Contrôle qualité

Projet

Diaby DRAME

Janvier 2023

Données

Affichons les premières lignes des 5 premières colonnes de nos données.

```
data <- read.csv("Qualite2022-15.csv")
data = data[,-1]
head(data)[,1:5]
```

Date	V2	V3	V4	V5
2.190245	8.840612	2.2891141	0.7947490	0.7796204
4.098199	10.814279	0.7640528	0.3370508	0.4068211
9.423776	6.312368	2.2005016	0.1882115	0.3644966
5.676483	15.845459	3.1136102	4.1551161	1.0897279
4.464613	4.912346	3.3717562	0.1057103	0.8004939
5.249883	5.454266	6.0138682	0.1124577	0.4474962

```
as.data.frame(cbind(ligne=dim(data)[1],colonne=dim(data)[2]),col.names=NULL)
```

ligne	colonne
1000	109

On a 1000 lignes et 109 variables au total.

Regardons les propriétés statistiques des 10 premières variables.

```
summary(data)[,1:10]
```

```
##       Date              V2                V3                V4
##  Min.   : 0.2778   Min.   : 0.1753   Min.   : 0.1457   Min.   :0.02869
##  1st Qu.: 2.7349   1st Qu.: 2.7904   1st Qu.: 1.7563   1st Qu.:0.23963
##  Median : 4.0710   Median : 5.2242   Median : 2.6221   Median :0.50092
##  Mean   : 4.3813   Mean   : 6.1777   Mean   : 2.9070   Mean   :0.90056
##  3rd Qu.: 5.7231   3rd Qu.: 8.2956   3rd Qu.: 3.7002   3rd Qu.:1.07156
##  Max.   :13.6901   Max.   :31.8631   Max.   :11.6983   Max.   :9.16003
##        V5                V6                  V7                V8
```

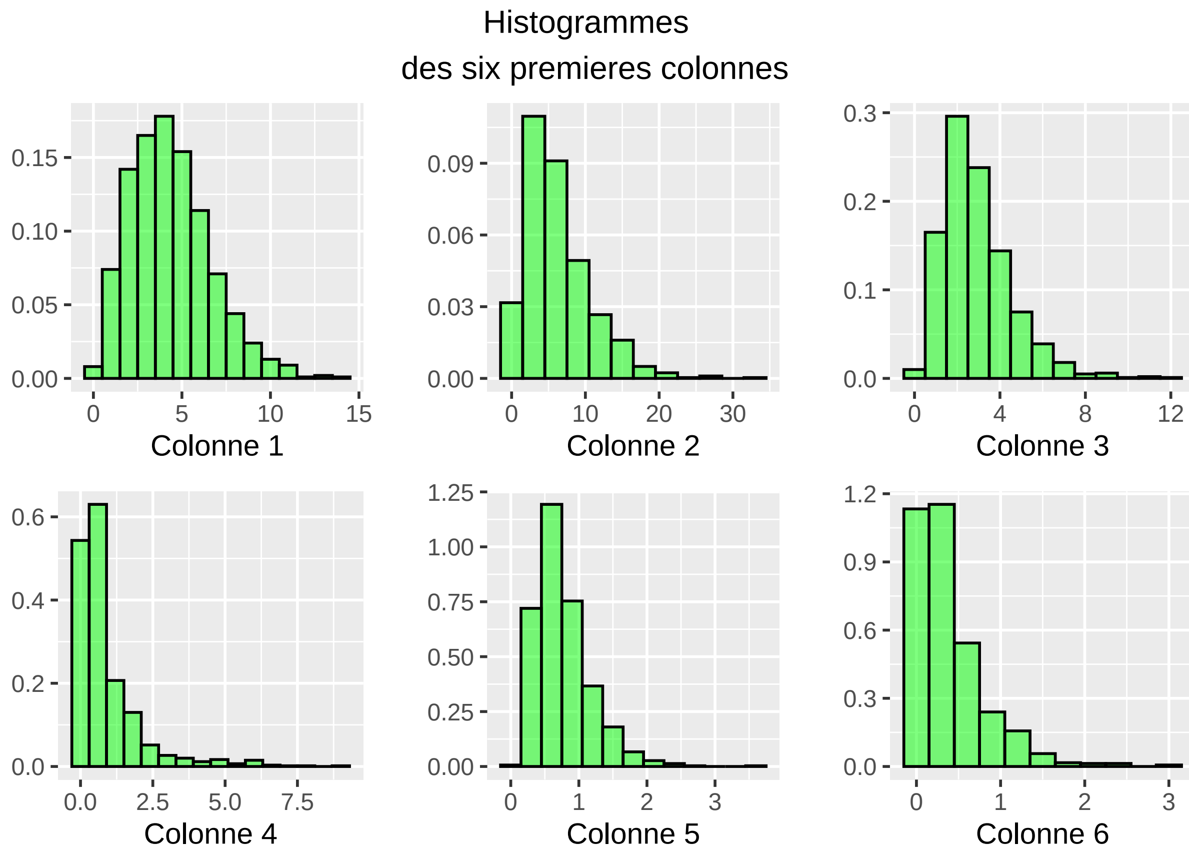
```
##  Min.   :0.09671   Min.   :0.0001438   Min.   :0.5307   Min.   :0.3086
##  1st Qu.:0.47977   1st Qu.:0.1023208   1st Qu.:1.6464   1st Qu.:1.6633
##  Median :0.67187   Median :0.2610318   Median :2.0082   Median :1.9925
##  Mean   :0.76614   Mean   :0.3909301   Mean   :1.9990   Mean   :1.9902
##  3rd Qu.:0.94672   3rd Qu.:0.5257761   3rd Qu.:2.3423   3rd Qu.:2.3077
##  Max.   :3.68219   Max.   :3.0516726   Max.   :3.5215   Max.   :3.8059
##        V9              V10
##  Min.   :0.3824   Min.   :0.6589
##  1st Qu.:1.6664   1st Qu.:1.6798
##  Median :1.9729   Median :2.0305
##  Mean   :1.9825   Mean   :2.0126
##  3rd Qu.:2.3027   3rd Qu.:2.3145
##  Max.   :3.3966   Max.   :3.3993
```

On peut constater que nos données ne sont pas normalisées.

