# MEMO_MATHS_POP

2024-11-11

## Contents


**1 TD 1 : 2**
- 1.1 Exercice 1 : Simulation de lois de Weibull . . . 2
  - 1.1.1 Q1 : Simulation d'une loi de weibul en passant par la loi exponentielle . . . 2
  - 1.1.2 Q2 : Simulation d'une loi de Weibul en passant par la fonction rweibull : . . . 3
  - 1.1.3 Q3 : fonction de survie empirique sur l'échantillon . . . 4
  - 1.1.4 Q4 : Estimation des paramètres de la loi de Weibull par régression linéaire . . . 5
  - 1.1.5 Q5 : Estimation par maximun de vraisemblance . . . 7
  - 1.1.6 Q6 : Estimation par la méthode de Newton-Raphson (en utilisant le package pracma) 7
  - 1.1.7 Q7 : Comparaisons graphiques des méthodes . . . 7
  - 1.1.8 Q8 : utilisation de fitdistr . . . 8
  - 1.1.9 Q9 : utilisation d'une autre méthode de Newton-Raphson . . . 9
- 1.2 Exercice 2 : Le modèle de Gompertz - Makeham . . . 10
  - 1.2.1 Q1 : Fonction estimation moyenne Monte Carlo . . . 10
  - 1.2.2 Q2 : Paramètres estimés (données) . . . 10
  - 1.2.3 Q3 : Fonction de Hasard : . . . 10
  - 1.2.4 Q4 : Simulation d'échantillon . . . 11
  - 1.2.5 Q5 : Fonction de survie . . . 14
  - 1.2.6 Q7 : Calcul approché de E[T] à partir de S theo . . . 15

**2 TD 2 : 16**
- 2.1 Exercice 1 : . . . 16
  - 2.1.1 Q1 : Importation des données : . . . 16
  - 2.1.2 Q2 : Modèle de Gompertz Makeham : . . . 16
  - 2.1.3 Q3 : Comparaison de modèles log taux et taux brute . . . 20

**3 TD3 : Examen 2019 22**
- 3.1 Exercice 1 : . . . 22
  - 3.1.1 Q0 : Importation des données . . . 22
  - 3.1.2 Q1 : Calculer l'espérance de vie résiduelle aux âges 0,20,40,60,80 ans pour l'année 2015 22
  - 3.1.3 Q2 : Tracer l'espérance de vie résiduelle à la naissance en fonction de l'année d'observation . . . 23
  - 3.1.4 Q3 : Commentaire de courbe . . . 27
- 3.2 Exercice 2 (13 du cours) : . . . 27
  - 3.2.1 Q1 : Simulation de l'énoncé . . . 27
  - 3.2.2 Q2 : Application de l'estimation . . . 27

**4 Examen 2017 : 28**
- 4.1 Exercice 1 : . . . 28
  - 4.1.1 Q0 : Importation des données . . . 28
- 4.2 Q1 : Taux brute de mortalité . . . 28
  - 4.2.1 Q2 : . . . 30
  - 4.2.2 Q4 : . . . 31

**5 Fonction déjà établies dans le cours** **31**
5.1 Exercice 2 : . . . 32
5.1.1 Q1 : Points fixes . . . 32
5.1.2 Q2 : . . . 32

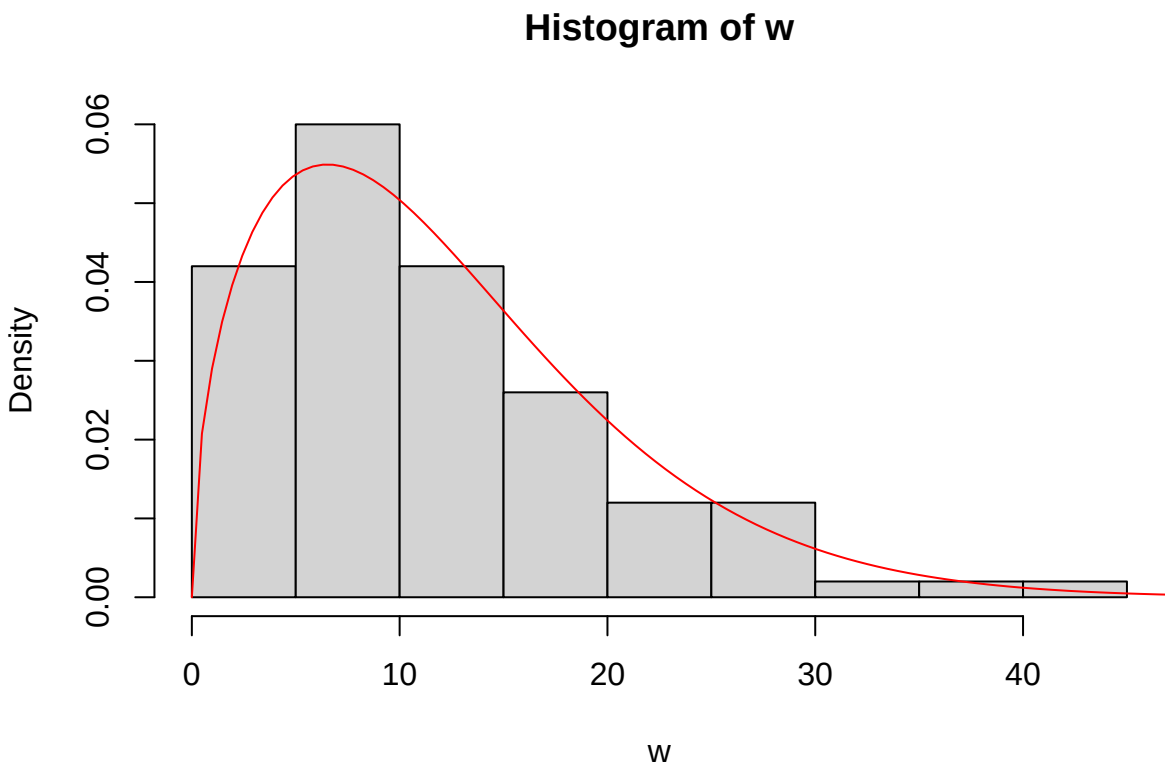
**6 Généralités sur les lois :** **33**
6.1 La loi exponentielle . . . 33
6.1.1 Définition . . . 33
6.1.2 Fonction de densité de probabilité (f.d.p) . . . 33
6.1.3 Fonction de répartition (F.d.R) . . . 33
6.1.4 Espérance et Variance . . . 33
6.1.5 Propriétés . . . 33
6.2 Loi de Weibul : . . . 33
6.2.1 Définition . . . 33
6.2.2 Fonction de Densité de Probabilité (f.d.p.) . . . 33
6.2.3 Fonction de Répartition . . . 33
6.2.4 Espérance et Variance . . . 34
6.2.5 Propriétés . . . 34
6.2.6 Transition de la Loi de Weibull à une Loi Exponentielle . . . 34
6.3 La loi Gamma . . . 34
6.3.1 Définition . . . 34
6.3.2 Fonction de Densité de Probabilité (f.d.p.) . . . 34
6.3.3 Fonction de Répartition . . . 34
6.3.4 Espérance et Variance . . . 34
6.3.5 Propriétés . . . 35
6.3.6 Transition de la Loi Gamma à une Loi Exponentielle . . . 35
6.4 Min de loi exponentielle . . . 35
6.4.1 Variable Aléatoire T . . . 35

# 1 TD 1 :

## 1.1 Exercice 1 : Simulation de lois de Weibull

### 1.1.1 Q1 : Simulation d'une loi de weibul en passant par la loi exponentielle

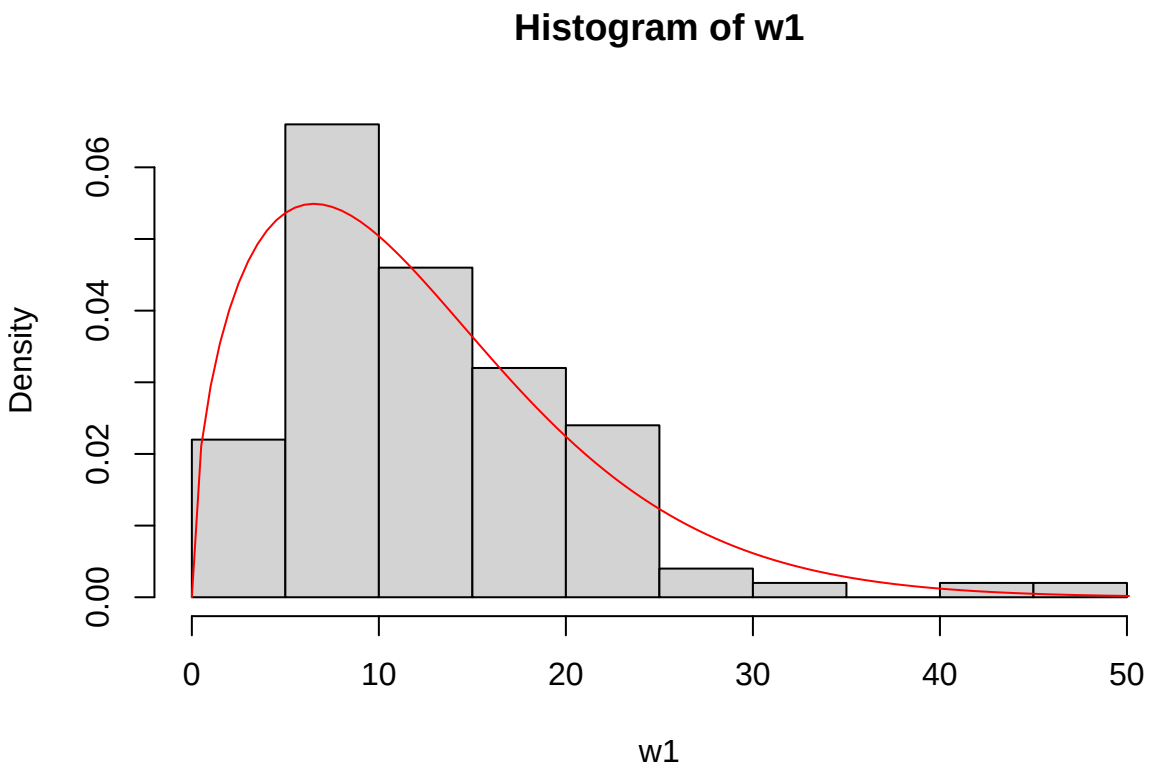
```
set.seed(14342)
n = 100 # taille de l'échantillon
l = 0.02 # paramètre de forme
a = 1.5 # paramètre d'échelle
x = rexp(n, l) # loi exponentielle
w = x ^ { 1 / a } # loi de Weibull
#hist(w)
hist(w, freq = FALSE) # densité
plot(function(x) l * a * x ^ { a - 1 } * exp(-l * x ^ a), 0, max(w) + 5,
     add = TRUE, col = 'red') # densité de probabilité de la loi de Weibull
```




```
# La fonction muette est tracée sur l'intervalle [0, max(w) + 5] en rouge.
```

