Modèles de durée : TD et Examens

2024-11-27

Contents

1 Les méthodes semi-paramétriques

1.1 Le modèle de Cox

1.1.1 Lecture des données traitement de la base :

```r
library(tidyverse)
Re = read.table("DATA/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.

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

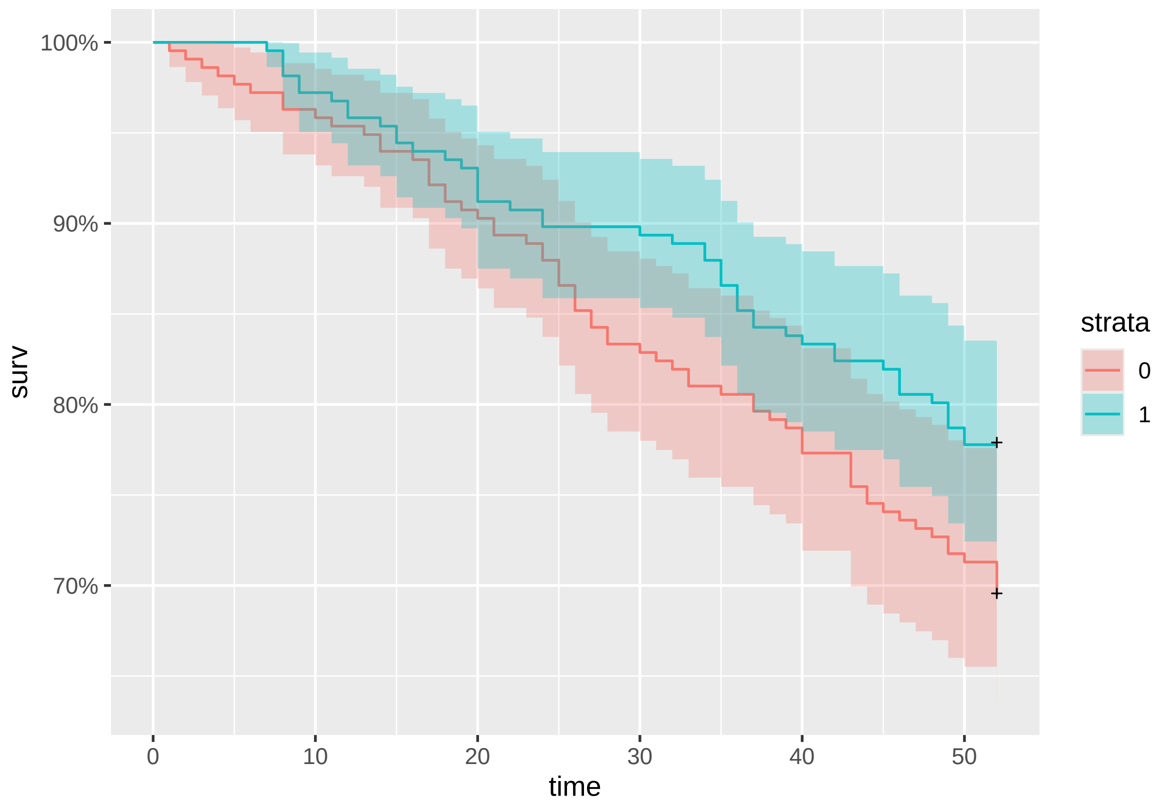
```r
library(survival)
# Test sur la variable financement :
survdiff(Surv(week, arrest) ~ fin, data = Re1)
```

```
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
```

```r
# 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 :

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

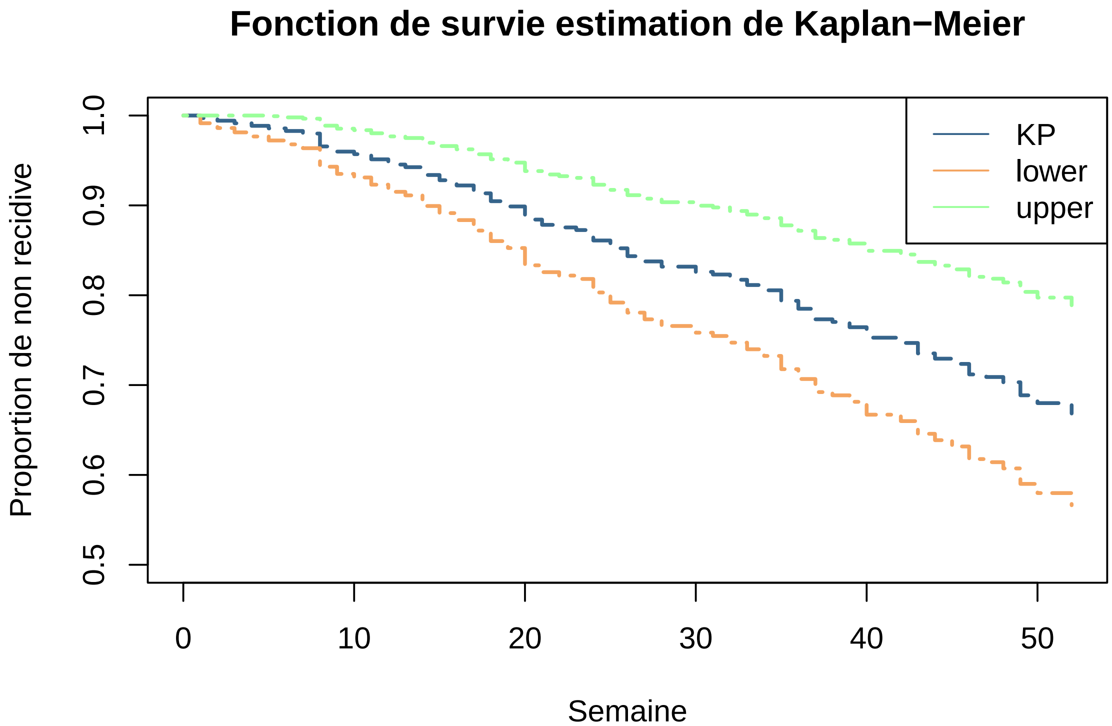
Dans le cadre des fonction de Kaplan Meier, Aalen par défaut les covariables sont fixées à la valeur moyenne.

```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])
```

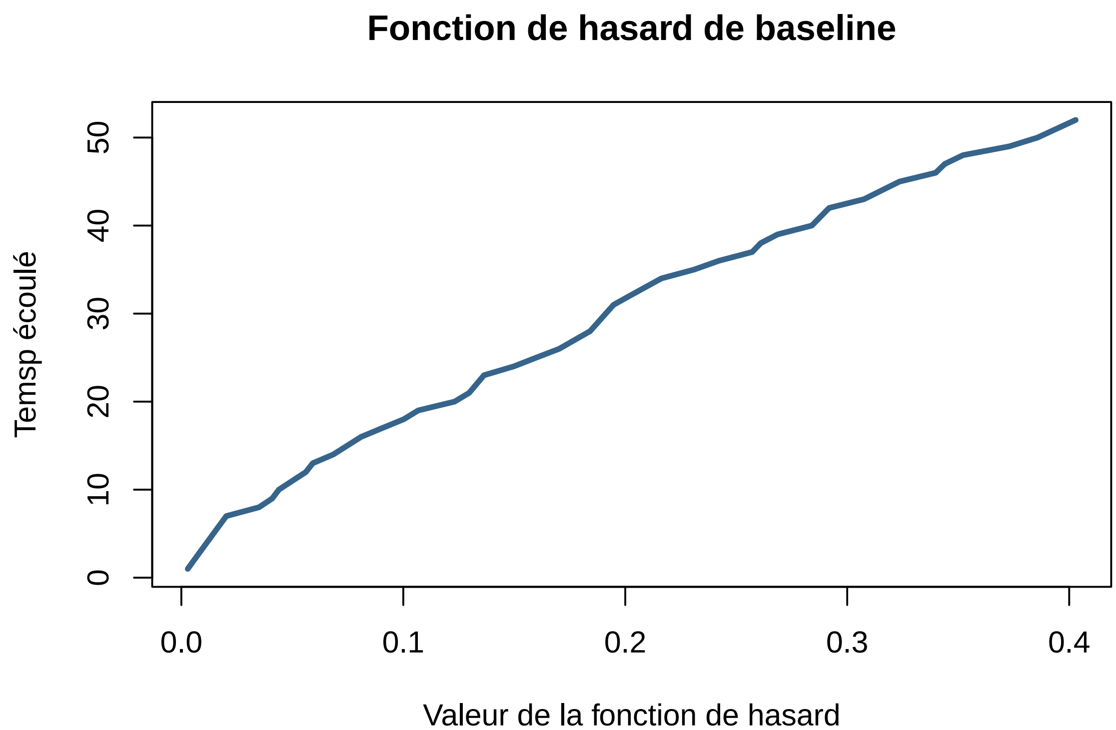


1.1.6 Fonction de hasard cumulée avec l'estimateur de Breslow :

Interprétation Intuitive:

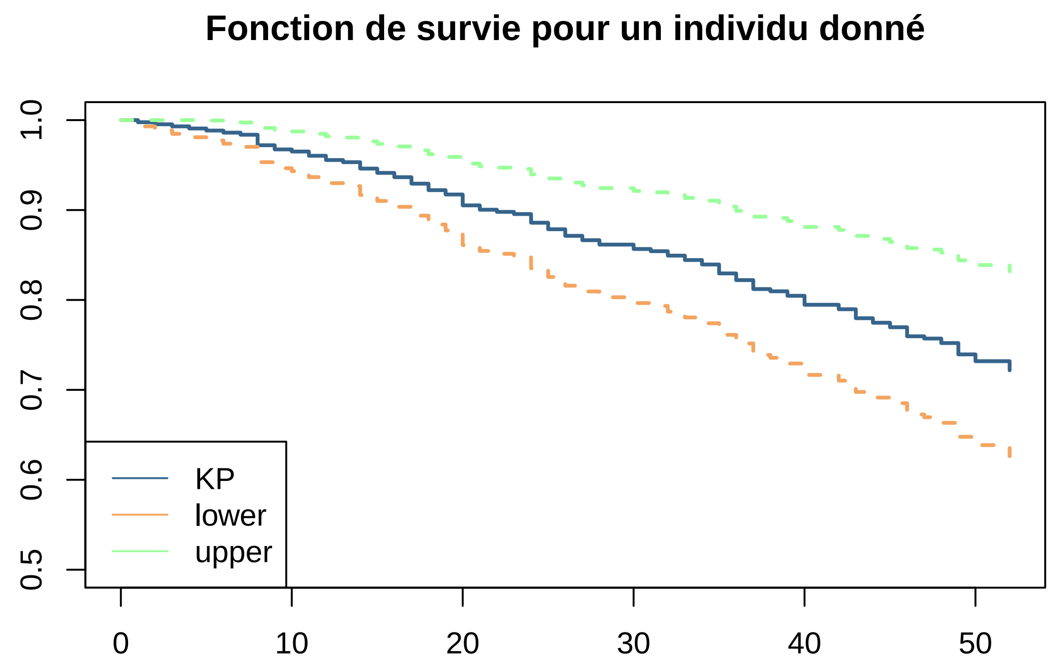
Taux Instantané: La fonction de hasard représente le taux instantané de survenue de l'événement à un moment donné.
Par exemple, si h(t)=0.05 à t=10 semaines, cela signifie que le taux de survenance de évènement à 10 semaines est de 5% par unité de temps.
Conditionnelle à la Survie: La fonction de hasard 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])
```

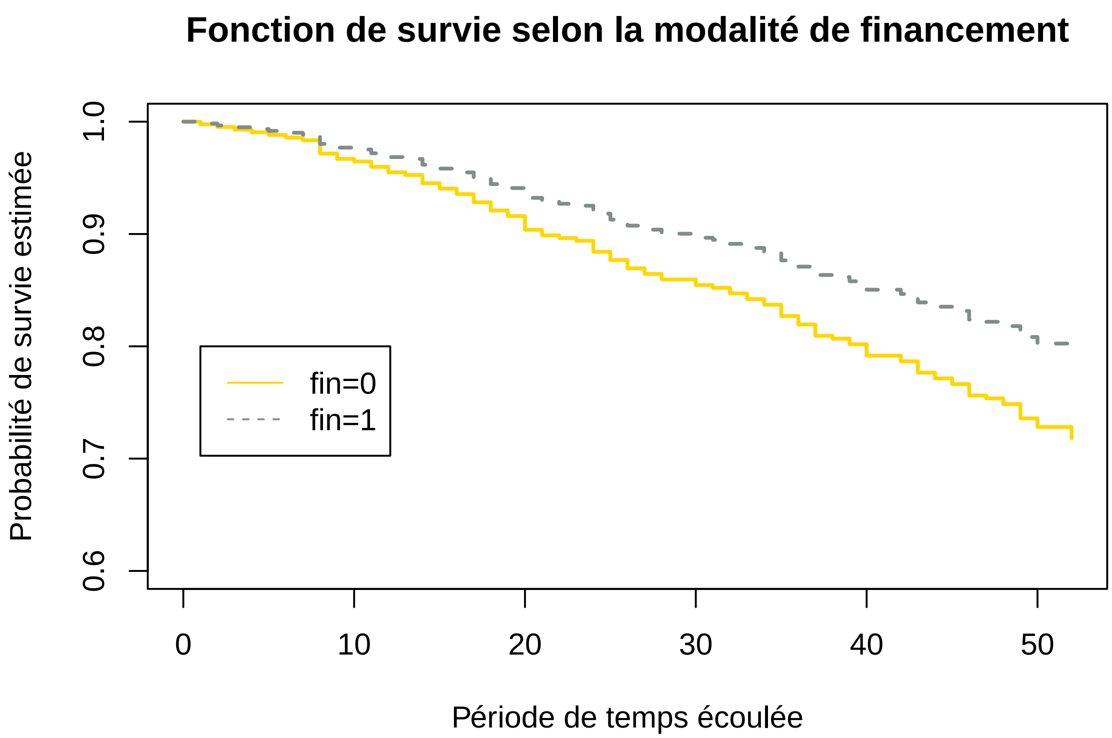


1.1.8 Etude de l'effet d'une covariable (les autres étant fixées) :

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)

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

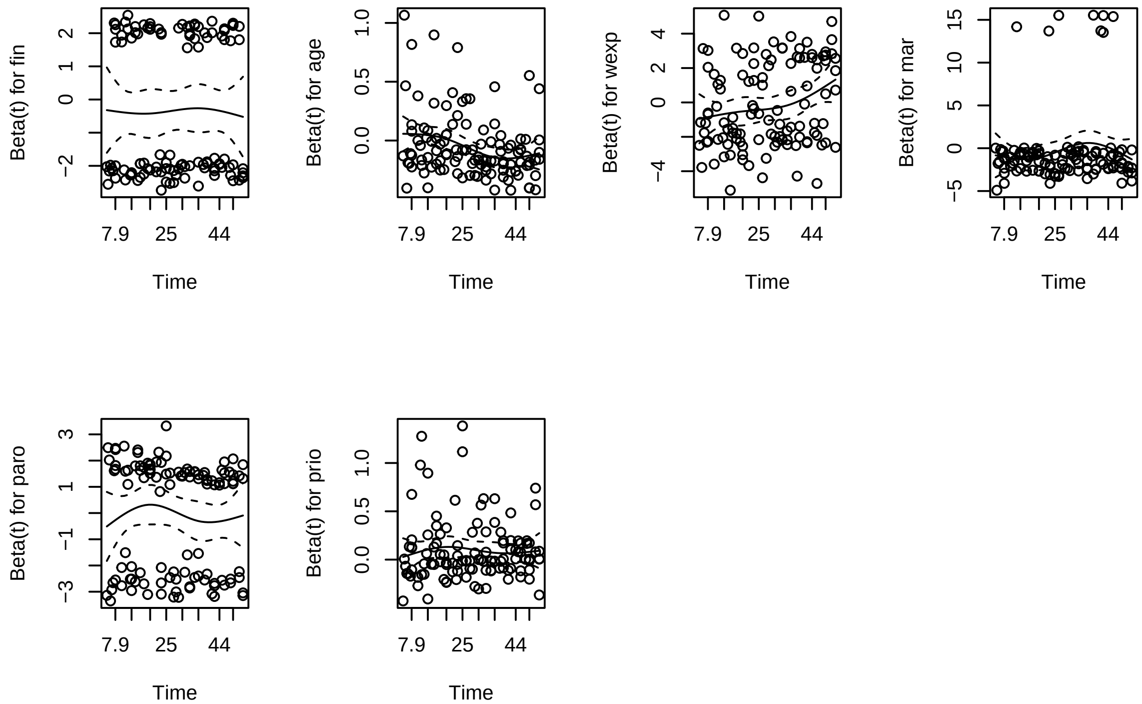
Explication : Si H0 est rejetée, alors les résidus dépendent du temps

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.

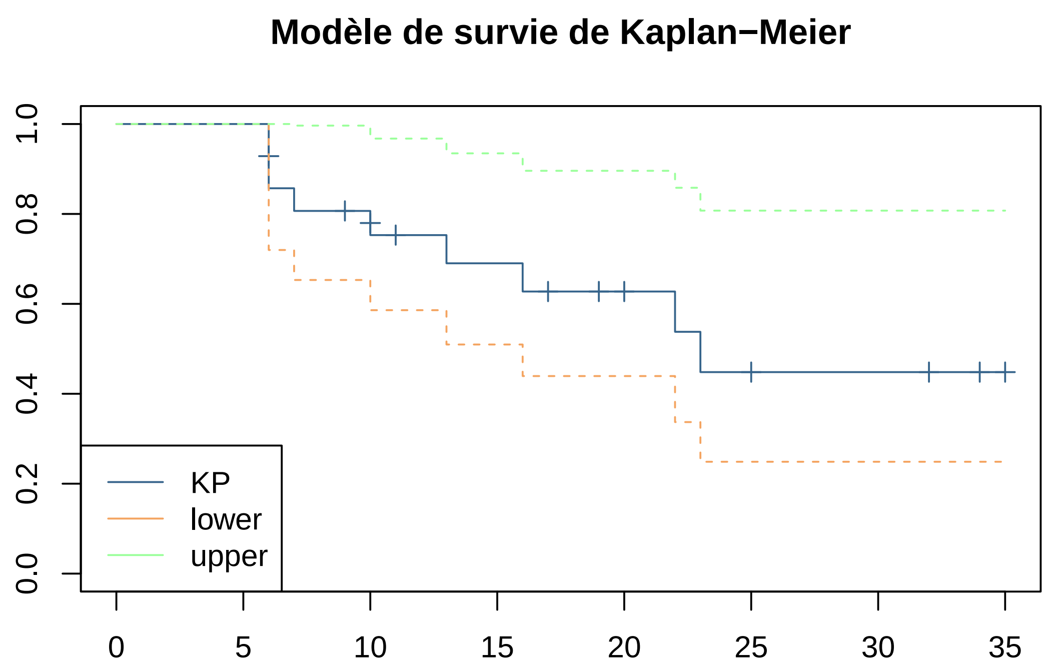
```r
library(survival)
tempsGMP = c(rep(6, 4), 7, 9, 10, 10, 11, 13, 16, 17, 19, 20, 22, 23, 25, 32,
             32, 34, 35) # liste des observations
finGMP = c(1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, rep(0, 5))
# indication des obs censurées
donnF = Surv(tempsGMP, finGMP)
head(donnF)
```

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

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

```r
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)
```

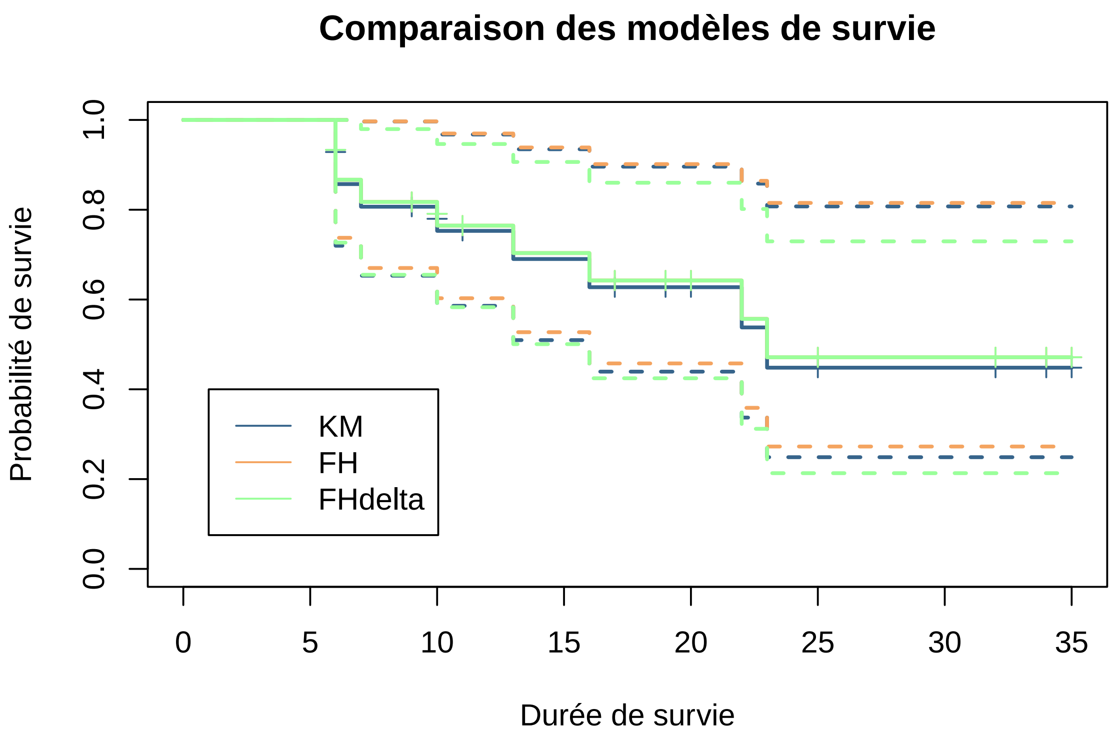
2.2.3 Comparaison des résultats sur l'estimation du 10e individu de la base :

```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
```

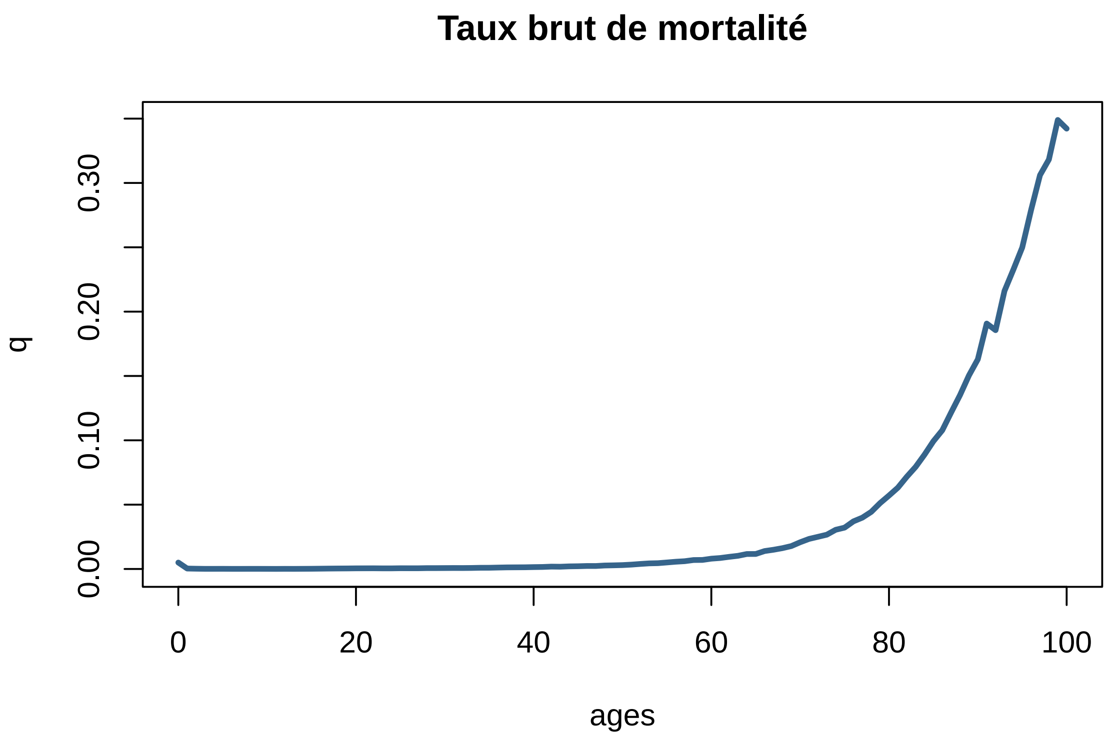
```
(Intercept)
 0.02506964
```

3 Examen 2018 :

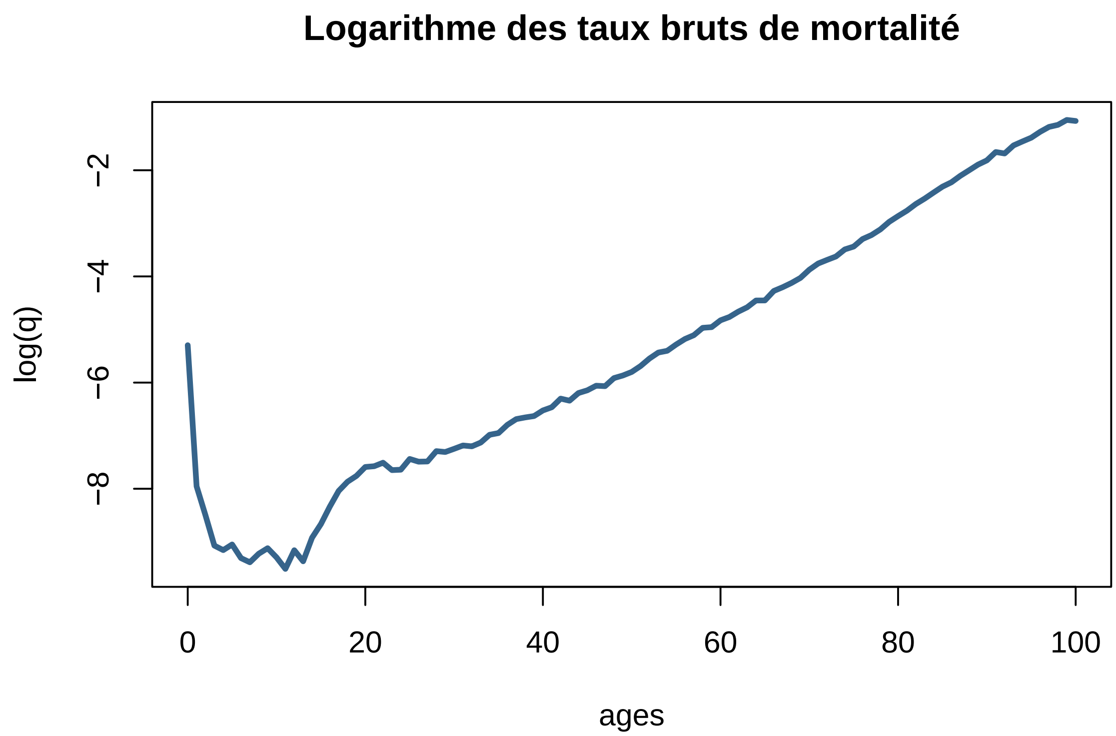
3.1 Exercice 2 :

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
```

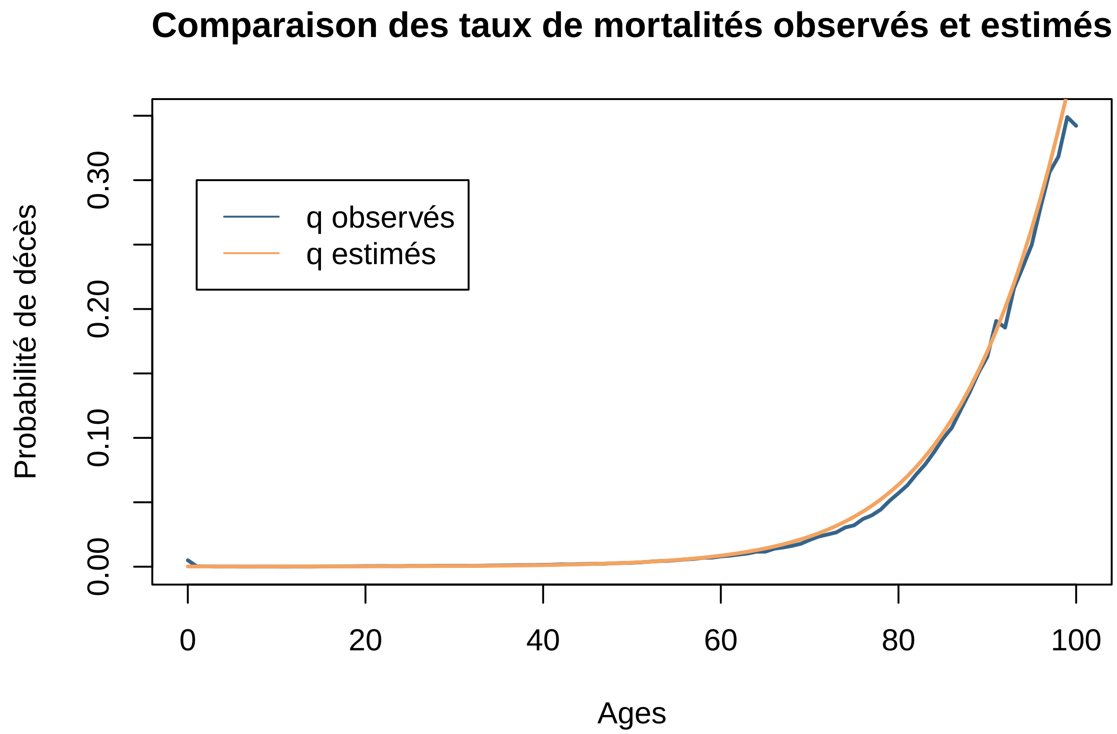
```
# Lien avec l'autre paramétrage :
alpha2 = C
beta2 = A
gamma2 = exp(B)
cat("Apha, Beta, Gamma : \n")
```

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

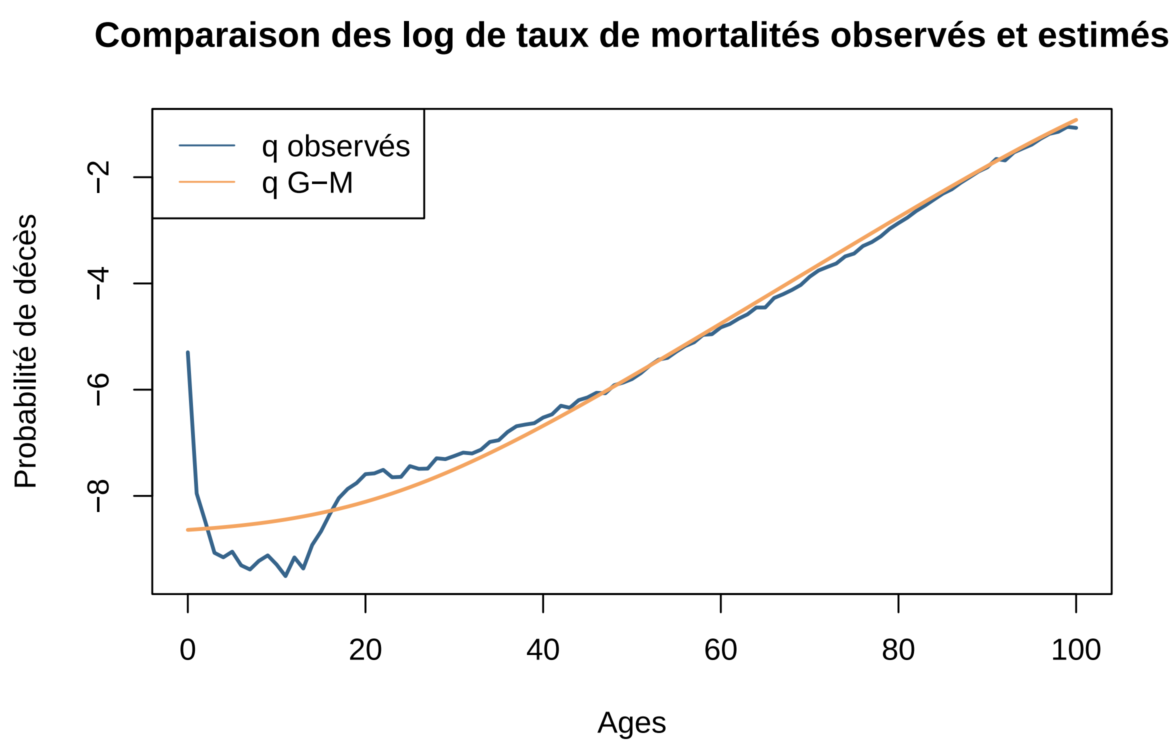
```r
c(alpha2, beta2, gamma2)
```

```
[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 
```

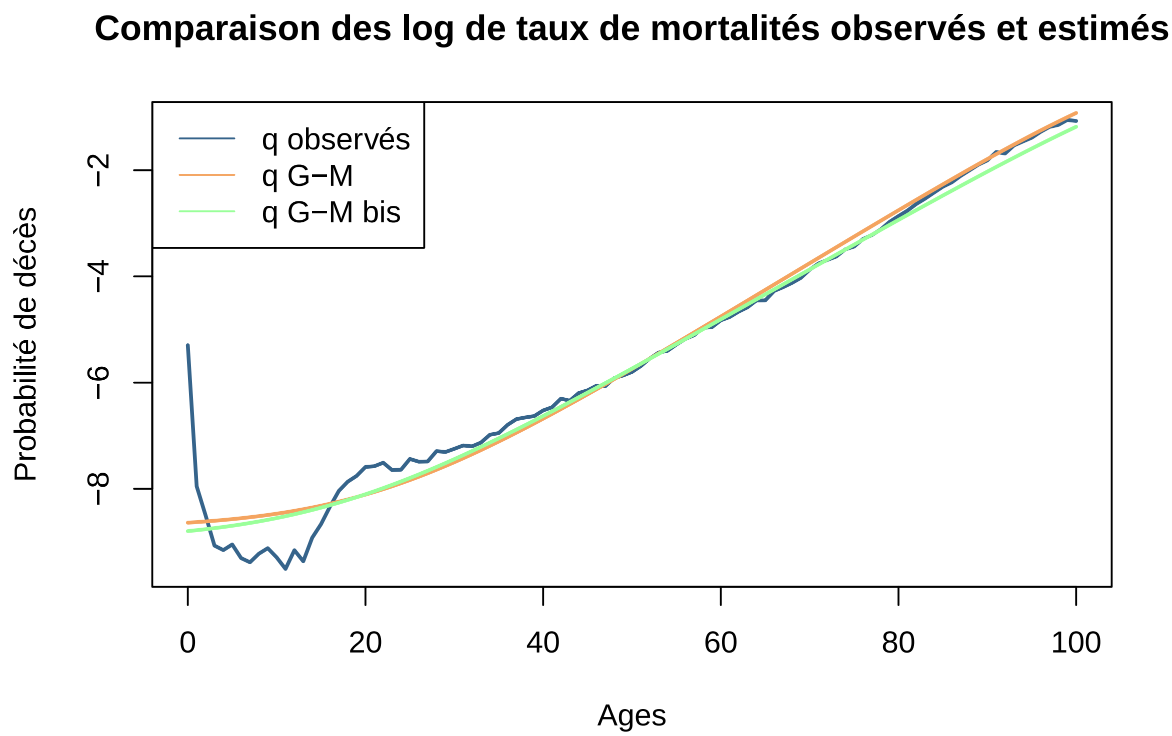
```r
A = fit$coefficients["A"]
B = fit$coefficients["B"]
C = fit$coefficients["C"]
c(A, B, C)
```

```
           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]
)
```



3.1.3 Modélisation de Lee Carter :

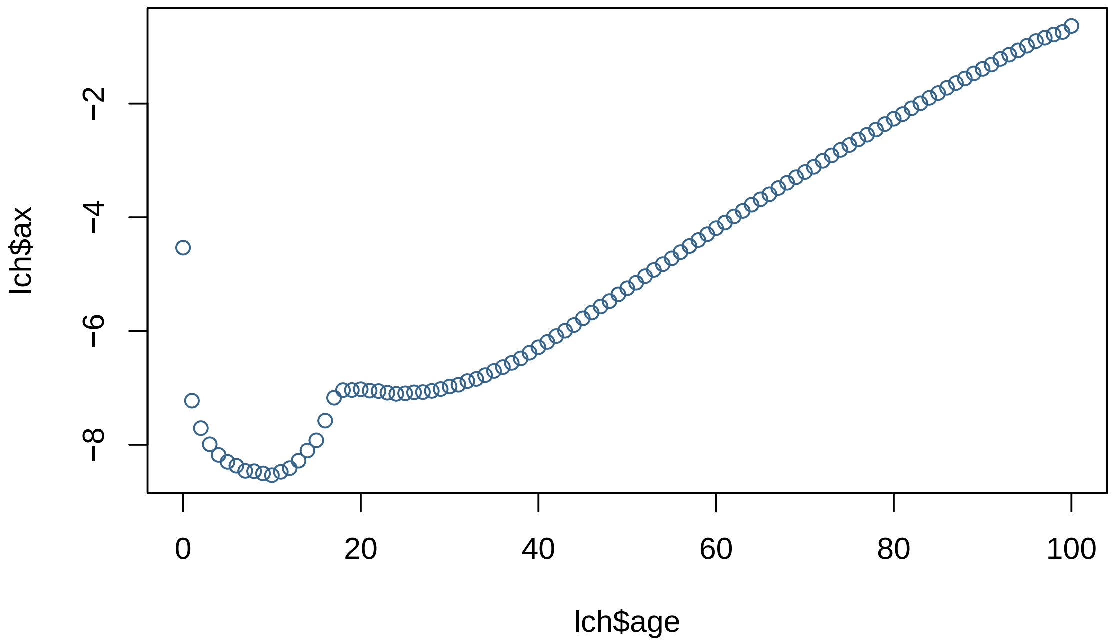
Rappels sur la modélisation de Lee Carter :

