Modèles de durée : TD et Examens

2024-11-10

Contents

1 Les méthodes semi-paramétriques

1.1 Le modèle de Cox

1.1.1 Lecture des données traitement de la base :

```
library(tidyverse)
Re = read.table("rossi.txt", header = TRUE)
glimpse(Re)

# Suppression de la variable race :
Re1 = Re[, -5]
```

1.1.2 Etude la durée de survie selon la valeur d'une variable. (test de log-Rank)

On regarde si les fonctions de survies des individus discriminés selon les modalités d'une variable, sont significativement similaires.

On effectue pour ça le test du log-rank à l'aide de la fonction Surv du package survival.

$$\begin{cases} H0 : \text{les fonctions de survie sont les mêmes, p-value} \geq 0.05 \\ H1 : \text{les fonctions de survie sont différentes} \end{cases}$$

```
library(survival)
# Test sur la variable financement :
survdiff(Surv(week, arrest) ~ fin, data = Re1)
```

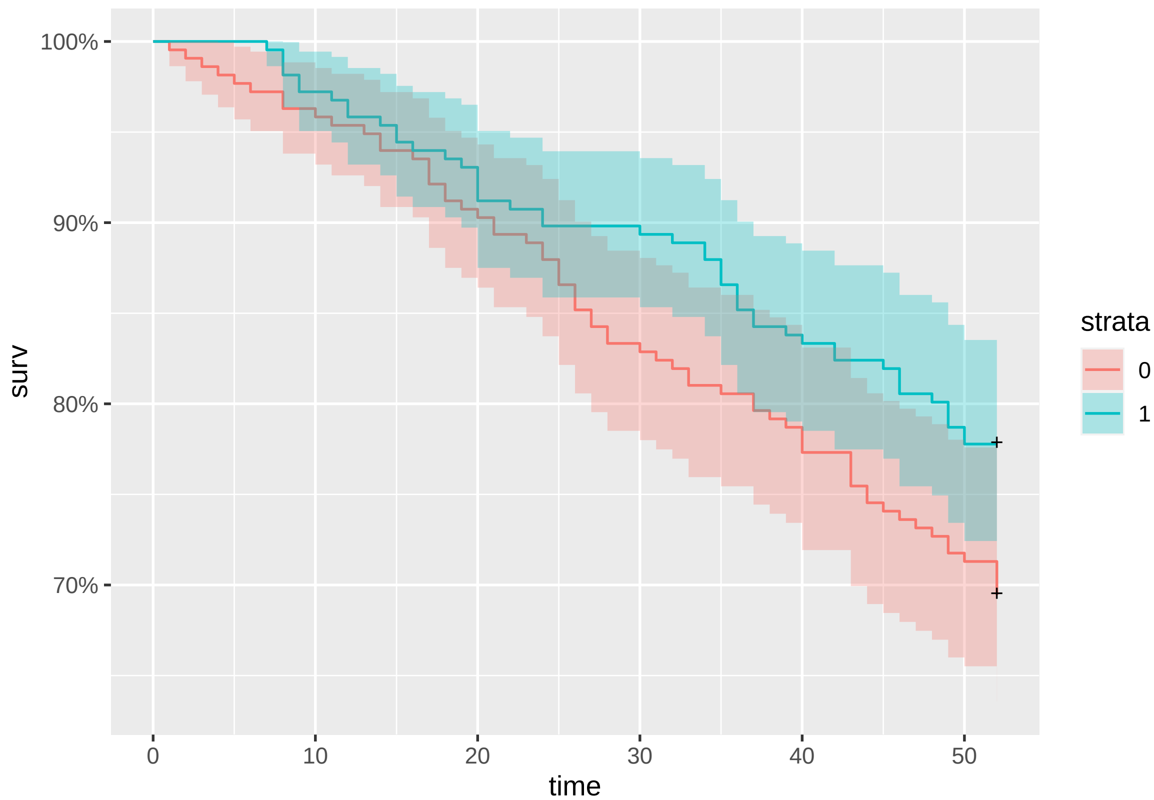
```
Call:
survdiff(formula = Surv(week, arrest) ~ fin, data = Re1)

        N Observed Expected (O-E)^2/E (O-E)^2/V
fin=0 216       66     55.6      1.96      3.84
fin=1 216       48     58.4      1.86      3.84

 Chisq= 3.8  on 1 degrees of freedom, p= 0.05
```

```
# Surv créer un objet avec week le temps de survie et arrest l'indicateur
# d'évènement. Fin est la variable servant à comparer les courbes.
```

1.1.3 Modélisation de Kaplan Meier :

```
# Modélisation de kaplan meier, distinction sur la variable financement
s = survfit(Surv(week, arrest) ~ fin, data = Re1)
```

```r
library(ggfortify)
library(ggplot2)
autoplot(s)
```



1.1.4 Ajustement d'un modèle de Cox :

```r
cox1 = coxph(formula = Surv(week, arrest) ~ fin + age + wexp + mar +
               paro + prio, data = Re1)
summary(cox1)
```

```
Call:
coxph(formula = Surv(week, arrest) ~ fin + age + wexp + mar +
    paro + prio, data = Re1)

  n= 432, number of events= 114

         coef exp(coef) se(coef)      z Pr(>|z|)
fin  -0.36554   0.69382  0.19090 -1.915  0.05552 .
age  -0.05633   0.94523  0.02189 -2.573  0.01007 *
wexp -0.15699   0.85471  0.21208 -0.740  0.45916
mar  -0.47130   0.62419  0.38027 -1.239  0.21520
paro -0.07792   0.92504  0.19530 -0.399  0.68991
prio  0.08966   1.09380  0.02871  3.123  0.00179 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     exp(coef) exp(-coef) lower .95 upper .95
```

```
fin     0.6938     1.4413    0.4773    1.0087
age     0.9452     1.0579    0.9055    0.9867
wexp    0.8547     1.1700    0.5640    1.2952
mar     0.6242     1.6021    0.2962    1.3152
paro    0.9250     1.0810    0.6308    1.3564
prio    1.0938     0.9142    1.0340    1.1571

Concordance= 0.639  (se = 0.027 )
Likelihood ratio test= 32.14  on 6 df,   p=2e-05
Wald test            = 30.79  on 6 df,   p=3e-05
Score (logrank) test = 32.28  on 6 df,   p=1e-05
```

Explication du test :

$$\begin{cases} H0: \ \beta_j = 0, \text{Pr(>|z|), prob(|U|> z), où U} \quad \text{N(0,1)} \\ H1: beta_j \neq 0, \text{p-value} \leq 0.05 \end{cases}$$

Le se(coef) correspond au sqrt(var(beta)). On en déduit que les variables significatives sont l'âge et le prio.

