la loi exponentielle ou la loi normale.

```
# Répartition théorique de la loi exponentielle prenant pour paramètre
# un lambda = 1/moyenne empirique
```

```r
plot_repartion_exp_1 <- geom_line(aes(x = seq(1,10000,0.1),
                                      y = pexp(seq(1,10000,0.1),
                                               rate = 1/estim_mean_1)),
                                  color = 'red')

# Répartion théorique de la loi exponentielle prenant pour paramètre
# la moyenne empirique et la variance empirique

plot_repartion_norm_1 <- geom_line(aes(x = seq(1,10000,0.1),
                                       y = pnorm(seq(1,10000,0.1),
                                                 mean = estim_mean_1,
                                                 sd = sqrt(estim_var_1))),
                                   color = 'green')

ggplot() +
    plot_repartion_empirique_1 +
    plot_repartion_exp_1 +
    plot_repartion_norm_1 +
    labs(x = "nombre de cycle avant panne",
         y = "Répartion",
         title = "Comparaison entre la répartition empirique et les des
         distributions possibles",
         subtitle = "En rouge la loi exponentiell et
         en vert la loi normale")
```



La courbe en rouge (loi exponentielle) semble être plus proche que la courbe en vert (loi normale) des données empiriques.

On peut ensuite appliquer l'analyse tel que l'on a fait pour les précédents jeux de données.

Présence de tendance

On cherche à établir la présence d'une tendance. On peut considérer le jeu de donnnées comme une suite de 60 mesures d'un système avec réparation.

On peut utiliser le test de Laplace.

```
compute_laplace_stat <- function(ti){

  # Prend le tableau des Ti en entrée de fonction et retourne U
  # Fonction du test de Laplace

  n <- length(ti)
  U <- sqrt((12)/((n-1)*ti[n]^2))*(sum(ti)-(n+1)*ti[n]/2)

  U %>% as.numeric %>% return()

}
```

```
xi_1 <- nbr_cycle_avant_panne_1
ti_1 <- cumsum(nbr_cycle_avant_panne_1)
```

```
compute_laplace_stat(xi_1)
```

```
## [1] -7.638036
```

```
qnorm(1-0.5/2, mean = 0, sd = 1)
```

```
## [1] 0.6744898
```

Dans ce cas on a $\{|U| > F^{-1}_{N(0,1)}(1 - \frac{\alpha}{2})\}$

On peut donc rejeter l'absence de tendance.

Exercice 5, Partie 1.

Cette exercice s'intéresse à la simulation et l'optimisation d'une politique de maintenance.

Je décide pour cet exercice de moi même simuler l'ensemble des loi utilisées.

J'utilise uniquement la fonction `runif(1)` qui me permet d'obtenir une réalisation de la loi uniforme entre 0 et 1.

```
# Simulation d'une réalisation de la loi de Weibul de paramètre
# thêta et beta
# remplace la fonction rweibull() de R

realisation_aleatoire_weibull <- function(scale = 1, shape = 1) {
    scale * (-log(runif(1)))^(1/shape) %>% return()
}
```

Partie 1 : Système à un composant

Dans cette première partie on ne traite qu'un composant : une ampoule. Cette dernière a une durée de vie qui suit une loi de Weibull de paramètres $\eta = 3an$ (soit $3 * 365 = 1095jour$, je néglige les années bissextiles). (échelle) et $\beta = 2.5$ (forme).

On planifie des maintenances préventives (MP) tout les x jours avec un coût de $c_{mp} = 200euros$.

En cas de panne du système, on effectue une maintenance corrective (MC) d'un coût de $c_{mc} = 1000euros$.

Toute maintenance remet à neuf le système. On est donc dans le cas d'une maintenance parfaite pour une loi de Weibull (Processus de poisson homogène).

Le temps de maintenance est négligé (remise en marche immédiate).

Sur l'intervalle de temps considéré H (l'horizon de simulation). On peut poser :

- M_n le type de la n-ième maintenance (MP/MC)
- T_n la date de la n-ième maintenance
- C_t le coût total des maintenances effectuées à l'intant t.

Ces trois grandeurs sont des variables aléatoires qui dépendent de x.

Observer une trajectoire du système consiste à simuler les pannes successives du système avec des paramètres données (x,H,η,β,c_{mp},c_{mc}) et de calculer le coût total de cette trajectoire (qui est une variable synthétique dépendant de ces paramètres : $Cf(x, H, \eta, \beta, c_{mp}, c_{mc})$).

Pour calculer le coût moyen associé à une valeurs de x on va simuler un grand nombre de trajectoire avec le jeu de paramètre fixe et calculer la moyenne des coûts obtenus.

On répétera le processus en faisant varier les valeurs de x pour tracer la courbe du prix moyen.

Il nous faut donc :

- une fonction pour simuler une réalisation de $Cf(x, H, \eta, \beta, c_{mp}, c_{mc})$,
- calculer la moyenne des Cf_i pour p réalisations (p grand),
- répéter en faisant varier x.

```
# Fonction pour simuler une trajectoire

realisation_trajectoire_sys1 <- function(param, info = FALSE) {

    # param est un argument de type list() de la forme :
    # param = list(H,x,scale,shape,cmp,cmc)
```

```r
# Attention :
# H, scale et x sont en jours

# les dates de maintenance préventives prévues
echeancier <- param$x * 1:floor(param$H/param$x)
if (info) {
    #debug
    print("=== l'échéancier de maintenance préventive ===")
    print(echeancier)
    print("=== Début de simulation ===")
}
# Le temps courant
t <- 0

# Cout final actuel
Cf <- 0

i <- 1 # un indice

while (t < param$H & i < length(echeancier)) {

    next_panne <- t + realisation_aleatoire_weibull(scale = param$scale,
                                                    shape = param$shape)

    if (info) {
        # debug
        print(paste("Temps courant",t))
        print(paste("Date de la prochaine panne : ",next_panne))
        print(paste(
            "Date de la prochaine maintenance préventive : ",
            echeancier[i]))
    }

    if (next_panne < echeancier[i]) {
        # Dans ce cas la prochaine panne si on laisse vivre le système
        # arrive avant la prochaine maintenance préventive
        # donc on effectue la maintenance corrective

        if (info) {print("MC")}

        # on met à jour le temps courant
        t <- next_panne
        # on effectue une maintenance corrective
        Cf <- Cf + param$cmc
    } else {

        if (info) {print("MP")}

        # on effecute la maintenance préventive avant une survenue de panne
        t <- echeancier[i]
        i = i + 1
        # on effectue une maintenance préventive
        Cf <- Cf + param$cmp
```

```
        }

        if (info) {
            print(paste("Temps courant",t))
            print(paste("Coût actuel",Cf))
            print(paste("val de i",i))
            print("---")
        }
    }

    return(Cf)
}
```

On peut donc simuler une seule trajectoire du système :

```
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 3, # on effectue une maintenance préventive tous les
                   #trois ans
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
realisation_trajectoire_sys1(parametres)
```

```
## [1] 300400
```

Cette trajectoire nous donne un coût total de 300 400€, ce donne un coût moyen de 300,4€ par ans.

On peut ensuite construire une fonction pour calculer le coût moyen unitaire moyen pour une liste de paramètre donnée.

```
compute_mean_cost_by_time <- function(p = 10000, param_sys) {

  # Calcul un coût moyen par unité de temps Cinf(x)
  # On chaque trajectoire est divisée par H, l'horizon de simulation

  frame <- replicate(
    p, realisation_trajectoire_sys1(param = param_sys, info = FALSE))
  frame <- mean(frame/param_sys$H)
  return(frame)
}
```

On peut ensuite calculer le coût moyen pour les paramètres suivants :

- H : 1000 ans
- x : 3 ans (on effectue une MP tout les trois ans)
- $\eta = 3ans$
- $\beta = 2.5$
- $cmc = 1000$, coût d'une maintenance corrective à 1000€
- $cmp = 200$, coût d'une maintenance préventive égale à 200€

```
compute_mean_cost_by_time(p = 10000, param_sys = parametres)
```

```
## [1] 299.6046
```

Cette simulation nous donne donc un coût moyen de 299.60£ par ans si l'on choisit d'effectuer une maintenance préventive tout les trois ans.

On va maintenant chercher à tracer l'évolution du coût moyen en fonction du paramètre x : le temps de cycle entre chaque maintenance préventive.

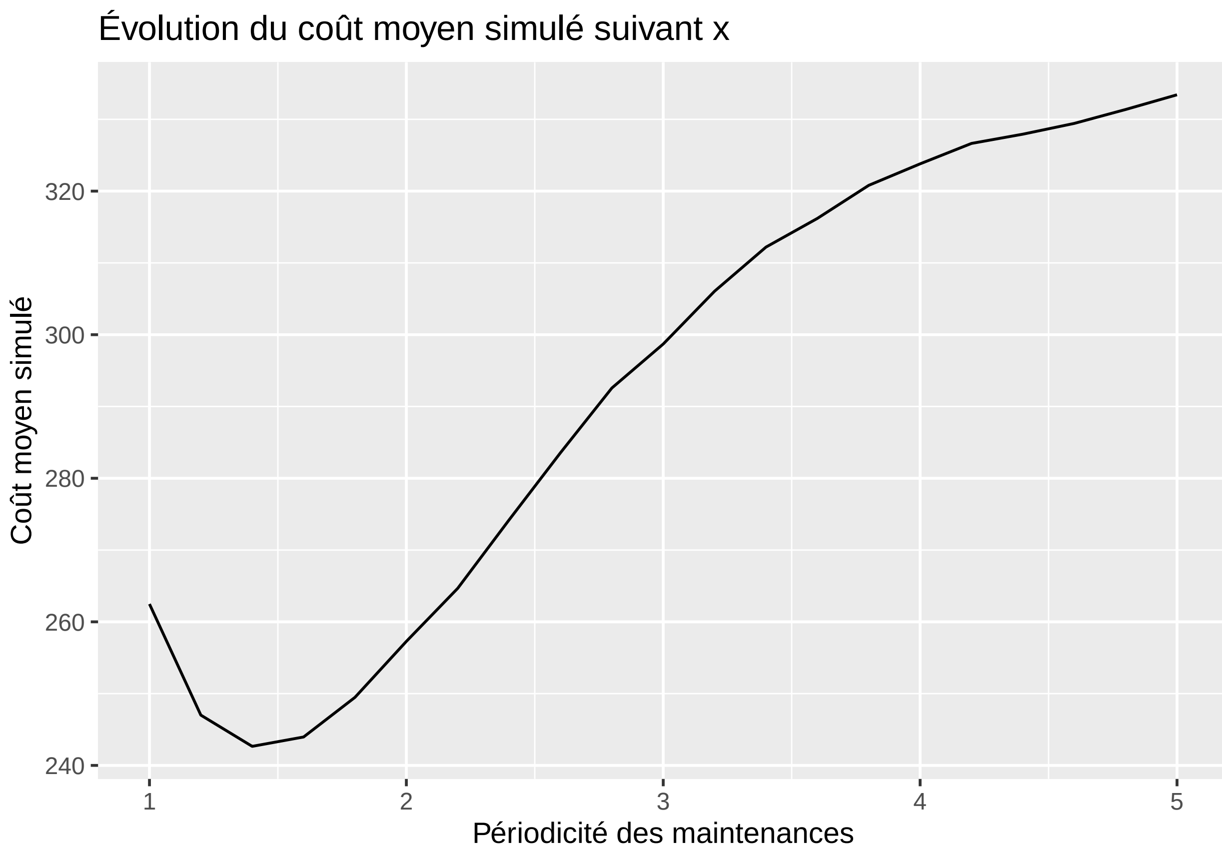
Je vais donc écrire une fonction qui prend deux paramètres :

- une liste de valeurs de x à tester ;
- un jeu de paramètre.

