DEVOIR 1: Assurance non vie

CHARET Mohamed

2022-10-10

Numéro 10

Exercice 1

1.1 Importation des données

```
M<-read.csv("montantssinistre10.csv", sep=";",dec = ",")
head(M)
```

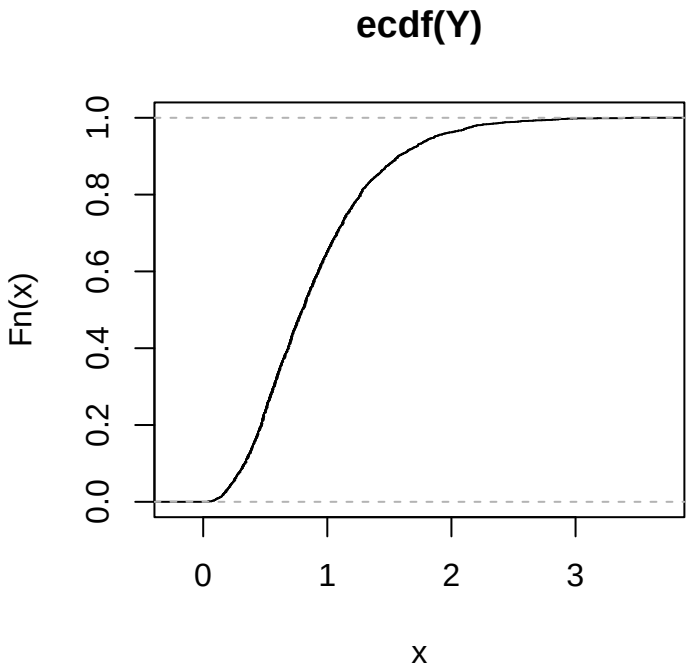
```
##   X         x
## 1 1 1.4336612
## 2 2 1.2629764
## 3 3 0.6615985
## 4 4 0.8315340
## 5 5 0.3920018
## 6 6 0.5767425
```

```
Y<-M$x
head(Y)
```

```
## [1] 1.4336612 1.2629764 0.6615985 0.8315340 0.3920018 0.5767425
```

1.2 Analyse Exploratoire

```
# Traçons la fonction de répartition empirique
plot(ecdf(Y))
```



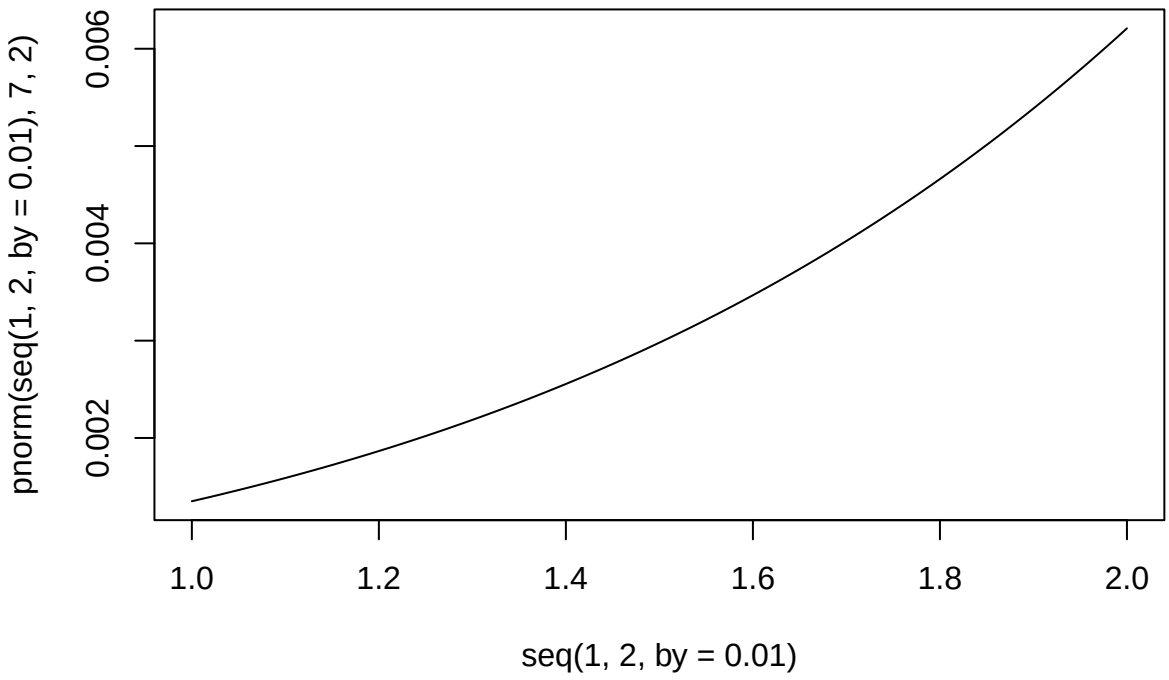
Maintenant pour choisir la bonne modélisation de cette distribution nous allons visualiser les fonctions de répartition suivantes: normal, lognormal, exponontielle, Gamma.

```
library(MASS)

fitnormal<-fitdistr(Y,"normal")$estimate
fitlognormal<-fitdistr(Y,"lognormal")$estimate
fitexpo<-fitdistr(Y,"exponential")$estimate
fitGamma<-fitdistr(Y,"gamma")$estimate
```

