Projet-RM04

Baptiste Toussaint

2024-04-18

Contents

Exercice 2

```
# 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).

```
# 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)}
```

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 $\lambda = 0.1$.

```
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 $\lambda = 0.1$:

```
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.

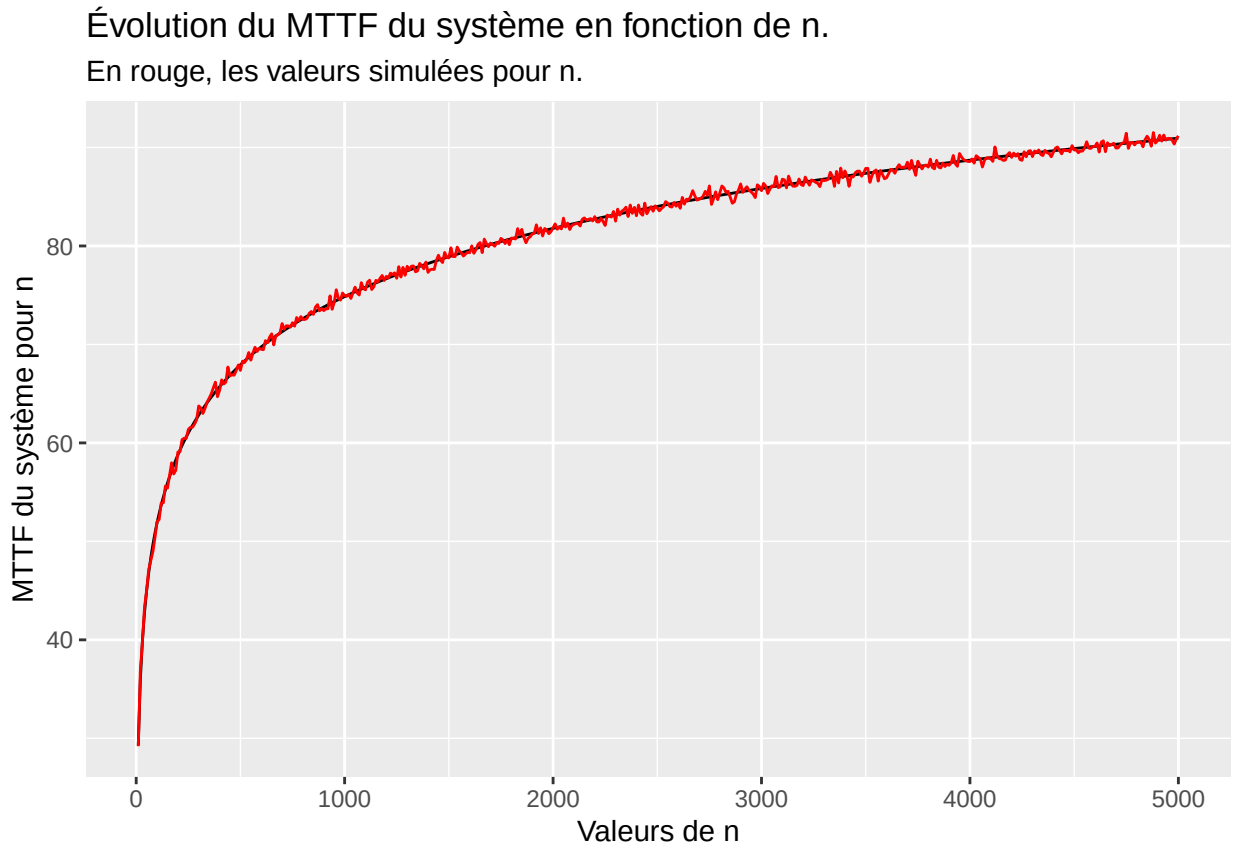
Pour trouver le $MTTF$ par simulation, il suffit de répliquer la fonction : `exe2_simu_panne_systeme` p fois (avec p relativement grand) et de prendre la moyenne des simulations réalisées, ce que fait la fonction `exe2_simu_mttf_system`.

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

```
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 :

```
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.")
```



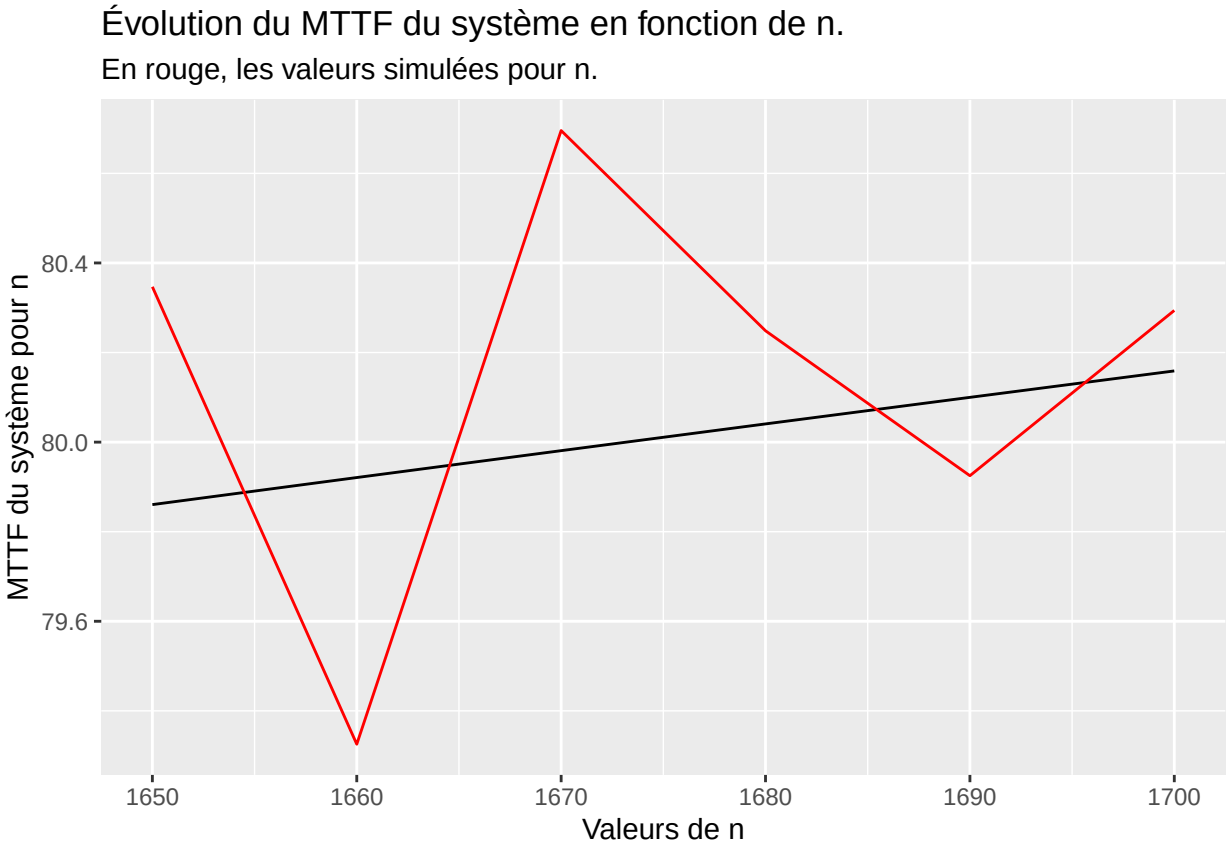
On observe donc que les valeurs simulés sont bel et bien proches des valeurs théoriques.

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 :

```
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 $MTTF_{sys}$.