Cette fonction va ensuite tracer la courbe du prix moyen en fonction des valeurs de x.

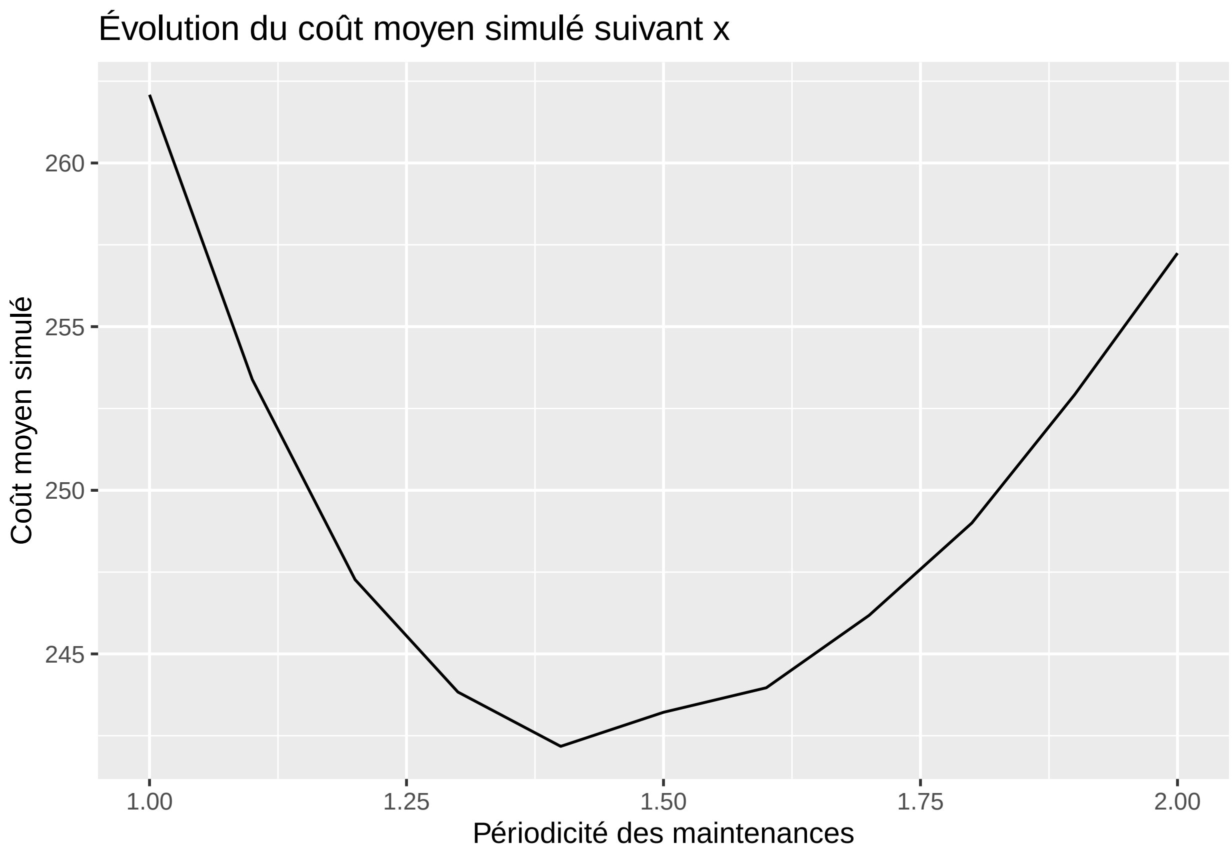
```r
plot_mean_cost_evolution <- function(list_x, param, p = 1000) {
  list_simu <- c()
  for (x in list_x) {
    param$x <- x
    list_simu <- c(list_simu,
                   compute_mean_cost_by_time(p = p,
                                             param_sys = param))
  }

  ggplot() +
    geom_line(aes(x = list_x, y = list_simu)) +
    labs(x = "Périodicité des maintenances",
         y = "Coût moyen simulé",
         title = "Évolution du coût moyen simulé suivant x")
}
```

```r
x_values <- seq(1,5,0.2)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les x ans
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```

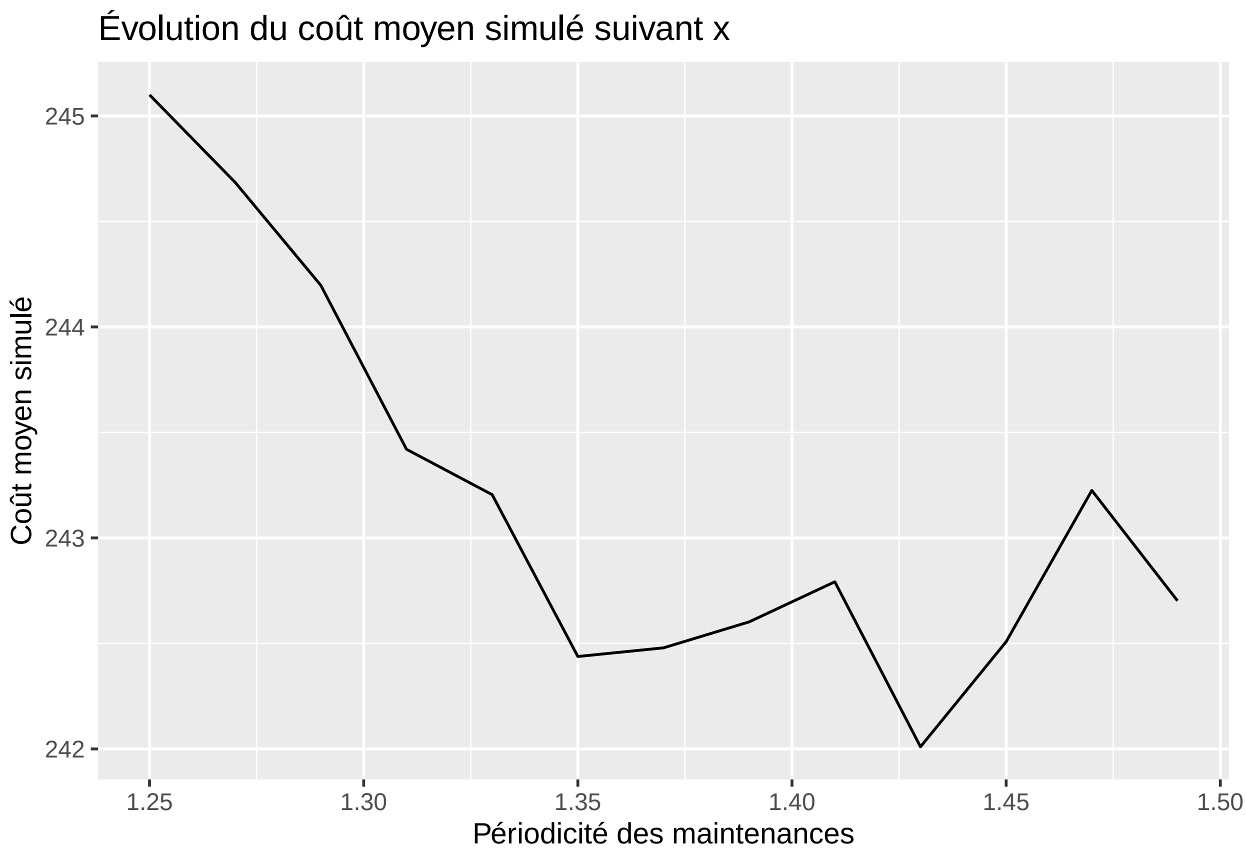


On voit que l'optimum se trouve entre $x = 1an$ et $x = 2ans$. On peut rafiner la simulation.

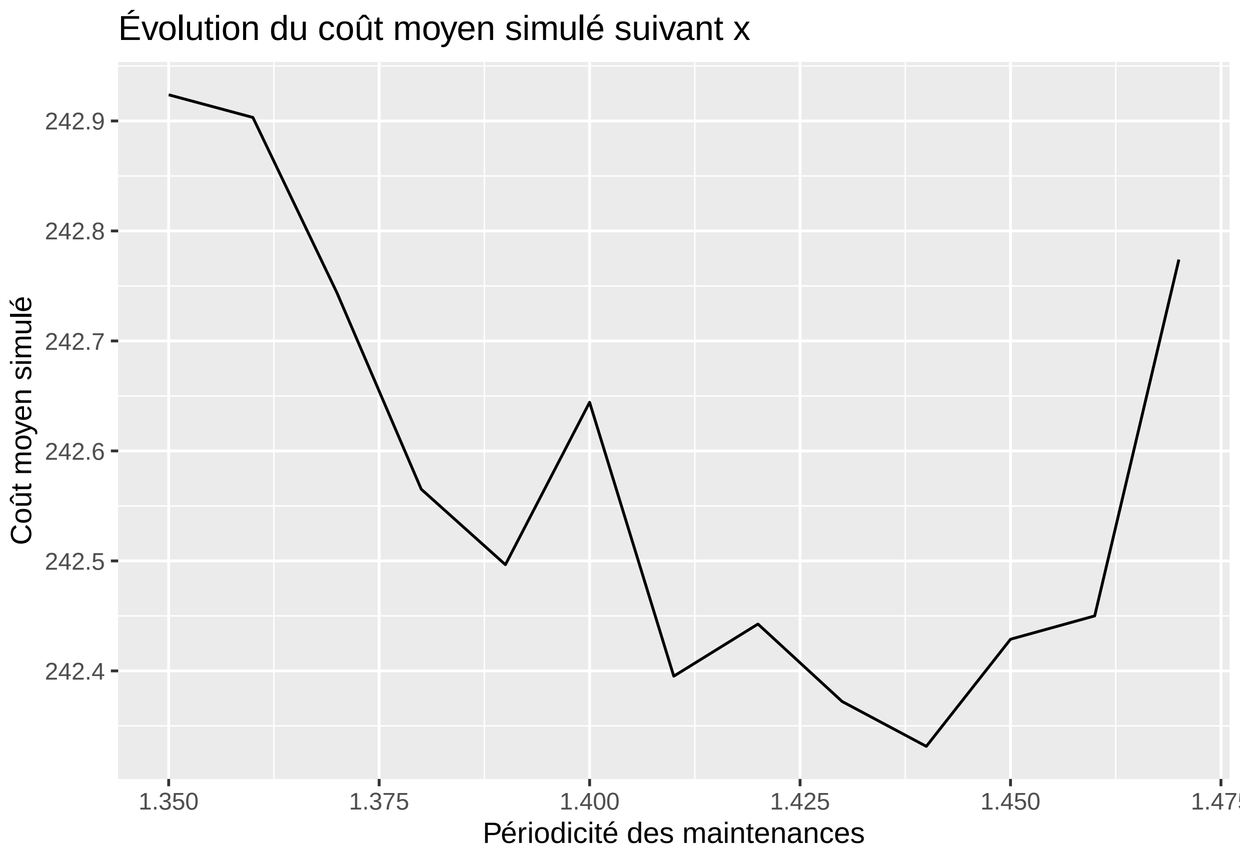
```
x_values <- seq(1,2,0.1) # entre 1 et 2
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les trois ans
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```




```
x_values <- seq(1.25,1.5,0.02) #entre 1.25 et 1.5
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les trois ans
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```