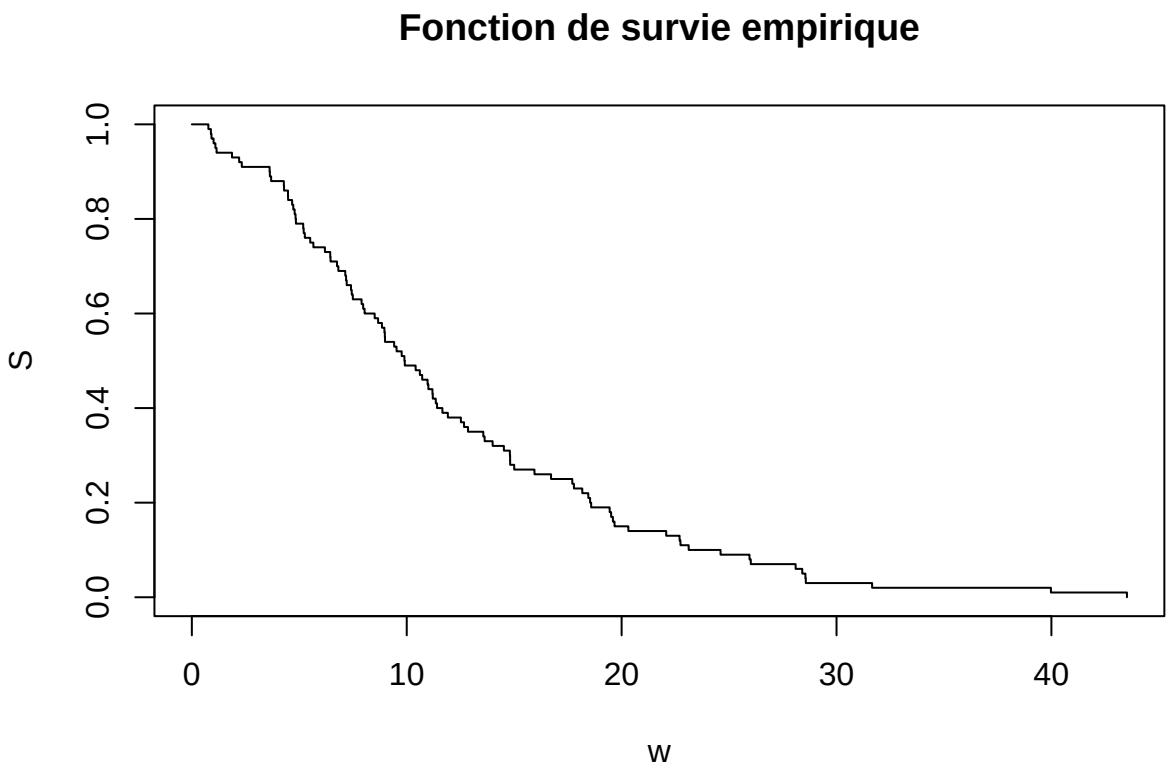
### 1.1.2 Q2 : Simulation d'une loi de Weibul en passant par la fonction rweibull :

```
set.seed(14342)
n = 100 # taille de l'échantillon
l = 0.02 # paramètre de forme
a = 1.5 # paramètre d'échelle

# Remarque : équivalent avec rweibull : paramétrage a=Shape et L=scale
# hypothèse : P[T> t] =exp(- (t/L)^a )
L = l ^ (-1 / a) # paramètre d'échelle
w1 = rweibull(n, a, L)    # simuler une loi de Weibull
hist(w1, freq = FALSE)
plot(function(x) l * a * x ^ { a - 1 } * exp(-l * x ^ a), 0, max(w1) + 5,
     add = TRUE, col = 'red')
```




### 1.1.3 Q3 : fonction de survie empirique sur l'échantillon

```
t = sort(w) # ordonner les valeurs simulés par la loi de weibul (des âges sans doute)
Se = 1 - (1:n) / n # fonction de survie empirique
plot(
  c(0, t),
  c(1, Se),
  type = 's',
  xlab = "w",
  ylab = "S",
  main = 'Fonction de survie empirique'
)
```



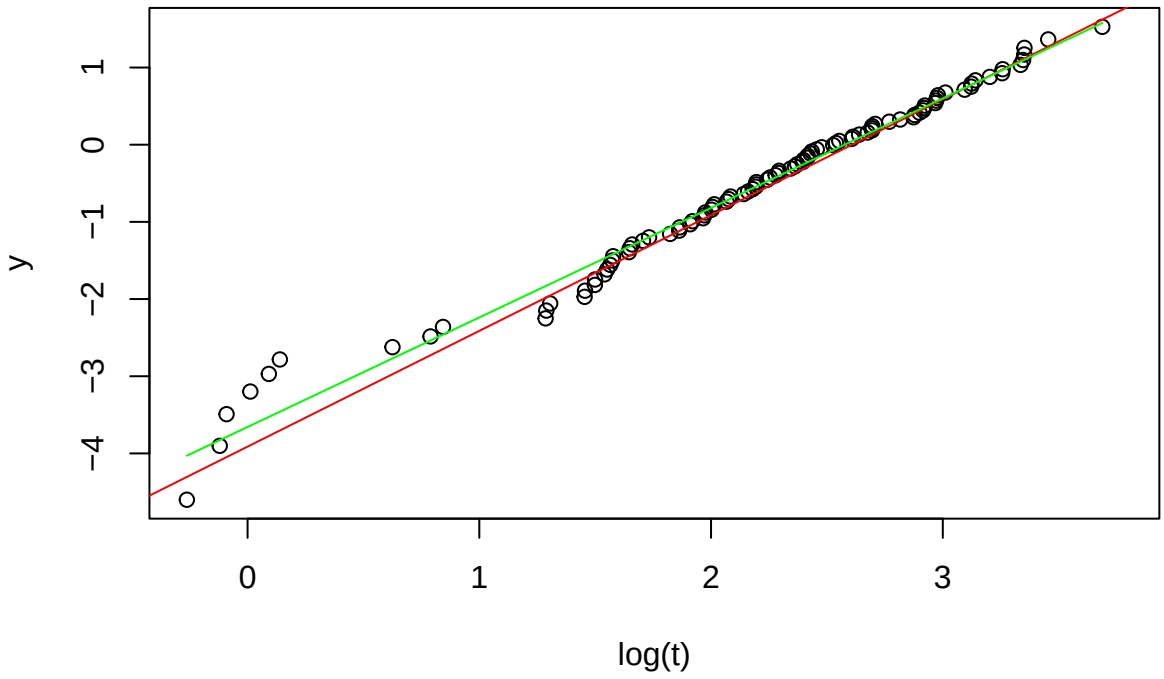

### 1.1.4 Q4 : Estimation des paramètres de la loi de Weibull par régression linéaire

```
# Adéquation graphique à la loi de Weibull ?
# Observe-t-on une tendance linéaire ?
y = log(-log(Se)) # transformation de la fonction de survie
plot(log(t), y, type = 'p')
plot(function(x)
  log(l) + a * x, -1, 4, add = TRUE, col = 'red')


# Méthode Régression pour estimer les paramètres
logt = log(t[-n])
z = log(-log(Se[-n])) # on enlève la dernière sinon : Inf
reg = lm(z ~ logt)

summary(reg)
```

```
##
## Call:
## lm(formula = z ~ logt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.57105 -0.04235  0.01082  0.05483  0.67684
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.65722    0.04434  -82.47   <2e-16 ***
## logt         1.41920    0.01896   74.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1589 on 97 degrees of freedom
## Multiple R-squared:  0.983,  Adjusted R-squared:  0.9828
## F-statistic:  5606 on 1 and 97 DF,  p-value: < 2.2e-16
```

```
y = fitted(reg)
lines(logt, y, col = 'green')
```




```
a1 = reg$coefficients[2]
l1 = exp(reg$coefficients[1])

cat("valeurs estimées : ",c(a1, l1), "\n",
          "valeurs exactes : " , c(a, l))
```

```
## valeurs estimées :  1.419203 0.02580405
##  valeurs exactes :  1.5 0.02
```

```
# # valeurs estimées des paramètres :
# c(a1, l1)
# # valeurs exactes :
# c(a, l)
```

### 1.1.5 Q5 : Estimation par maximun de vraisemblance

```
f = function(x)
{
  1 / x - sum(t ^ x * log(t)) / sum(t ^ x) + sum(log(t) / n)
}

MV = uniroot(f, interval = c(a1 - 0.5, a1 + 0.5)) # solution f(a) = 0
a2 = MV$root
l2 = n / sum(t ^ a2)

cat("Régression linéaire : " , c(a1, l1), "\n" , "Méthode de vraisemblance : " ,c(a2, l2), "\n", "Valeu
```

```
## Régression linéaire :  1.419203 0.02580405
##  Méthode de vraisemblance :  1.456527 0.02303103
##  Valeur exacte :  1.5 0.02
```

```
# c(a1, l1) # reg.
# c(a2, l2) # MV1
# c(a, l) # val. exactes
```

### 1.1.6 Q6 : Estimation par la méthode de Newton-Raphson (en utilisant le package pracma)

```
#install.packages(pracma)
library(pracma)
```

```
## Warning: le package 'pracma' a été compilé avec la version R 4.3.3
```

```
n4 = newtonRaphson(f, a1)
a3 = n4$root
l3 = n / sum(t ^ a3)

cat("Valeur Newton-Raphson : ", c(a3, l3) )
```

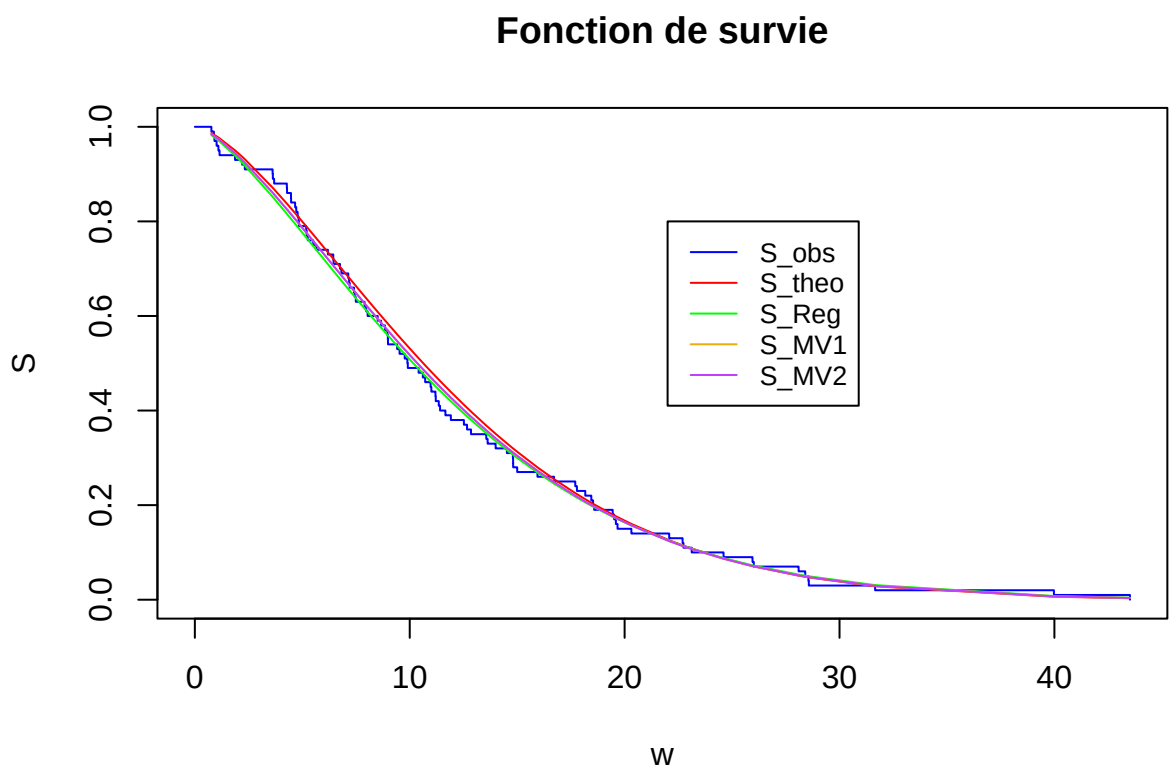
```
## Valeur Newton-Raphson :  1.456525 0.0230311
```

```
# c(a3, l3) # Newton-Raphson
```

### 1.1.7 Q7 : Comparaisons graphiques des méthodes

```
# graphes :
plot(
  c(0, t),
  c(1, Se),
  type = 's',
  col = 'blue',
  xlab = "w",
  ylab = "S",
  main = 'Fonction de survie'
)
Stheo = exp(-l * t ^ a)
Stheoestime1 = exp(-l1 * t ^ a1) # méthode régression
Stheoestime2 = exp(-l2 * t ^ a2) # méthode MV1
Stheoestime3 = exp(-l3 * t ^ a3) # méthode raphson (pracma)
lines(t, Stheo, col = 'red')
```

```r
lines(t, Stheoestime1, col = 'green')
lines(t, Stheoestime2, col = 'darkgoldenrod2')
lines(t, Stheoestime3, col = 'darkorchid1')

legend(
  22,
  0.8,
  legend = c("S_obs", "S_theo", "S_Reg", "S_MV1" , "S_MV2"),
  col = c("blue", "red",  "green", "darkgoldenrod2", 'darkorchid1'),
  lty = 1,
  cex = 0.8
)
```




### 1.1.8 Q8 : utilisation de fitdistr

```r
library(MASS)
p = fitdistr(w, "weibull")
#p
a5 = p$estimate[1]
L = p$estimate[2]
l5 = L ^ (-a5)
c(a5, l5)
```

#### 1.1.8.1 Estimation fitdistr

```
##       shape       scale
```

```
## 1.45652561 0.02303108
```

```r
# en utilisant les valeurs initiales estimées précédemment :
p = fitdistr(w, start = list(shape = a1, scale = l1 ^ (-1 / a1)), "weibull")
a6 = p$estimate[1]
L = p$estimate[2]
l6 = L ^ (-a6)
c(a6, l6)
```

#### 1.1.8.2 Utilisation des valeurs initiales estimées précédemment :

```
##     shape     scale
## 1.4565255 0.0230311
```

