Codes R pour le cours de Lyon du 2 au 9 novembre 2017

Etienne Marceau

14 novembre 2017

Résumé

Le présent document contient les exemples de codes R construits pendant le cours de Lyon du 2 au 9 novembre 2017.

Table des matières

1 Préface **3**

2 Notions de base **4**
2.1 Loi gamma et mesures de risque 4
2.2 Loi binomiale et mesures de risque 5
2.3 Mutualisation des risques 7
2.4 Générateur congruentiel linéaire 9
2.5 Simulation Monte-Carlo 10

3 Modèles de bases en actuariat non-vie **12**
3.1 Loi Poisson composée avec sinistre individuel de loi gamma 12
3.2 Loi Binomiale négative composée (sinistres de loi lognormale) 15

4 Méthodes d'agrégation **18**
4.1 Algorithme de Panjer 18
4.2 FFT 22

5 Lois multivariées discrètes et composées **26**
5.1 Loi Poisson bivariée de Teicher 26
5.2 Loi composée bivariée 29

6 Remerciements **32**

Références **33**

1 Préface

Outil pédagogique. Le présent document est un outil pédagogique qui porte sur la modélisation des risques en actuariat. Il contient des codes R qui ont été rédigés dans le cadre d'un cours sur le sujet (ISFA, Université Lyon 1). Il est un complément à l'ouvrage de référence (Marceau 2013) pour le cours.

Prérequis. Les prérequis pour ce document sont principalement des cours de bases en mathématiques, en probabilité et en statistique.

Conditions d'utilisation. Ce document est en cours de rédaction, ce qui implique que son contenu est continuellement révisé et mis à jour. Bien qu'il utilise R depuis 2000, son auteur se considère comme un débutant en R. Les codes R peuvent aussi être conçus de manière plus efficiente. De plus, il peut y avoir encore des erreurs et son contenu doit être encore amélioré. Pour cette raison, le lecteur à inviter à nous communiquer tout commentaire et / ou correction qu'il peut avoir. Les conditions suivantes d'utilisation doivent être respectées :

- Ce document a été conçu pour des fins pédagogiques, personnelles et non-commerciales. Toute utilisation commerciale ou reproduction est interdite.
- Son contenu demeure la propriété de son auteur.

Calculs et illustrations. Toutes les calculs et les illustrations ont été réalisés dans le langage R grâce au logiciel GNU R mis à disposition par le R Project. Les codes R ont été conçus dans l'environnement de développement intégré RStudio.

Le logiciel GNU R et les bibliothèques sont disponibles sur le site du R Project et du Comprehensive R Archive Network (CRAN) :

- https://cran.r-project.org/

L'environnement RStudio est disponible sur le site suivant :

- https://www.rstudio.com/products/rstudio/download/.

Édition. Le présent document a été rédigé en R Markdown dans l'environnement R Studio. Pour cette raion, ce document peut être vu comme un outil interactif. Notamment, il est possible d'éxécuter l'ensemble du contenu en modifiant les paramètres de codes R. Pour une introduction à R Markdown, voir :

- http://archimede.mat.ulaval.ca/dokuwiki/doku.php?id=r:communication:rmarkdown
- https://rmarkdown.rstudio.com/

Citation : Marceau, E. (2017). Codes R pour le cours de Lyon 2-9 novembre 2017. Disponible sur : https://www.actrisk.act.ulaval.ca

Dernière version : 14 novembre 2017.

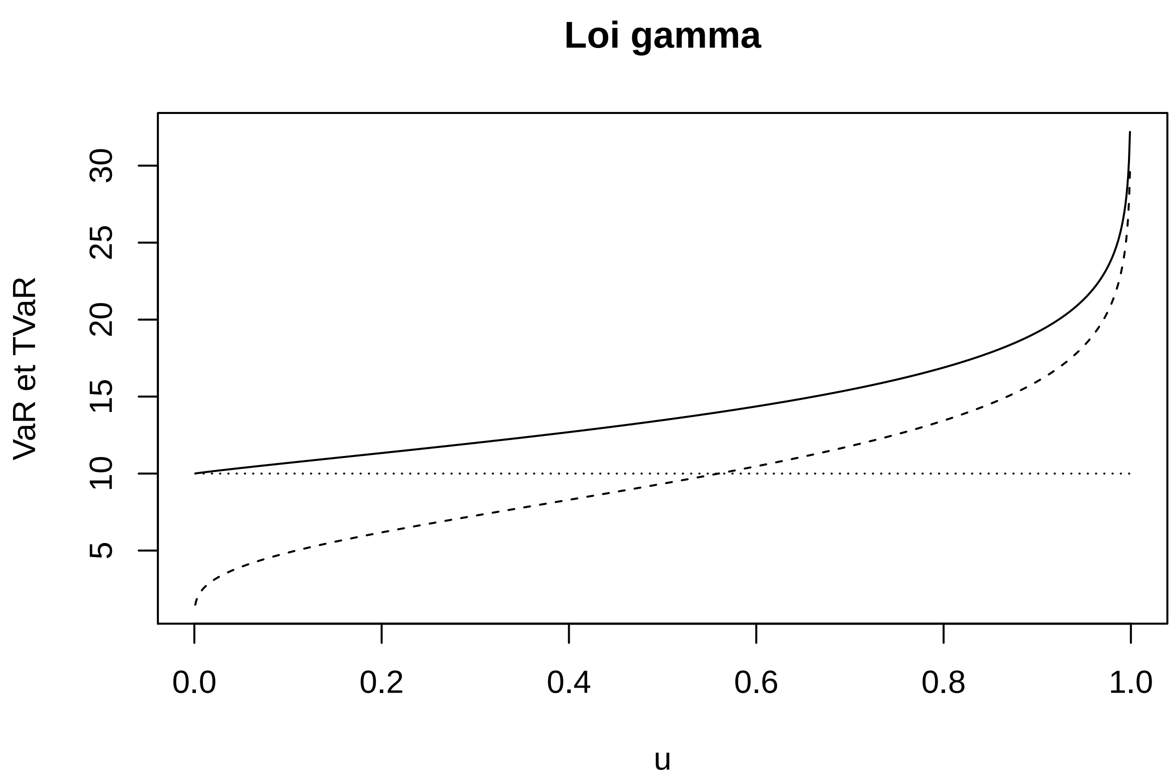
2 Notions de base

2.1 Loi gamma et mesures de risque

Soit $X \sim Gamma(\alpha, \beta)$. Le code R contient des calculs pour $VaR_\kappa(X)$ et $TVaR_\kappa(X)$.

Code R :

```
# Loi gamma
#
alp<-5
bet<-1/2
vu<-(1:999)/1000
VaRX<-qgamma(vu,alp,bet)
EX<-alp/bet
vEX<-rep(EX,999)
TVaRX<-EX*(1-pgamma(VaRX,alp+1,bet))/(1-vu)
matplot(vu,cbind(TVaRX,VaRX,vEX),type="l",main="Loi gamma",xlab="u",ylab="VaR et TVaR", col=rep(1,3))
```



2.2 Loi binomiale et mesures de risque

Soit $X \sim Bin(r, q)$. Le code R contient des calculs pour $VaR_\kappa(X)$ et $TVaR_\kappa(X)$.

