GAL Buckle 95

François Pelletier

24 mai 2014

Table des matières

Chapitre 1

Initialisation

1.1 Chargement des paquets

```
> setwd("~/maitrise/GAL-Buckle95/")
> library(actuar)
> library(MASS)
> library(xtable)
> library(parallel)
> library(moments)
> library(FourierStuff)
> library(GeneralizedAsymmetricLaplace)
> library(GMMStuff)
> library(OptionPricingStuff)
> library(QuadraticEstimatingEquations)
```

1.2 Constantes et données

```
> #Nombre de décimales affichées
> options(digits=6)
> #Marge pour intervalles de confiance
> alpha.confint <- 0.05
> #Marge pour test d'hypothèses
> alpha.test <- 0.05
> #Chargement des données
> RETURNS <- head(read.csv("abbeyn.csv",sep="\t",header=TRUE)[,1],-1)
> #Taille de l'échantillon
> n <- length(RETURNS)
> #Nom de l'échantillon
> strData <- "Buckle95"
```

1.3 Test de normalité

```
> EppsPulley.test(RETURNS)

Epps-Pulley Normality test

 T: 0.626033
 T*: 0.635568
p-value: 0.007178


$Tstat
[1] 0.626033


$Tmod
[1] 0.635568
```

```
$Zscore
[1] 2.44824

$Pvalue
[1] 0.00717788

$Reject
[1] TRUE
```

Chapitre 2

Estimation

2.1 Données mises à l'échelle

```
> sRET <- as.vector(scale(RETURNS))
```

2.2 Première estimation par QEE

```
> ## Point de départ
> pt.depart <- startparamGAL(sRET)
> ## Fonctions pour les moments
> meanQEE <- function(param) mGAL(param,1)
> varianceQEE <- function(param) cmGAL(param,2)
> sdQEE <- function(param) sqrt(cmGAL(param,2))
> skewnessQEE <- function(param) cmGAL(param,3)
> kurtosisQEE <- function(param) cmGAL(param,4)
> ## Fonctions pour les dérivées
> dmeanQEE <- function(param) dmGAL(param,1)
> dsdQEE <- function(param) dmGAL(param,2)
> ## Estimation gaussienne
> optim1 <- optim(pt.depart,obj.gauss,gr=NULL,sRET,
+                 meanQEE,varianceQEE,dmeanQEE,dsdQEE)
> pt.optim1 <- optim1$par
> ## Estimation de crowder
> optim2 <- optim(pt.depart,obj.Crowder,gr=NULL,sRET,
+                 meanQEE,varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE)
> pt.optim2 <- optim2$par
> ## Estimation de crowder modifiée
> optim3 <- optim(pt.depart,obj.Crowder.Mod,gr=NULL,sRET,
+                 meanQEE,varianceQEE,dmeanQEE,dsdQEE)
> pt.optim3 <- optim3$par
```

2.3 Résultats de la première estimation par QEE

```
> cov.optim1 <- covariance.QEE(M.gauss(pt.optim1,sRET,
+                             meanQEE,varianceQEE,dmeanQEE,dsdQEE),
+               V.gauss(pt.optim1,sRET,meanQEE,varianceQEE,
+                             skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),n)
> cov.optim2 <- covariance.QEE(M.Crowder(pt.optim2,sRET,
+                             varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),
+               V.Crowder(pt.optim2,sRET,varianceQEE,
+                             skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),n)
> cov.optim3 <- covariance.QEE(M.Crowder.Mod(pt.optim3,sRET,
+                             varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),
+               V.Crowder.Mod(pt.optim3,sRET,varianceQEE,dmeanQEE,dsdQEE),n)
> confidence.interval.QEE(pt.optim1,cov.optim1,n)
```

```
          LOWER   ESTIMATE      UPPER
[1,] -0.780018 -0.726048 -0.672077
[2,]  0.436002  0.596316  0.756630
[3,]  0.262650  0.359186  0.455722
[4,]  1.994757  2.021370  2.047982


> confidence.interval.QEE(pt.optim2,cov.optim2,n)

          LOWER   ESTIMATE      UPPER
[1,] -0.694457 -0.627404 -0.560351
[2,]  0.413764  0.640292  0.866820
[3,]  0.232650  0.334028  0.435405
[4,]  1.839966  1.878296  1.916626


> confidence.interval.QEE(pt.optim3,cov.optim3,n)

          LOWER   ESTIMATE      UPPER
[1,] -0.765288 -0.711439 -0.657589
[2,]  0.455485  0.606642  0.757798
[3,]  0.264669  0.362932  0.461195
[4,]  1.932691  1.960299  1.987906
```