### 1.1.9 Q9 : utilisation d'une autre méthode de Newton-Raphson

```r
##############################################
# remarque : autre algorithme pour Newton-Raphson -----
# Methode de Newton-Raphson
# https://rpubs.com/aaronsc32/newton-raphson-method
library(numDeriv)

newton.raphson <- function(f, a, b, tol = 1e-5, n = 1000) {
  # require(numDeriv) # Package for computing f'(x)

  x0 <- a # Set start value to supplied lower bound
  k <- n # Initialize for iteration results

  # Check the upper and lower bounds to see if approximations result in 0
  fa <- f(a)
  if (fa == 0.0) {
    return(a)
  }

  fb <- f(b)
  if (fb == 0.0) {
    return(b)
  }

  for (i in 1:n) {
    dx <- genD(func = f, x = x0)$D[1] # First-order derivative f'(x0)
    x1 <- x0 - (f(x0) / dx) # Calculate next value x1
    k[i] <- x1 # Store x1
    # Once the difference between x0 and x1 becomes sufficiently small, output the results.
    if (abs(x1 - x0) < tol) {
      root.approx <- tail(k, n = 1)
      res <-
        list('root approximation' = root.approx,
             'iterations' = k)
      return(res)
    }
    # If Newton-Raphson has not yet reached convergence set x1 as x0 and continue
    x0 <- x1
```

```
  }
  print('Too many iterations in method')
}

# avec Newton-Raphson (voir fonction ci-dessus)
a3 = newton.raphson(f, a1 - 0.5, a1 + 0.5)
a3 = a3$`root approximation`
l3 = n / sum(t ^ a3)
c(a3, l3) # MV2
```

## 1.2 Exercice 2 : Le modèle de Gompertz - Makeham

### 1.2.1 Q1 : Fonction estimation moyenne Monte Carlo

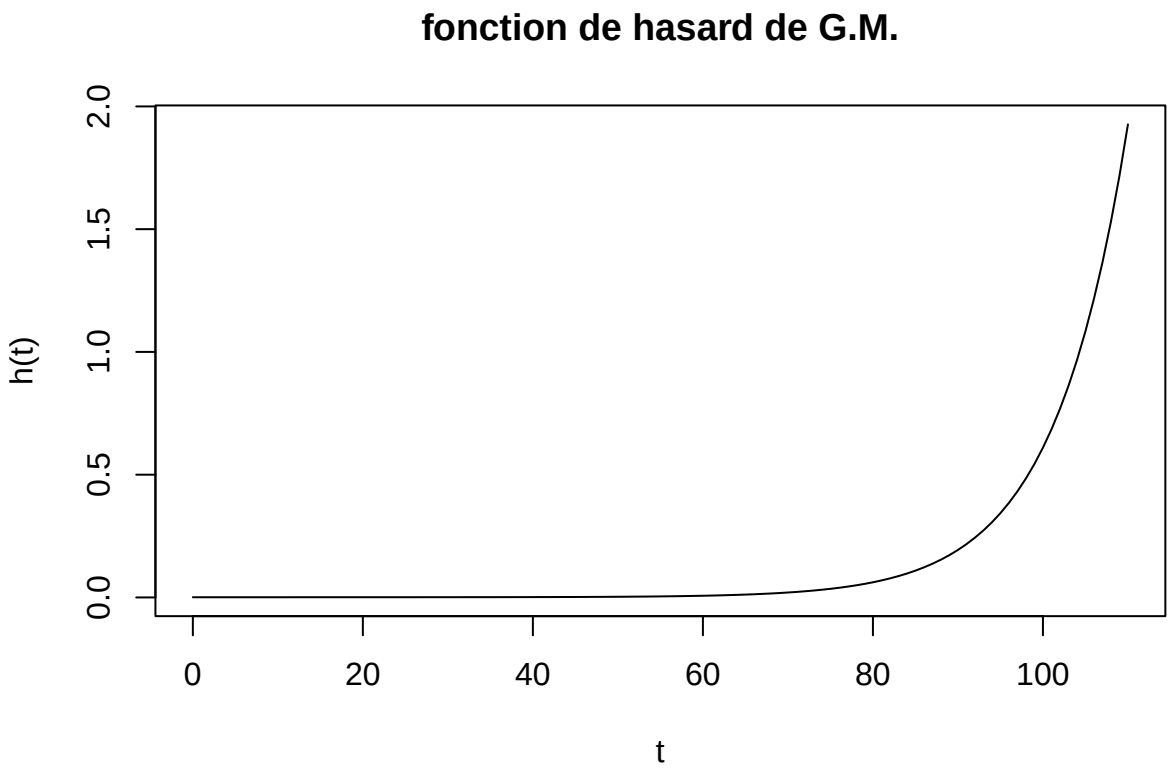
```
# fonction d'estimation de moyenne par Méthode Monte Carlo
# avec intervalle de confiance au niveau de confiance 1- alpha
MCmean = function(x, alpha)
{
  n = length(x)
  m = mean(x)
  S = sd(x)
  t = qnorm(1 - alpha / 2)
  Iinf = m - t * S / sqrt(n)
  Isup = m + t * S / sqrt(n)
  data.frame(m, Iinf, Isup, S, n, alpha)
}
```

### 1.2.2 Q2 : Paramètres estimés (données)

```
# paramètres estimés :
a = 7.6655e-04
b = 6.1041e-06
l = 0.11511
g = exp(l)
```

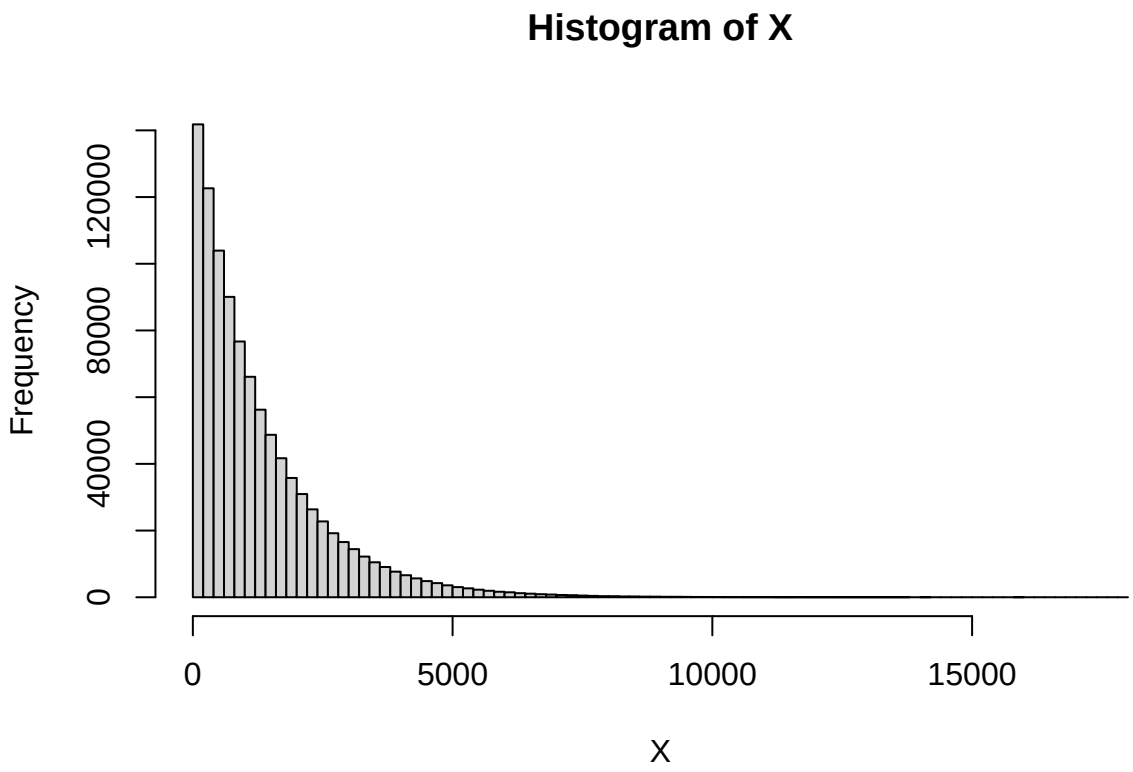
### 1.2.3 Q3 : Fonction de Hasard :

```
# Fonctiond de hasard : ----
# fonction de hasard h(t)  = a + b *g^t
h = function(t)
{
  a + b * g ^ t
}
t = 0:110
plot(t, h(t), type = 'l', main = 'fonction de hasard de G.M.')
```

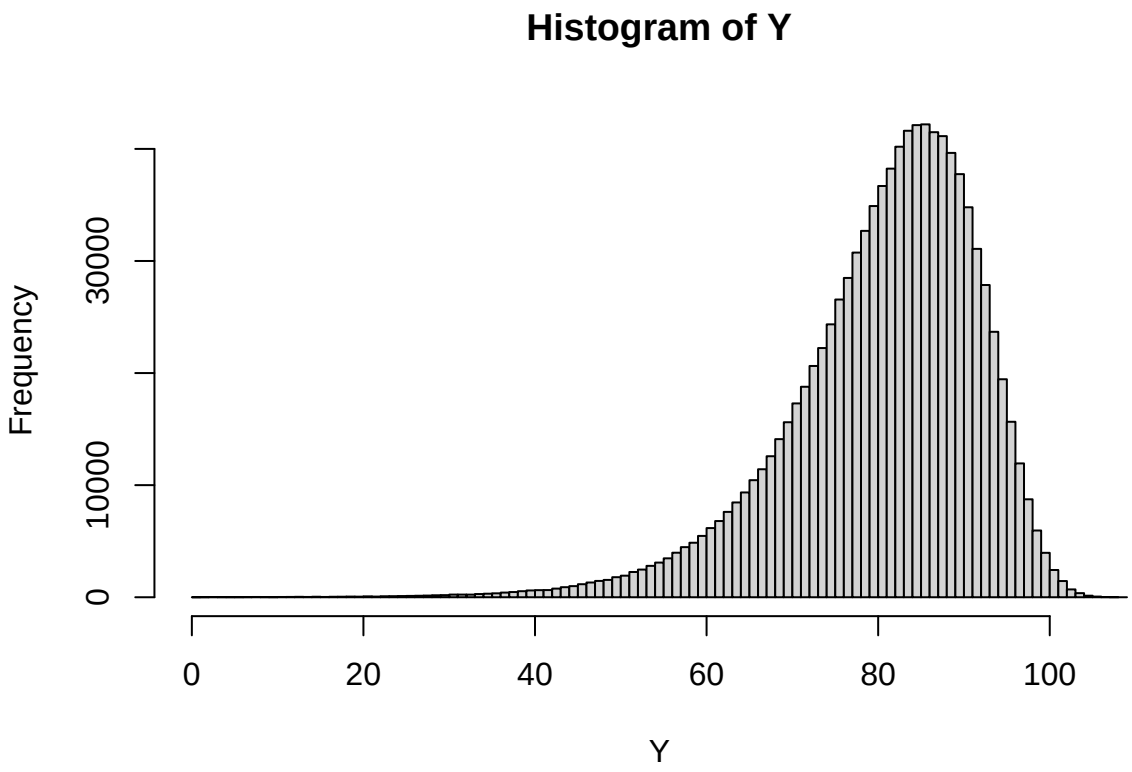



#### 1.2.4 Q4 : Simulation d'échantillon

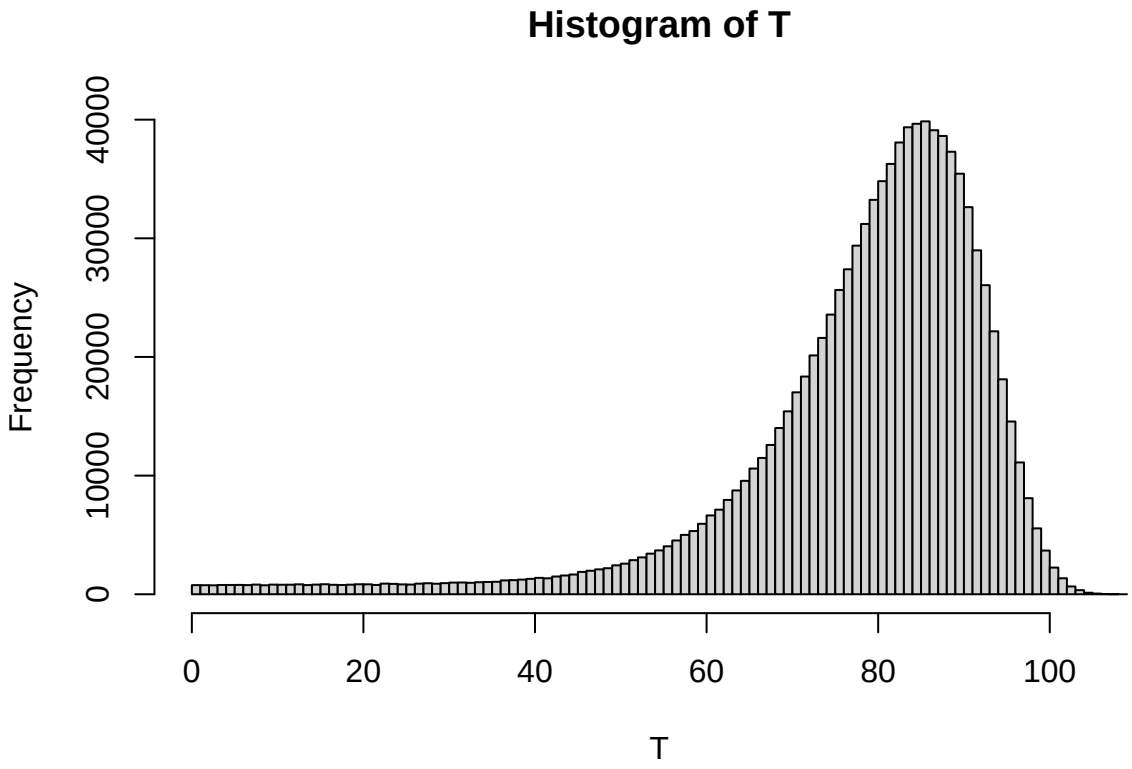
```
# simulation échantillon de Durées de vie selon le modèle de Gompertz - Makeham
n = 1e6 # taille de l'échantillon
X = rexp(n, a)  # exponentielle
Y = 1 / l * log(1 - l / b * log(1 - runif(n)))  # Gompertz en inversant la fonction de répartition

T = pmin(X, Y)
hist(X, 100)
```