1.1.5 Graphique de la fonction de survie :

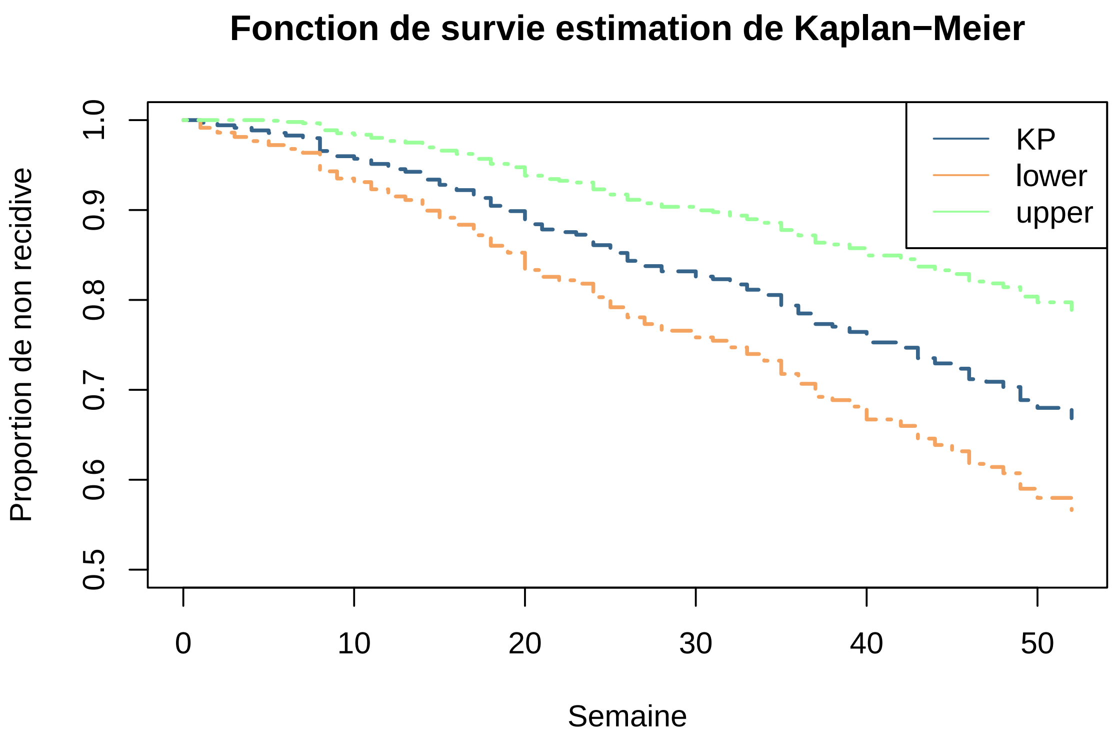
Dans le cadre des fonction de Kaplan Meier, Aalen par défaut les covariables sont fixées à la valeur moyenne.

```r
kpmr = survfit(cox1) # Fonction de survie de Kaplan-Meier pour le modèle de cox

summary(kpmr)
```

```
Call: survfit(formula = cox1)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    1    432       1    0.997 0.00292        0.991        1.000
    2    431       1    0.994 0.00419        0.986        1.000
    3    430       1    0.991 0.00520        0.981        1.000
    4    429       1    0.989 0.00609        0.977        1.000
    5    428       1    0.986 0.00690        0.972        0.999
    6    427       1    0.983 0.00766        0.968        0.998
    7    426       1    0.980 0.00838        0.964        0.997
    8    425       5    0.966 0.01165        0.943        0.989
    9    420       2    0.960 0.01285        0.935        0.985
   10    418       1    0.957 0.01343        0.931        0.984
   11    417       2    0.951 0.01459        0.923        0.980
   12    415       2    0.945 0.01573        0.915        0.977
   13    413       1    0.943 0.01629        0.911        0.975
   14    412       3    0.934 0.01794        0.899        0.970
   15    409       2    0.928 0.01903        0.891        0.966
   16    407       2    0.922 0.02009        0.884        0.962
   17    405       3    0.913 0.02167        0.872        0.957
   18    402       3    0.905 0.02322        0.860        0.951
   19    399       2    0.899 0.02424        0.853        0.948
   20    397       5    0.884 0.02674        0.833        0.938
   21    392       2    0.878 0.02772        0.826        0.934
   22    390       1    0.875 0.02820        0.822        0.933
   23    389       1    0.873 0.02868        0.818        0.931
   24    388       4    0.861 0.03057        0.803        0.923
   25    384       3    0.852 0.03196        0.792        0.917
   26    381       3    0.843 0.03332        0.781        0.911
```

```
   27    378       2    0.838 0.03422        0.773         0.907
   28    376       2    0.832 0.03512        0.766         0.904
   30    374       2    0.826 0.03601        0.758         0.900
   31    372       1    0.823 0.03645        0.755         0.898
   32    371       2    0.817 0.03732        0.747         0.894
   33    369       2    0.811 0.03819        0.740         0.890
   34    367       2    0.805 0.03906        0.732         0.886
   35    365       4    0.794 0.04077        0.718         0.878
   36    361       3    0.785 0.04202        0.707         0.872
   37    358       4    0.773 0.04365        0.692         0.864
   38    354       1    0.770 0.04405        0.689         0.862
   39    353       2    0.764 0.04485        0.681         0.858
   40    351       4    0.753 0.04641        0.667         0.849
   42    347       2    0.747 0.04717        0.660         0.845
   43    345       4    0.735 0.04867        0.646         0.837
   44    341       2    0.729 0.04941        0.639         0.833
   45    339       2    0.724 0.05014        0.632         0.829
   46    337       4    0.712 0.05157        0.618         0.820
   47    333       1    0.709 0.05191        0.614         0.818
   48    332       2    0.703 0.05261        0.607         0.814
   49    330       5    0.689 0.05430        0.590         0.804
   50    325       3    0.680 0.05527        0.580         0.797
   52    322       4    0.668 0.05653        0.566         0.789
```

```r
plot(
  kpmr,
  ylim = c(0.5, 1),
  lty = 5,
  xlab = 'Semaine',
  ylab = 'Proportion de non recidive',
  main = 'Fonction de survie estimation de Kaplan-Meier',
  col = palette_couleur[1:3],
  lwd = 2)

legend(
  "topright",  # Position de la légende
  lty = 1,
  cex = 1,
  legend = c("KP", "lower", "upper"),
  col = palette_couleur[1:3])
```