2.4   Seconde estimation par QEE

```
> ## Estimation gaussienne
> optim4 <- optim(pt.optim1,obj.gauss,gr=NULL,sRET,
+                 meanQEE,varianceQEE,dmeanQEE,dsdQEE,
+                 ginv(V.gauss(pt.optim1,sRET,meanQEE,
+                                         varianceQEE,skewnessQEE,kurtosisQEE,
+                                         dmeanQEE,dsdQEE)))
> pt.optim4 <- optim4$par
> ## Estimation de crowder
> optim5 <- optim(pt.optim2,obj.Crowder,gr=NULL,sRET,
+                 meanQEE,varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE,
+                 ginv(V.Crowder(pt.optim2,sRET,varianceQEE,skewnessQEE,
+                                         kurtosisQEE,dmeanQEE,dsdQEE)))
> pt.optim5 <- optim5$par
> ## Estimation de crowder modifiée
> optim6 <- optim(pt.optim3,obj.Crowder.Mod,gr=NULL,sRET,
+                 meanQEE,varianceQEE,dmeanQEE,dsdQEE,
+                 ginv(V.Crowder.Mod(pt.optim3,sRET,varianceQEE,
+                                         dmeanQEE,dsdQEE)))
> pt.optim6 <- optim6$par
```

2.5   Résultats de la seconde estimation par QEE

```
> cov.optim4 <- covariance.QEE(M.gauss(pt.optim4,sRET,
+                              meanQEE,varianceQEE,dmeanQEE,dsdQEE),
+               V.gauss(pt.optim4,sRET,meanQEE,varianceQEE,
+                              skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),n)
> cov.optim5 <- covariance.QEE(M.Crowder(pt.optim5,sRET,
+                              varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),
+               V.Crowder(pt.optim5,sRET,varianceQEE,skewnessQEE,
+                              kurtosisQEE,dmeanQEE,dsdQEE),n)
> cov.optim6 <- covariance.QEE(M.Crowder.Mod(pt.optim6,sRET,
+                              varianceQEE,skewnessQEE,kurtosisQEE,dmeanQEE,dsdQEE),
+               V.Crowder.Mod(pt.optim6,sRET,varianceQEE,dmeanQEE,dsdQEE),n)
> confidence.interval.QEE(pt.optim4,cov.optim4,n)
```

```
         LOWER  ESTIMATE     UPPER
[1,] -0.779792 -0.725853 -0.671914
[2,]  0.436017  0.596319  0.756622
[3,]  0.262456  0.358969  0.455482
[4,]  1.995452  2.022048  2.048644

> confidence.interval.QEE(pt.optim5,cov.optim5,n)

         LOWER  ESTIMATE     UPPER
[1,] -0.692712 -0.625874 -0.559036
[2,]  0.414139  0.640445  0.866750
[3,]  0.231568  0.332845  0.434122
[4,]  1.842116  1.880376  1.918636

> confidence.interval.QEE(pt.optim6,cov.optim6,n)

         LOWER  ESTIMATE     UPPER
[1,] -0.766288 -0.712450 -0.658612
[2,]  0.455051  0.606193  0.757334
[3,]  0.264972  0.363196  0.461419
[4,]  1.934050  1.961614  1.989178
```