```r
x_values <- seq(1.35,1.47,0.01) # entre 1.35 et 1.47
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les trois ans
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres, p = 5000)
```



Notre x optimal se trouve donc entre 1.350 et 1.475. La variabilité des simulation pourrait être réduite avec d'avantage de simulation (au prix du temps de calcul).

On peut donc conclure qu'avec ce jeu de paramètre, on minimise le coût total en choisissant un $x = 1.45ans$.

```
compute_mean_cost_by_time(p = 10000,
                          param_sys = list(H=1000,
                                           x = 1.45,
                                           scale = 3, # eta en jours
                                           shape = 2.5, # sans unités
                                           cmc = 1000, # euros
                                           cmp = 200)) # euros
```

```
## [1] 242.3686
```

On a dans ce cas un coût moyen optimal de 242€ par an.

Exercice 5, Partie 2.

Présentation du problème

Dans cette partie on étudie un système composé de quatres ampoules en série. On considère les ampoules *iid* suivant une loi de Weibull avec $\eta = 3an$ (paramètre d'échelle) et $\beta = 2.5$ (paramètre de forme).

Dans ce cas on a :

$$R_i(t) = e^{-(\frac{t}{\eta})^\beta}; \forall i \in [1;4] R_{sys}(t) = \prod_{i=1}^{4} R_i(t) = \prod_{i=1}^{4} e^{-(\frac{t}{\eta})^\beta} = e^{-\sum_{i=1}^{4}(\frac{t}{\eta})^\beta} = e^{-4(\frac{t}{\eta})^\beta} = e^{-\left(\frac{t}{4^{-\frac{1}{\beta}}\eta}\right)^\beta}$$

Théroiquement, le système suit une loi de Weibull de paramètre : $\eta_{sys} = 4^{-\frac{1}{\beta}}\eta = \frac{\eta}{4^{-\frac{1}{\beta}}}$ (paramètre d'échelle) et $\beta_{sys} = \beta$ (paramètre de forme).

Dans ce cas on a :

$$MTTF_{sys} = \eta_{sys}\Gamma(1 + \frac{1}{\beta_{sys}}) = \frac{\eta}{4^{-\frac{1}{\beta}}}\Gamma(1 + \frac{1}{\beta})$$

On a aussi le résultat théorique : $T_{sys} = min(T_i|i = 1...4)$ car le système est en série.

Simulation du système

On peut commencer par créer les fonctions permettant de simuler le système et vérifier le résultat théorique trouvé plus haut.

```
# Simulation d'une réalisation de la loi de Weibul de paramètre
# thêta et beta
# remplace la fonction rweibull() de R

realisation_aleatoire_weibull <- function(scale = 3, shape = 2.5) {

    # Simulation d'une réalisation de la loi de Weibul de paramètre
    # thêta et beta
    # remplace la fonction rweibull() de R

    scale * (-log(runif(1)))^(1/shape) %>% return()
}

realisation_panne_système <- function(n = 4, scale = 3, shape = 2.5) {

    # Simulation aléatoire d'une date de panne du système

    replicate(n = n,
              realisation_aleatoire_weibull(scale = scale, shape = shape)) %>%
        min() %>%
        return()
}

realisation_panne_système <- function(n = 4, scale = 3, shape = 2.5) {
    list_simu <- c()
    for (i in 1:n) {
        list_simu <- c(list_simu,
                       realisation_aleatoire_weibull(scale = scale,
                                                     shape = shape))
```

```r
    }
    return(min(list_simu))
}
```

```r
# Calcul théorique du MTTFsys
sprintf("MTTFsys théorique calculé : %f ans",
       4^(-1/2.5) * 3 * gamma(1 + 1/2.5))
```

```
## [1] "MTTFsys théorique calculé : 1.528798 ans"
```

```r
# Simulation d'un MTTF :
sprintf("Une simulation du MTTFsys : %f ans",
        mean(replicate(100000,
                       realisation_panne_système())))
```

```
## [1] "Une simulation du MTTFsys : 1.529806 ans"
```

On trouve des résultat cohérent ce qui me donne confiance dans mon modèle théorique.

Optimisation de la politique de maintenance

On va ensuite appliquer la politique de maintenance suivante :

- Après une panne, une maintenance corrective d'un coût $cmc = 10000£$ est effectuée remettant à neuf l'ensemble du système.
- Si le système fonctionne un temps x sans être maintenu, on effectue une maintenance préventive d'un coût $cmp = 1500£$ qui remet à neuf le système.

On cherche alors le $x*$ qui minimise le coût total.

On déterminera ce $x*$ en utilisant la même logique que dans l'exercice 5, partie 1.

```r
realisation_trajectoire_sys <- function(param, info = FALSE) {

    # Fonction pour simuler une trajectoire du système et
    # retourner le coût total.
    # param est un argument de type list() de la forme :
    # param = list(H,x,scale,shape,cmp,cmc)

    # Attention :
    # H, scale et x sont en jours

    # les dates de maintenance préventives prévues
    echeancier <- param$x * 1:floor(param$H/param$x)
    if (info) {
        #debug
        print("=== l'échéancier de maintenance préventive ===")
        print(echeancier)
        print("=== Début de simulation ===")
    }
    # Le temps courant
    t <- 0

    # Cout final actuel
    Cf <- 0

    i <- 1 # un indice
```

```
    while (t < param$H & i < length(echeancier)) {

        next_panne <- t + realisation_panne_système(n = param$n,
                                                    scale = param$scale,
                                                    shape = param$shape)

        if (info) {
            # debug
            print(paste("Temps courant",t))
            print(paste("Date de la prochaine panne : ",next_panne))
            print(paste(
                "Date de la prochaine maintenance préventive : ",
                echeancier[i]))
        }

        if (next_panne < echeancier[i]) {
            # Dans ce cas la prochaine panne si on laisse vivre le système
            # arrive avant la prochaine maintenance préventive
            # donc on effectue la maintenance corrective

            if (info) {print("MC")}

            # on met à jour le temps courant
            t <- next_panne
            # on effectue une maintenance corrective
            Cf <- Cf + param$cmc
        } else {

            if (info) {print("MP")}

            # on effecute la maintenance préventive avant une survenue de panne
            t <- echeancier[i]
            i = i + 1
            # on effectue une maintenance préventive
            Cf <- Cf + param$cmp
        }

        if (info) {
            print(paste("Temps courant",t))
            print(paste("Coût actuel",Cf))
            print(paste("val de i",i))
            print("---")
        }
    }

    return(Cf)
}

compute_mean_cost_by_time <- function(p = 10000, param_sys) {

    # Calcul un coût moyen par unité de temps Cinf(x)
    # On chaque trajectoire est divisée par H, l'horizon de simulation
```

```
    frame <- c()

    for (i in 1:p) {
        frame <- c(frame,
                   realisation_trajectoire_sys(param = param_sys,
                                               info = FALSE))
    }

    frame <- mean(frame/param_sys$H)
    return(frame)
}

plot_mean_cost_evolution <- function(list_x, param, p = 1000) {
  list_simu <- c()
  for (x in list_x) {
    param$x <- x
    list_simu <- c(list_simu,
                   compute_mean_cost_by_time(p = p,
                                             param_sys = param))
  }

  ggplot() +
    geom_line(aes(x = list_x, y = list_simu)) +
    labs(x = "Périodicité des maintenances",
         y = "Coût moyen simulé",
         title = "Évolution du coût moyen simulé suivant x")
}
```

On peut commencer par effectuer une simulation de la trajectoire du système :

```
parametres <- list(H = 1000, # on observe sur 1000 ans
                   x = 1, # on effectue une maintenance préventive tous les
                   # ans car on a un MTTF de 1.5 environ
                   n = 4, # nombre de composants
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
realisation_trajectoire_sys(parametres)
```