Code R :

```
#
# Loi binomiale
#
qq<-0.0017
bb<-100000
EX<-bb*qq
EX
```

```
## [1] 170
```

```
kap<-0.995
VaRX<-bb*qbinom(kap,1,qq)
VaRX
```

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

```
TVaRX<-EX/(1-kap)
TVaRX
```

```
## [1] 34000
```

```
vn<-c(1,10,100,1000,10000,100000,1000000)
vVaRN<-qbinom(kap,vn,qq)
vVaRN
```

```
## [1]    0    1    2    6   28  204 1807
```

```
vVaRS<-bb*vVaRN
vVaRS
```

```
## [1]         0    100000    200000    600000   2800000  20400000 180700000
```

```
nono<-length(vn)
vTVaRN<-rep(0,nono)
for (i in 1:nono)
{
 vk<-0:vn[i]
 partie1<-sum(vk*dbinom(vk,vn[i],qq)*1*(vk>vVaRN[i]))
 partie2<-vVaRN[i]*(pbinom(vVaRN[i],vn[i],qq)-kap)
 vTVaRN[i]<-(partie1+partie2)/(1-kap)
}
cbind(kap,vVaRN,vTVaRN)
```

```
##        kap vVaRN      vTVaRN
## [1,] 0.995     0    0.340000
## [2,] 0.995     1    1.025892
## [3,] 0.995     2    2.146405
## [4,] 0.995     6    6.463491
## [5,] 0.995    28   30.106109
## [6,] 0.995   204  208.894835
## [7,] 0.995  1807 1820.361010
```

```
vTVaRS<-bb*vTVaRN
round(cbind(vn,vVaRS/vn,vTVaRS/vn,TVaRX-vTVaRS/vn),2)
```

```
##         vn
## [1,] 1e+00     0.0 34000.00     0.00
## [2,] 1e+01 10000.0 10258.92 23741.08
## [3,] 1e+02  2000.0  2146.41 31853.59
## [4,] 1e+03   600.0   646.35 33353.65
## [5,] 1e+04   280.0   301.06 33698.94
## [6,] 1e+05   204.0   208.89 33791.11
## [7,] 1e+06   180.7   182.04 33817.96
```

```r
round(cbind(vn,vVaRS/vn,vTVaRS/vn,TVaRX-vTVaRS/vn),2)
```


```
##         vn
## [1,] 1e+00     0.0 34000.00     0.00
## [2,] 1e+01 10000.0 10258.92 23741.08
## [3,] 1e+02  2000.0  2146.41 31853.59
## [4,] 1e+03   600.0   646.35 33353.65
## [5,] 1e+04   280.0   301.06 33698.94
## [6,] 1e+05   204.0   208.89 33791.11
## [7,] 1e+06   180.7   182.04 33817.96
```

2.3 Mutualisation des risques

Soit $\underline{X}) = (X_1, ..., X_n)$ un vecteur de v.a. i.i.d. On définit

$$S_n = X_1 + ... + X_n$$

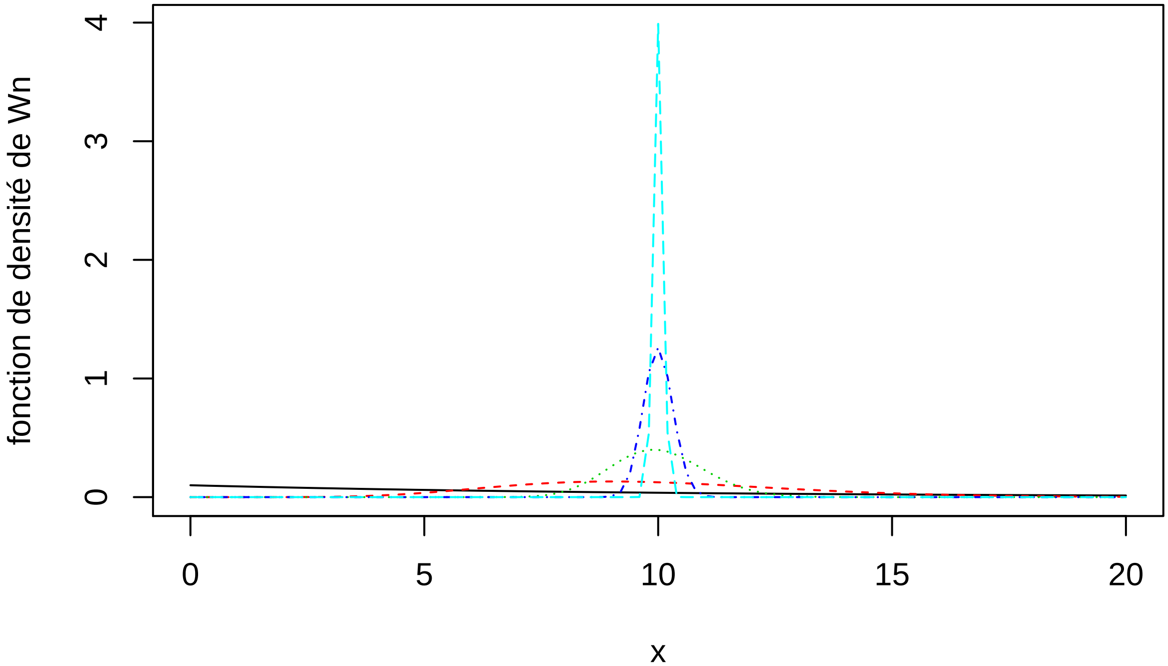
et

$$W_n = \frac{Sn}{n}.$$

On illustre le comportement de W_n.

Code R :

```
# Aggrégation des coûts par contrat
# Loi de Xi : exponentielle(bet)
# nb de contrats : n
# Sn = coûts totaux pour n contrats
# Wn = coûts par contrat pour un ptf de n contrats
# Loi de Sn : gamma(n,bet)
# Loi de Wn : gamma(n,bet*n)
bet<-1/10
vn<-10^(0:4)
vx<-(0:100)/5
matfWn<-matrix(0,101,5)
for (i in (1:5))
{
 matfWn[,i]<-dgamma(vx,vn[i],bet*vn[i])
}
matplot(vx,matfWn,type="l",xlab="x",ylab="fonction de densité de Wn")
```


fonction de densité de Wn
x
0
1
2
3
4
0
5
10
15
20

2.4 Générateur congruentiel linéaire

Soit $U \sim Unif(0,1)$. Le code R contient une illustration du générateur congruentiel linéaire permettant de produire des réalisations $U^{(j)}$ de la v.a. U.

Code R :

```
# générateur de réalisations U(j) de la v.a. U \sim Unif(0,1)
#
aa<-41358
mm<-2^31-1
x0<-2017
nn<-1000000
vx<-rep(0,nn)
vx[1]<-(aa*x0)%%mm
for (i in 2:nn)
{
 vx[i]<-(aa*vx[i-1])%%mm
}
 #cbind(1:nn,vx,vx/mm)
vU<-vx/mm
v1<-vU[1:(nn-1)]
v2<-vU[2:nn]
#plot(v1,v2)
mean(vU)
```

```
## [1] 0.4999013
```

```
mean(qexp(vU))
```

```
## [1] 1.000076
```

```
mean(qgamma(vU,2,1/5))
```

```
## [1] 9.999847
```

```
#
#
```

2.5 Simulation Monte-Carlo

Soit les v.a. indépendantes $X_1 \sim Gamma(\alpha_1, \beta)$ et $X_2 \sim Gamma(\alpha_2, \beta)$. Le code R contient des calculs en lien avec la simulation Monte-Carlo.