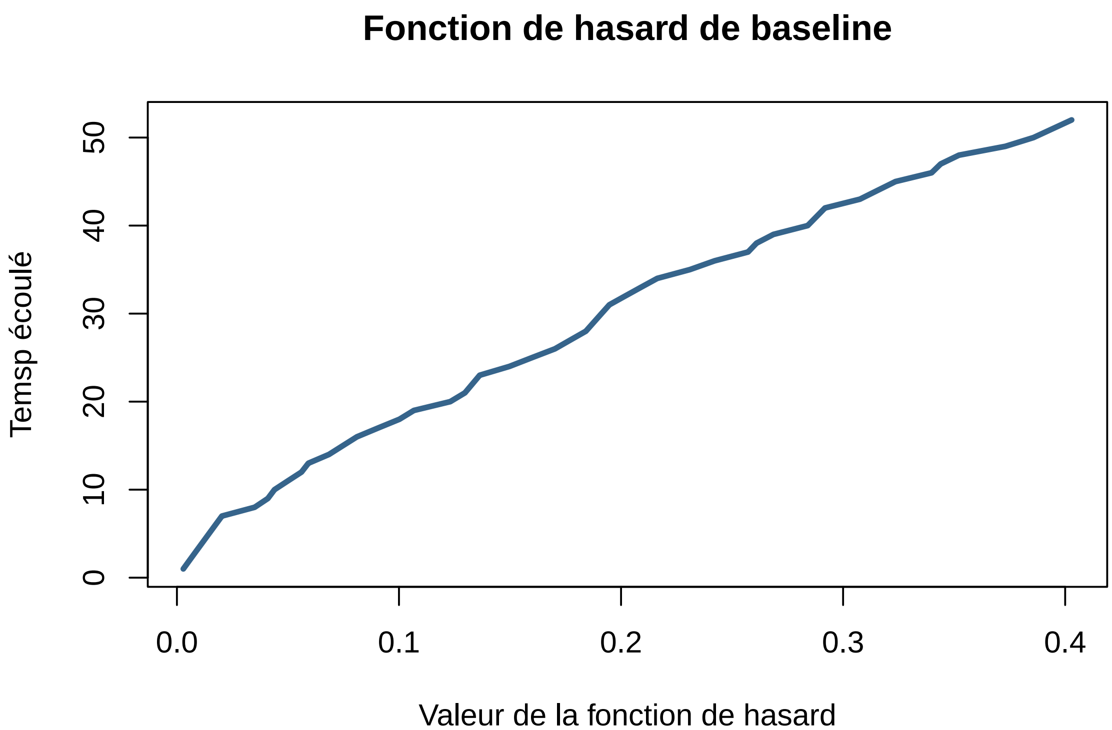


1.1.6 Fonction de hasard cumulée avec l'estimateur de Breslow :

Interprétation Intuitive:

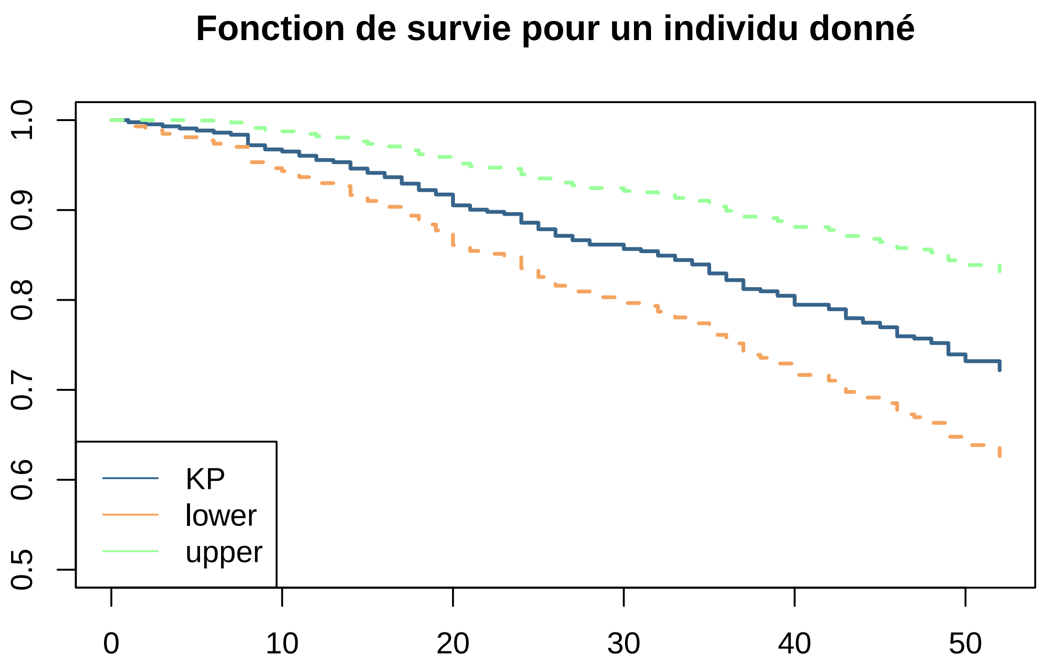
Taux Instantané: La fonction de hasard représente le taux instantané de survenue de l'événement à un moment donné.
Par exemple, si h(t)=0.05h(t)=0.05 à t=10t=10 semaines, cela signifie que le taux de survenue de l'événement à 10 semaines est de 5% par unité de temps.
Conditionnelle à la Survie: La fonction de hasard est conditionnelle à la survie jusqu'à ce moment. Elle ne prend en compte que les individus qui n'ont pas encore subi l'événement.

```
plot(
  basehaz(cox1),
  main = 'Fonction de hasard de baseline',
  xlab = 'Valeur de la fonction de hasard',
  ylab = "Temsp écoulé",
  type = 'l',
  col = palette_couleur[1],
  lwd = 3
)
```



1.1.7 Fonction survie pour l'individu ayant les caractéristiques du premier individu :

```
# plot(survfit(cox1, newdata = Re1)) # fonction de survie pour tous les individus
# title("Fonction de survie pour tous les individus")

plot(survfit(cox1, newdata = Re1[1, ]),
     main = "Fonction de survie pour un individu donné",
     col = palette_couleur[1:3],
     ylim = c(0.5,1),
     lwd = 2,
     lty = 1)

legend("bottomleft",
       lty = 1,
       cex = 1,
       legend = c("KP", "lower", "upper"),
       col = palette_couleur[1:3])
```