```
## [1] 3788500
```

On a donc un coût de 3 788 500€, soit un coût moyen de 378,85€ par ans.

On peut ensuite calculer le coût moyen par ans avec ce système et ce jeux de paramètre donné : - une maintenance préventive après 1 ans de fonctionnement, - $cmc = 10000$£ - $cmp = 1500$£ - $n = 4$, 4 composants en série - $\eta = 3$, $\beta = 2.5$

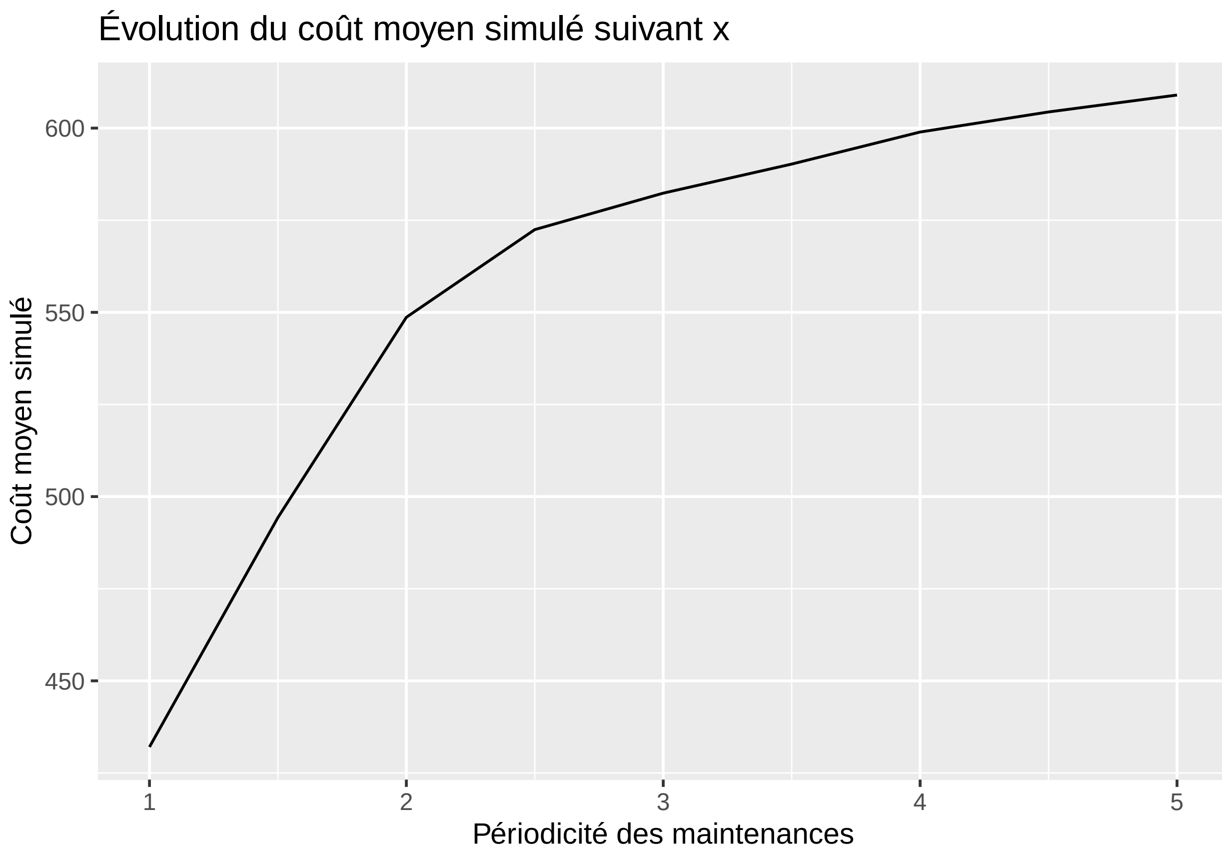
```
sprintf("Coût moyen pour x = %d an : %f",
        parametres$x,
        compute_mean_cost_by_time(p = 1000, parametres))
```

```
## [1] "Coût moyen pour x = 1 an : 3808.160000"
```

Attention, avec $p = 1000$, je ne calcule le coût moyen qu'à partir de 1000 simulation ce qui est peu mais réduit grandement le temps de calcul.

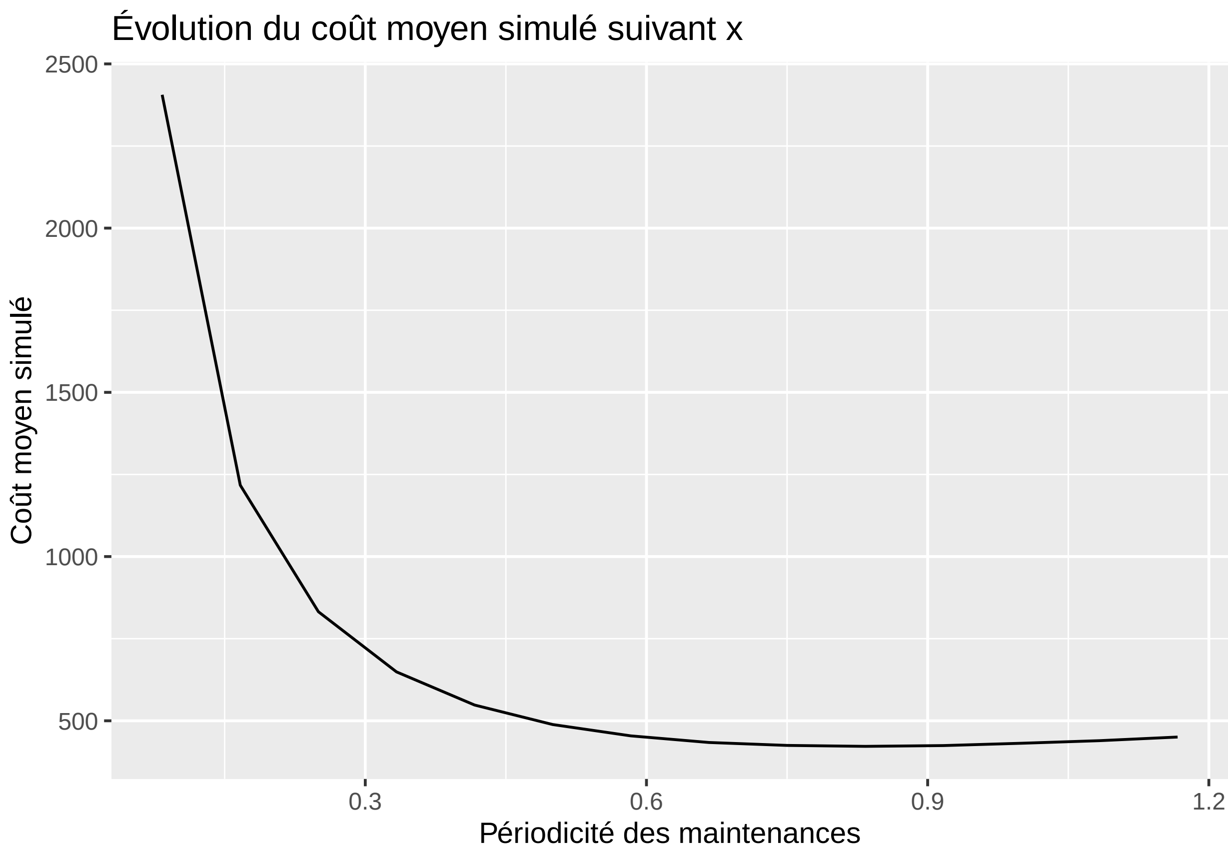
On va ensuite simuler le coût moyen par ans en faisant varier la valeur de x pour trouver graphiquement la valeur optimale $x*$.

```
x_values <- seq(1,5,0.5)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les x ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```



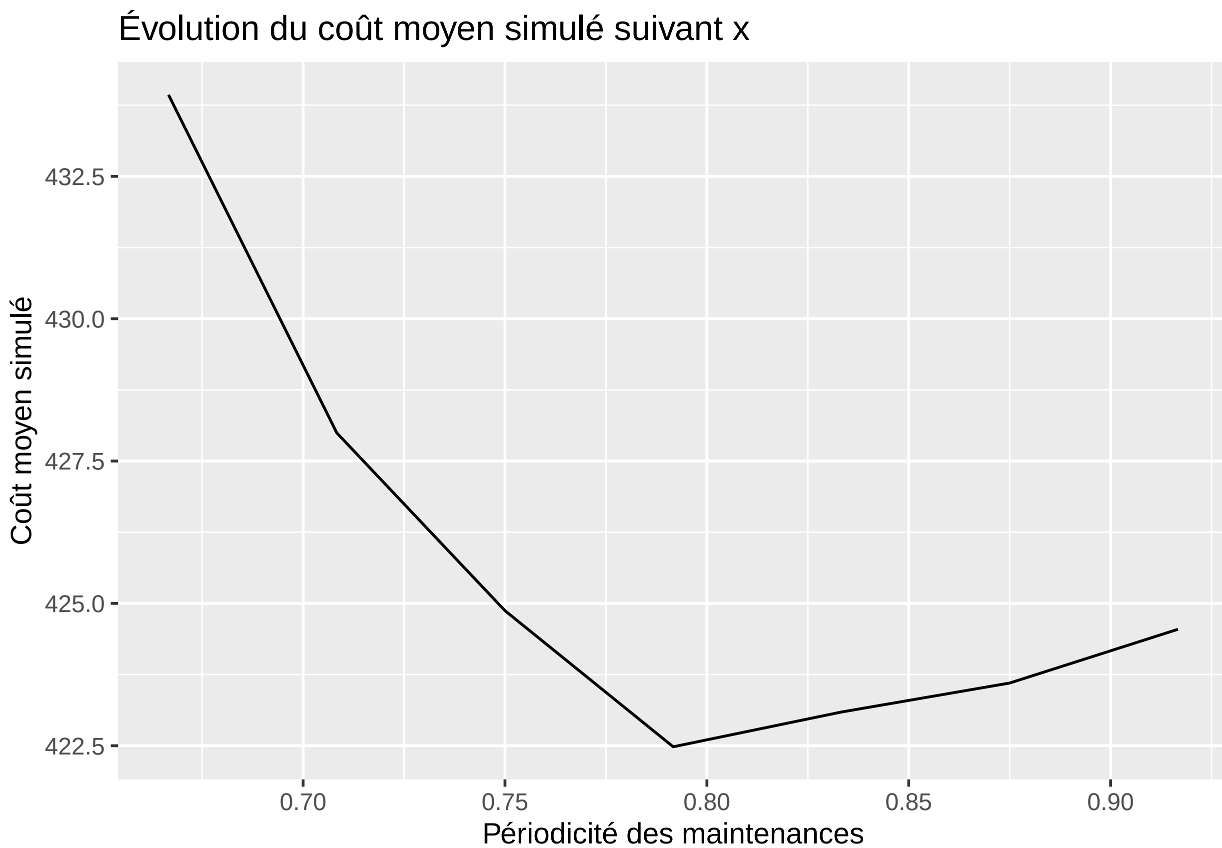
On peut observer sur le graphe que le coût optimal semble se trouver à moins d'un ans

```
x_values <- seq(1/12,14/12,1/12)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les x ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```



Le minimum semble être atteint vers $x^* \simeq 0.8an$ soit $9.6mois$.