Code R :

```
# somme de 2 v.a. indépendantes
#
# loi de X1: gamma(a1,bet)
# loi de X2: gamma(a2,bet)
a1<-2.5
a2<-1.5
bet<-1/10
nsim<-10^6
set.seed(2017)
matU<-matrix(runif(nsim*2),nsim,2,byrow=T)
#matU
X1<-qgamma(matU[,1],a1,bet)
X2<-qgamma(matU[,2],a2,bet)
matX<-cbind(X1,X2)
S<-X1+X2
#cbind(1:nsim,X1,X2,S)
mean(S)
```

```
## [1] 39.98555
```

```
mean(1*(S>50))
```

```
## [1] 0.264783
```

```
quantile(S,c(0.5,0.9),type=1)
```

```
##      50%      90%
## 36.69448 66.81653
```

```
EX1<-a1/bet
EX2<-a2/bet
ES<-EX1+EX2
ES
```

```
## [1] 40
```

```
xx<-50
mean(1*(S>xx))
```

```
## [1] 0.264783
```

```
1-pgamma(xx,a1+a2,bet)
```

```
## [1] 0.2650259
```

```
xx<-100
mean(1*(S>xx))
```

```
## [1] 0.010242
```

```
1-pgamma(xx,a1+a2,bet)
```

```
## [1] 0.01033605
```

```r
kap<-c(0.5,0.9,0.99,0.999)
quantile(S,kap,type=1)
```

```
##       50%       90%       99%     99.9%
##  36.69448  66.81653 100.29979 130.25710
```

```r
qgamma(kap,a1+a2,bet)
```

```
## [1]  36.72061  66.80783 100.45118 130.62241
```

```r
kap1<-0.99999
VaRSapp<-quantile(S,kap1,type=1)
TVaRSapp<-sum(S*1*(S>VaRSapp))/nsim/(1-kap1)
VaRS<-qgamma(kap1,a1+a2,bet)
TVaRS<-ES*(1-pgamma(VaRS,a1+a2+1,bet))/(1-kap1)
c(kap1,VaRSapp,VaRS,TVaRSapp,TVaRS)
```

```
##             99.999%
##   0.99999 183.63744 186.65797 198.15706 198.33697
```

3 Modèles de bases en actuariat non-vie

3.1 Loi Poisson composée avec sinistre individuel de loi gamma

Le code R permet d'effectuer des calculs en lien avec la loi Poisson composée (avec sinistres indivuels de loi gamma) et les mesures de risque.

Code R :

```
# Lyon
# A2017
# Cours Lundi 2017-11-06
# Loi Poisson composée avec sinistre individuel de loi gamma
# Loi de M : Poisson
lambda=0.5
EM<-lambda
VarM<-lambda
# Loi de B : Gamma
alp<-5
bet<-1/200
EB<-alp/bet
VarB<-EB/bet
#
EX<-EM*EB
VarX<-EM*VarB+VarM*(EB^2)
EX
```

```
## [1] 500
```

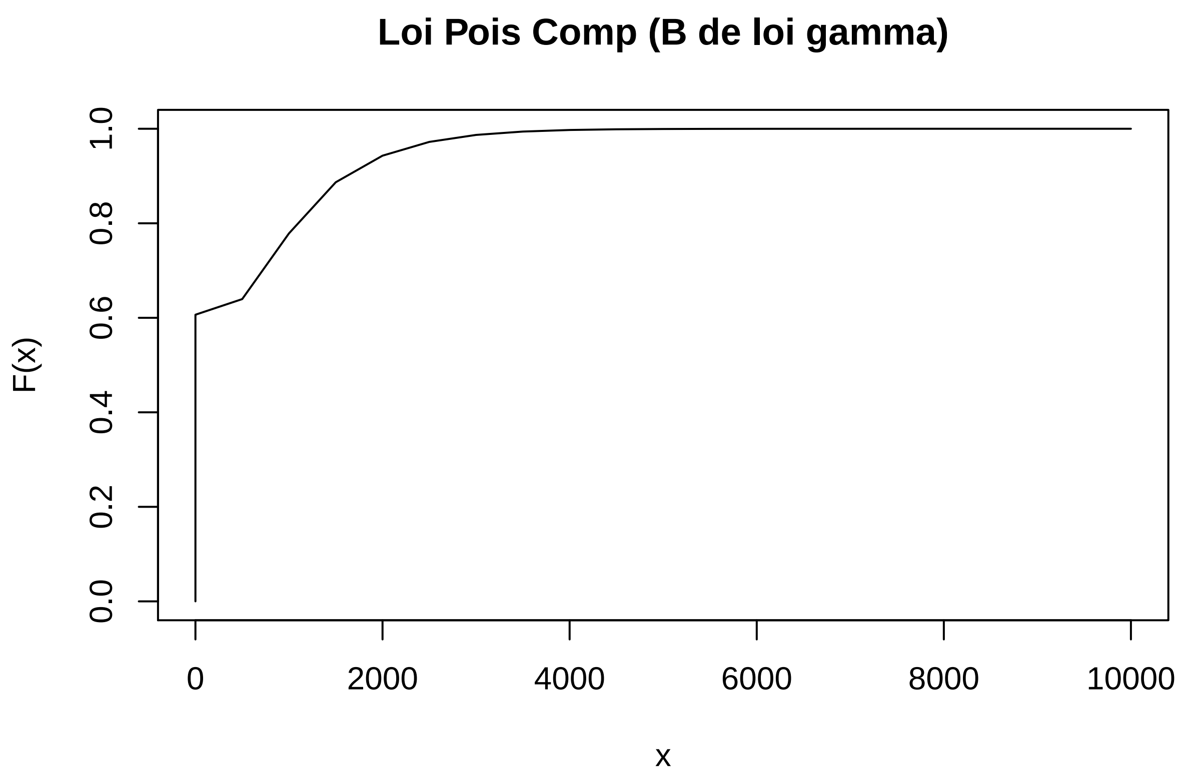
```
VarX
```

```
## [1] 6e+05
```

```
#
# Fonction de répartition de X
#
Fpoisgamma<-function(x,la,aa,bb,kmax=1000)
{
 p0<-dpois(0,la)
 vk<-1:kmax
 pk<-dpois(vk,la)
 vprob<-pgamma(x,aa*vk,bb)
 FX<-p0+sum(pk*vprob)
 return(FX)
}
Fpoisgamma(x=3200,la=lambda,aa=alp,bb=bet,kmax=1000)
```

```
## [1] 0.9905002
```

```
vx<-(0:20)*500
long<-length(vx)
vFx<-rep(0,long)
for(i in 1:long)
{
 vFx[i]<-Fpoisgamma(x=vx[i],la=lambda,aa=alp,bb=bet,kmax=1000)
}
plot(c(0,vx),c(0,vFx),type="l",xlab="x",ylab="F(x)",main="Loi Pois Comp (B de loi gamma)")
```




```
#
# VaR et TVaR
#
# on utilise cette approche pour kappa > F_X(0)
Fpoisgamma(0,la=lambda,aa=alp,bb=bet,kmax=1000)
```

```
## [1] 0.6065307
```

```
kappa<-0.9999
f<-function(x) abs(Fpoisgamma(x,la=lambda,aa=alp,bb=bet,kmax=1000)-kappa)
res<-optimize(f, c(0,10000),tol=0.000000001)
res
```

```
## $minimum
## [1] 5868.077
##
## $objective
## [1] 6.348255e-12
```

```
VaRX<-res$minimum
Fpoisgamma(VaRX,la=lambda,aa=alp,bb=bet,kmax=1000)
```

```
## [1] 0.9999
```

```
TVaRpoisgamma<-function(u,la,aa,bb,kmax=1000,bornes=c(0,10000))
{
 kappa<-u
 f<-function(x) abs(Fpoisgamma(x,la=lambda,aa=alp,bb=bet,kmax=1000)-kappa)
 res<-optimize(f, bornes,tol=0.000000001)
```

```r
 VaR<-res$minimum
 vk<-1:kmax
 pk<-dpois(vk,la)
 vEtronc<-(1-pgamma(VaR,aa*vk+1,bb))*aa/bb*(vk)
 TVaR<-sum(pk*vEtronc)/(1-u)
 return(c(VaR,TVaR))
}
TVaRpoisgamma(0.9999,la=lambda,aa=alp,bb=bet,kmax=1000,bornes=c(0,10000))
```

```
## [1] 5868.077 6407.860
```