Histogram of X
Frequency
0
40000
80000
120000
0
5000
10000
15000
X

```
hist(Y, 100)
```




```
hist(T, nclass = 100)
```




```
# Fonction Monte-Carlo estimation
MCmean(T, 0.05)
```

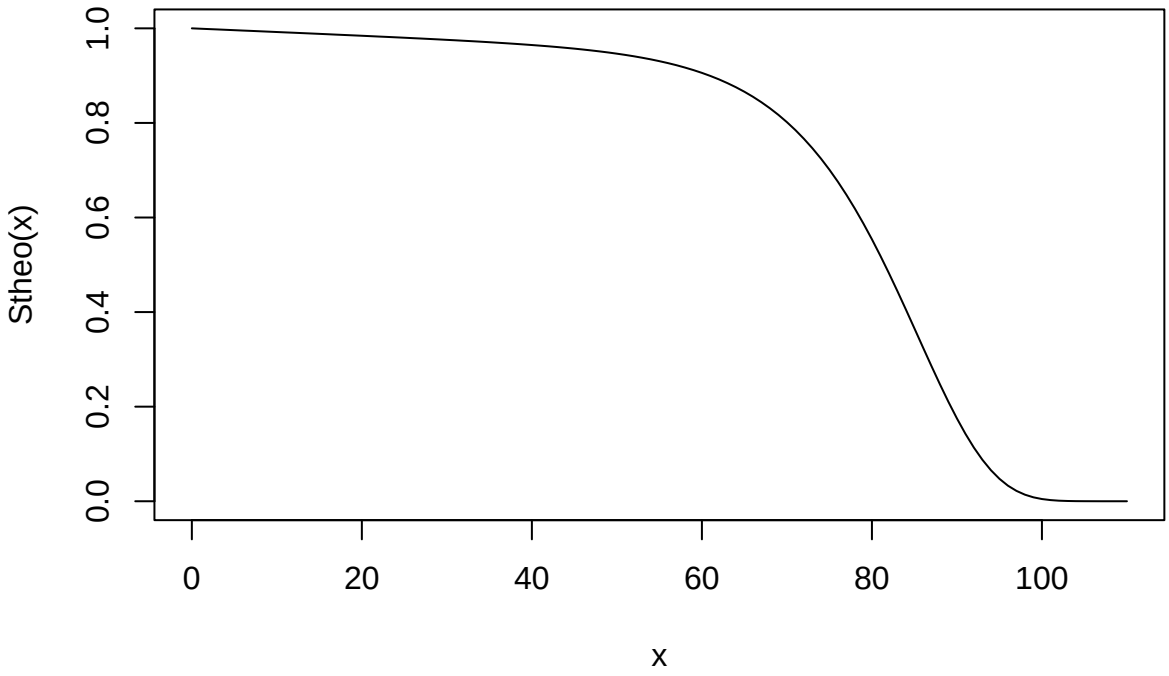
```
##         m     Iinf     Isup        S       n alpha
## 1 78.0439 78.01334 78.07446 15.59199 1000000  0.05
```

```
MCmean(T > 75, 0.05)
```

```
##          m      Iinf      Isup        S       n alpha
## 1 0.701139 0.7002418 0.7020362 0.457759 1000000  0.05
```

### 1.2.5 Q5 : Fonction de survie

```
#fonction de survie S  = S_T= S_X * S_Y
Stheo=function(x)
{exp(-a*x-b/l*(g^x-1))}

x=0:110
plot(x,Stheo(x),type='l')
```




### Q6 : Calcul exact de P[T>75]= P[Y>75] P[X>75]

```
# Remarque : Calcul exact de P[T>75]= P[Y>75] P[X>75]
Stheo(75)
```

```
## [1] 0.7010272
```

```
MCmean(T > 75, 0.01)
```

```
##          m      Iinf      Isup        S       n alpha
## 1 0.701139 0.6999599 0.7023181 0.457759 1000000  0.01
```

#### 1.2.6 Q7 : Calcul approché de E[T] à partir de S theo

```
x = 0:110
sum(Stheo(x))
sum(Stheo(x + 1))
sum(Stheo(x + 0.5))

(sum(Stheo(x))+sum(Stheo(x+1)))/2

# mediane ?
m=sort(T)[n/2]
m
mean(T<=m)
# ou bien :
median(T)
```

# 2 TD 2 :

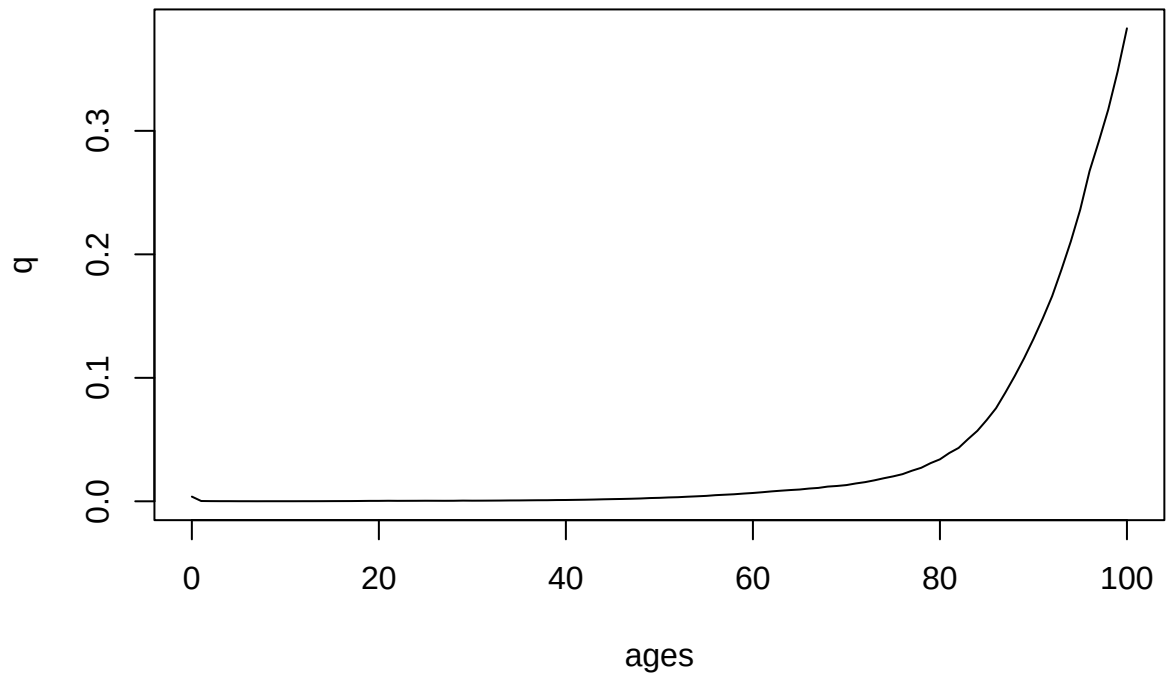
## 2.1 Exercice 1 :

### 2.1.1 Q1 : Importation des données :

```
# Importation des données : ----

## France, Total Population, Deaths (period 1x1),
#Last modified: 12 Aug 2022;  Methods Protocol: v6 (2017)
De = read.csv("DATA/DeathsFrance2022.csv", header = TRUE, sep = ";")
#str(De)
#De[1, ]
#unique(De$Year)

# Remarque : la classe d'âge "110" est en réalité "110 et plus".
E = read.csv("DATA/ExposuresFrance2022.csv", header = TRUE, sep = ";")
# str(E)
```

### 2.1.2 Q2 : Modèle de Gompertz Makeham :

```
#  Adéquation graphique (linéaire) au modèle de Gompertz-Makeham + modèle linéaire
# on se restreint aux âges de 0 à 100 ans, et on considère l'année 2018
N = E$Total[(E$Year == 2018) & (E$Age < 101)]
D = De$Total[(De$Year == 2018) & (De$Age < 101)]

ages = 0:100 #ages pris en compte dans le modèle
q = D / N  # taux bruts de mortalité (population totale)
plot(ages, q, type = 'l')
```




```
L = length(q)
```

```
## Calcul du modèle :

t = ages + 1 #indices
y = log(q[t + 1] - q[t])
```

#### 2.1.2.1 Ajustement linéaire :

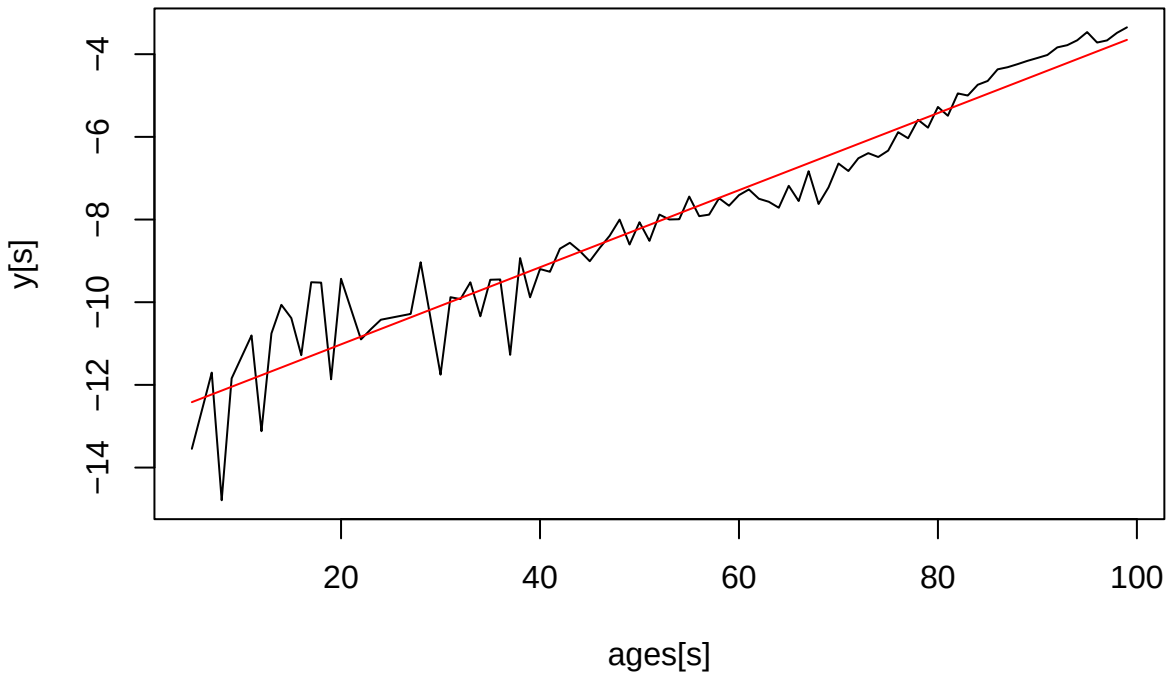
```
## Warning in log(q[t + 1] - q[t]): Production de NaN
```

```
#y


# problème si q n'est pas strictement croissante partout ==> y est infini !
# ==> on garde uniquement les t t.q. y soit fini <=> q[t+1]-q[t]>0
s = which(q[t + 1] - q[t] > 0)
reg1 = lm(y ~ ages, subset = s)
summary(reg1)
```

```
##
## Call:
## lm(formula = y ~ ages, subset = s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

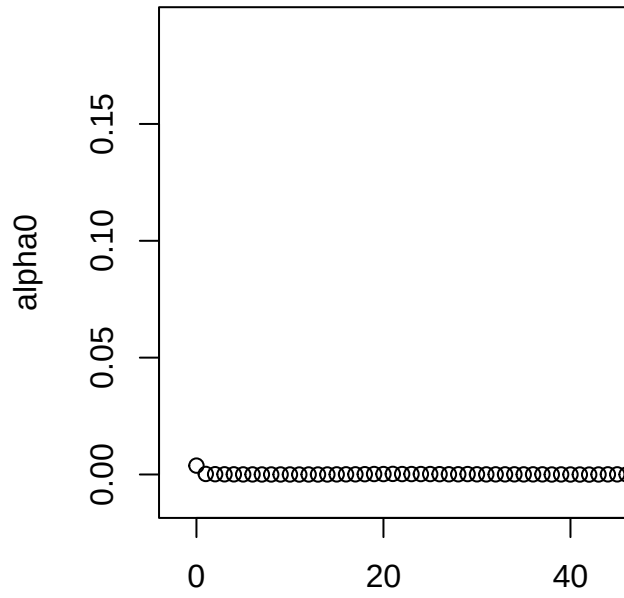
```
## -2.65265 -0.31645  0.07764  0.31218  1.78217
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -12.884162   0.167075  -77.12   <2e-16 ***
## ages          0.093243   0.002762   33.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7001 on 87 degrees of freedom
## Multiple R-squared:  0.9291, Adjusted R-squared:  0.9282
## F-statistic:  1139 on 1 and 87 DF,  p-value: < 2.2e-16
```