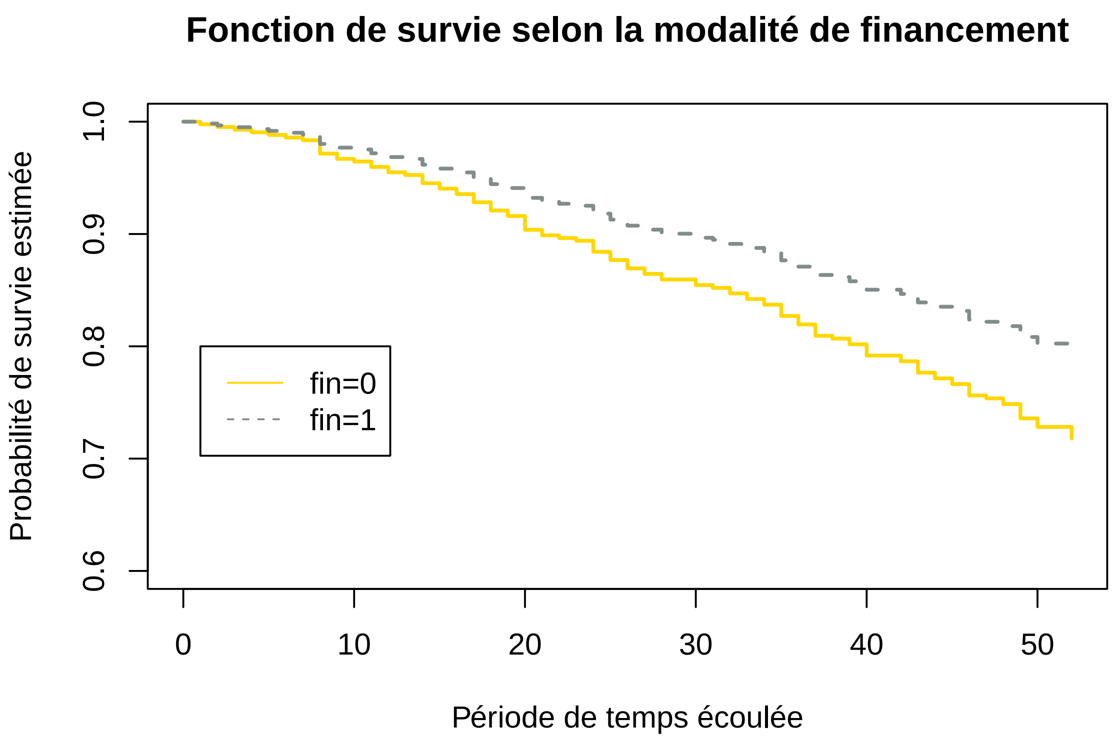


1.1.8 Etude de l'effet d'une covariable (les autres étant fixées) :

Exemple : effet de la var "financement" (0 ou 1) On fixe les autres à leur valeur moyenne.

```
ReFin = data.frame(
  fin = c(0, 1),
  age = rep(mean(Re1$age), 2),
  wexp = rep(mean(Re1$wexp), 2),
  mar = rep(mean(Re1$mar), 2),
  paro = rep(mean(Re1$paro), 2),
  prio = rep(mean(Re1$prio), 2)
)

plot(
  survfit(cox1, newdata = ReFin),
  lty = c(1, 2),
  ylim = c(.6, 1),
  col = palette_couleur[4:5],
  lwd = 2,
  main = "Fonction de survie selon la modalité de financement",
  ylab = "Probabilité de survie estimée",
  xlab = "Période de temps écoulée"
)
legend(
  1,
  0.8,
  legend = c("fin=0", "fin=1"),
  lty = c(1, 2),
```

```
  col =  palette_couleur[4:5]
)
```



1.1.9 Sélection de variable une à une :

Remarque : on peut faire de la sélection de variables en enlevant de façon itérative celles expliquant le moins (p-value la plus forte) exemple :

```
cox2= coxph(formula=Surv(week,arrest)~fin+age+wexp+mar+prio,data=Re1)
summary(cox2)
```

```
Call:
coxph(formula = Surv(week, arrest) ~ fin + age + wexp + mar +
    prio, data = Re1)

  n= 432, number of events= 114

         coef exp(coef) se(coef)      z Pr(>|z|)
fin  -0.36094   0.69702  0.19052 -1.894   0.0582 .
age  -0.05536   0.94614  0.02172 -2.549   0.0108 *
wexp -0.16039   0.85181  0.21201 -0.757   0.4493
mar  -0.47935   0.61919  0.37989 -1.262   0.2070
prio  0.09134   1.09564  0.02840  3.216   0.0013 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     exp(coef) exp(-coef) lower .95 upper .95
fin     0.6970     1.4347    0.4798    1.0126
```

```
age      0.9461     1.0569    0.9067    0.9873
wexp     0.8518     1.1740    0.5622    1.2906
mar      0.6192     1.6150    0.2941    1.3037
prio     1.0956     0.9127    1.0363    1.1583

Concordance= 0.641  (se = 0.027 )
Likelihood ratio test= 31.98  on 5 df,   p=6e-06
Wald test            = 30.73  on 5 df,   p=1e-05
Score (logrank) test = 32.2  on 5 df,   p=5e-06
```

Test hypothèse de Hasard Proportionnel : (proportionnalité des risques)

$$\begin{cases} H0 : \text{les résidus sont indépendants du temps} \\ H1 : \text{les résidus dépendent du temps} \end{cases}$$

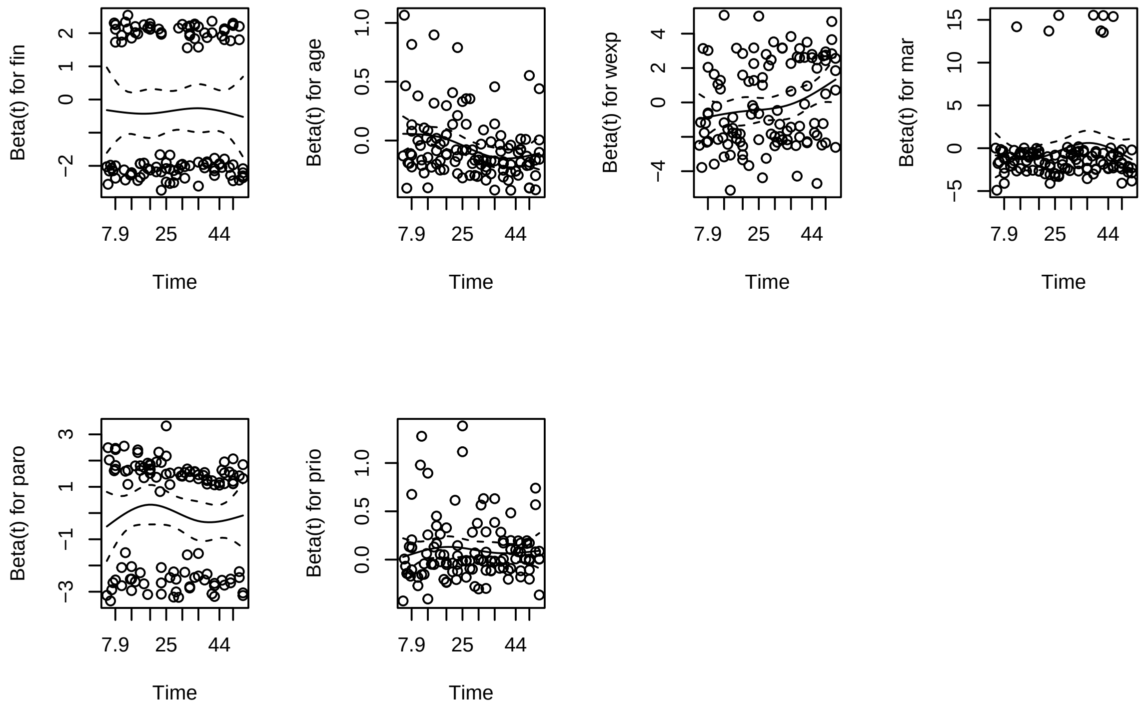
Explication : Si H0 est rejetée, alors les résidus dépendent du temps