3.2 Loi Binomiale négative composée (sinistres de loi lognormale)

Le code R permet d'effectuer des calculs en lien avec la loi binomiale négative composée. la méthode de simulation Monte Carlo et les mesures de risque.

Code R :

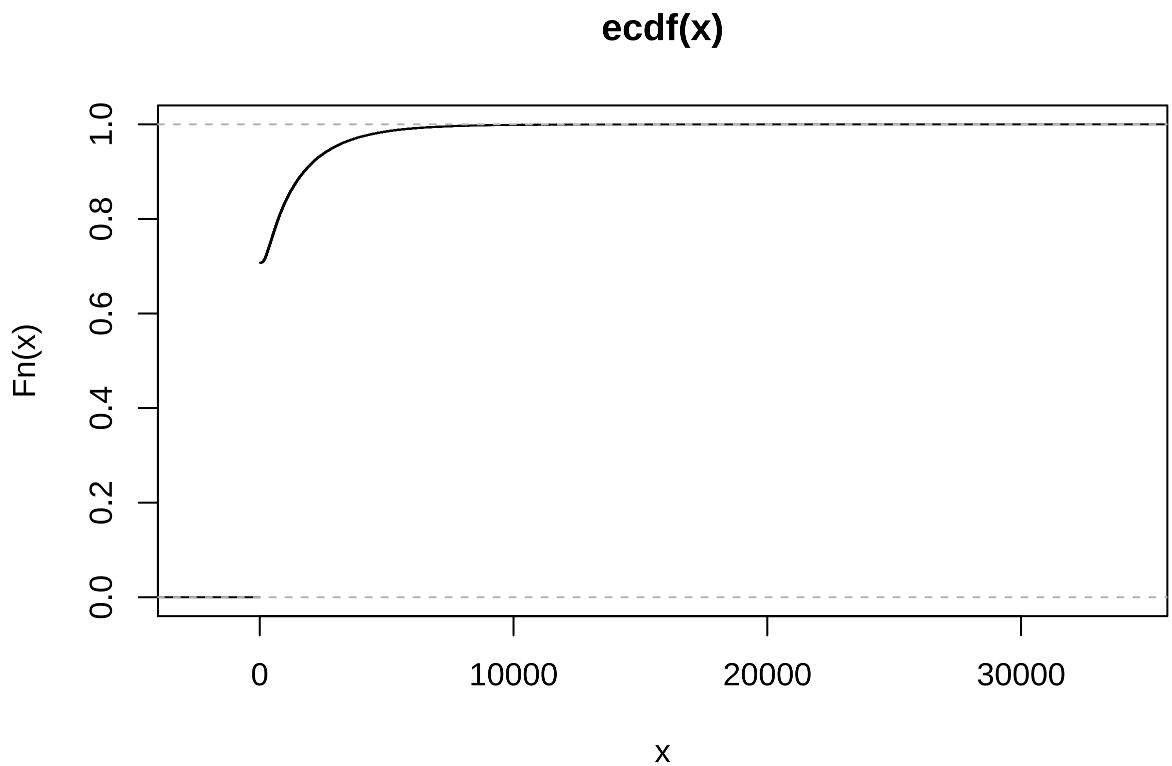
```
# Loi Binomiale négative composée (sinistres de loi lognormale)
# loi de M: binomiale négative
rr<-0.5
qq<-0.5
EM<-rr*(1-qq)/qq
VarM<-EM/qq
# Loi de B : LNormale
mu<-log(1000)-0.32
sig<-0.8
EB<-exp(mu+(sig^2)/2)
EB2<-exp(2*mu+2*(sig^2))
VarB<-EB2-(EB^2)
# X:
EX<-EM*EB
VarX<-EM*VarB+VarM*(EB^2)
EX
```

```
## [1] 500
```