Test d'adéquation

Traçons l'histogramme des 6 premières colonnes afin d'essayer de voir si les données suivent une loi en particulier.

```r
ggp1 <- ggplot(data.frame(data[,1]), aes(x = data[,1])) +
    geom_histogram(aes(data[,1], after_stat(density)), binwidth = 1,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("Colonne 1") + ylab(" ")

ggp2 <- ggplot(data.frame(data[,2]), aes(x = data[,2])) +
    geom_histogram(aes(data[,2], after_stat(density)), binwidth = 3,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("Colonne 2") + ylab(" ")

ggp3 <- ggplot(data.frame(data[,3]), aes(x = data[,3])) +
    geom_histogram(aes(data[,3], after_stat(density)), binwidth = 1,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("Colonne 3") + ylab(" ")

ggp4 <- ggplot(data.frame(data[,4]), aes(x = data[,4])) +
    geom_histogram(aes(data[,4], after_stat(density)), binwidth = 0.6,alpha=0.5,bins =30,
                   fill="green",color="black") +
  xlab("Colonne 4") + ylab(" ")

ggp5 <- ggplot(data.frame(data[,5]), aes(x = data[,5])) +
    geom_histogram(aes(data[,5], after_stat(density)), binwidth = 0.3,alpha=0.5,bins =10,
                   fill="green",color="black") +
  xlab("Colonne 5") + ylab(" ")

ggp6 <- ggplot(data.frame(data[,6]), aes(x = data[,6])) +
    geom_histogram(aes(data[,6], after_stat(density)), binwidth = 0.3,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("Colonne 6") + ylab(" ")

grid.arrange(ggp1, ggp2, ggp3, ggp4, ggp5, ggp6, nrow=2, ncol=3, top=textGrob("Histogrammes
  des six premieres colonnes",gp=gpar(face="bold", hjust=0.5)) )
```

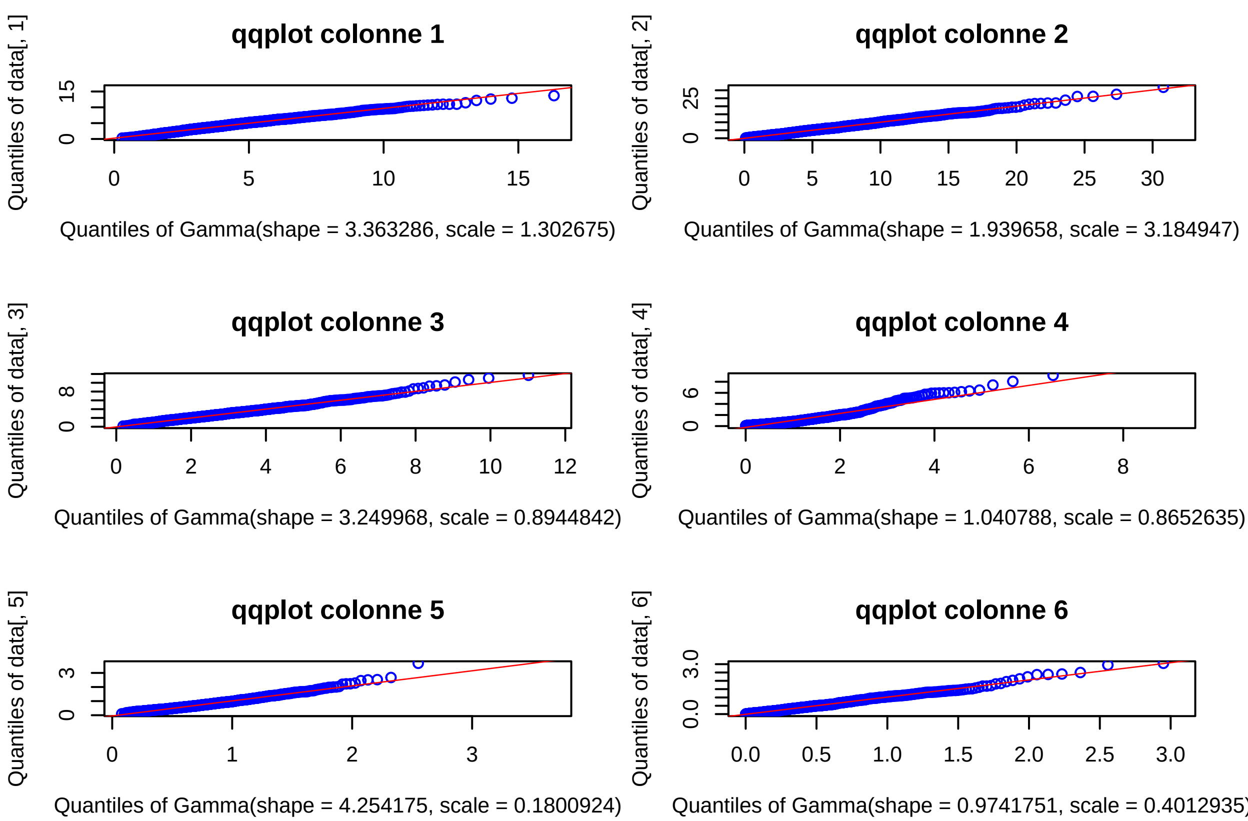


A priori nos données semblent suivre une loi Gamma.

• Pour vérifier cela, traçons des QQ-plot.

On utilisera le package *EnvStats* qui permet de tracer les qqplot et estime aussi les paramètres de la loi que les données suivent.

```
par(mfrow=c(3,2))
qqPlot(data[,1],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 1")

qqPlot(data[,2],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 2")

qqPlot(data[,3],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 3")

qqPlot(data[,4],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 4")

qqPlot(data[,5],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 5")

qqPlot(data[,6],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
       points.col = "blue", line.col = "red",main = "qqplot colonne 6")
```



On peut constater qu'hormis les données de la colonne 4, la droite diagonale ajuste bien le nuage de points. On peut envisager que nos données suivent une loi gamma.

- Confirmons cela par un test d'adéquation de Kolmogorov-Smirnov sous un risque de 5%.

On utilisera la méthode des moments pour estimer les paramètres. En effet, si $X \sim \Gamma(a,b)$, on a $\mathbb{E}[X] = \frac{a}{b}$ et $var(X) = \frac{a}{b}$

```
c1 = data[,1]
ks.test(c1,"pgamma",mean(c1)**2/var(c1),mean(c1)/var(c1))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c1
## D = 0.028267, p-value = 0.4012
## alternative hypothesis: two-sided
```

```
c2 = data[,2]
ks.test(c2,"pgamma",mean(c2)**2/var(c2),mean(c2)/var(c2))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c2
## D = 0.016245, p-value = 0.9545
## alternative hypothesis: two-sided
```

```
c3 = data[,3]
ks.test(c3,"pgamma",mean(c3)**2/var(c3),mean(c3)/var(c3))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c3
## D = 0.021299, p-value = 0.7547
## alternative hypothesis: two-sided
```

```
c4 = data[,4]
ks.test(c4,"pgamma",mean(c4)**2/var(c4),mean(c4)/var(c4))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c4
## D = 0.16605, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
c5 = data[,5]
ks.test(c5,"pgamma",mean(c5)**2/var(c5),mean(c5)/var(c5))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c5
## D = 0.044978, p-value = 0.03498
## alternative hypothesis: two-sided
```

```r
c6 = data[,6]
ks.test(c6,"pgamma",mean(c6)**2/var(c6),mean(c6)/var(c6))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  c6
## D = 0.023833, p-value = 0.621
## alternative hypothesis: two-sided
```

Nous pouvons conclure que :

• les données des colonnes 1, 2, 3 et 6 suivent bien des lois Gamma car les pvalues obtenues sont supérieures au seuil de 5%

• pour les données de la colonne 5, on a une pvalue de 0.03498. Ainsi, les données suivraient une loi gamma sous un seuil de 3%, mais cette hypothèse est rejeté sous un seuil de 5%.