```
x_values <- seq(8/12,11/12,1/24)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance
                   #préventive tous les x ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 1000, # euros
                   cmp = 200) # euros
plot_mean_cost_evolution(list_x = x_values, param = parametres)
```



Il semble que la valeur optimale de x est $x^* = 0.875 ans$ soit $10.5 mois$.

Modèle théorique avec maintenance minimale

On se place maintenant dans un cas de maintenance minimale. Quand un composant (une des ampoules) tombe en panne à une date s, il est instantanément remit en marche mais son état est inchangé.

Les composants agissent donc selon une loi de Weibull avec maintenance minimale ce qui correspond à un processus de poisson non Homogène (NHPP).

Après une panne à un instant s on cherche à simuler la prochaine date de panne.

Dans le cas initial pour simuler une date de panne d'un composant, on utilisait une simulation par méthode de la transformée inverse sur F :

$$x_{sim} = F^{-1}(RAND); RAND \sim U[0,1]$$

Dans notre cas on cherche à simuler un temps de fonctionnement avant la panne sachant que la composant à vécu un temps s avant une première panne.

On pourrait appliquer cette logique de la transformée inserve sur F avec la survie $R(t)$. On aurait alors :

$$x_{sim} = R^{-1}(RAND); RAND \sim U[0,1]$$

Le résultat serait le même par symétrie axiale entre R et F.

Si le système si l'un cherche la loi du composant après un temps de vie de s on observe alors $RUL(t,s) = P(T > t + s | T > s)$ puisque avec une loi uniforme sur $[0,1]$, on a $P(X > s) = P(X < 1 - s)$

Pour simuler un nouveau temps de panne après une durée de fonctionnement de s on va alors calculer la fonction inverse du RUL : $RUL(t,s)^{-1}$ (ou la fonction inverse de $(1 - RUL(t,s))$), ce qui est équivalent).

Schéma explicatif

Dans le cas d'un composant de loi de Weibull on a :

$$RUL(t,s) = P(T > t + s | T > s) = \frac{P(T > t + s \cap T > s)}{P(T > s)}$$

$$RUL(t,s) = \frac{e^{-(\frac{t+s}{\eta})^\beta}}{e^{-(\frac{s}{\eta})^\beta}} = e^{(\frac{s}{\eta})^\beta} e^{-(\frac{t+s}{\eta})^\beta}$$

$$RUL^{-1}(t,s) = \eta \left((\frac{s}{\eta})^\beta - ln(t) \right)^{\frac{1}{\beta}} - s$$

On va donc avoir :

$$x_{sim} = RUL^{-1}(RAND, s) = \eta \left((\frac{s}{\eta})^\beta - ln(RAND) \right)^{\frac{1}{\beta}} - s$$

avec x_{sim} une durée de fonctionnement simulée après la réparation minimale effectuée à s.

On peut donc créer une fonction $RUL(t,s)$:

```r
realisation_aleatoire_rul <- function(s, scale = 3, shape = 2.5) {
    return(
        scale * ((s/scale)^(shape)-log(runif(1)))^(1/shape) - s
    )
}
```

On peut alors simuler deux pannes successives d'un composant :

> Attention : la date de la seconde panne est égale à : $s + RUL^{-1}(RAND, s)$ avec s simulée aléatoirement.

```r
s <- realisation_aleatoire_weibull(scale = 3, shape = 2.5)
sprintf("Date première panne : %f ans", s)
```

```
## [1] "Date première panne : 3.232269 ans"
```

```r
sprintf("Date seconde panne : %f ans",
        s + realisation_aleatoire_rul(s, scale = 3, shape = 2.5))
```

```
## [1] "Date seconde panne : 3.770381 ans"
```

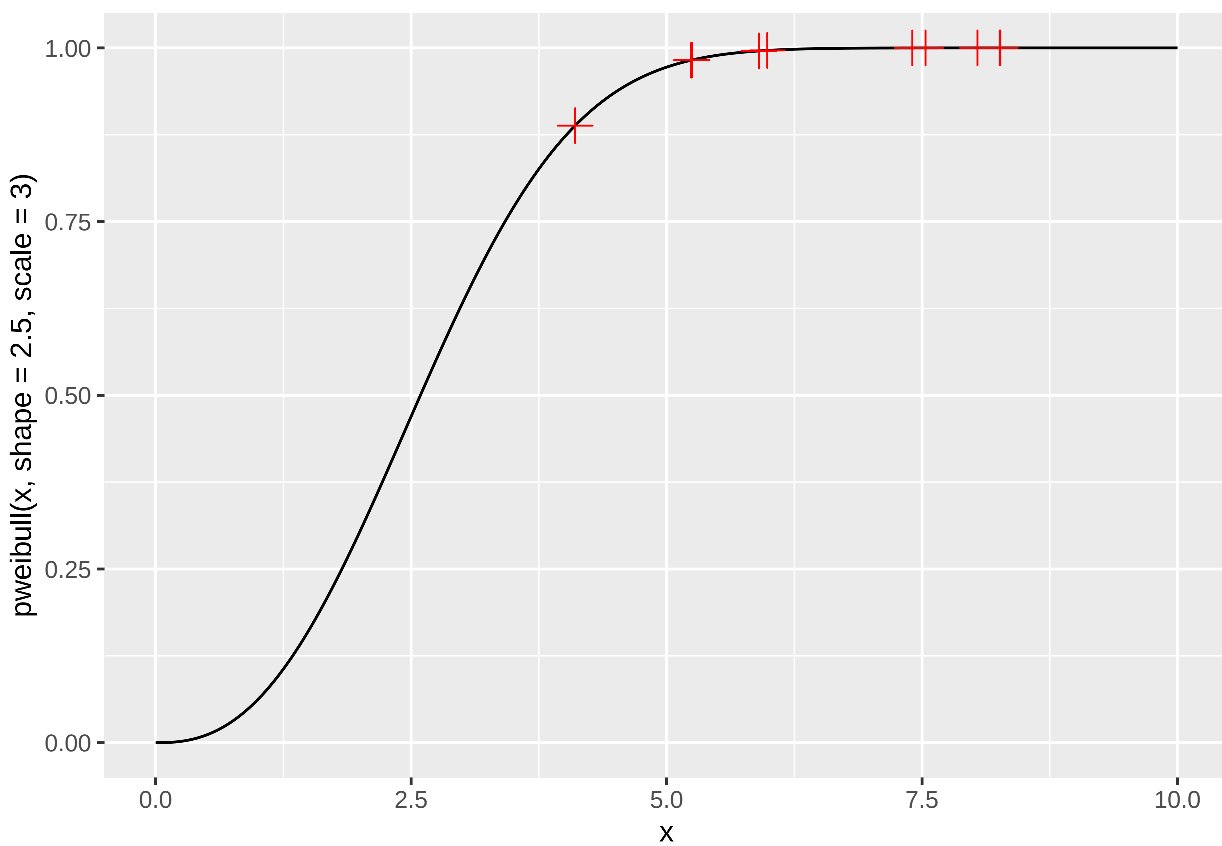
On peut ensuite créer une fonctionne qui simule k panne de suite pour le composant :

```r
realisation_k_panne_composant <- function(k = 2,
                                          scale = 3,
                                          shape = 2.5) {
    if (k == 1) {
        realisation_aleatoire_weibull(scale = scale,
                                      shape = shape) %>%
            return()
    } else {
        dates_pannes <- c(
            realisation_aleatoire_weibull(scale = scale,
                                          shape = shape)
        )
        for (i in 2:k) {
```

```
            dates_pannes <- c(
                dates_pannes,
                dates_pannes[i-1] +
                    realisation_aleatoire_rul(s = dates_pannes[i-1],
                                              scale = scale,
                                              shape = shape)
            )
        }

        return(dates_pannes)
    }
}
```

On peut alors simuler 10 instants de panne successif pour un même composant avec maintenance minimale :

```
pannes <- realisation_k_panne_composant(k = 10, scale = 3, shape = 2.5)

x <- seq(0,10,0.01)
ggplot() +
    geom_line(aes(x,pweibull(x, shape = 2.5, scale = 3))) +
    geom_point(aes(x = pannes,
                   y = pweibull(pannes, shape = 2.5, scale = 3)),
               color = 'red',
               shape = 3,
               size = 4)
```



On voit que les pannes sont de plus en plus fréquentes au fur et à mesure que le système vieillit et il faut le "remettre en route" de plus en plus fréquemment.

Nouvelle politique de maintenance

Dans cette politique de maintenace, la maintenance corrective ne remplace que l'ampoule défectueuse.

Dans ce cas on effectue une maintenance préventive que si **aucune ampoule** ne tombe en panne pendant une période x.

Dans ce cas il faut changer en partie la technique de simulation. Il faut modifier la fonction `realisation_trajectoire_sys`

```
realisation_trajectoire_sys_2 <- function(param, info = FALSE) {

    # Le temps courant
    t <- 0
    # Cout final actuel
    Cf <- 0
    i <- 1 # un indice

    if (info) {
        print("=== Début de simulation ===")
    }

    # On simule les dates de panne des composants
    pannes_composants <- replicate(
        param$n,
        t + realisation_aleatoire_weibull(param$scale,
                                          param$shape)
    )

    # la prochaine date de maintenance prev
    date_maintenance_prev <- t + param$x

    if (info) {
        # debug
        print(paste("Temps courant",t))
        print(paste("Date de la prochaine panne : ",
                    pannes_composants))
        print(paste(
            "Date de la prochaine maintenance préventive : ",
            date_maintenance_prev))
    }

    while (min(pannes_composants,date_maintenance_prev) < param$H) {

        # Tant que l'on a un évènement à traiter avant la fin de l'horizon de
        # simulation

        if (min(pannes_composants) < date_maintenance_prev) {

            # Le premier évènement à venir est une maintenance corrective
            # Sur l'un des composants

            # On se place à la date de l'évènement
```

```
            t <- min(pannes_composants)
            # On effectue la maintenance corrective
            Cf <- Cf + param$cmc

            # On remet à neuf le composant on change donc sa prochaine date de
            # panne
            pannes_composants[which.min(pannes_composants)] <- t +
               realisation_aleatoire_weibull(param$scale,
                                            param$shape)

            # On met à jour la prochaine date de maintenance préventive
            date_maintenance_prev <- t + param$x

            if (info) {print("Maintenance corrective")}

        } else {

            # Dans ce cas on fait une maintenance préventive
            t <- date_maintenance_prev
            Cf <- Cf + param$cmp
            date_maintenance_prev <- t + param$x

            if (info) {print("Maintenance préentive")}

        }

        if (info) {
            print(paste("Temps courant",t))
            print(paste("Coût actuel",Cf))
            print(paste("val de i",i))
            print("---")
        }
    }

    return(Cf)
}
```

```
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 3, # on effectue une maintenance préventive tous les
                   #trois ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
realisation_trajectoire_sys_2(parametres, info = FALSE)
```