```
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

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

# nombre de réplications
p = 10000

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()

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"
```

Système avec B en redondance uniquement

On a donc un système série/parallèle. 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 :

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

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 :

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

```r
(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 :

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

Par la simulation on trouve :

```
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()
}
```

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

sprintf(message_mttf_simule_a, simulation)
```

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

```
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 :

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

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 :

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

```
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 $MTTF$:

```
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 : Simulation

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 :

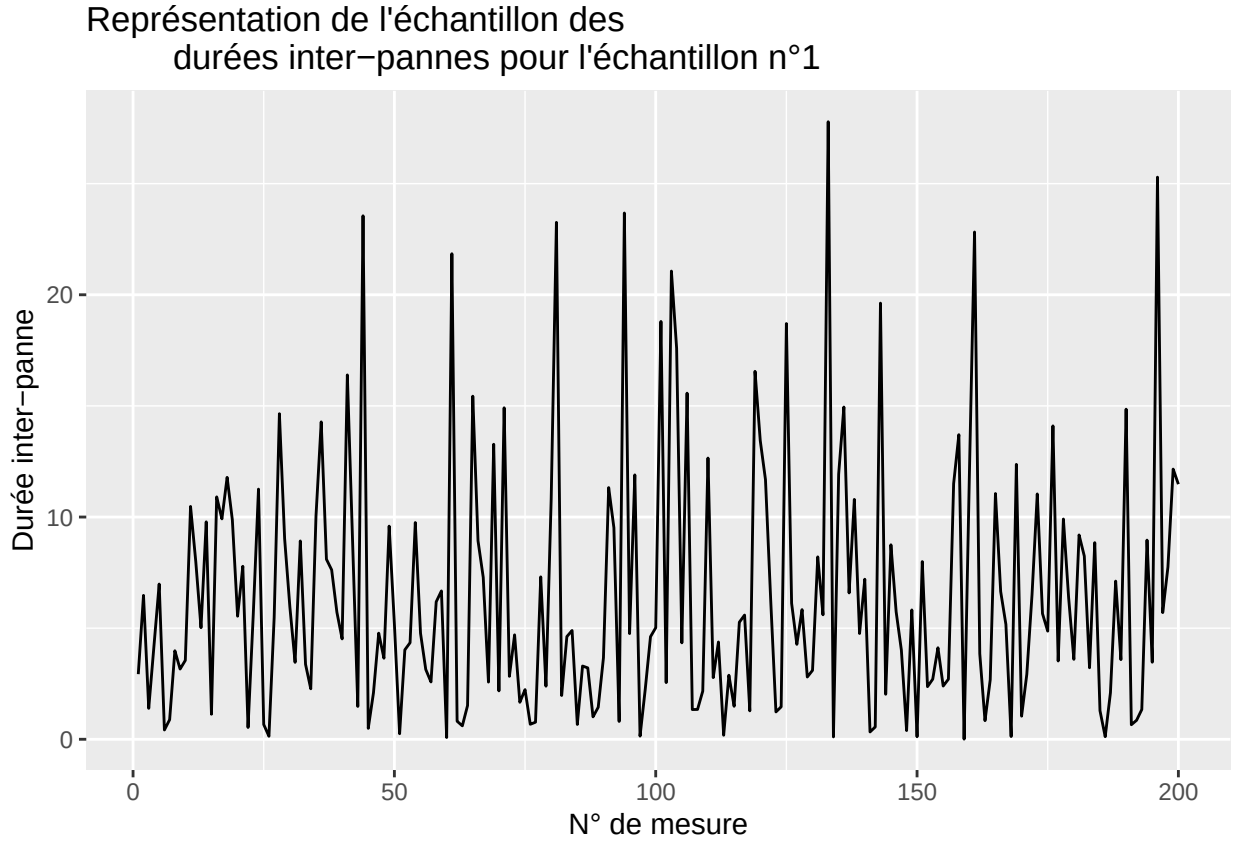
$$\forall i \in [1, n] : T_i = \sum_{k=1}^{i} x_k$$

Échantillon n°1

On peut commencer par tracer les données :