• Au vu de la valeur de la pvalue, les données de la colonne 4 ne semblent pas suivre une loi gamma. Ce qui n'est pas surprenant au vu de son qqplot.

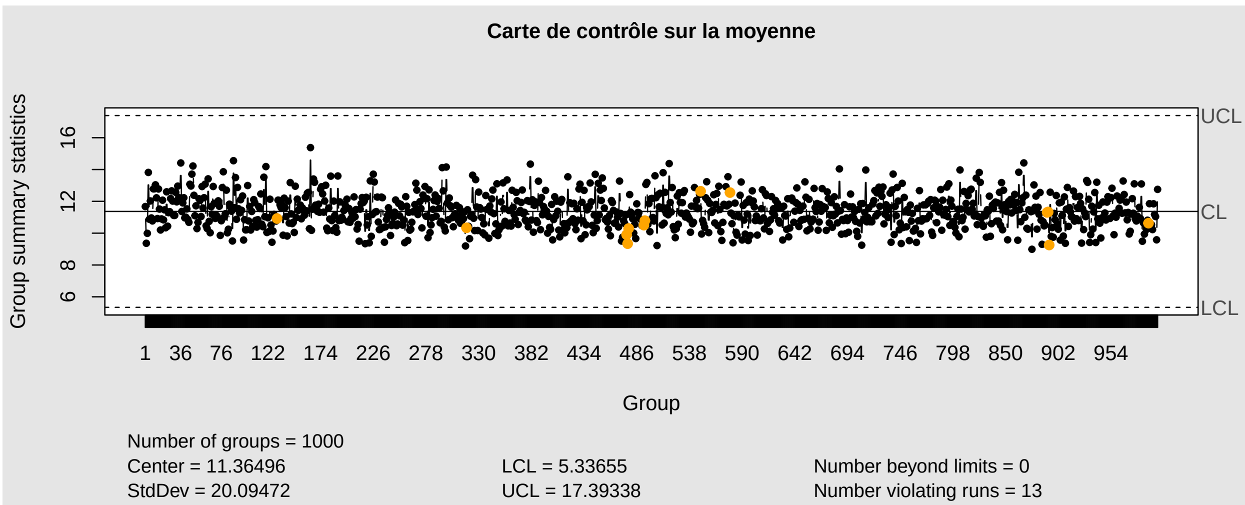
Cartes de contrôle sur la moyenne, la variance et l'étendu

On va créer des cartes de contrôles pour les colonnes 7 à 106.

Pour ce faire, on a utilisera le package *qcc*.

- Pour la moyenne:

```
carte_moyenne = qcc(data[,7:106], type = "xbar", title = "Carte de contrôle sur la moyenne")
```

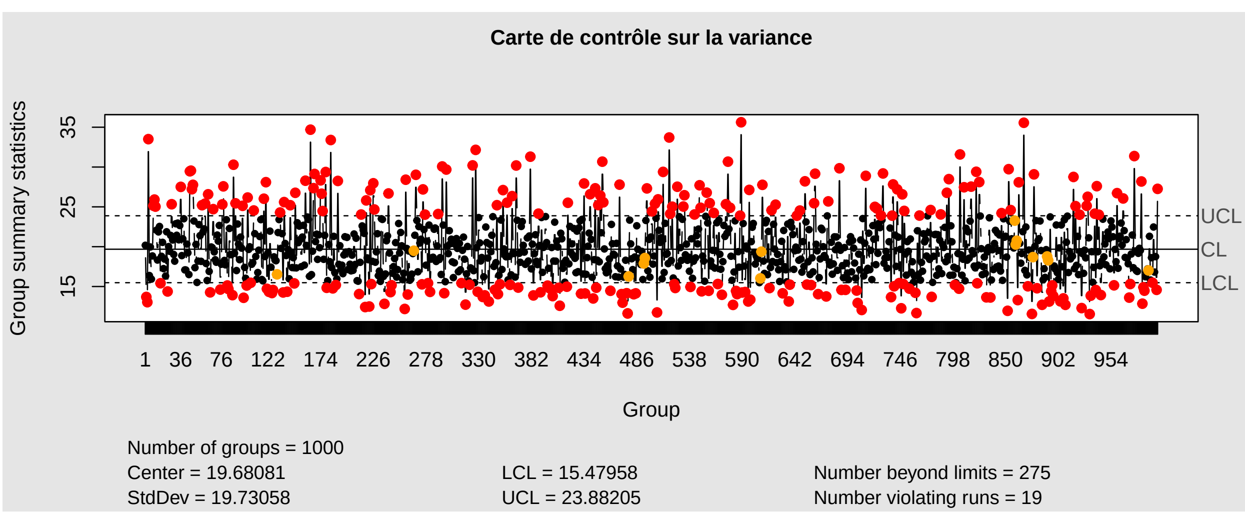


Aucun point n'est hors des intervalles de contrôle.

Nous sommes sous contrôle.

- Pour la variance:

```
carte_variance = qcc(data[,7:106], type = "S", title = "Carte de contrôle sur la variance")
```

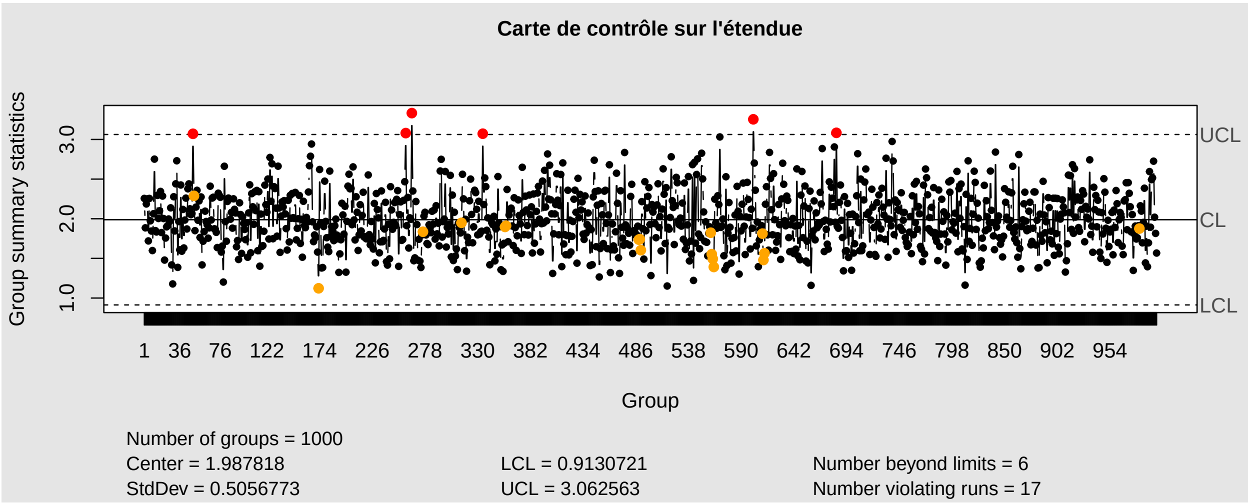


Plusieurs points sont hors des intervalles de contrôle.