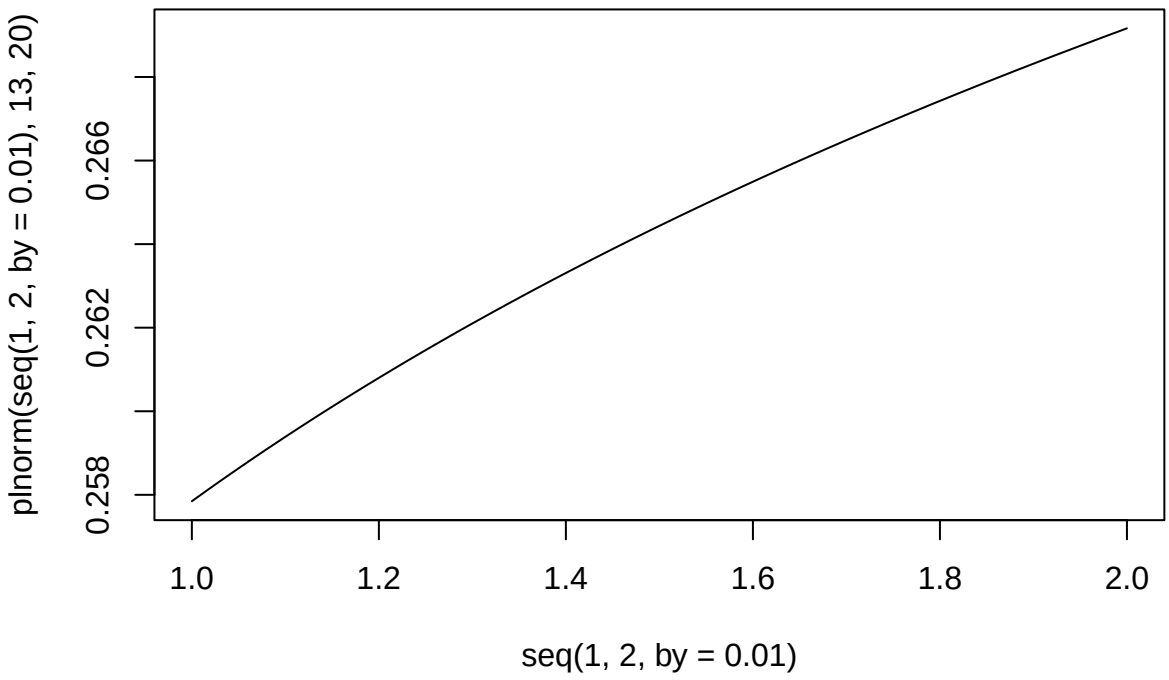
```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

Représentation Graphique:

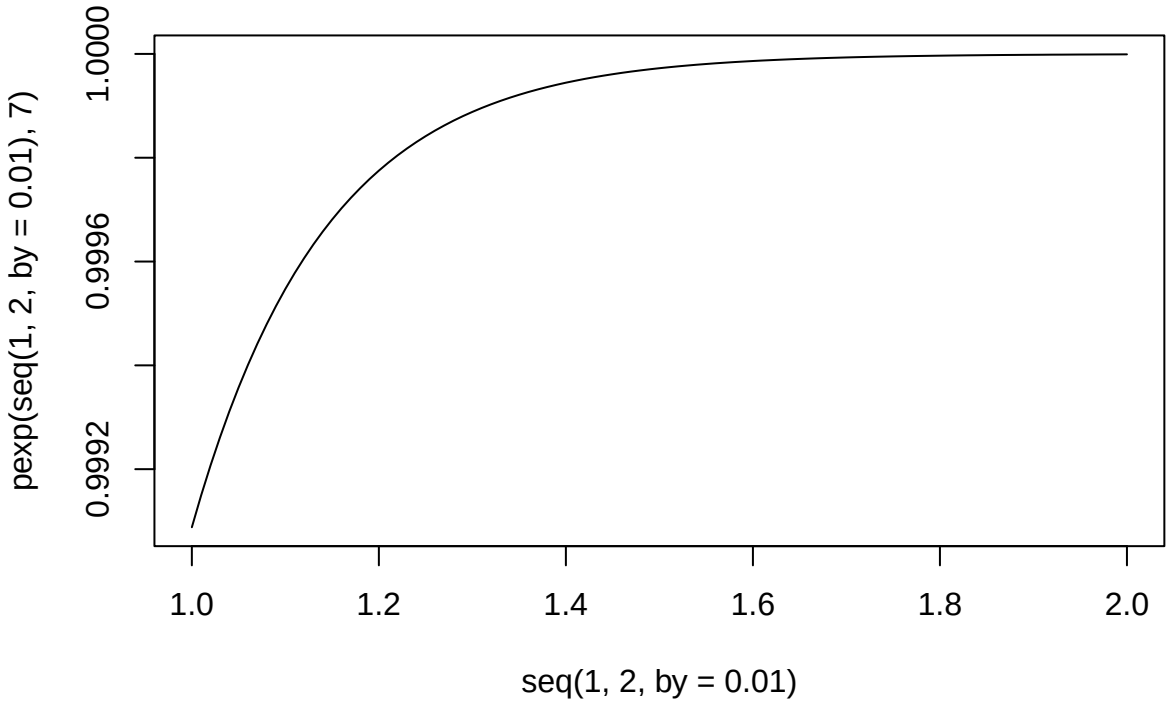
```
plot(seq(1,2,by=0.01),pnorm(seq(1,2,by=0.01),7,2),type="l")
```



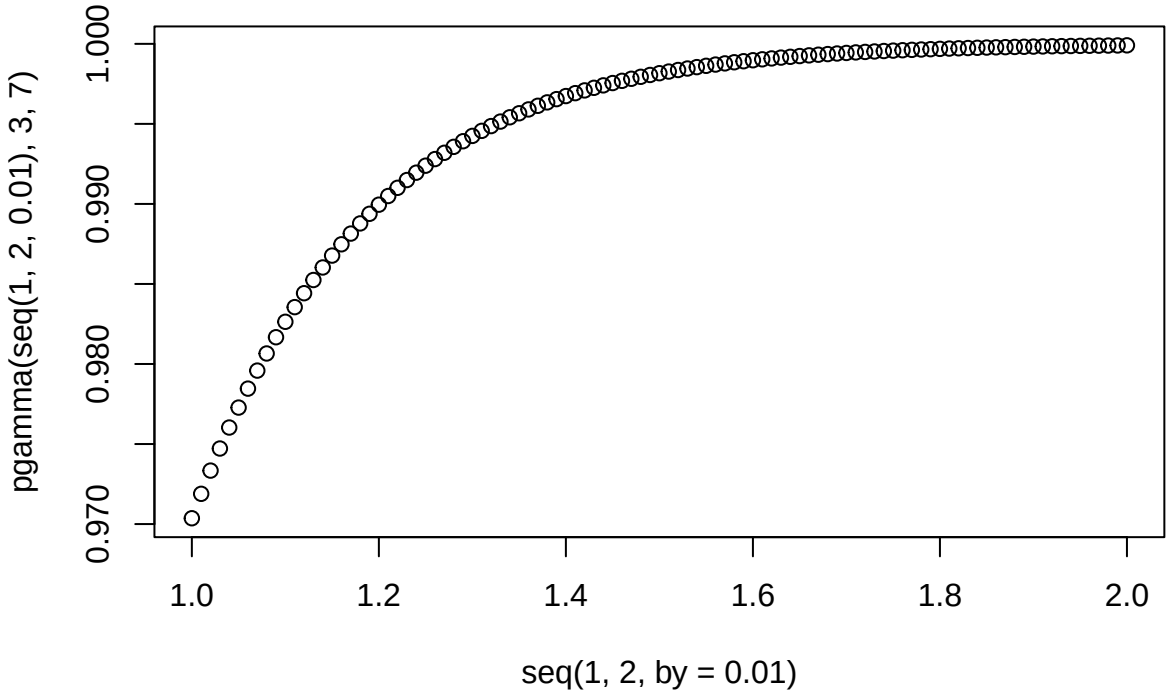