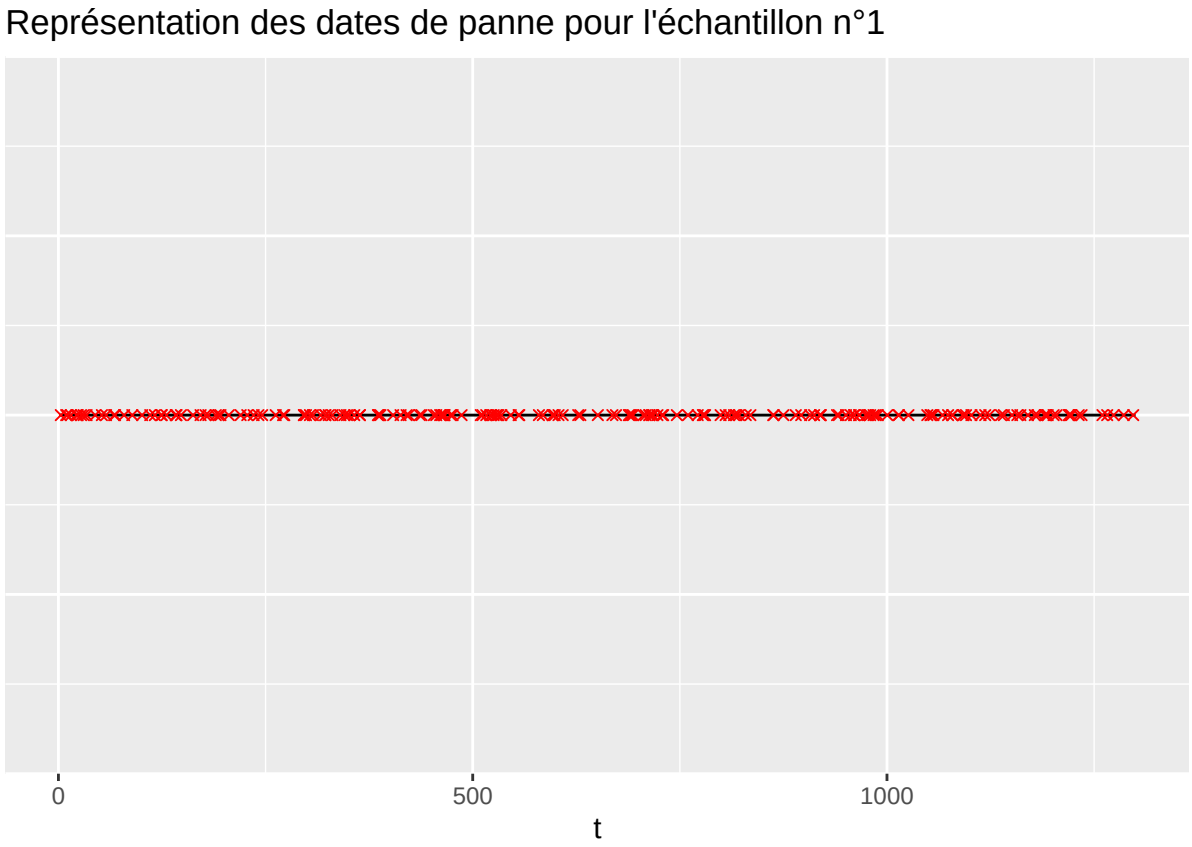
- l'évolution des durées inter-pannes,
- les dates de panne.

```
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 issue 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) {

  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 le quantile de la loi du χ^2 :

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

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

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

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

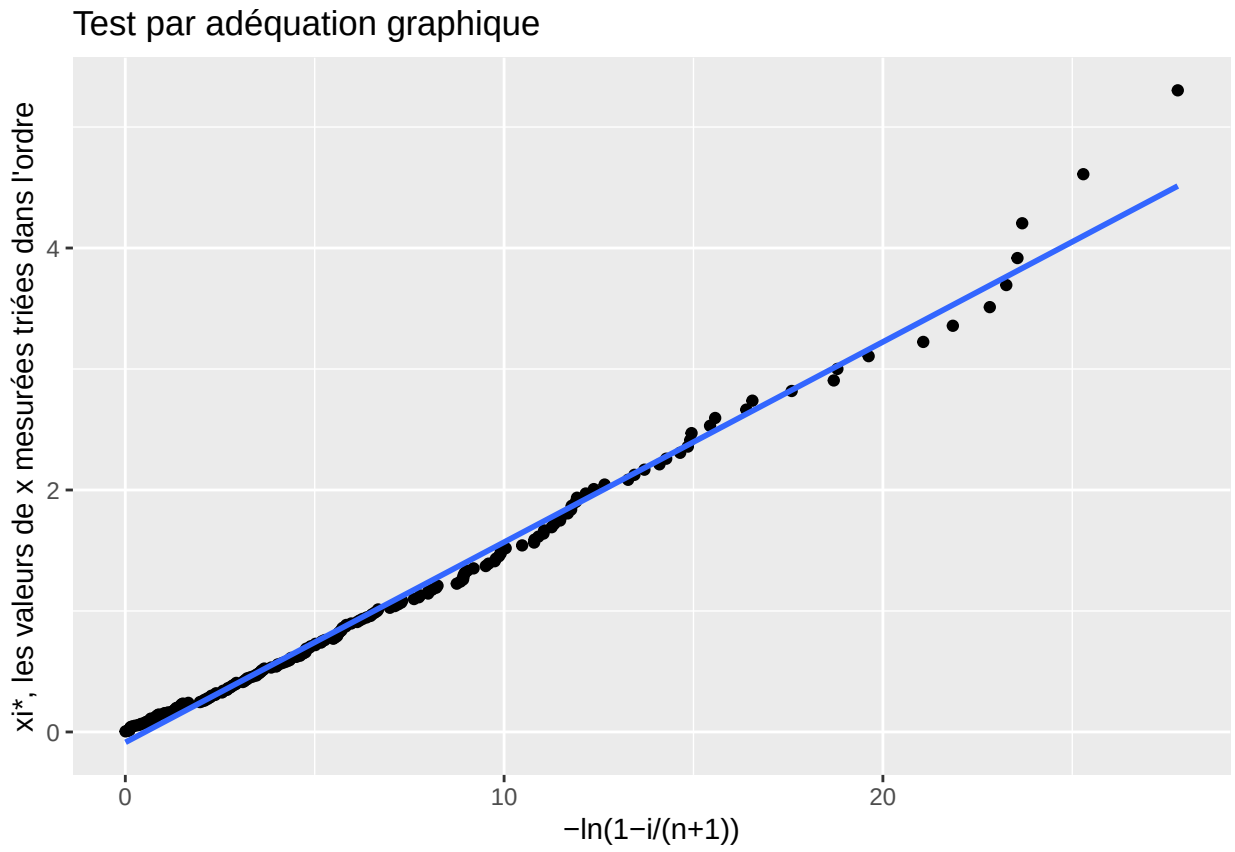
Dans ce cas on ne va pas rejeter H_0 et considérer que l'échantillon est issu d'un processus de poisson homogène (PPH).

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 $\hat{\lambda}_{gph}^{sys1}$

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

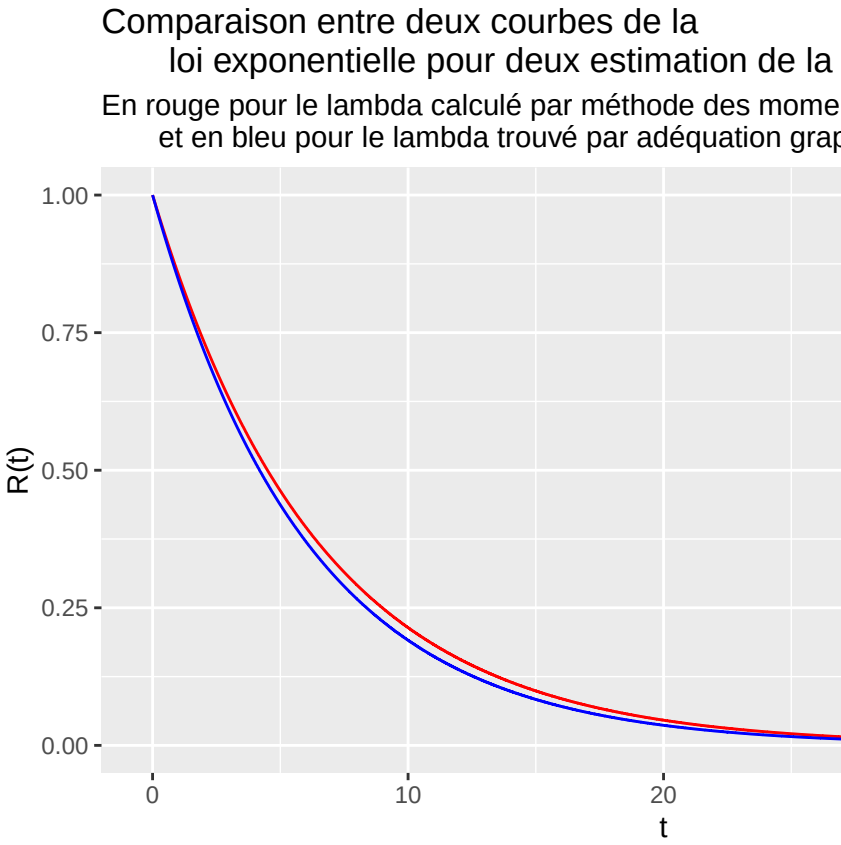
```
estim_param_sys1_mm = 1/mean(df[,"xi_sys1"])
estim_param_sys1_mm
```

```
## [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.

```
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.")
```



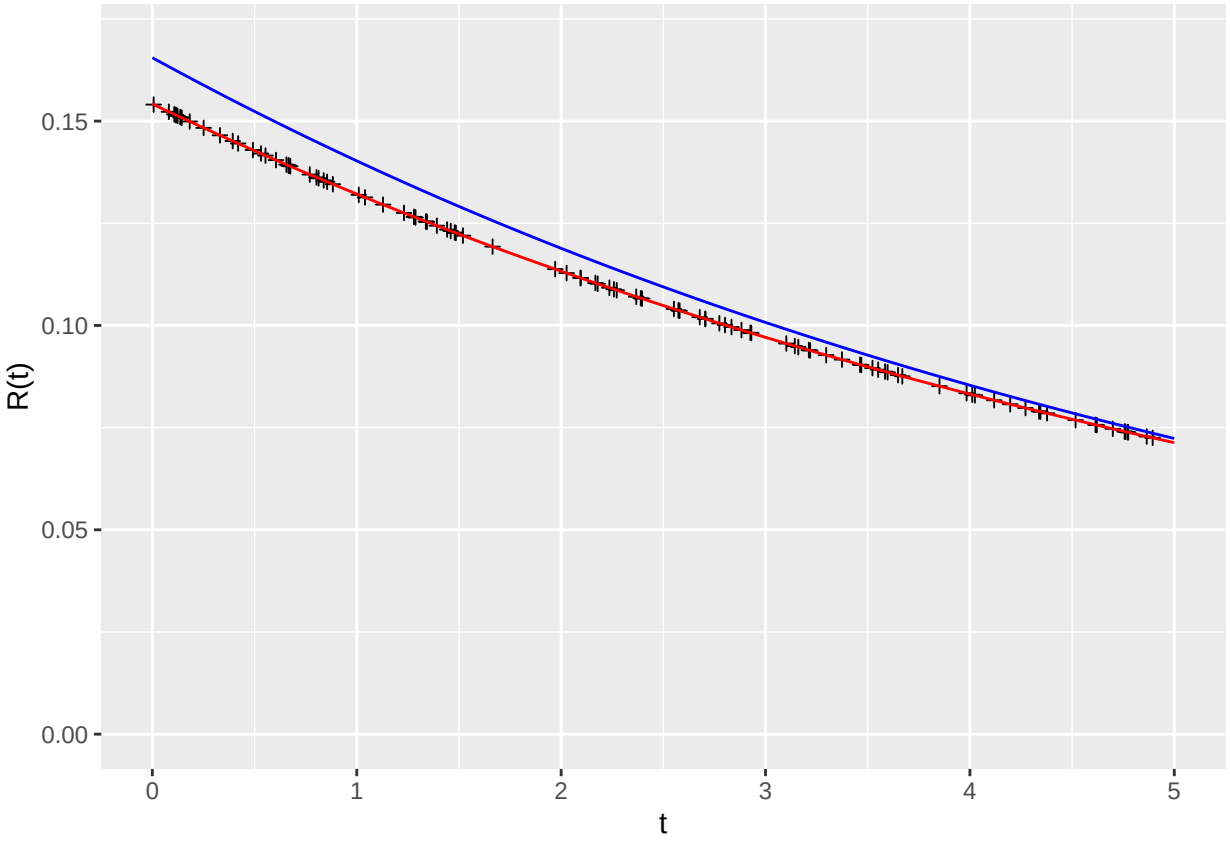
Représentation graphique des deux modèles