1.1.10 Test de hasard proportionnel, les résidus de Schoenfeld

```r
res = cox.zph(cox1)
res
```

```
         chisq df     p
fin     0.0621  1 0.803
age     5.9161  1 0.015
wexp    4.2983  1 0.038
mar     1.0207  1 0.312
paro    0.0140  1 0.906
prio    0.5254  1 0.469
GLOBAL 16.4474  6 0.012
```

```r
# Représentation graphique
par(mfrow = c(2, 4))
plot(res)
```



2 Les méthodes non-paramétriques

2.1 La méthode de Kaplan meier :

2.1.1 Génération de la base et importation des données

On créer une base de données avec des observations censurées.

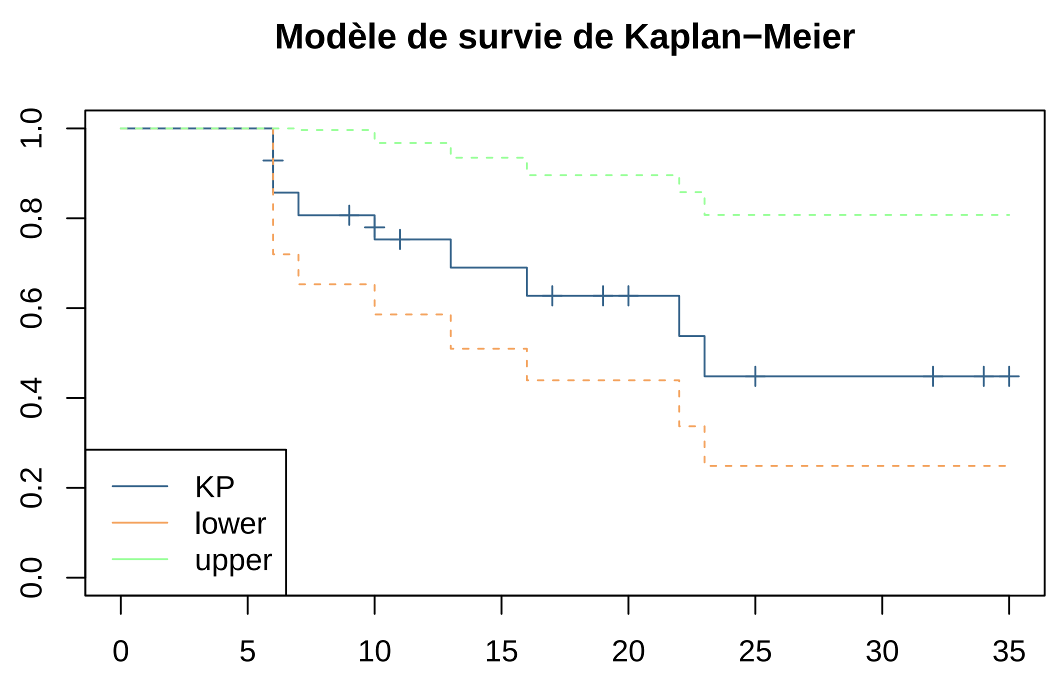
```
library(survival)
tempsGMP = c(rep(6, 4), 7, 9, 10, 10, 11, 13, 16, 17, 19, 20, 22, 23, 25, 32,
             32, 34, 35) # liste des observations
finGMP = c(1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, rep(0, 5))
# indication des obs censurées
donnF = Surv(tempsGMP, finGMP)
head(donnF)
```

```
[1] 6   6   6   6+ 7   9+
```

2.1.2 Ajustement d'un modèle de survie avec la méthode de Kaplan Meier :

```
survKM = survfit(donnF ~ 1,
                 data = donnF,
                 type = "kaplan-meier",
                 error = "greenwood")

# Graphique de la fonction de survie moyenne :

plot(survKM, mark.time = TRUE, col = palette_couleur[1:3])
```

```r
title("Modèle de survie de Kaplan-Meier")
legend("bottomleft",
       lty = 1,
       cex = 1,
       legend = c("KP", "lower", "upper"),
       col = palette_couleur[1:3])
```




```r
# Intervalle de confiance et valeur modélisée pour l'individu 10 :
IC_KM = round(c(survKM$lower[10], survKM$surv[10], survKM$upper[10]),4)
```

2.2 Le modèle de Fleming-Harrington :

2.2.1 Modèle de Fleming-Harrington, intervalle méthode Tsiatis :

```r
# Par defaut, intervalle de confiance : conf.type='log' :
survFH = survfit(donnF ~ 1,
                 data = donnF,
                 type = "fleming-harrington",
                 error = "tsiatis")

# Intervalle de confiance et valeur modélisée pour l'individu 10 :
IC_FH = round(c(survFH$lower[10], survFH$surv[10], survFH$upper[10]),4)
```

2.2.2 Modèle de Fleming-Harrington, intervalle méthode delta :

```r
survFHdelta = survfit(
  donnF ~ 1,
```

```r
  data = donnF,
  type = "fleming-harrington",
  error = "tsiatis",
  conf.type = "plain")

IC_FHdelta = round(c(survFHdelta$lower[10], survFHdelta$surv[10], survFHdelta$upper[10]),4)
```