```
## [1] 14900000
```

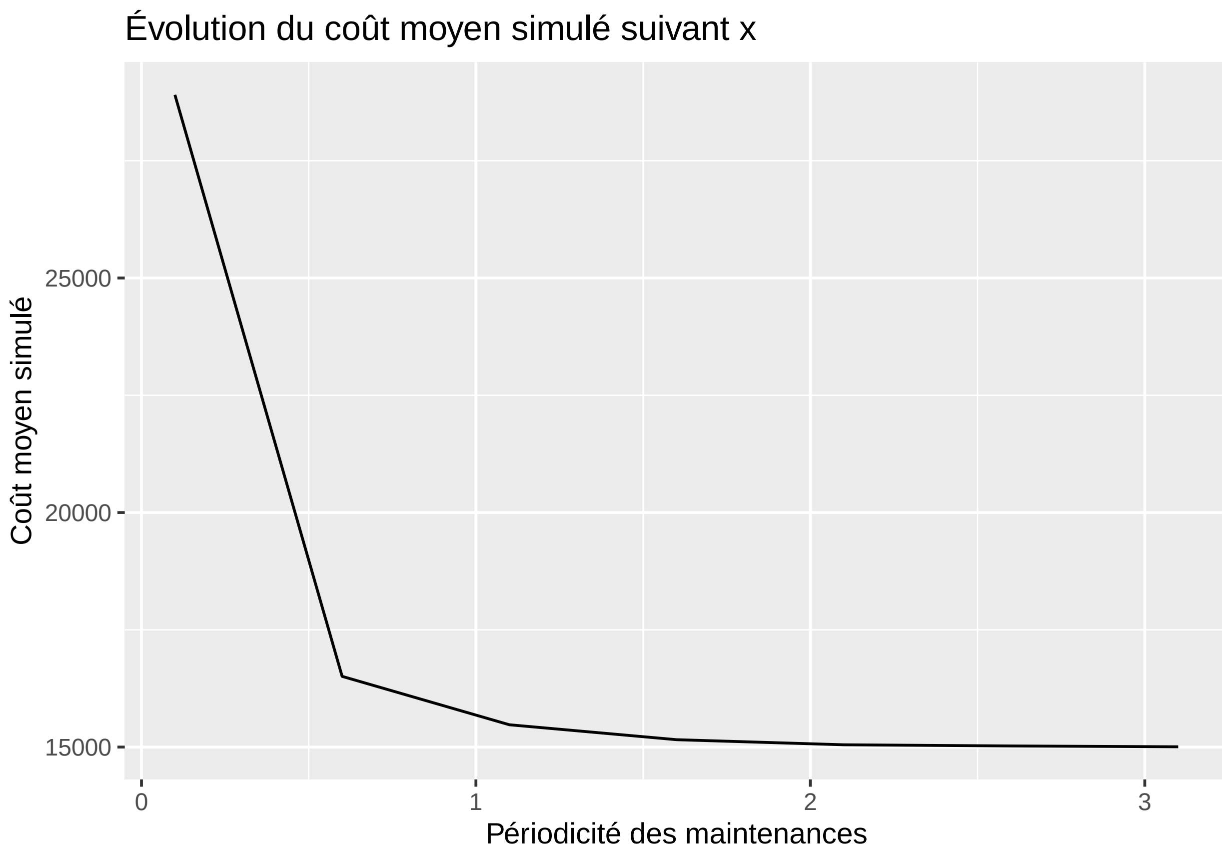
Dans ce cas on a un coût total de 15041500€.

On crée de nouvelles fonctions pour calculer le coût moyen et afficher les courbes.

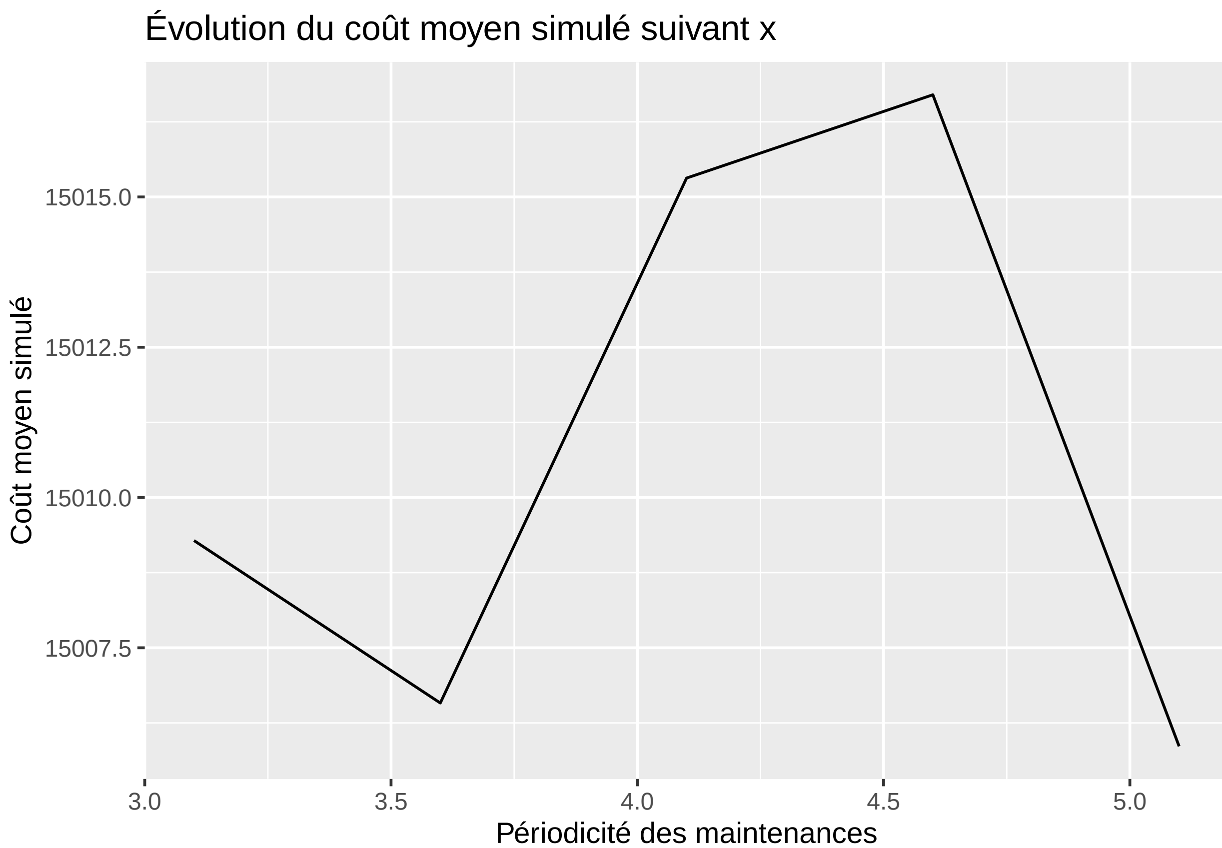
```
compute_mean_cost_by_time_2 <- function(p = 10000, param_sys, info = FALSE) {
```

```r
    # Calcul un coût moyen par unité de temps Cinf(x)
    # On chaque trajectoire est divisée par H, l'horizon de simulation

    frame <- c()

    for (i in 1:p) {
        frame <- c(frame,
                   realisation_trajectoire_sys_2(param = param_sys,
                                                 info = info))
    }

    frame <- mean(frame/param_sys$H)
    return(frame)
}

plot_mean_cost_evolution_2 <- function(list_x, param, p = 1000, info = FALSE) {
  list_simu <- c()
  for (x in list_x) {
    param$x <- x
    list_simu <- c(list_simu,
                   compute_mean_cost_by_time_2(p = p,
                                               param_sys = param,
                                               info = info))
  }

  ggplot() +
    geom_line(aes(x = list_x, y = list_simu)) +
    labs(x = "Périodicité des maintenances",
         y = "Coût moyen simulé",
         title = "Évolution du coût moyen simulé suivant x")
}
```

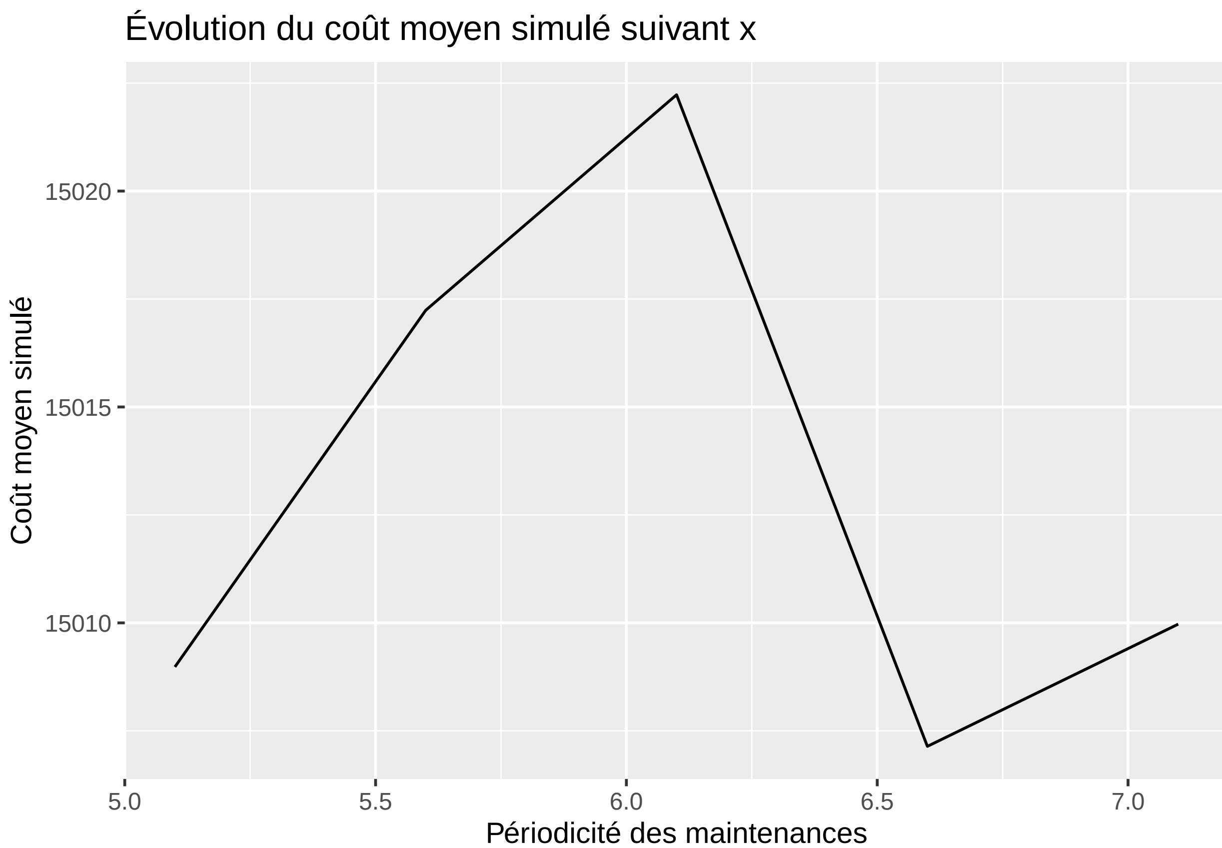
```r
x_values <- seq(0.1,3.1,0.5)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance préventive tous les
                   #trois ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
plot_mean_cost_evolution_2(list_x = x_values, param = parametres, info = FALSE)
```




```
x_values <- seq(3.1,5.1,0.5)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance préventive tous les
                   #trois ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
plot_mean_cost_evolution_2(list_x = x_values, param = parametres, info = FALSE)
```