```
plot(seq(1,2,by=0.01),plnorm(seq(1,2,by=0.01),13,20),type="l")
```




```
plot(seq(1,2,by=0.01),pexp(seq(1,2,by=0.01),7),type="l")
```

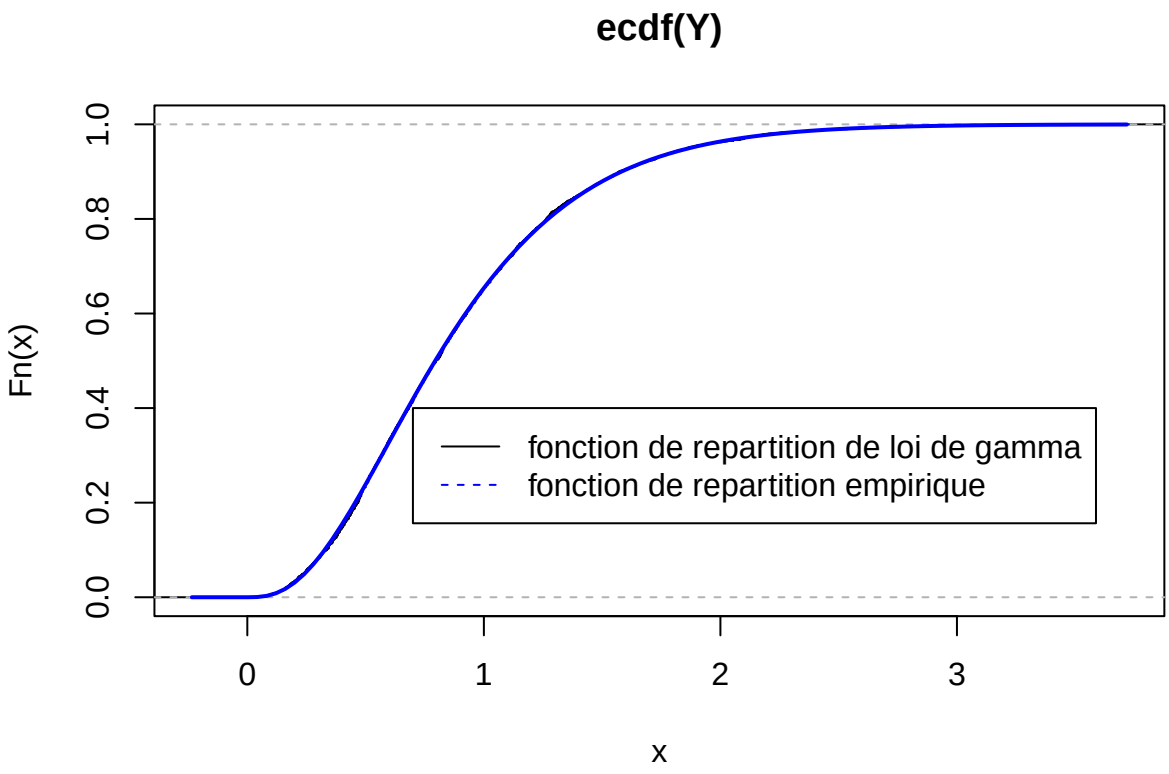