On remarque que la courbe rouge semple mieux correspondre aux données.

```
ggplot() +
  geom_point(aes(
      x = df[,"xi_sys1"],
      y = estim_param_sys1_mm*exp(-1*estim_param_sys1_mm*df[,"xi_sys1"])
    ),
    shape = 3,
    color = 'black') +
  geom_line(aes(t, dexp(t,estim_param_sys1_mm)), color = 'red') +
  geom_line(aes(t, dexp(t,estim_param_sys1_gph)), color = 'blue') +
  xlim(0,5) +
  ylim(0,0.17) +
  labs(x = "t",
       y = "R(t)")
```

```
## Warning: Removed 98 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 35000 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 35000 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



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

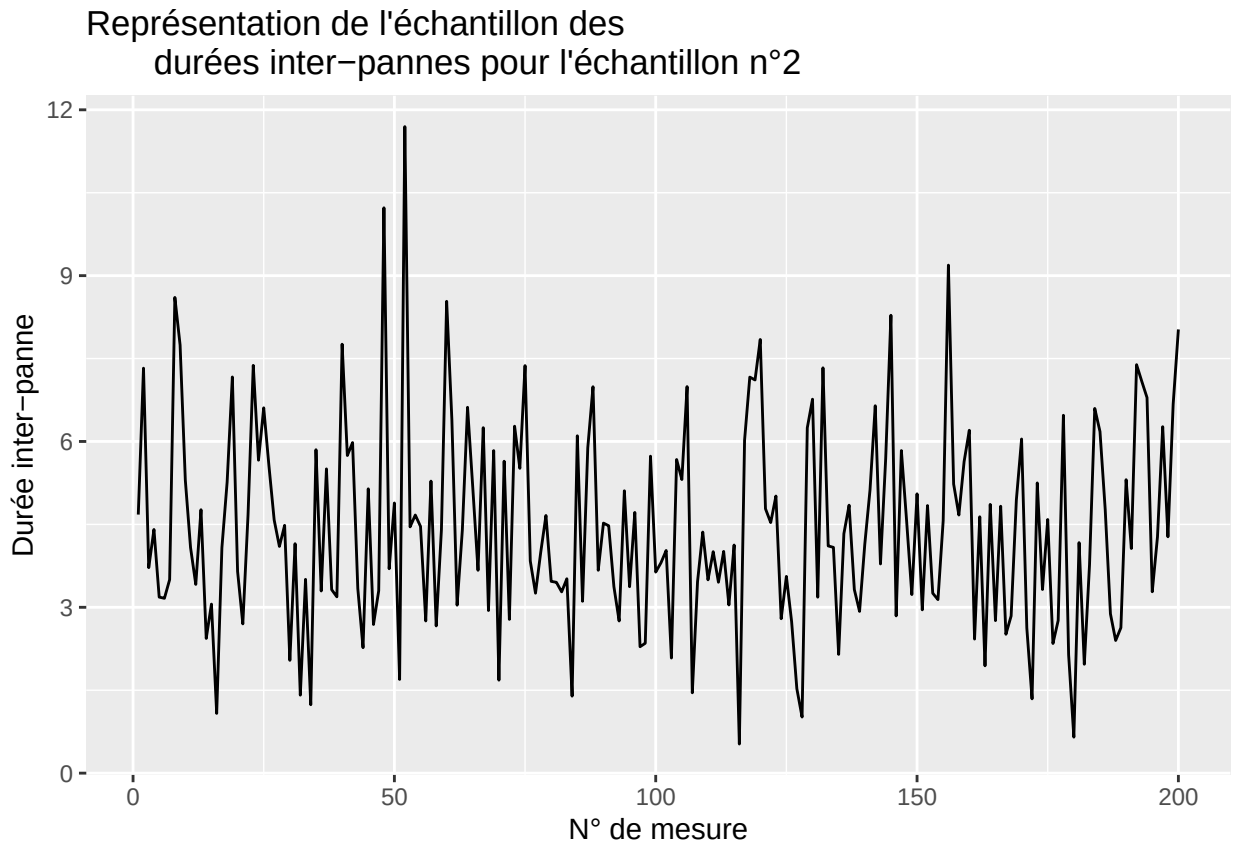
```
1/estim_param_sys1_mm
```

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

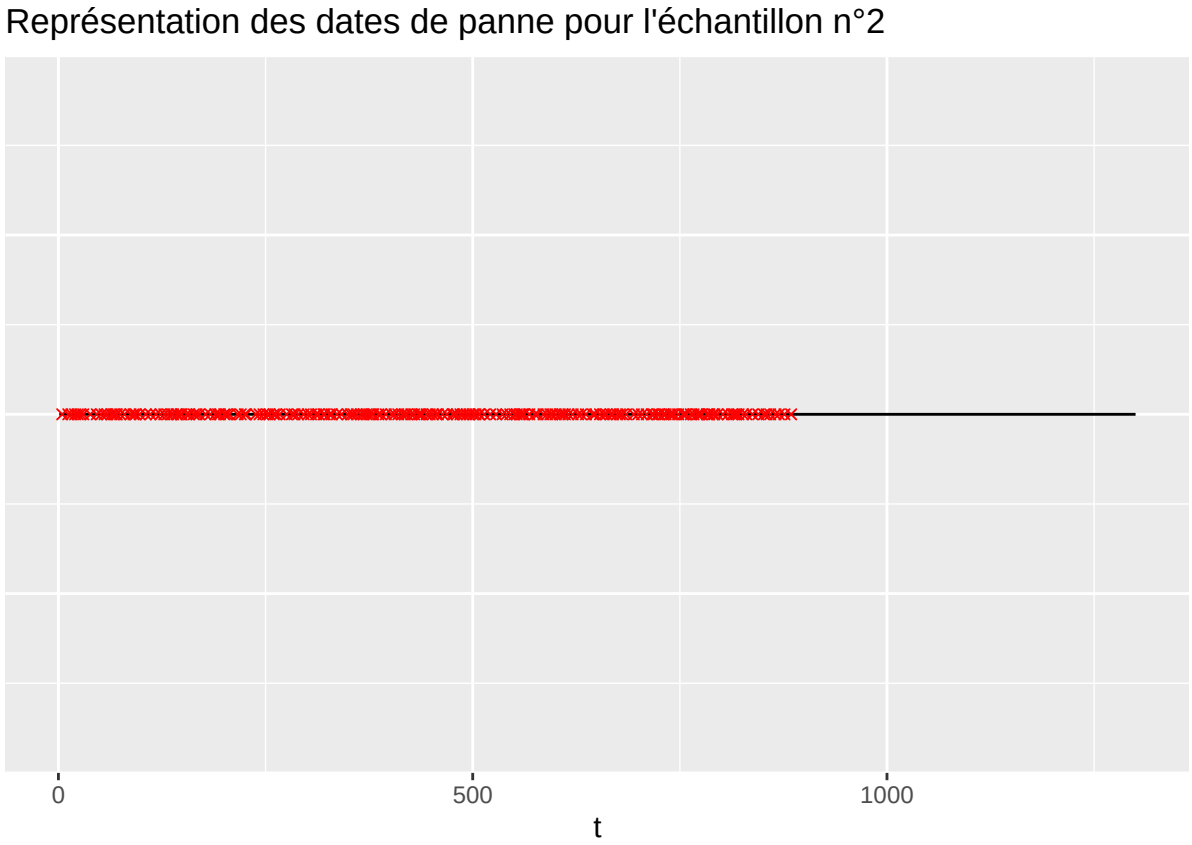
Échantillon n°2

On commence par tracer l'échantillon :

```
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")
```




```
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.