2.6   Estimation par GMM

```
> ## GMM régulier
> optim7 <- optim.GMM(pt.depart,
+                 conditions.vector=meanvariance.gmm.vector,
+                 data=sRET,W=diag(2),
+                 meanf=meanQEE,variancef=varianceQEE)
> pt.optim7 <- optim7$par
> cov.optim7 <- mean.variance.GMM.gradient.GAL(pt.optim7,sRET) %*%
+                 covariance.GMM(pt.optim7,meanvariance.gmm.vector,sRET,
+                            meanf=meanQEE,variancef=varianceQEE) %*%
+                 t(mean.variance.GMM.gradient.GAL(pt.optim7,sRET)) / n
> ## GMM itératif
> optim8 <- iterative.GMM(pt.depart,
+                 conditions.vector=meanvariance.gmm.vector,
+                 data=sRET,W=diag(2),
+                 meanf=meanQEE,variancef=varianceQEE)
> pt.optim8 <- optim8$par
> cov.optim8 <-  mean.variance.GMM.gradient.GAL(pt.optim8,sRET) %*%
+                 optim8$cov %*%
+                 t(mean.variance.GMM.gradient.GAL(pt.optim8,sRET)) / n
> confidence.interval.QEE(pt.optim7,cov.optim7,n)

         LOWER  ESTIMATE     UPPER
[1,] -0.878702 -0.641646 -0.404589
[2,] -0.469225  0.625908  1.721040
[3,] -0.192234  0.326366  0.844965
[4,]  1.696121  1.965995  2.235869

> confidence.interval.QEE(pt.optim8,cov.optim8,n)

         LOWER  ESTIMATE     UPPER
[1,] -0.874031 -0.636980 -0.399929
[2,] -0.473292  0.626346  1.725984
[3,] -0.193600  0.322895  0.839390
[4,]  1.704166  1.972716  2.241265
```

Chapitre 3

Comparaison des résultats

```
> # Aggrégation des estimateurs (pour simplifier les calculs)
> pts.estim <- cbind(pt.optim1,pt.optim2,pt.optim3,pt.optim4,
+                    pt.optim5,pt.optim6,pt.optim7,pt.optim8)
> l.pts.estim <- as.list(data.frame(pts.estim))
```

3.1 Fonction de répartition

```
> # Points d'évaluation
> xi <- seq(2*min(sRET),2*max(sRET),length.out=2^6)
> # Fonction de répartition par intégration de la fonction caractéristique
> dist1 <- cbind(cftocdf(xi,cfGAL,param=pt.optim1),
+                cftocdf(xi,cfGAL,param=pt.optim2),
+                cftocdf(xi,cfGAL,param=pt.optim3),
+                cftocdf(xi,cfGAL,param=pt.optim4),
+                cftocdf(xi,cfGAL,param=pt.optim5),
+                cftocdf(xi,cfGAL,param=pt.optim6),
+                cftocdf(xi,cfGAL,param=pt.optim7),
+                cftocdf(xi,cfGAL,param=pt.optim8))
> # Fonction de répartition par point de selle
> dist2 <- cbind(psaddleapproxGAL(xi,pt.optim1),
+                psaddleapproxGAL(xi,pt.optim2),
+                psaddleapproxGAL(xi,pt.optim3),
+                psaddleapproxGAL(xi,pt.optim4),
+                psaddleapproxGAL(xi,pt.optim5),
+                psaddleapproxGAL(xi,pt.optim6),
+                psaddleapproxGAL(xi,pt.optim7),
+                psaddleapproxGAL(xi,pt.optim8))
> # Fonction de répartition par intégration de la fonction de densité
> dist3 <- cbind(pGAL(xi,pt.optim1),
+                pGAL(xi,pt.optim2),
+                pGAL(xi,pt.optim3),
+                pGAL(xi,pt.optim4),
+                pGAL(xi,pt.optim5),
+                pGAL(xi,pt.optim6),
+                pGAL(xi,pt.optim7),
+                pGAL(xi,pt.optim8))
```