```
plot(seq(1,2,by=0.01),pgamma(seq(1,2,0.01),3,7))
```



D'après ces figures on en déduit que la loi la plus adéquat pour cette distribution est la loi Gamma.

```
plot(ecdf(Y))
curve(pgamma(x,fitGamma[1],fitGamma[2]),add= TRUE,lwd=2,col="blue")
legend(x=0.7,y=0.4,legend = c("fonction de repartition de loi de gamma", "fonction de repartition empir
```



1.3 Validation Modèle paramétrique àl'aide de test statistique Y: Le Montant de sinistre.

$$Y \sim \mathcal{G}(\alpha, \beta)$$

Pour valider ce module on va utiliser le test de **Kolmogrov-Smirnov**

$$\begin{cases} H_0 : F_0 \textit{ le modèle exponentiel modélise bien notre base de données}, \\ H_1 : F \textit{ le modèle exponentiel ne modélise pas notre base de données}. \end{cases}$$

F_0 : est la fonction de répartition de la loi exponentielle, la loi gamma, loi de Gamma, loi normale ou la loi lognormale. F est la fonction de répartition empirique de la base de données.

```
# Estimation paramétrique:
library(MASS)
lambda <-fitdistr(Y,"exponential")$estimate
lambda
```

```
##     rate
## 1.122191
```

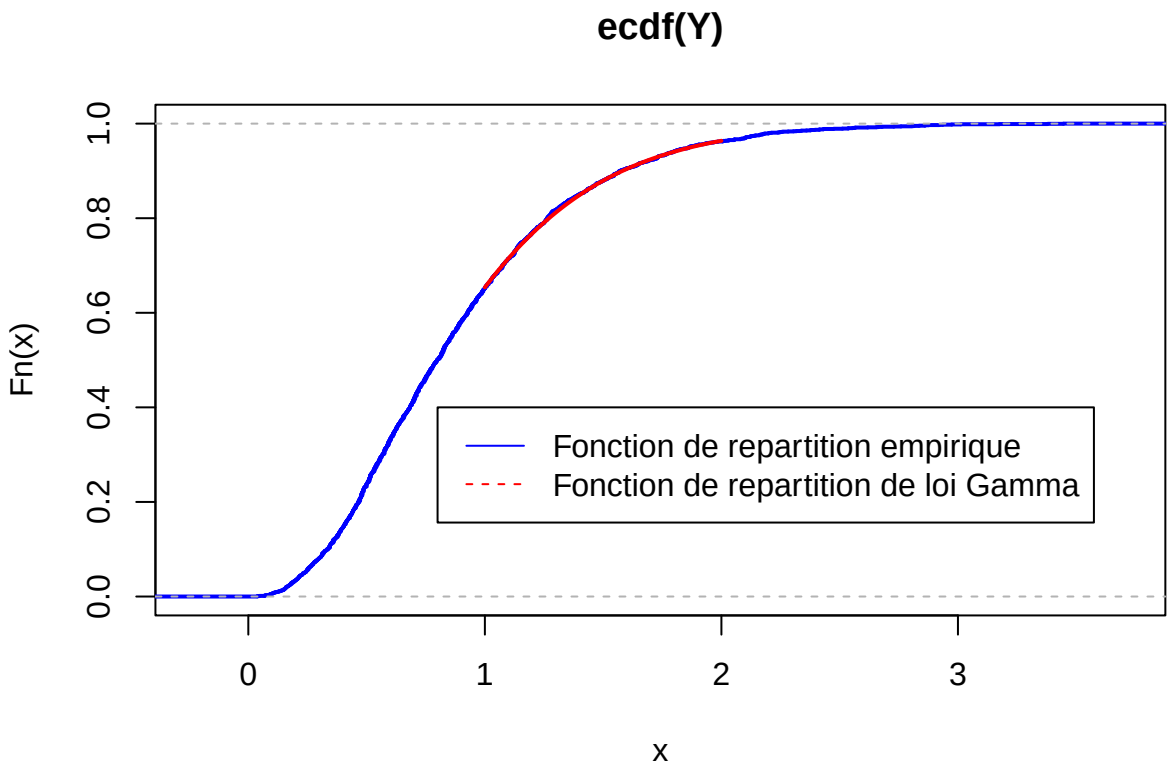
```
a<-fitdistr(Y,"gamma")$estimate
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
a
```