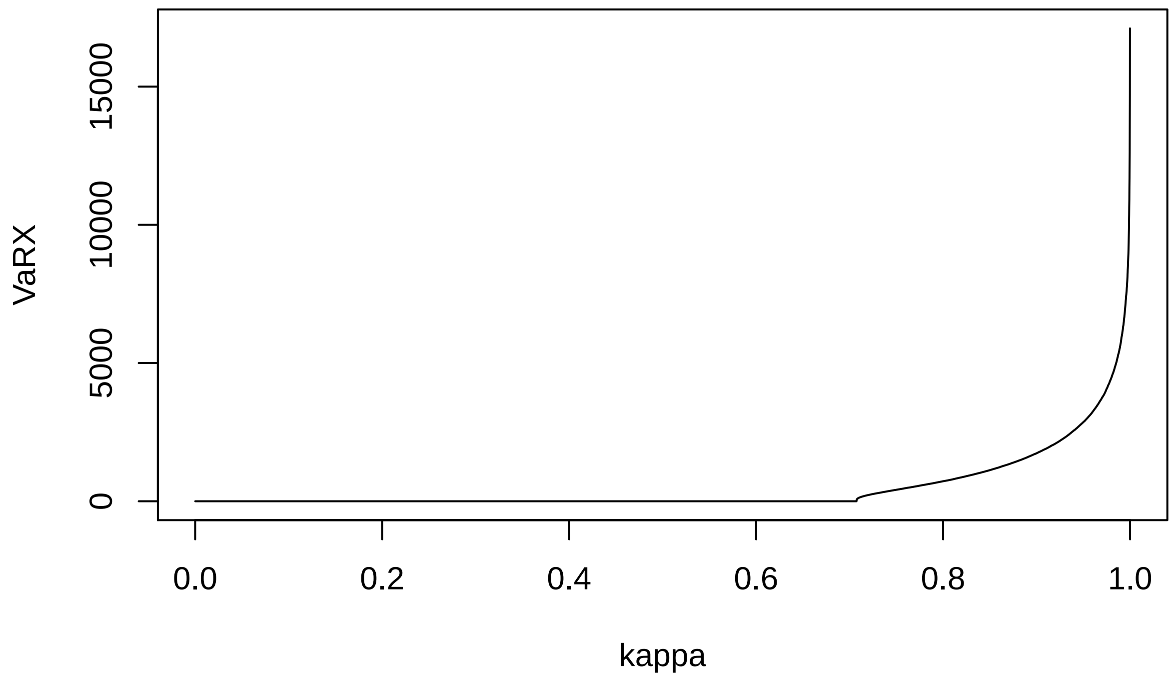
```
VarX
```

```
## [1] 1448240
```

```
# Simulons :
set.seed(2017)
nsim<-100000
vM<-rep(0,nsim)
vX<-rep(0,nsim)
for(i in 1:nsim)
{
 U<-runif(1)
 vM[i]<-qnbinom(U,rr,qq)
 if (vM[i]>0)
 {
 vU<-runif(vM[i])
 vX[i]<-sum(qlnorm(vU,mu,sig))
 }
}
#cbind(vM,vX)
plot.ecdf(vX)
```




```
vkap<-(1:9999)/10000
VaRX<-quantile(vX,prob=vkap,type=1)
plot(vkap,VaRX,type="l",xlab="kappa",ylab="VaRX")
```


VaRX
0
5000
10000
15000
0.0
0.2
0.4
0.6
0.8
1.0
kappa

4 Méthodes d'agrégation

4.1 Algorithme de Panjer

Le code R comporte 1 exercice en lien avec l'algorithme de Panjer.

Code R :

```
# Lyon
# A2017
# Cours Mardi 2017-11-06
#
# Algorithme de Panjer
#
# Fonction
#
panjer.poisson<-function(lam,ff,smax)
{
 aa<-0
 bb<-lam
 ll<-length(ff)
 ffs<-exp(lam*(ff[1]-1))
 ff<-c(ff,rep(0,smax-ll+1))
 for (i in 1 :smax)
 {
 j<-i+1
 ffs<-c(ffs,(1/(1-aa*ff[1]))*sum(ff[2 :j]*ffs[i :1]*(bb*(1 :i)/i+aa)))
 }
 return(ffs)
}
#
# Pareto
ppareto<-function(x,aa,la)
{
 FF<-1-((la/(la+x))^aa)
 return(FF)
}
ppareto(0:10,aa=2,la=5)
```

```
##  [1] 0.0000000 0.3055556 0.4897959 0.6093750 0.6913580 0.7500000 0.7933884
##  [8] 0.8263889 0.8520710 0.8724490 0.8888889
```

```
# Utilisation de l'algo de Panjer
#
# loi de X : Poisson composée
# loi de M : Poisson
# paramètre de la loi de Poisson : lambda
# représentation de X : X = \sum_{k=1}^M B_k
# fmp de B : fB
# fmp de X : fX
#
#
alphaP<-3
lambdaP<-20
vk<-1:10000
```

```r
fB<-c(0,ppareto(vk,aa=alphaP,la=lambdaP)-ppareto(vk-1,aa=alphaP,la=lambdaP))
sum(fB)
```

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

```r
lambda<-2
EM<-lambda
EB<-sum(fB*c(0,vk))
EB2<-sum(fB*c(0,vk^2))
EX<-EM*EB
VarX<-lambda*EB2
EM
```

```
## [1] 2
```

```r
EB
```

```
## [1] 10.51237
```

```r
EX
```

```
## [1] 21.02474
```

```r
VarX
```

```
## [1] 815.911
```

```r
#
fX<-panjer.poisson(lam=lambda,ff=fB,smax=18000)
#
# Vérifications
sum(fX)
```

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

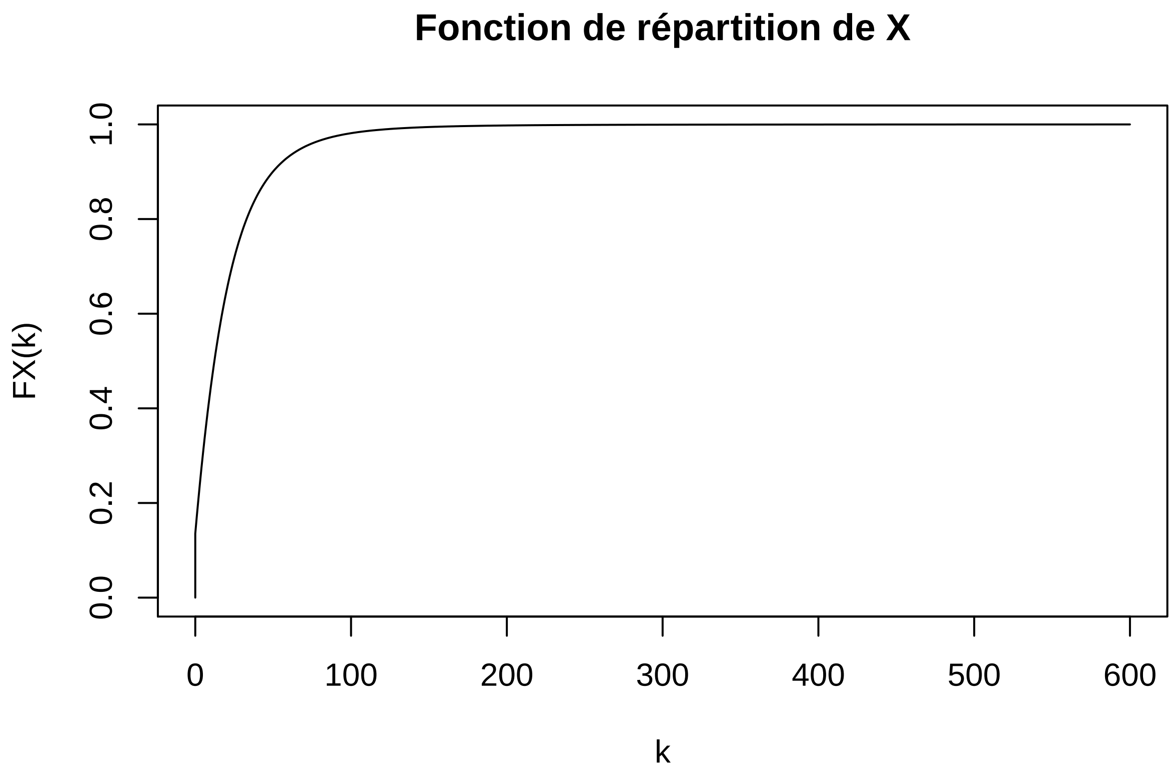
```r
#
EXv<-sum((0:18000)*fX)
EX2v<-sum(((0:18000)^2)*fX)
VarXv<-EX2v-(EXv^2)
c(EX,EXv)
```

```
## [1] 21.02474 21.02474
```

```r
c(VarX,VarXv)
```

```
## [1] 815.911 815.911
```

```r
# FX
FX<-cumsum(fX)
plot(c(0,0:600),c(0,FX[1:601]),type="l",xlab="k",ylab="FX(k)",main="Fonction de répartition de X")
```




```
# prime stop-loss
long<-length(FX)-1
vk<-0:long
k<-100
SL<-sum(pmax(vk-k,0)*fX)
SL
```

```
## [1] 0.9240394
```

```
#
# VaR
#
FX[1:10]
```

```
##  [1] 0.1353353 0.1721904 0.2076653 0.2418035 0.2746370 0.3061935 0.3364993
##  [8] 0.3655814 0.3934683 0.4201904
```

```
kap<-0.00001
kap
```

```
## [1] 1e-05
```

```
VaRX.1<-sum((FX<kap)*1)
VaRX.2<-min(vk[(FX>=kap)])
VaRX.1
```