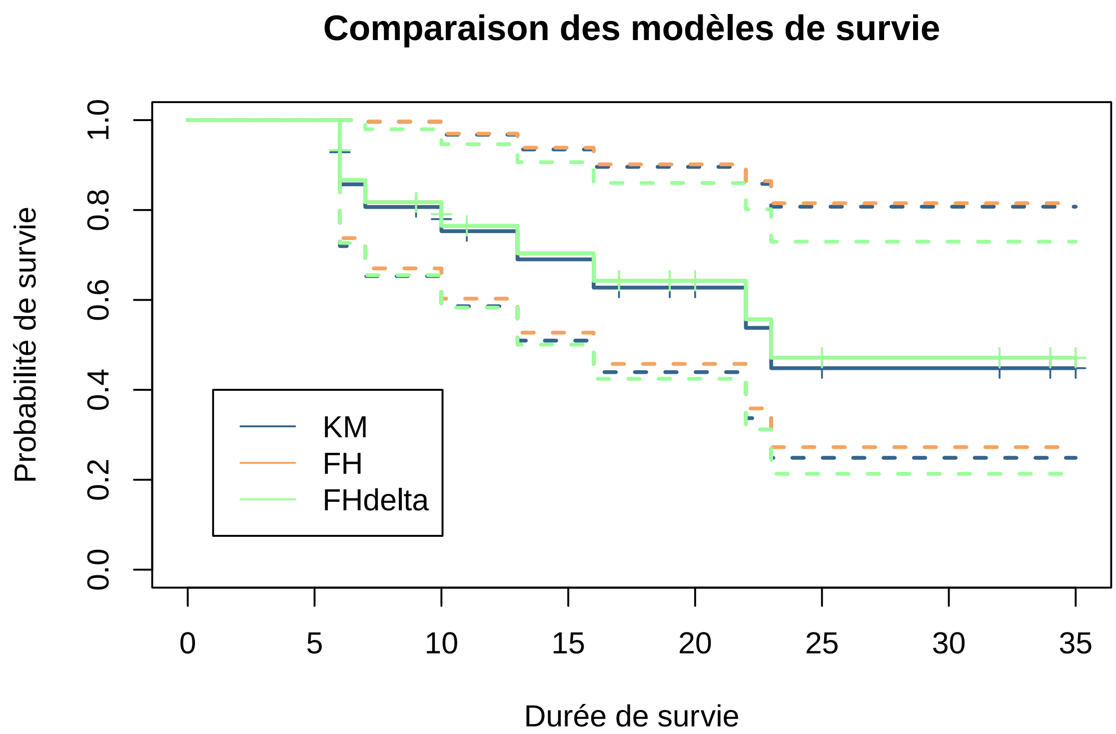
2.2.3 Comparaison des résultats sur l'estimation du 10e individu de la base :

```r
#Comparaison des modèles pour le 10e individu de la base
dt = data.frame(KM = IC_KM, FH = IC_FH, FHdelta = IC_FHdelta)
rownames(dt) = c("lower","pred","upper")
dt
```

```
          KM     FH FHdelta
lower 0.4394 0.4577  0.4246
pred  0.6275 0.6424  0.6424
upper 0.8960 0.9016  0.8601
```

2.2.4 Représentation graphiques des trois modèles :

```r
# Graphiques des trois modèles :
plot(
  survKM,
  mark.time = TRUE,
  col = palette_couleur[1],
  lwd = 2,
  xlab = "Durée de survie",
  ylab = "Probabilité de survie"
)
lines(survFH,
      mark.time = TRUE,
      col = palette_couleur[2],
      lwd = 2)
lines(survFHdelta,
      mark.time = TRUE,
      col = palette_couleur[3],
      lwd = 2)
title("Comparaison des modèles de survie")
legend(
  1,
  0.4,
  lty = 1,
  cex = 1,
  legend = c("KM", "FH", "FHdelta"),
  col = palette_couleur[1:3]
)
```



2.3 Estimation par des lois usuelles :

2.3.1 Estimation de la loi de X par une loi de Weibull :

```
survweib = survreg(donnF ~ 1, dist = "weibull")
survweib
```

```
Call:
survreg(formula = donnF ~ 1, dist = "weibull")

Coefficients:
(Intercept)
   3.519429

Scale= 0.7386973

Loglik(model)= -41.7   Loglik(intercept only)= -41.7
n= 21
```

2.3.2 Estimation de la loi de X par une loi exponentielle :

```
theta = sum(finGMP) / sum(tempsGMP)
theta
```

```
[1] 0.02506964
```

```
survexp = survreg(donnF ~ 1, dist = "exponential")
lambda = exp(-survexp$coefficients)
```

```
lambda
```