```
##    shape     rate
## 2.995545 3.361574
```

```r
plot(ecdf(Y),col="blue",lwd=2)
lines(seq(1,2,by=0.01),pgamma(seq(1,2,by=0.01),shape=a[1],rate=a[2]),type="l",col="red",lwd=2)
legend(x=0.8,y=0.4,legend = c("Fonction de repartition empirique","Fonction de repartition de loi Gamma
```



effectuons le test de **Kolmogrov-Smirov** par R

```r
ks.test(Y,pgamma,fitGamma[1],fitGamma[2])
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  Y
## D = 0.0099791, p-value = 0.976
## alternative hypothesis: two-sided
```

```r
ks.test(Y,pexp,fitexpo[1])
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  Y
## D = 0.21704, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
ks.test(Y,pnorm,fitnormal[1],fitnormal[2])
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
```

```
## data:  Y
## D = 0.077584, p-value = 1.888e-12
## alternative hypothesis: two-sided
```

```
ks.test(Y,plnorm,fitlognormal[1],fitlognormal[2])
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  Y
## D = 0.045147, p-value = 0.0001695
## alternative hypothesis: two-sided
```

Résultat: Donc ce test confirme que la loi qui bien modélise la distribution est la loi de Gamma.

1.4 Critère bayesien de Schwarz(SBC) et Critère d'information de Akaike(AIC)

Ces critère sert à choisir entre les modèles qui modélisent la base de donnée, tel que on choisi celui qui a le petit AIC. Pour le critère schwarz, on choisi celui qui a la plus grande valeur de SBC.\ Dans notre cas,l'utilisation de ces critères n'ont aucun importance car On a un seul modèle qui modélise notre base de donnée.. Calculons le coefficient AIC pour chaque loi dans R:

```
AICGamma<-4-fitdistr(Y,"gamma")$loglik
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
AICGamma
```

```
## [1] 1462.737
```

```
AICexp<-2-fitdistr(Y,"exponential")$loglik
AICexp
```

```
## [1] 2036.849
```

```
AIClog<-4-fitdistr(Y,"lognormal")$loglik
AIClog
```

```
## [1] 1539.568
```

```
AICnorm<-4-fitdistr(Y,"normal")$loglik
AICnorm
```

```
## [1] 1728.906
```

Calculons le coefficient SBC pour chaque loi dans R:

```
SBCgamma<-fitdistr(Y,"gamma")$loglik-log(length(Y))*0.5*2
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced

## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
SBCgamma
```

```
## [1] -1466.477
```

```
SBCexp<-fitdistr(Y,"exponential")$loglik-log(length(Y))*0.5
SBCexp
```

```
## [1] -2038.72
```

```
SBClognor<-fitdistr(Y,"lognormal")$loglik-log(length(Y))*0.5*2
SBClognor
```

```
## [1] -1543.309
```

```
SBCnor<-fitdistr(Y,"normal")$loglik-log(length(Y))*0.5*2
SBCnor
```

```
## [1] -1732.647
```

2.Exercice 2(nombresinistres10)

2.1 Importation des données

```
library(MASS)
Mohamed<-read.csv("nombresinistre10.csv",sep = ";")
head(Mohamed)
```

```
##   X x
## 1 1 6
## 2 2 5
## 3 3 3
## 4 4 4
## 5 5 3
## 6 6 5
```

```
y<-Mohamed$x
head(y)
```

```
## [1] 6 5 3 4 3 5
```

2.2 Analyse Exploratoire

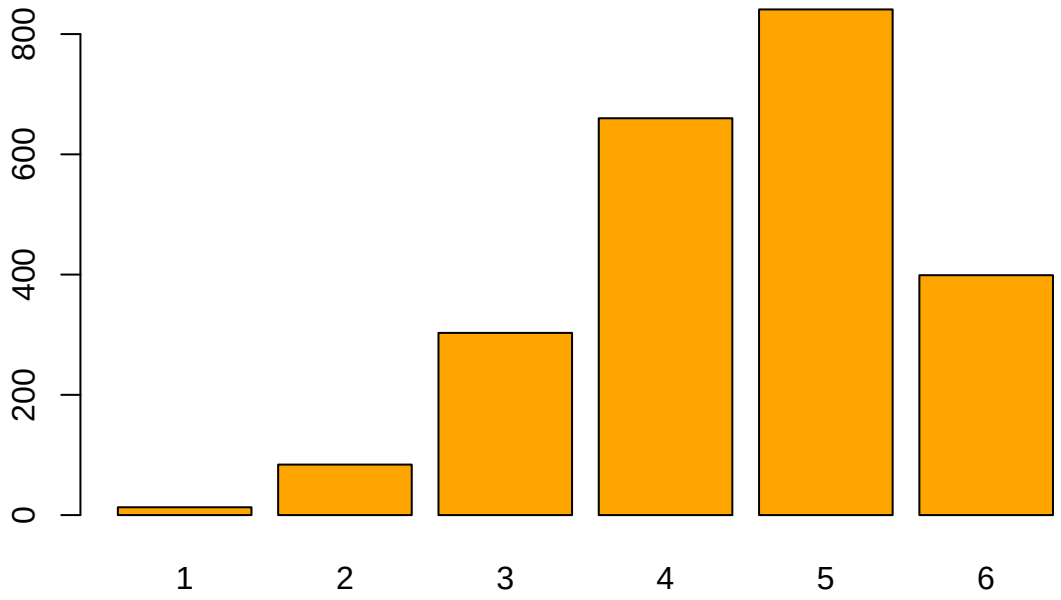
```
x<-table(y)
mean(y)
```