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

```
VaRX.2
```

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

```r
VaRX<-VaRX.2
EXtron<-sum(vk*fX*(vk>VaRX))
TVaRX<-(EXtron+VaRX*(FX[1+VaRX]-kap))/(1-kap)
c(kap,VaRX,TVaRX)
```

```
## [1]  0.00001  0.00000 21.02495
```

```r
c(EX,VarX)
```

```
## [1]  21.02474 815.91104
```

4.2 FFT

Le code R comporte 3 exercices simples utilisant la méthode FFT.

Code R :

```
# Lyon
# A2017
# Cours Mardi 2017-11-06
#
# FFT = Transformée de Fourier rapide
#
# ------ Exercice de réchauffement - Approche naïve -------
f1<-c(0.3,0.4,0.2,0.1)
nbim<-1i
vk<-0:3
f1
```

```
## [1] 0.3 0.4 0.2 0.1
```

```
sum(f1)
```

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

```
#
f1t<-rep(0,4)
# construction
for (j in 0:3)
{
 f1t[j+1]<-sum(exp(nbim*2*pi*vk*j/4)*f1)
}
f1t
```

```
## [1]  1.000000e+00+0.000000e+00i  1.000000e-01+3.000000e-01i
## [3] -2.775558e-17+3.673819e-17i  1.000000e-01-3.000000e-01i
```

```
f1v<-rep(0,4)
# inversion
for (k in 0:3)
{
 f1v[k+1]<-(1/4)*sum(exp(-nbim*2*pi*vk*k/4)*f1t)
}
Re(f1v)
```

```
## [1] 0.3 0.4 0.2 0.1
```

```
# ----- Exercice pour s'amuser un peu -------
f1<-c(0.3,0.4,0.2,0.1)
f2<-c(0.2,0.5,0.25,0.05)
nn<-8
f1c<-c(f1,rep(0,4))
f2c<-c(f2,rep(0,4))
nbim<-1i
vk<-0:(nn-1)
f1c
```

```
## [1] 0.3 0.4 0.2 0.1 0.0 0.0 0.0 0.0
```

```r
sum(f1c)
```

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

```r
f2c
```

```
## [1] 0.20 0.50 0.25 0.05 0.00 0.00 0.00 0.00
```

```r
sum(f2c)
```

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

```r
#
f1t<-rep(0,nn)
f2t<-rep(0,nn)
# construction
for (j in 0:(nn-1))
{
 f1t[j+1]<-sum(exp(nbim*2*pi*vk*j/nn)*f1c)
 f2t[j+1]<-sum(exp(nbim*2*pi*vk*j/nn)*f2c)
}
fst<-f1t*f2t
cbind(f1t,f2t,fst)
```

```
##                              f1t                   f2t
## [1,]  1.000000e+00+0.000000e+00i  1.0000000+0.0000000i
## [2,]  5.121320e-01+5.535534e-01i  0.5181981+0.6389087i
## [3,]  1.000000e-01+3.000000e-01i -0.0500000+0.4500000i
## [4,]  8.786797e-02+1.535534e-01i -0.1181981+0.1389087i
## [5,] -2.775558e-17+3.673819e-17i -0.1000000+0.0000000i
## [6,]  8.786797e-02-1.535534e-01i -0.1181981-0.1389087i
## [7,]  1.000000e-01-3.000000e-01i -0.0500000-0.4500000i
## [8,]  5.121320e-01-5.535534e-01i  0.5181981-0.6389087i
##                              fst
## [1,]  1.000000e+00+0.000000e+00i
## [2,] -8.828427e-02+6.140559e-01i
## [3,] -1.400000e-01+3.000000e-02i
## [4,] -3.171573e-02-5.944084e-03i
## [5,]  2.775558e-18-3.673819e-18i
## [6,] -3.171573e-02+5.944084e-03i
## [7,] -1.400000e-01-3.000000e-02i
## [8,] -8.828427e-02-6.140559e-01i
```

```r
fsv<-rep(0,nn)
# inversion
for (k in 0:(nn-1))
{
 fsv[k+1]<-(1/nn)*sum(exp(-nbim*2*pi*vk*k/nn)*fst)
}
fs<-rep(0,nn)
for (k in 1:nn)
{
 fs[k]<-sum(f1c[1:k]*f2c[k:1])
}
cbind(0:(nn-1),round(Re(fsv),6),fs)
```

```
##                fs
```

```
## [1,] 0 0.060 0.060
## [2,] 1 0.230 0.230
## [3,] 2 0.315 0.315
## [4,] 3 0.235 0.235
## [5,] 4 0.120 0.120
## [6,] 5 0.035 0.035
## [7,] 6 0.005 0.005
## [8,] 7 0.000 0.000
```

```
2^15
```

```
## [1] 32768
```

```
# ----------- Exercice Poisson composée -----------
#
# Pareto continue
ppareto<-function(x,aa,la)
{
 FF<-1-((la/(la+x))^aa)
 return(FF)
}
ppareto(0:10,aa=2,la=5)
```

```
##  [1] 0.0000000 0.3055556 0.4897959 0.6093750 0.6913580 0.7500000 0.7933884
##  [8] 0.8263889 0.8520710 0.8724490 0.8888889
```

```
#
# loi de X : Poisson composée
# loi de M : Poisson
# paramètre de la loi de Poisson : lambda
# représentation de X : X = \sum_{k=1}^M B_k
# fmp de B : fB
# fmp de X : fX
#
#
alphaP<-3
lambdaP<-20
vk<-1:10000
# définition du vecteur fB (fonction de masses de prob de la v.a. B)
fB<-c(0,ppareto(vk,aa=alphaP,la=lambdaP)-ppareto(vk-1,aa=alphaP,la=lambdaP))
sum(fB)
```