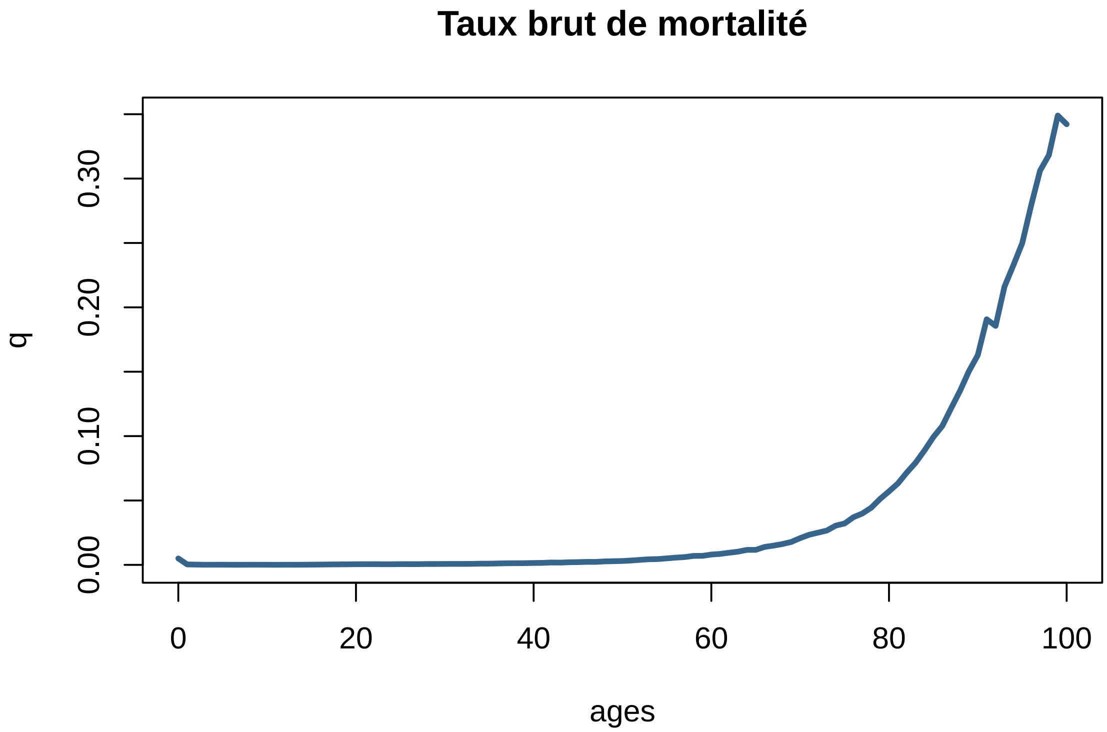
```
(Intercept)
 0.02506964
```

3 Examen 2018 :

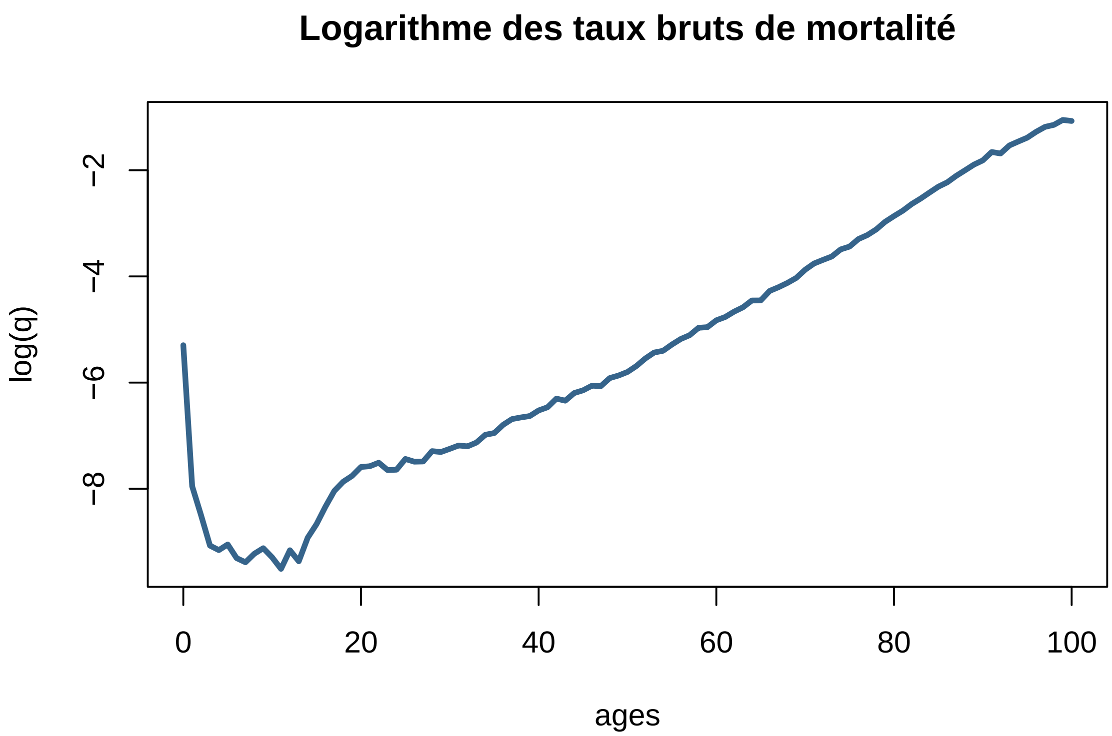
3.1 Exercice 2 :

3.1.1 Importation des données et traitement de la base :

```
library(StMoMo)
d = EWMaleData
De = d$Dxt # décès

ages = d$ages
annees = d$years

Ex = EWMaleData$Ext  # Expositions en milieu d'années
Lx = Ex + De / 2 # Exposition en début d'année (approximation)

# Calcul des taux de mortalité bruts pour 2011 :
q = De[, "2011"] / Lx[, "2011"] # taux bruts

plot(
  ages,
  q,
  type = 'l',
  main = "Taux brut de mortalité",
  col = palette_couleur[1],
  lwd = 3
)
```




```r
plot(
  ages,
  log(q),
  type = 'l',
  main = "Logarithme des taux bruts de mortalité",
  col = palette_couleur[1],
  lwd = 3
)
```



3.1.2 Calibration d'un modèle de Makeham-Gompertz :

3.1.2.1 Utilisation du package fmsb : Utilisation de la fonction fitGm pour calibrer le modèle $h(x) = C + A \times exp(\beta_x)$

Avec la fonction fitGm on peut faire le lien avec l'autre paramétrage du type :

$h(x) = \alpha + \beta \times \gamma^x$ où x représente l'âge.

```
library(fmsb)
fit = fitGM(data = q)

A = fit[1]
B = fit[2]
C = fit[3]
cat("Modélisation fitGM :  \n")
```

```
Modélisation fitGM :
```

```
c(A, B, C)
```

```
[1] 1.742762e-05 1.022779e-01 1.586628e-04
```

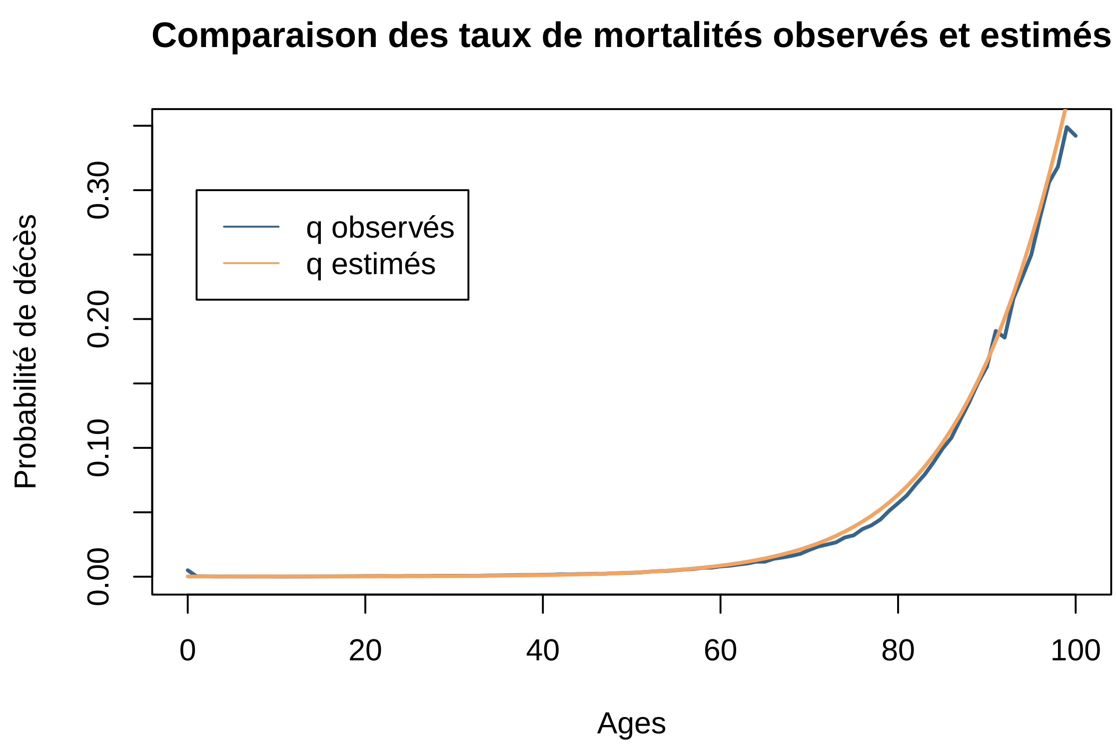
```
# Lien avec l'autre paramétrage :
alpha2 = C
beta2 = A
gamma2 = exp(B)
cat("Apha, Beta, Gamma : \n")
```

```
Apha, Beta, Gamma :
```