```
## [1] 4.49087
```

```
var(y)
```

```
## [1] 1.147807
```

```
barplot(x,col="orange")
```



On constate que la moyenne de nombres de sinistres est supérieur à la variance, donc on peut modéliser cette base de données par la loi binomiale. ### 2.3 Modèle paramétrique Soit N une variable aleatoire qui suit la loi Binomiale:

$$N \sim \mathcal{B}(n,p),\ n \in \mathbf{N},\ p \in [0,1].$$

2.3.1 estimation paramétrique 1.Loi binomial

```
library(vcd)
```

```
## Loading required package: grid
```

```
goodfit(y,"binom")
```

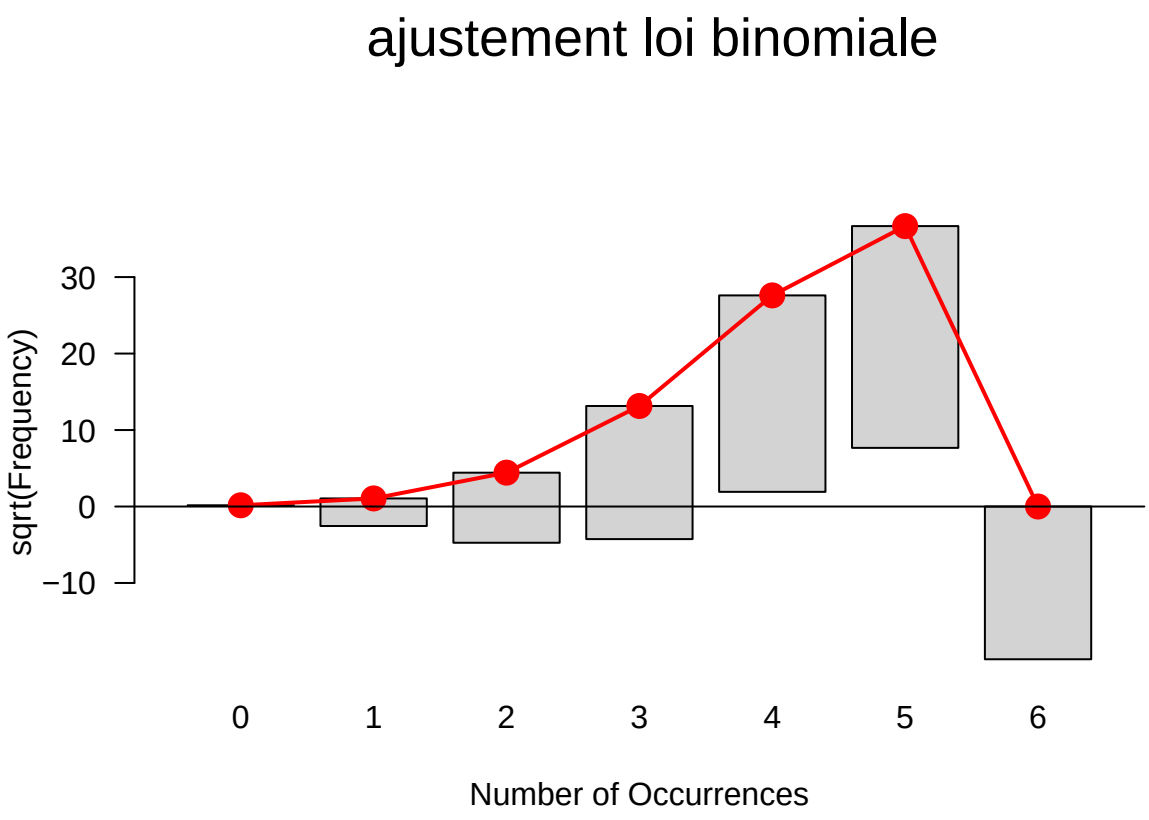
```
## Warning in goodfit(y, "binom"): size was not given, taken as maximum count
```

```
##
## Observed and fitted values for binomial distribution
## with parameters estimated by `ML'
##
##  count observed      fitted pearson residual
##      0        0   0.5823459       -0.7631159
##      1       13  10.3976672        0.8070397
##      2       84  77.3534318        0.7557144
##      3      303 306.9177364       -0.2236270
##      4      660 684.9942307       -0.9549845
##      5      841 815.3621906        0.8978538
##      6      399 404.3923973       -0.2681516
```