$ln(\mu(x,t)) = \alpha_x + \beta_x \times k_t + \epsilon_{(x,t)}$

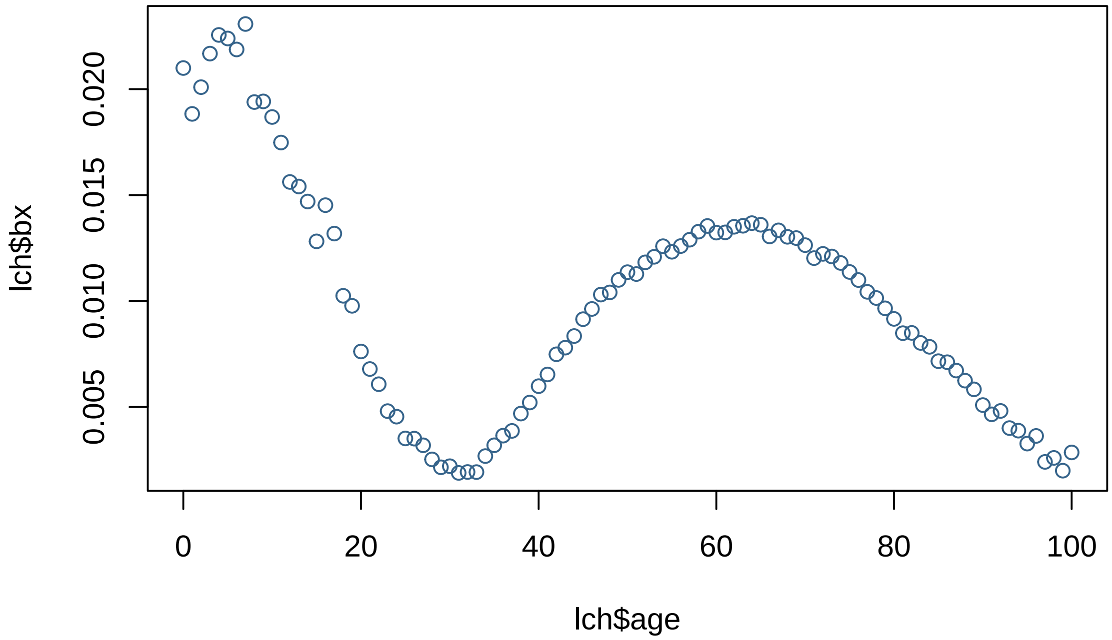
Avec : _x = la valeur moyenne

$\begin{cases} \alpha_x \text{ : la valeur moyenne} \\ k_t \text{ : correspond à une évolution générale dans le temps} \beta_x \text{ : la sensibilité du taux instantané par rapport à une variation d} \end{cases}$

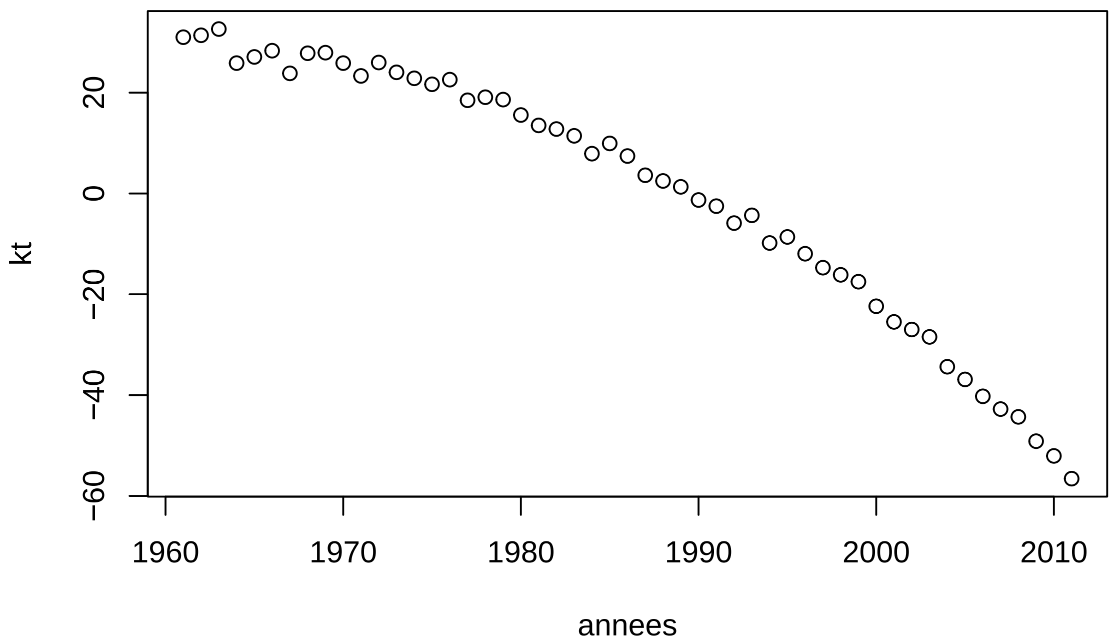
```
library(forecast)
library(demography)
muh = De / Ex
Baseh = demogdata(
  data = muh,
  pop = Ex,
  ages = ages,
  years = annees,
  type = "mortality",
  label = 'G.B.',
  name = 'Hommes',
  lambda = 1) #

lch = lca(Baseh) # Lancement du modèle de Lee-Carter

# Estimation de alpha_x
plot(lch$age, lch$ax, col = palette_couleur[1])
```




```
# Estimation de beta_x
plot(lch$age, lch$bx, col = palette_couleur[1])
```



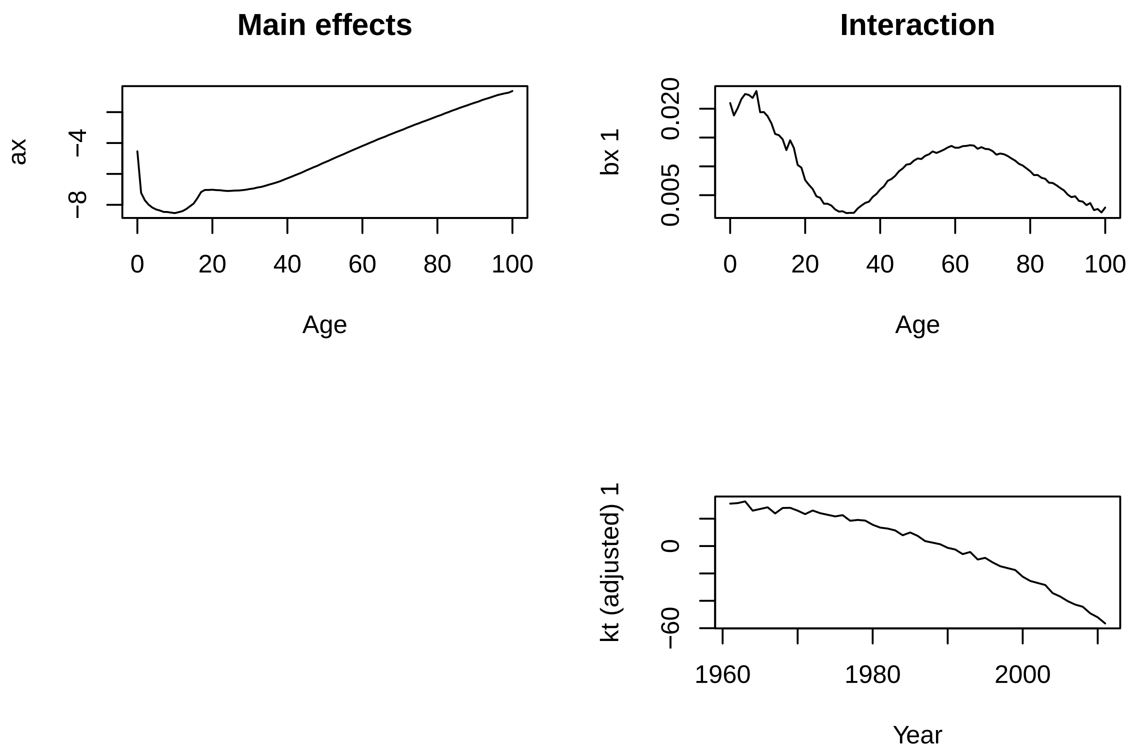

```r
# Estimation des k_t
kt = lch$kt
plot(annees, kt)
```



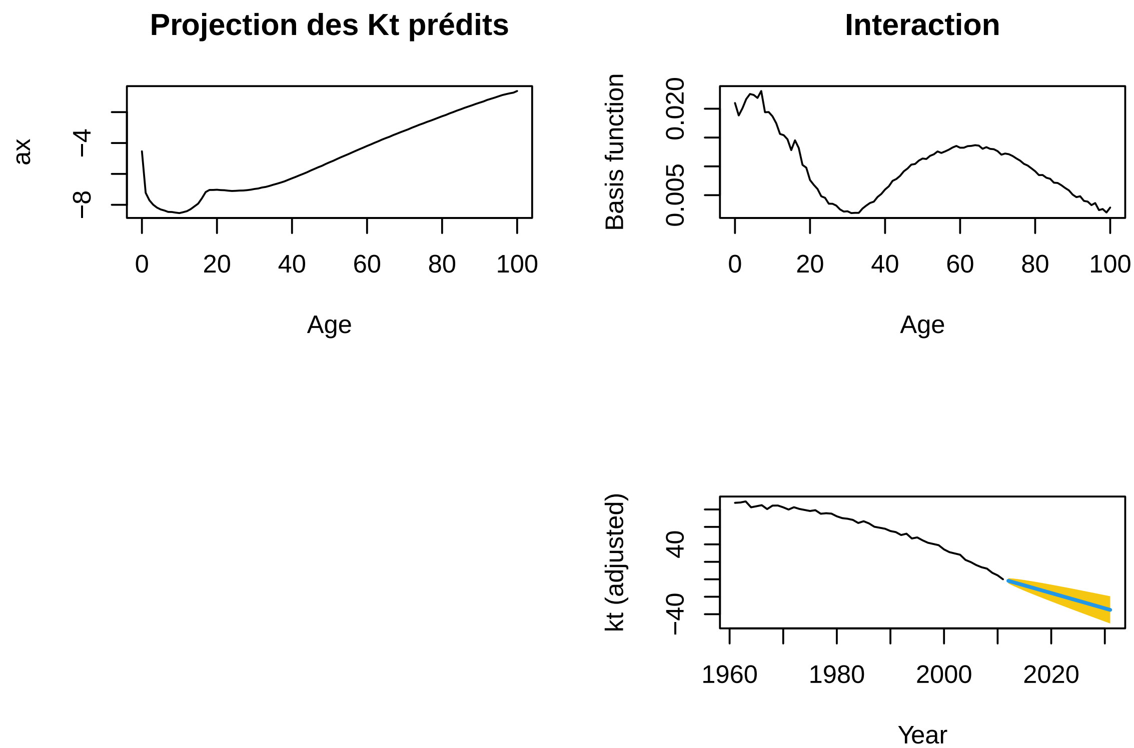
3.1.3.1 Méthode de Lee-Carter 1992 : Projection des Kt Rappel: les Kt représentent

Hypothèse : $k_t = k_{t-1} + d + e_t$

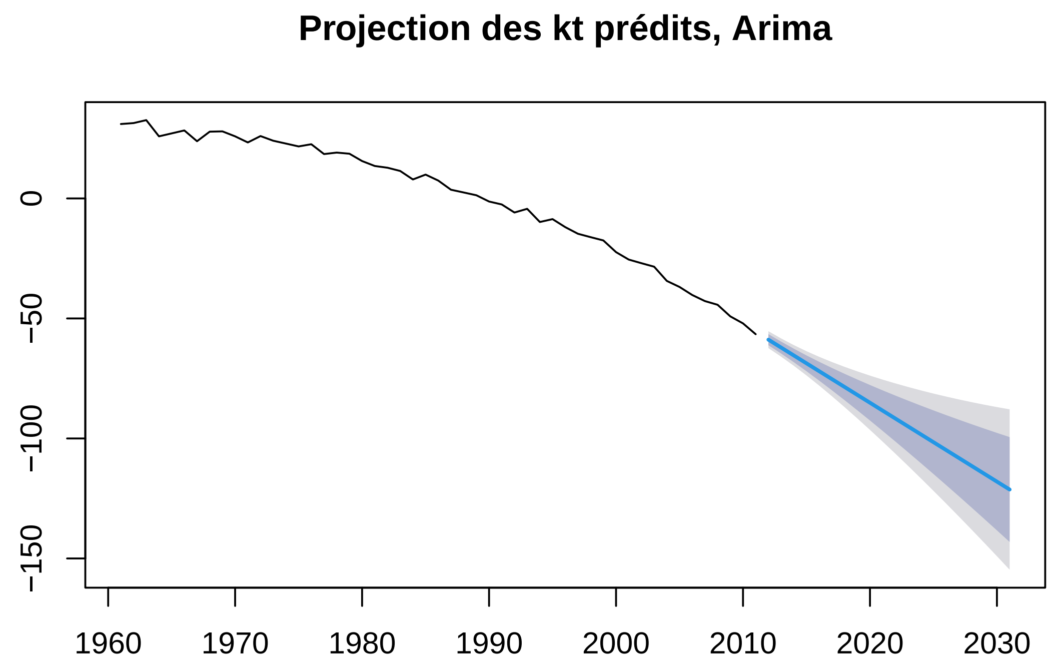
```r
# Projection des Kt à l'aide du modèle initial :
plot(lch)
```




```r
proj = forecast(lch, h = 20)
plot(proj, plot.type = "component", main = "Projection des Kt prédits")
```




```r
# Projection des Kt à l'aide du modèle ARIMA :
ar = auto.arima(kt)
plot(forecast(ar, h = 20), main = "Projection des kt prédits, Arima")
```



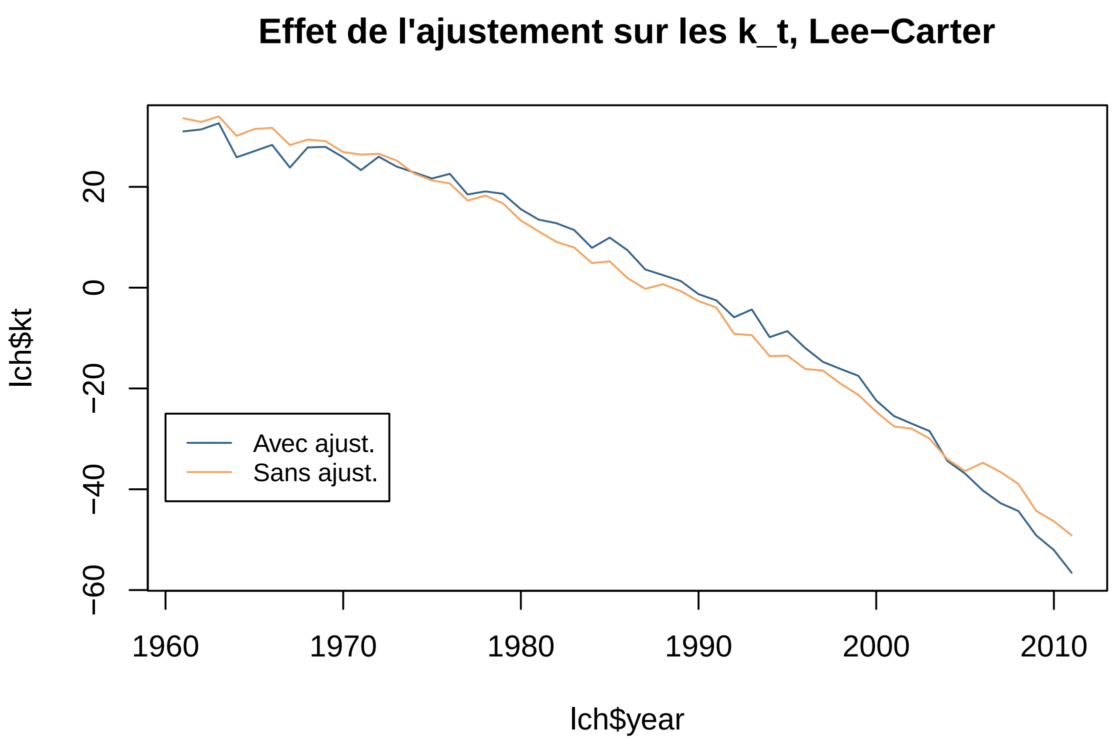
Interprétation (BA) :

- a_x donne une indication sur la valeur de la mortalité moyenne
- b_x la variation du taux instantané comporte trois phases. Le taux est de moins en moins déterminant sur les années de 0 à 20 ans ainsi que sur l'intervalle 60 à 100 ans. En revanche ce taux est croissant entre 20 à 60 ans. Ce qui correspond souvent à la période durant laquelle l'Homme est le plus actif. Le risque additionnel de décès à tendance à croître sur cette période. Enfin la période de 0 à 10 est celle qui admet un coefficient de taux instantané le plus fort du fait notamment de la mortalité infantile.
- k_t est décroissant sur toute la période, ce qui permet de conclure que la mortalité tend à décroître sur la période observée et ainsi maintient le constat d'une diminution des causes de mortalité annexes.

3.1.3.2 Modèle de Lee Carter sans ajustement des Kt : Dans cette partie on fait l'hypothèse que les k_t sont constants dans le temps.

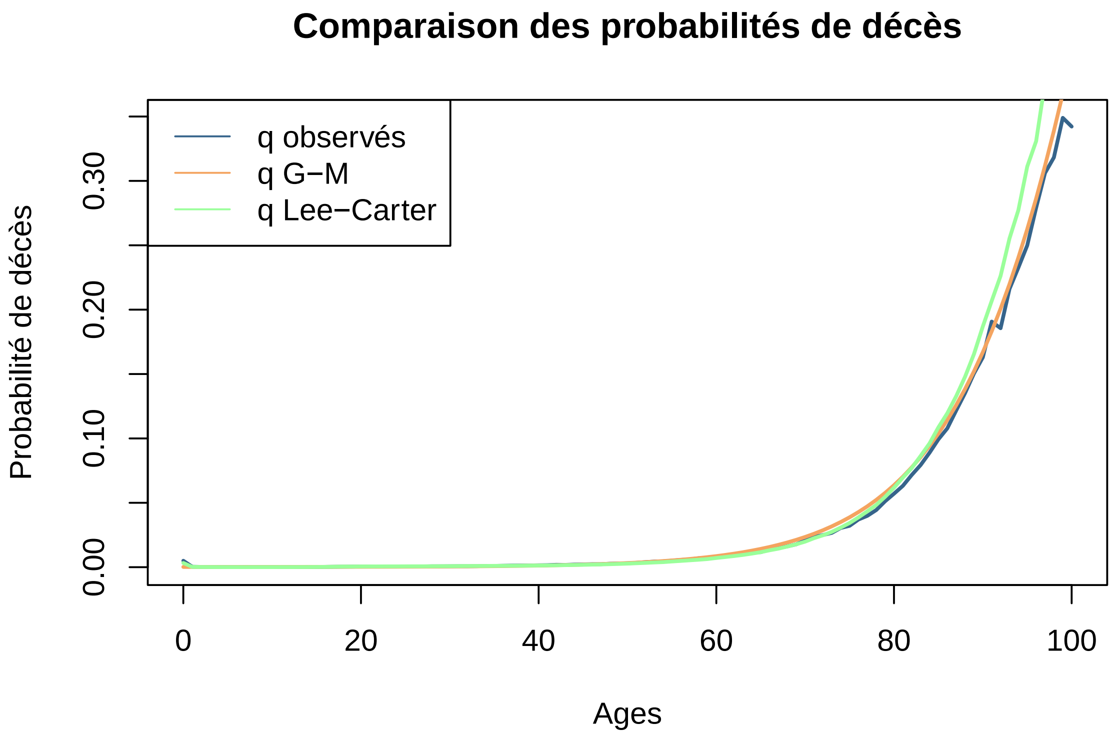
```
## L.C. sans ajustement des k_t
lch_sans = lca(Baseh, adjust = "none")
plot(lch$year,
     lch$kt,
     col = palette_couleur[1],
     type = 'l',
     main = "Effet de l'ajustement sur les k_t, Lee-Carter")
lines(lch_sans$year, lch_sans$kt, col = palette_couleur[2])
legend(
  1960,
  -25,
  legend = c("Avec ajust.", "Sans ajust."),
  col = palette_couleur[1:2],
```

```
  lty = 1,
  cex = 0.8
)
```

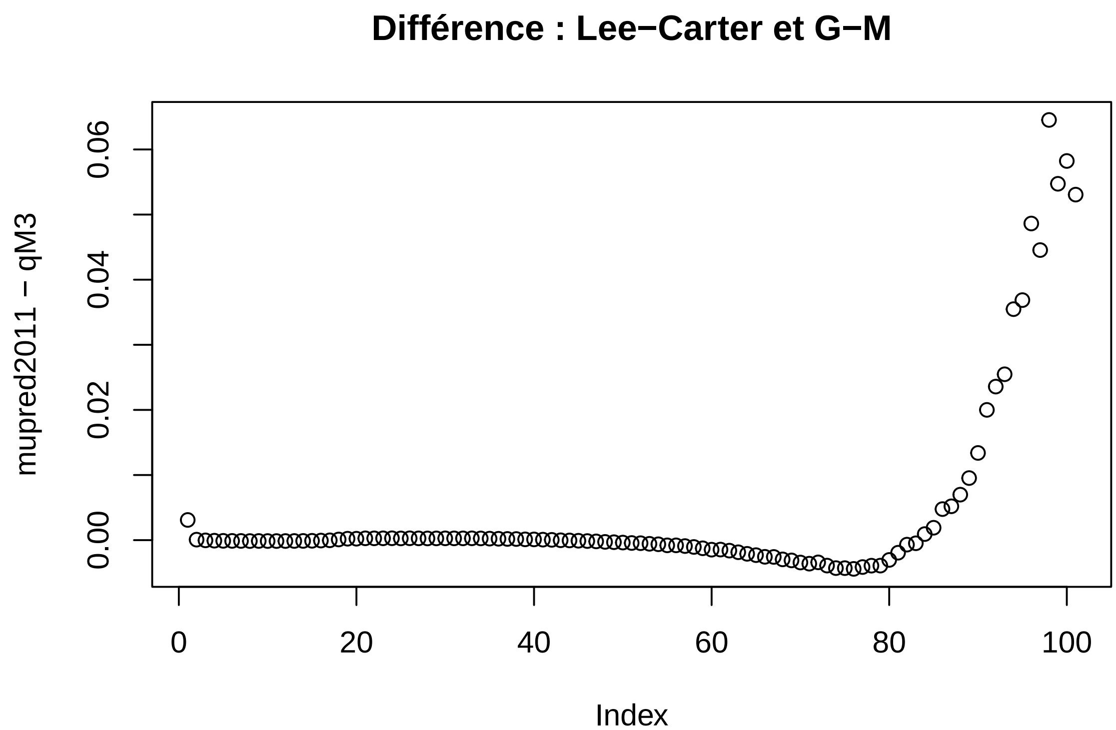



```
# Modéle de Lee Carter :
predh = lch$fitted$y  # c'est log(mu_{x,t}) qui est prédit
mupred2011 = exp(predh[, 51])

plot(
  ages,
  q,
  type = 'l',
  ylab = "Probabilité de décès",
  xlab = "Ages",
  main = "Comparaison des probabilités de décès",
  col = palette_couleur[1],
  lwd = 2
)
lines(ages, qM3, col = palette_couleur[2], lwd = 2)
lines(ages,mupred2011,col = palette_couleur[3], lwd = 2)
legend("topleft",
  lty = 1,
  cex = 1,
  legend = c("q observés", "q G-M", "q Lee-Carter"),
  col = palette_couleur[1:3]
)
```

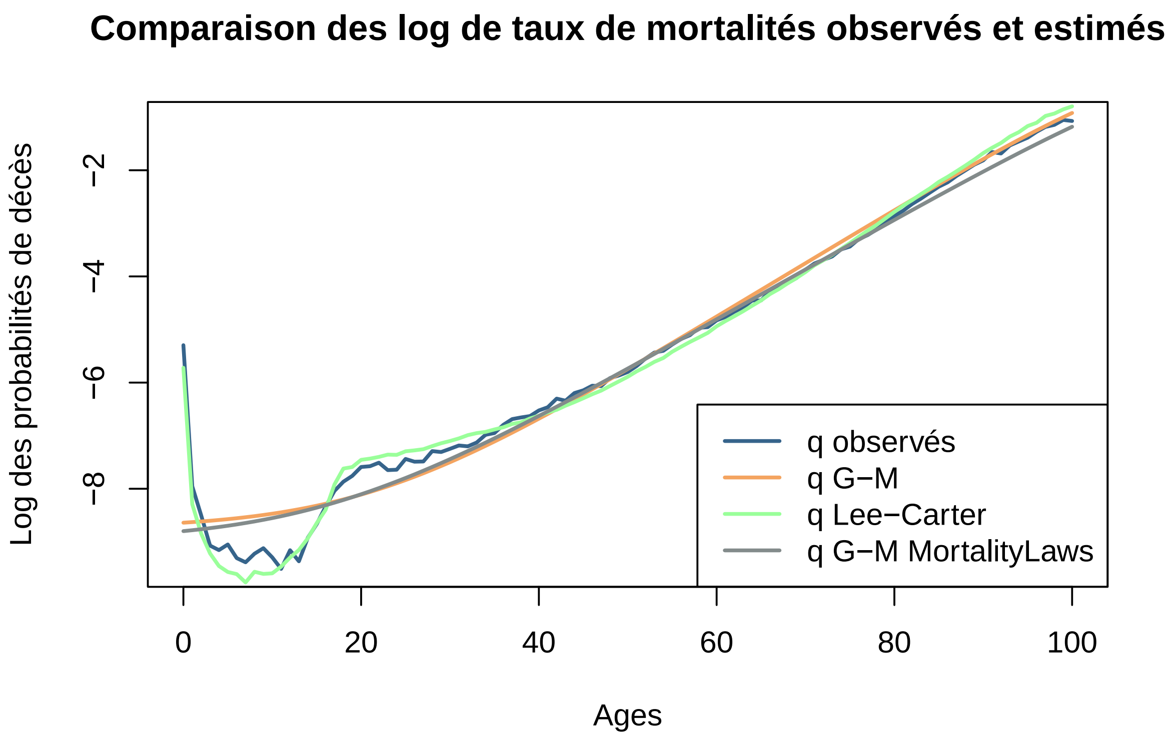
3.1.3.3 Comparaison des modèles :




```r
# Représentation graphique de la différence entre les modèles :
plot(mupred2011 - qM3,
     main = "Différence : Lee-Carter et G-M")
```




```r
#max(abs(mupred2011 - qM3))

# Comparaison graphique log(q) :
plot(
  ages,
  log(q),
  type = 'l',
  ylab = "Log des probabilités 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,predh[,51],col = palette_couleur[3], lwd = 2)
lines(ages, log(qM4),col = palette_couleur[5], lwd = 2)
legend("bottomright",
  lty = 1,
  cex = 1,
  lwd = 2,
  legend = c("q observés", "q G-M", "q Lee-Carter", "q G-M MortalityLaws"),
  col = palette_couleur[c(1:3,5)]
)
```



3.1.4 Calcul des rentes :

Nous souhaitons calculer la prime pure d'une rente viagère à partir de 2012 pour l'âge de 65 ans.

$a_x(t) = \sum_{k\geq 0} \prod_{j=0}^{k} exp(-\mu_{x+j}(t+j)) * 1/(1+r)^(k+1)$

```
# Projections des \mu{x,t} dans le futur avec le modèle de Lee-Carter
projh = forecast(lch, h = 70)$rate$Hommes

#dim(projh) # L'objet projh est de dimension 101 x 70
colnames(projh) = 2012:(2012 + 69)
rownames(projh) = 0:100

r = 0.035 # valeur du taux choisi pour le facteur d'actualisation

# calcul de a_65(2012) pour les hommes :

L = length(66:101)
mu = projh[66:101, 1:L] # on limite aux âges 65-100
dmu = diag(mu)
prodexpmu = cumprod(exp(-dmu))
a = 0
for (k in 1:length(dmu))
{
  a = a + 1 / (1 + r) ^ (k) * prodexpmu[k]
}
cat("En s'arretant à 110 ans : ")
```

3.1.4.1 Calcul à l'aide du modèle de Lee-Carter :

```
En s'arretant à 110 ans :
```

```r
a  # 13.164
```

```
[1] 13.16419
```

```r
# Remarque : si on prolonge jusqu'à 120 ans avec les mêmes \mu(x,t) ?
# (pour vérifier si négliger les âges > 110 est justifié)
dmu120 = c(dmu, rep(dmu[L], 20))
prodexpmu120 = cumprod(exp(-dmu120))
a120 = 0
for (k in 1:(L + 20))
{
  a120 = a120 + 1 / (1 + r) ^ (k) * prodexpmu120[k]
}
cat("Avec la table jusque 120 ans : ")
```

```
Avec la table jusque 120 ans :
```

```r
a120 # 13.174
```

```
[1] 13.17422
```

```r

# Comparaison avec G.M. I  (fmsb)
dmu = qM3[66:101]
prodexpmu = cumprod(exp(-dmu))
a = 0
for (k in 1:length(dmu))
{
  a = a + 1 / (1 + r) ^ (k) * prodexpmu[k]
}
cat("GM fmsb :")
```

3.1.4.2 Calcul à l'aide du modèle de Gompertz-Makeham :

```
GM fmsb :
```

```r
a
```

```
[1] 12.1337
```

```r
# 12.13

# Comparaison avec G.M. II  (Mortalitylaw)
dmu = qM4[66:101]
prodexpmu = cumprod(exp(-dmu))
a = 0
for (k in 1:length(dmu))
{
  a = a + 1 / (1 + r) ^ (k) * prodexpmu[k]
}
cat("GM LawMortality : ")
```

```
GM LawMortality :
```

```r
a
```

```
[1] 12.77115
```

```r
# 12.77
```

4 Examen 2019 :

4.1 Exercice 1 :

4.1.1 Importation des données :

```r
Re = read.table(file = 'DATA/emploi.txt', header = TRUE)
str(Re)
```

```
'data.frame':   600 obs. of  15 variables:
 $ id     : int  1 2 2 2 3 3 3 3 3 4 ...
 $ noj    : int  1 1 2 3 1 2 3 4 5 1 ...
 $ tstart : int  555 593 639 673 688 700 730 742 817 872 ...
 $ tfin   : int  982 638 672 892 699 729 741 816 828 926 ...
 $ sex    : int  1 2 2 2 2 2 2 2 2 2 ...
 $ ti     : int  982 982 982 982 982 982 982 982 982 982 ...
 $ tb     : int  351 357 357 357 473 473 473 473 473 604 ...
 $ te     : int  555 593 593 593 688 688 688 688 688 872 ...
 $ pres   : int  34 22 46 46 41 41 44 44 44 55 ...
 $ edu    : int  17 10 10 10 11 11 11 11 11 13 ...
 $ tfp    : int  428 46 34 220 12 30 12 75 12 55 ...
 $ des    : int  0 1 1 1 1 1 1 1 1 1 ...
 $ cohorte: int  1 1 1 1 2 2 2 2 2 3 ...
 $ lfx    : int  0 0 46 80 0 12 42 54 129 0 ...
 $ pnoj   : int  0 0 1 2 0 1 2 3 4 0 ...
```

```r
#Re[Re$sex==2,5]=0
library(survival)
```

4.1.2 Estimateur de Kaplan-Meier : test de comparaison

Rappel sur les tests :

- Le test du log-rank et test de Gehan :

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

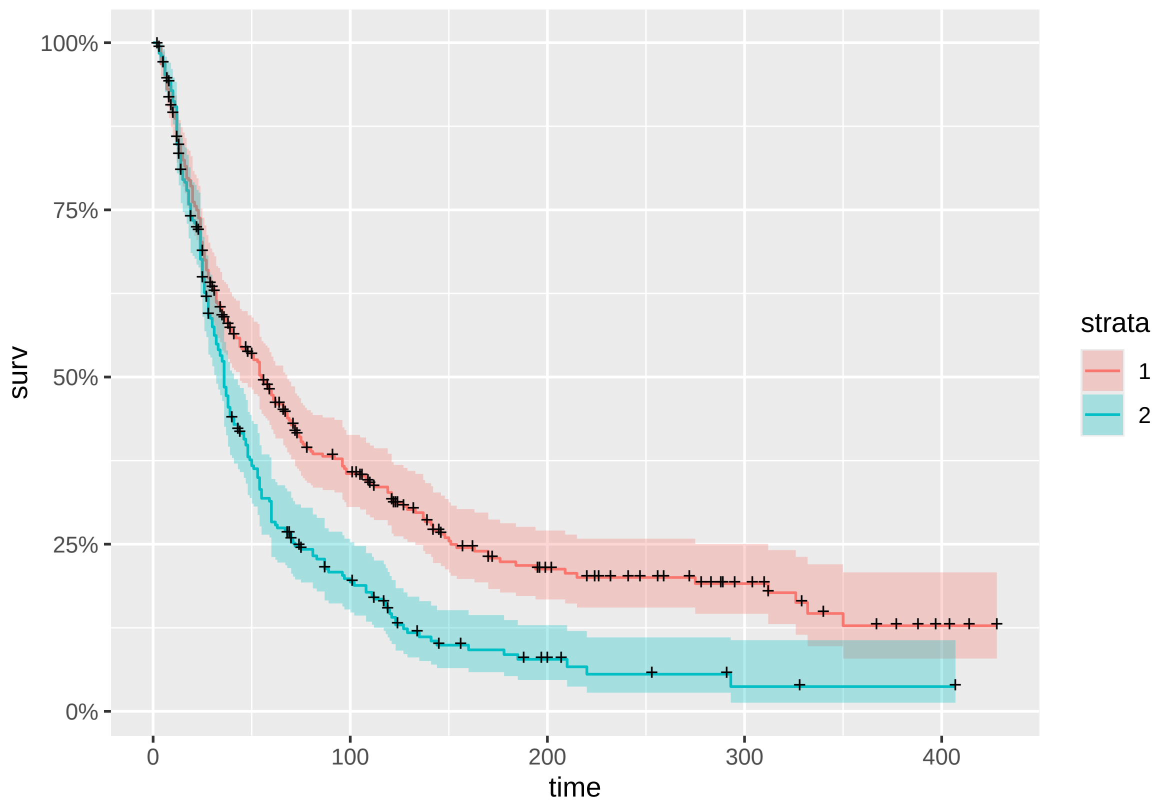
```r
# Test de comparaison des durées de survie selon le sexe
survdiff(Surv(tfp, des) ~ sex, data = Re, rho = 0) # log-rank
```

```
Call:
survdiff(formula = Surv(tfp, des) ~ sex, data = Re, rho = 0)

        N Observed Expected (O-E)^2/E (O-E)^2/V
sex=1 348      245      291      7.24      20.6
sex=2 252      213      167     12.60      20.6

 Chisq= 20.6  on 1 degrees of freedom, p= 6e-06
```

```r
survdiff(Surv(tfp, des) ~ sex, data = Re, rho = 1) # Gehan
```

```
Call:
survdiff(formula = Surv(tfp, des) ~ sex, data = Re, rho = 1)

        N Observed Expected (O-E)^2/E (O-E)^2/V
sex=1 348      146      168      2.92      10.8
sex=2 252      130      108      4.56      10.8

 Chisq= 10.8  on 1 degrees of freedom, p= 0.001
```

```r
library(ggfortify)
s = survfit(Surv(tfp, des) ~ sex, data = Re, type = "kaplan-meier")
autoplot(s)
```




```r
s
```

```
Call: survfit(formula = Surv(tfp, des) ~ sex, data = Re, type = "kaplan-meier")

        n events median 0.95LCL 0.95UCL
sex=1 348    245     55      44      68
sex=2 252    213     36      32      41
```

4.1.3 Estimation par un modèle de Cox :

4.1.3.1 Modélisation :

Remarque :

ties=c("efron","breslow","exact") permet de choisir la méthode à adopter en cas d'événements simultanés par défaut, c'est ici l'approximation d'Efron qui est utilisée.

Ecriture du modèle de Cox :

$$h(t) = h_0(t)\exp(\beta_1 \text{pnoj} + \beta_2 \text{edu} + \beta_3 \text{sex} + \beta_4 \text{pres} + \beta_5 \text{lfx})$$

où : - $h(t)$ est la fonction de hasard à l'instant t. - $h_0(t)$ est la fonction de hasard de base à l'instant t. - $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ sont les coefficients des covariables. - $\text{pnoj}, \text{edu}, \text{sex}, \text{pres}, \text{lfx}$ sont les covariables incluses dans le modèle.