3.1.1 Graphiques

```
> for (i in 1:8)
+ {
+        file<-paste(strData,"-repart-",i,".pdf",sep="")
+        pdf(file=file,paper="special",width=6,height=6)
+        plot.ecdf(sRET,main=paste("Fonction de répartition ",i))
+        lines(xi,dist1[,i],col="green")
+        lines(xi,dist2[,1],col="red")
+        lines(xi,dist3[,1],col="pink")
+        lines(xi,pnorm(xi),type="l",col="blue")
+        dev.off()
+        cat("\\includegraphics[height=4in,width=4in]{",
+                        file,"}\n",sep="")
+ }
```

Buckle95-repart-1.pdf	Buckle95-repart-2.pdf

Buckle95-repart-3.pdf

Buckle95-repart-4.pdf

Buckle95-repart-5.pdf

Buckle95-repart-6.pdf

Buckle95-repart-7.pdf	Buckle95-repart-8.pdf

3.1.2 Statistiques

Test du χ^2, Méthode avec intégration

```
> chisquare.test1 <- function(param,DATA.hist,FUN,method)
+ {
+           chisquare.test(DATA.hist,FUN,param,method=method)
+ }
> xtable(do.call(rbind,lapply(l.pts.estim,chisquare.test1,hist(sRET),
+                                              cfGAL,"integral")),digits=6)
```

	chisquare.stat	df	p.value
pt.optim1	5.473824	6.000000	0.484626
pt.optim2	5.329673	6.000000	0.502277
pt.optim3	5.388158	6.000000	0.495076
pt.optim4	5.474310	6.000000	0.484567
pt.optim5	5.337004	6.000000	0.501372
pt.optim6	5.390662	6.000000	0.494769
pt.optim7	5.454256	6.000000	0.487003
pt.optim8	5.476963	6.000000	0.484245

Test du χ^2, Méthode avec point de selle

```
> xtable(do.call(rbind,lapply(l.pts.estim,chisquare.test1,hist(sRET),
+                                              pGAL,"saddlepoint")),digits=6)
```

Statistique de Kolmogorov-Smirnov

```
> ks.test1 <- function(param,x,y) ks.test(x,y,param)
> xtable(do.call(rbind,mclapply(l.pts.estim,ks.test1,sRET,"pGAL")),digits=6)
```

Statistique de distance minimale

```
> tvariate1 <- seq(-.1,.1,by=0.01)
> xtable(do.call(rbind,mclapply(l.pts.estim,
+                                              md.test,sRET,tvariate1,cfGAL,empCF)),digits=6)
```

	chisquare.stat	df	p.value
pt.optim1	9.293574	6.000000	0.157728
pt.optim2	8.345592	6.000000	0.213862
pt.optim3	9.050625	6.000000	0.170751
pt.optim4	9.292836	6.000000	0.157767
pt.optim5	8.344140	6.000000	0.213959
pt.optim6	9.062381	6.000000	0.170100
pt.optim7	8.616379	6.000000	0.196330
pt.optim8	8.610490	6.000000	0.196698

	statistic	p.value	alternative	method	data.name
pt.optim1	0.158220	0.171912	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim2	0.140346	0.289345	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim3	0.156772	0.179751	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim4	0.158159	0.172235	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim5	0.139916	0.292753	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim6	0.156960	0.178718	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim7	0.141230	0.282437	two-sided	One-sample Kolmogorov-Smirnov test	x
pt.optim8	0.140016	0.291954	two-sided	One-sample Kolmogorov-Smirnov test	x

3.2   Fonction de densité

Intégration de la fonction de densité approximée avec le point de selle, pour la normaliser en fonction qui intègre à 1.

```
> f_integrale_saddle <- function(param,f,lower,upper)
+         integrate(f,lower,upper,param)$value
> norm_int_saddle <- sapply(l.pts.estim,f_integrale_saddle,
+              f=dsaddleapproxGAL,lower=-Inf,upper=Inf)
```

Séquence de points pour les graphiques

```
> x_sRET <- seq(min(sRET)-sd(sRET),max(sRET)+sd(sRET),length.out=50)
```