```
plot(ages[s], y[s], type ='l')
yf = fitted(reg1, subset = s)
lines(ages[s], yf, col = 'red')
```



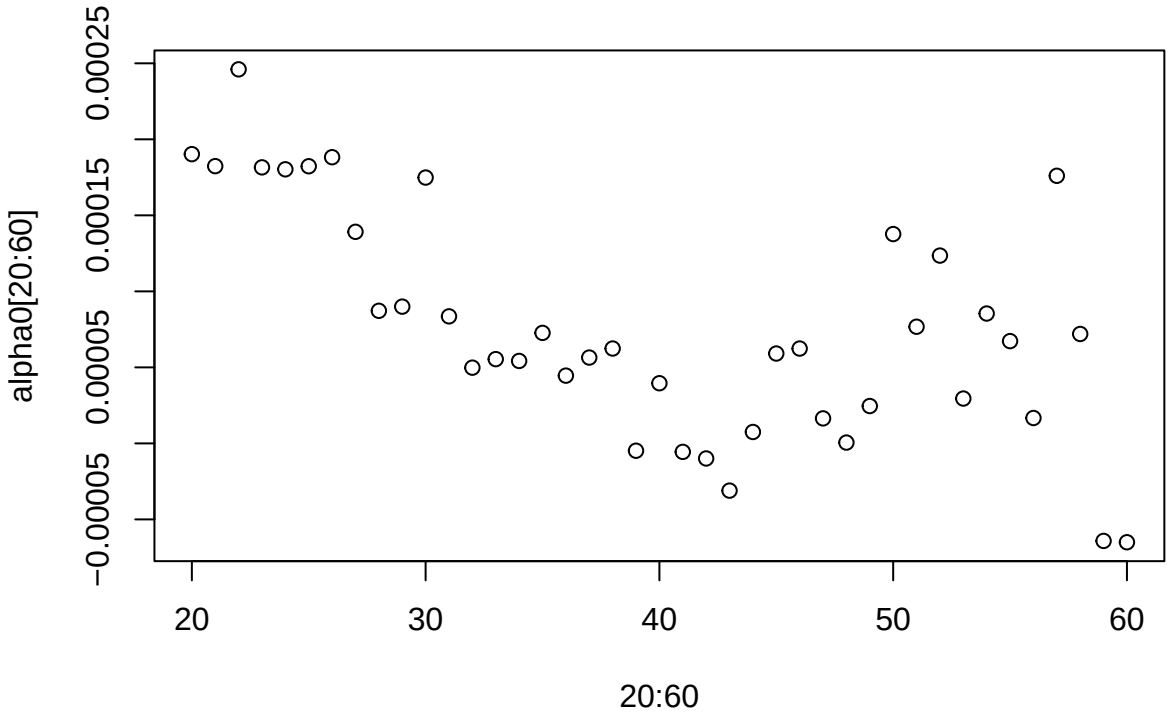

```
## Ajustement du modèle ----
## Ajustement du modèle de Makeham via régression linéaire y= a x + b
a = reg1$coefficients[2]
b = reg1$coefficients[1]

#### on en déduit les paramètres du modèle de G.M. :
gamma = exp(a)
beta = exp(b) * log(gamma) / (gamma - 1) ^ 2
```

```r
# pour alpha, a priori, on pourrait prendre n'importe quel âge.
# Voyons si c'est le cas :
alpha0 = -log(1 - q) - beta / log(gamma) * gamma ^ ages * (gamma - 1)
plot(ages, alpha0)  # variations importantes
```




#### 2.1.2.2 Estimation des paramètres via modélisation linéaire

```r
plot(20:60, alpha0[20:60]) # on se restreint sur un domaine où les variations sont moindres
```




```
alpha = mean(alpha0[25:55])  # difficile de choisir à partir des données...on prend une moyenne (par ex
```

```
param = data.frame(alpha = alpha,
                   beta = beta,
                   gamma = gamma)
param  # paramètres du modèle de G.M.
```

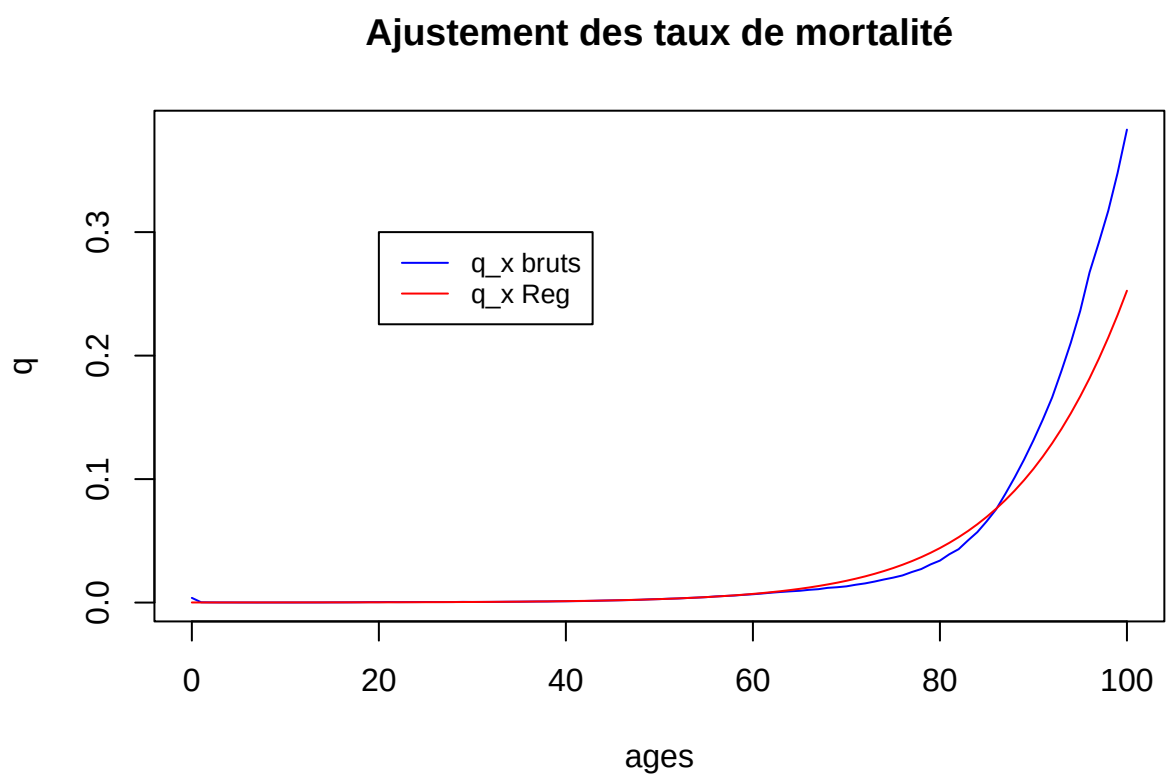
#### 2.1.2.3 Les paramètres finaux du modèles

```
##                     alpha         beta    gamma
## (Intercept) 6.515038e-05 2.477741e-05 1.097728
```

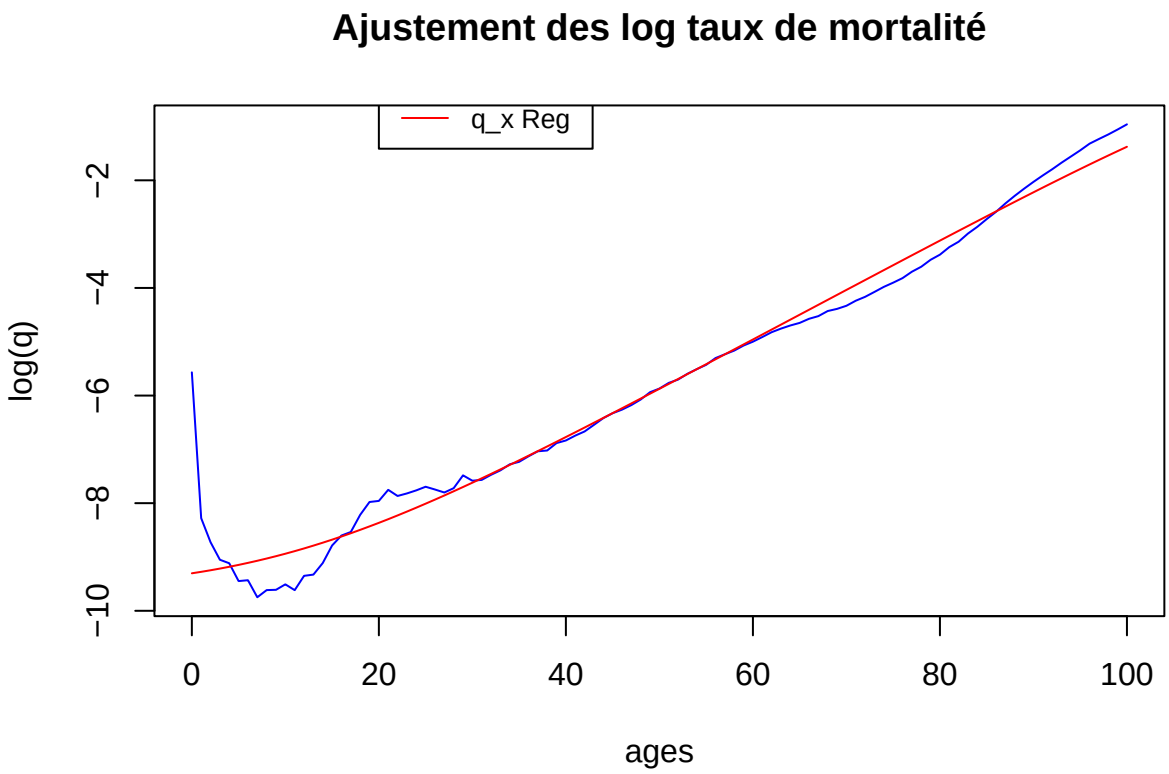
### 2.1.3 Q3 : Comparaison de modèles log taux et taux brute

```
# comparaison entre taux bruts et modèle de Makeham avec ces paramètres
plot(ages,
     q,
     type = 'l',
     main = 'Ajustement des taux de mortalité',
     col = 'blue') # q_x observés
qM1 = 1 - exp(-alpha) * exp(-beta / log(gamma) * gamma ^ ages * (gamma -
                                                                 1))  # q_x donnés par le modèle
lines(ages, qM1, col = 'red')
legend(
  20,
```

```
  0.30,
  legend = c("q_x bruts", "q_x Reg"),
  col = c("blue", "red"),
  lty = 1,
  cex = 0.8
)
```




```
## Comparaison des log taux bruts et modèle G-M :  ----
# comparaison entre log taux bruts et  modèle de Makeham avec ces paramètres
plot(ages,
     log(q),
     type = 'l',
     main = 'Ajustement des log taux de mortalité',
     col = 'blue') # q_x observés
qM1 = 1 - exp(-alpha) * exp(-beta / log(gamma) * gamma ^ ages * (gamma -1))  # q_x donnés par le modèle
lines(ages, log(qM1), col = 'red')
legend(
  20,
  0.30,
  legend = c("q_x bruts", "q_x Reg"),
  col = c("blue", "red"),
  lty = 1,
  cex = 0.8
)
```




# 3 TD3 : Examen 2019

## 3.1 Exercice 1 :

### 3.1.1 Q0 : Importation des données