Analyse des résultats :

- On teste si les coefficients sont significativement différents de 0 au seuil de 0.05%

se(coef) <=> sqrt(var(beta)) Test H0 : beta_j=0 ==> Pr(>|z|) : prob(|U|> z), où U N(0,1)

```r
cox1 = coxph(formula = Surv(tfp, des) ~ pnoj + edu + sex + pres + lfx,
             data = Re)
summary(cox1)
```

```
Call:
coxph(formula = Surv(tfp, des) ~ pnoj + edu + sex + pres + lfx,
    data = Re)

  n= 600, number of events= 458

          coef exp(coef)  se(coef)      z Pr(>|z|)
pnoj  0.106887  1.112809  0.043897  2.435  0.01489 *
edu   0.066008  1.068235  0.023896  2.762  0.00574 **
sex   0.391422  1.479083  0.097445  4.017 5.90e-05 ***
pres -0.022698  0.977557  0.005315 -4.271 1.95e-05 ***
lfx  -0.004618  0.995392  0.000896 -5.154 2.55e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     exp(coef) exp(-coef) lower .95 upper .95
pnoj    1.1128     0.8986    1.0211    1.2128
edu     1.0682     0.9361    1.0194    1.1195
sex     1.4791     0.6761    1.2219    1.7904
pres    0.9776     1.0230    0.9674    0.9878
lfx     0.9954     1.0046    0.9936    0.9971

Concordance= 0.621  (se = 0.014 )
Likelihood ratio test= 78.74  on 5 df,   p=2e-15
Wald test            = 71.22  on 5 df,   p=6e-14
Score (logrank) test = 72.22  on 5 df,   p=4e-14
```

```r
# (Kaplan Meier ou Aalen, Aalen par défaut)
# les covariables sont fixées à la valeur moyenne
summary(survfit(cox1))
```

4.1.3.2 Représentation Graphique :

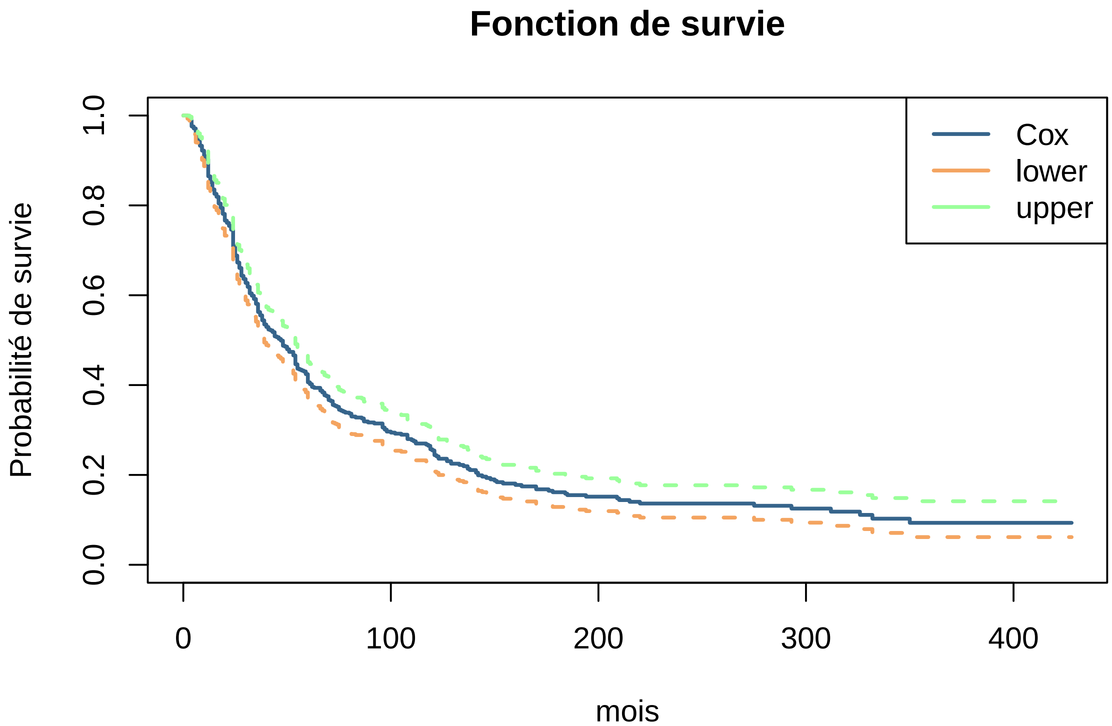
```
Call: survfit(formula = cox1)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    2    600       2   0.9970 0.00213       0.9928        1.000
```

3	597	5	0.9894	0.00398	0.9817	0.997
4	590	9	0.9758	0.00601	0.9641	0.988
5	581	3	0.9712	0.00654	0.9585	0.984
6	577	10	0.9559	0.00807	0.9402	0.972
7	567	9	0.9420	0.00921	0.9241	0.960
8	557	6	0.9327	0.00990	0.9135	0.952
9	548	7	0.9218	0.01064	0.9012	0.943
10	540	8	0.9093	0.01142	0.8872	0.932
11	528	4	0.9030	0.01179	0.8802	0.926
12	524	24	0.8647	0.01376	0.8382	0.892
13	499	8	0.8518	0.01434	0.8242	0.880
14	488	10	0.8355	0.01502	0.8066	0.865
15	477	6	0.8257	0.01541	0.7960	0.856
16	471	4	0.8191	0.01566	0.7890	0.850
17	467	9	0.8043	0.01619	0.7731	0.837
18	458	6	0.7943	0.01653	0.7626	0.827
19	452	8	0.7810	0.01696	0.7485	0.815
20	443	9	0.7660	0.01742	0.7326	0.801
21	434	3	0.7610	0.01756	0.7274	0.796
22	431	4	0.7543	0.01775	0.7203	0.790
23	426	5	0.7459	0.01798	0.7115	0.782
24	420	22	0.7087	0.01890	0.6727	0.747
25	398	12	0.6883	0.01934	0.6514	0.727
26	383	9	0.6727	0.01965	0.6353	0.712
27	374	7	0.6606	0.01987	0.6228	0.701
28	365	10	0.6432	0.02017	0.6048	0.684
29	354	4	0.6362	0.02028	0.5976	0.677
30	349	5	0.6274	0.02041	0.5886	0.669
31	342	5	0.6185	0.02054	0.5795	0.660
32	336	8	0.6042	0.02074	0.5649	0.646
33	328	3	0.5989	0.02081	0.5594	0.641
34	325	4	0.5917	0.02090	0.5521	0.634
35	319	6	0.5808	0.02102	0.5410	0.624
36	312	10	0.5626	0.02120	0.5225	0.606
37	301	4	0.5552	0.02127	0.5151	0.599
38	297	6	0.5442	0.02136	0.5039	0.588
39	289	5	0.5349	0.02143	0.4945	0.579
40	281	3	0.5293	0.02147	0.4888	0.573
41	277	3	0.5236	0.02151	0.4831	0.568
42	273	1	0.5217	0.02152	0.4812	0.566
43	272	2	0.5179	0.02155	0.4774	0.562
44	269	5	0.5084	0.02160	0.4678	0.553
45	263	1	0.5066	0.02161	0.4659	0.551
46	262	2	0.5028	0.02163	0.4621	0.547
47	260	2	0.4989	0.02165	0.4583	0.543
48	257	6	0.4874	0.02170	0.4467	0.532
49	250	1	0.4855	0.02170	0.4448	0.530
50	249	3	0.4797	0.02172	0.4390	0.524
51	245	3	0.4739	0.02174	0.4331	0.518
53	242	4	0.4660	0.02176	0.4253	0.511
54	238	10	0.4462	0.02177	0.4055	0.491
55	228	5	0.4363	0.02177	0.3957	0.481
56	223	1	0.4343	0.02177	0.3937	0.479
57	221	1	0.4323	0.02176	0.3917	0.477

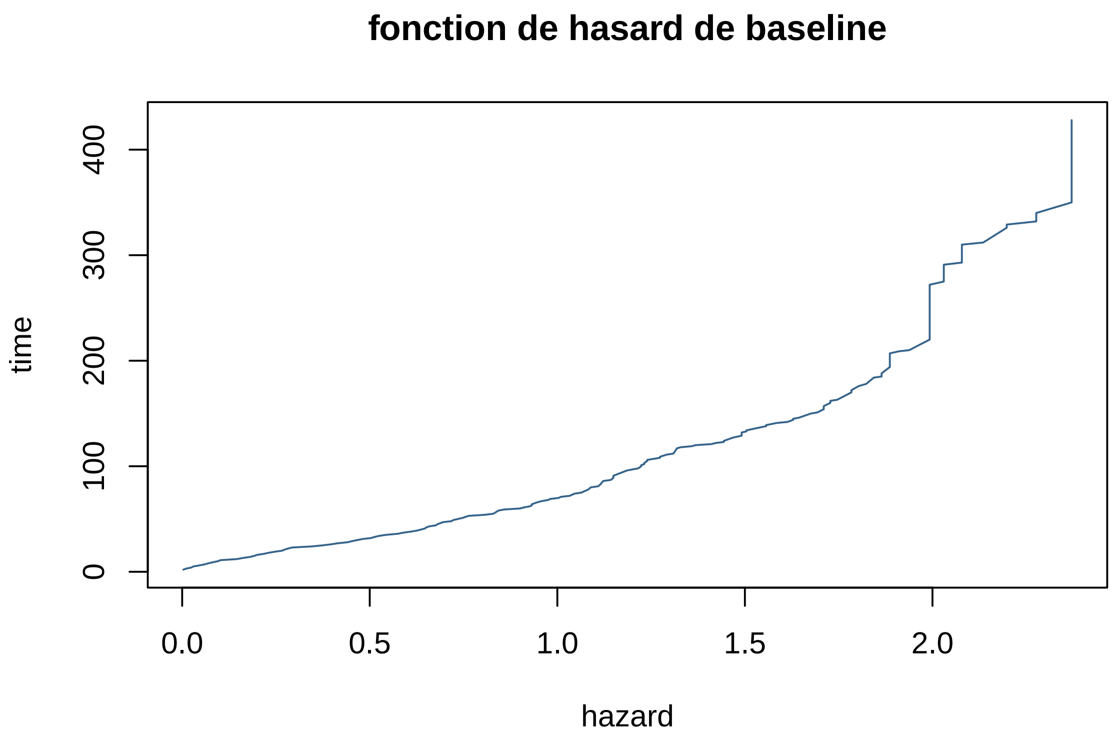
58	220	1	0.4303	0.02176	0.3897	0.475
59	217	3	0.4243	0.02175	0.3838	0.469
60	213	9	0.4061	0.02170	0.3657	0.451
61	204	2	0.4020	0.02169	0.3617	0.447
62	202	3	0.3959	0.02166	0.3556	0.441
63	198	1	0.3938	0.02165	0.3536	0.439
66	196	3	0.3876	0.02162	0.3474	0.432
67	191	2	0.3834	0.02160	0.3433	0.428
68	188	3	0.3772	0.02157	0.3372	0.422
69	184	1	0.3750	0.02155	0.3351	0.420
70	182	4	0.3665	0.02150	0.3267	0.411
71	177	1	0.3644	0.02149	0.3246	0.409
72	175	4	0.3558	0.02143	0.3162	0.400
73	170	1	0.3537	0.02141	0.3141	0.398
74	168	1	0.3515	0.02140	0.3120	0.396
75	166	3	0.3451	0.02135	0.3057	0.390
76	162	1	0.3429	0.02133	0.3036	0.387
77	161	1	0.3408	0.02131	0.3015	0.385
78	160	1	0.3386	0.02129	0.2994	0.383
80	158	1	0.3365	0.02127	0.2973	0.381
81	157	3	0.3300	0.02121	0.2909	0.374
83	154	1	0.3278	0.02119	0.2888	0.372
86	153	1	0.3256	0.02117	0.2867	0.370
87	152	3	0.3190	0.02110	0.2802	0.363
89	148	1	0.3168	0.02108	0.2781	0.361
92	145	1	0.3145	0.02105	0.2759	0.359
96	144	4	0.3055	0.02095	0.2671	0.349
97	140	2	0.3010	0.02090	0.2627	0.345
98	138	2	0.2965	0.02084	0.2583	0.340
100	136	1	0.2942	0.02081	0.2561	0.338
102	133	1	0.2919	0.02078	0.2538	0.336
105	130	1	0.2895	0.02075	0.2515	0.333
108	125	4	0.2798	0.02064	0.2421	0.323
110	120	1	0.2773	0.02061	0.2397	0.321
111	118	1	0.2749	0.02058	0.2374	0.318
112	117	2	0.2699	0.02052	0.2326	0.313
117	113	1	0.2674	0.02049	0.2301	0.311
118	111	1	0.2648	0.02046	0.2276	0.308
119	110	3	0.2570	0.02037	0.2201	0.300
120	106	1	0.2544	0.02033	0.2175	0.298
121	105	4	0.2439	0.02018	0.2074	0.287
122	100	1	0.2413	0.02014	0.2049	0.284
123	98	2	0.2360	0.02006	0.1998	0.279
127	93	2	0.2306	0.01997	0.1946	0.273
129	89	2	0.2251	0.01989	0.1893	0.268
133	86	1	0.2222	0.01985	0.1866	0.265
135	84	1	0.2194	0.01980	0.1838	0.262
137	83	2	0.2137	0.01971	0.1783	0.256
138	81	1	0.2108	0.01966	0.1756	0.253
141	79	2	0.2050	0.01956	0.1700	0.247
142	77	2	0.1991	0.01945	0.1644	0.241
144	74	1	0.1962	0.01940	0.1616	0.238
146	71	1	0.1932	0.01934	0.1588	0.235
148	68	1	0.1901	0.01929	0.1558	0.232

150	67	1	0.1870	0.01923	0.1529	0.229
151	66	1	0.1839	0.01917	0.1499	0.226
154	65	1	0.1808	0.01911	0.1470	0.222
160	62	1	0.1777	0.01904	0.1440	0.219
163	60	1	0.1745	0.01898	0.1410	0.216
170	59	2	0.1680	0.01884	0.1348	0.209
176	55	1	0.1647	0.01877	0.1318	0.206
178	54	1	0.1615	0.01869	0.1287	0.203
184	53	1	0.1582	0.01861	0.1257	0.199
185	52	1	0.1550	0.01852	0.1226	0.196
194	50	1	0.1516	0.01843	0.1195	0.192
209	41	1	0.1478	0.01838	0.1158	0.189
210	40	1	0.1440	0.01831	0.1122	0.185
215	39	1	0.1401	0.01825	0.1086	0.181
220	38	1	0.1363	0.01816	0.1050	0.177
275	26	1	0.1313	0.01821	0.1001	0.172
293	20	1	0.1251	0.01842	0.0938	0.167
312	16	1	0.1182	0.01871	0.0867	0.161
326	14	1	0.1110	0.01895	0.0795	0.155
332	11	1	0.1026	0.01937	0.0709	0.149
350	9	1	0.0934	0.01980	0.0616	0.141

```r
plot(
  survfit(cox1),
  ylim = c(0, 1),
  xlab = 'mois',
  ylab = 'Probabilité de survie',
  main = 'Fonction de survie',
  col = palette_couleur[1:3],
  lwd = 2
)
legend(
  "topright",
  legend = c("Cox" , "lower" , "upper"),
  lwd = 2,
  col = palette_couleur[1:3]
)
```




```r
# la fonction de hasard cumulée (estimateur de Breslow)
plot(basehaz(cox1),
     main = 'fonction de hasard de baseline',
     type = 'l',
     col = palette_couleur[1])
```




```r
# Fonctions de survie pour des individus ayant les caractéristiques observées
#plot(survfit(cox1, newdata = Re))

# fonction de survie pour des indiv. ayant les var explicat.
# identiques à l'ind. 1 :
#plot(survfit(cox1, newdata = Re[1,]))
```

4.1.3.3 Hasard proportionnel pour chaque variable :

4.1.3.3.1 Les résidus Schoenfeld : Test hypothèse de Hasard Proportionnel le test des résidus de Schoenfeld : (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

Graphiquement on cherche à ne pas avoir de tendance pour attester que les résidus ne dépendent pas du temps.

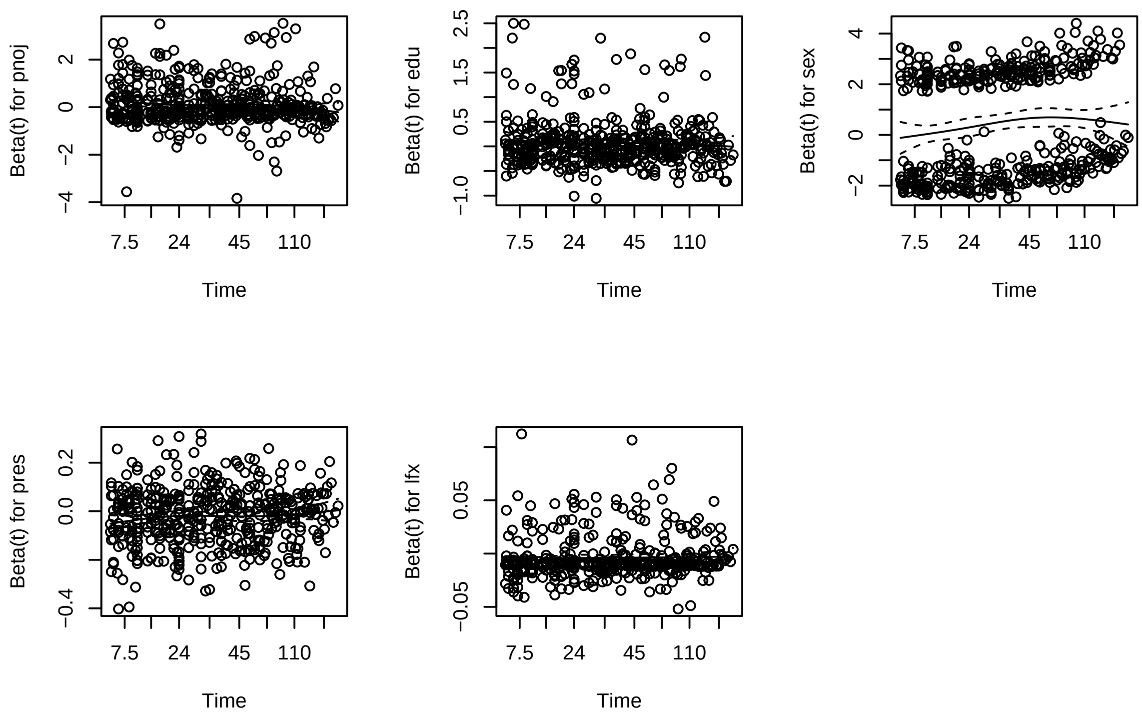
```r
# Test hypothèse de Hasard Proportionnel :
# Résidus de Schoenfeld
res = cox.zph(cox1)
res
```

```
        chisq df      p
pnoj    1.499  1 0.2208
edu     0.332  1 0.5646
sex     6.974  1 0.0083
```

```
pres     1.186  1 0.2762
lfx      0.129  1 0.7196
GLOBAL 12.944  5 0.0239
```

```r
# Projection graphique :
par(mfrow = c(2, 3))
plot(res)

# Remarque :
#on ne prend en compte que les événements correspondants à des obs
#et non des censures pour les résidus de Schoenfeld

temps = as.numeric(rownames(res$y))
length(temps)
```

```
[1] 458
```



Interprétation : Les résidus de Schoenfeld nous montrent que la proportionnalité n'est pas vérifiée dans le cadre de la variable sex. Nous pouvons réaliser un modèle stratifié sur la variable sex pour contourner le problème.

4.1.3.3.2 L'estimation linéaire : (p35 cours) On cherche à tester la nullité du coefficient β_1 dans l'équation suivante :

$$r^*_{ik} = \beta_0 + \beta_1 \times t_i + \epsilon_i$$

où :

- r^*_{ik} représente les résidus de Schoenfeld standardisé pour la k-ème covariable au temps t_i.

- β_0 est l'ordonnée à l'origine, représentant la valeur moyenne des résidus de Schoenfeld lorsque $t_i = 0$.
- β_1 est le coefficient de pente, représentant la variation des résidus de Schoenfeld en fonction du temps t_i.
- t_i est le temps d'événement pour le i-ème individu.
- ϵ_i est le terme d'erreur, représentant la variabilité non expliquée par le modèle.

```r
### test corrélations par méthode de régression linéaire (ne marche pas ?)
# temps = Re[Re$des == 1, ]$tfp  # on ne prend pas les censures
# regpnoj = lm(res$y[, 1] ~ temps)
# summary(regpnoj)
#
# regedu = lm(res$y[, 2] ~ temps)
# summary(regedu)
#
# regsex = lm(res$y[, 3] ~ temps)
# summary(regsex)
# # cela ne marche pas pour la var. "sex" .
#
# regpres = lm(res$y[, 4] ~ temps)
# summary(regpres)
#
# reglfx = lm(res$y[, 5] ~ temps)
# summary(reglfx)
```

4.1.4 Modélisation stratifiée sur la variable Sex :

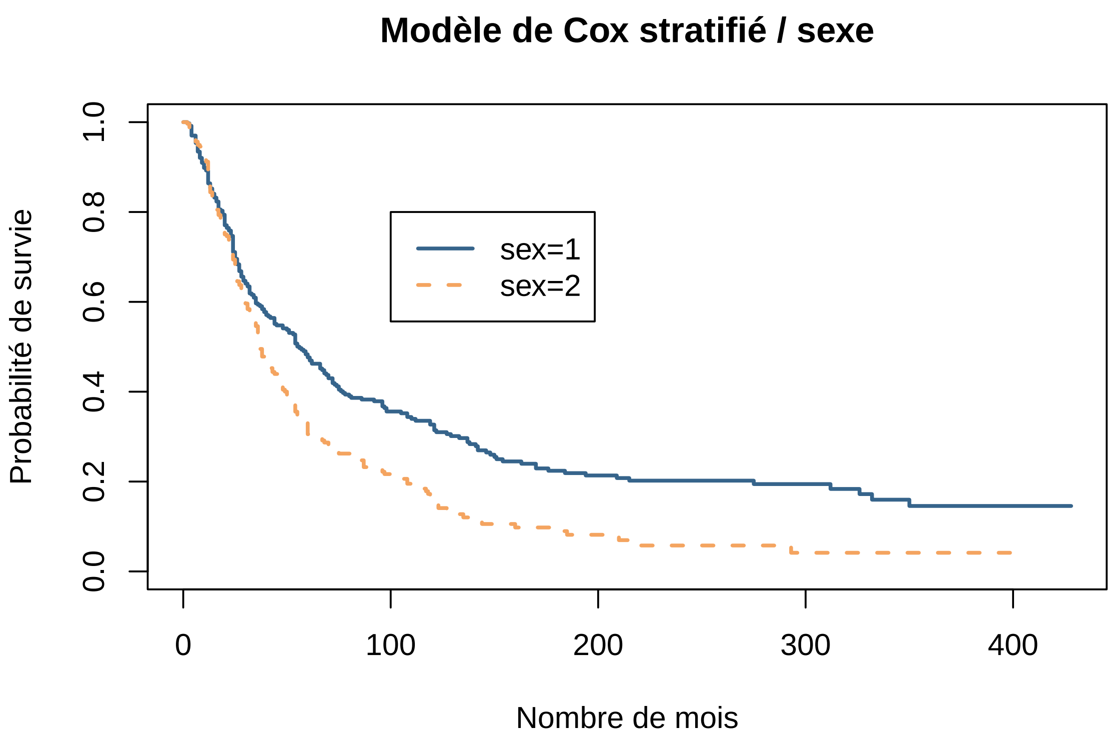
On scinde la population en deux groupe puis on applique un modèle par groupe. On rappelle que Sex 1 = Homme et Sex 2 = Femme.

```r
coxStrafin =  coxph(formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu +
                      pres + lfx,data = Re)
summary(coxStrafin)
```

```
Call:
coxph(formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu + pres +
    lfx, data = Re)

  n= 600, number of events= 458

           coef  exp(coef)   se(coef)      z Pr(>|z|)
pnoj  0.1052343  1.1109709  0.0437751  2.404  0.01622 *
edu   0.0674389  1.0697649  0.0239890  2.811  0.00494 **
pres -0.0234445  0.9768282  0.0053410 -4.390 1.14e-05 ***
lfx  -0.0045184  0.9954918  0.0008924 -5.063 4.13e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     exp(coef) exp(-coef) lower .95 upper .95
pnoj    1.1110     0.9001    1.0196    1.2105
edu     1.0698     0.9348    1.0206    1.1213
pres    0.9768     1.0237    0.9667    0.9871
lfx     0.9955     1.0045    0.9938    0.9972

Concordance= 0.619  (se = 0.015 )
```

```
Likelihood ratio test= 58.22  on 4 df,   p=7e-12
Wald test            = 52.12  on 4 df,   p=1e-10
Score (logrank) test = 52.81  on 4 df,   p=9e-11
```

```r
plot(
  survfit(coxStrafin),
  ylim = c(0, 1),
  lty = c(1, 2),
  main = 'Modèle de Cox stratifié / sexe',
  ylab = 'Probabilité de survie',
  xlab = "Nombre de mois",
  col = palette_couleur[1:2],
  lwd = 2
)
legend(100,
       0.8,
       legend = c("sex=1", "sex=2"),
       lty = c(1, 2),
       col = palette_couleur[1:2],
       lwd = 2)
```



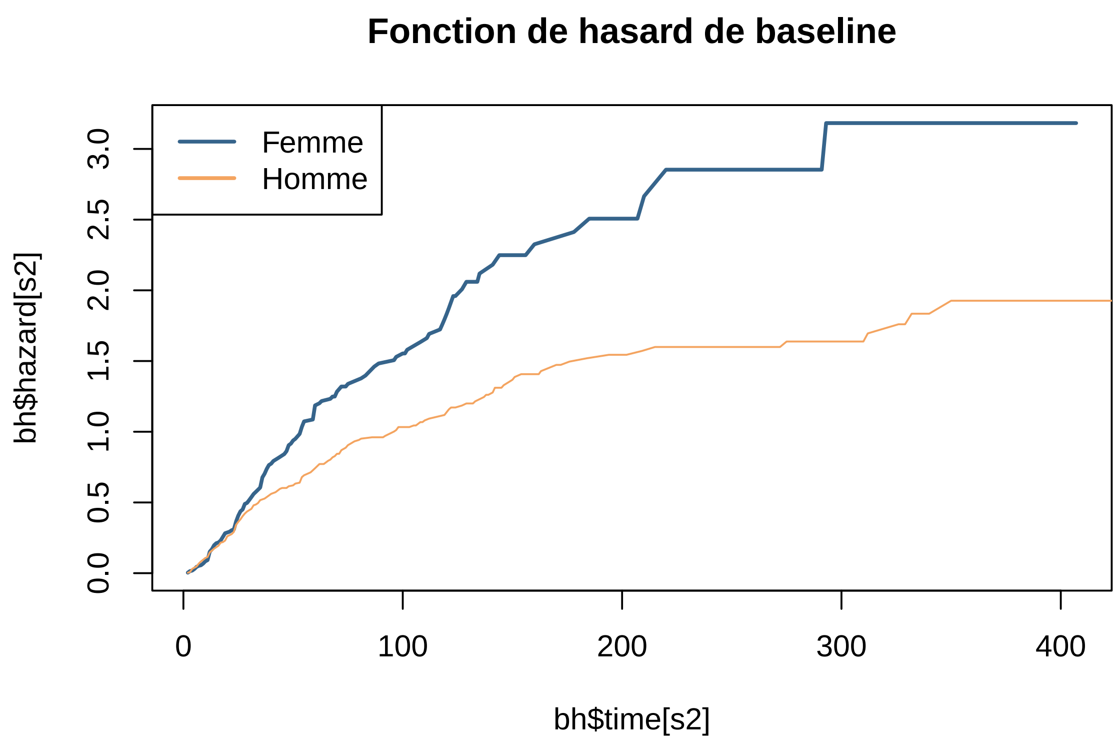
Interprétation modèle :

Exemple sur l'indicateur de prestige de l'emploi courant.

Une augmentation d'une unité de prestige est associée à une diminution de 2.3% du risque de fin d'emploi (p-valeur = 1.14e-05, très significatif).

```r
# la fonction de hasard cumulée (estimateur de Breslow)
bh = basehaz(coxStrafin)
s1 = which(bh$strata == "sex=1")
s2 = which(bh$strata == "sex=2")
plot(
  bh$time[s2],
  bh$hazard[s2],
  main = 'Fonction de hasard de baseline',
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
lines(bh$time[s1], bh$hazard[s1], col = palette_couleur[2])
legend(
  "topleft",
  lwd = 2,
  col = palette_couleur[1:2],
  legend = c("Femme", "Homme")
)
```

4.1.4.1 La fonction de hasard par sexe :



4.1.4.2 Comparaison entre les deux modèles :

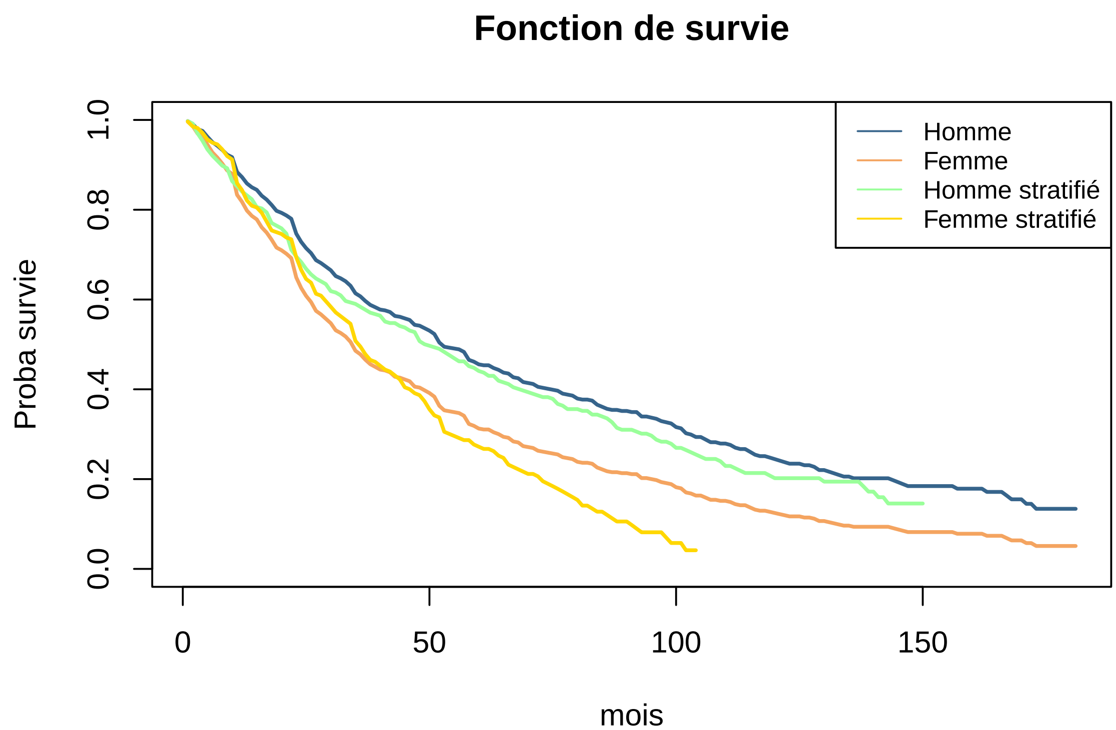
```r
# Modèle unique pour hommes :
indH = data.frame(
  sex = 1,
  pnoj = mean(Re$pnoj),
  edu = mean(Re$edu),
  pres = mean(Re$pres),
  lfx = mean(Re$lfx)
)
sH = survfit(cox1, newdata = indH)
m_homme = c(sH$surv[sH$time == 100], # 0.354
  sH$surv[sH$time == 200] # 0.202
  )


# Modèle unique pour hommes et femmes :
indF = data.frame(
  sex = 2,
  pnoj = mean(Re$pnoj),
  edu = mean(Re$edu),
  pres = mean(Re$pres),
  lfx = mean(Re$lfx)
)

sF = survfit(cox1, newdata = indF)
m_femme = c(
sF$surv[sF$time == 100], # 0.215
sF$surv[sF$time == 200] # 0.094
)

# Modèle stratifié pour hommes :
sH1 = survfit(coxStrafin, newdata = indH)
mh_strat = c(
sH1$surv[sH1$time == 101], # 0.355
sH1$surv[sH1$time == 202] # 0.213
)

# Modèle stratifié pour femmes :
sF1 = survfit(coxStrafin, newdata = indF)
mf_strat <- c(
sF1$surv[sF1$time == 100], # 0.211
sF1$surv[sF1$time == 200]  # 0.081
)

# Tableau résultats :

tab = matrix(
  c(m_homme, m_femme, mh_strat, mf_strat),
  nrow = 2,
  byrow = TRUE)
rownames(tab) = c('Mois 100 :', 'Mois 200: ')
colnames(tab) = c('Homme', 'Femme', 'Homme-strat', 'Femme-strat')
round(tab, 3)
```

4.1.4.2.1 Résultats de projection sur les mois 100 et 200 :

```
           Homme Femme Homme-strat Femme-strat
Mois 100 : 0.354 0.202       0.215       0.094
Mois 200:  0.356 0.214       0.211       0.082
```

```r
plot(
  survfit(cox1, newdata = indH)$surv,
  ylim = c(0, 1),
  xlab = 'mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie',
  col = palette_couleur[1],
  lwd = 2,
  type = 'l'
)

lines(
  survfit(cox1, newdata = indF)$surv,
  ylim = c(.1, 1),
  xlab = 'mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie',
  col = palette_couleur[2],
  lwd = 2
)

lines(
  survfit(coxStrafin, newdata = indH)$surv,
  ylim = c(0, 1),
  xlab = 'mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie',
  col = palette_couleur[3],
  lwd = 2
)

lines(
  survfit(coxStrafin, newdata = indF)$surv,
  ylim = c(0, 1),
  xlab = 'mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie',
  col = palette_couleur[4],
  lwd = 2)

legend(
  "topright",
  legend = c("Homme", "Femme", "Homme stratifié", "Femme stratifié"),
  col = palette_couleur[1:4],
  lty = 1,
  cex = 0.8
)
```

4.1.4.2.2 Représentation graphique :



4.1.5 Ajout de la variable Age au début de l'emploi :

```
# Création de la variable :
agedeb = Re$tstart - Re$tb
Re1 = data.frame(Re, agedeb)

# Génération du modèle :
coxStrafin1 =  coxph(
  formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu + pres + lfx + agedeb,
  data = Re1
)
summary(coxStrafin1)
```

```
Call:
coxph(formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu + pres +
    lfx + agedeb, data = Re1)

  n= 600, number of events= 458

            coef exp(coef)  se(coef)      z Pr(>|z|)
pnoj    0.104067  1.109675  0.043703  2.381  0.01725 *
edu     0.079170  1.082388  0.027785  2.849  0.00438 **
pres   -0.021982  0.978258  0.005626 -3.907 9.35e-05 ***
lfx    -0.003023  0.996982  0.002060 -1.467  0.14229
agedeb -0.001556  0.998445  0.001933 -0.805  0.42103
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
       exp(coef) exp(-coef) lower .95 upper .95
pnoj      1.1097     0.9012    1.0186    1.2089
edu       1.0824     0.9239    1.0250    1.1430
pres      0.9783     1.0222    0.9675    0.9891
lfx       0.9970     1.0030    0.9930    1.0010
agedeb    0.9984     1.0016    0.9947    1.0022

Concordance= 0.618  (se = 0.015 )
Likelihood ratio test= 58.87  on 5 df,   p=2e-11
Wald test            = 53.13  on 5 df,   p=3e-10
Score (logrank) test = 53.94  on 5 df,   p=2e-10
```

```
# Attention à la corrélation entre les variables explicatives : (BA)
# library(corrplot)
# corrplot(cor(Re1[, c("pnoj", "edu", "pres", "lfx", "agedeb")]),
# method = "circle", diag = TRUE)
```

Interprétation des résultats :

- On s'aperçoit que l'ajout de la variable age au début de l'emploi n'est pas significative.
- De plus la variable Expérience sur le marché de l'emploi n'est pas significative.

```
coxStrafin2 =  coxph(formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu +
                       pres + agedeb,
                     data = Re1)
summary(coxStrafin2)
```

```
Call:
coxph(formula = Surv(tfp, des) ~ strata(sex) + pnoj + edu + pres +
    agedeb, data = Re1)

  n= 600, number of events= 458

             coef  exp(coef)   se(coef)      z Pr(>|z|)
pnoj    0.0844503  1.0881187  0.0414722  2.036 0.041719 *
edu     0.0984673  1.1034784  0.0239151  4.117 3.83e-05 ***
pres   -0.0197442  0.9804495  0.0053856 -3.666 0.000246 ***
agedeb -0.0041336  0.9958749  0.0008197 -5.043 4.58e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

       exp(coef) exp(-coef) lower .95 upper .95
pnoj      1.0881     0.9190    1.0032    1.1803
edu       1.1035     0.9062    1.0529    1.1564
pres      0.9804     1.0199    0.9702    0.9909
agedeb    0.9959     1.0041    0.9943    0.9975

Concordance= 0.614  (se = 0.015 )
Likelihood ratio test= 56.71  on 4 df,   p=1e-11
Wald test            = 53.26  on 4 df,   p=8e-11
Score (logrank) test = 53.4  on 4 df,   p=7e-11
```

Interprétation :

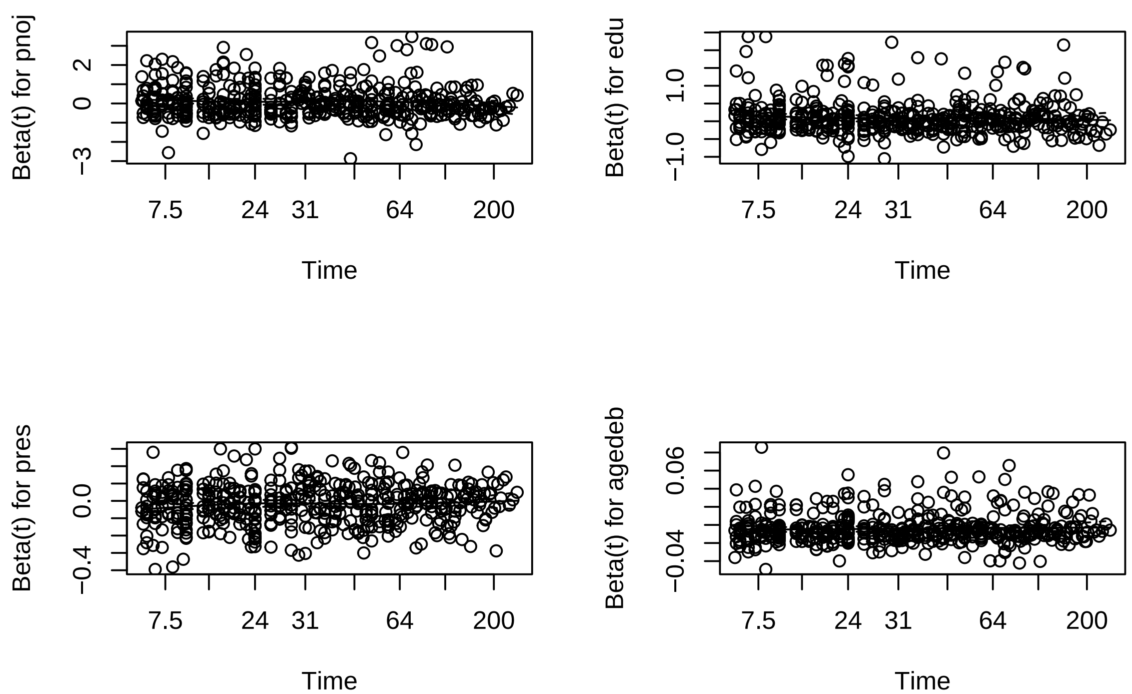
- Toutes les variables sont significatives au seuil de 5%.

```r
res = cox.zph(coxStrafin2)
res
```

4.1.5.0.1 Etude de la proportionnalité des risques :

```
        chisq df    p
pnoj   0.6697  1 0.41
edu    0.0121  1 0.91
pres   2.1183  1 0.15
agedeb 0.1839  1 0.67
GLOBAL 4.8419  4 0.30
```

```r
par(mfrow = c(2, 2))
plot(res)
```



Interprétation :

- Les résidus de Schoenfeld ne dépendent pas du temps pour les variables explicatives.