Graphique de la fonction de densité

```
> colors2=c("black","red","green","blue","grey")
> for (i in 1:dim(pts.estim)[2])
+ {
+         file=paste(strData,"-densite-", i, ".pdf", sep="")
+         pdf(file=file, paper="special", width=6, height=6)
+         plot(density(sRET),ylim=c(0,.7),type="l",
+                     main=paste("Densité de",strData, i),xlab=strData,
+                     ylab="f",lwd=2,lty=1)
+         points(x_sRET,
+                     dGAL(x_sRET,pts.estim[,i]),
+                     type="b",ylim=c(0,4),col="red",pch=19,lwd=2,lty=2)
+         points(x_sRET,
+                     dsaddleapproxGAL(x_sRET,pts.estim[,i])/norm_int_saddle[i],
+                     type="b",ylim=c(0,4),col="green",pch=20,lwd=2,lty=3)
+
+         lines(x_sRET,dnorm(x_sRET),type="b",col="blue",
+                     pch=21,lwd=2,lty=4)
+         points(seq(-2,4,length.out=1000)[seq(40,1000,by=40)],
+                     cftodensity.fft(cfGAL,1000,-2,4,pts.estim[,i])$dens[seq(40,1000,by=40)],
+                     type="b",col="grey",pch=23,lty=6)
+         legend(quantile(sRET,0.9),0.7, c("emp","est.GAL","pt.selle","appx.nrm","fft"),
+                     cex=0.8, col=colors2, pch=c(NA,19:23), lty=1:6, title="Courbes")
+         dev.off()
+         cat("\\includegraphics[height=4in, width=4in]{"
+                     ,file, "}\n", sep="")
+ }
```

	md.stat	df	p.value
pt.optim1	0.000422	21.000000	0.000000
pt.optim2	0.120174	21.000000	0.000000
pt.optim3	0.001384	21.000000	0.000000
pt.optim4	0.000388	21.000000	0.000000
pt.optim5	0.123295	21.000000	0.000000
pt.optim6	0.001451	21.000000	0.000000
pt.optim7	0.007980	21.000000	0.000000
pt.optim8	0.010416	21.000000	0.000000

Buckle95-densite-1.pdf

Buckle95-densite-2.pdf

Buckle95-densite-3.pdf

Buckle95-densite-4.pdf

Buckle95-densite-5.pdf	Buckle95-densite-6.pdf
Buckle95-densite-7.pdf	Buckle95-densite-8.pdf

3.2.1 Tests avec contraintes

Test de Wald

```
> R <- matrix(c(0,0,1,0,
+                              0,0,0,1),ncol=4)
> r <- matrix(c(0,0),ncol=1)
> V <- lapply(l.pts.estim,covariance.GMM,meanvariance.gmm.vector,
+             sRET,meanQEE,varianceQEE)
> D <- lapply(l.pts.estim,mean.variance.GMM.gradient.GAL,sRET)
> xtable(mapply(Wald.Test,l.pts.estim,n,list(R),list(r),V,D),
```

```
+                 caption="Test de Wald", digits=2)

% latex table generated in R 3.1.0 by xtable 1.7-3 package
% Sat May 24 10:56:20 2014
\begin{table}[ht]
\centering
\begin{tabular}{rrrrrrrrr}
  \hline
 & pt.optim1 & pt.optim2 & pt.optim3 & pt.optim4 & pt.optim5 & pt.optim6 & pt.optim7 & pt.optim8 \\
  \hline
wald.stat & 1861.21 & 1796.75 & 1690.26 & 1865.01 & 1814.81 & 1690.62 & 2111.08 & 2175.45 \\
  p.value & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
  reject & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\
   \hline
\end{tabular}
\caption{Test de Wald}
\end{table}
```