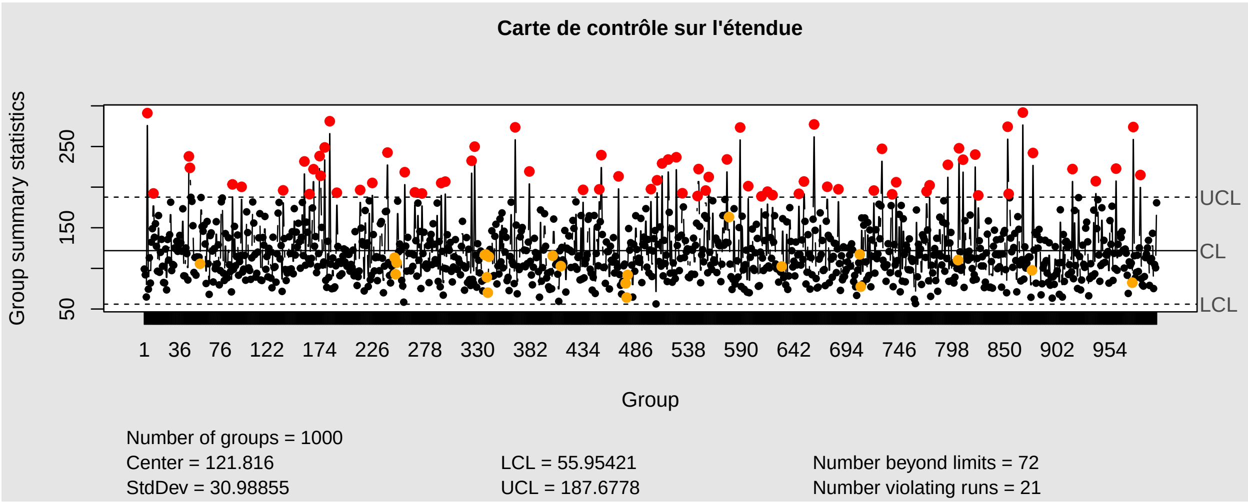
Le processus n'est pas sous contrôle.

- Pour l'etendue:

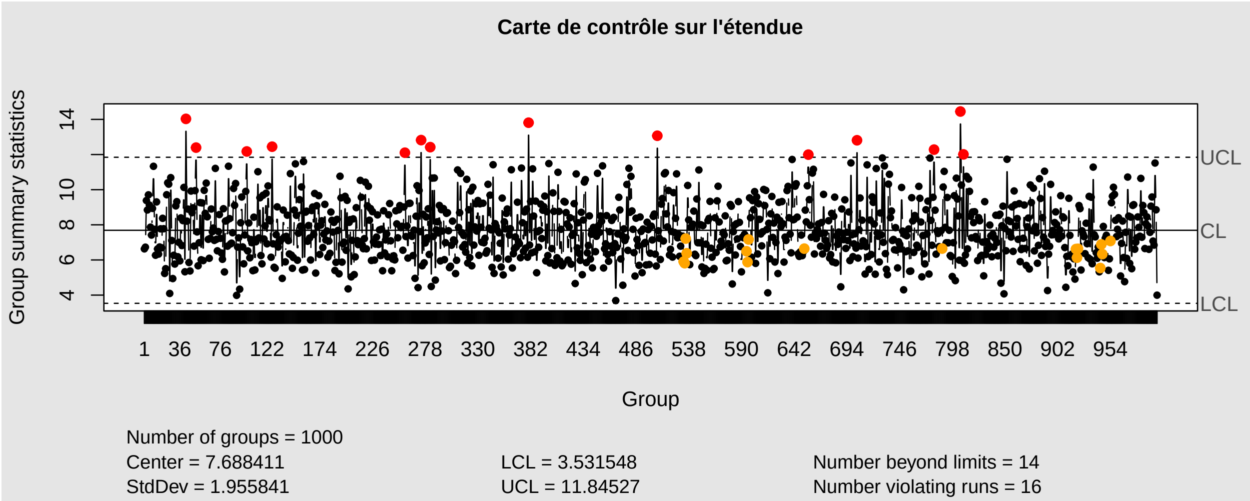
```
c1 = qcc(data[,7:31], type = "R", title = "Carte de contrôle sur l'étendue")
```



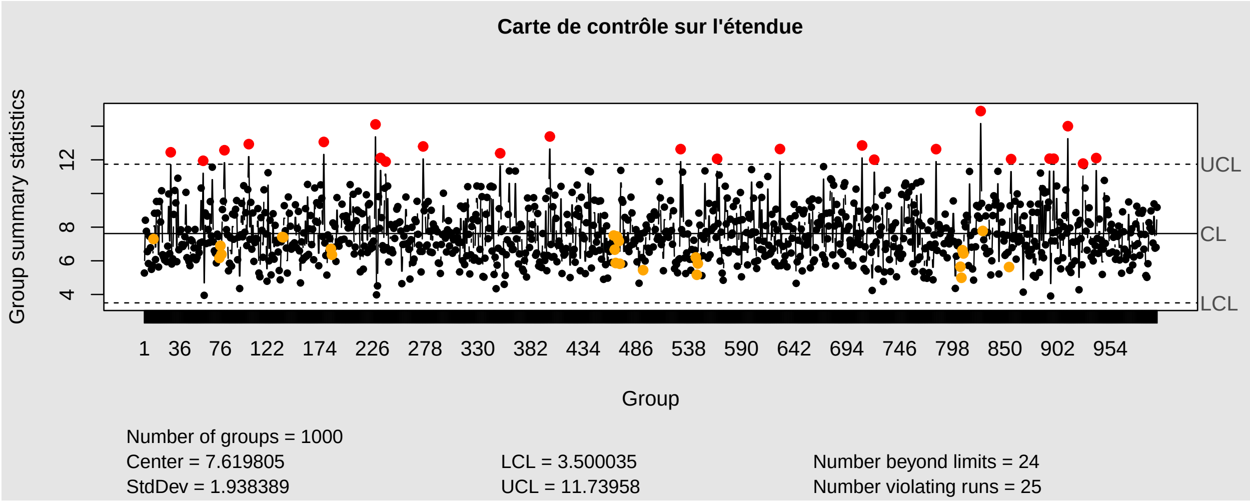

```
c2 = qcc(data[,32:56], type = "R", title = "Carte de contrôle sur l'étendue")
```




```
c3 = qcc(data[,57:81], type = "R", title = "Carte de contrôle sur l'étendue")
```




```
c4 = qcc(data[,82:106], type = "R", title = "Carte de contrôle sur l'étendue")
```



De même que sur la carte de la variance, de nombreux points sont hors contrôle.