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

```
# paramètre de la loi Poisson
lambda<-2
# calculs
EM<-lambda
EB<-sum(fB*c(0,vk))
EB2<-sum(fB*c(0,vk^2))
EX<-EM*EB
VarX<-lambda*EB2
EM
```

```
## [1] 2
```

```
EB
```

```
## [1] 10.51237
```

```
EX
```

```
## [1] 21.02474
```

```
VarX
```

```
## [1] 815.911
```

```
#
# - On utilise FFT
nn<-2^15
nn
```

```
## [1] 32768
```

```
long<-length(fB)
# On ajoute des "0"
fBc<-c(fB,rep(0,nn-long))
# On utilise fft pour calculer les valeurs de la fn caractéristique de B
fBt<-fft(fBc)
# on calculer les valeurs de la fn caractéristique de X
fXt<-exp(lambda*(fBt-1))
# on inverse avec fft pour calculer les valeurs de fX
fX<-Re(fft(fXt,inverse=TRUE)/nn)
#
# Vérifications
sum(fX)
```

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

```
#
EXv<-sum((0:18000)*fX)
```

```
## Warning in (0:18000) * fX: la taille d'un objet plus long n'est pas
## multiple de la taille d'un objet plus court
```

```
EX2v<-sum(((0:18000)^2)*fX)
```

```
## Warning in ((0:18000)^2) * fX: la taille d'un objet plus long n'est pas
## multiple de la taille d'un objet plus court
```

```
VarXv<-EX2v-(EXv^2)
c(EX,EXv)
```

```
## [1] 21.02474 21.02474
```

```
c(VarX,VarXv)
```

```
## [1] 815.911 815.911
```

5 Lois multivariées discrètes et composées

5.1 Loi Poisson bivariée de Teicher

Soit une paire de v.a. (M_1, M_2) avec

$$\mathcal{P}_{M_1,M_2}(t_1,t_2) = ^{(\lambda_1-\alpha_0)(t_1-1)} \mathrm{e}^{(\lambda_2-\alpha_0)(t_2-1)}\mathrm{e}^{\alpha_0(t_1t_2-1)},\ |t_i| \leq 1,\ i=1,2.$$

On définit $N = M_1 + M_2$.

On déduit

$$\mathcal{P}_N(t) = \mathcal{P}_{M_1,M_2}(t,t),\ |t| \leq 1,$$

$$\phi_N(t) = \mathrm{e}^{(\lambda_1-\alpha_0)(\mathrm{e}^{it}-1)}\mathrm{e}^{(\lambda_2-\alpha_0)(\mathrm{e}^{it}-1)}\mathrm{e}^{\alpha_0(\mathrm{e}^{it\times 2}-1)} = \mathcal{P}_{M_1,M_2}(\phi_B(t),\phi_B(t)),$$

où $\phi_B(t) = \mathrm{e}^{it}$.

Objectif : Calculer $\Pr(N=k)$, $k \in \mathbb{N}$, avec Panjer et FFT.

Code R :

```
# Lyon
# A2017
# Cours Mercredi 2017-11-08
#
# But : calculer Pr(N=k) où N=M1+M2
# (M1,M2) obéit à une loi Poisson bivariée Teicher
# 2 options : FFt ou Panjer
# important : les valeurs calculées sont exactes
#
# Algorithme de Panjer
#
# Fonction
#
panjer.poisson<-function(lam,ff,smax)
{
 aa<-0
 bb<-lam
 ll<-length(ff)
 ffs<-exp(lam*(ff[1]-1))
 ff<-c(ff,rep(0,smax-ll+1))
 for (i in 1 :smax)
 {
 j<-i+1
 ffs<-c(ffs,(1/(1-aa*ff[1]))*sum(ff[2 :j]*ffs[i :1]*(bb*(1 :i)/i+aa)))
 }
 return(ffs)
}
#
# Loi Poisson Bivariée Teicher
#
la1<-2
la2<-3
al0<-1
```

```r
mm<-2^10
EN<-la1+la2
CovM1M2<-al0
VarN<-la1+la2+2*CovM1M2
# option #1 : FFT
fB<-rep(0,mm)
fB[2]<-1
fBt<-fft(fB)
fNt<-exp((la1-al0)*(fBt-1))*exp((la2-al0)*(fBt-1))*exp(al0*(fBt^2-1))
fN<-Re(fft(fNt,inverse=TRUE)/mm)
sum(fN)
```

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

```r
vk<-0:(mm-1)
ENv<-sum(vk*fN)
EN2v<-sum((vk^2)*fN)
VarNv<-EN2v-(ENv^2)
c(EN,ENv)
```

```
## [1] 5 5
```

```r
c(VarN,VarNv)
```

```
## [1] 7 7
```

```r
# option #2 : Panjer
laN<-la1+la2-al0
fC1<-(la1+la2-2*al0)/laN
fC2<-al0/laN
fC<-c(0,fC1,fC2)
fNpanjer<-panjer.poisson(lam=laN,ff=fC,smax=1000)
sum(fNpanjer)
```

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

```r
vk<-0:(mm-1)
ENw<-sum(vk*fN)
EN2w<-sum((vk^2)*fN)
VarNw<-EN2w-(ENw^2)
c(EN,ENv,ENw)
```

```
## [1] 5 5 5
```

```r
c(VarN,VarNv,VarNw)
```

```
## [1] 7 7 7
```

```r
round(cbind(0:30,fN[1:31],fNpanjer[1:31]),6)
```

```
##       [,1]     [,2]     [,3]
##  [1,]    0 0.018316 0.018316
##  [2,]    1 0.054947 0.054947
##  [3,]    2 0.100736 0.100736
##  [4,]    3 0.137367 0.137367
##  [5,]    4 0.153393 0.153393
##  [6,]    5 0.146983 0.146983
##  [7,]    6 0.124623 0.124623
##  [8,]    7 0.095405 0.095405
```

```
##  [9,]    8 0.066932 0.066932
## [10,]    9 0.043512 0.043512
## [11,]   10 0.026440 0.026440
## [12,]   11 0.015122 0.015122
## [13,]   12 0.008187 0.008187
## [14,]   13 0.004216 0.004216
## [15,]   14 0.002073 0.002073
## [16,]   15 0.000977 0.000977
## [17,]   16 0.000442 0.000442
## [18,]   17 0.000193 0.000193
## [19,]   18 0.000081 0.000081
## [20,]   19 0.000033 0.000033
## [21,]   20 0.000013 0.000013
## [22,]   21 0.000005 0.000005
## [23,]   22 0.000002 0.000002
## [24,]   23 0.000001 0.000001
## [25,]   24 0.000000 0.000000
## [26,]   25 0.000000 0.000000
## [27,]   26 0.000000 0.000000
## [28,]   27 0.000000 0.000000
## [29,]   28 0.000000 0.000000
## [30,]   29 0.000000 0.000000
## [31,]   30 0.000000 0.000000
```

5.2 Loi composée bivariée

Soit une paire de v.a. (M_1, M_2) avec

$$\mathcal{P}_{M_1,M_2}(t_1,t_2) = (p_{00} + p_{10}t_1 + p_{01}t_2 + p_{11}t_1t_2)^n,\ |t_i| \leq 1, i = 1, 2.$$

On définit une paire de v.a. v.a. (X_1, X_2) avec

$$X_1 = \sum_{k_1=1}^{M_1} B_{1,k_1} \text{ et } X_2 = \sum_{k_2=1}^{M_2} B_{2,k_2}$$

où

- $\underline{B}_1 = \{B_{1,k_1}, k_1 \in \mathbb{N}^+\}$ forme une suite de v.a. i.i.d. avec $B_{1,k_1} \sim B_1$, $k_1 \in \mathbb{N}^+$;
- $\underline{B}_2 = \{B_{2,k_2}, k_2 \in \mathbb{N}^+\}$ forme une suite de v.a. i.i.d. avec $B_{2,k_2} \sim B_2$, $k_2 \in \mathbb{N}^+$;
- $\underline{B}_1$, $\underline{B}_2$ et (M_1, M_2) sont indépendantes ;
- $B_1 \sim Pois(\lambda)$ et $B_2 \sim BNég(r, q)$.