```
# Importation des données :
expofrance <- read.csv2("DATA/ExposuresFrance2022.csv")
deathfrance <- read.csv2("DATA/DeathsFrance2022.csv")
```

### 3.1.2 Q1 : Calculer l'espérance de vie résiduelle aux âges 0,20,40,60,80 ans pour l'année 2015

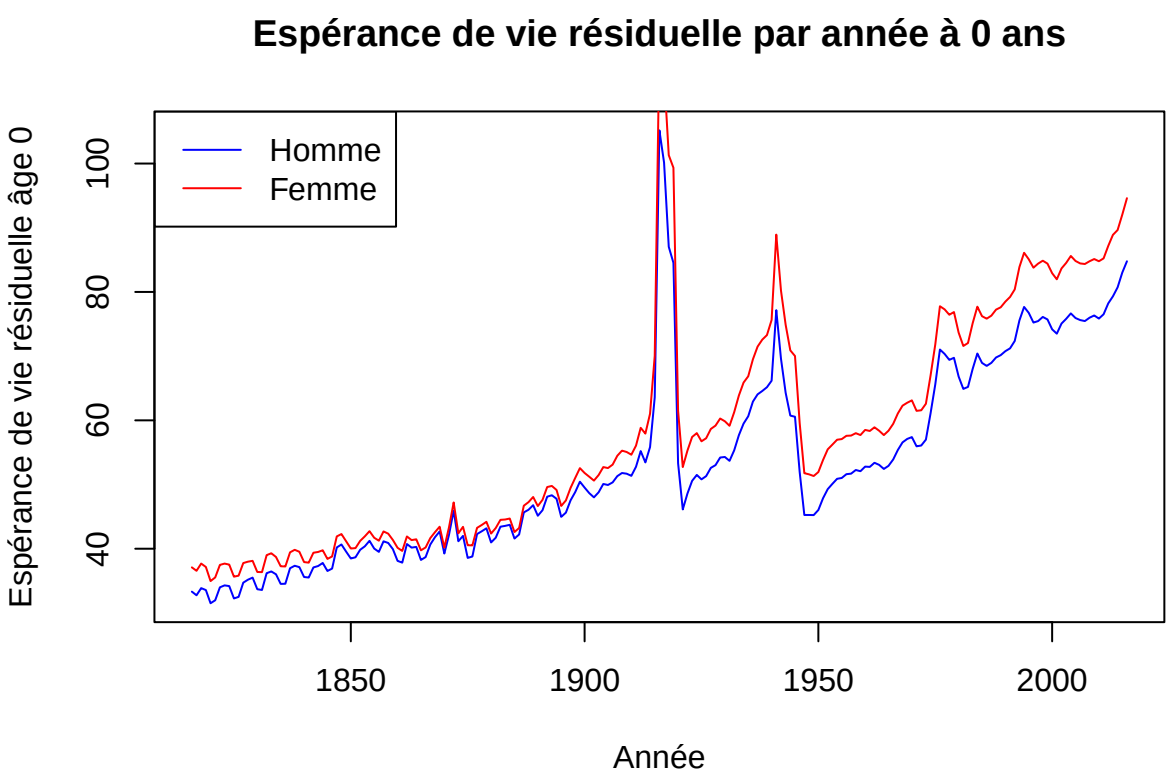
```
# Fonction de l'espérance de de vie résiduelles
esp_vie_resid <- function(table, year) {
  table = table[table$Year == year,]
  table[,3:4] <- apply(table[,3:4], 2, as.numeric)
  esp_age_homme = c()
  esp_age_femme = c()
  for (i in 1:nrow(table)) {
    esp_age_homme[i] <- 1 / table[i,4] * sum(table[(i+1):nrow(table),4])
    esp_age_femme[i] <- 1 / table[i,3] * sum(table[(i+1):nrow(table),3])
  }

  df = as.data.frame(cbind((1:nrow(table)-1),esp_age_homme, esp_age_femme))
  colnames(df) = c("Age", "Homme", "Femme")
```

```
  return(df)
}

esp_vie_resid(expofrance, 2015)[1+c(0,20,40,60,80),]
```

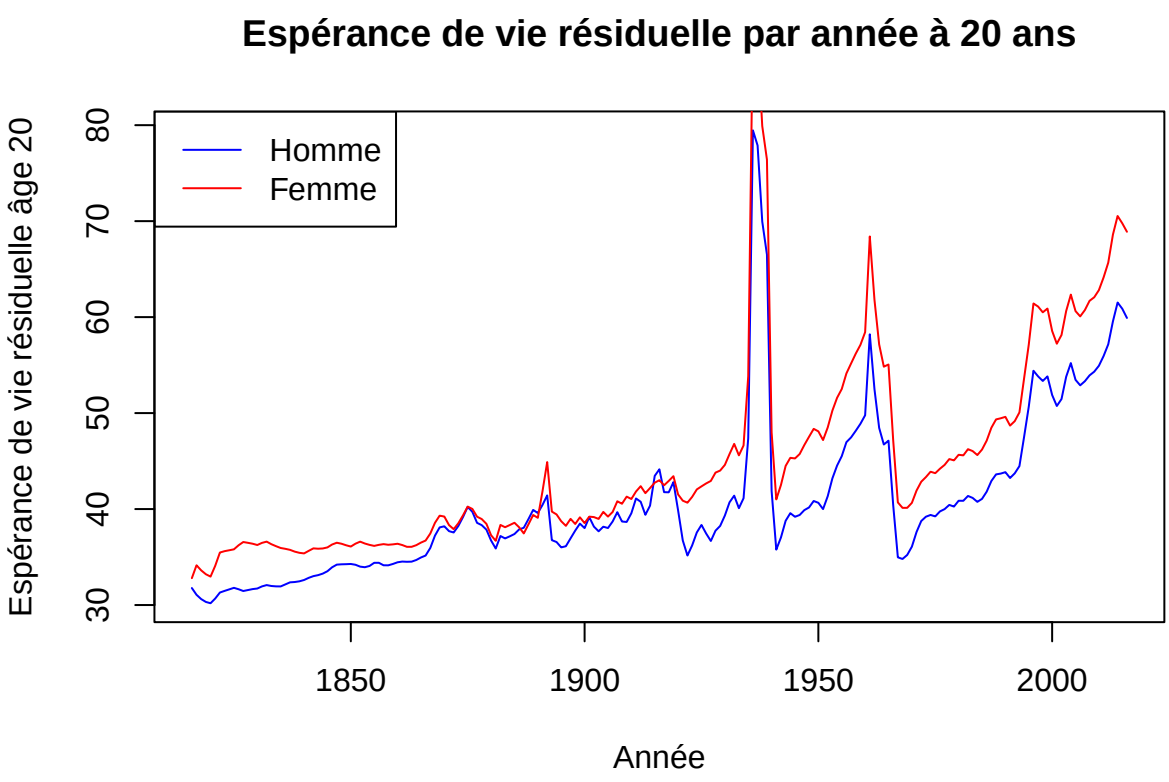
```
##    Age     Homme     Femme
## 1    0 83.011704 92.038737
## 21  20 60.871152 69.774362
## 41  40 37.075480 42.160198
## 61  60 17.341041 20.666607
## 81  80  7.075688  9.529193
```

### 3.1.3 Q2 : Tracer l'espérance de vie résiduelle à la naissance en fonction de l'année d'observation

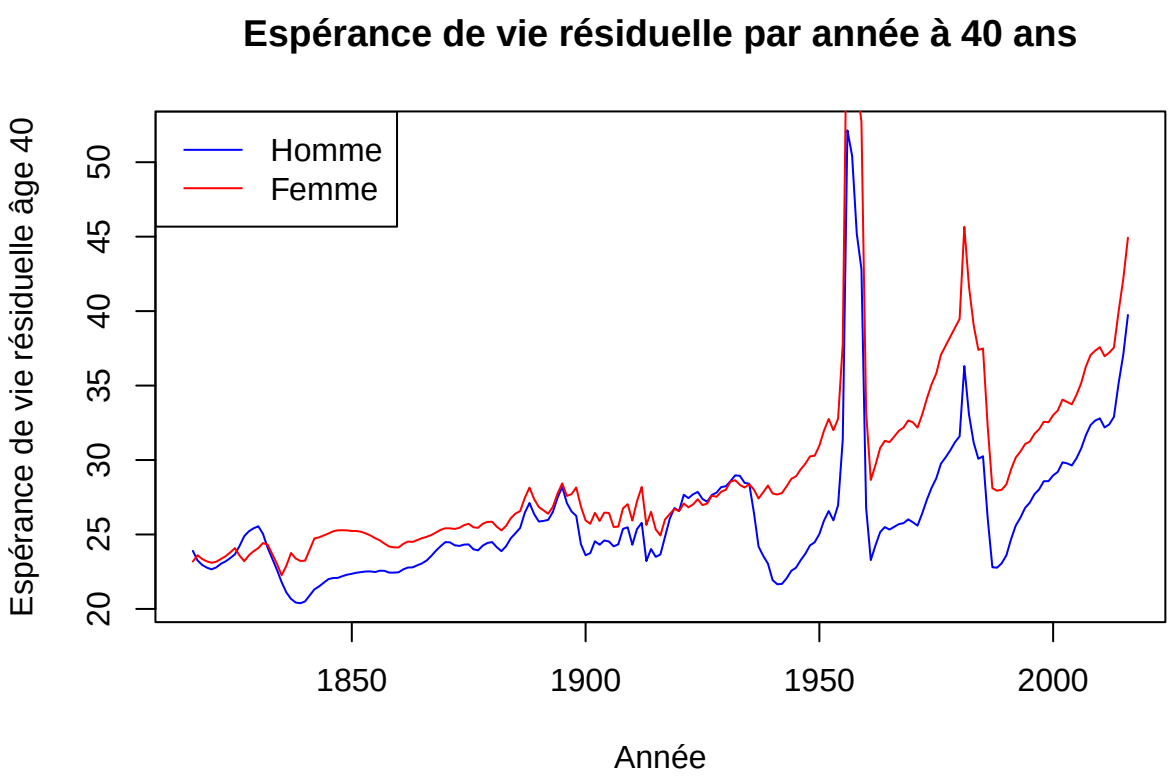
```
n = 2016-1816 +1
esp_0 = as.data.frame(cbind(1816:2016, rep(0,n),rep(0,n)),nrow = n, ncol = 3)
colnames(esp_0) = c("Year", "Homme", "Femme")
esp_20 = as.data.frame(cbind(1816:2016, rep(0,n),rep(0,n)),nrow = n, ncol = 3)
colnames(esp_20) = c("Year", "Homme", "Femme")
esp_40 = as.data.frame(cbind(1816:2016, rep(0,n),rep(0,n)),nrow = n, ncol = 3)
colnames(esp_40) = c("Year", "Homme", "Femme")
esp_60 = as.data.frame(cbind(1816:2016, rep(0,n),rep(0,n)),nrow = n, ncol = 3)
colnames(esp_60) = c("Year", "Homme", "Femme")

for (i in 1816:2016){
  esp_0[i-1815,2:3] = esp_vie_resid(expofrance, i)[1+c(0),2:3]
  esp_20[i-1815,2:3] = esp_vie_resid(expofrance, i)[1+c(20),2:3]
  esp_40[i-1815,2:3] = esp_vie_resid(expofrance, i)[1+c(40),2:3]
  esp_60[i-1815,2:3] = esp_vie_resid(expofrance, i)[1+c(60),2:3]
}