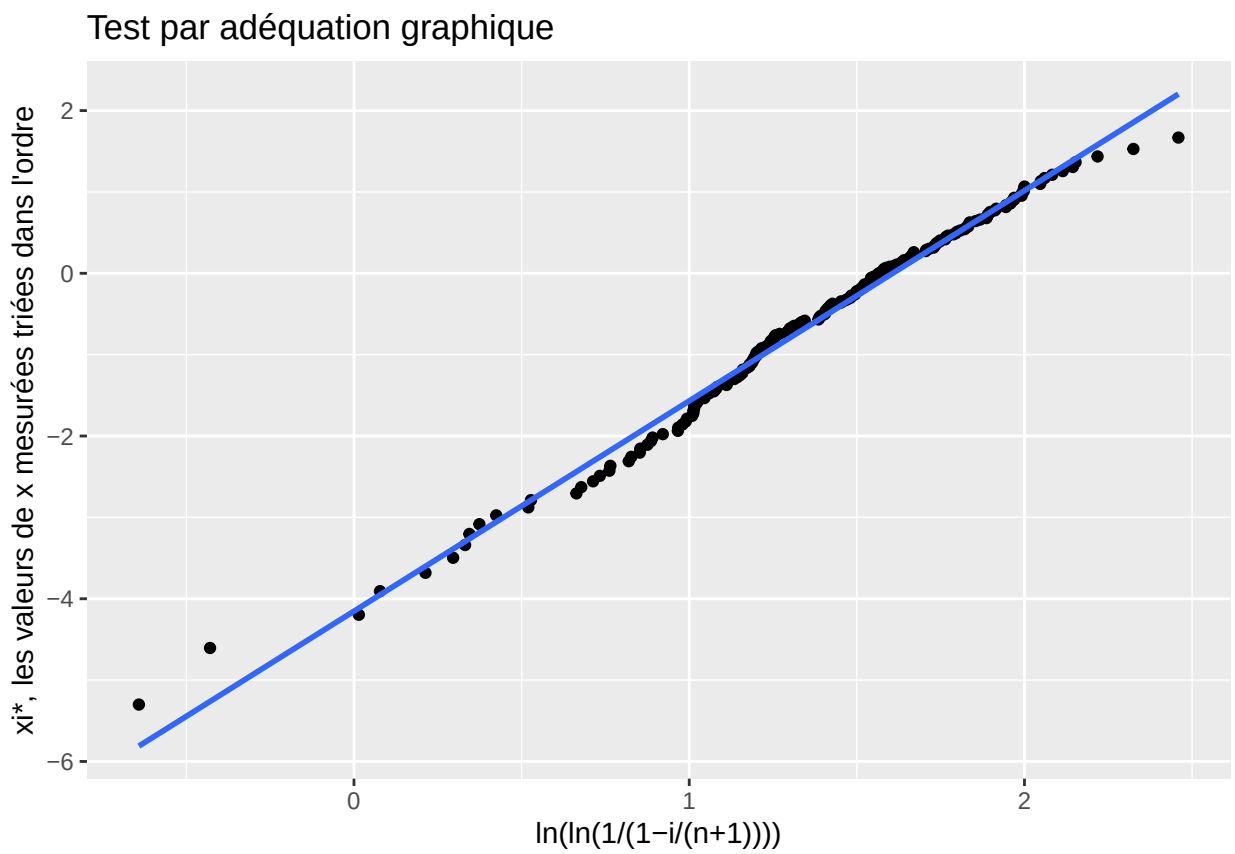
Pour celà on doit tracer le nuage de point :

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

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.

```r
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 :

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

Donc on a :

```r
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 :

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

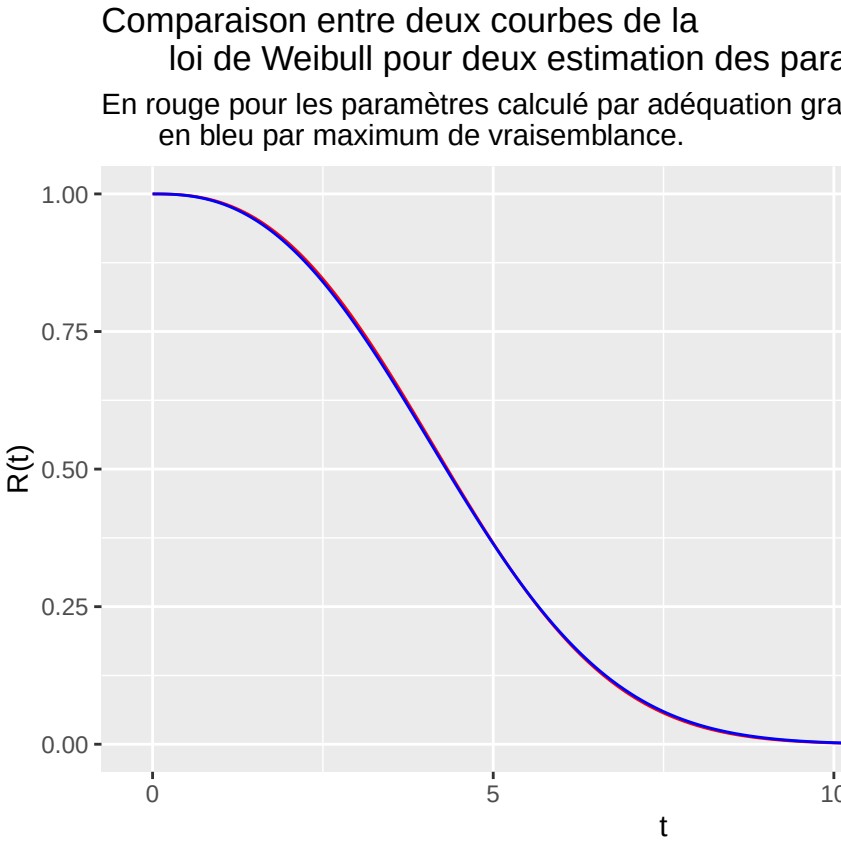
```
## [1] 4.980278
```

```r
estim_param_sys2_mv$beta
```

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

Les deux estimations sont très proches.

```
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.")
```