5 Examen 2020-2021 :

5.1 Exercice 1 :

5.1.1 Importation des données :

```r
Ex = read.csv("DATA/ExposuresJapon1.csv", header = TRUE, sep = ";")
DC = read.csv("DATA/DeathsJapon1.csv", header = TRUE, sep = ";")
ex = Ex
```

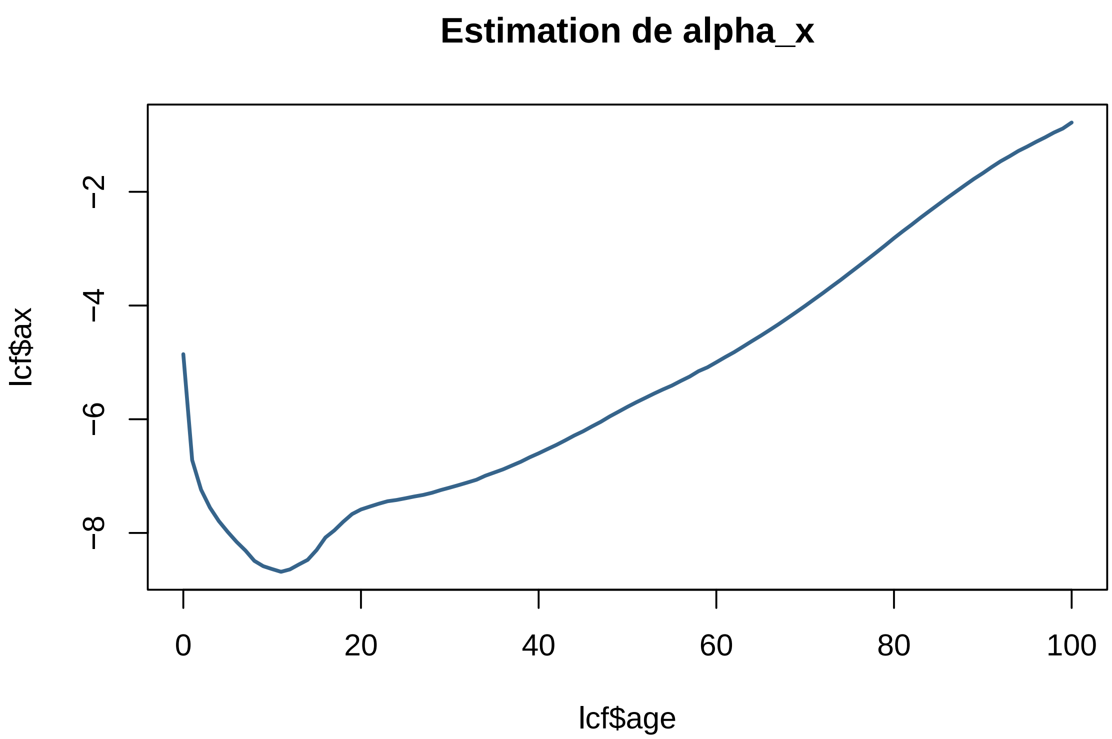
```r
de = DC
annee = unique(de$Year)
nc = length(annee)
age = unique(de$Age)
nl = length(age)
```

5.1.2 Calibration de Lee-Carter sur la base Femme :

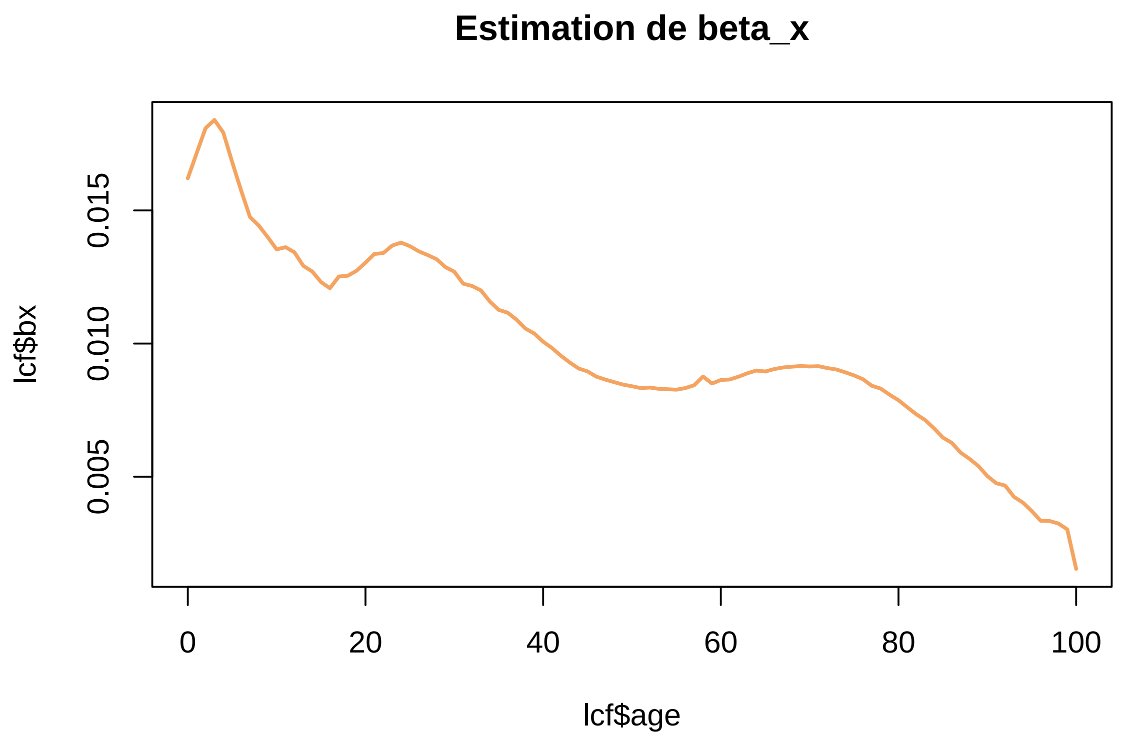
```r
library(forecast)
library(demography)

muf = matrix(de$Female / ex$Female, nl, nc)  # Données Femmes
muh = matrix(de$Male / ex$Male, nl, nc)  # H

popf = matrix(ex$Female, nl, nc)
poph = matrix(ex$Male, nl, nc)

Baseh = demogdata(
  data = muh,
  pop = poph,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Hommes',
  lambda = 1
)

Basef = demogdata(
  data = muf,
  pop = popf,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Femmes',
  lambda = 1
)

# Estimation sur la base femme :
lcf = lca(Basef)
```

```r

# Estimation de alpha_x
plot(
  lcf$age,
  lcf$ax,
  col = palette_couleur[1],
  main = "Estimation de alpha_x",
  type = 'l',
  lwd =  2
)
```

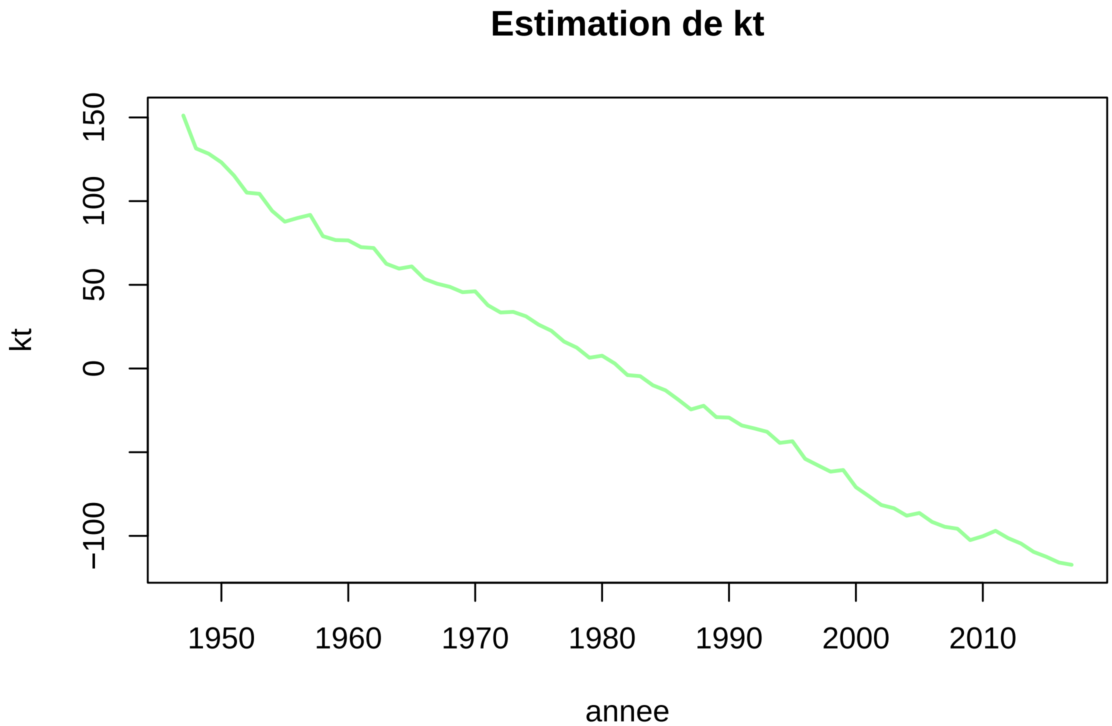
5.1.2.1 Affichage des coefficients estimés :




```r
# Estimation de beta_x
plot(
  lcf$age,
  lcf$bx,
  col = palette_couleur[2],
  main = "Estimation de beta_x",
  type = 'l',
  lwd =  2
)
```

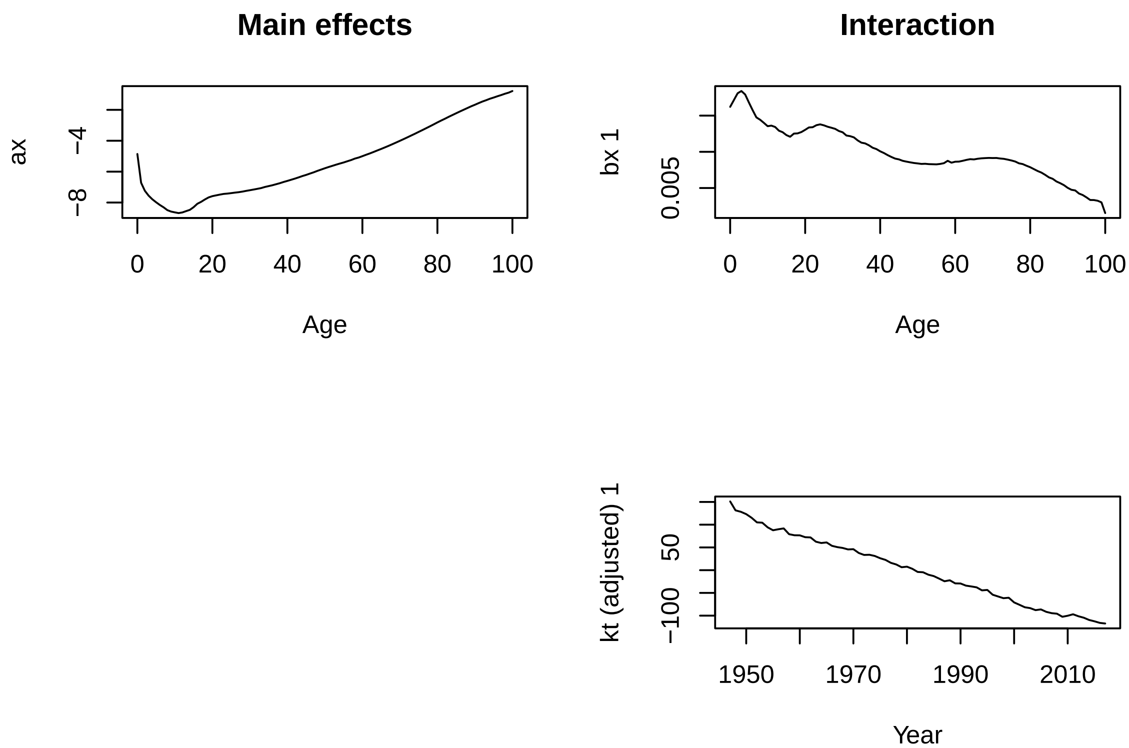



```r
# Estimation de kt :
kt = lcf$kt

plot(
  annee,
  kt,
  main = "Estimation de kt",
  col = palette_couleur[3],
  type = 'l',
  lwd =  2
)
```

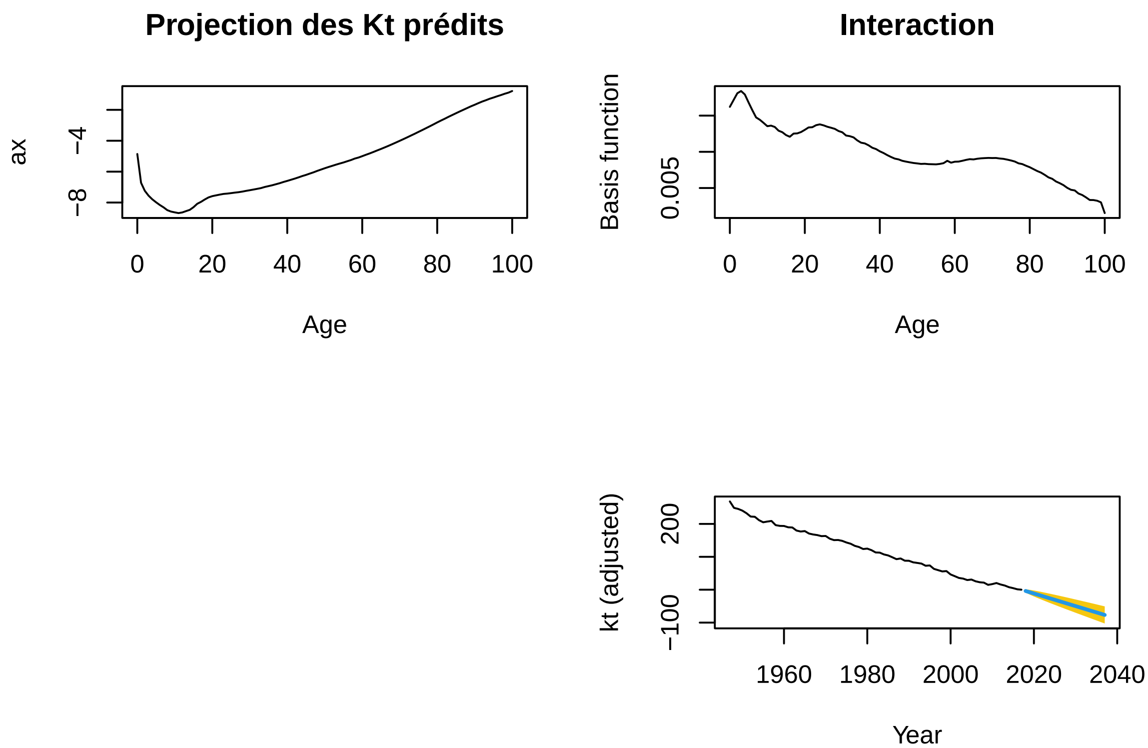


5.1.2.2 Projection des Kt : On peut projeter avec la méthode forecast ou avec une modélisation autorégressive.

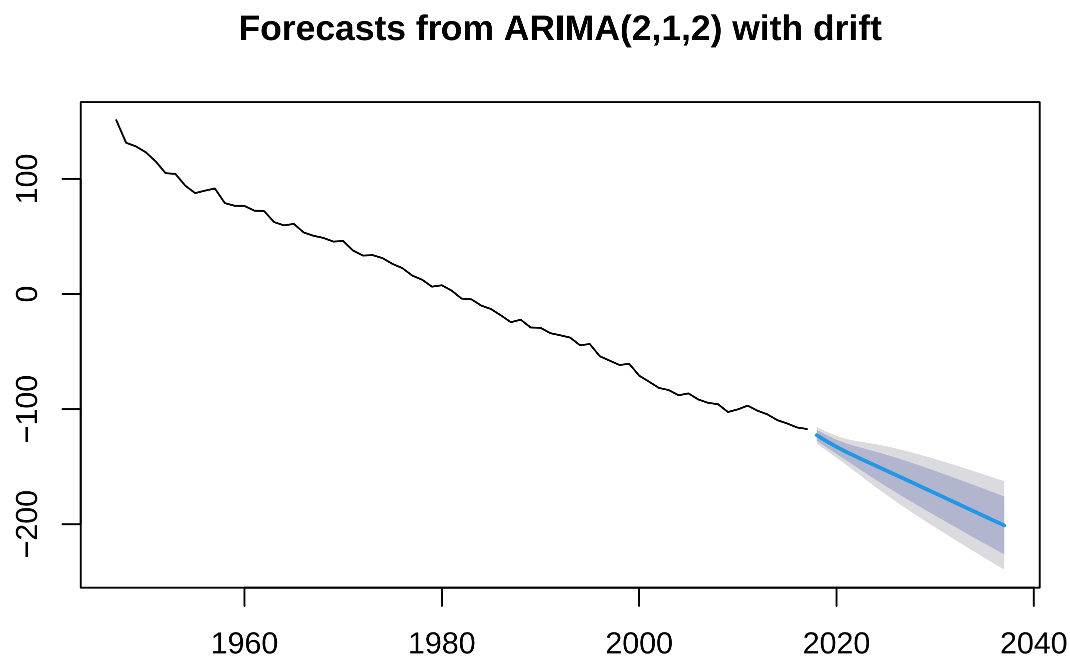
```
plot(lcf)
```




```
proj = forecast(lcf, h = 20)
plot(proj, plot.type = "component", main = "Projection des Kt prédits")
```

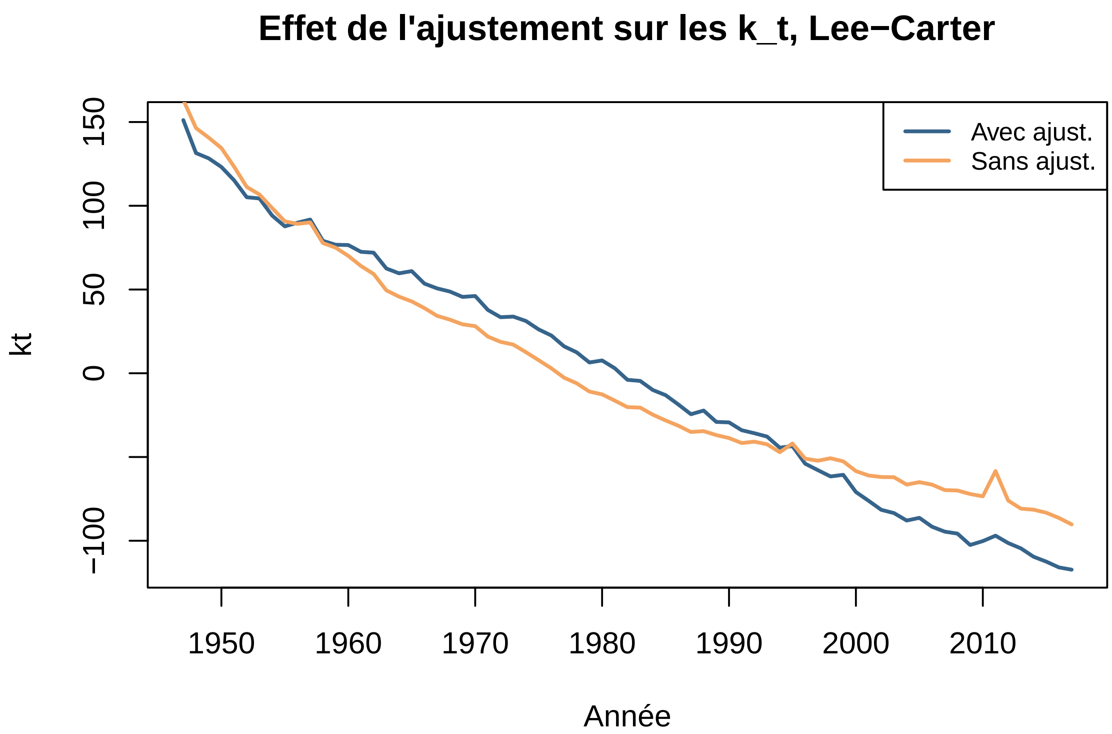



```r
ar = auto.arima(kt)
plot(forecast(ar, h = 20))
```



5.1.3 Modélisation sans ajustement des KT :

```
lcf_sans = lca(Basef, adjust = "none")
plot(lcf$year,
     lcf$kt,
     col = palette_couleur[1],
     type = 'l',
     main = "Effet de l'ajustement sur les k_t, Lee-Carter",
     ylab = "kt",
     xlab = "Année",
     lwd = 2)
lines(lcf_sans$year, lcf_sans$kt, col = palette_couleur[2], lwd = 2)
legend('topright',
  legend = c("Avec ajust.", "Sans ajust."),
  col = palette_couleur[1:2],
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```



5.2 Exercice 2 :

5.2.1 Importation des données :

```
Ex = read.csv("DATA/ExposuresJapon1.csv", header = TRUE, sep = ";")
DC = read.csv("DATA/DeathsJapon1.csv", header = TRUE, sep = ";")
ex = Ex
de = DC
annee = unique(de$Year)
nc = length(annee)
age = unique(de$Age)
nl = length(age)
```

5.2.2 Question 1 : Calibration de Lee-Carter

```
library(forecast)
library(demography)

ind = which((de$Age > 29) & (de$Age < 101) & (de$Year < 2011))

annee = 1947:2010
nc = length(annee)
age = 30:100
nl = length(age)

muf = matrix(de$Female[ind] / ex$Female[ind], nl, nc)
```

```r
muh = matrix(de$Male[ind] / ex$Male[ind], nl, nc)
mui = matrix(de$Total[ind] / ex$Total[ind], nl, nc)

popf = matrix(ex$Female[ind], nl, nc)
poph = matrix(ex$Male[ind], nl, nc)
popi = matrix(ex$Total[ind], nl, nc)

Baseh = demogdata(
  data = muh,
  pop = poph,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Hommes',
  lambda = 1
)
Basef = demogdata(
  data = muf,
  pop = popf,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Femmes',
  lambda = 1
)
Basei = demogdata(
  data = mui,
  pop = popi,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Individus',
  lambda = 1
)

lch = lca(Baseh)
lcf = lca(Basef)
lci = lca(Basei)

predh = lch$fitted$y
predf = lcf$fitted$y  # c'est log(mu_{x,t})
predi = lci$fitted$y

# RMSE sur période calibration (idem précédemment)
rmsef = sqrt(sum((log(muf) - (predf)) ^ 2) / nl / nc)
rmseh = sqrt(sum((log(muh) - (predh)) ^ 2) / nl / nc)
rmsei = sqrt(sum((log(mui) - (predi)) ^ 2) / nl / nc)
c(rmsef, rmseh, rmsei)
```

5.2.2.1 Modélisation sur la base Femme :

```
[1] 0.1540581 0.1074303 0.1204481
```

5.2.2.2 Projection du modèle de Lee-Carter : On projette de 2011 à 2017 avec la méthode forecast.

```
projh = forecast(lch, h = 7)$rate$Hommes
projf = forecast(lcf, h = 7)$rate$Femmes
proji = forecast(lci, h = 7)$rate$Individus

# On peut le calculer d'une autre manière :

kp = forecast(lcf, h = 7)$kt.f$mean[7] + lcf$kt[length(lcf$kt)]
pf = exp(lcf$ax + lcf$bx * kp)
```

Commentaire sur la deuxième méthode :

forecast(lcf,h=7)$kt.f$ mean renvoie les delta kt d'une marche aléatoire avec drift il faut ajouter la valeur du dernier kt avant projection.

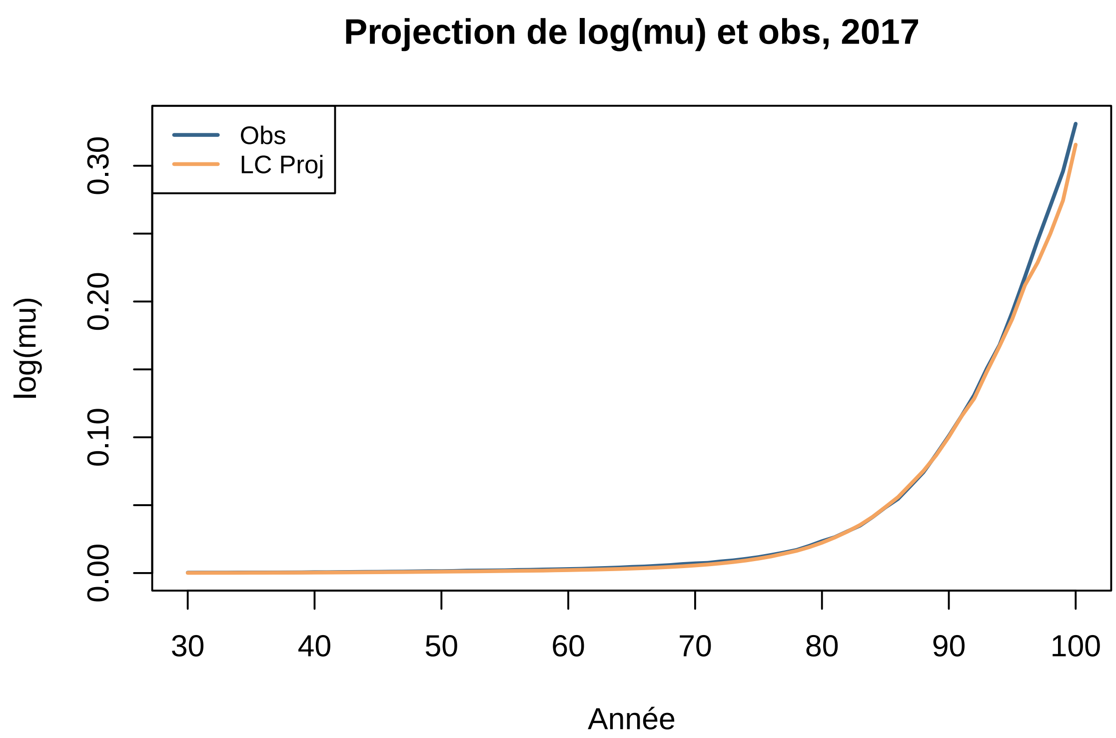
```
ind = which((de$Age > 29) & (de$Age < 101) & (de$Year >= 2011))
annee = 2011:2017
nc = length(annee)

muf = matrix(de$Female[ind] / ex$Female[ind], nl, nc)
muh = matrix(de$Male[ind] / ex$Male[ind], nl, nc)
mui = matrix(de$Total[ind] / ex$Total[ind], nl, nc)
rmsef = sqrt(sum((log(muf) - log(projf)) ^ 2) / nl / nc)
rmseh = sqrt(sum((log(muh) - log(projh)) ^ 2) / nl / nc)
rmsei = sqrt(sum((log(mui) - log(proji)) ^ 2) / nl / nc)
E1 = c(rmsef, rmseh, rmsei)
E1
```

5.2.2.3 Calcul du RMSE de prédiction :

```
[1] 0.3264988 0.1418898 0.1994374
```

```
plot(
  30:100,
  muf[,7],
  col = palette_couleur[1],
  type = 'l',
  main = "Projection de log(mu) et obs, 2017",
  ylab = "log(mu)",
  xlab = "Année",
  lwd = 2
)
lines(30:100, projf[,7], col = palette_couleur[2], lwd = 2)
legend(
  "topleft",
  legend = c("Obs", "LC Proj"),
  col = palette_couleur[1:2],
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```

5.2.2.4 Représentation graphique des log(mu et obs, 2017)




```
colnames(muf)= annee
rownames(muf)= age
idx = which(rownames(muf)==90) #Récupération de l'âge 90

dt = data.frame(calcul_main_kt = pf[idx],
                estimé = projf[idx,7],
                observé = muf[idx,7])
dt
```

5.2.2.5 Comparaison des q_obs(x=90,2017) et mu(x=90,2017) :

```
   calcul_main_kt    estimé   observé
90      0.1002805 0.1002805 0.1009978
```

5.2.3 Question 2 : Modélisation log-Linéaire

```
ind = which((de$Age > 29) & (de$Age < 101) & (de$Year < 2011))
annee = 1947:2010
nc = length(annee)
age = 30:100
nl = length(age)

muf1 = matrix(de$Female[ind] / ex$Female[ind], nl, nc)

al = rep(0, nl)
```

```r
be = rep(0, nl)

lg = log(muf1/(1-muf1))

# On utilise la fonction lm pour chaque âges :
for (i in 1:nl)
{
  reg = lm(lg[i, ] ~ annee)
  be[i] = reg$coefficients[2]
  al[i] = reg$coefficients[1]
}

# Autres méthodes possibles :

# al2 = rep(0, nl)
# be2 = rep(0, nl)
# mt = mean(annee)
# mt2 = mean(annee ^ 2)
# deno = mt2 - mt ^ 2
# for (i in 1:nl)
# {
#   be2[i] = (sum(annee * lg[i, ]) / nc - mt / nc * sum(lg[i, ])) / deno
#   al2[i] = 1 / nc * sum(lg[i, ]) - be2[i] * mt
# }

# Prédiction sur les années 1947- 2010 :
# lgpred = lg
# for (i in 1:nl)
# {
#   for (j in 1:nc)
#   {
#     lgpred[i, j] = al[i] + be[i] * annee[nc]
#   }
# }
#
# # on en déduit les q(x,t) =exp(lg(x,t))/(1+exp(lg(x,t)))
# qpred = exp(lgpred) / (1 + exp(lgpred))
# dim(qpred)
```

5.2.3.1 Calibration de modèle pour les femmes :

```r
# Prédiction pour les femmes 2011-2017 :
npa = 7

lgpred = matrix(0, nl, npa)
an = 2011:2017
for (i in 1:nl)
{
  for (j in 1:npa)
  {
    lgpred[i, j] = al[i] + be[i] * an[j]
  }
```

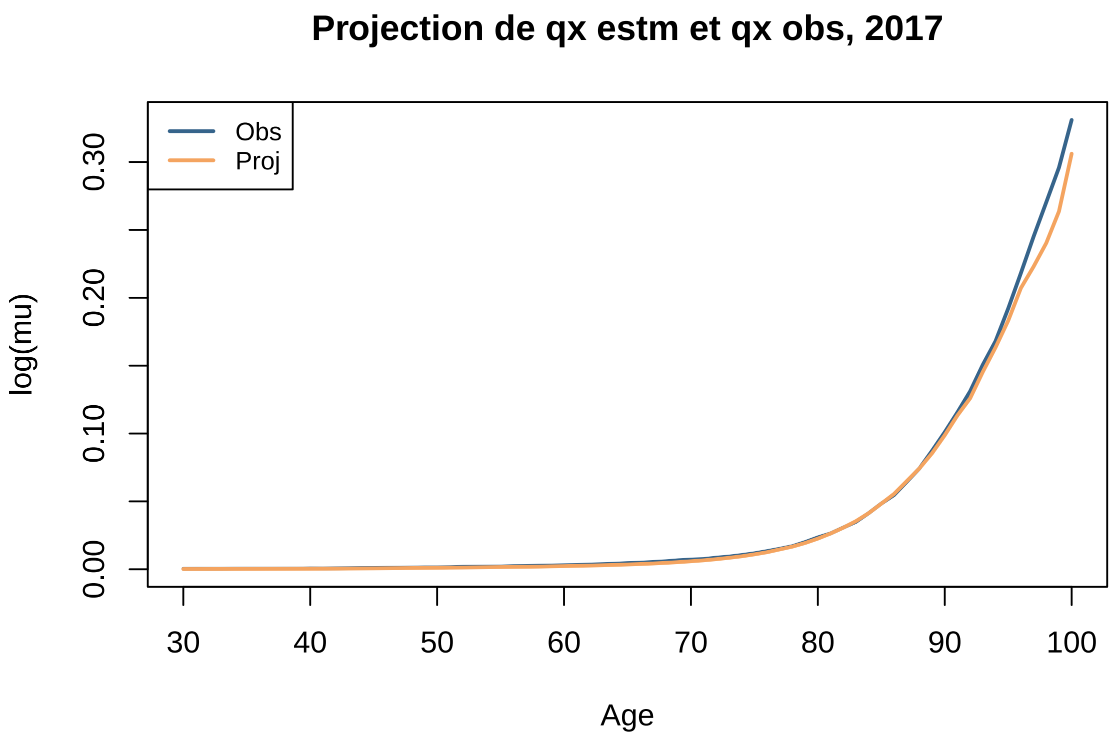
```
}

qpred = exp(lgpred) / (1 + exp(lgpred))
dim(qpred)
```

5.2.3.2 Prédiction sur les années 2011-2017 :

```
[1] 71  7
```

```
plot(x = age,
     y = muf[,7],
     main = "Projection de qx estm et qx obs, 2017",
     xlab = "Age",
     ylab = "log(mu)",
     type = "l",
     col = palette_couleur[1],
     lwd = 2
     )
lines(x = age,
      y = qpred[,7],
      col = palette_couleur[2],
      lwd = 2
      )
legend("topleft",
       legend = c("Obs", "Proj"),
       col = palette_couleur[1:2],
       lty = 1,
       cex = 0.8,
       lwd = 2
       )
```

5.2.3.3 Représentation graphiques de l’estimation et de l’observé :




```
idx = which(rownames(muf) == 90)
dt = round(data.frame(q_obs = muf[idx,7],
                LC_estim = projf[idx,7],
                Log_lin_estim = qpred[idx,7]),6)
dt
```

5.2.3.4 Comparaison des coefficients :

```
     q_obs LC_estim Log_lin_estim
1 0.100998  0.10028      0.098725
```

```
npa = 7
nl = length(age)
E2 = sqrt(sum((log(muf)-log(qpred))^2)/nl/npa)
E2
```

5.2.3.5 Calcul du RMSE projection log_lin :

```
[1] 0.2612912
```

5.2.4 Comparaion du modèle question 1 et question 2 :

```
dt = round(data.frame(err_m1 = E1[1],
                err_mod2 = E2),5)
rownames(dt) = c("Mod F")
dt
```

```
      err_m1 err_mod2
Mod F 0.3265  0.26129
```

6 Examen 2022-2023 :

Examen de Mars 2023. Données Veteran. L'objectif est d'étudier les relations entre le temps de survie pour les patients atteints du cancer des poumons et les variables explicatives trt, celltype , karno , age et prior.