x = 1816:2016
plot(x, esp_0$Homme, type = "l", col = "blue",
     xlab = "Année", ylab = "Espérance de vie résiduelle âge 0",
     main = "Espérance de vie résiduelle par année à 0 ans")
lines(x, esp_0$Femme, col = "red")
legend("topleft", legend = c("Homme", "Femme"), col = c("blue", "red"), lty = 1)
```



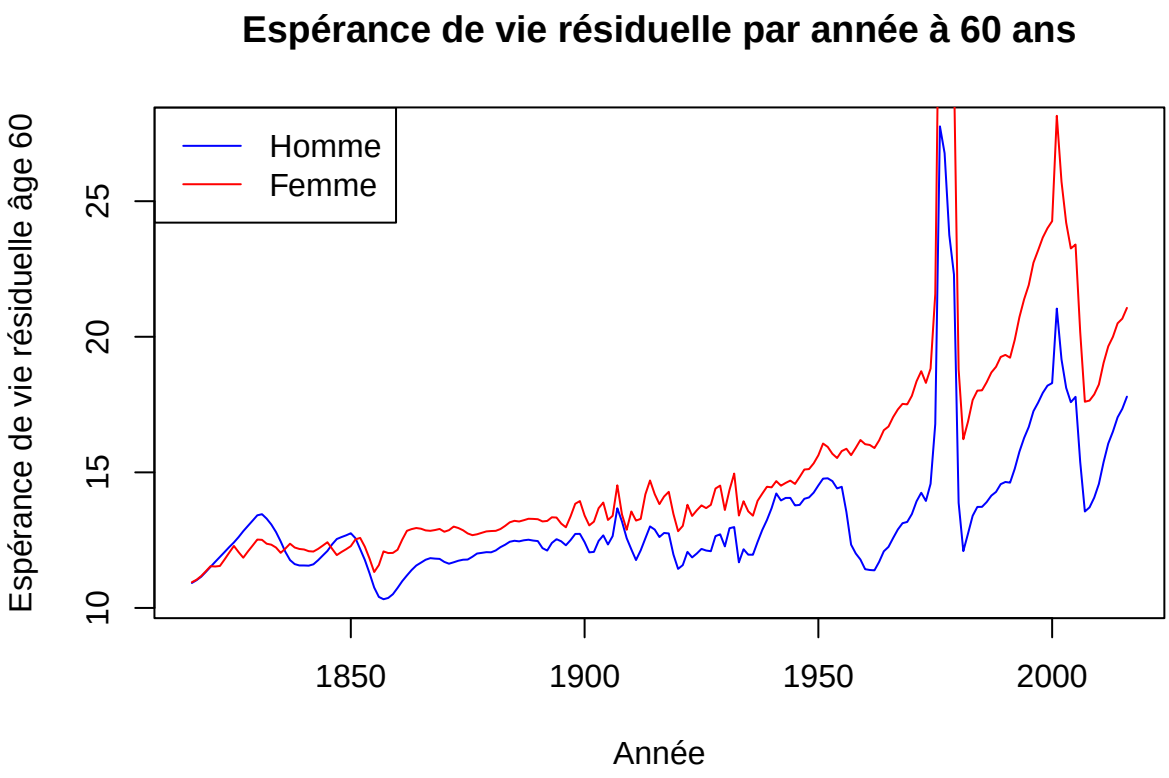

```
X = 1816:2016
plot(X, esp_20$Homme, type = "l", col = "blue",
     xlab = "Année", ylab = "Espérance de vie résiduelle âge 20",
     main = "Espérance de vie résiduelle par année à 20 ans")
lines(X, esp_20$Femme, col = "red")
legend("topleft", legend = c("Homme", "Femme"), col = c("blue", "red"), lty = 1)
```




```
plot(X, esp_40$Homme, type = "l", col = "blue",
     xlab = "Année", ylab = "Espérance de vie résiduelle âge 40",
      main = "Espérance de vie résiduelle par année à 40 ans")
lines(X, esp_40$Femme, col = "red")
legend("topleft", legend = c("Homme", "Femme"), col = c("blue", "red"), lty = 1)
```




```
plot(X, esp_60$Homme, type = "l", col = "blue",
     xlab = "Année", ylab = "Espérance de vie résiduelle âge 60",
     main = "Espérance de vie résiduelle par année à 60 ans")
lines(X, esp_60$Femme, col = "red")
legend("topleft", legend = c("Homme", "Femme"), col = c("blue", "red"), lty = 1)
```




#### 3.1.4 Q3 : Commentaire de courbe

### 3.2 Exercice 2 (13 du cours) :

#### 3.2.1 Q1 : Simulation de l'énoncé

```
lambda = 1  # paramètre de la loi de Ci et Xi
beta = 0.5  # paramètre de la loi de Ci
n = 1000000 # Nombre de simulations
Xi = rexp(n,lambda) # Simulation de Xi
Ci = rexp(n, beta*lambda) # Simulation de Ci

Ti = pmin(Xi,Ci) # Simulation de Ti
Di = Xi <= Ci ## Simulation de Di
```

#### 3.2.2 Q2 : Application de l'estimation

```
# Méthode d'estimations possible
lambda_chap = sum(Di)/sum(Ti)
beta_chap = n/sum(Di) - 1

cat("Lambda estimé : ", lambda_chap , "\n",
    "Beta estimé : ", beta_chap , "\n",
    "Lambda réel : ", lambda , "\n",
    "Beta réel : ", beta , "\n")
```

```
## Lambda estimé :  0.9983694
##  Beta estimé :  0.499383
##  Lambda réel :  1
##  Beta réel :  0.5
```

# 4 Examen 2017 :

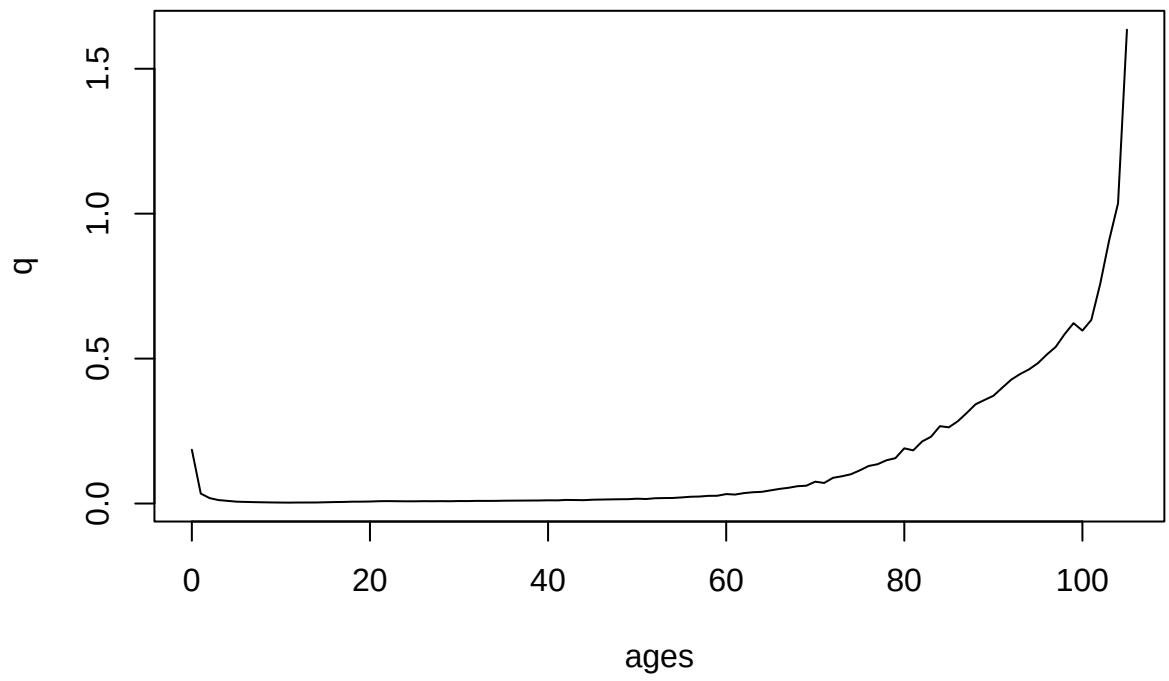
## 4.1 Exercice 1 :

### 4.1.1 Q0 : Importation des données

```r
E <- read.csv2("DATA/ExposuresFrance2022.csv")
D <- read.csv2("DATA/DeathsFrance2022.csv")

TABLE_D <- D[D$Year == 1900,] # nombre de décès
TABLE_E <- E[E$Year == 1900,] # nombre individus
```

## 4.2 Q1 : Taux brute de mortalité

```r
q = as.numeric(TABLE_D$Total)/as.numeric(TABLE_E$Total)
q = q[1:106] # il y a des valeurs manquantes sinon
ages = 0:105
plot(ages, q, type = 'l') # On affiche le graphique
```



```r
# On estime le taux brut de mortalité
t = ages + 1
y = log(q[t+1] - q[t])
```

```
## Warning in log(q[t + 1] - q[t]): Production de NaN
```

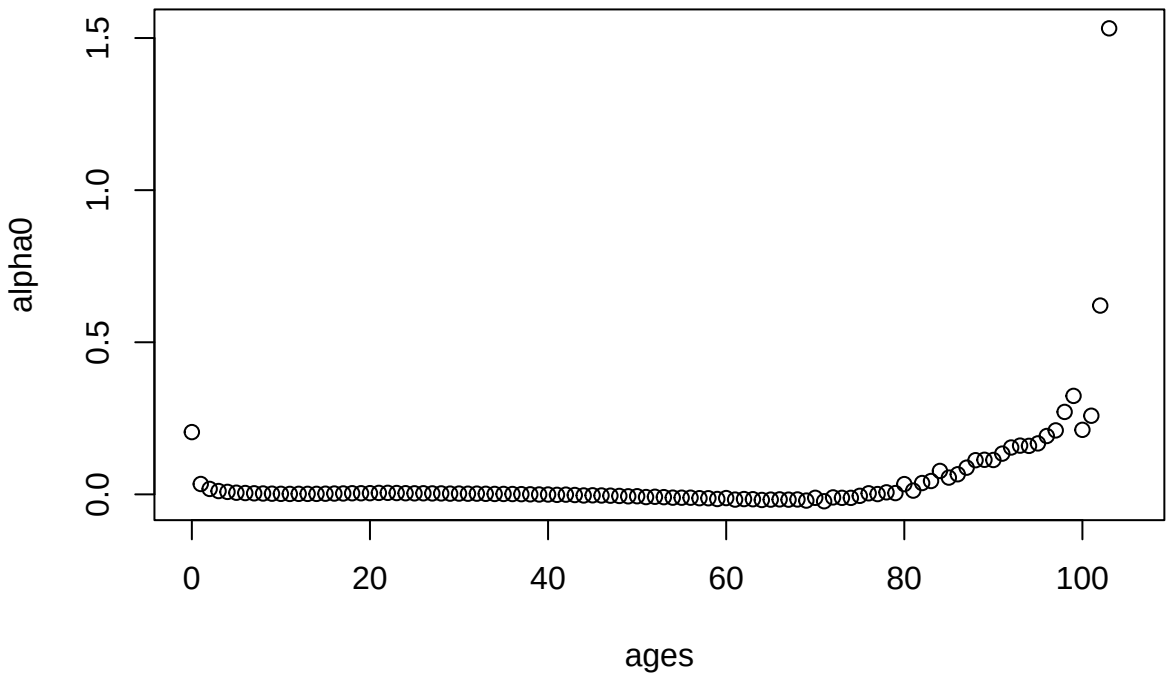
```r
s = which(q[t + 1] - q[t] > 0)
reg1 = lm(y ~ ages, subset = s)
summary(reg1)
```

```
##
## Call:
## lm(formula = y ~ ages, subset = s)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.51275 -0.41928  0.04383  0.47587  2.22722
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.840049   0.253059  -38.88   <2e-16 ***
## ages         0.068274   0.003834   17.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9057 on 75 degrees of freedom
## Multiple R-squared:  0.8087, Adjusted R-squared:  0.8062
## F-statistic: 317.1 on 1 and 75 DF,  p-value: < 2.2e-16
```

```r
# Récupération des paramètres
a = reg1$coefficients[2]
b = reg1$coefficients[1]
#### on en déduit les paramètres du modèle de G.M. :
gamma = exp(a)
beta = exp(b) * log(gamma) / (gamma - 1) ^ 2


# pour alpha, a priori, on pourrait prendre n'importe quel âge.
# Voyons si c'est le cas :
alpha0 = -log(1 - q) - beta / log(gamma) * gamma ^ ages * (gamma - 1)
```

```
## Warning in log(1 - q): Production de NaN
```

```r
plot(ages, alpha0) # variations importantes
```




```
alpha = mean(alpha0[25:60])
param = data.frame(alpha = alpha,
beta = beta,
gamma = gamma)
param
```