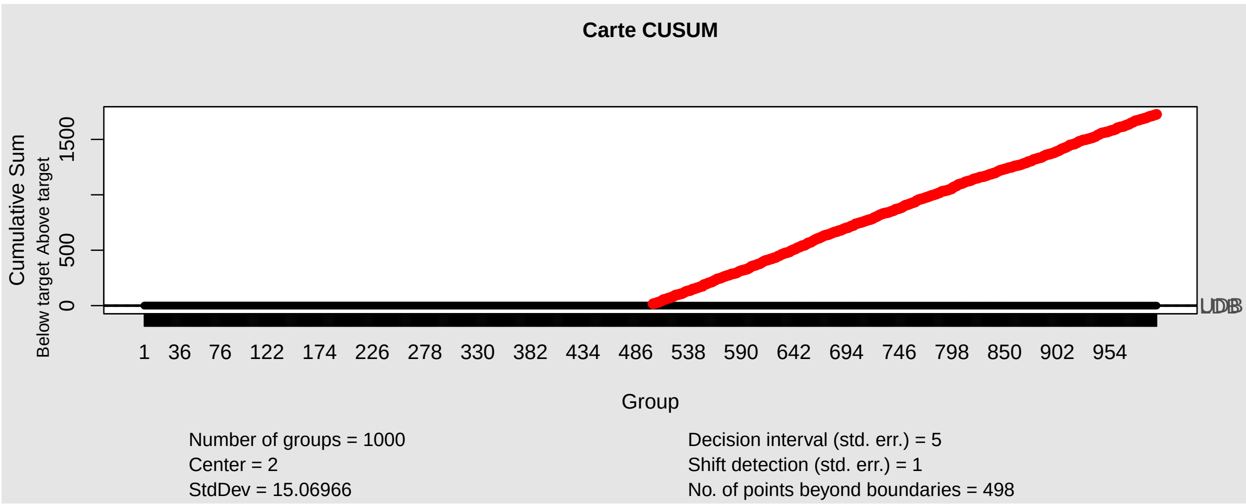
Nous ne sommes donc pas sous contrôle.

Cartes CUSUM et EWMA

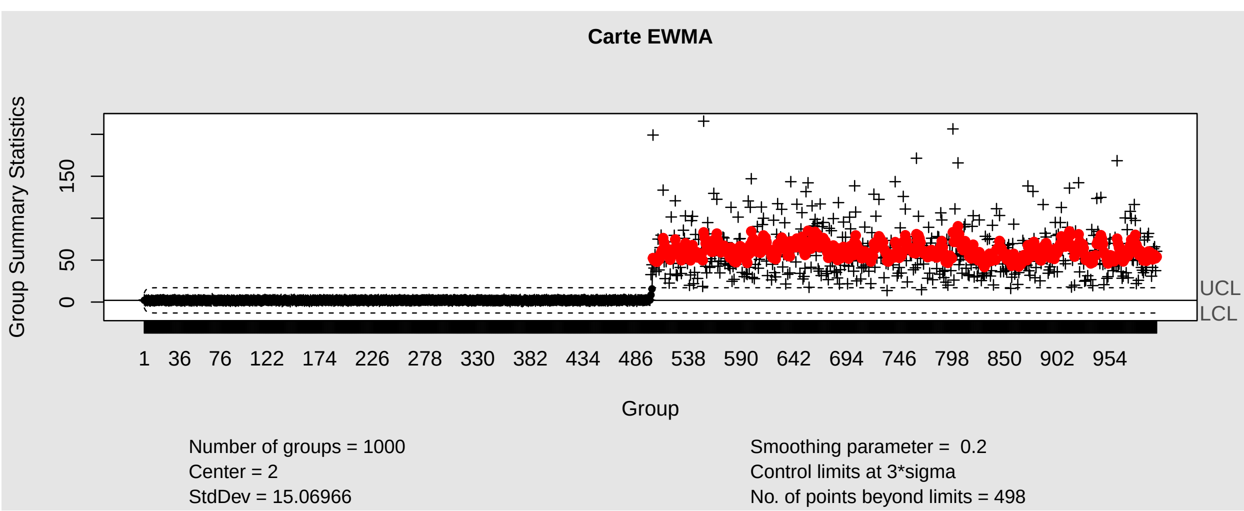
Pour les colonnes 107 et 108, on va essayer de détecter un changement par rapport à une moyenne égale à 2.

- Colonne 107:

```
cusum = cusum(data[,107], center=2, title="Carte CUSUM")
```




```
ewma = ewma(data[,107], center=2, title="Carte EWMA")
```

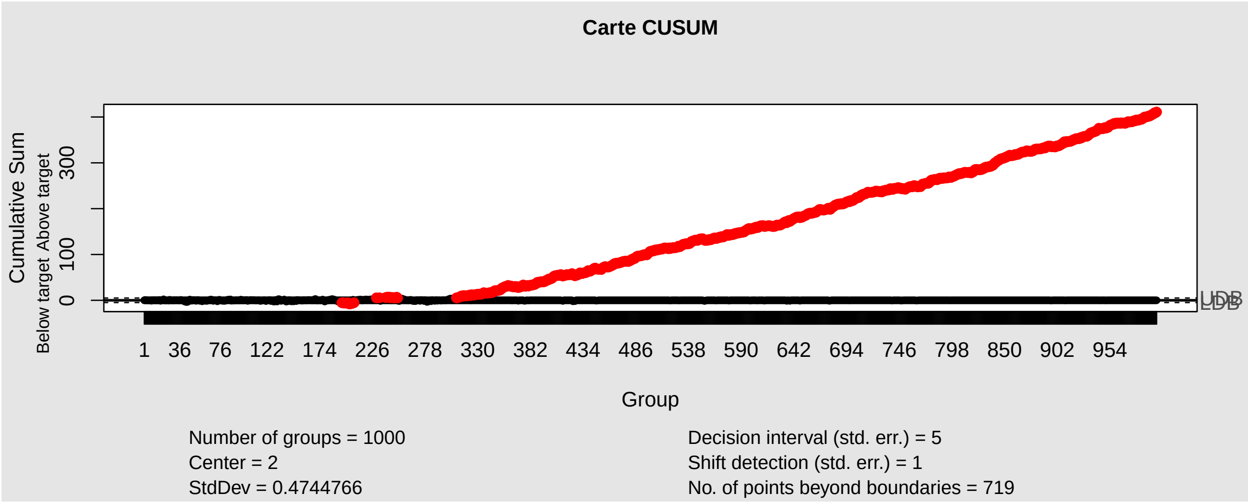


On peut voir que sur les deux cartes, les 498 dernières observations sont hors contrôle..