6.1 Exercice 1 :

6.1.1 Remplacer la variable prior par une variable binaire :

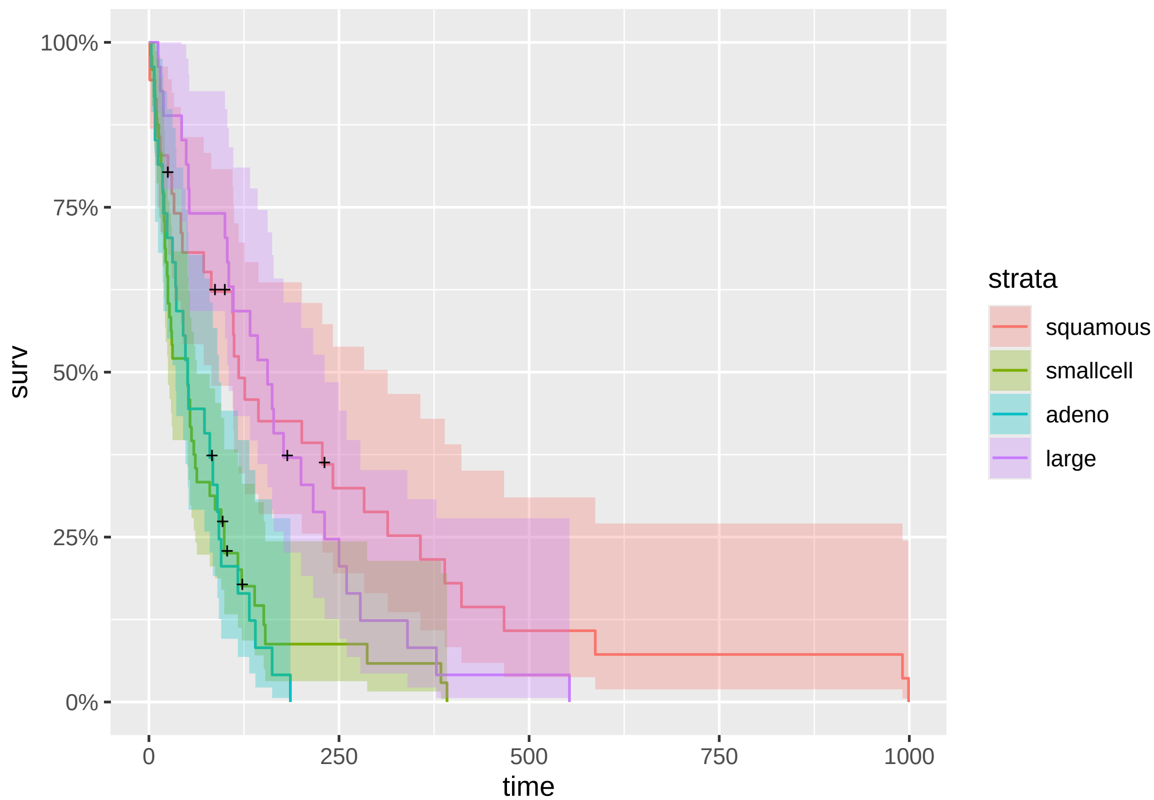
```
library(survival)
x = veteran
x$prior = 1*(x$prior == 10)
levels(x$prior) = c("Non", "Oui")
```

6.1.2 Test de comparaison des durées

```
survdiff(Surv(time,status)~celltype,data=x)
```

```
Call:
survdiff(formula = Surv(time, status) ~ celltype, data = x)

                    N Observed Expected (O-E)^2/E (O-E)^2/V
celltype=squamous  35       31     47.7      5.82     10.53
celltype=smallcell 48       45     30.1      7.37     10.20
celltype=adeno     27       26     15.7      6.77      8.19
celltype=large     27       26     34.5      2.12      3.02

 Chisq= 25.4  on 3 degrees of freedom, p= 1e-05
```

```
library(ggfortify)
s = survfit(Surv(time,status)~celltype,data = x,type = "kaplan-meier")
autoplot(s)
```




```
s
```

```
Call: survfit(formula = Surv(time, status) ~ celltype, data = x, type = "kaplan-meier")

                    n events median 0.95LCL 0.95UCL
celltype=squamous  35     31    118      82     314
celltype=smallcell 48     45     51      25      63
celltype=adeno     27     26     51      35      92
celltype=large     27     26    156     105     231
```

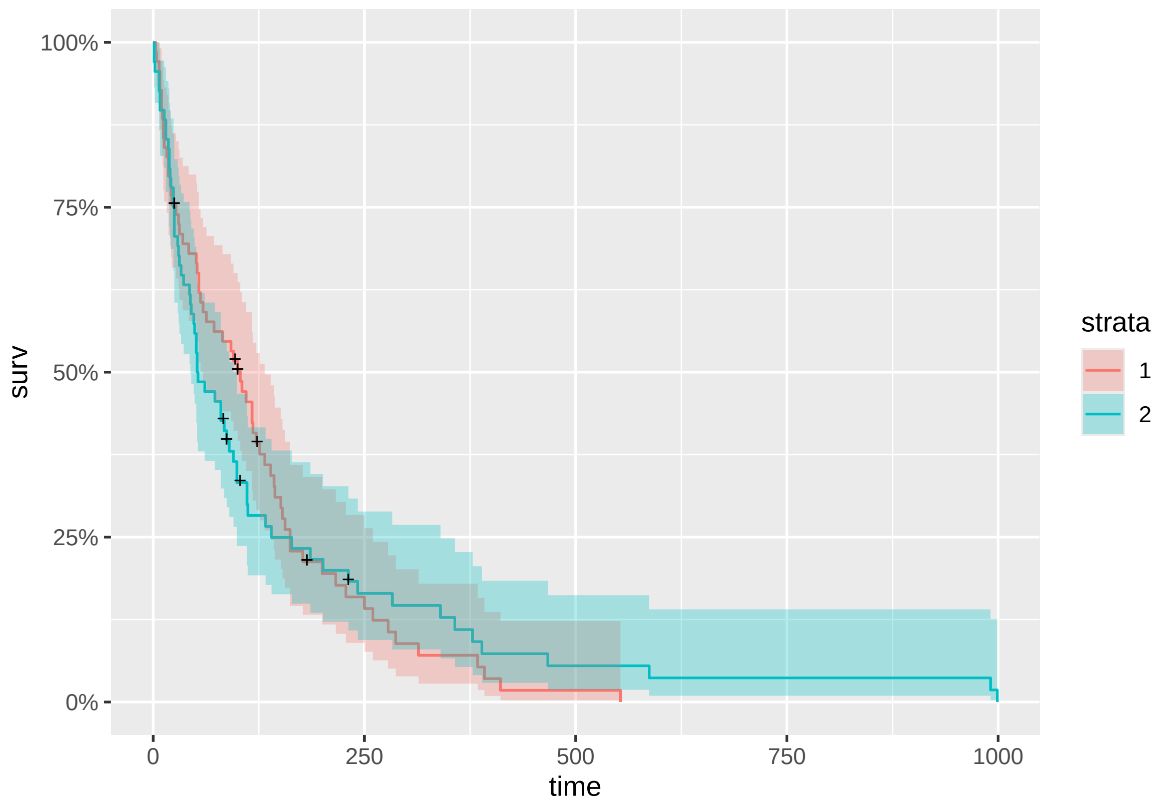
La p-value est très petite, la différence entre les lois de survie selon celltype est donc significative.

6.1.3 Test de comparaison des durées de survie selon le traitement.

```
survdiff(Surv(time,status)~trt,data=x)
```

```
Call:
survdiff(formula = Surv(time, status) ~ trt, data = x)

       N Observed Expected (O-E)^2/E (O-E)^2/V
trt=1 69       64     64.5   0.00388   0.00823
trt=2 68       64     63.5   0.00394   0.00823

 Chisq= 0  on 1 degrees of freedom, p= 0.9
```

```
s = survfit(Surv(time,status)~trt,data = x,type = "kaplan-meier")
autoplot(s)
```




```
s
```

```
Call: survfit(formula = Surv(time, status) ~ trt, data = x, type = "kaplan-meier")

       n events median 0.95LCL 0.95UCL
trt=1 69     64  103.0      59     132
trt=2 68     64   52.5      44      95
```

La p-value est grande, la différence entre les lois de survie selon le traitement n'est donc pas significative.

6.1.4 Modélisation de Cox (variables explicatives):

```
cox0 = coxph(
  formula = Surv(time, status) ~ trt + celltype + karno + age + prior +
    diagtime,
  data = x
)
summary(cox0)
```

6.1.4.1 Sélection des variables explicatives :

```
Call:
coxph(formula = Surv(time, status) ~ trt + celltype + karno +
    age + prior + diagtime, data = x)

  n= 137, number of events= 128

                        coef  exp(coef)   se(coef)      z Pr(>|z|)
trt                2.946e-01  1.343e+00  2.075e-01  1.419  0.15577
```

```
celltypesmallcell  8.616e-01  2.367e+00  2.753e-01  3.130  0.00175 **
celltypeadeno      1.196e+00  3.307e+00  3.009e-01  3.975 7.05e-05 ***
celltypelarge      4.013e-01  1.494e+00  2.827e-01  1.420  0.15574
karno             -3.282e-02  9.677e-01  5.508e-03 -5.958 2.55e-09 ***
age               -8.706e-03  9.913e-01  9.300e-03 -0.936  0.34920
prior              7.159e-02  1.074e+00  2.323e-01  0.308  0.75794
diagtime           8.132e-05  1.000e+00  9.136e-03  0.009  0.99290
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

                  exp(coef) exp(-coef) lower .95 upper .95
trt                  1.3426     0.7448    0.8939    2.0166
celltypesmallcell    2.3669     0.4225    1.3799    4.0597
celltypeadeno        3.3071     0.3024    1.8336    5.9647
celltypelarge        1.4938     0.6695    0.8583    2.5996
karno                0.9677     1.0334    0.9573    0.9782
age                  0.9913     1.0087    0.9734    1.0096
prior                1.0742     0.9309    0.6813    1.6937
diagtime             1.0001     0.9999    0.9823    1.0182

Concordance= 0.736  (se = 0.021 )
Likelihood ratio test= 62.1  on 8 df,   p=2e-10
Wald test            = 62.37  on 8 df,   p=2e-10
Score (logrank) test = 66.74  on 8 df,   p=2e-11
```

Dans le summary, les hypothèses $H_0 : \beta_j = 0$ sont testées. Les quantités $Pr(>|z|)$ sont $P(|U| > z)$, avec U de loi normale $N(0,1)$. La quantité $se(coef)$ est l'écart-type de l'estimateur de β_j.

On peut calibrer des modèles plus simples en retirant les variables les moins significatives.

```r
cox1 = coxph(formula = Surv(time, status) ~ trt + celltype + karno + age + prior, data = x)
summary(cox1)
```

```
Call:
coxph(formula = Surv(time, status) ~ trt + celltype + karno +
    age + prior, data = x)

  n= 137, number of events= 128

                       coef exp(coef)  se(coef)      z Pr(>|z|)
trt                0.294784  1.342837  0.206542  1.427  0.15351
celltypesmallcell  0.861956  2.367788  0.271676  3.173  0.00151 **
celltypeadeno      1.196000  3.306864  0.300834  3.976 7.02e-05 ***
celltypelarge      0.401366  1.493865  0.282563  1.420  0.15548
karno             -0.032823  0.967710  0.005438 -6.036 1.58e-09 ***
age               -0.008716  0.991322  0.009239 -0.943  0.34547
prior              0.072526  1.075221  0.207315  0.350  0.72646
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

                  exp(coef) exp(-coef) lower .95 upper .95
trt                  1.3428     0.7447    0.8958    2.0129
celltypesmallcell    2.3678     0.4223    1.3902    4.0327
celltypeadeno        3.3069     0.3024    1.8338    5.9633
celltypelarge        1.4939     0.6694    0.8586    2.5991
karno                0.9677     1.0334    0.9575    0.9781
```

```
age                    0.9913     1.0088    0.9735    1.0094
prior                  1.0752     0.9300    0.7162    1.6142

Concordance= 0.736  (se = 0.021 )
Likelihood ratio test= 62.1  on 7 df,   p=6e-11
Wald test            = 62.36  on 7 df,   p=5e-11
Score (logrank) test = 66.63  on 7 df,   p=7e-12
```

```
# Sélection des variables explicatives :

cox2 = coxph(formula=Surv(time,status)~trt+celltype+karno+age,data=x)
#summary(cox2)

cox3 = coxph(formula=Surv(time,status)~trt+celltype+karno,data=x)
#summary(cox3)
```

Nous pouvons tracer le graphe de la fonction de survie (Kaplan Meier ou Aalen, c'est Aalen par défaut). Les covariables sont fixées à leur valeur moyenne.

```
summary(survfit(cox1))
```

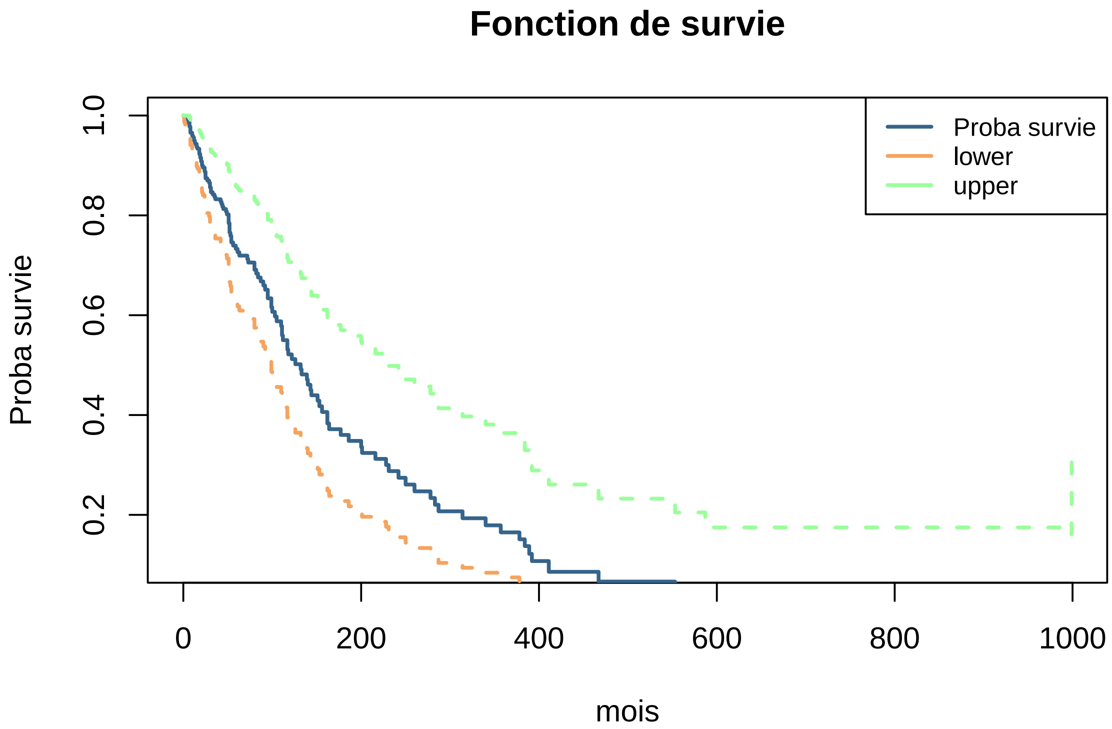
6.1.4.2 Affichage graphique du modèle à sept variables :

```
Call: survfit(formula = cox1)

 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    1    137       2  0.99460 0.00403     9.87e-01        1.000
    2    135       1  0.99188 0.00507     9.82e-01        1.000
    3    134       1  0.98912 0.00602     9.77e-01        1.000
    4    133       1  0.98630 0.00693     9.73e-01        1.000
    7    132       3  0.97763 0.00950     9.59e-01        0.996
    8    129       4  0.96550 0.01278     9.41e-01        0.991
   10    125       2  0.95924 0.01439     9.31e-01        0.988
   11    123       1  0.95607 0.01519     9.27e-01        0.986
   12    122       2  0.94971 0.01678     9.17e-01        0.983
   13    120       2  0.94325 0.01835     9.08e-01        0.980
   15    118       2  0.93664 0.01994     8.98e-01        0.977
   16    116       1  0.93330 0.02073     8.94e-01        0.975
   18    115       3  0.92289 0.02317     8.79e-01        0.969
   19    112       2  0.91556 0.02485     8.68e-01        0.966
   20    110       2  0.90793 0.02657     8.57e-01        0.962
   21    108       2  0.89994 0.02832     8.46e-01        0.957
   22    106       1  0.89580 0.02921     8.40e-01        0.955
   24    105       2  0.88740 0.03100     8.29e-01        0.950
   25    103       3  0.87425 0.03373     8.11e-01        0.943
   27     99       1  0.86972 0.03467     8.04e-01        0.940
   29     98       1  0.86517 0.03560     7.98e-01        0.938
   30     97       2  0.85588 0.03748     7.85e-01        0.933
   31     95       2  0.84655 0.03933     7.73e-01        0.927
   33     93       1  0.84186 0.04024     7.67e-01        0.925
   35     92       1  0.83712 0.04117     7.60e-01        0.922
   36     91       1  0.83226 0.04212     7.54e-01        0.919
   42     90       1  0.82736 0.04306     7.47e-01        0.916
   43     89       1  0.82248 0.04400     7.41e-01        0.913
```

44	88	1	0.81756	0.04493	7.34e-01	0.911
45	87	1	0.81264	0.04585	7.28e-01	0.908
48	86	1	0.80756	0.04679	7.21e-01	0.905
49	85	1	0.80196	0.04781	7.14e-01	0.901
51	84	3	0.78422	0.05091	6.91e-01	0.891
52	81	3	0.76534	0.05402	6.66e-01	0.879
53	78	1	0.75889	0.05504	6.58e-01	0.875
54	77	2	0.74591	0.05706	6.42e-01	0.867
56	75	1	0.73941	0.05804	6.34e-01	0.862
59	74	1	0.73290	0.05901	6.26e-01	0.858
61	73	1	0.72615	0.06000	6.18e-01	0.854
63	72	1	0.71935	0.06098	6.09e-01	0.849
72	71	1	0.71243	0.06196	6.01e-01	0.845
73	70	1	0.70553	0.06293	5.92e-01	0.840
80	69	2	0.69110	0.06489	5.75e-01	0.831
82	67	1	0.68340	0.06589	5.66e-01	0.826
84	65	1	0.67557	0.06690	5.56e-01	0.820
87	64	1	0.66765	0.06789	5.47e-01	0.815
90	62	1	0.65951	0.06888	5.37e-01	0.809
92	61	1	0.65104	0.06985	5.28e-01	0.803
95	60	2	0.63379	0.07172	5.08e-01	0.791
99	57	2	0.61575	0.07355	4.87e-01	0.778
100	55	1	0.60666	0.07441	4.77e-01	0.772
103	53	1	0.59741	0.07527	4.67e-01	0.765
105	51	1	0.58782	0.07608	4.56e-01	0.758
110	50	1	0.57830	0.07685	4.46e-01	0.750
111	49	2	0.55942	0.07827	4.25e-01	0.736
112	47	1	0.54994	0.07894	4.15e-01	0.729
117	46	2	0.53106	0.08018	3.95e-01	0.714
118	44	1	0.52151	0.08074	3.85e-01	0.706
122	43	1	0.51203	0.08126	3.75e-01	0.699
126	41	1	0.50181	0.08179	3.65e-01	0.691
132	40	1	0.49162	0.08230	3.54e-01	0.683
133	39	1	0.48132	0.08275	3.44e-01	0.674
139	38	1	0.47102	0.08314	3.33e-01	0.666
140	37	1	0.46074	0.08348	3.23e-01	0.657
143	36	1	0.45009	0.08377	3.13e-01	0.648
144	35	1	0.43958	0.08398	3.02e-01	0.639
151	34	1	0.42878	0.08426	2.92e-01	0.630
153	33	1	0.41757	0.08450	2.81e-01	0.621
156	32	1	0.40604	0.08463	2.70e-01	0.611
162	31	2	0.38310	0.08466	2.48e-01	0.591
164	29	1	0.37164	0.08455	2.38e-01	0.580
177	28	1	0.36012	0.08433	2.28e-01	0.570
186	26	1	0.34815	0.08398	2.17e-01	0.559
200	25	1	0.33603	0.08351	2.06e-01	0.547
201	24	1	0.32401	0.08291	1.96e-01	0.535
216	23	1	0.31213	0.08225	1.86e-01	0.523
228	22	1	0.29988	0.08146	1.76e-01	0.511
231	21	1	0.28764	0.08065	1.66e-01	0.498
242	19	1	0.27431	0.07971	1.55e-01	0.485
250	18	1	0.26071	0.07875	1.44e-01	0.471
260	17	1	0.24712	0.07758	1.34e-01	0.457
278	16	1	0.23374	0.07623	1.23e-01	0.443

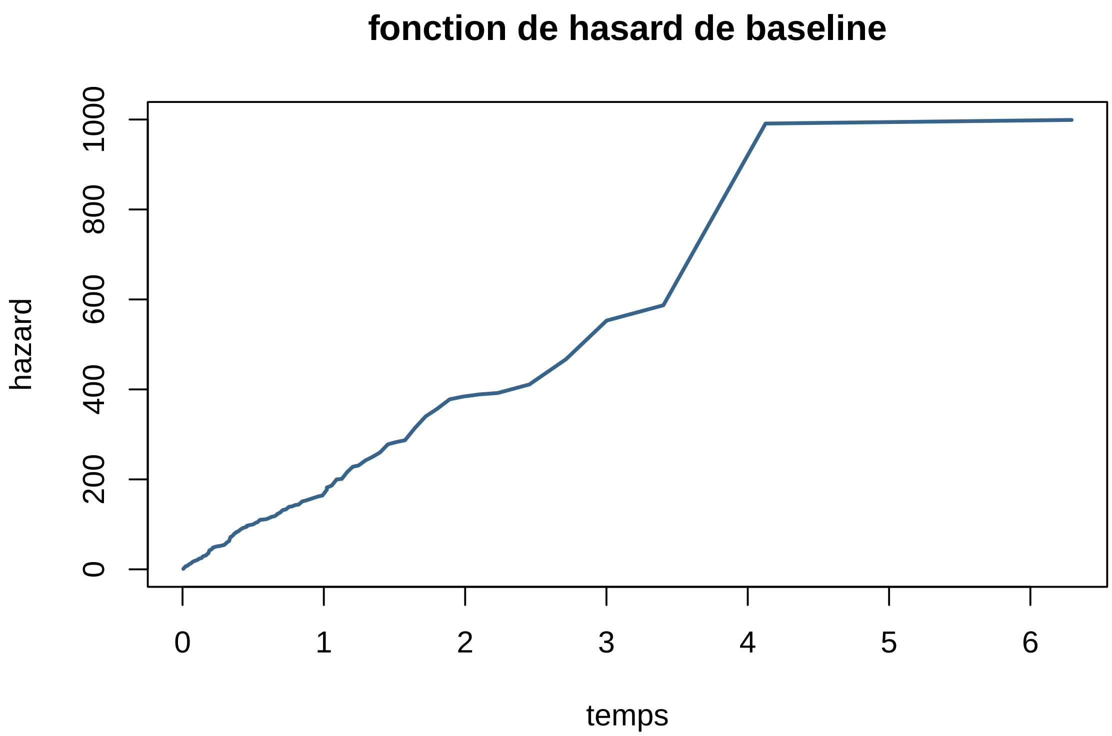
```
  283     15       1  0.22021 0.07471     1.13e-01         0.428
  287     14       1  0.20713 0.07308     1.04e-01         0.414
  314     13       1  0.19321 0.07105     9.40e-02         0.397
  340     12       1  0.17898 0.06898     8.41e-02         0.381
  357     11       1  0.16488 0.06658     7.47e-02         0.364
  378     10       1  0.15102 0.06408     6.57e-02         0.347
  384      9       1  0.13740 0.06131     5.73e-02         0.329
  389      8       1  0.12186 0.05785     4.81e-02         0.309
  392      7       1  0.10742 0.05423     3.99e-02         0.289
  411      6       1  0.08590 0.04869     2.83e-02         0.261
  467      5       1  0.06632 0.04246     1.89e-02         0.233
  553      4       1  0.04972 0.03592     1.21e-02         0.205
  587      3       1  0.03329 0.02817     6.34e-03         0.175
  991      2       1  0.01615 0.01894     1.62e-03         0.161
  999      1       1  0.00185 0.00486     1.07e-05         0.320
```

```r
plot(
  survfit(cox1),
  ylim = c(.1, 1),
  xlab = 'mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie',
  col = palette_couleur[1:3],
  lwd = 2
)
legend(
  "topright",
  legend = c('Proba survie', "lower", "upper"),
  col = palette_couleur[1:3],
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```



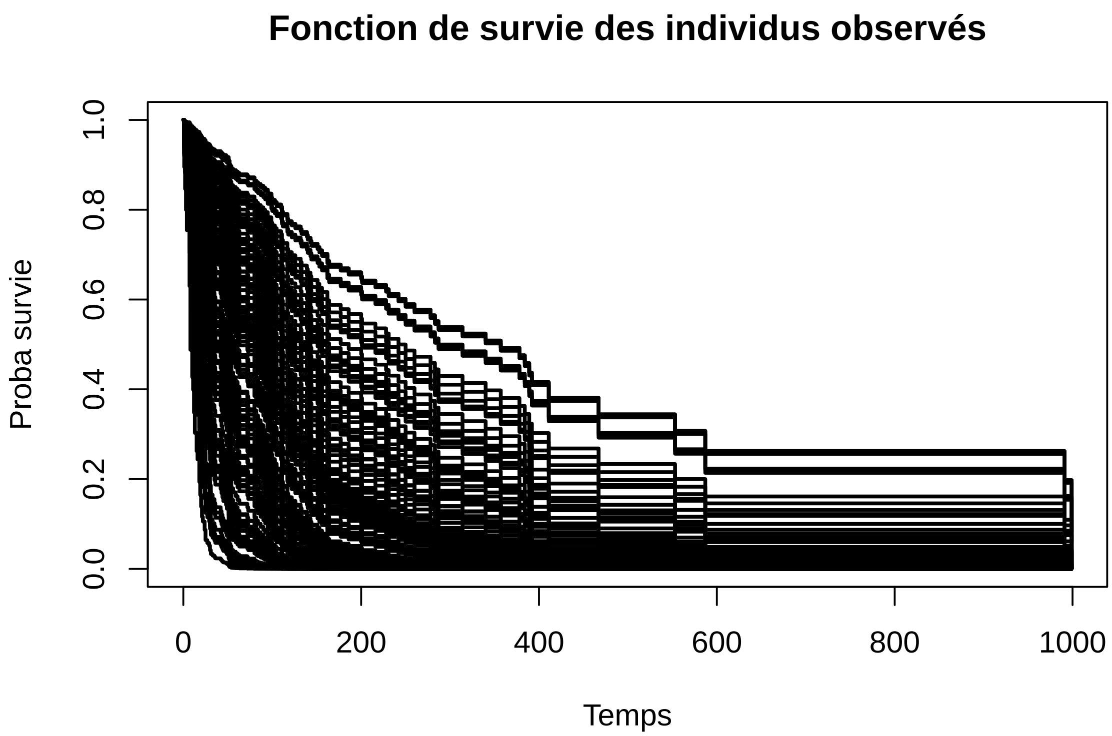
6.1.4.3 Fonction de hasard du modèle à 7 paramètres : Nous pouvons tracer la fonction de hasard cumulée (estimateur de Breslow).

```
plot(
  basehaz(cox1),
  main = 'fonction de hasard de baseline',
  xlab = 'temps',
  ylab = 'hazard',
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
```



6.1.4.4 Fonction de survie des individus observés : Nous pouvons tracer les fonctions de survie pour des individus ayant les caractéristiques observées.

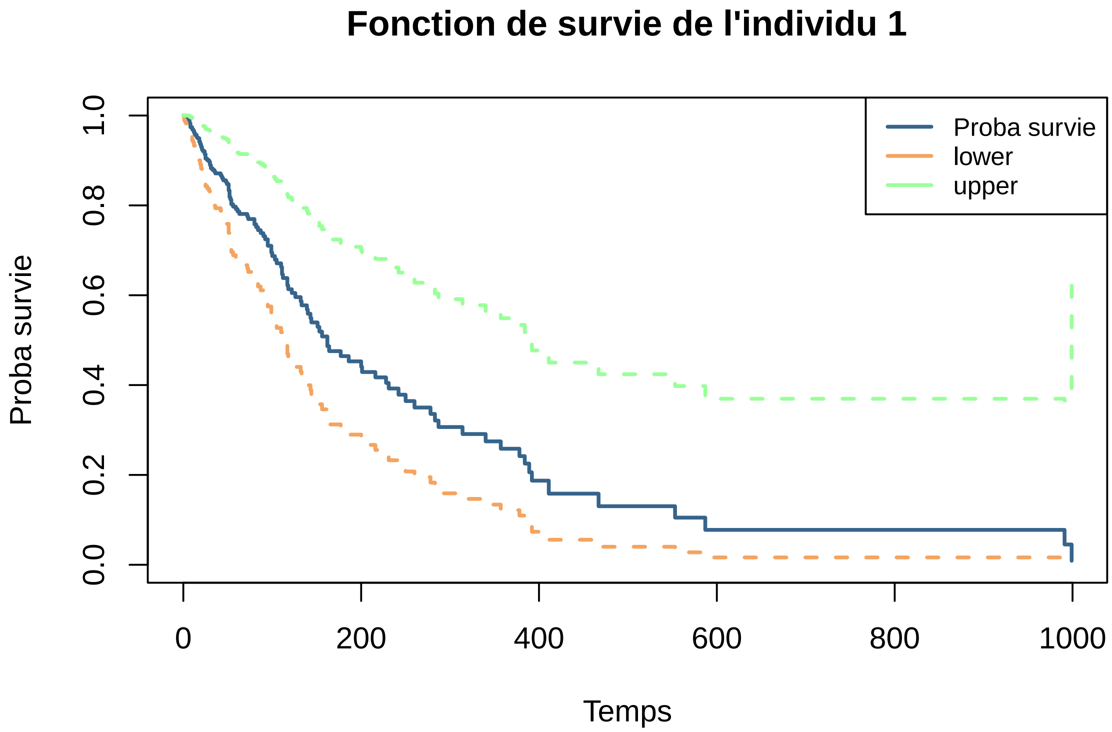
```
plot(survfit(cox1,newdata=veteran),
     main = 'Fonction de survie des individus observés',
     xlab = 'Temps',
     ylab = 'Proba survie',
     lwd = 2)
```



Ou la fonction de survie pour des individus ayant les var explicatives identiques à l'individu 1 par exemple.

```
tcox1 = survfit(cox1, newdata = veteran[1, ])

plot(tcox1,
     main = "Fonction de survie de l'individu 1",
     xlab = "Temps",
     ylab = "Proba survie",
     lwd = 2 ,
     col = palette_couleur[1:3])
legend("topright",
  legend = c('Proba survie', "lower", "upper"),
  col = palette_couleur[1:3],
  lty = 1,
  cex = 0.8,
  lwd = 2)
```

6.1.4.5 Représentation graphique du premier individu :

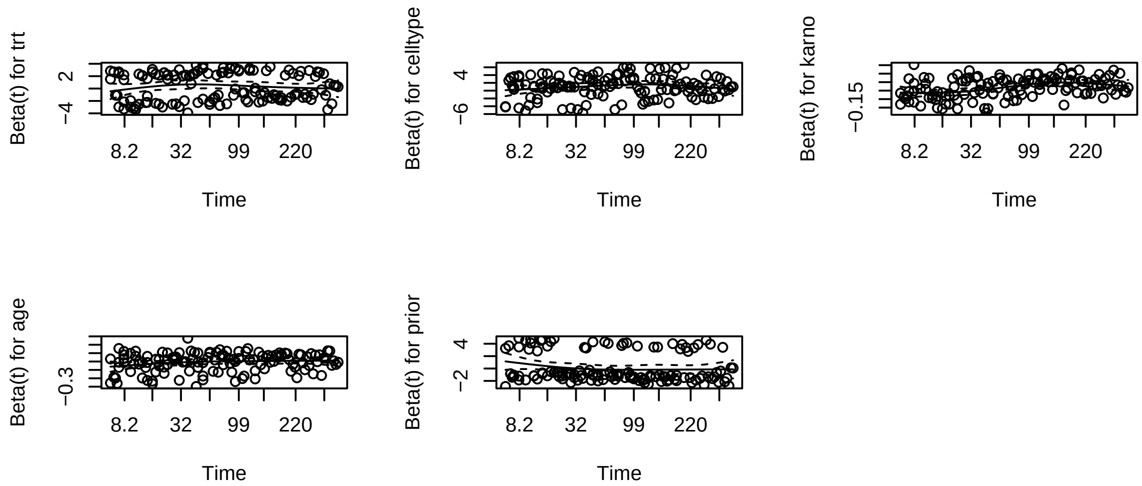


6.1.4.6 Test de proportionnalité des résidus : On test l'hypothèse de Hasard Proportionnel, avec les résidus de Schoenfeld.

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

```
          chisq df       p
trt       0.262  1 0.60896
celltype 15.118  3 0.00172
karno    12.913  1 0.00033
age       1.826  1 0.17655
prior     2.131  1 0.14437
GLOBAL   31.716  7 4.6e-05
```

```r
par(mfrow = c(3, 3))
plot(res)
```



Les variables celltype, karno et Global ne respectent pas l'hypothèse de proportionnalité.

6.2 Exercice 2 :Forêt aléatoire de survie : package randomForestSRC

Voir :

Référence1

Référence2

Référence3

Référence4

6.2.1 Importation des données et modélisation :

```r
library(randomForestSRC)
data(veteran, package = "randomForestSRC")
v.obj <- rfsrc(Surv(time, status) ~ ., data = veteran,
                   ntree = 100)

## plot tree number 3
plot(get.tree(v.obj, 3))
```

```r
print(v.obj)
```

6.2.1.1 Résultats de l'apprentissage :

```
                         Sample size: 137
```

```
                     Number of deaths: 128
                      Number of trees: 100
            Forest terminal node size: 15
        Average no. of terminal nodes: 6.21
No. of variables tried at each split: 3
               Total no. of variables: 6
        Resampling used to grow trees: swor
     Resample size used to grow trees: 87
                             Analysis: RSF
                               Family: surv
                       Splitting rule: logrank *random*
        Number of random split points: 10
                           (OOB) CRPS: 61.89332342
                    (OOB) stand. CRPS: 0.06195528
   (OOB) Requested performance error: 0.28174828
```

```r
get.cindex(
  time = veteran$time,
  censoring = veteran$status,
  predicted = v.obj$predicted.oob
)
```

6.2.1.1.1 Affichage du C-index du modèle :

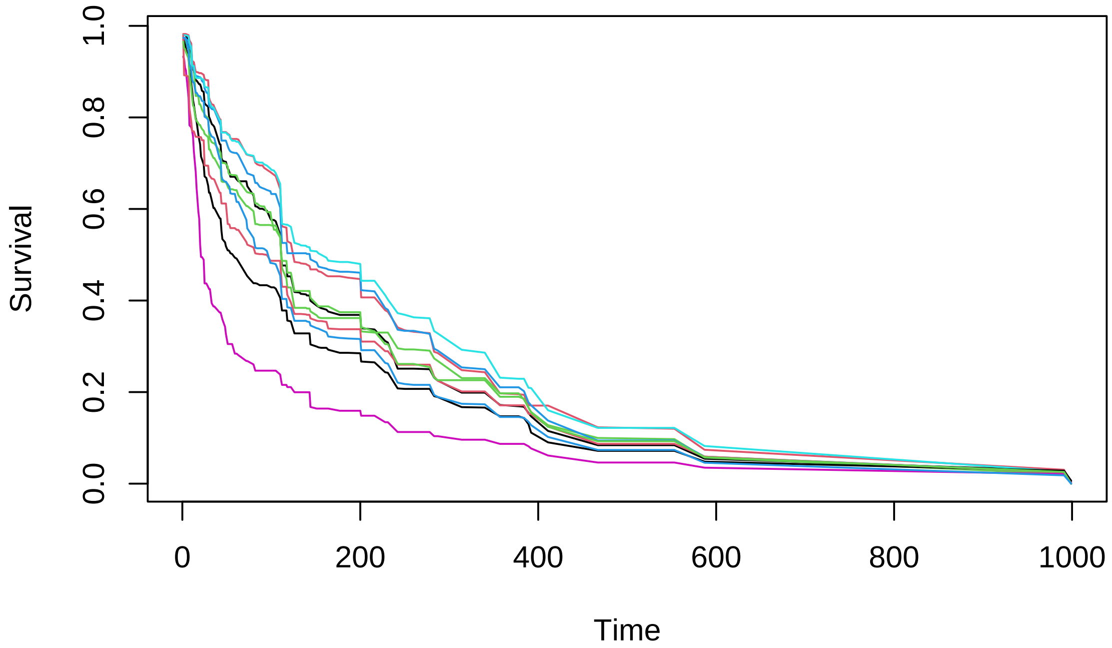
```
[1] 0.2817483
```

Explication du C-index :

Le C-index est une mesure de la qualité du modèle de forêt aléatoire, plus est élevé meilleur est le modèle.

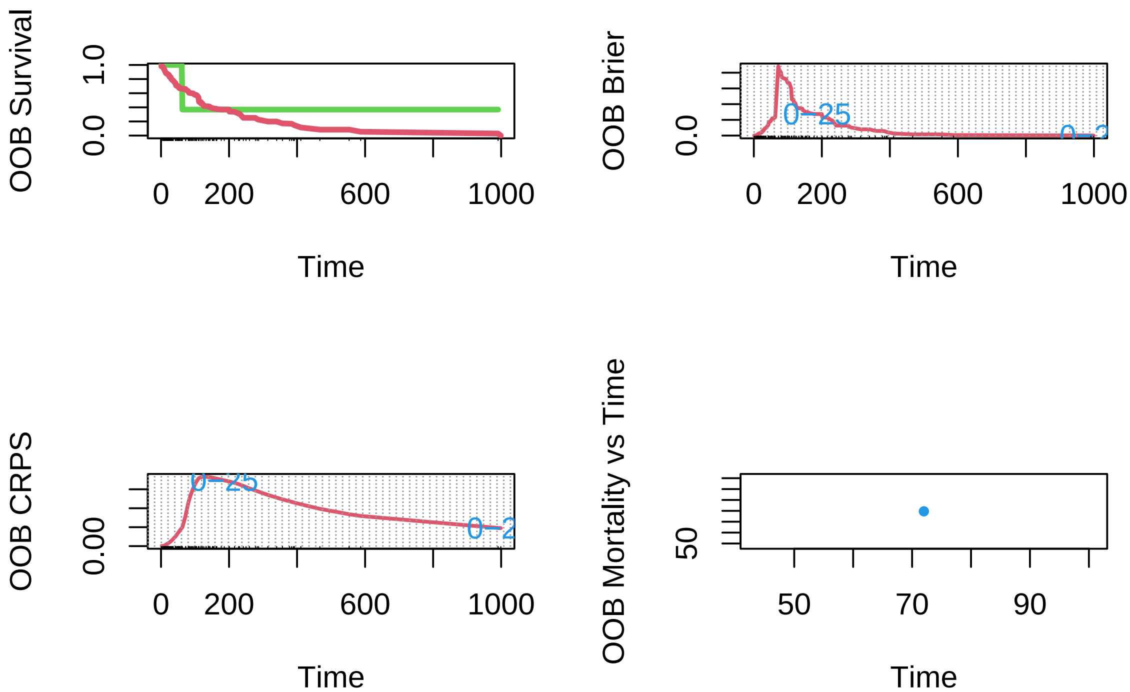
```r
matplot(
  v.obj$time.interest,
  t(v.obj$survival.oob[1:10,]),
  xlab = "Time",
  ylab = "Survival",
  type = "l",
  lty = 1
)
```

6.2.1.2 Graphique de la fonction de survie pour les 10 premiers individus :



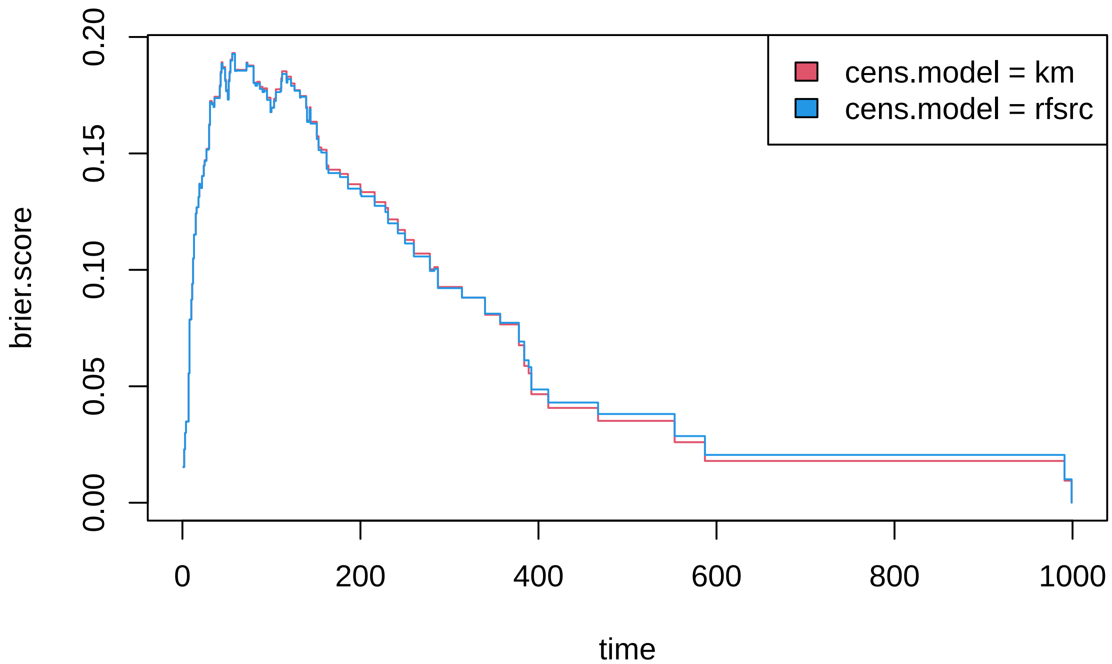
6.2.1.3 Synthèse des résultats : La fonction plot survival permet d'avoir une synthèse graphique de résultats :

```
plot.survival(v.obj, subset = 1)
```




```r
## obtain Brier score using KM and RSF censoring distribution estimators
bs.km <- get.brier.survival(v.obj, cens.model = "km")$brier.score
bs.rsf <- get.brier.survival(v.obj, cens.model = "rfsrc")$brier.score

# plot the brier score
 plot(bs.km, type = "s", col = 2)
 lines(bs.rsf, type ="s", col = 4)
 legend("topright", legend = c("cens.model = km", "cens.model = rfsrc"), fill = c(2,4))
```

6.2.1.4 Performances du modèle RSF :



6.2.2 Importance des variables (VIMP) :

(Problème sur les sorties pdf mais sortent en local)

Plusieurs méthodes sont possibles pour l'importance d'une variable x :

- importance = "permute" : calcul d'importance par permutation aléatoire des valeurs de x observées sur les exemples OOB.
- importance = "random" : calcul d'importance par choix aléatoire gauche droite lorsqu'une coupure se fait avec la variable x.
- importance = "anti" : calcul d'importance en choisissant le choix opposé à celui proposé.