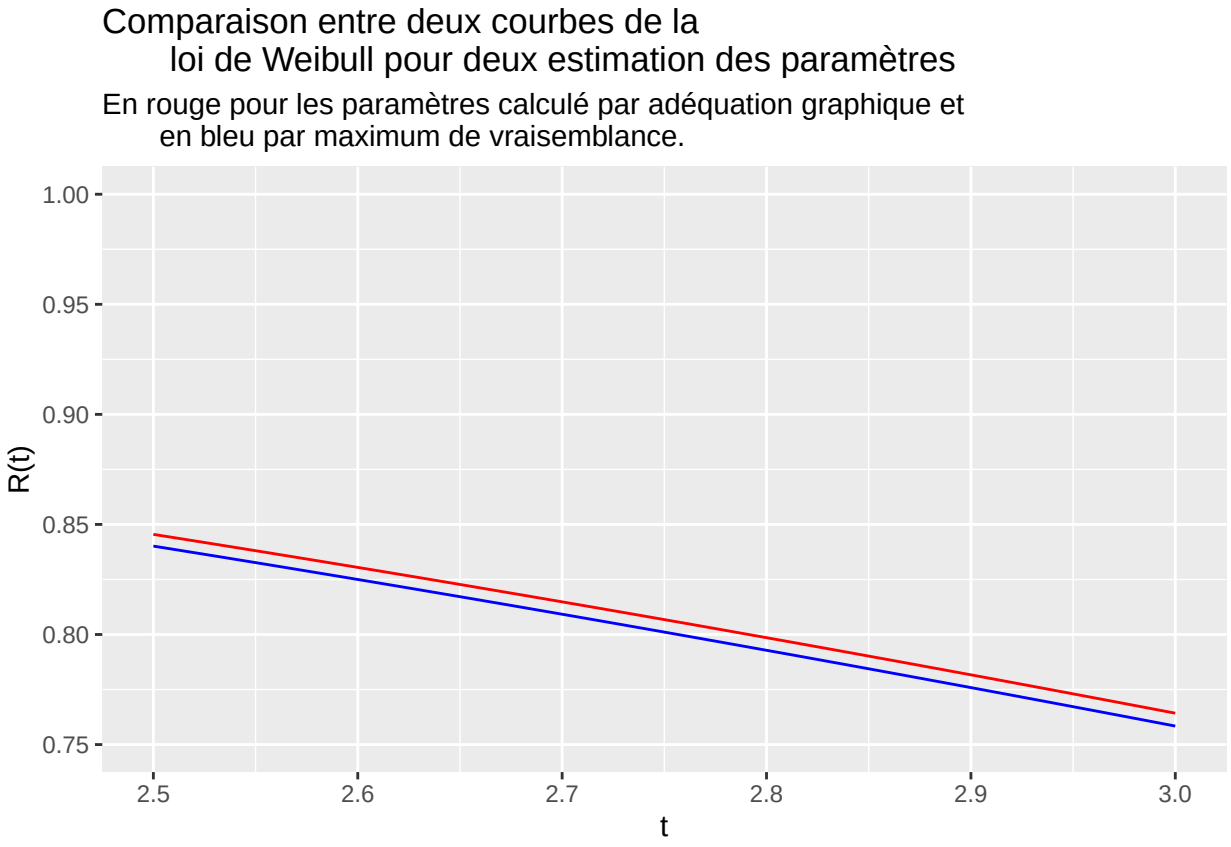


Représentation graphique des deux modèles

```
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.") +
  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)
```

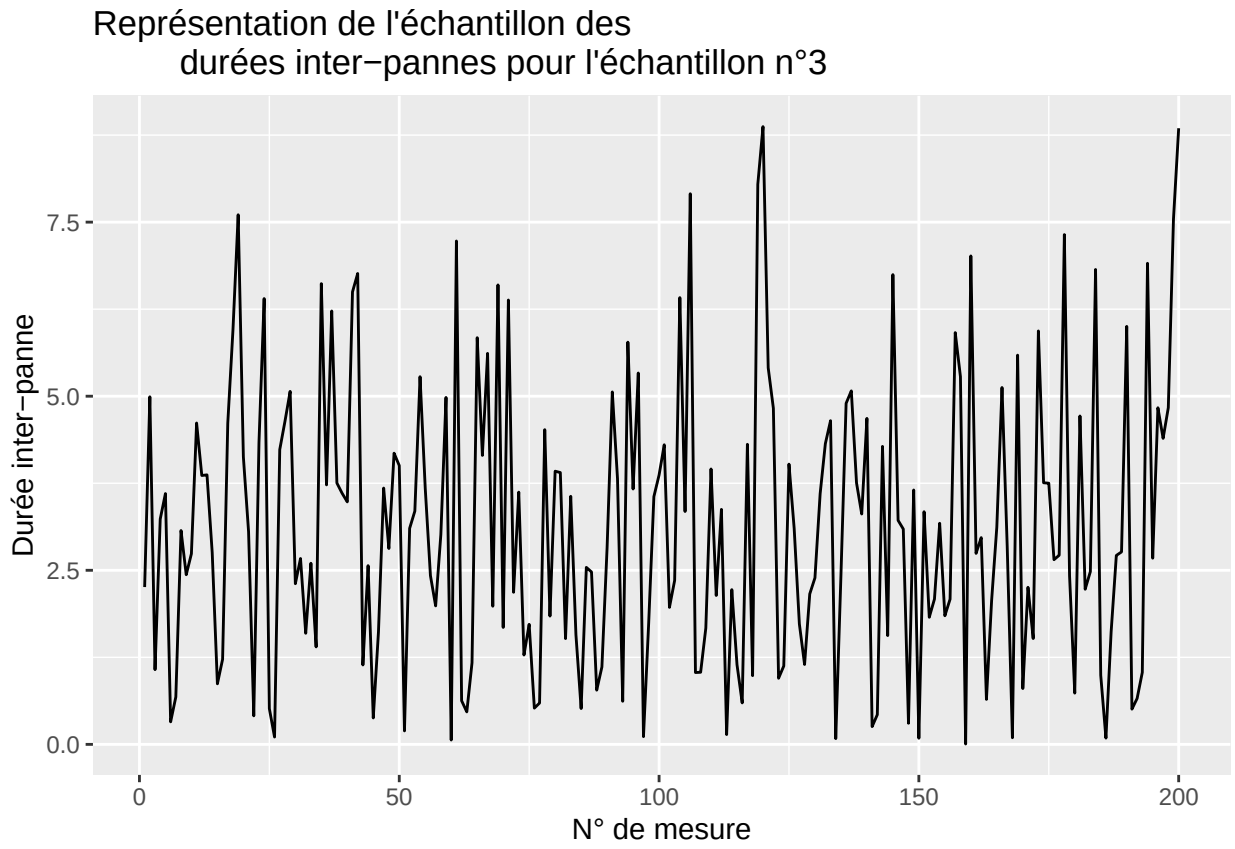
```
## [1] 4.420431
```

Échantillon n°3

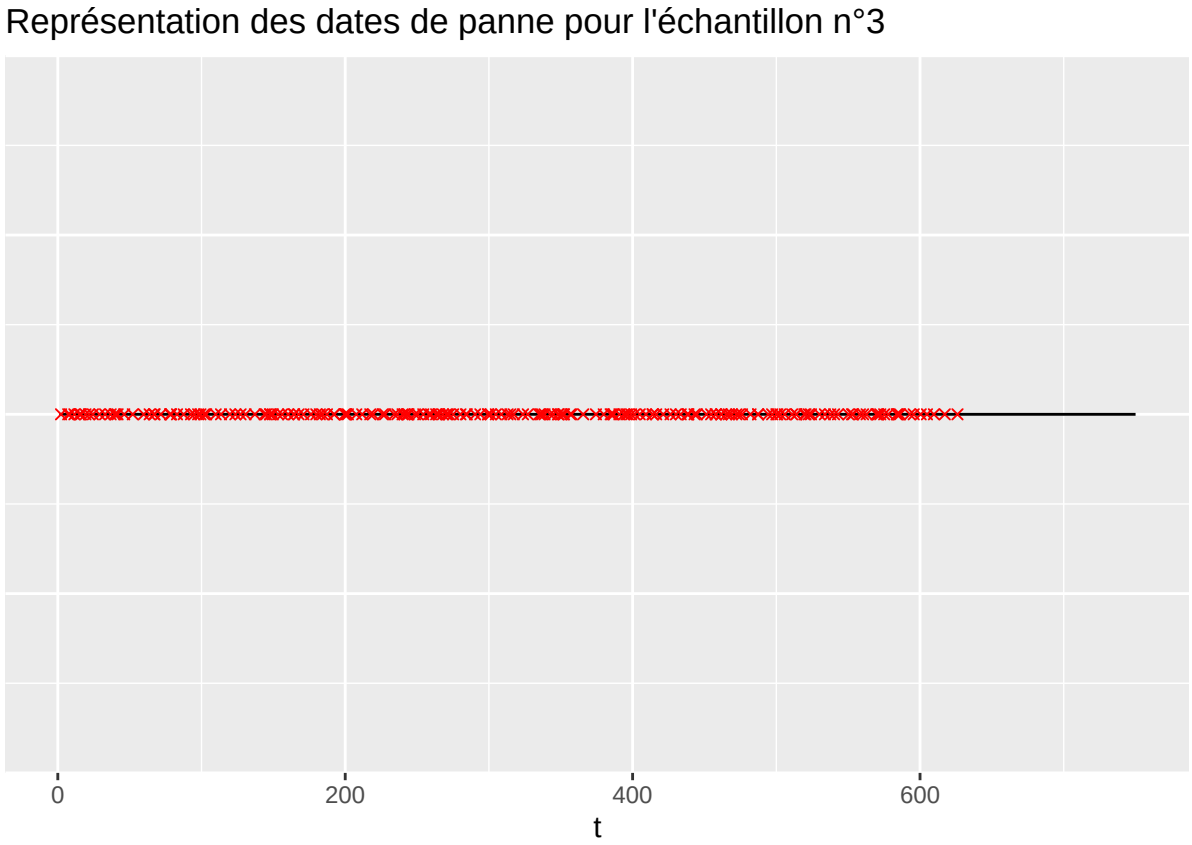
On peut commencer par tracer les données :