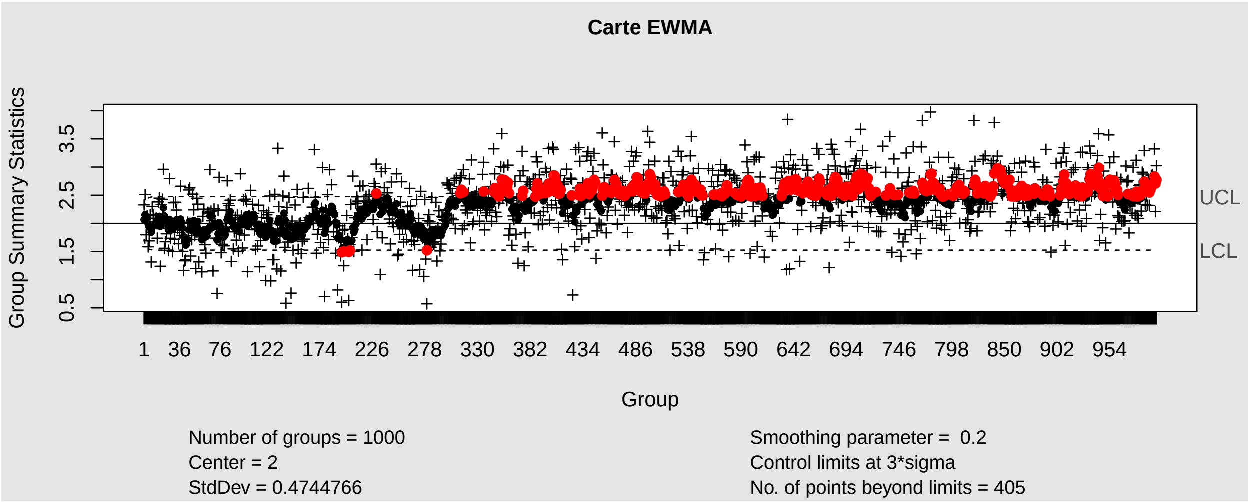
Le changement a lieu à peu près à mis-parcours.

- Colonne 108:

```
cusum = cusum(data[,108], center=2, title="Carte CUSUM")
```




```
ewma = ewma(data[,108], center=2, title="Carte EWMA")
```



On peut voir que sur la carte CUSUM, 719 points sont hors contrôle tandis que pour la carte EWMA, c'est 405 points qui sont hors contrôle.

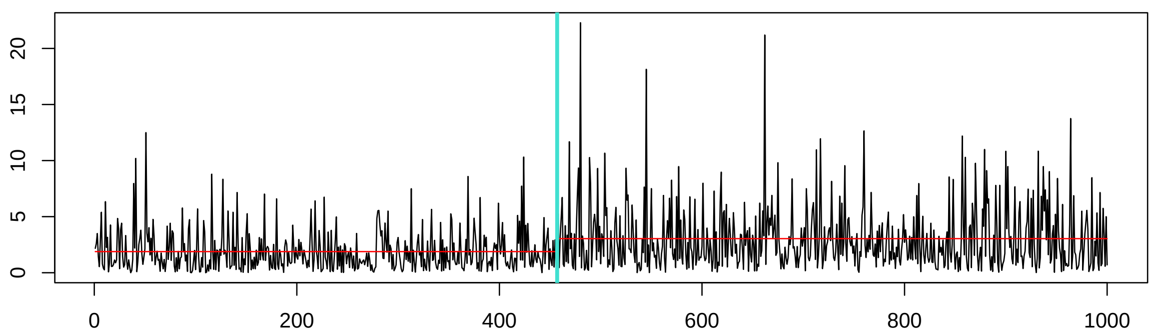
La carte EWMA est donc préférable à la carte CUSUM.

Instant de rupture

• Essayons de détecter l'instant de rupture dans les données de la colonne 109, s'il a lieu.

En s'aidant de la fonction *cpt.mean* du package *chamgepoint*, regardons s'il y a un changement dans la moyenne.

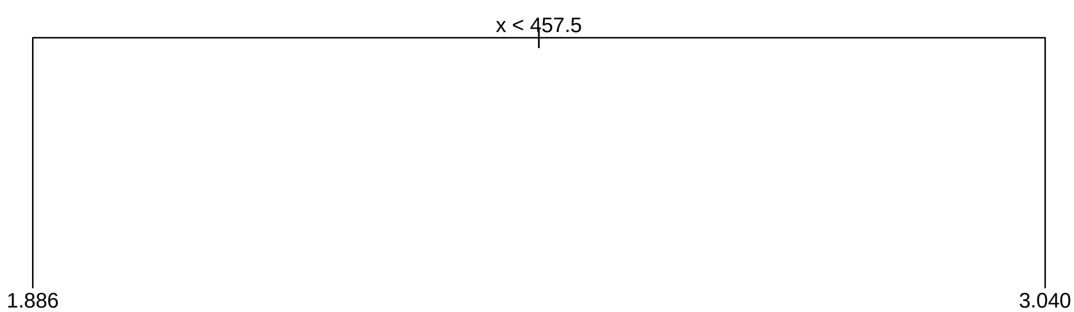
```
m = cpt.mean(data[,109])
r = cpts(m)
plot(m,ylab=NA, xlab=NA)
abline(v = r, lwd = 3, col="turquoise")
```



Le changement a eu lieu à l'instant de rupture 457.

Faisons un arbre de décision pour voir si nous tombons sur le même instant de rupture.

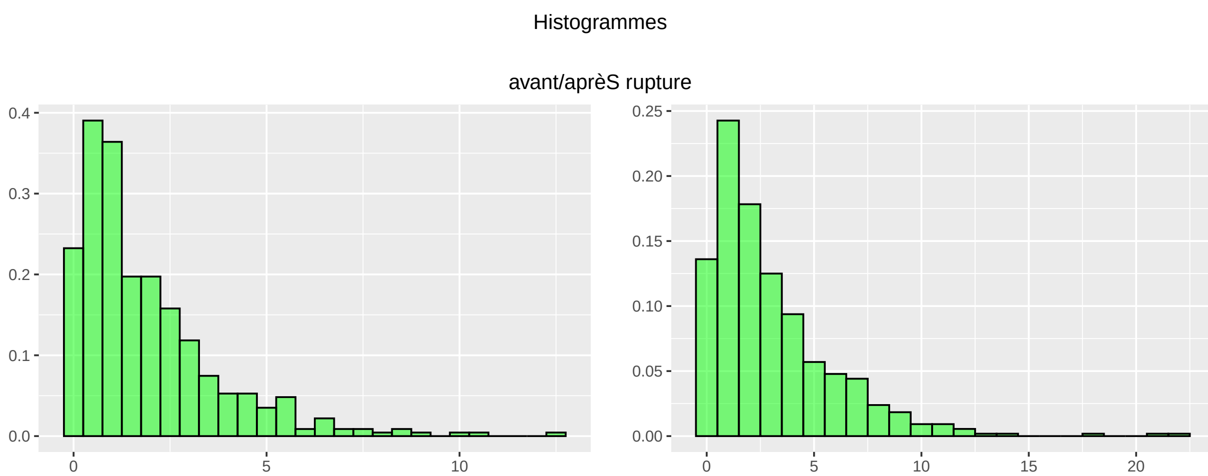
```
d = data.frame(x=1:1000,y=data[,109])
plot(tree(y~.,d))
text(tree(y~.,d))
```



On retrouve correctement le même instant de rupture.