```
# imp1 = subsample(v.obj, importance = "anti")
# plot(imp1)
```

```
  # imp2 = subsample(v.obj, importance = "permute")
  # plot(imp2)
```

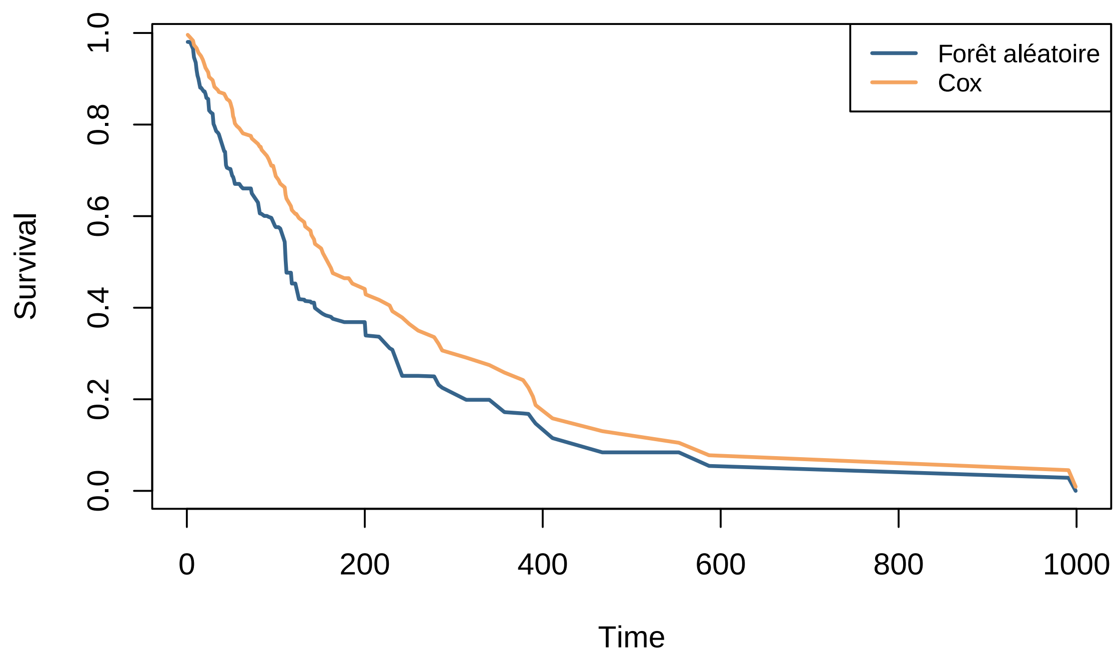
```
  # imp3 = subsample(v.obj, importance = "random")
  # plot(imp3)
```

6.2.3 Comparaison de modèle entre Cox et forêt aléatoire de survie :

Comparaison des fonctions de survie entre une forêt aléatoire de survie et le modèle de Cox, pour l'individu 1:

```
plot(
  v.obj$time.interest,
  t(v.obj$survival.oob[1, ]),
```

```
  xlab = "Time",
  ylab = "Survival",
  type = "l",
  lty = 1,
  col = palette_couleur[1],
  lwd = 2
)
lines(tcox1$time, tcox1$surv, col = palette_couleur[2],
      lwd = 2)
legend(
  "topright",
  col = palette_couleur[1:2],
  legend = c("Forêt aléatoire", "Cox"),
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```



7 Examen 2023-2024 :

Examen de Janvier 2024. Données PBC (Primary Biliary Cirrhosis). L'objectif est d'étudier le temps de survie pour les patients atteints de cirrhose biliaire primitive. La cirrhose biliaire primitive est une maladie chronique du foie rare mais mortelle, de cause inconnue, avec une prévalence d'environ 50 cas par million d'habitants. L'événement pathologique primaire semble être la destruction des canaux biliaires interlobulaires, qui peut être médiée par des mécanismes immunologiques. Entre janvier 1974 et mai 1984, la Mayo Clinic a mené un essai randomisé en double aveugle sur la cirrhose biliaire primitive du foie (CBP), comparant le médicament D-pénicillamine (DPCA) à un placebo. Quatre cent vingt-quatre patients répondant aux critères d'éligibilité ont été vus à la clinique pendant la période d'inscription à l'essai. Le médecin traitant et le patient ont accepté de participer à l'essai randomisé dans 312 des 424 cas. La date de la randomisation et un grand nombre de paramètres cliniques, biochimiques, sérologiques et histologiques ont été enregistrés pour chacun des 312 patients de l'essai clinique. Les données de l'essai ont été analysées en 1986 pour être présentées dans la littérature clinique. Pour cette analyse, l'état de la maladie et de la survie en juillet 1986 a été enregistré pour le plus grand nombre possible de patients. À cette date, 125 des 312 patients étaient décédés, dont 11 seulement n'étaient pas attribuables à la CBP. Huit patients avaient été perdus de vue et 19 avaient subi une transplantation hépatique.

7.1 Importation des données et suppression des variables avec valeurs manquantes :

```r
library(randomForestSRC)
library(survival)
data(pbc, package = "randomForestSRC")
x = pbc[1:312, ]
p = dim(x)[2]
E = c()

for (i in 1:p)
{
  if (sum(is.na(x[, i])) > 0)
  {
    E = c(E, i)
  }
}
x = x[, -E] # On a enlevé les variables avec valeurs manquantes (NA)

cat("Dimension de la base de données : \n")
```

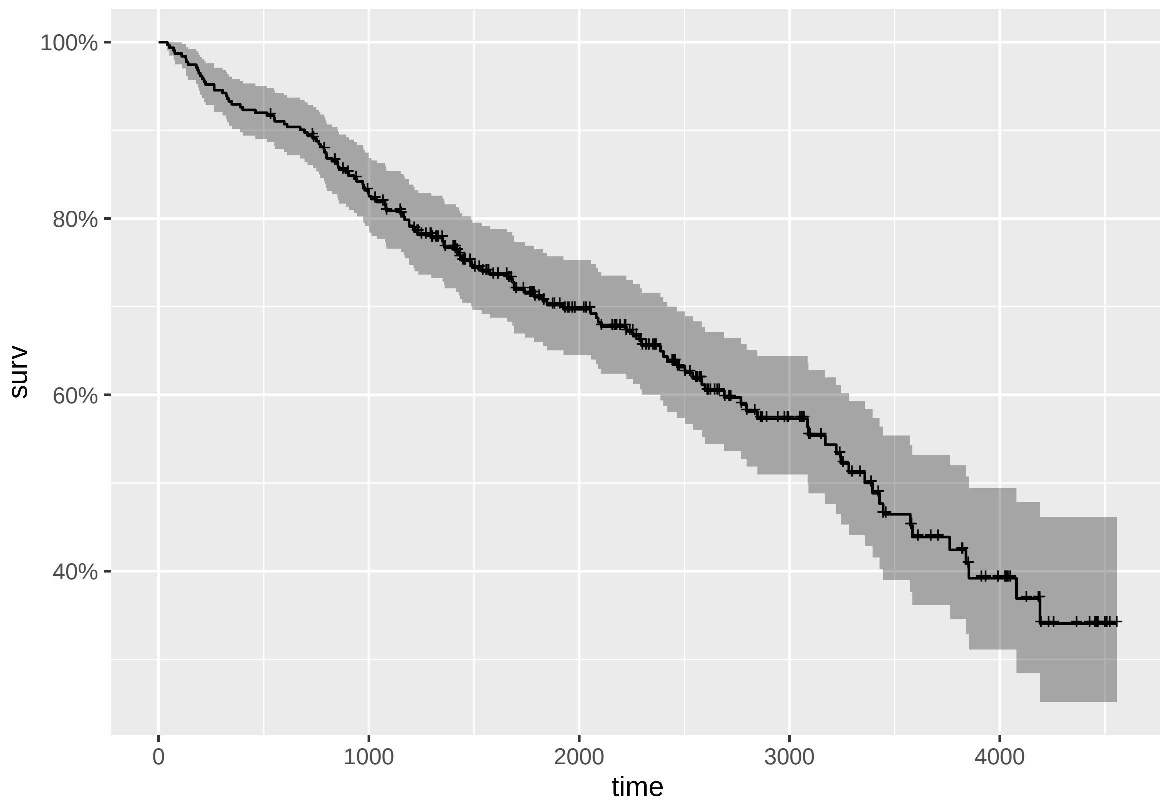
```
Dimension de la base de données :
```

```r
dim(x)
```

```
[1] 312  15
```

7.2 Estimation de Kaplan Meier simple :

```r
library(ggfortify)
s = survfit(Surv(days, status) ~ 1, data = x, type = "kaplan-meier")
autoplot(s)
```

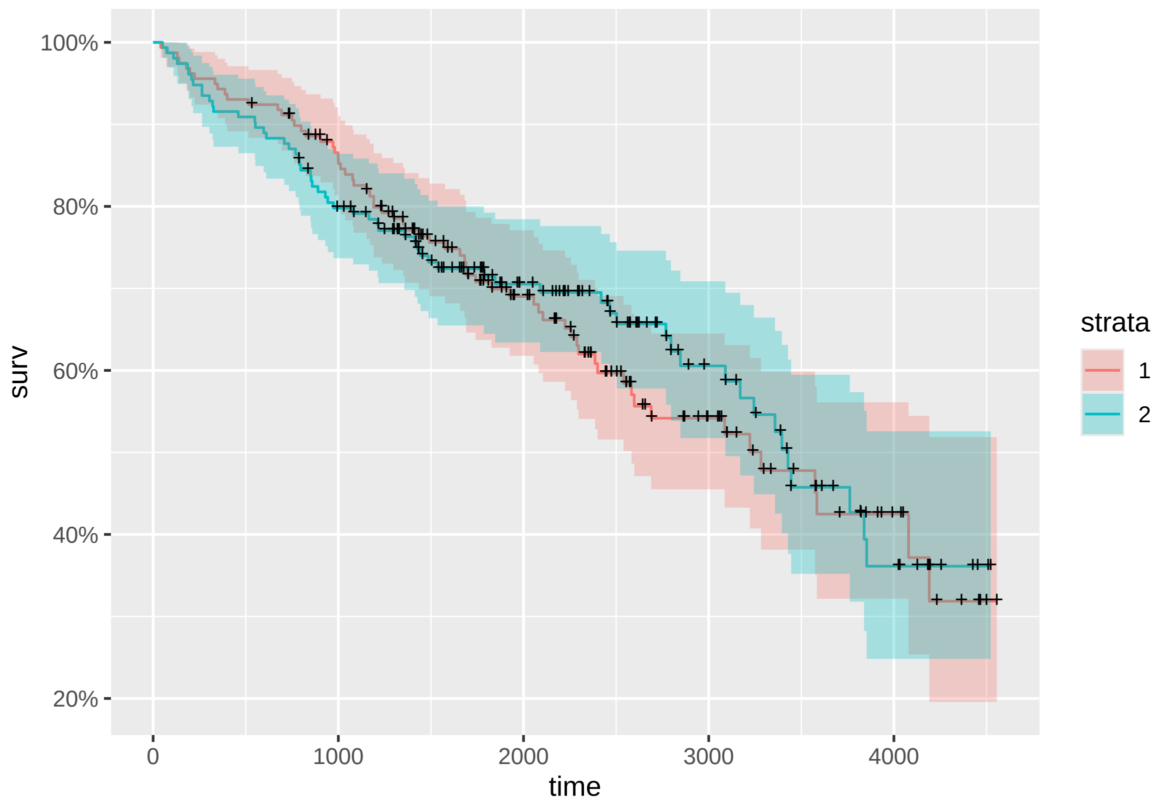


7.3 Test de comparaison de survie selon le traitement :

```
survdiff(Surv(days,status)~treatment,data=x)
```

```
Call:
survdiff(formula = Surv(days, status) ~ treatment, data = x)

              N Observed Expected (O-E)^2/E (O-E)^2/V
treatment=1 158       65     63.2    0.0502     0.102
treatment=2 154       60     61.8    0.0513     0.102

 Chisq= 0.1  on 1 degrees of freedom, p= 0.7
```

```
s = survfit(Surv(days,status)~treatment,data = x,type = "kaplan-meier")
autoplot(s)
```




```
s
```

```
Call: survfit(formula = Surv(days, status) ~ treatment, data = x, type = "kaplan-meier")

              n events median 0.95LCL 0.95UCL
treatment=1 158     65   3282    2583      NA
treatment=2 154     60   3428    3090      NA
```

Commentaire analyse (mistral ai) :

Traitement 1 : - 158 sujets ont été suivis. - 65 événements ont été observés. - Le temps médian de survie est de 3282 jours. - L'intervalle de confiance à 95% pour le temps médian de survie est [2583, NA], ce qui signifie que la limite supérieure n'est pas définie, probablement parce que plus de la moitié des sujets n'ont pas encore eu d'événement.

Commentaire (Prof) :

La p-value est élevée, la différence entre les lois de survie selon le traitemnet n'est donc pas significative. Au seuil de 5% la fonction de survie ne change pas selon le type d'évenement.

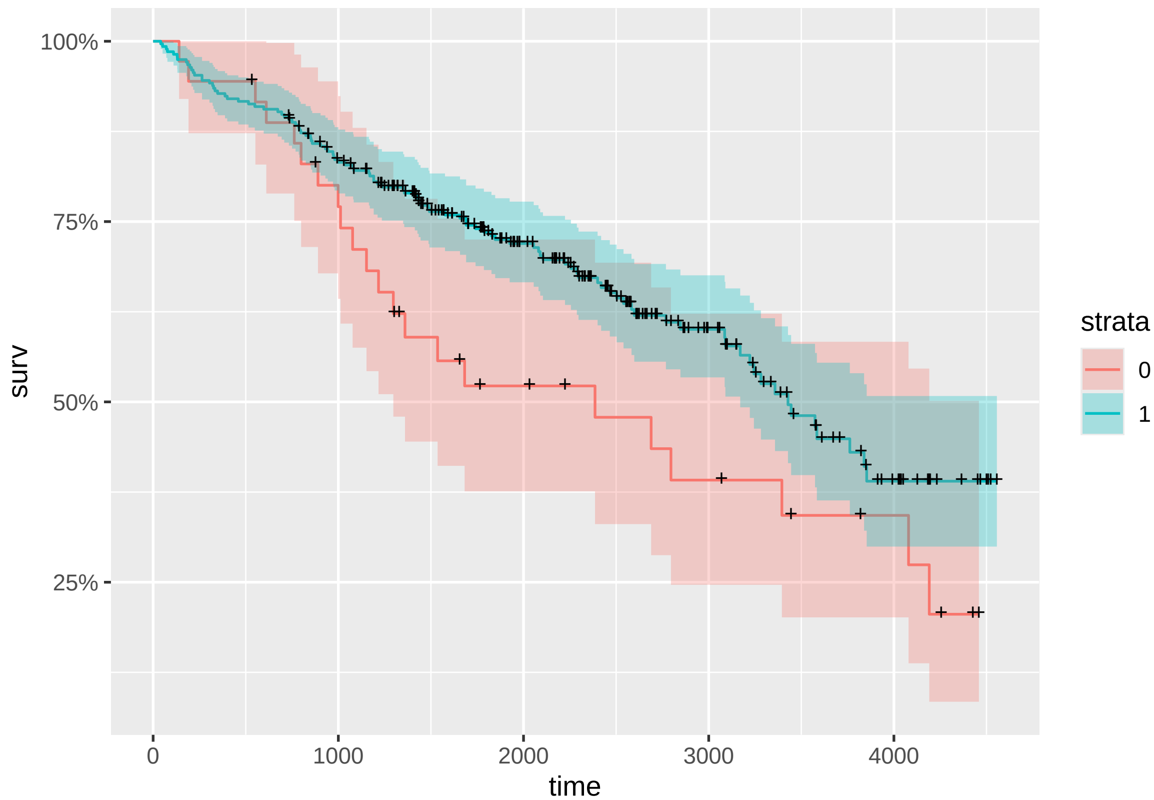
7.4 Modélisation de Kaplan-Meier en distinguant le sexe des individus :

```
survdiff(Surv(days,status)~sex,data=x)
```

```
Call:
survdiff(formula = Surv(days, status) ~ sex, data = x)

        N Observed Expected (O-E)^2/E (O-E)^2/V
sex=0  36       22     14.6     3.728      4.27
sex=1 276      103    110.4     0.494      4.27
```

```
 Chisq= 4.3  on 1 degrees of freedom, p= 0.04
```

```r
library(ggfortify)
s = survfit(Surv(days,status)~sex,data = x,type = "kaplan-meier")
autoplot(s)
```




```r
s
```

```
Call: survfit(formula = Surv(days, status) ~ sex, data = x, type = "kaplan-meier")

        n events median 0.95LCL 0.95UCL
sex=0  36     22   2386    1297      NA
sex=1 276    103   3428    3170      NA
```

Commentaire (prof) :

La p-value est de 0.04, la différence entre les lois de survie selon le sexe est donc significative au niveau de rejet 0.05.

7.5 Expliquer la durée de survie par des variables (modèle de Cox) :

Remarque :

ties=c("efron","breslow","exact") permet de choisir la méthode à adopter en cas d'événements simultanés Par défaut, c'est ici l'approximation d'Efron qui est utilisée.

7.5.1 Calibration du modèle :

```r
cox0= coxph(formula=Surv(days,status)~.,data=x)
summary(cox0)
```

```
Call:
coxph(formula = Surv(days, status) ~ ., data = x)

  n= 312, number of events= 125

                  coef  exp(coef)   se(coef)      z Pr(>|z|)
treatment    2.090e-03  1.002e+00  1.863e-01  0.011 0.991051
age          9.010e-05  1.000e+00  2.794e-05  3.225 0.001260 **
sex         -3.989e-01  6.711e-01  2.661e-01 -1.499 0.133883
ascites      3.312e-01  1.393e+00  3.105e-01  1.067 0.286086
hepatom      2.807e-01  1.324e+00  2.320e-01  1.210 0.226319
spiders      1.400e-01  1.150e+00  2.192e-01  0.639 0.523007
edema        8.441e-01  2.326e+00  3.114e-01  2.711 0.006717 **
bili         8.626e-02  1.090e+00  1.911e-02  4.514 6.35e-06 ***
albumin     -7.193e-01  4.871e-01  2.764e-01 -2.602 0.009268 **
alk          1.194e-05  1.000e+00  3.546e-05  0.337 0.736301
sgot         5.334e-03  1.005e+00  1.584e-03  3.369 0.000755 ***
prothrombin  2.564e-01  1.292e+00  9.211e-02  2.784 0.005366 **
stage        3.480e-01  1.416e+00  1.502e-01  2.317 0.020485 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

            exp(coef) exp(-coef) lower .95 upper .95
treatment      1.0021     0.9979    0.6955    1.4438
age            1.0001     0.9999    1.0000    1.0001
sex            0.6711     1.4902    0.3983    1.1305
ascites        1.3927     0.7181    0.7578    2.5594
hepatom        1.3241     0.7552    0.8403    2.0864
spiders        1.1503     0.8693    0.7485    1.7677
edema          2.3260     0.4299    1.2633    4.2825
bili           1.0901     0.9174    1.0500    1.1317
albumin        0.4871     2.0530    0.2833    0.8374
alk            1.0000     1.0000    0.9999    1.0001
sgot           1.0053     0.9947    1.0022    1.0085
prothrombin    1.2923     0.7738    1.0789    1.5480
stage          1.4162     0.7061    1.0551    1.9007

Concordance= 0.851  (se = 0.017 )
Likelihood ratio test= 193  on 13 df,   p=<2e-16
Wald test            = 198.6  on 13 df,   p=<2e-16
Score (logrank) test = 319.9  on 13 df,   p=<2e-16
```

Commentaire :

Dans le summary, les hypothèses $H_0 : \beta_j = 0$ sont testées. Les quantités $Pr(>|z|)$ sont $P(|U| > z)$, avec U de loi normale \$ N(0,1)\$. La quantité $se(coef)$ est l'écart-type de l'estimateur de β_j.

7.5.2 Probabilité de survie au moins 400 jours pour individu 1 :

```r
tcox0 = survfit(cox0, newdata = x[1, ])
i = which(tcox0$time == 400)
```

```r
tcox0$surv[i]
```

```
[1] 0.2137505
```

7.6 Simplification du modèle avec 7 variables explicatives :

7.6.1 Calibration modèle :

On peut calibrer des modèles plus simples en retirant les variables les moins significatives.

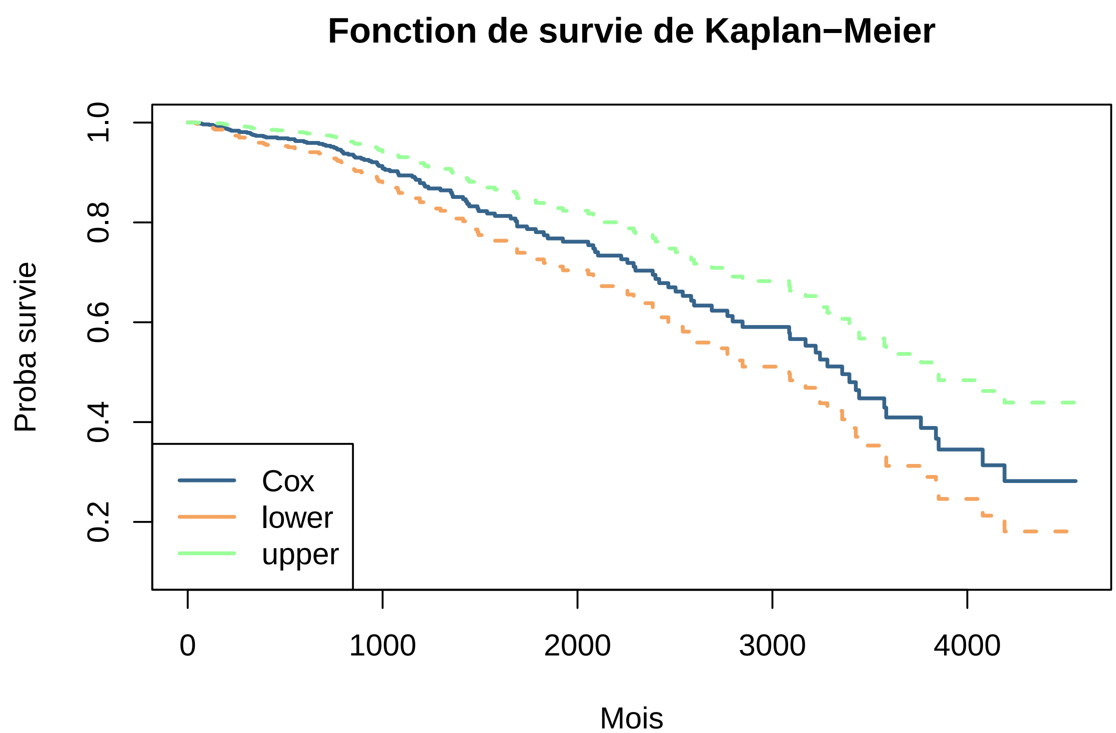
```r
cox1 = coxph(
  formula = Surv(days, status) ~ age + edema + bili + albumin +
    sgot + prothrombin + stage,
  data = x
)
summary(cox1)
```

```
Call:
coxph(formula = Surv(days, status) ~ age + edema + bili + albumin +
    sgot + prothrombin + stage, data = x)

  n= 312, number of events= 125

                  coef  exp(coef)   se(coef)      z Pr(>|z|)
age          0.0001051  1.0001051  0.0000262  4.011 6.05e-05 ***
edema        0.8620524  2.3680159  0.3056603  2.820 0.004798 **
bili         0.0926756  1.0971058  0.0176114  5.262 1.42e-07 ***
albumin     -0.8201687  0.4403574  0.2502915 -3.277 0.001050 **
sgot         0.0056309  1.0056468  0.0015553  3.621 0.000294 ***
prothrombin  0.2680882  1.3074625  0.0878696  3.051 0.002281 **
stage        0.4623129  1.5877420  0.1327310  3.483 0.000496 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

            exp(coef) exp(-coef) lower .95 upper .95
age            1.0001     0.9999    1.0001    1.0002
edema          2.3680     0.4223    1.3008    4.3109
bili           1.0971     0.9115    1.0599    1.1356
albumin        0.4404     2.2709    0.2696    0.7192
sgot           1.0056     0.9944    1.0026    1.0087
prothrombin    1.3075     0.7648    1.1006    1.5532
stage          1.5877     0.6298    1.2240    2.0595

Concordance= 0.842  (se = 0.018 )
Likelihood ratio test= 186.8  on 7 df,   p=<2e-16
Wald test            = 189.3  on 7 df,   p=<2e-16
Score (logrank) test = 291.1  on 7 df,   p=<2e-16
```

7.6.1.1 Graphique fonction de survie : Nous pouvons tracer le graphe de la fonction de survie (Kaplan Meier ou Aalen, c'est Aalen par défaut). Les covariables sont fixées à leur valeur moyenne.

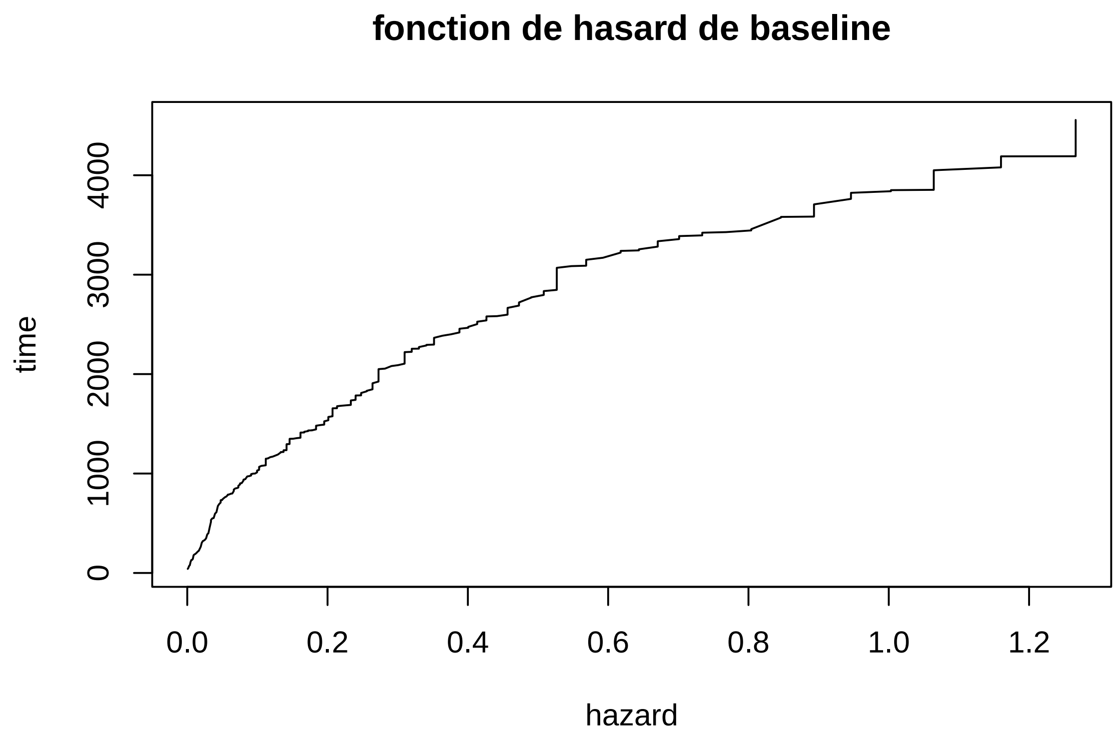
```r
#summary(survfit(cox1)) # Affichage des probabilités de survie
plot(
  survfit(cox1),
  ylim = c(.1, 1),
```

```r
  xlab = 'Mois',
  ylab = 'Proba survie',
  main = 'Fonction de survie de Kaplan-Meier',
  col = palette_couleur[1:3],
  lwd = 2
)
legend(
  "bottomleft",
  legend = c("Cox" , "lower" , "upper"),
  lwd = 2,
  col = palette_couleur[1:3],
)
```




```r
# Nous pouvons tracer la fonction de hasard cumulée
# (estimateur de Breslow).

plot(basehaz(cox1), main = 'fonction de hasard de baseline', type = 'l')
```

7.6.1.2 Fonction de hasard :




```r
#plot(survfit(cox1,newdata=x))
```

7.6.1.3 Fonction de survie des individus observés :

7.6.2 Vérification de l'hypothèse de proportionnalité du modèle :

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

```
             chisq df      p
age          1.365  1 0.2427
edema        3.576  1 0.0586
bili         8.696  1 0.0032
albumin      0.431  1 0.5113
sgot         3.631  1 0.0567
prothrombin  5.136  1 0.0234
stage        4.571  1 0.0325
GLOBAL      20.497  7 0.0046
```

```r
#par(mfrow=c(3,3))
#plot(res)
```

Interprétation :

- Les résidus de la variables bili ne sont pas indépendants du temps.
- Il faut relancer le modèle en stratifiant sur la variable bili.

7.6.3 Probabilité de survie 400 jours premier individus de la base :

```
# Ou la fonction de survie pour des individus ayant les var
# explicatives identiques à l'individu 1 par exemple.
#plot(survfit(cox1,newdata=x[1,]))
tcox1=survfit(cox1,newdata=x[1,])
i=which(tcox1$time==400)
#i
tcox1$surv[i]
```

```
[1] 0.2496241
```

7.7 Modélisation Forêt aléatoire de survie :

7.7.1 Calibration du modèle

```
library(randomForestSRC)
v.obj <- rfsrc(Surv(days,status)~., data = x,
                   ntree = 100)

## plot tree number 3
plot(get.tree(v.obj, 3))
```

7.7.2 Récupération du C-index :

```
get.cindex(time=x$days,censoring=x$status,predicted=v.obj$predicted.oob)
```

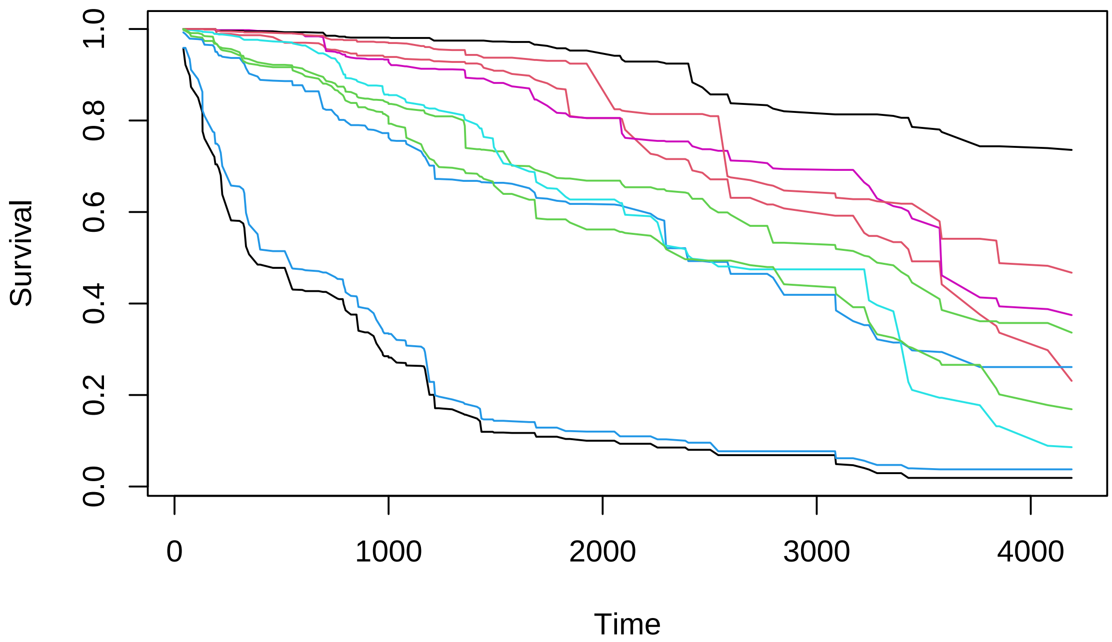
```
[1] 0.17094
```

Interprétation du C-index :

- Le C-index est de 0.16 ainsi le modèle de forêt aléatoire prédit très mal la mortalité.

```
#Le graphe de la fonction de survie pour les 10 premiers individus :

matplot(v.obj$time.interest,  t(v.obj$survival.oob[1:10, ]),
    xlab = "Time", ylab = "Survival", type = "l", lty = 1)
```

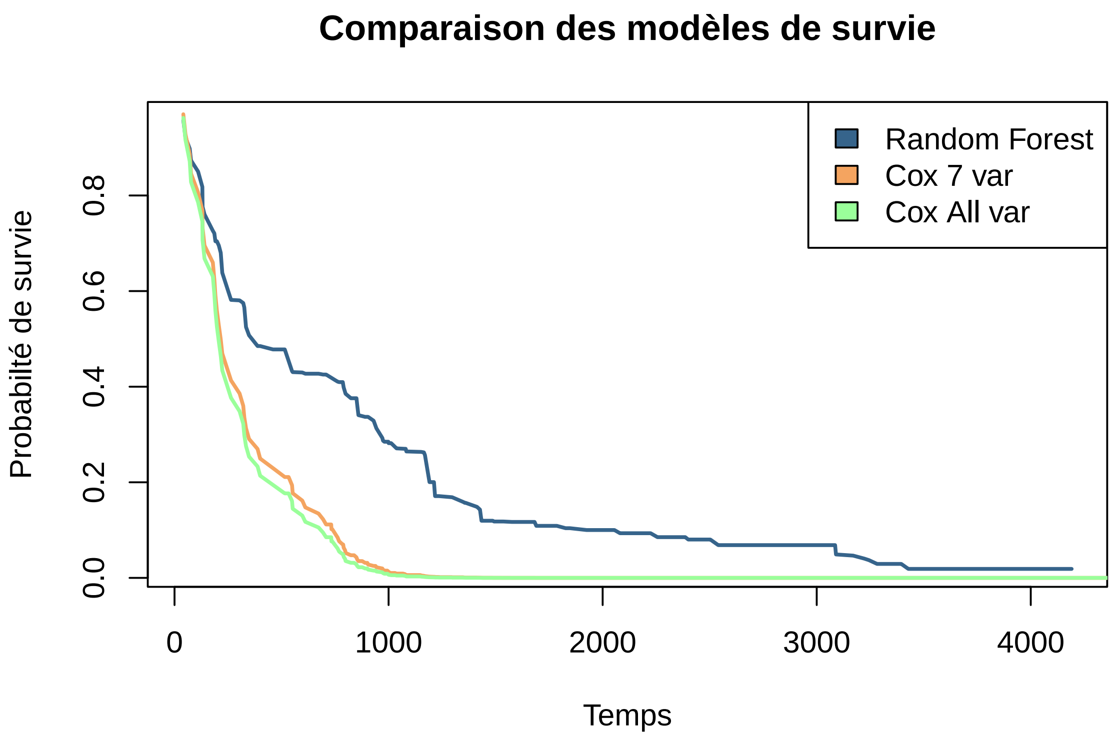
7.7.2.1 Graphique de la fonction de survie :



7.7.3 Comparaison des modèles Cox et Forêt aléatoire pour l'individu 1 :

Comparaison des fonctions de survie entre une forêt aléatoire de survie et le modèle de Cox, pour l'individu 1:

```
plot(
  v.obj$time.interest,
  t(v.obj$survival.oob[1,]),
  xlab = "Temps",
  ylab = "Probabilté de survie",
  main = "Comparaison des modèles de survie",
  type = "l",
  lty = 1,
  col = palette_couleur[1],
  lwd = 2
)
lines(tcox1$time, tcox1$surv, col = palette_couleur[2], lwd = 2)
lines(tcox0$time, tcox0$surv, col = palette_couleur[3], lwd = 2)
legend(
  "topright",
  legend = c("Random Forest", "Cox 7 var", "Cox All var"),
  fill = palette_couleur[1:3]
)
```




```r
# Prédiction de la mortalité au 400e jour pour l'individu 1 :
i=which(v.obj$time.interest==400)
i
```

```
[1] 23
```