- 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
```

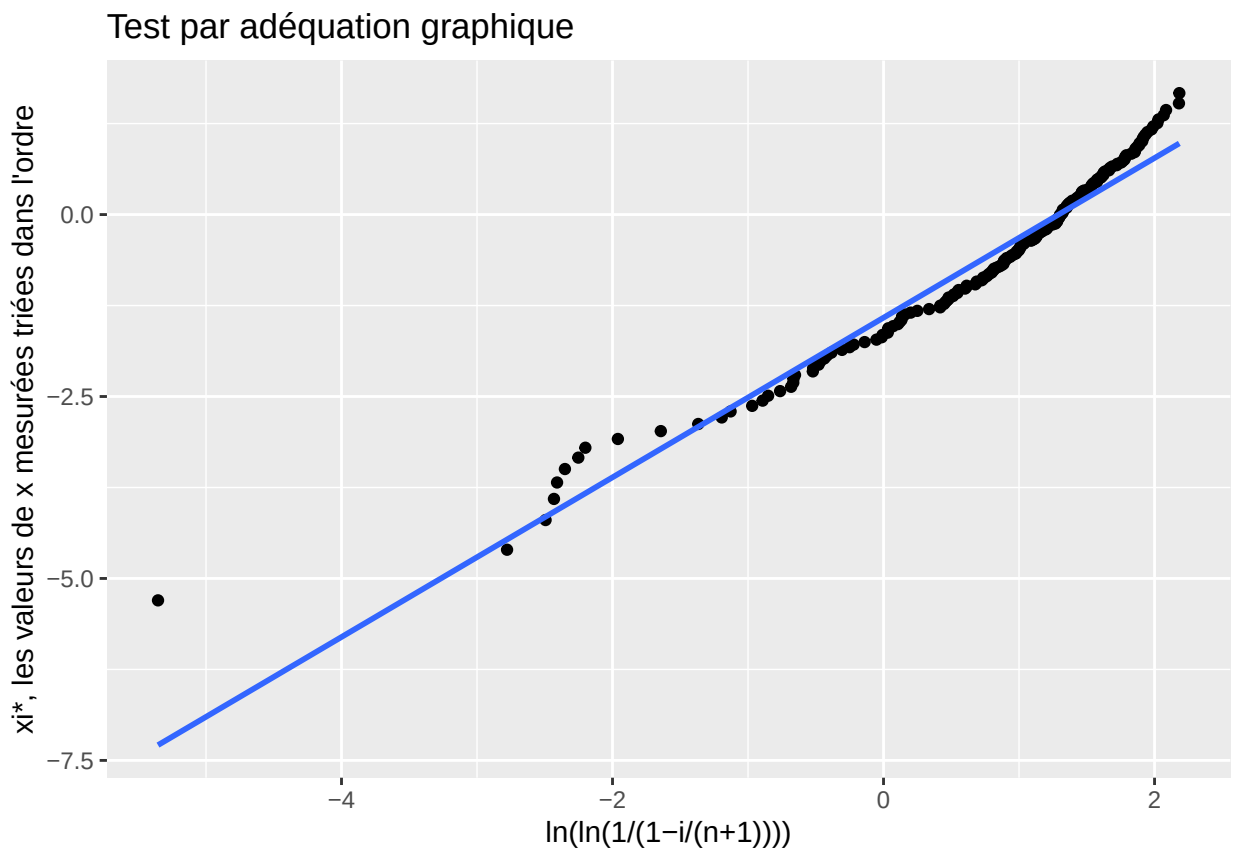
Cependant on rejette H_0 d'assez peu il faut donc rester critique sur les résultats obtenus par la suite.

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 *fit* pas parfaitement avec les données mais reste plutôt bonne. On peut ajouter la resgression 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.

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

Estimation des paramètres

```
## [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.

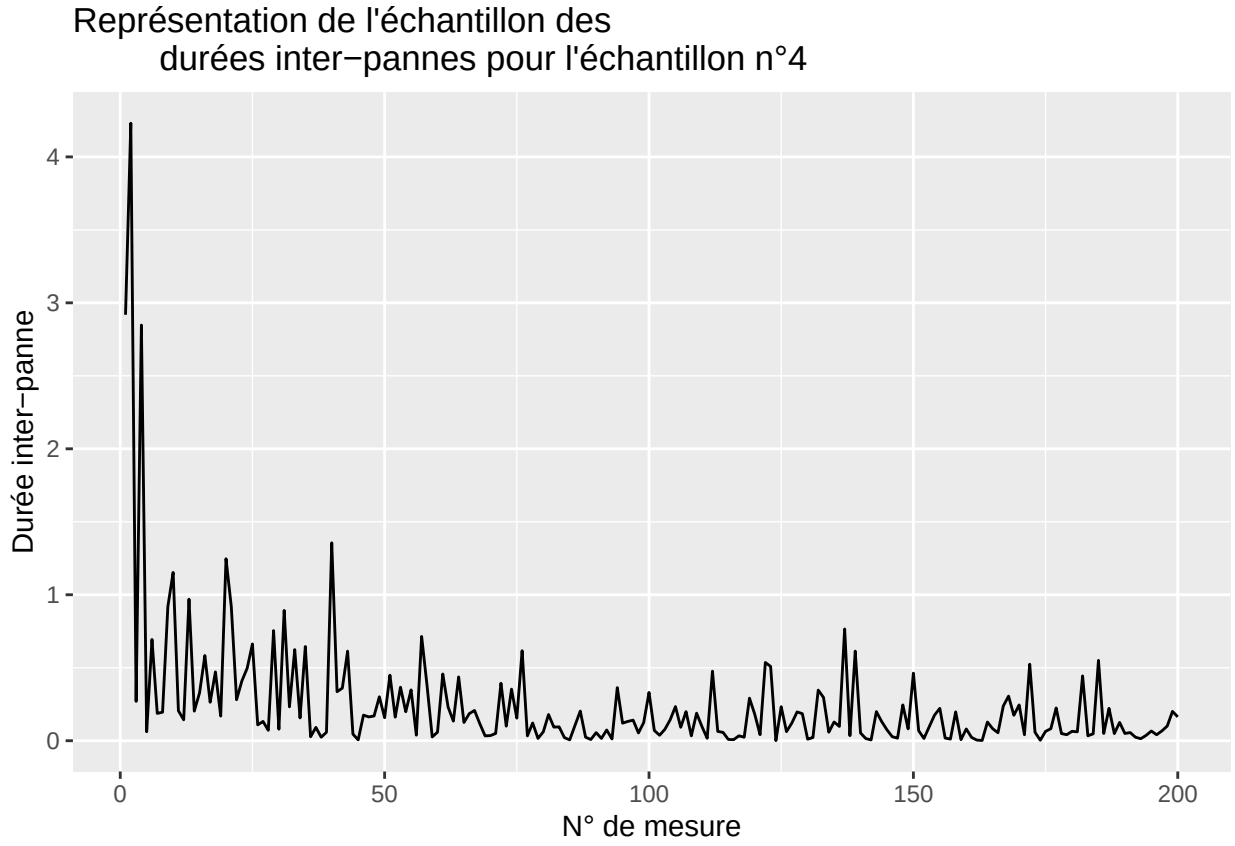
Conclusion Plus d'analyse devraient être menées pour conclure sur ce jeu de données.