Regardons la loi des données avant et après l'instant de rupture.

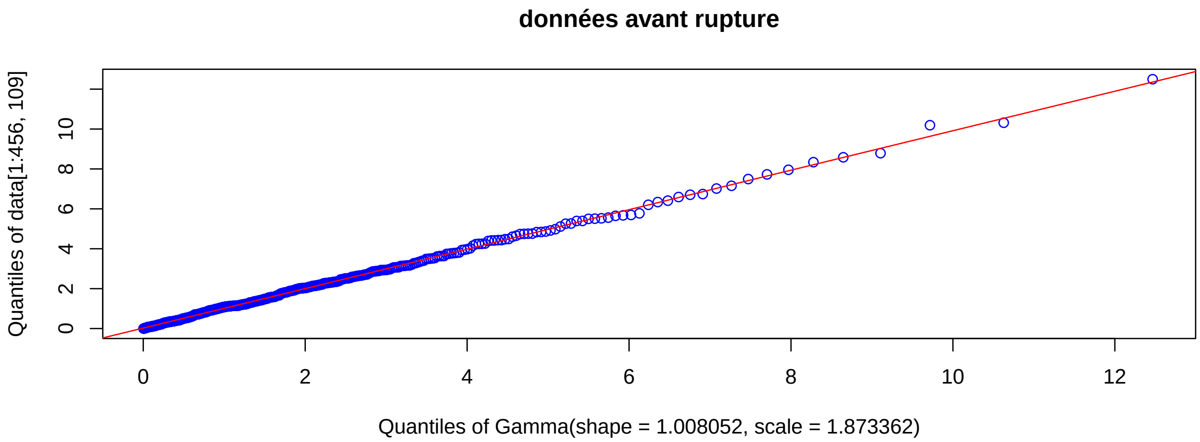
```
ggp1 <- ggplot(data.frame(data[1:456,109]), aes(x = data[1:456,109])) +
    geom_histogram(aes(data[1:456,109], after_stat(density)), binwidth = 0.5,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("") + ylab(" ")

ggp2 <- ggplot(data.frame(data[457:1000,109]), aes(x = data[457:1000,109])) +
    geom_histogram(aes(data[457:1000,109], after_stat(density)), binwidth = 1,alpha=0.5,bins =50,
                   fill="green",color="black") +
  xlab("") + ylab(" ")

grid.arrange(ggp1, ggp2, nrow=1, ncol=2, top=textGrob("Histogrammes

avant/aprèS rupture",gp=gpar(face="bold",hjust=0.5)) )
```



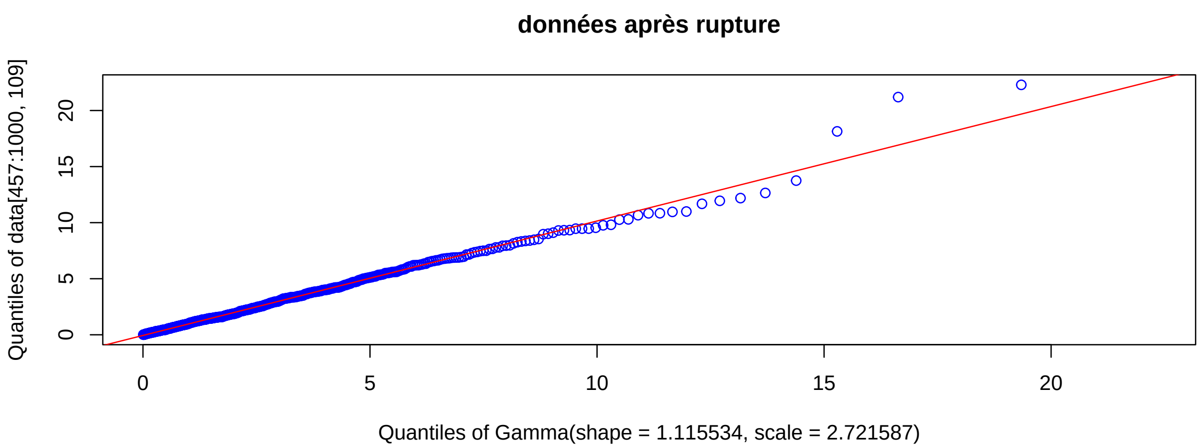
Cela semble être des lois gamma.

Traçons les qqplot et faisons le test de Kolmogorov-Smirnov.

```
qqPlot(data[1:456,109],dist = "gamma", estimate.params = TRUE, add.line = TRUE,
      points.col = "blue", line.col = "red",main = "données avant rupture")
```




```
qqPlot(data[457:1000,109],dist = "gamma", estimate.params = TRUE,add.line = TRUE,
       points.col = "blue", line.col = "red",main = "données après rupture")
```




```
av = data[1:456,109]
ks.test(av,"pgamma",mean(av)**2/var(av),mean(av)/var(av))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  av
## D = 0.023772, p-value = 0.9589
## alternative hypothesis: two-sided
```

```
ap = data[457:1000,109]
ks.test(ap,"pgamma",mean(ap)**2/var(ap),mean(ap)/var(ap))
```

```
##
```

```
##  One-sample Kolmogorov-Smirnov test
##
## data:  ap
## D = 0.023611, p-value = 0.9221
## alternative hypothesis: two-sided
```

On peut donc en déduire que, nos données d'avant et d'après rupture suivent des lois gamma.

- À présent, donnons la borne supérieur du délai à la détection et le taux de fausse alarme.