```
parmbino<-goodfit(y,"binom", par=list(size=5))$par
parmbino
```

```
## $prob
## [1] 0.8981739
##
## $size
## [1] 5
```

```
plot(goodfit(y,"binom",par=list(size=5)),main="ajustement loi binom
iale")
```



2.Loi de poisson

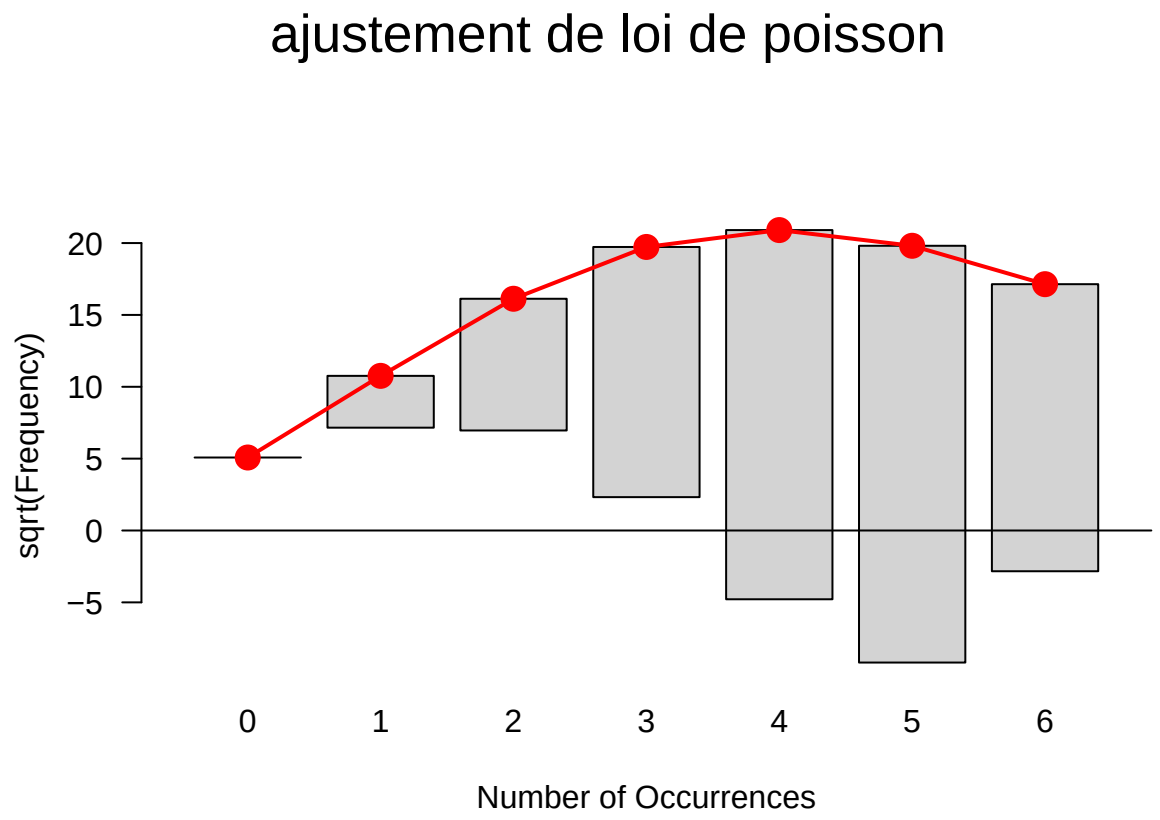
```
goodfit(y,"pois")
```

```
##
## Observed and fitted values for poisson distribution
## with parameters estimated by `ML'
##
##  count observed    fitted pearson residual
##      0        0  25.78505        -5.077898
##      1       13 115.79729        -9.552839
##      2       84 260.01527       -10.915683
##      3      303 389.23155        -4.370813
##      4      660 436.99703        10.667715
##      5      841 392.49934        22.638290
##      6      399 293.77722       -10.765952
```

```
parmpois=goodfit(y,"pois")$par
parmpois$lambda
```

```
## [1] 4.49087
```

```
plot(goodfit(y,"pois"),main="ajustement de loi de poisson")
```



3. Loi binomiale négative

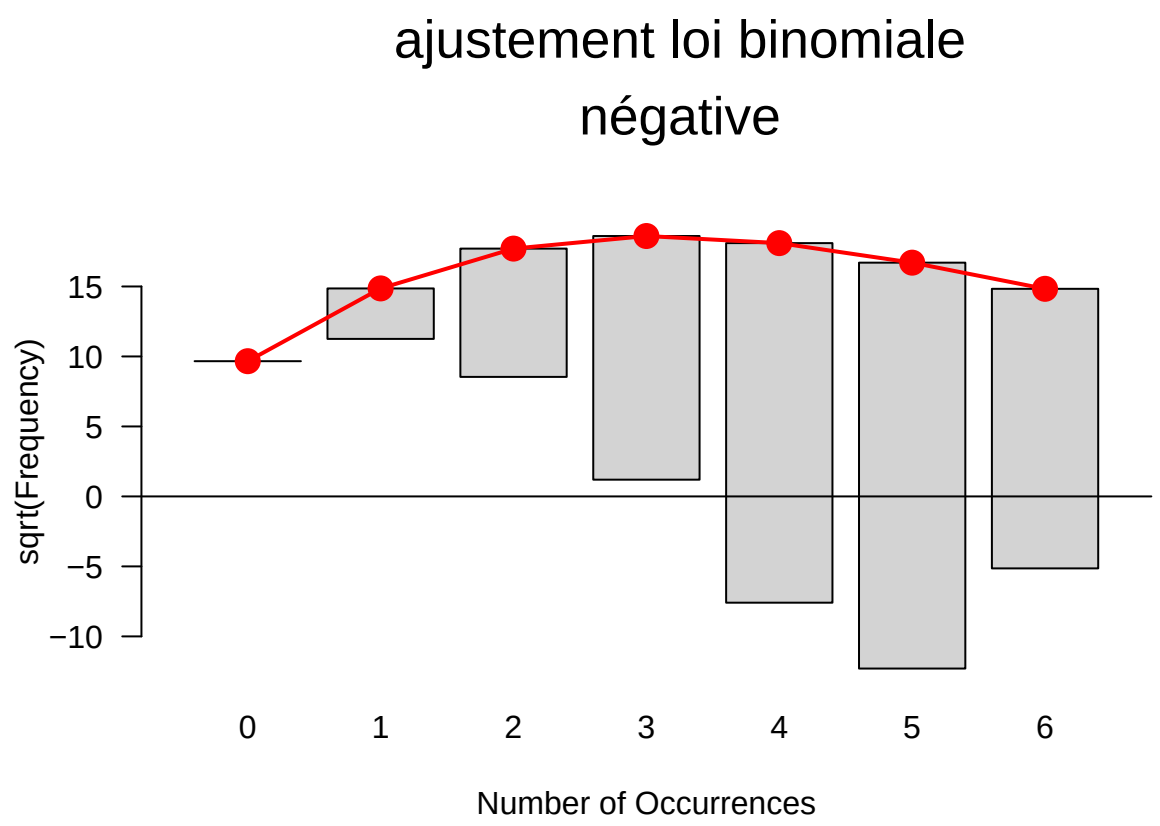
```
goodfit(y,"nbinom",par=list(size=5))
```