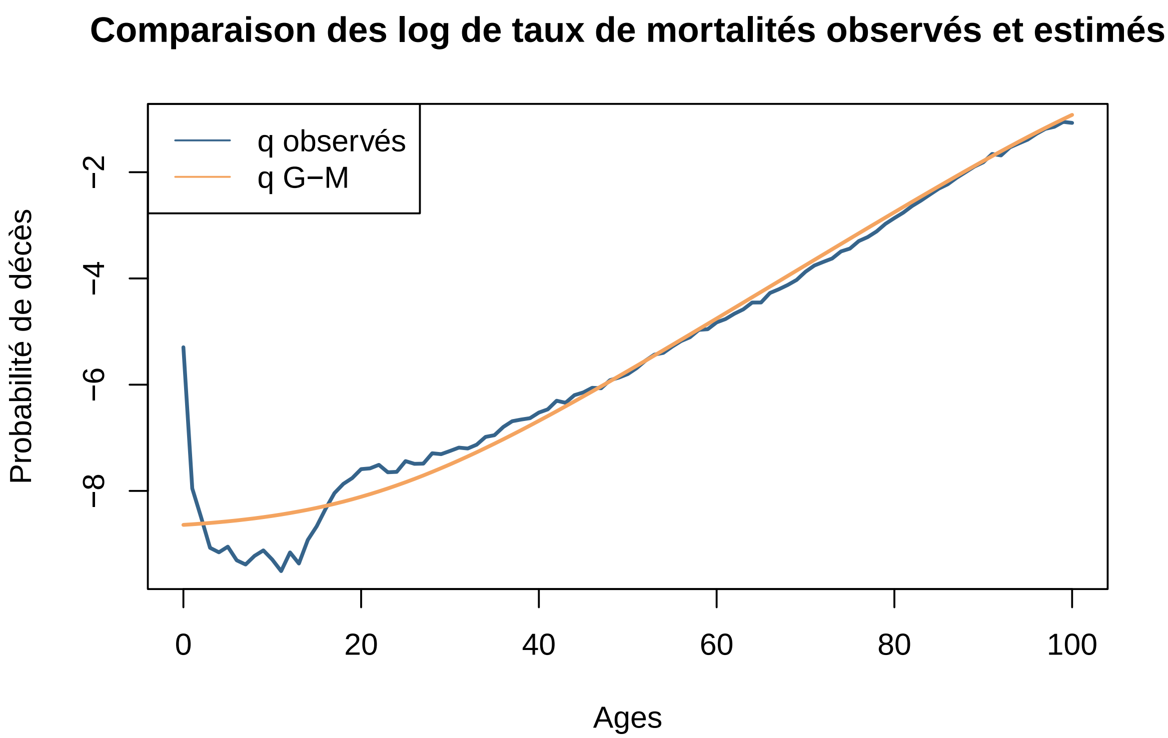
```r
c(alpha2, beta2, gamma2)
```

```
[1] 1.586628e-04 1.742762e-05 1.107691e+00
```

```r
# Construction du vecteur des probabilités de décès :
qM3 = 1 - exp(-C) * exp(-A / B * exp(B * ages) * (exp(B) - 1))

# Représentation graphique de l'âge des individus :
plot(
  ages,
  q,
  type = 'l',
  ylab = "Probabilité de décès",
  xlab = "Ages",
  main = "Comparaison des taux de mortalités observés et estimés",
  col = palette_couleur[1],
  lwd = 2
)
lines(ages, qM3, col = palette_couleur[2], lwd = 2)
legend(
  1,
  0.3,
  lty = 1,
  cex = 1,
  legend = c("q observés", "q estimés"),
  col = palette_couleur[1:2]
)
```



```r
# Comparaison des taux de mortalités logarithmiques :

plot(
  ages,
  log(q),
  type = 'l',
  ylab = "Probabilité de décès",
  xlab = "Ages",
  main = "Comparaison des log de taux de mortalités observés et estimés",
  col = palette_couleur[1],
  lwd = 2)
lines(ages, log(qM3), col = palette_couleur[2], lwd = 2)
legend("topleft",
  lty = 1,
  cex = 1,
  legend = c("q observés", "q G-M"),
  col = palette_couleur[1:2]
)
```



Interprétation des résultats :

Le modèle de Gompertz - Makeham, avec h croissant, ne peut pas modéliser correctement la mortalité aux âges inférieurs à 20 ans.

```r
library(MortalityLaws)

#availableLaws() # Liste des modèle de mortalité du package
```

```r
fit = MortalityLaw(x = 0:100, qx = q, law = "makeham")  #modèle h(x)= C + A exp(Bx)
fit$coefficients
```

3.1.2.2 Utilisation du package MortalityLaws :

```
           A            B            C 
0.0000251412 0.0953918954 0.0001246354 
```

```r
A = fit$coefficients["A"]
B = fit$coefficients["B"]
C = fit$coefficients["C"]
c(A, B, C)
```

```
           A            B            C 
0.0000251412 0.0953918954 0.0001246354 
```

```r
# Lien avec l'autre paramétrage (h(x)= alpha + beta gamma^x)
alpha2 = C
beta2 = A
gamma2 = exp(B)
c(alpha2, beta2, gamma2)
```

```
           C            A            B 
0.0001246354 0.0000251412 1.1000898909 
```

```r
# Estimation du taux de moralité de Lee-Carter
qM4 = 1 - exp(-C) * exp(-A / B * exp(B * ages) * (exp(B) - 1))


# Représentation graphique et comparaison :

plot(
  ages,
  log(q),
  type = 'l',
  ylab = "Probabilité de décès",
  xlab = "Ages",
  main = "Comparaison des log de taux de mortalités observés et estimés",
  col = palette_couleur[1],
  lwd = 2)
lines(ages, log(qM3), col = palette_couleur[2], lwd = 2)
lines(ages, log(qM4), col = palette_couleur[3], lwd = 2)
legend("topleft",
  lty = 1,
  cex = 1,
  legend = c("q observés", "q G-M", "q G-M bis"),
  col = palette_couleur[1:3]
)
```