```r
x_values <- seq(5.1,7.1,0.5)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance préventive tous les
                   #trois ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
plot_mean_cost_evolution_2(list_x = x_values, param = parametres, info = FALSE)
```

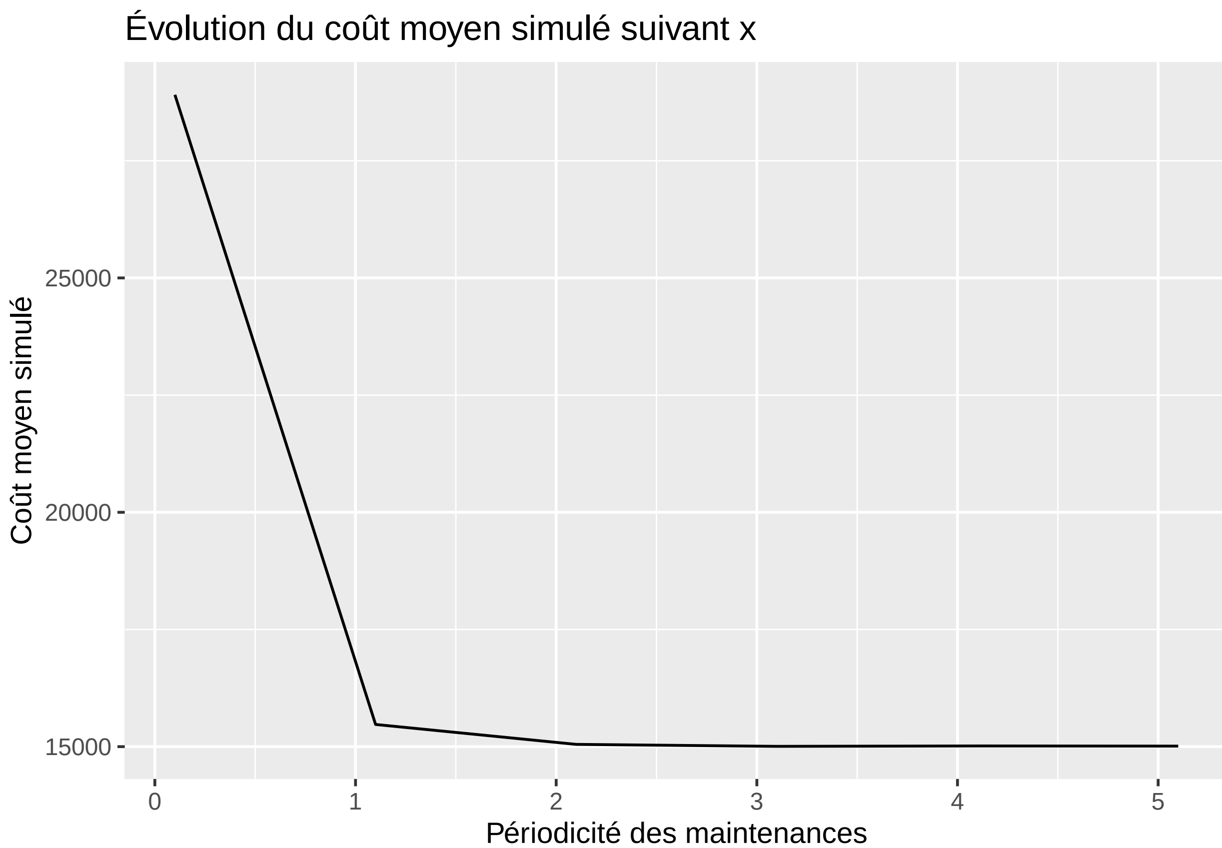


Il semble que les variations de coûts deviennent minimes une fois passé $x > 3$ et que le prix moyen se stabilise à environ 15000€ pas ans.

Donc, on peut choisir n'importe qu'elle valeur de x.

Ce résultat signifie qu'avec cette nouvelle stratégie, il n'y a pas d'intérêt à effectuer de maintenance préventive.

```
x_values <- seq(0.1,5.1,1)
parametres <- list(H = 1000, # on observe sur 1002 ans
                   x = 0, # on effectue une maintenance préventive tous les
                   #trois ans
                   n = 4,
                   scale = 3, # eta en jours
                   shape = 2.5, # sans unités
                   cmc = 10000, # euros
                   cmp = 1500) # euros
plot_mean_cost_evolution_2(list_x = x_values, param = parametres, info = FALSE)
```


Évolution du coût moyen simulé suivant x
Coût moyen simulé
25000
20000
15000
0
1
2
3
4
5
Périodicité des maintenances

Exercice 6

Les données pour cet exercice sont stockées dans le fichier `RM04data.csv`. Commençons par charger ces données. Nous ne prennons que les lignes 5 et 6.

```r
# La fonction read_csv2 est issue du package readr qui permet des imports
# exports facilités dans R
df <- read.csv("../data/RM04data.csv", sep = ',', header = FALSE)
df <- df[5:6,]

columns_names = c("xi","ui")
observations_labels <- paste('x',as.character(1:200),sep = '')

rownames(df) <- columns_names
colnames(df) <- observations_labels

df <- t(df)

head(df)
```

```
##              xi ui
## x1 0.003407508  1
## x2 0.014933541  1
## x3 0.009207314  1
## x4 0.650709545  1
## x5 0.486458950  1
## x6 0.017409099  0
```

Dans ce jeu de données, on a pour chaque observations (200) :

- la durée inter-panne : x_i,
- le type de maintenance : $u_i = \{0,1\}$ avec $u_i = 1$: Maintenance préventive et $u_i = 0$: Maintenance corrective.

On suppose que les durées inter-pannes suivent une loi exponentielle avec une maintenance parfaite.

Vérification de l'hypothèse exponentielle

Si les maintenance sont parfaite, on observera une absence de tendance sur les durées inter-panne.

Dans ce cas les X_i sont *iid*.

On peut vérifier cette hypothèse avec un test de Spearman.

Le test se présente ainsi :

$$\begin{cases} H_0 : \text{les données ne présentent pas de tendance} \\ H_1 : \text{les données présentent une tendance} \end{cases}$$

On calcul la statistique de test :

$$T = \sqrt{n-2}\frac{\rho_s}{\sqrt{1-\rho_s^2}} \sim St(n-2)\rho_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n^3 - n}$$

On va refuser la présence de tendance si

$$|T| < F^{-1}_{St(n-2)}\left(1 - \frac{\alpha}{2}\right)$$

Avec : $d_i = i - R_i$, et R_i le rang des observations

On peut d'abord construire les vecteurs R_i, i, et d_i.

```
n <- length(df[,"xi"])
i <- 1:n
Ri <- match(df[,"xi"], sort(df[,"xi"]))
di <- i - Ri
```

On calcul ensuite le statistique de test

```
ps <- 1 - (6*sum(di^2))/(n^3-3)
T <- sqrt(n-2) * ((ps)/(sqrt(1-ps^2)))
sprintf("Valeur de la statistique de test : %f", T)
```

```
## [1] "Valeur de la statistique de test : -0.630558"
```

On regarde le quantile de la loi de Student :

```
qt(1-0.05/2,n-2)
```

```
## [1] 1.972017
```

Dans notre cas on a : $|T| < F^{-1}_{St(n-2)}\left(1 - \frac{\alpha}{2}\right)$, donc on va rejeter la présence d'une tendance, conformément à l'hypothèse initiale.

On peut ensuite tester l'hypothèse de la loi exponentielle avec un test de Bartlett.

```
compute_bartlett_stat <- function(tn, xi) {

  # Fonction qui calcule la statistique de test du test de Bartlett

  n <- length(xi)

  2*n*(log(tn/n)-sum(log(xi))/n)/(1+((n+1)/(6*n))) %>% return()
}
```

On ne dispose que des x_i il faut donc calculer les t_i :

```
xi <- df[,"xi"]
ti <- cumsum(xi)
```

```
compute_bartlett_stat(as.vector(ti[n]), xi)
```

```
## [1] 237.8812
```

```
qchisq(0.05/2, n-1)
```

```
## [1] 161.8262
```

```
qchisq(1-0.05/2, n-1)
```

```
## [1] 239.9597
```

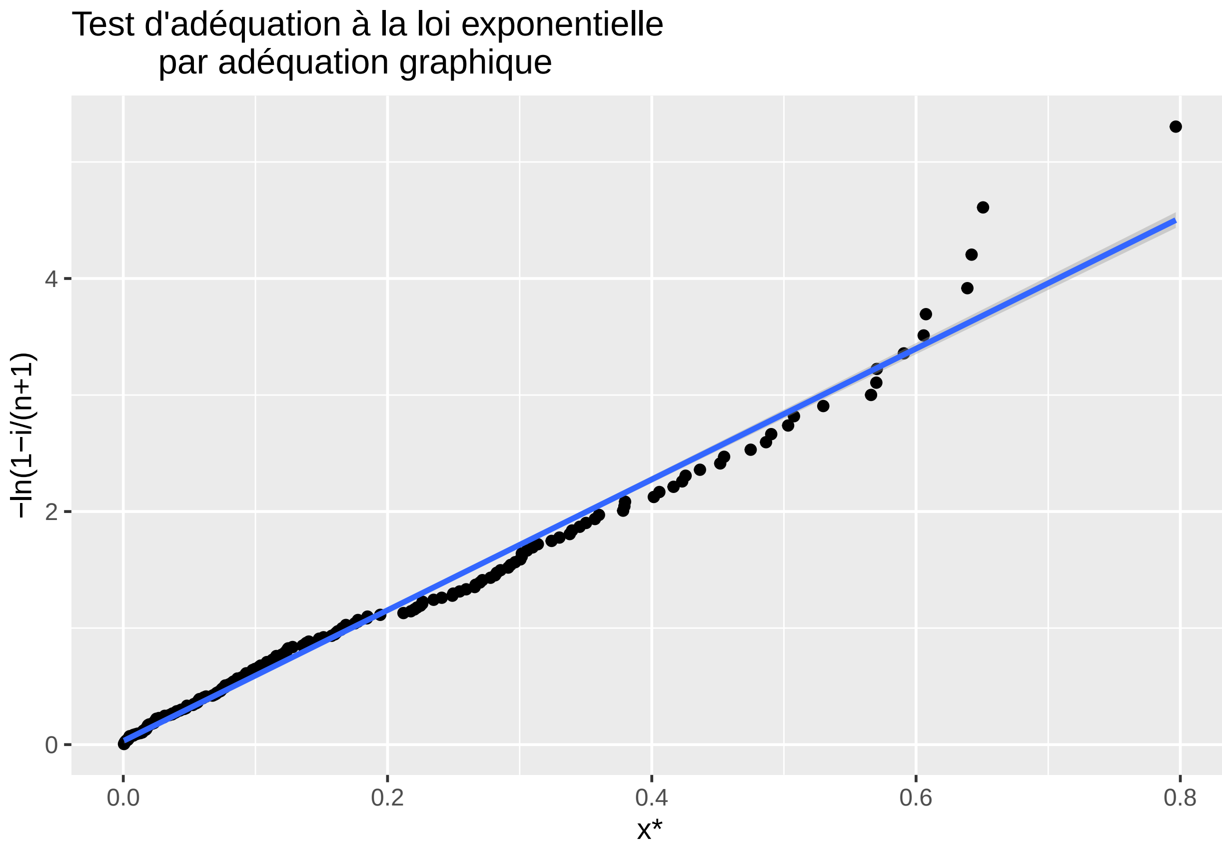
Dans notre cas, on va accepter H_0 de justesse car on a $237.8812 \in [161.8262; 239.9597]$.

On peut vérifier à l'aide d'un test par adéquation graphique :

```
ggplot() +
    geom_point(aes(x = sort(xi), y = -log(1-i/(n+1)))) +
    geom_smooth(aes(x = sort(xi), y = -log(1-i/(n+1))),
                method = lm) +
    labs(x = "x*",
```

```
        y = "-ln(1-i/(n+1)",
        title = "Test d'adéquation à la loi exponentielle
        par adéquation graphique")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



On trouve une droite de régression. On peut chercher les paramètres de la régression linéaire pour déterminer les paramètres de la loi exponentielle.

```
lm(-log(1-i/(n+1)) ~ sort(xi)) %>% summary()
```

```
##
## Call:
## lm(formula = -log(1 - i/(n + 1)) ~ sort(xi))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.20628 -0.08132  0.01726  0.03967  0.92745
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03121    0.01259    2.48    0.014 *
## sort(xi)     5.61157    0.05258  106.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1251 on 198 degrees of freedom
```

```
## Multiple R-squared:  0.9829, Adjusted R-squared:  0.9828
## F-statistic: 1.139e+04 on 1 and 198 DF,  p-value: < 2.2e-16
```

On a $R^2 = 0.9828$, et $p_{value} = 2.2 \times 10^{-16}$, ce qui nous conforte dans l'adéquation des durées inter-pannes à une loi exponentielle.