```
##
## Observed and fitted values for nbinomial distribution
## with parameters estimated by `ML with size fixed'
##
##  count observed    fitted pearson residual
##      0        0  93.33571        -9.661041
##      1       13 220.82197       -13.985253
##      2       84 313.46421       -12.960477
##      3      303 346.09011        -2.316237
##      4      660 327.52437        18.371220
##      5      841 278.95912        33.650973
##      6      399 219.99547       -11.957302
```

```
parmnbino=goodfit(y,"nbinom",par=list(size=5))$par;
parmnbino
```

```
## $size
## [1] 5
##
## $prob
## [1] 0.5268221
```

```
plot(goodfit(y,"nbinom",par=list(size=5)),main="ajustement loi bino
miale négative")
```



2.3.2 Vérification des résultats à l'aide de test de $\tilde{\chi}^2(k-n-1)$

1. Loi de poisson:

```
k=rep(0,length(x))
k
```

```
## [1] 0 0 0 0 0 0
```

```
k[1:length(x)]=x[1:length(x)]
k
```

```
## [1]  13  84 303 660 841 399
```

```
length(y)*dpois(0:6,parmpois$lambda)<5
```

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

```
#le nombre espere des donnees dans chacun des classes
pp<-c(dpois(0:4,parmpois$lambda),1-sum(dpois(0:4,parmpois$lambda)))
pp
```

```
## [1] 0.01121089 0.05034665 0.11305012 0.16923111 0.18999871 0.46616252
```

```
length(y)*pp
```

```
## [1]   25.78505  115.79729  260.01527  389.23155  436.99703 1072.17380
```

```
chisq.test(k,p=pp)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  k
## X-squared = 1006.7, df = 5, p-value < 2.2e-16
```

```
qchisq(0.95,4)
```

```
## [1] 9.487729
```

```
pvalue=1-pchisq(chisq.test(k,p=pp)$statistic,4)
pvalue
```

```
## X-squared
##         0
```

-On remarque que p-value<0.05, alors on en déduit que la loi de poisson ne modélise pas notre base de données.

2. Loi de binomiale negative:

```
length(y)*dnbinom(1:4,size=parmnbino$size,prob=parmnbino$prob)
```

```
## [1] 220.8220 313.4642 346.0901 327.5244
```

```
k=rep(0,length(x))
k[1:length(x)]=x[1:length(x)]
k
```

```
## [1]  13  84 303 660 841 399
```

```
nn<-c(dnbinom(1:4,size = parmnbino$size,prob = parmnbino$prob),1-sum(dbinom(1:4,size=parmnbino$size,prob
```

3. Loi binomiale:

```
length(y)*dbinom(0:5,size=parmbino$size,prob=parmbino$prob)<5
```

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

```
k<-rep(0,length(x))
k
```

```
## [1] 0 0 0 0 0 0
```

```
k[1:length(x)]=x[1:length(x)]
pp=c(dbinom(0:4,size=parmbino$size,prob=parmbino$prob),1-sum(dbinom
(0:4,size=parmbino$size,prob=parmbino$prob)))
pp
```

```
## [1] 0.0000109470 0.0004827993 0.0085172234 0.0751275836 0.3313376650
## [6] 0.5845237817
```

```
length(y)*pp
```

```
## [1] 2.517811e-02 1.110438e+00 1.958961e+01 1.727934e+02 7.620766e+02
## [6] 1.344405e+03
```

```
chisq.test(k,pp)
```

```
## Warning in chisq.test(k, pp): Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  k and pp
## X-squared = 30, df = 25, p-value = 0.2243
```

```
qchisq(0.95,4)
```

```
## [1] 9.487729
```

```
pvalue=1-pchisq(chisq.test(k,p=pp)$statistic,4)
```

```
## Warning in chisq.test(k, p = pp): Chi-squared approximation may be incorrect
```

```
pvalue
```

```
## X-squared
##         0
```

Conclusion: Le modèle le plus adéquat pour notre base de données est le modèle Binomiale.