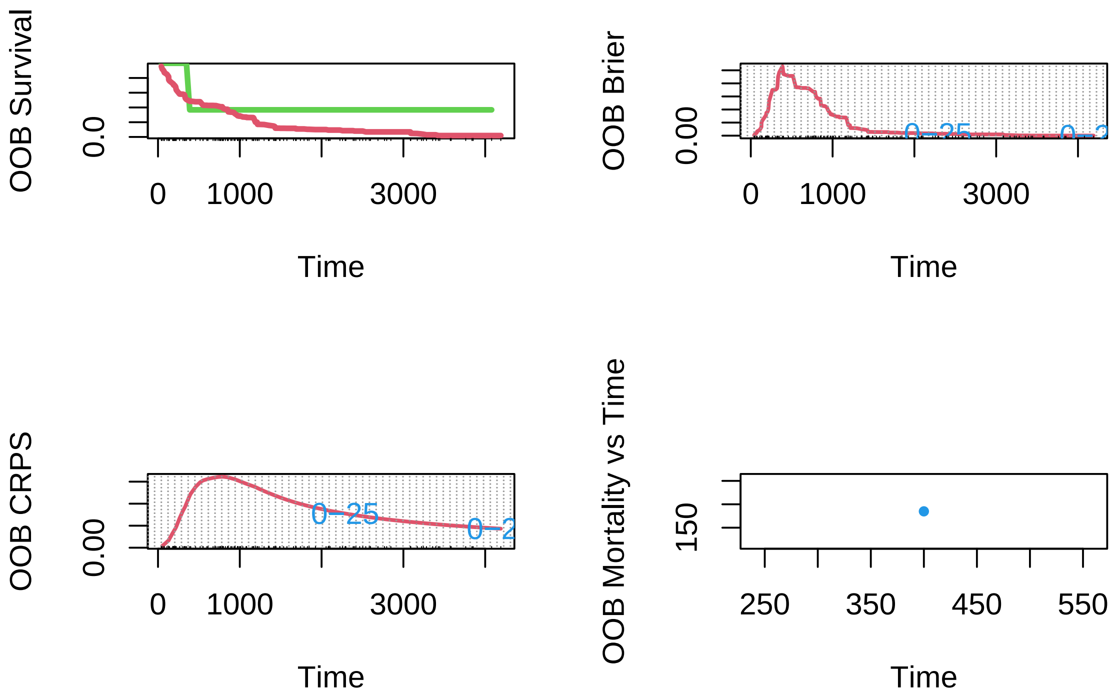
```r
v.obj$survival.oob[1,i ]
```

```
[1] 0.4849954
```

7.7.4 Etude approfondie du modèle de forêt aléatoire :

La fonction plot.survival permet d'avoir une synthèse graphique de résultats :

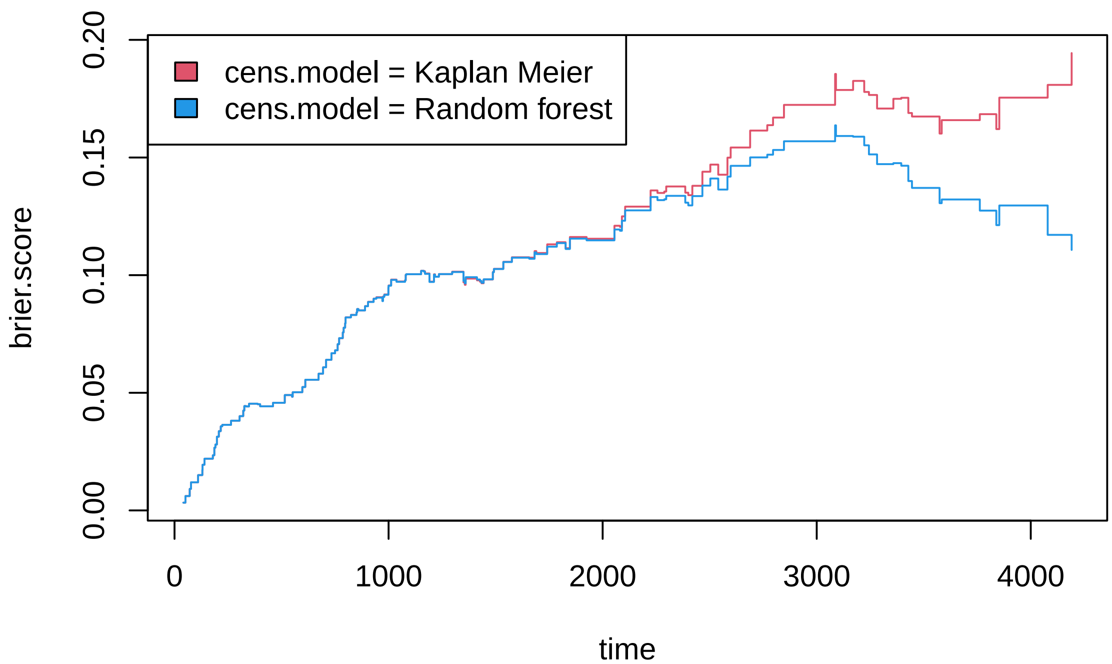
```r
plot.survival(v.obj, subset = 1)
```



7.7.4.1 Performances du modèle RSF : Commentaire et explication :

- Le modèle de forêt aléatoire de survie est plus performant que le modèle de Cox simple (7 variables explicatives) et le modèle de Cox complet (toutes les variables explicatives).

```r
## obtain Brier score using KM and RSF censoring distribution estimators
bs.km <- get.brier.survival(v.obj, cens.model = "km")$brier.score
bs.rsf <- get.brier.survival(v.obj, cens.model = "rfsrc")$brier.score

## plot the brier score : Evolution de l'erreur de prédiction
plot(bs.km, type = "s", col = 2)
lines(bs.rsf, type ="s", col = 4)
legend("topleft", legend = c("cens.model = Kaplan Meier",
                             "cens.model = Random forest"), fill = c(2,4))
```



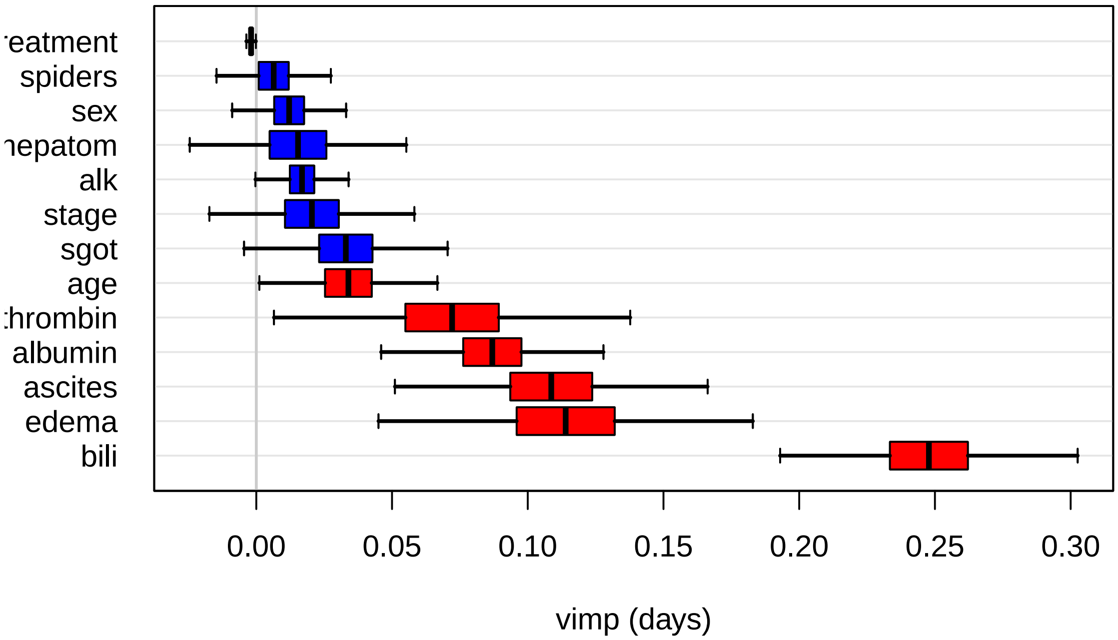
7.7.5 Prise en compte de l'importance des variables :

Importance des variables (VIMP) :

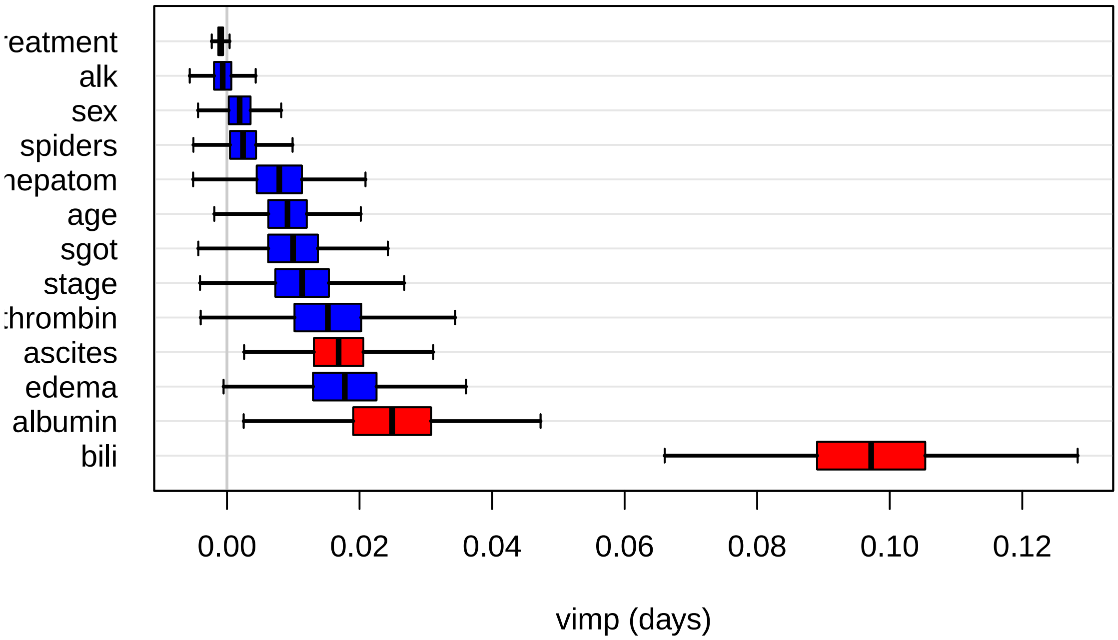
Plusieurs méthodes sont possibles pour l'importance d'une variable x :

- importance = "permute" : calcul d'importance par permutation aléatoire des valeurs de x observées sur les exemples OOB.
- importance = "random" : calcul d'importance par choix aléatoire gauche/droite lorsqu'une coupure se fait avec la variable x.
- importance = "anti" : calcul d'importance en choisissant le choix opposé à celui proposé.

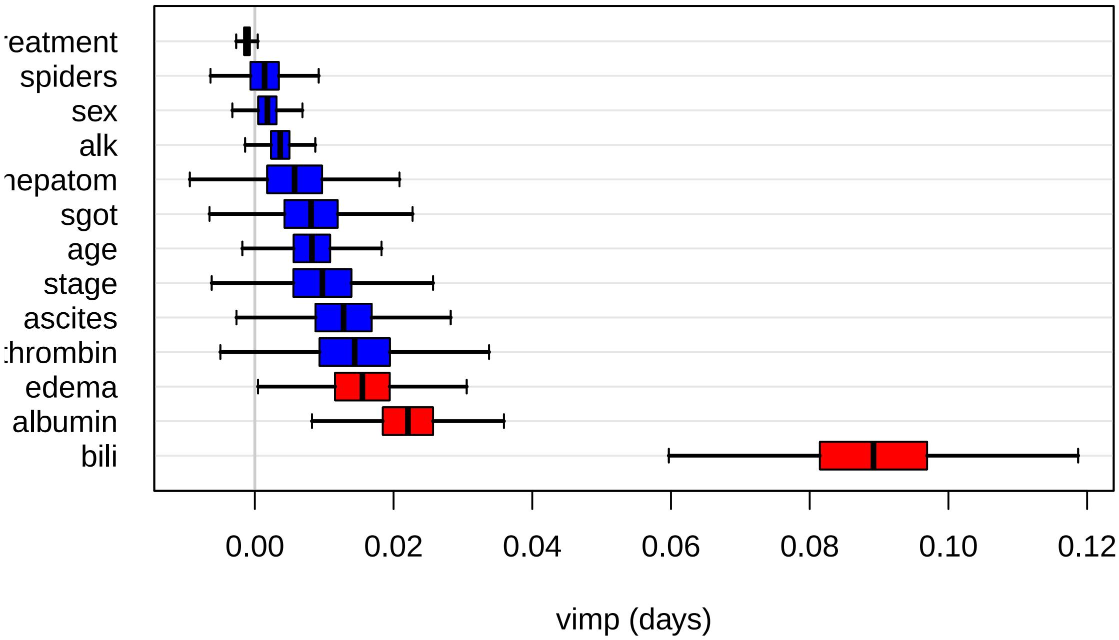
```
imp1 = subsample(v.obj, importance = "anti")
plot(imp1)
```




```r
imp2 = subsample(v.obj, importance = "permute")
plot(imp2)
```




```r
imp3 = subsample(v.obj, importance = "random")
plot(imp3)
```




```r
library(randomForestSRC)
v.obj <- rfsrc(
  Surv(days, status) ~ age + edema + bili + albumin +
    sgot + prothrombin + stage,
  data = x,
  ntree = 100
)

## plot tree number 3
plot(get.tree(v.obj, 3))
```

```r
get.cindex(
  time = x$days,
  censoring = x$status,
  predicted = v.obj$predicted.oob
)
```

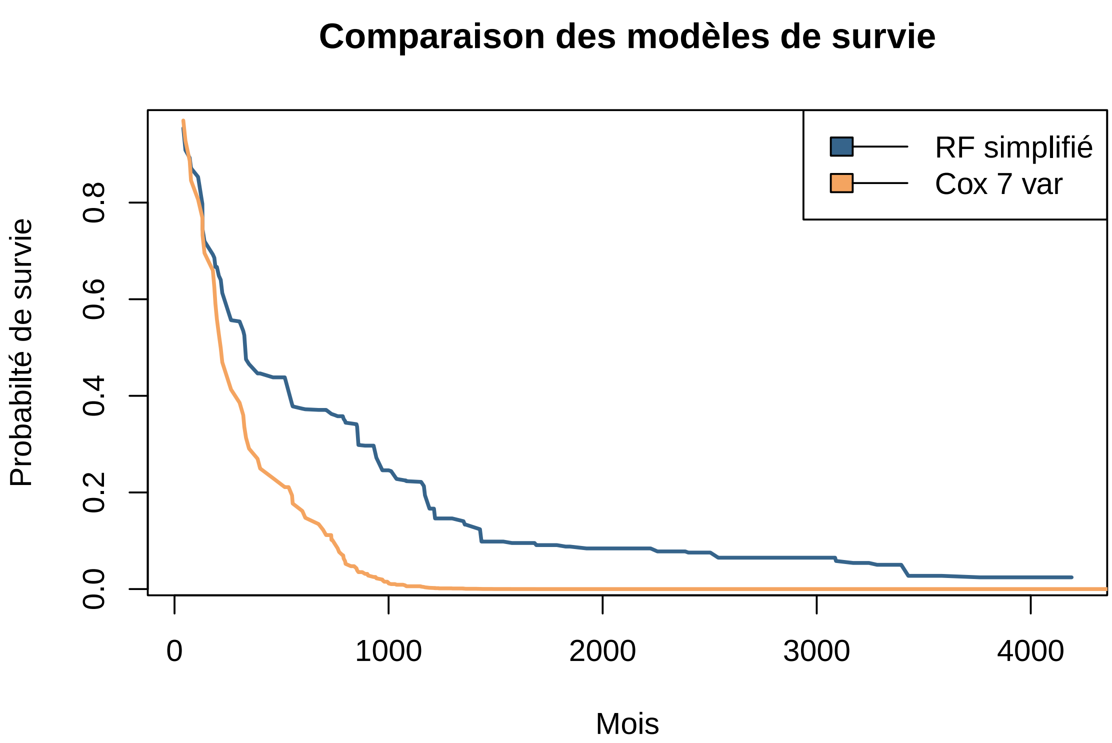
7.7.5.1 Estimation sur les variables ayant le plus d'importance :

```
[1] 0.16726
```

```r
plot(
  v.obj$time.interest,
  t(v.obj$survival.oob[1,]),
  xlab = "Mois",
```

```
  ylab = "Probabilté de survie",
  type = "l",
  main = "Comparaison des modèles de survie",
  lty = 1,
  col = palette_couleur[1],
  lwd = 2
)
lines(tcox1$time, tcox1$surv, col = palette_couleur[2], lwd = 2)
legend("topright",
       legend = c("RF simplifié", "Cox 7 var"),
       fill = palette_couleur[1:2], lwd = 1)
```

7.7.5.2 Affichage grahique de la fonction de survie sur le modèle simplifié :




```
# Probabilité de survie au 400e jour pour l'individu 1 :
i=which(v.obj$time.interest==400)
#i
v.obj$survival.oob[1,i ]
```

```
[1] 0.4463523
```

8 Autres exercices :

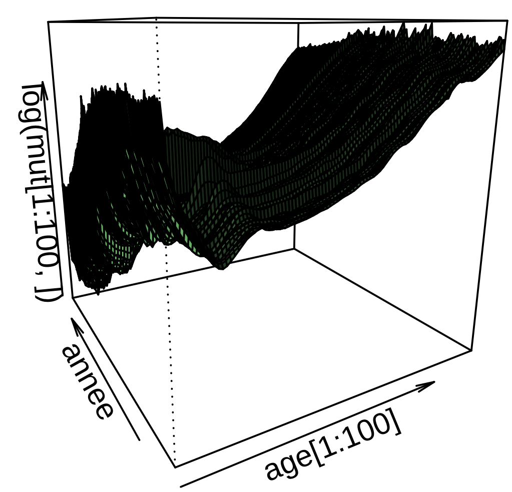
8.1 Exercice sur la modélisation de Lee-Carter

8.1.1 Importation des données :

Importation des données de mortalité pour la France de 1816 à 2021.

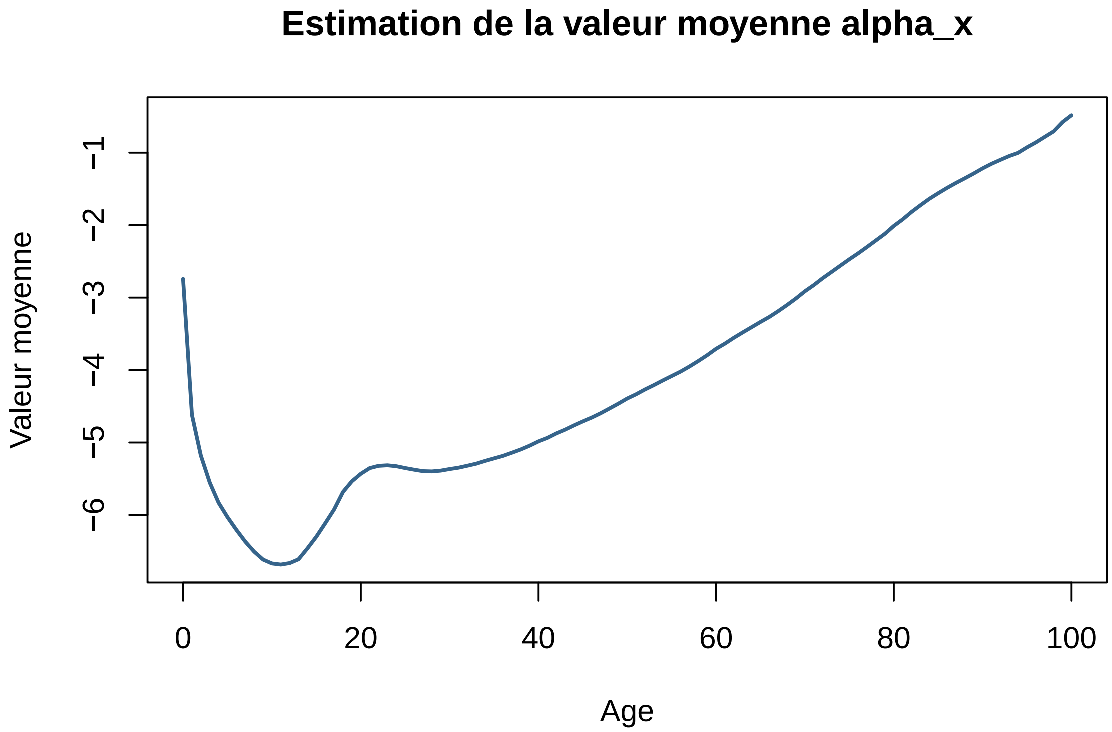
Référence du site HMD.

```
# Décès
de = read.csv("DATA/DeathsFrance2024.csv", header = TRUE, sep = ";")
#str(de)
# remarque : la classe d'âge "110" est en réalité "110 et plus".

# Expositions
ex = read.csv("DATA/ExposuresFrance2024.csv", header = TRUE, sep = ";")
#str(ex)

# Force de mortalité : \mu_{x,t}= m_{x,t} (m_{x,t} taux de mortalité)
age = 0:110
annee = 1816:2021
mu = de[, 3:5] / ex[, 3:5]
mut = matrix(mu[, 3], length(age), length(annee))
persp(
  age[1:100],
  annee,
  log(mut[1:100, ]),
  theta = -30,
  col = "light green",
  shade = TRUE
)
```

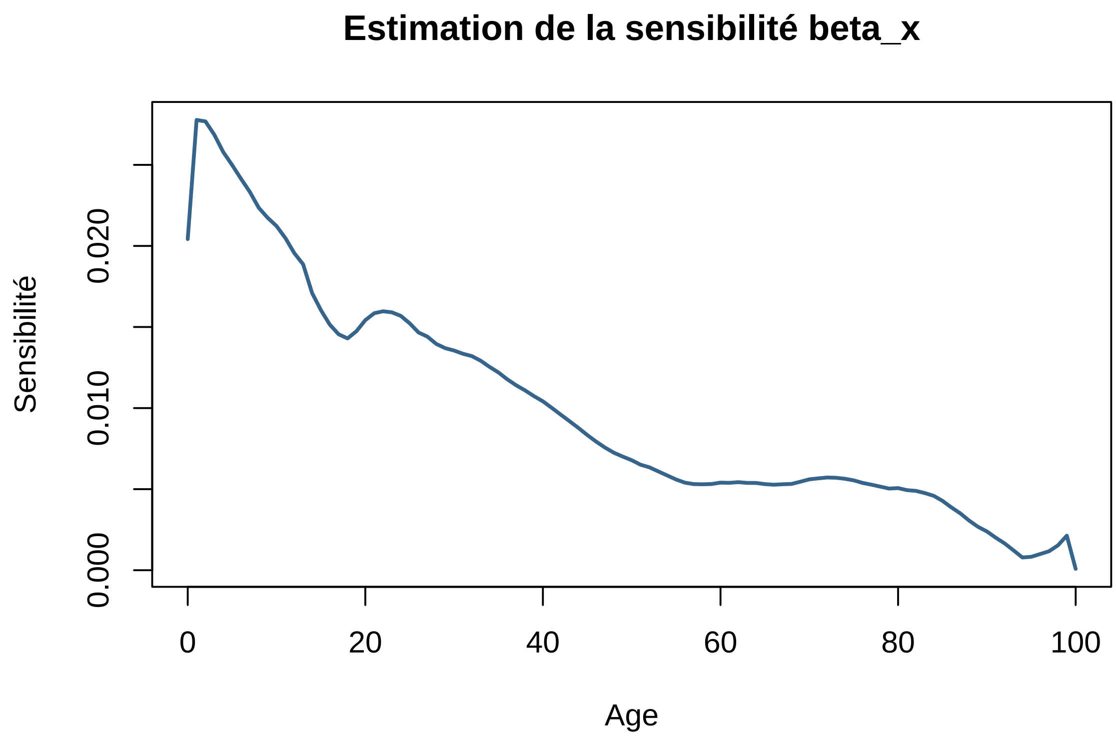


8.1.2 Modélisation de Lee-Carter :

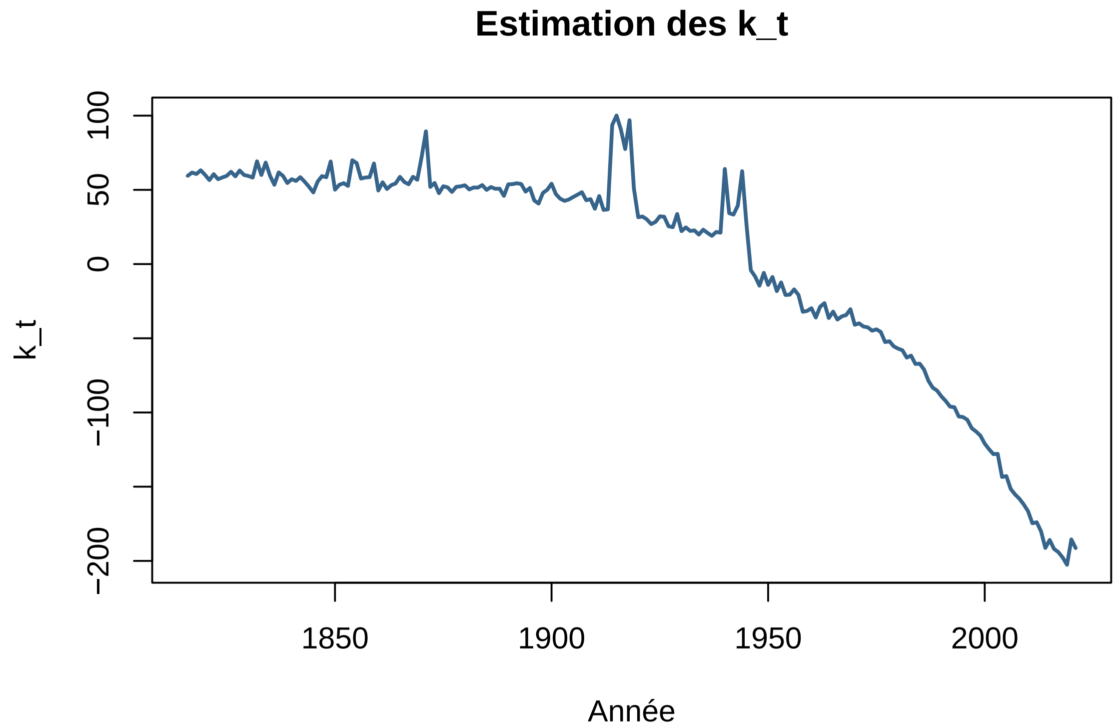
```
library(forecast)
library(demography)

# Calibrage Lee-Carter avec l'ensemble de données
annee = unique(de$Year)
nc = length(annee)
age = unique(de$Age)
nl = length(age)

muf = matrix(de$Female / ex$Female, nl, nc)  # Données Femmes
muh = matrix(de$Male / ex$Male, nl, nc)  # Données Hommes

popf = matrix(ex$Female, nl, nc)
poph = matrix(ex$Male, nl, nc)

Baseh = demogdata(
  data = muh,
  pop = poph,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Hommes',
  lambda = 1
)

Basef = demogdata(
  data = muf,
  pop = popf,
  ages = age,
  years = annee,
  type = "mortality",
  label = 'France',
  name = 'Femmes',
  lambda = 1
)

lch = lca(Baseh)

# Estimation de alpha_x
plot(
  lch$age,
  lch$ax,
  main = "Estimation de la valeur moyenne alpha_x",
  col = palette_couleur[1],
  xlab = "Age",
  ylab = "Valeur moyenne",
  type = 'l',
  lwd = 2
)
```




```r
# Estimation de beta_x
plot(
  lch$age,
  lch$bx,
  main = "Estimation de la sensibilité beta_x",
  col = palette_couleur[1],
  xlab = "Age",
  ylab = "Sensibilité",
  type = 'l',
  lwd = 2
)
```




```r
# Estimation des k_t
kt = lch$kt
plot(
  annee,
  kt,
  main = "Estimation des k_t",
  col = palette_couleur[1],
  xlab = "Année",
  ylab = "k_t",
  type = 'l',
  lwd = 2
)
```



8.1.3 Projection des k_t méthode de Lee & Carter (1992) :

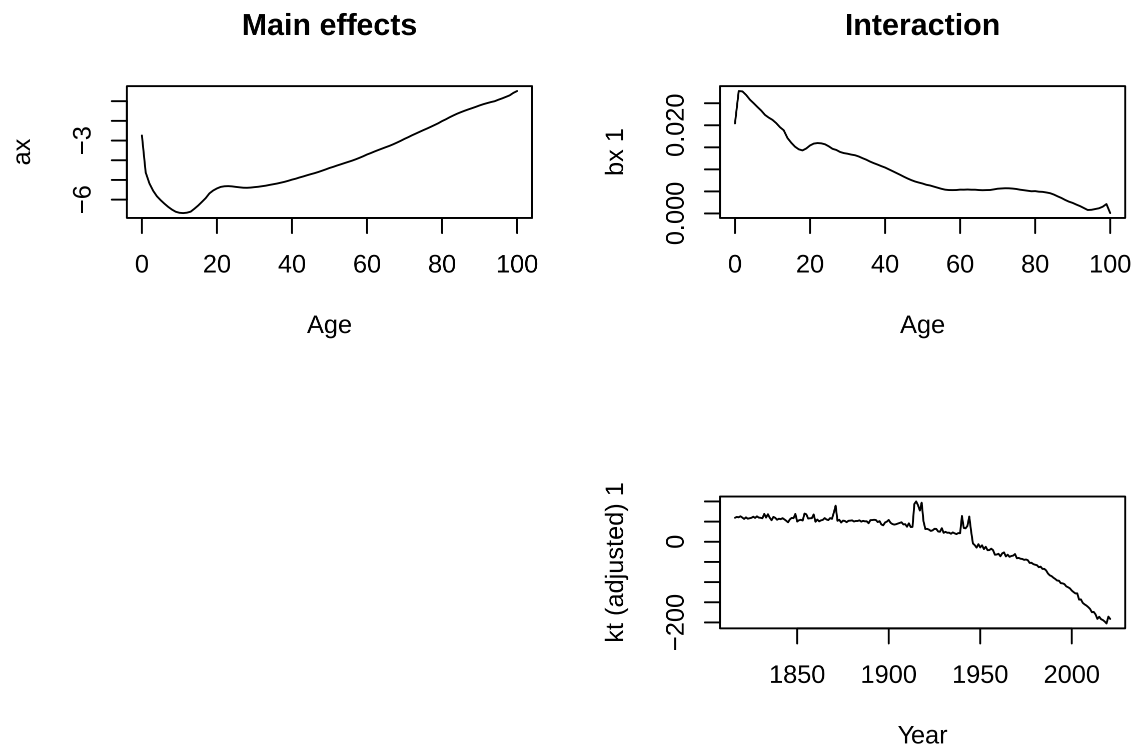
On fait l'hypothèse que les k_t sont déterminés par la relation suivante :

$$k_t = k_{t-1} + d + \epsilon_t$$

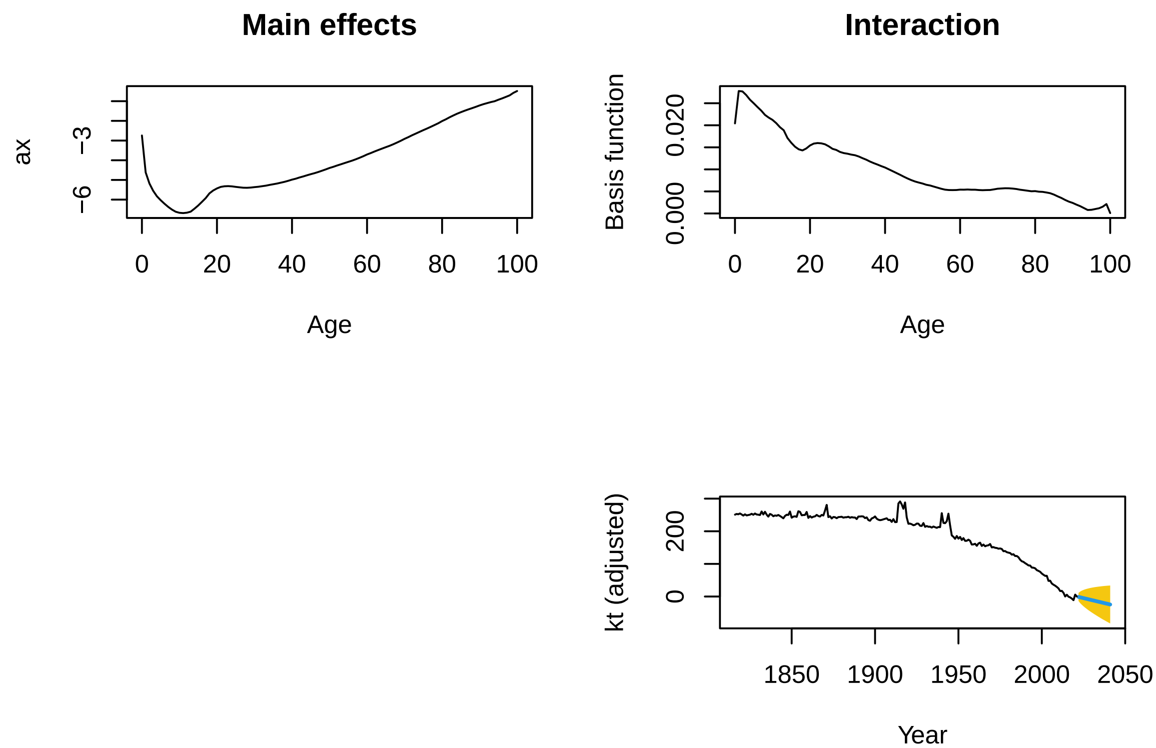
avec :

- k_t : les valeurs de k à l'instant t.
- k_{t-1} : les valeurs de k à l'instant $t-1$.
- d : une constante.
- $epsilon_t$: un bruit blanc.

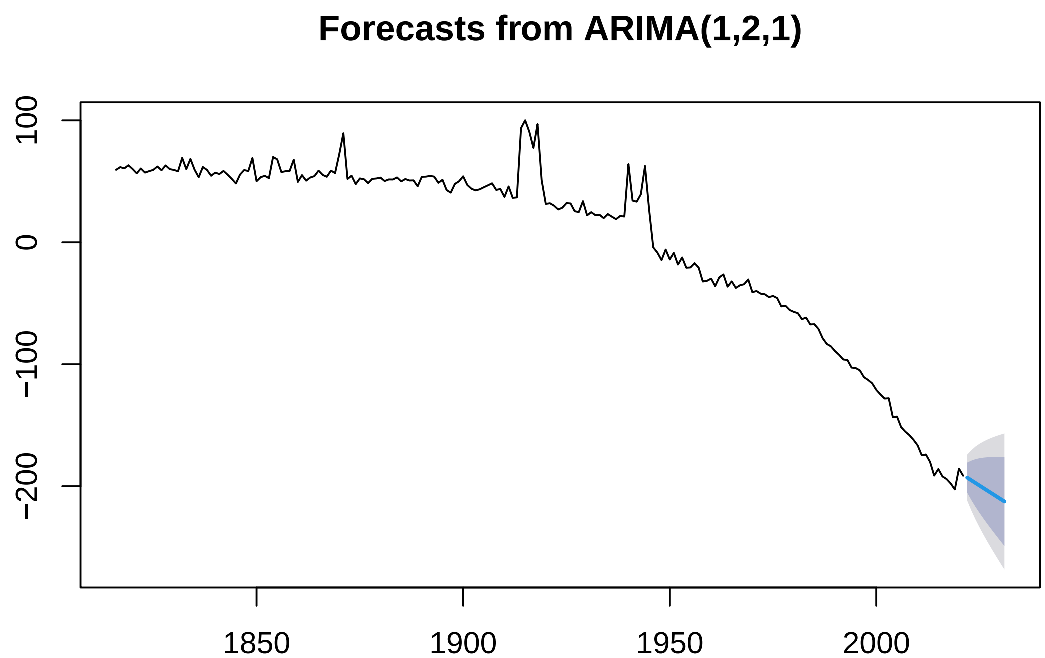
```
plot(lch) # Affichage des résultats
```




```r
# Projection des k_t à l'aide du modèle initial :
proj = forecast(lch, h = 20)
plot(proj, plot.type = "component")
```



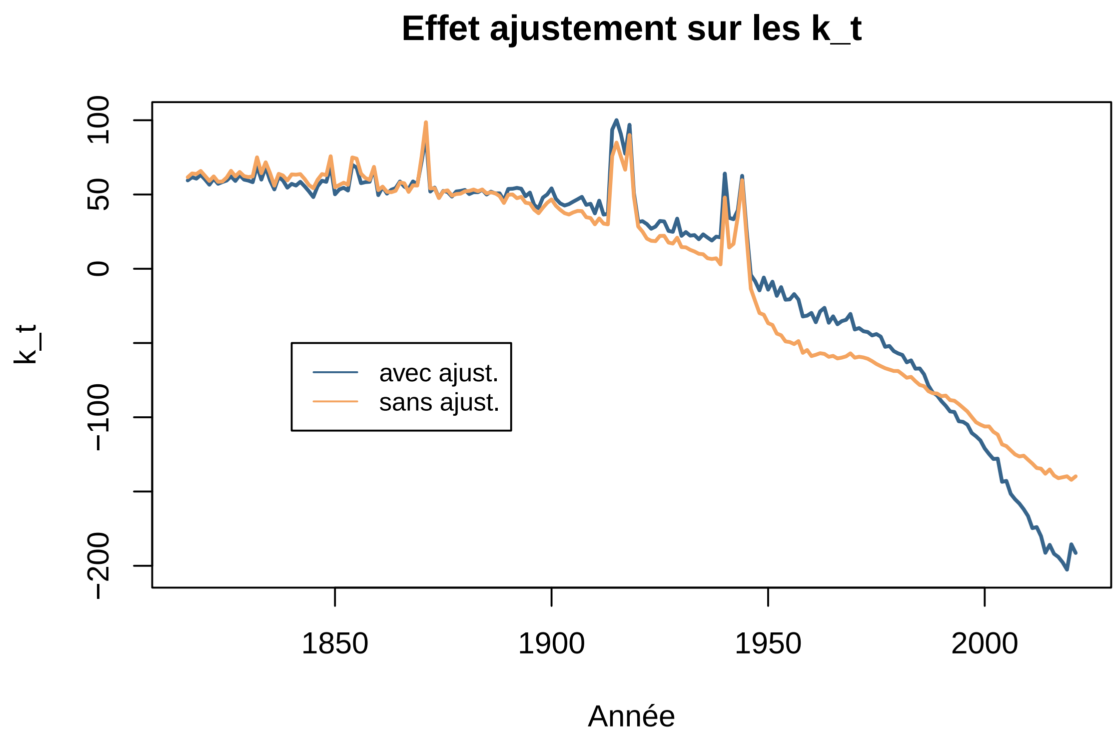

```r
par(mfrow = c(1, 1))

# Ou bien un auto-arima pour modéliser et projeter les k_t :
ar = auto.arima(kt)
plot(forecast(ar, h = 10))
```




```r
# Modèle sans ajustement des Kt :
lch_sans = lca(Baseh, adjust = "none")
```

```r
plot(
  lch$year,
  lch$kt,
  col = palette_couleur[1],
  type = 'l',
  main = "Effet ajustement sur les k_t",
  xlab = "Année",
  ylab = "k_t",
  lwd = 2
)
lines(lch_sans$year, lch_sans$kt, col = palette_couleur[2], lwd = 2)
legend(
  1840,-50,
  legend = c("avec ajust.", "sans ajust."),
  col = palette_couleur[1:2],
  lty = 1,
  cex = 0.8
)
```

8.1.3.1 Comparaison avec et sans ajustement des k_t :



8.2 Exercice 2 : La modélisation de log-Poisson

Human Mortality Database HMD

8.2.1 Importation des données :

```
# Décès
de = read.csv("DATA/DeathsFrance2024.csv", header = TRUE, sep = ";")
# remarque : la classe d'âge "110" est en réalité "110 et plus".

# Expositions
ex = read.csv("DATA/ExposuresFrance2024.csv", header = TRUE, sep = ";")
#str(ex)
```

8.2.2 Modélisation log-Poisson :

On peut calibrer ce modèle avec la fonction gnm (generalized nonlinear model).