Pour calculer l'instant de détection, nous utilisons la formule du cours suivante:

$$T_c = inf\{t \geq : \max_{1 \leq k < t} S_k^t\}$$

$$S_k^t = \sum_{i=k}^{t} \frac{P_{\theta_1}(X_i)}{P_{\theta_0}(X_i)}$$

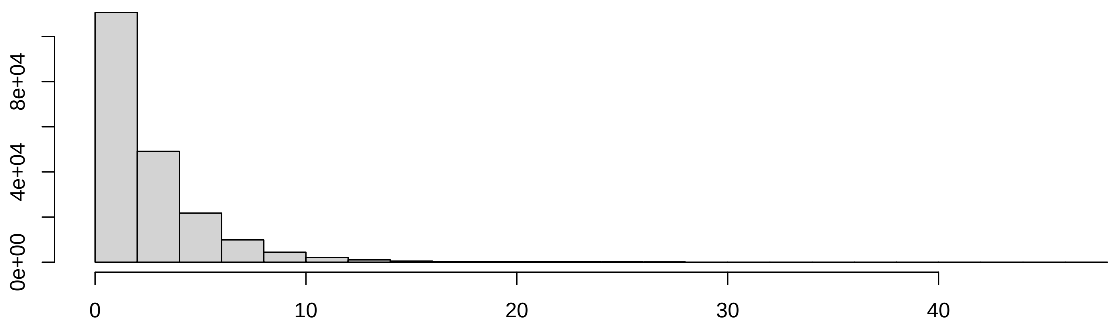
On fixe le seuil h à 5.

On va faire 456 réalistions de nos données avant rupture et $1000 - 457 = 543$ réalisations de nos données après rupture.

Il est à noter qu'on peut estimer les paramètres de nos données via la méthode des moments.

On répétera 200 fois l'étape précédente.

```r
set.seed(10)

N_simulation <- function(N,n,rupture){

  r = matrix(0,nrow = 999, ncol = N)
  for (i in 1:N){
   r[,i] = c(rgamma(rupture-1,mean(av)**2/var(av),mean(av)/var(av)),
             rgamma(n-rupture,mean(ap)**2/var(ap),mean(ap)/var(ap)))
  }

  return (r)
}
N = 200
rep = N_simulation(N,1000,457)
hist(rep,ylab=NA, xlab=NA,main = NA)
```




```
S_kt = function(X,k,t){

  m0 = mean(av)
  v0 = var(av)
  m1 = mean(ap)
  v1 = var(ap)

  s = 0
  for(i in k:t){
    s = s + log(dgamma(X[i],m1**2/v1,m1/v1)/dgamma(X[i],m0**2/v0,m0/v0))
  }

  return (s)
}


detection <- function(X,t=1){

  ind = which.max(sapply(1:t,function(k) S_kt(X,k,t)))
  Max =  S_kt(X,ind,t)

  while(Max < 5){

    t = t+1
    ind = which.max(sapply(1:t,function(k) S_kt(X,k,t)))

    Max =  S_kt(X,ind,t)

  }

  return (t)
}

library(parallel)
library(foreach)
library(doParallel)
```

```r
f <- function(N,rupture,X){

  a = 0
  m = NULL

  Ncpus <- parallel::detectCores() - 1
  cl <- parallel::makeCluster(Ncpus)
  doParallel::registerDoParallel(cl)

  foreach::foreach(i=1:N, .packages=c("e1071")) %do%  {

    x = X[,i]
    d = detection(x)
    print(d)
    m[i] = abs(d-rupture)

    if (d <= rupture){

        a = a+1
    }

  }

   parallel::stopCluster(cl)


  return (list(alarme = a/N, sup = max(m)))
}
```

```r
rupture=457

f = f(N,rupture,rep)
```

```
## [1] 169
## [1] 145
## [1] 73
## [1] 488
## [1] 504
## [1] 495
## [1] 497
## [1] 565
## [1] 470
## [1] 486
## [1] 479
## [1] 499
## [1] 478
## [1] 478
## [1] 486
## [1] 527
## [1] 515
## [1] 462
## [1] 480
## [1] 479
```

```
## [1] 469
## [1] 479
## [1] 95
## [1] 473
## [1] 485
## [1] 502
## [1] 493
## [1] 500
## [1] 483
## [1] 481
## [1] 151
## [1] 410
## [1] 478
## [1] 486
## [1] 512
## [1] 464
## [1] 492
## [1] 545
## [1] 469
## [1] 468
## [1] 142
## [1] 540
## [1] 559
## [1] 508
## [1] 550
## [1] 464
## [1] 547
## [1] 386
## [1] 481
## [1] 522
## [1] 486
## [1] 467
## [1] 483
## [1] 501
## [1] 100
## [1] 508
## [1] 463
## [1] 468
## [1] 330
## [1] 535
## [1] 478
## [1] 502
## [1] 336
## [1] 479
## [1] 464
## [1] 492
## [1] 483
## [1] 528
## [1] 491
## [1] 521
## [1] 489
## [1] 478
## [1] 476
## [1] 476
```

```
## [1] 466
## [1] 461
## [1] 465
## [1] 498
## [1] 480
## [1] 267
## [1] 480
## [1] 498
## [1] 468
## [1] 473
## [1] 348
## [1] 489
## [1] 470
## [1] 332
## [1] 475
## [1] 490
## [1] 471
## [1] 576
## [1] 458
## [1] 471
## [1] 310
## [1] 484
## [1] 197
## [1] 479
## [1] 151
## [1] 555
## [1] 498
## [1] 470
## [1] 469
## [1] 493
## [1] 476
## [1] 470
## [1] 481
## [1] 475
## [1] 516
## [1] 493
## [1] 490
## [1] 463
## [1] 501
## [1] 462
## [1] 500
## [1] 537
## [1] 480
## [1] 127
## [1] 506
## [1] 520
## [1] 336
## [1] 489
## [1] 467
## [1] 505
## [1] 499
## [1] 492
## [1] 576
## [1] 498
```

```
## [1] 486
## [1] 528
## [1] 500
## [1] 511
## [1] 469
## [1] 482
## [1] 511
## [1] 490
## [1] 486
## [1] 466
## [1] 537
## [1] 486
## [1] 558
## [1] 489
## [1] 480
## [1] 498
## [1] 464
## [1] 490
## [1] 468
## [1] 379
## [1] 502
## [1] 490
## [1] 464
## [1] 482
## [1] 623
## [1] 502
## [1] 285
## [1] 462
## [1] 498
## [1] 509
## [1] 481
## [1] 493
## [1] 486
## [1] 486
## [1] 222
## [1] 514
## [1] 462
## [1] 487
## [1] 464
## [1] 497
## [1] 571
## [1] 487
## [1] 488
## [1] 483
## [1] 479
## [1] 463
## [1] 467
## [1] 523
## [1] 494
## [1] 490
## [1] 476
## [1] 510
## [1] 490
## [1] 473
```

```
## [1] 493
## [1] 466
## [1] 479
## [1] 463
## [1] 461
## [1] 565
## [1] 87
## [1] 509
## [1] 462
## [1] 478
## [1] 425
## [1] 474
## [1] 476
## [1] 495
## [1] 476
## [1] 495
## [1] 494
## [1] 516
```

La borne supérieur du délai à la détection :

```
f$sup
```

```
## [1] 384
```

Le taux de fausse :

```
f$alarme
```

```
## [1] 0.12
```