3.2.2 Vrais paramètres

Comme nous avons estimé avec des données centrées et réduites, nous utilisons une propriété de la distribution GAL qui nous permet d'obtenir les paramètres des rendements non réduits.

```
> pts.estim.ns <- apply(pts.estim,2,scaleGAL,type="mu",
+                 mean(RETURNS),sd(RETURNS))

>         xtable(pts.estim.ns,
+                     caption="Paramètres des données non centrées et réduites",
+                     digits=4)
```

	pt.optim1	pt.optim2	pt.optim3	pt.optim4	pt.optim5	pt.optim6	pt.optim7	pt.optim8
1	-0.0092	-0.0080	-0.0090	-0.0092	-0.0079	-0.0091	-0.0081	-0.0081
2	0.0078	0.0083	0.0079	0.0078	0.0083	0.0079	0.0081	0.0081
3	0.0033	0.0031	0.0033	0.0033	0.0031	0.0033	0.0030	0.0030
4	2.0214	1.8783	1.9603	2.0220	1.8804	1.9616	1.9660	1.9727

TABLE 3.1 – Paramètres des données non centrées et réduites

3.3 Prix d'options

3.3.1 Données de base

```
> #Taux sans risque
> rfrate <- .05/365
> #Échéance
> T <- 30
> #Pas de discrétisation courbe des prix
> pas <- 0.005
> #Prix initial
> stock0 <- 299
> #Prix d'exercice dans le cours (put)
> strike1 <- stock0*seq(0.98,1,pas)
> #Prix d'exercice hors le cours (put)
> strike2 <- stock0*seq(1+pas,1.02,pas)
> #Prix d'exercice combinés
> strike <- c(strike1,strike2)
> #Damping parameter
> alpha <- 3
```

3.3.2 Paramètres neutres au risque

```
> pts.estim.ns.rn <- apply(pts.estim.ns,2,riskneutralparGAL,rfrate)
> l.pts.estim.ns.rn <- as.list(data.frame(pts.estim.ns.rn))

> xtable(pts.estim.ns.rn,caption="Paramètres neutres au risque",digits=4)
```

	pt.optim1	pt.optim2	pt.optim3	pt.optim4	pt.optim5	pt.optim6	pt.optim7	pt.optim8
1	-0.0066	-0.0057	-0.0065	-0.0066	-0.0057	-0.0065	-0.0058	-0.0058
2	0.0078	0.0083	0.0079	0.0078	0.0083	0.0079	0.0081	0.0081
3	0.0033	0.0031	0.0033	0.0033	0.0031	0.0033	0.0030	0.0030
4	2.0214	1.8783	1.9603	2.0220	1.8804	1.9616	1.9660	1.9727

TABLE 3.2 – Paramètres neutres au risque

3.3.3 Méthode de Epps

```
>         f_putEpps <- function(param,strikeprice,char.fn,eval.time,expiry.time,rate,...)
+               putEpps(strikeprice,char.fn,param,eval.time,expiry.time,rate,...)
>       prix_Epps <- as.data.frame(sapply(l.pts.estim.ns.rn,f_putEpps,strike/stock0,cfLM,0,T,rfrate))

>       xtable(prix_Epps,caption="Prix unitaire de l'option de vente, Méthode de Epps",digits=6)
```

	pt.optim1	pt.optim2	pt.optim3	pt.optim4	pt.optim5	pt.optim6	pt.optim7	pt.optim8
1	0.015415	0.015785	0.015431	0.015417	0.015794	0.015427	0.015792	0.015819
2	0.017360	0.017737	0.017377	0.017362	0.017746	0.017372	0.017742	0.017770
3	0.019456	0.019838	0.019474	0.019458	0.019847	0.019469	0.019843	0.019871
4	0.021705	0.022090	0.021724	0.021707	0.022099	0.021719	0.022093	0.022122
5	0.024107	0.024492	0.024126	0.024109	0.024501	0.024121	0.024495	0.024523
6	0.026661	0.027044	0.026681	0.026663	0.027053	0.026676	0.027046	0.027074
7	0.029365	0.029745	0.029385	0.029367	0.029753	0.029381	0.029745	0.029773
8	0.032217	0.032591	0.032238	0.032219	0.032600	0.032233	0.032591	0.032617
9	0.035214	0.035580	0.035235	0.035216	0.035589	0.035231	0.035579	0.035605

TABLE 3.3 – Prix unitaire de l'option de vente, Méthode de Epps