```
##                     alpha         beta    gamma
## (Intercept) -0.002836237 0.0007285295 1.070658
```

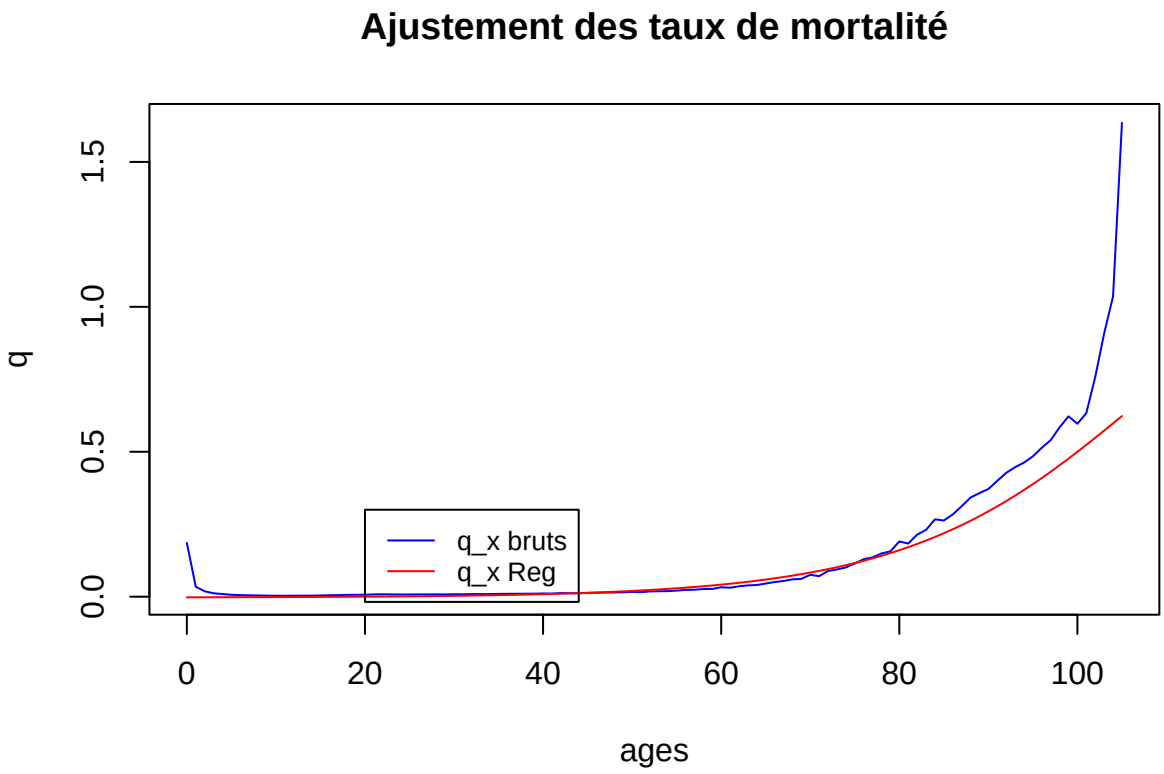
#### 4.2.1 Q2 :

```
plot(ages,
q,
type = 'l',
main = 'Ajustement des taux de mortalité',
col = 'blue') # q_x observés
21
```

```
## [1] 21
```

```
qM1 = 1 - exp(-alpha) * exp(-beta / log(gamma) * gamma ^ ages * (gamma -
1)) # q_x donnés par le modèle
lines(ages, qM1, col = 'red')
legend(
20,
0.30,
legend = c("q_x bruts", "q_x Reg"),
```

```
col = c("blue", "red"),
lty = 1,
cex = 0.8)
```




### Q3 : commentaires

#### 4.2.2 Q4 :

# 5 Fonction déjà établies dans le cours

```
MCmean = function(x, alpha)
{
n = length(x)
m = mean(x)
S = sd(x)
t = qnorm(1 - alpha / 2)
Iinf = m - t * S / sqrt(n)
Isup = m + t * S / sqrt(n)
data.frame(m, Iinf, Isup, S, n, alpha)
}
```

```
# Paramètres estimés :
n = 10^6
a = 7.10^(-3)
b = 10^(-4)
```

```r
gamma = 1.11
l = log(gamma)

# Simulation de X
X = 1 / l * log(1 - l / b * log(1 - runif(n)))

# Simulation de Y
l = log(1.13)
Y = 1 / l * log(1 - l / b * log(1 - runif(n)))

# Estimations
MCmean(X,0.05)
```

#### 5.0.0.1 Q4 :

```
##          m     Iinf     Isup        S       n alpha
## 1 61.15769 61.13402 61.18135 12.07394 1000000  0.05
```

```r
MCmean(Y,0.05)
```

```
##          m     Iinf     Isup        S       n alpha
## 1 53.47691 53.45661 53.49721 10.35596 1000000  0.05
```

```r
MCmean(X>Y, 0.05)
```

```
##          m      Iinf      Isup        S       n alpha
## 1 0.710154 0.7092648 0.7110432 0.453691 1000000  0.05
```

## 5.1 Exercice 2 :

### 5.1.1 Q1 : Points fixes

Si x = 1 on a un point fixe

f(x) = x

### 5.1.2 Q2 :

Analyse –> Stabilité à termes de la série

```r
x0 = 0.12
r = 2
n = 1000

x_n_r <- function(x0,r,n){
  x_n = x0
  for (i in 1:n){
    x_n[i+1] = x_n[i]*exp(r*(1-x_n[i]))
  }
tps = seq(0,n)
plot(tps ,x_n, 'l', main = paste("Pour r = ", r))
}

for (r in seq(0,4,0.25)){
  x_n_r(x0,r,n)
  }
```

# 6 Généralités sur les lois :

## 6.1 La loi exponentielle

### 6.1.1 Définition

La loi exponentielle est une loi de probabilité continue qui est souvent utilisée pour modéliser le temps d'attente entre des événements indépendants qui se produisent à un taux constant.

### 6.1.2 Fonction de densité de probabilité (f.d.p)

Soit $X$ une variable aléatoire suivant une loi exponentielle de paramètre $\lambda > 0$. La fonction de densité de probabilité est donnée par :

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & \text{si } x \geq 0, \\ 0 & \text{si } x < 0. \end{cases}$$

### 6.1.3 Fonction de répartition (F.d.R)

La fonction de répartition $F_X(x)$ de la loi exponentielle est :

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & \text{si } x \geq 0, \\ 0 & \text{si } x < 0. \end{cases}$$

### 6.1.4 Espérance et Variance

Pour une variable aléatoire $X$ suivant une loi exponentielle de paramètre $\lambda$ :

- **Espérance** : $\mathbb{E}(X) = \frac{1}{\lambda}$
- **Variance** : $\mathrm{Var}(X) = \frac{1}{\lambda^2}$

### 6.1.5 Propriétés

- La loi exponentielle est sans mémoire : $P(X > s + t \mid X > t) = P(X > s)$ pour tout $s, t \geq 0$.
- La somme de $n$ variables aléatoires indépendantes et identiquement distribuées selon une loi exponentielle de paramètre $\lambda$ suit une loi Gamma de paramètres $n$ et $\lambda$.

## 6.2 Loi de Weibul :

### 6.2.1 Définition

La loi de Weibull est une loi de probabilité continue utilisée pour modéliser la fiabilité des systèmes et le temps jusqu'à la défaillance.

### 6.2.2 Fonction de Densité de Probabilité (f.d.p.)

Soit $X$ une variable aléatoire suivant une loi de Weibull de paramètres $k > 0$ (paramètre de forme) et $\lambda > 0$ (paramètre d'échelle). La fonction de densité de probabilité est donnée par :

$$f_X(x) = \begin{cases} \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & \text{pour } x \geq 0 \\ 0 & \text{pour } x < 0 \end{cases}$$

### 6.2.3 Fonction de Répartition

La fonction de répartition (F.d.R.) de $X$ est :

$$F_X(x) = P(X \leq x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 - e^{-(x/\lambda)^k} & \text{si } x \geq 0 \end{cases}$$

### 6.2.4 Espérance et Variance

L'espérance (ou moyenne) $\mathbb{E}(X)$ et la variance $\text{Var}(X)$ d'une loi de Weibull de paramètres $k$ et $\lambda$ sont :

$$\mathbb{E}(X) = \lambda\Gamma\left(1 + \frac{1}{k}\right)$$

$$\text{Var}(X) = \lambda^2\left[\Gamma\left(1 + \frac{2}{k}\right) - \left(\Gamma\left(1 + \frac{1}{k}\right)\right)^2\right]$$

où $\Gamma(\cdot)$ est la fonction Gamma.

### 6.2.5 Propriétés

- Modélisation flexible : La loi de Weibull peut modéliser différents comportements de défaillance :
  - $k < 1$ : Taux de défaillance décroissant (période de jeunesse).
  - $k = 1$ : Taux de défaillance constant (équivalent à la loi exponentielle).
  - $k > 1$ : Taux de défaillance croissant (période de vieillissement).
- Relation avec d'autres lois : La loi exponentielle est un cas particulier de la loi de Weibull avec $k = 1$.

### 6.2.6 Transition de la Loi de Weibull à une Loi Exponentielle

La loi de Weibull devient une loi exponentielle lorsque le paramètre de forme $k$ est égal à 1. En effet, pour $k = 1$, la fonction de densité de probabilité de la loi de Weibull devient :

$$f_X(x) = \frac{1}{\lambda}e^{-(x/\lambda)} \quad \text{pour } x \geq 0$$

Ce qui correspond à la fonction de densité de probabilité d'une loi exponentielle de paramètre $\lambda$.

## 6.3 La loi Gamma

### 6.3.1 Définition

La loi Gamma est une loi de probabilité continue utilisée pour modéliser le temps jusqu'à la réalisation de $n$ événements indépendants et identiquement distribués (i.i.d.) suivant une loi exponentielle.

### 6.3.2 Fonction de Densité de Probabilité (f.d.p.)

Soit $X$ une variable aléatoire suivant une loi Gamma de paramètres $\alpha > 0$ (paramètre de forme) et $\beta > 0$ (paramètre d'échelle). La fonction de densité de probabilité est donnée par :

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x} \quad \text{pour } x \geq 0$$

### 6.3.3 Fonction de Répartition

La fonction de répartition (F.d.R.) de $X$ est :

$$F_X(x) = P(X \leq x) = \int_0^x \frac{\beta^\alpha}{\Gamma(\alpha)}t^{\alpha-1}e^{-\beta t}\,dt$$

### 6.3.4 Espérance et Variance

L'espérance (ou moyenne) $\mathbb{E}(X)$ et la variance $\text{Var}(X)$ d'une loi Gamma de paramètres $\alpha$ et $\beta$ sont :

$$\mathbb{E}(X) = \frac{\alpha}{\beta}$$

$$\text{Var}(X) = \frac{\alpha}{\beta^2}$$

### 6.3.5 Propriétés

- Somme d'exponentielles : Si $X_1, X_2, ..., X_n$ sont $n$ variables aléatoires indépendantes de paramètres $\lambda$ suivant une loi exponentielle, alors la somme $S = X_1 + X_2 + ... + X_n$ suit une loi Gamma de paramètres $\alpha = n$ et $\beta = \lambda$.
- Additivité : Si $X$ et $Y$ sont deux variables aléatoires indépendantes suivant des lois Gamma de mêmes paramètres d'échelle $\beta$ et de paramètres de forme $\alpha_1$ et $\alpha_2$ respectivement, alors $X + Y$ suit une loi Gamma de paramètres $\alpha = \alpha_1 + \alpha_2$ et $\beta$.

### 6.3.6 Transition de la Loi Gamma à une Loi Exponentielle

La loi exponentielle est un cas particulier de la loi Gamma. En effet, lorsque le paramètre de forme $\alpha$ est égal à 1, la loi Gamma devient une loi exponentielle. Plus précisément, pour $\alpha = 1$ et $\beta = \lambda$, la fonction de densité de probabilité de la loi Gamma devient :

$$f_X(x) = \frac{\lambda^1}{\Gamma(1)} x^{1-1} e^{-\lambda x} = \lambda e^{-\lambda x} \quad \text{pour } x \geq 0$$

Ce qui correspond à la fonction de densité de probabilité d'une loi exponentielle de paramètre $\lambda$.

## 6.4 Min de loi exponentielle

### 6.4.1 Variable Aléatoire T

Considérons deux variables aléatoires indépendantes $X_1$ et $X_2$, suivant des lois exponentielles de paramètres $\lambda_1$ et $\lambda_2$ respectivement. Soit $T = \min(X_1, X_2)$ la variable aléatoire représentant le minimum de $X_1$ et $X_2$. Pour déterminer la fonction de répartition (F.d.R.) de $T$, nous calculons $P(T \leq t)$.

$$P(T \leq t) = 1 - P(T > t)$$

Puisque $T$ est le minimum de $X_1$ et $X_2$, $T > t$ si et seulement si $X_1 > t$ et $X_2 > t$. Étant donné que $X_1$ et $X_2$ sont indépendants :

$$P(T > t) = P(X_1 > t) \cdot P(X_2 > t)$$

Pour des variables exponentielles, la probabilité que $X_i$ soit supérieure à $t$ est :

$$P(X_i > t) = e^{-\lambda_i t}$$

Ainsi,

$$P(T > t) = e^{-\lambda_1 t} \cdot e^{-\lambda_2 t} = e^{-(\lambda_1 + \lambda_2)t}$$

Donc, la fonction de répartition de $T$ est :

$$F_T(t) = P(T \leq t) = 1 - e^{-(\lambda_1 + \lambda_2)t} \quad \text{pour } t \geq 0$$

Ainsi, $T$ suit une loi exponentielle de paramètre $\lambda_1 + \lambda_2$. En d'autres termes, si $T$ est le minimum entre deux variables aléatoires exponentielles indépendantes $X_1$ et $X_2$, alors $T$ suit une loi exponentielle avec un paramètre qui est la somme des paramètres des deux lois exponentielles initiales.