Échantillon n°4

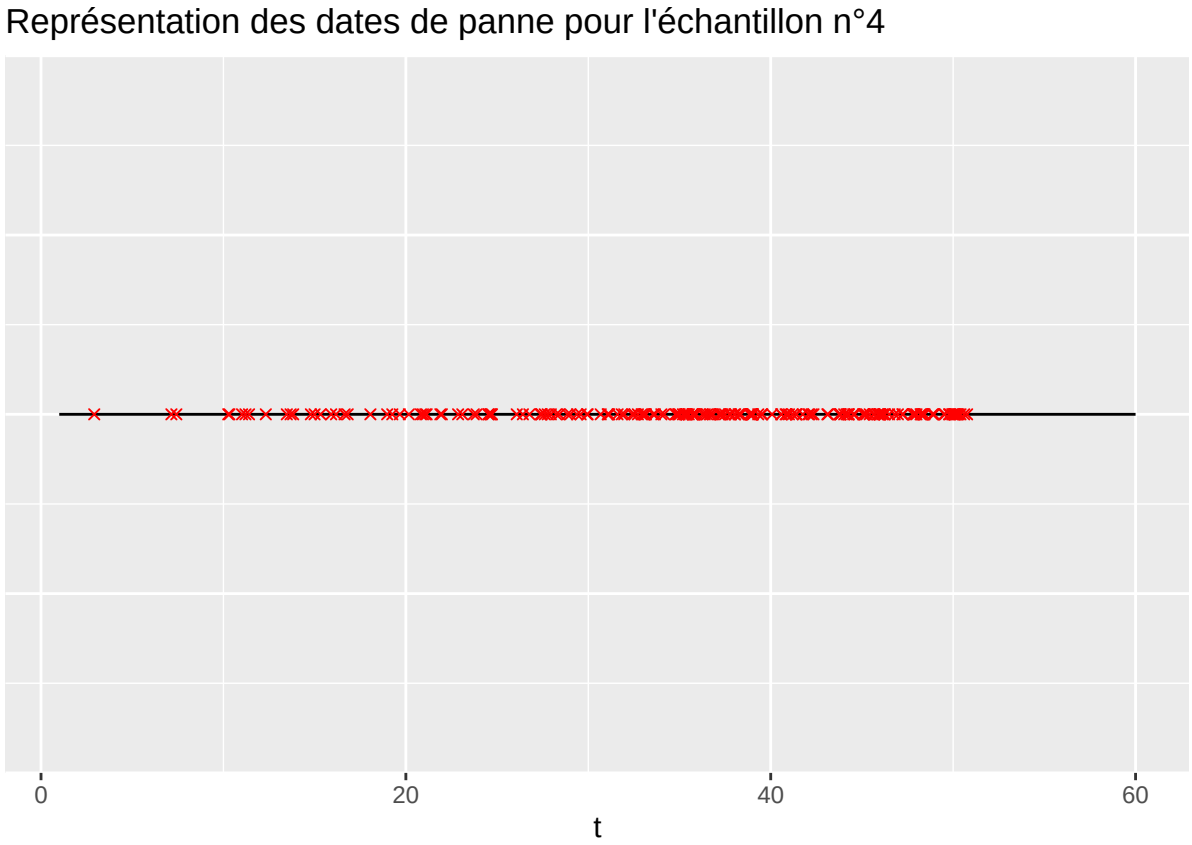
On peut commencer par tracer les données :

- 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 observer 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)*.

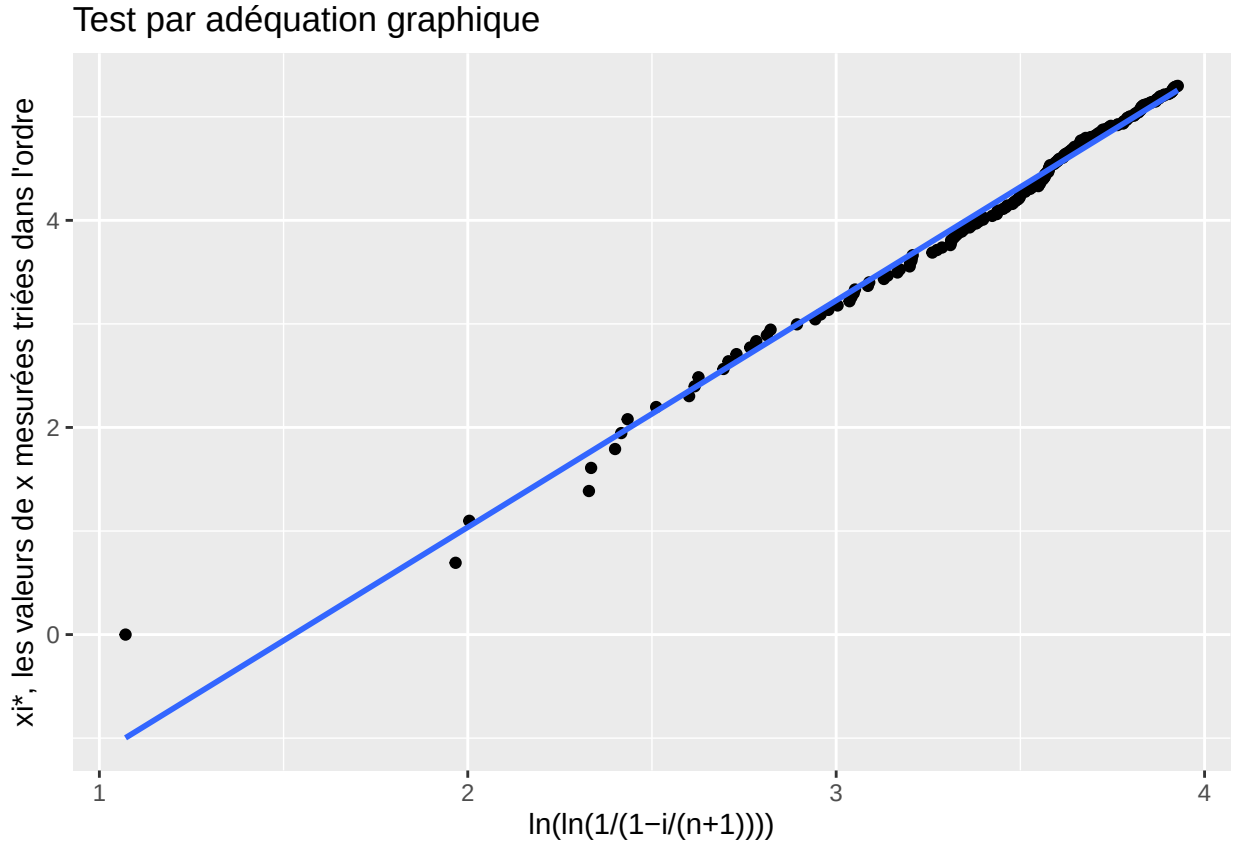
Dans ce cas on parle de **loi de gamma-Weibull**.

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` :

```
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 :

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

```
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 :

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

```
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.

On peut calculer les $MTTF$ selon les deux modèles trouvés :