Le modèle de Lee-Carter suppose que les résidus sont homoscédastiques c'est à dire que l'aléa de nuisance $\epsilon_{x,t}$ est constant dans le temps et par âge. Le modèle de Log-Poisson permet de relacher cette hypothèse en considérant que la variance des taux de décès augmente pour les âges élevés.

On va modéliser le nombre de décès $D_{x,t}$ par une loi de Poisson conditionnelle à l'exposition $L_{x,t}$ et à un terme multiplicatif $\mu_{x,t}$ qui dépend de l'âge et de l'année. Ainsi on obtient que $\mu_{x,t} = \exp(\alpha_x + \beta_x k_t)$, tel que $D_{x,t} \sim \mathcal{P}(\mu_{x,t} \times L_{x,t})$.

On se rapproche d'un modèle de Lee-Carter pour qui la fonction de mortalité est donnée par $\mu_{x,t} = \exp(\alpha_x + \beta_x k_t)$ mais on considère une loi de Poisson pour les décès et non pas une loi normale centrée réduite.

Afin d'utiliser la fonction gnm, utilisée pour les modélisations non-linéaire généralisées on doit adapter les données.

$\mu_{x,t} = \exp(\alpha_x + \beta_x k_t) = \sum_{a=x_{\min}}^{x_{\max}} \alpha_\alpha \mathbf{1}_{[\alpha]}(x) + \sum_{a=x_{\min}}^{x_{\max}} \sum_{b=t_{\min}}^{t_{\max}} \beta_\alpha k_b \mathbf{1}_{[\alpha]}(x) \mathbf{1}_{[b]}(t)$

Pour obtenir cette équation dans R on applique des facteurs aux variables afin de récréer les indicatrices dans la fonction gnn.

```
library(gnm)

# Sélection des données :
ind = which((de$Age > 44) & (de$Age < 100) &
              (de$Year > 1949) & (de$Year < 2013))
annee = 1950:2012
nc = length(annee)
age = 45:99
nl = length(age)

D = de$Male[ind]
E = ex$Male[ind]
x = as.factor(ex$Age[ind])
t = as.factor(ex$Year[ind])

regp <-
  gnm(D ~ 0 + x + Mult(x, t),
      offset = log(E),
      family = poisson(link = "log"))
```

8.2.2.1 Modélisation et adaptation des donnéés :

```
Initialising
Running start-up iterations..
Running main iterations.....
Done
```

```
nomvar = names(regp$coefficients)

set.seed(123)
#Simulation aléatoire de 10 variables explicatives pour visualiser :
nomvar[sample(1:length(nomvar), 10)]
```

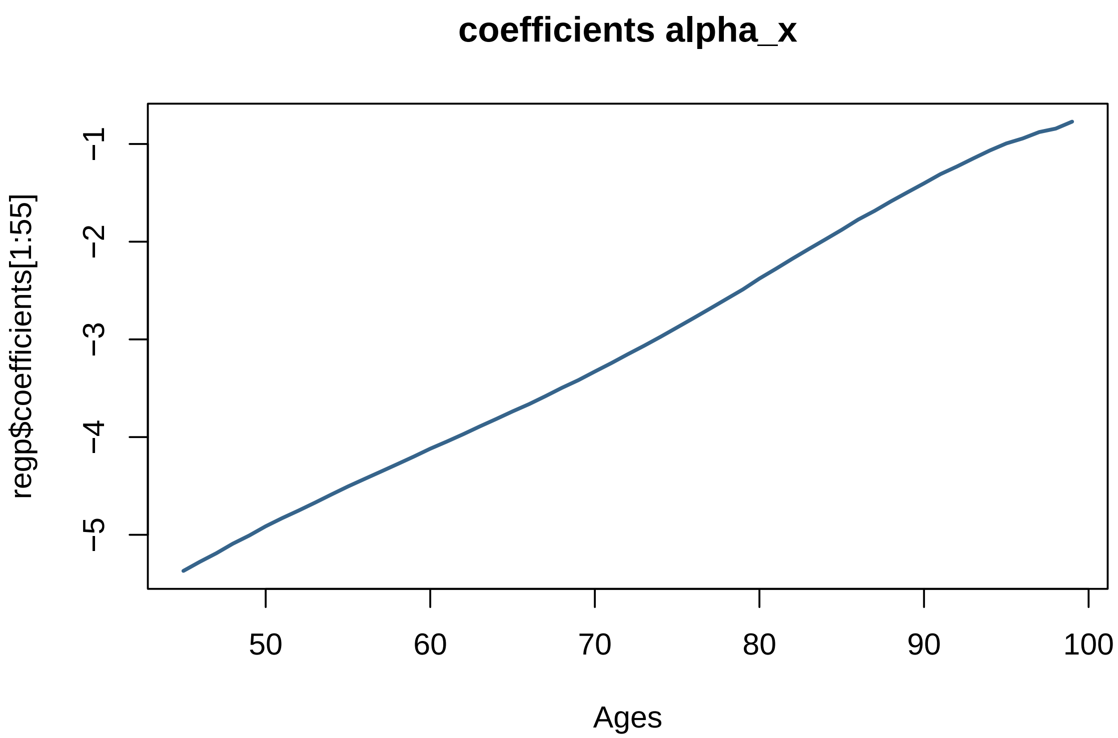
```
 [1] "Mult(x, .).t1998" "x58"              "Mult(x, .).t2009" "x94"
 [5] "Mult(x, .).t1957" "x87"              "Mult(x, .).t2011" "Mult(x, .).t2008"
 [9] "Mult(x, .).t1992" "Mult(., t).x79"
```

```
# Explication de la fonction Mult(x,t) :
# Elle permet de modéliser l'interaction entre l'âge et l'année.
```

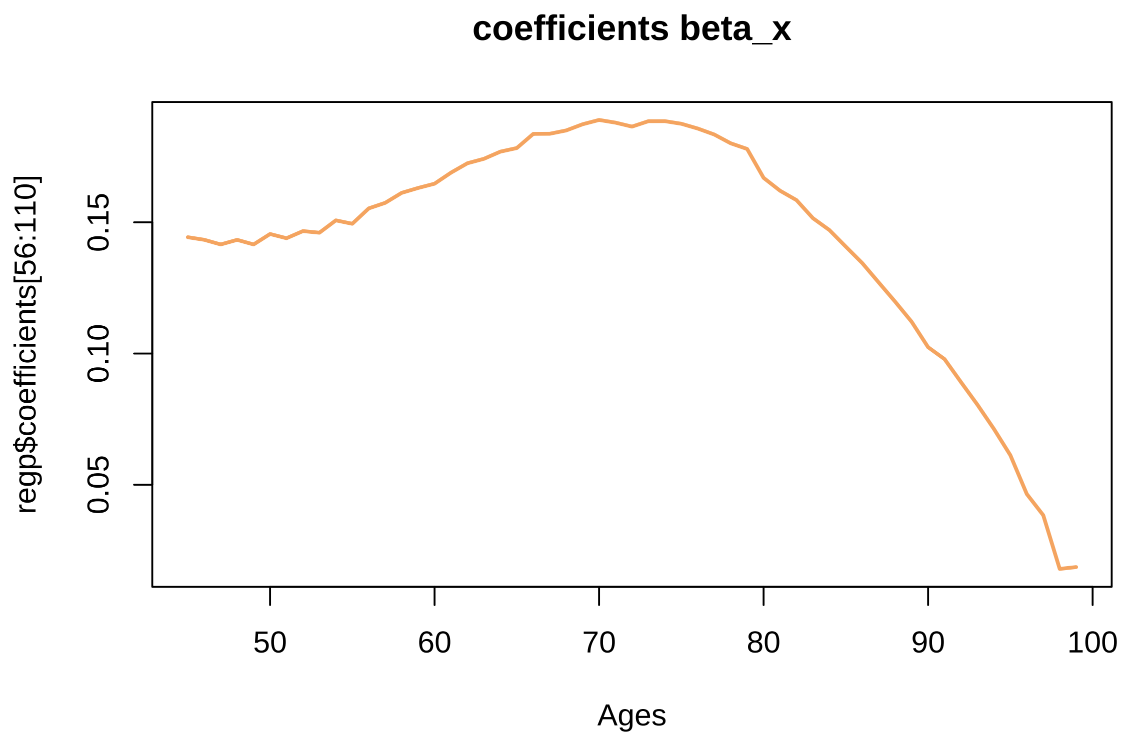
```
plot(
  45:99,
  regp$coefficients[1:55],
  main = "coefficients alpha_x",
  xlab =
    "Ages",
```

```r
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
```

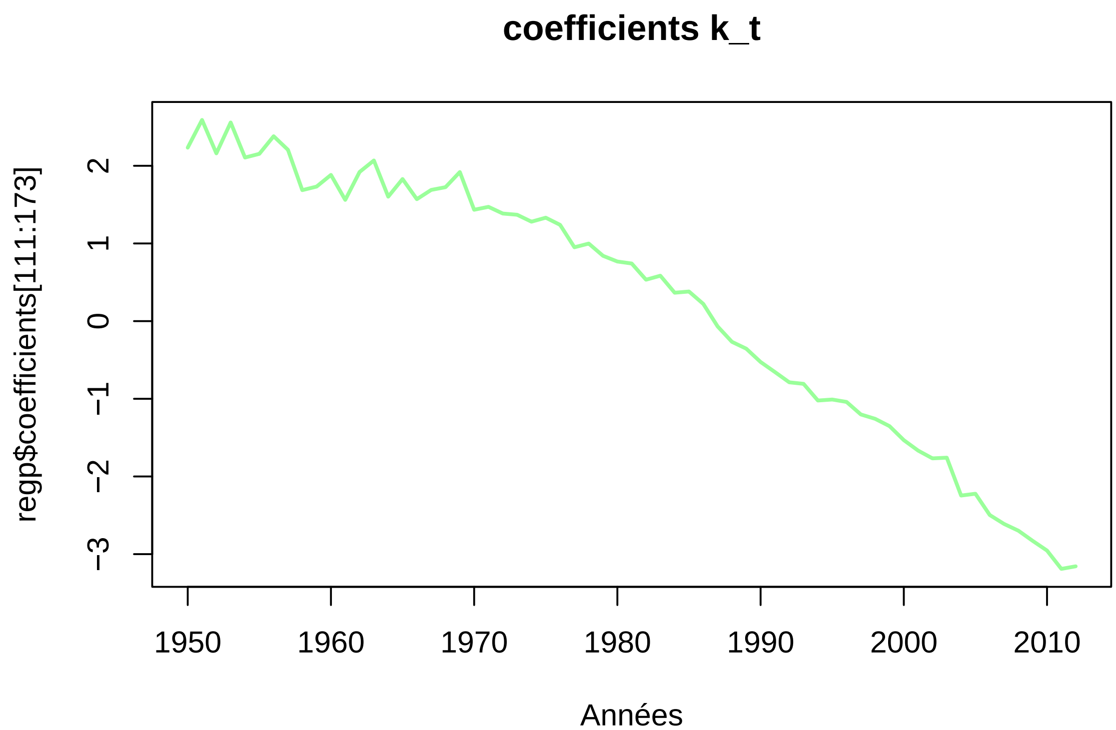
8.2.2.2 Estimation des coefficients :




```r
plot(
  45:99,
  regp$coefficients[56:110],
  main = "coefficients beta_x",
  xlab = "Ages",
  type = 'l',
  col = palette_couleur[2],
  lwd = 2
)
```




```r
plot(
  1950:2012,
  regp$coefficients[111:173],
  main = "coefficients k_t",
  xlab = "Années",
  type = 'l',
  col = palette_couleur[3],
  lwd = 2
)
```



Le coefficient α_x est la valeur moyenne de la mortalité pour l'âge x.

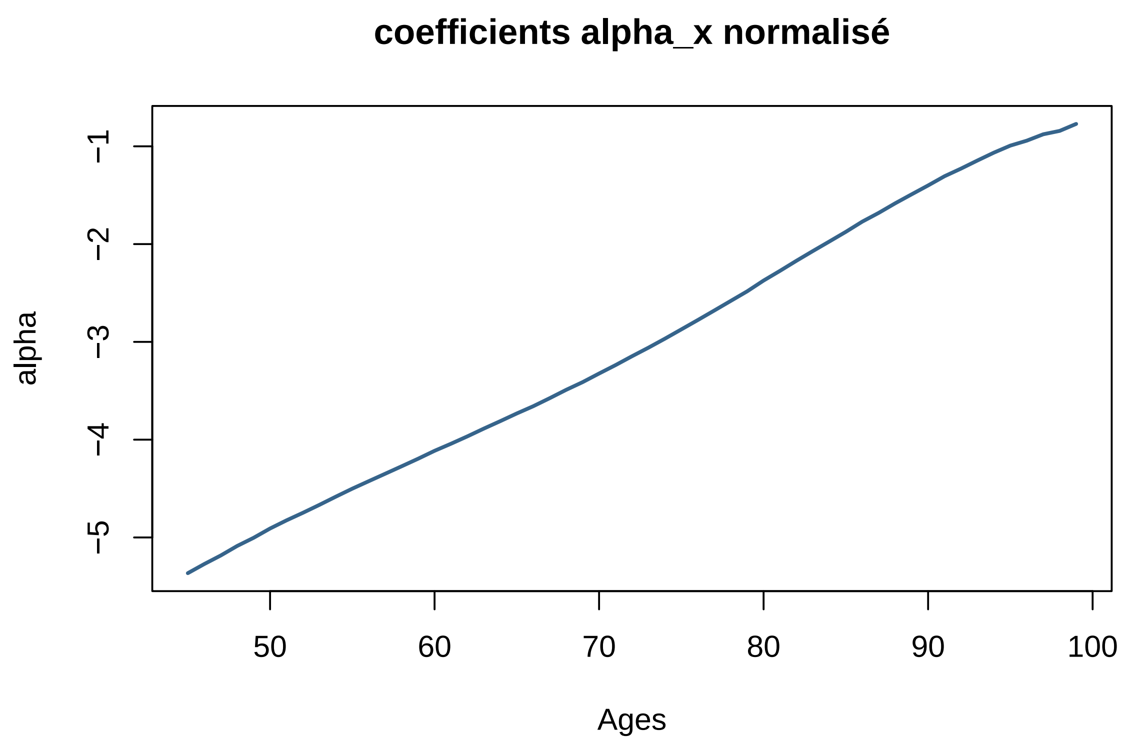
Le coefficient β_x est la sensibilité de la mortalité à l'âge x.

Le coefficient k_t est l'effet temporel.

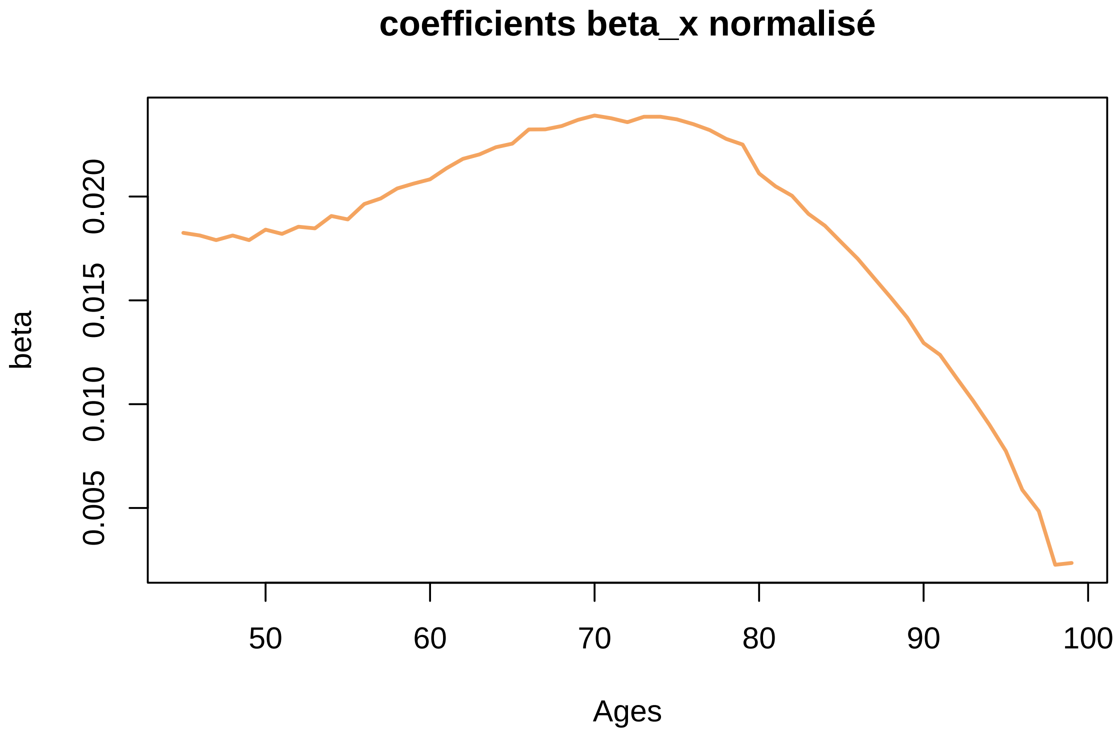
```
alpha = regp$coefficients[1:55]
k = regp$coefficients[111:173]
beta = regp$coefficients[56:110]

# On "normalise" les paramètres comme pour Lee-Carter :
sb = sum(beta)
beta = beta / sb
mk = mean(k)
k = (k - mk) * sb
alpha = alpha + beta * mk


plot(
  45:99,
  alpha,
  main = "coefficients alpha_x normalisé",
  xlab =
    "Ages",
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
```

```
)
```

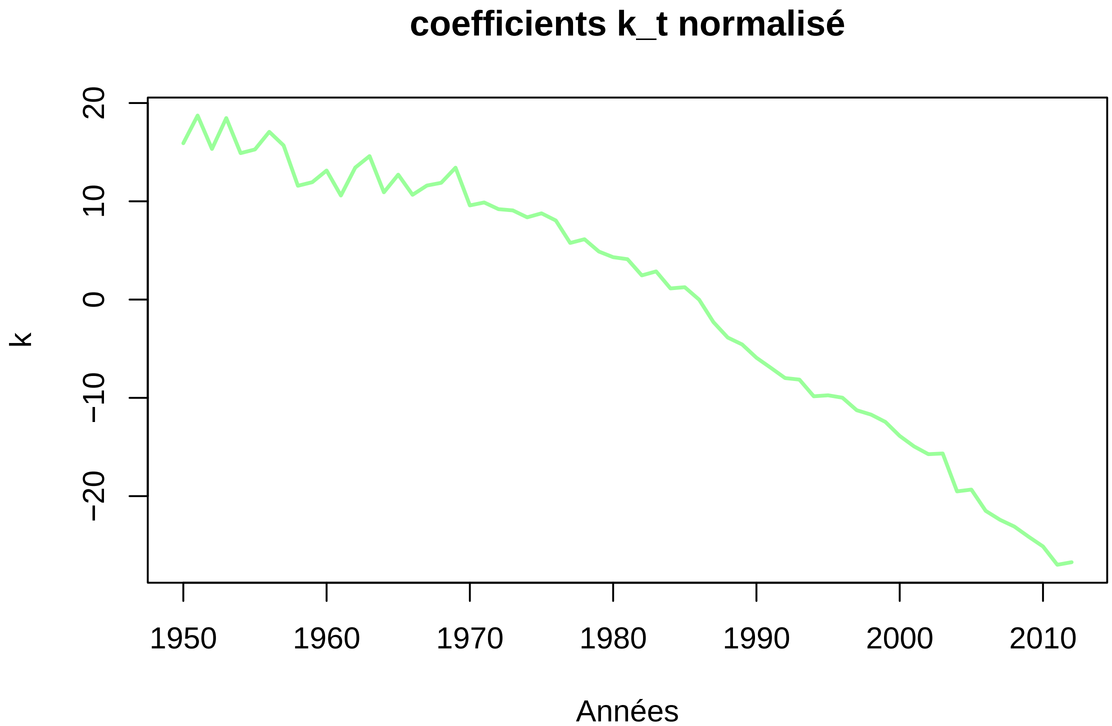
8.2.2.3 Normalisation des coefficients :




```
plot(
  45:99,
  beta,
  main = "coefficients beta_x normalisé",
  xlab = "Ages",
  type = 'l',
  col = palette_couleur[2],
  lwd = 2
)
```




```r
plot(
  1950:2012,
  k,
  main = "coefficients k_t normalisé",
  xlab = "Années",
  type = 'l',
  col = palette_couleur[3],
  lwd = 2
)
```



8.2.3 Comparaison avec Lee-Carter classique :

La fonction R du modèle de Lee-Carter calcule le logarithme de la mortalité :

$$\mu_{x,t} = \log(q_{x,t})$$

On utilise alors la relation ci-dessous pour faire le lien entre l'estimation de Lee-Carter et le modèle de Poisson :

$logit(q_{x,t}) = ln(\frac{q_{x,t}}{1-q_{x,t}}) \approx ln(\mu_{x,t})$

$logit(q_{x,t}) = \alpha_x + \beta_x k_t + \epsilon_{x,t}$

```
# calcul des log(mu_{x,t})
logmu = matrix(NA, nrow = 55, ncol = 63)
for (i  in 1:55)
{
  for (j in 1:63)
  {
    logmu[i, j] = alpha[i] + beta[i] * k[j]
  }
}
```

```
# Comparaison avec Lee Carter classique
library(demography)
muh = matrix(de$Male[ind] / ex$Male[ind], nl, nc)
poph = matrix(ex$Male[ind], nl, nc)
```

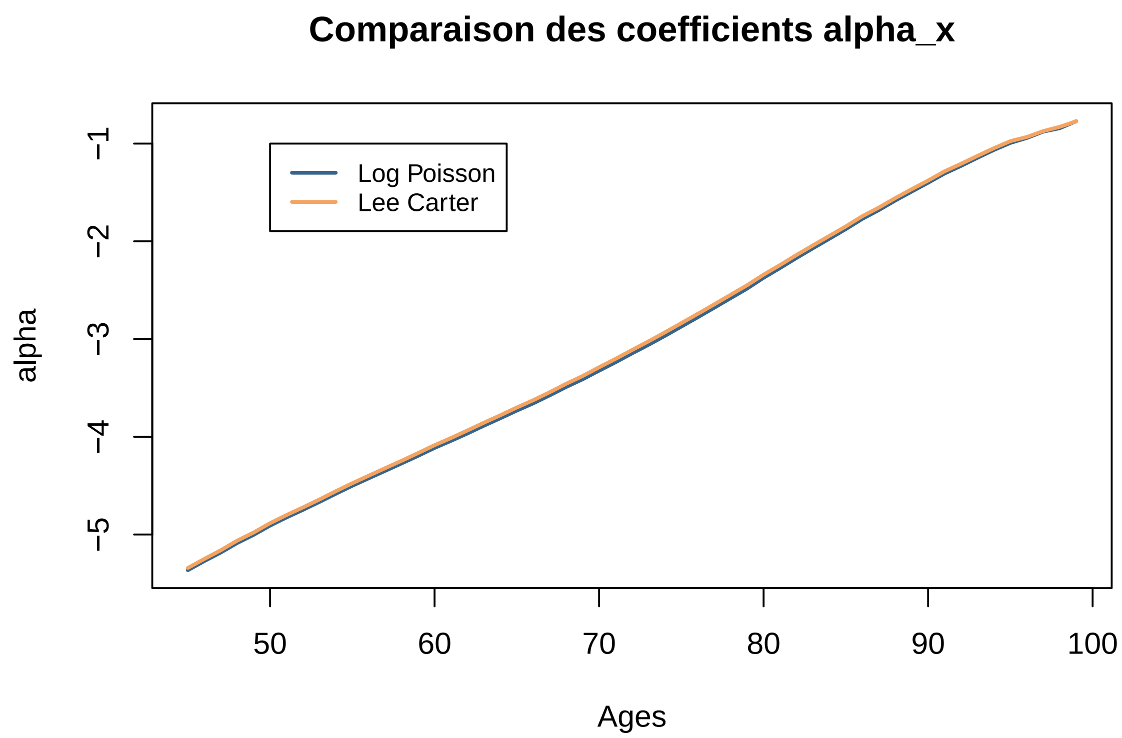
```r
Baseh = demogdata(
  data = muh, # taux de mortalité
  pop = poph, # population
  ages = age, # âges
  years = annee, # années
  type = "mortality", # type de données
  label = 'France', # label
  name = 'Hommes', # genre
  lambda = 1
)
lch = lca(Baseh)
plot(lch)
```

8.2.3.1 Modélisation de Lee-Carter classique :


```r
# Coefficients alpha_x :
plot(
  45:99,
  alpha,
  main = "Comparaison des coefficients alpha_x",
  xlab = "Ages",
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
lines(45:99, lch$ax, col = palette_couleur[2], lwd = 2)
```

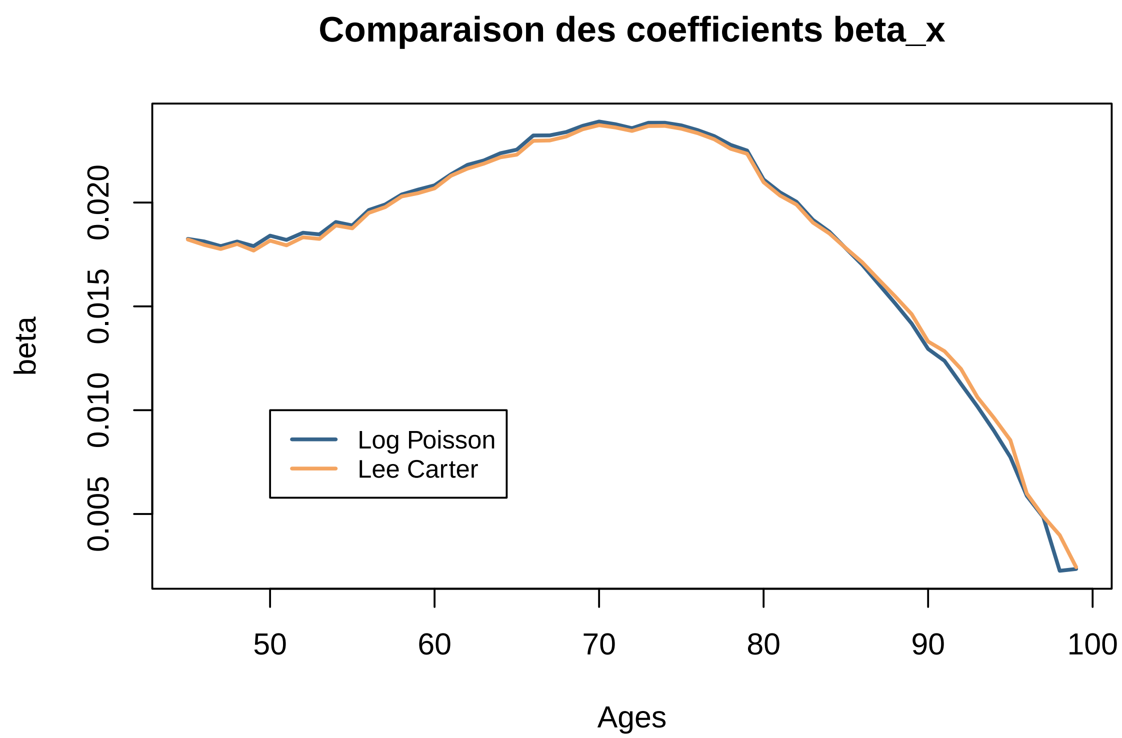
```
legend(
  50,
  -1,
  legend = c("Log Poisson", "Lee Carter"),
  col = c(palette_couleur[1], palette_couleur[2]),
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```

8.2.3.2 Comparaison des paramètres Log Poisson / Lee Carter :

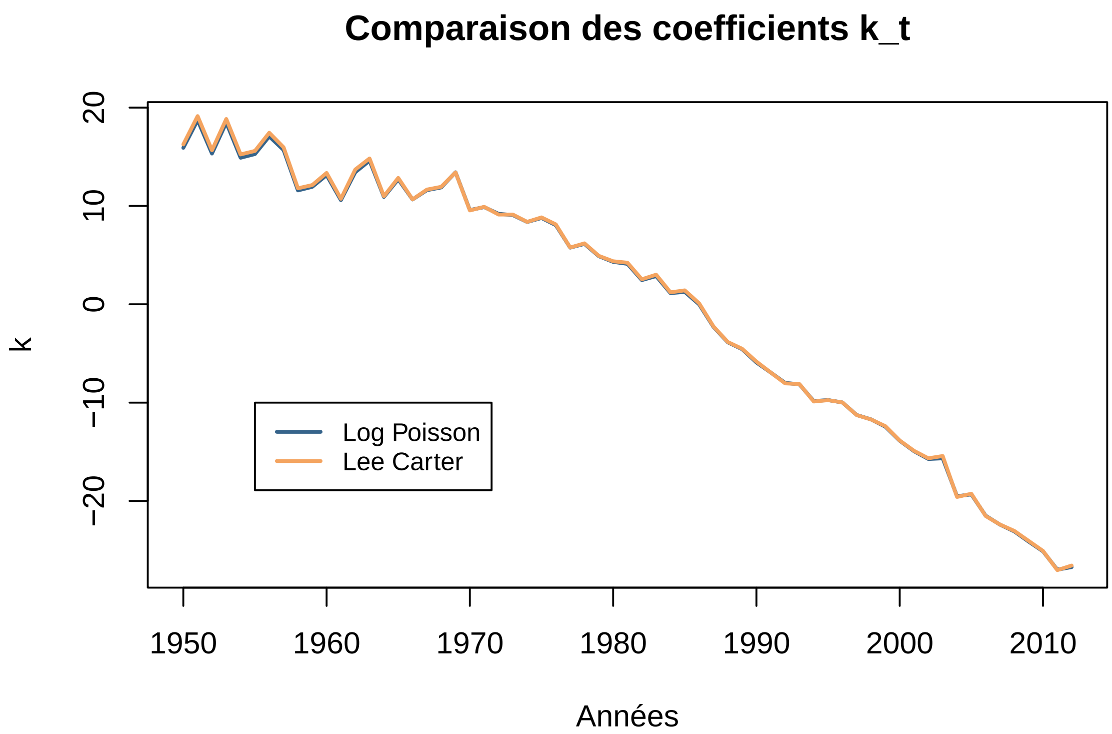



```
# Coefficients beta_x :
plot(
  45:99,
  beta,
  main = "Comparaison des coefficients beta_x",
  xlab = "Ages",
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
lines(45:99, lch$bx, col = palette_couleur[2], lwd = 2)
legend(
  50,
  0.01,
  legend = c("Log Poisson", "Lee Carter"),
  col = c(palette_couleur[1], palette_couleur[2]),
```

```
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```




```
# Coefficients k_t :
plot(
  1950:2012,
  k,
  main = "Comparaison des coefficients k_t",
  xlab = "Années",
  type = 'l',
  col = palette_couleur[1],
  lwd = 2
)
lines(1950:2012, lch$kt, col = palette_couleur[2], lwd = 2)
legend(
  1955,
  -10,
  legend = c("Log Poisson", "Lee Carter"),
  col = c(palette_couleur[1], palette_couleur[2]),
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```




```r
# Prédiction de Lee-Carter
predh=lch$fitted$y # c'est log(mu_{x,t}) qui est prédit

# Comparaison Lee-Carter / données réelles
rmseh=sqrt(sum((log(muh)-(predh))^2)/(nl-1)/nc)

# Comparaison Log Poisson / données réelles
rmseh1=sqrt(sum((logmu-(log(muh)))^2)/(nl-1)/nc)

# Comparaison Log Poisson / Lee-Carter
rmseh2=sqrt(sum((logmu-predh)^2)/(nl-1)/nc)

# Tableau comparatif des erreurs :
dt = t(round(data.frame("Lee Carter vs reel" = rmseh,
           "Log Poisson vs reel" =   rmseh1,
           "Log Poisson vs  Lee Carter" = rmseh2),4))
colnames(dt) = c("Erreur quadratique moyenne")
dt
```

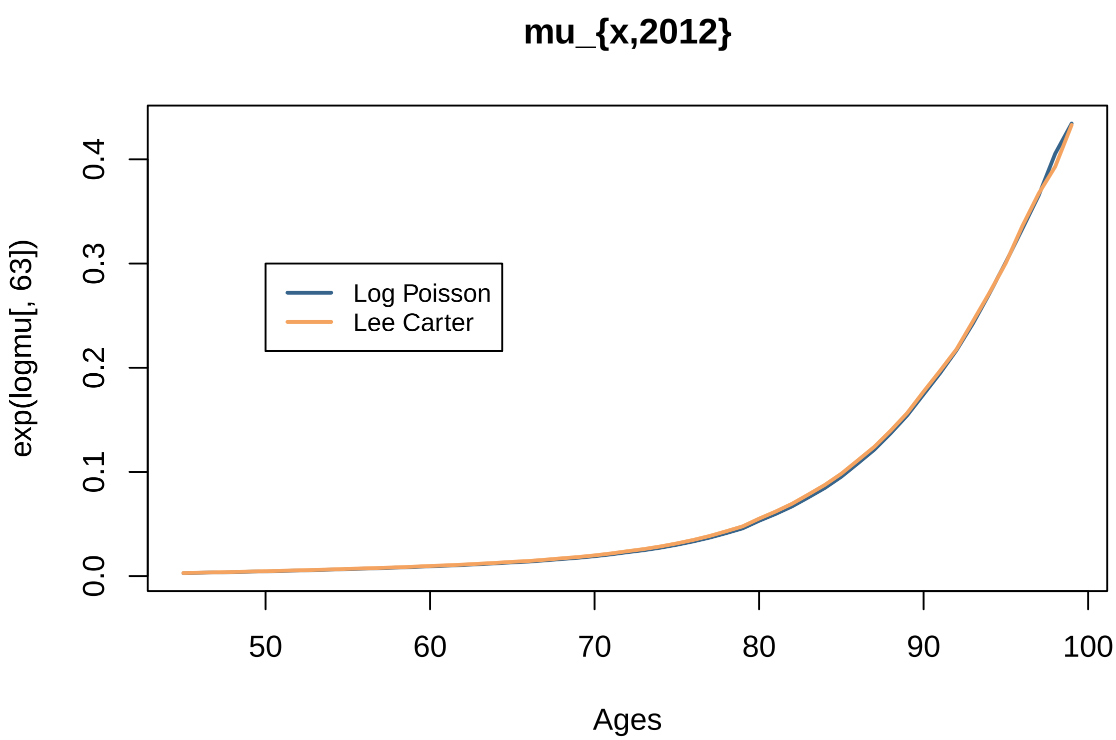
8.2.3.3 Etude de l'erreur de prédiction des modèles :

```
                           Erreur quadratique moyenne
Lee.Carter.vs.reel                             0.0456
Log.Poisson.vs.reel                            0.0537
Log.Poisson.vs..Lee.Carter                     0.0301
```

8.2.3.3.1 Comparaison graphique des prédictions :

- On compare graphiquement $\mu_{x,t}$ prédit par le modèle de Lee-Carter et le modèle de Log-Poisson pour l'année 2012.

```
plot(
  45:99,
  exp(logmu[, 63]),
  main = "mu_{x,2012}",
  type = 'l',
  xlab = "Ages",
  col = palette_couleur[1],
  lwd = 2
)
lines(45:99, exp(predh[, 63]), col = palette_couleur[2], lwd = 2)
legend(
  50,
  0.3,
  legend = c("Log Poisson", "Lee Carter"),
  col = c(palette_couleur[1], palette_couleur[2]),
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```



- Comparaison des log(mu_{x,2012}) prédits par les deux modèles et les données observées :

```
plot(
  45:99,
  logmu[, 63],
  main = "log(mu_{x,2012})",
```

```
  type = 'l',
  xlab = "Ages",
  col = palette_couleur[1],
  lwd = 2
)
lines(45:99, predh[, 63], col = palette_couleur[2], lwd = 2)
lines(45:99, log(muh[, 63]), col = palette_couleur[3], lwd = 2)
legend(
  50,
  -1,
  legend = c("Log Poisson", "Lee Carter", "obs."),
  col = c(palette_couleur[1], palette_couleur[2], palette_couleur[3]),
  lty = 1,
  cex = 0.8,
  lwd = 2
)
```