On a donc une estimation $\lambda_{gph} = 5.61157$.

On peut aussi chercher une estimation par méthode des moments :

```
1/mean(xi)
```

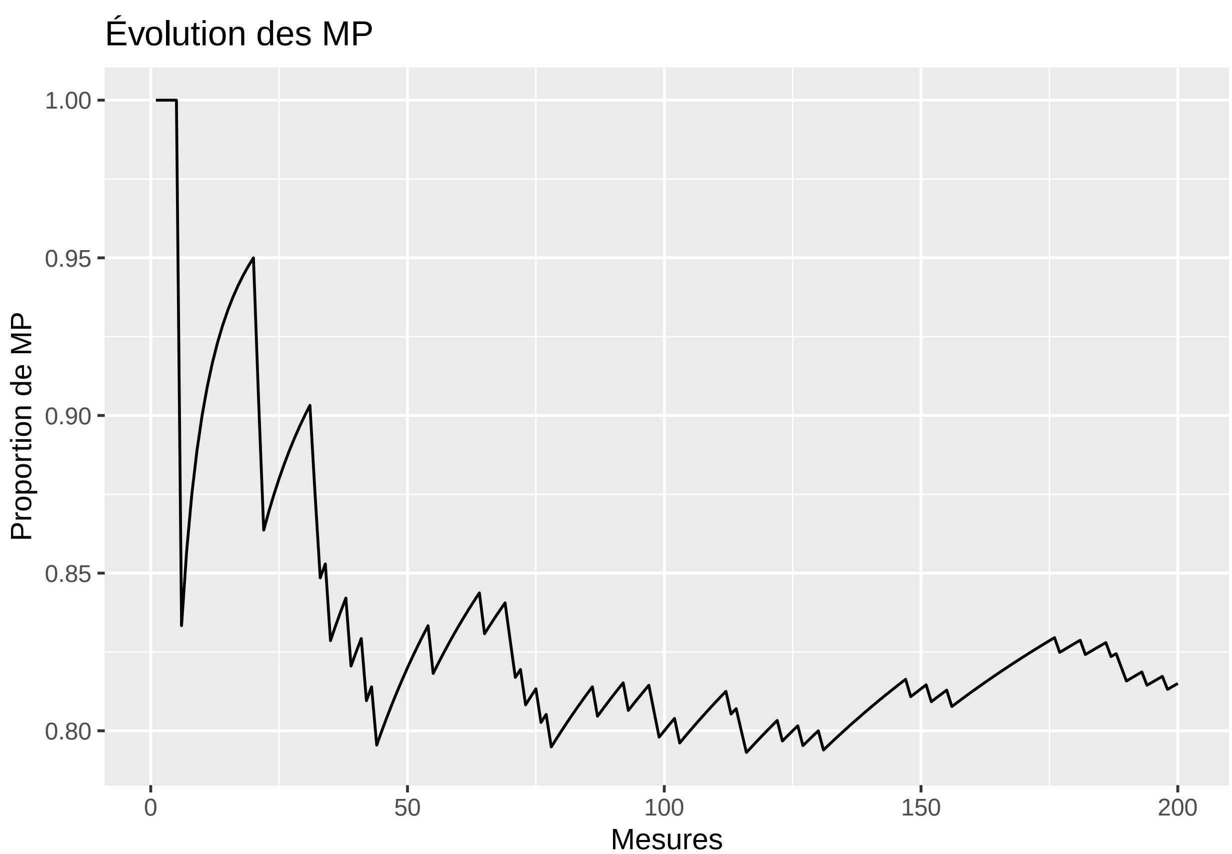
```
## [1] 5.870253
```

Dans ce cas on a : $\lambda_{gph} = 5.870253$.

Donc on peut sans problème conserver l'hypothèse selon laquelle les x_i suivent une loi exponentielle.

On peut tracer la courbe de la proportion de MP :

```
ggplot() +
    geom_line(aes(x = i, y = cumsum(df[,"ui"])/i)) +
    labs(x = "Mesures",
         y = "Proportion de MP",
         title = "Évolution des MP")
```



Après quelques mesure, la proportion de MP semble se stabiliser vers 80%. On effectue donc 80% de maintenances préventives d'après les données.

Politique de maintenance

Après une maintenance, on suppose que la durée jusqu'à une MP est Y et la durée jusqu'à une MC est Z avec Y et Z indépendants.

Dans ce cas la durée effective jusqu'à la prochaine maintenance est $W = min(Y, Z)$.

On suppose maintenant que $Y \sim Exp(\mu)$ et $Z \sim Exp(\lambda)$.

Dans ce cas on cherche : $P(W \leq w)$:

$$P(w \leq w) = P(W \leq w|z \leq w)P(z \leq w) + P(W \leq w|z > w)P(z > w)$$

Avec :

$P(z \leq w) = 1 - e^{-\lambda w} P(z > w) = e^{-\lambda w} P(W \leq w|z \leq w) = 1$en effet quelque soit la valeur de y alors$min(y, z) < w$ si $z \leq wP$

Dans ce cas :

$$P(W \leq w) = (1 - e^{-\lambda w}) + (1 - e^{-\mu w})e^{-\lambda w} = 1 - e^{-(\mu+\lambda)w}$$

Donc $W \sim Exp(\mu + \lambda)$ d'espérance $\frac{1}{\mu+\lambda}$

On cherche ensuite la probabilité π que la prochaine maintenance soit une MP :

$$\begin{aligned}
&\pi = P(y < z) = \int_{z \in \Omega_Z} F_Y(z) f_Z(z) dz \\
&= \int_{z \in \Omega_Z} (1 - e^{-\mu z})\lambda e^{-\lambda z} dz = \int_{z \in \Omega_Z} \lambda e^{-\lambda z} - \lambda e^{-\lambda z} e^{-\mu z} \\
&= \frac{\lambda}{\lambda} - \frac{\lambda}{\lambda+\mu} = 1 - \frac{\lambda}{\lambda+\mu} = \frac{\mu}{\lambda+\mu}
\end{aligned}$$

Dans notre jeu de données, la durée moyenne inter-pannes observée est de :

```
sprintf("Moyenne de durée inter-pannes : %f", mean(xi))
```

```
## [1] "Moyenne de durée inter-pannes : 0.170350"
```

Avec une proportion de MP de :

```
sprintf("Proportion de MP : %f", sum(df[,"ui"])/n)
```

```
## [1] "Proportion de MP : 0.815000"
```

Dans ce cas on a : $\pi = 0.815000$ et $E[W] = 0.170350$, soit :

$$\begin{cases} \pi = \frac{\mu}{\lambda+\mu} \\ E[W] = \frac{1}{\lambda+\mu} \end{cases}$$

Donc : $\pi = \mu * E[W]$ donc $\mu = \frac{\pi}{E[W]}$, et $\lambda = \frac{1}{E[W]}(1 - E[W]\mu)$.

On peut alors estimer les paramètres :

```
sprintf("mu = %f", 0.815000/0.170350)
```

```
## [1] "mu = 4.784268"
```

```
sprintf("lambda = %f", (1-0.170350*0.815000)/0.170350)
```

```
## [1] "lambda = 5.055267"
```

Donc on a $Y \sim Exp(4.784268)$ et $Z \sim Exp(5.055267)$.

Indépendance entre U et W

On va ensuite chercher à montrer que les variables aléatoires U et W sont indépendante. On a :

- U, le type de maintenance avec $P(U=0) = 1-\pi$ et $P(U=1) = \pi$
- W, la durée inter panne, de loi exponentielle de paramètre $\mu+\lambda$.

Deux variables aléatoires A et B sont indépendantes si on a : $P(A=a \cap B=b) = P(A=a)P(B=b), \forall a \in \Omega_A \forall b \in \Omega_B$

On cherche donc à montrer que par exemple $P(W>w|U=1) = P(W>w)P(U=1)$. Avec :

- $P(W>w) = e^{-(\mu+\lambda)w}$
- $P(U=1) = \pi$

$$P(W>w|U=1) = P(min(y,z)>w|y<z) = \int_{z\in\Omega_z} P(y>w|Z=z, y<z) f_Z(z)dz$$

$$\begin{array}{l} z<w, y<z \Rightarrow y<z<w \Rightarrow P(y>w|Z=z,y<z) = 0 \\ z>w, y<z \Rightarrow P(y>w|Z=z,y<z) = P(y\in[w,z]) = R_Y(w) - R_Y(z) \end{array}$$

Donc :

$$\begin{array}{ll} P(W>w|U=1) & = \int_{z\in\Omega_z} P(y>w|Z=z,y<z) f_Z(z)dz \\ & = \int_z^w 0 f_Z(z)dz + \int_w^{+\infty} (R_Y(w) - R_Y(z)) f_Z(z)dz \\ & = \int_w^{+\infty} \left(e^{-\mu w} - e^{-\mu z}\right) \lambda e^{-\lambda z} dz \\ & = \lambda e^{-\mu w} \int_w^{+\infty} e^{-\lambda z} dz - \lambda \int_w^{+\infty} e^{-(\mu+\lambda)z} dz \\ & = e^{-\mu w} e^{-\lambda w} - e^{-(\mu+\lambda)w} \frac{\lambda}{\lambda+\mu} = \frac{\mu}{\lambda+\mu} e^{-(\mu+\lambda)w} \\ & = P(U=1)P(W>w) \end{array}$$

Donc, la probabilité d'avoir une maintenance préventive est indépendante de la valeur de W.