On définit $S = X_1 + X_2$.

On déduit

$$\mathcal{P}_S(t) = \mathcal{P}_{X_1,X_2}(t,t) = \mathcal{P}_{M_1,M_2}(\mathcal{P}_{B_1}(t), \mathcal{P}_{B_2}(t)),\ |t| \leq 1,$$

et

$$\phi_S(t) = \phi_{X_1,X_2}(t,t) = \mathcal{P}_{M_1,M_2}(\phi_{B_1}(t), \phi_{B_2}(t)).$$

Expression de la f.g.p. de (X_1, X_2) :

$$\mathcal{P}_{X_1,X_2}(t_1,t_2) = \mathcal{P}_{M_1,M_2}(\mathcal{P}_{B_1}(t_1), \mathcal{P}_{B_2}(t_2)),\ |t_i| \leq 1,\ i = 1, 2.$$

Expression de la f.c. de (X_1, X_2) :

$$\phi_{X_1,X_2}(t_1,t_2) = \mathcal{P}_{M_1,M_2}(\phi_{B_1}(t_1), \phi_{B_2}(t_2)).$$

Objectif : Calculer $\Pr(S = k)$, $k \in \mathbb{N}$, avec FFT.

Code R :

```r
# Lyon
# A2017
# Cours Mercredi 2017-11-08
#
# Loi de (X_1,X_2) : binomiale composée bivariée
# Loi de B1 : discrètes
# Loi de B2 : discrètes
# S = X_1 + X_2
# But : calculer Pr(S=k) où N=M1+M2
# 1 option présentée : FFt
# important : les valeurs calculées sont exactes
#
p00<-0.7
p10<-0.15
p01<-0.05
p11<-0.1
p00+p01+p10+p11
```

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

```r
nn<-10
q1<-p10+p11
q2<-p01+p11
q1
```

```
## [1] 0.25
```

```r
q2
```

```
## [1] 0.15
```

```r
EM1<-nn*q1
EM2<-nn*q2
mm<-2^10
vk<-0:(mm-1)
fB1<-dpois(vk,2)
fB2<-dnbinom(vk,1.5,1/3)
EB1<-2
EB2<-1.5*(1-1/3)/(1/3)
EX1<-EM1*EB1
EX2<-EM2*EB2
ES<-EX1+EX2
ES
```

```
## [1] 9.5
```

```r
fB1t<-fft(fB1)
fB2t<-fft(fB2)
fSt<-(p00+p10*fB1t+p01*fB2t+p11*fB1t*fB2t)^nn
fS<-Re(fft(fSt,inverse=TRUE)/mm)
sum(fS)
```

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

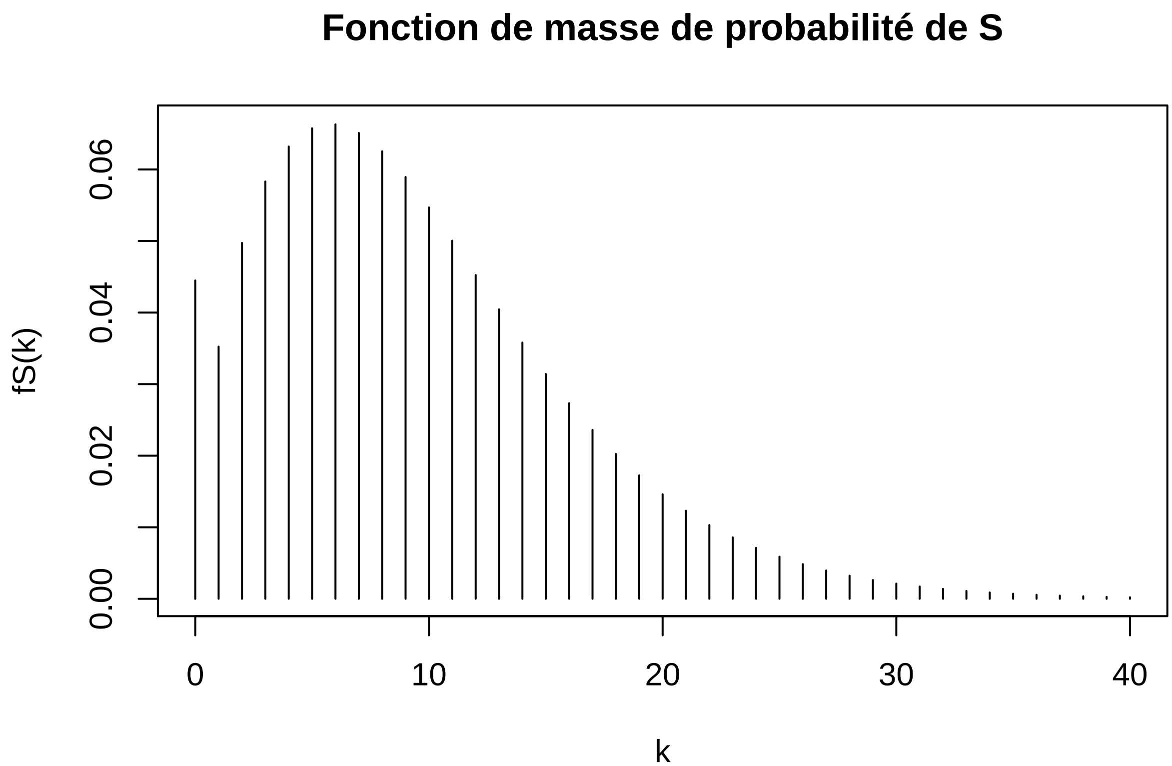
```r
ESv<-sum(vk*fS)
ES
```

```
## [1] 9.5
```

```r
ESv
```

```
## [1] 9.5
```

```r
plot(0:40,fS[1:41],type="h",xlab="k",ylab="fS(k)",main="Fonction de masse de probabilité de S")
```


Fonction de masse de probabilité de S
fS(k)
0.00
0.02
0.04
0.06
0
10
20
30
40
k

6 Remerciements

Merci à Christopher Blier-Wong et Simon-Pierre Gadoury pour le disponibilité et leur motivation.

Merci aux étudiantes et aux étudiants de notre laboratoire ACT&RISK pour leur collaboration.

Merci aux étudiants de mes cours (Université Laval, Université Lyon 1, McGill University) pour leur participation et leur patience.

Références

Marceau, Etienne. 2013. *Modélisation et évaluation Quantitative Des Risques En Actuariat: Modèles Sur Une Période.* Springer.