Análisis mtDNA: LSU

Borja Freire Castro

1/10/2020

```
FALSE Loading required package: ggplot2

FALSE Loading required package: pscl

FALSE Classes and Methods for R developed in the
FALSE Political Science Computational Laboratory
FALSE Department of Political Science
FALSE Stanford University
FALSE Simon Jackman
FALSE hurdle and zeroinfl functions by Achim Zeileis

FALSE Loading required package: boot

FALSE Loading required package: VGAM

FALSE Loading required package: stats4

FALSE Loading required package: splines

FALSE
FALSE Attaching package: 'VGAM'

FALSE The following objects are masked from 'package:boot':
FALSE
FALSE     logit, simplex

FALSE Loading required package: gamlss

FALSE Loading required package: gamlss.data

FALSE
FALSE Attaching package: 'gamlss.data'

FALSE The following object is masked from 'package:boot':
FALSE
FALSE     aids

FALSE The following object is masked from 'package:datasets':
FALSE
FALSE     sleep

FALSE Loading required package: gamlss.dist

FALSE Loading required package: MASS

FALSE Loading required package: nlme

FALSE Loading required package: parallel

FALSE  **********   GAMLSS Version 5.2-0  **********

FALSE For more on GAMLSS look at https://www.gamlss.com/

FALSE Type gamlssNews() to see new features/changes/bug fixes.

FALSE Loading required package: locfit

FALSE locfit 1.5-9.1     2013-03-22
```

```
FALSE
FALSE Attaching package: 'locfit'

FALSE The following object is masked from 'package:gamlss':
FALSE
FALSE     lp

FALSE Loading required package: KernSmooth

FALSE KernSmooth 2.23 loaded
FALSE Copyright M. P. Wand 1997-2009

FALSE Loading required package: SparseM

FALSE
FALSE Attaching package: 'SparseM'

FALSE The following object is masked from 'package:base':
FALSE
FALSE     backsolve

FALSE This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.

FALSE
FALSE Attaching package: 'mgcv'

FALSE The following object is masked from 'package:VGAM':
FALSE
FALSE     s

FALSE Package 'sm', version 2.2-5.6: type help(sm) for summary information

FALSE
FALSE Attaching package: 'sm'

FALSE The following object is masked from 'package:MASS':
FALSE
FALSE     muscle

FALSE The following object is masked from 'package:boot':
FALSE
FALSE     dogs
```

Carga y preprocesamiento de los datos.

```
data <- na.omit(read.csv('../output/lsu_df.csv', sep = ',', dec = '.', header = T))
dim(data)
```

```
## [1] 3651    5
```

```
head(data)
```

```
##   X Position  CVTOT  CVMIT GB.Seqs
## 1 0        1 -0.544 -0.642       0
## 2 1        2 -0.703 -0.879       0
## 3 2        3 -0.386 -0.518       0
## 4 3        4 -0.411 -0.777       0
## 5 4        5  0.183 -0.803       0
## 6 5        6  0.256 -0.642       0
```

```
data$CVTOT <- as.numeric(as.character(data$CVTOT))
```

```
## Warning: NAs introducidos por coerción
```

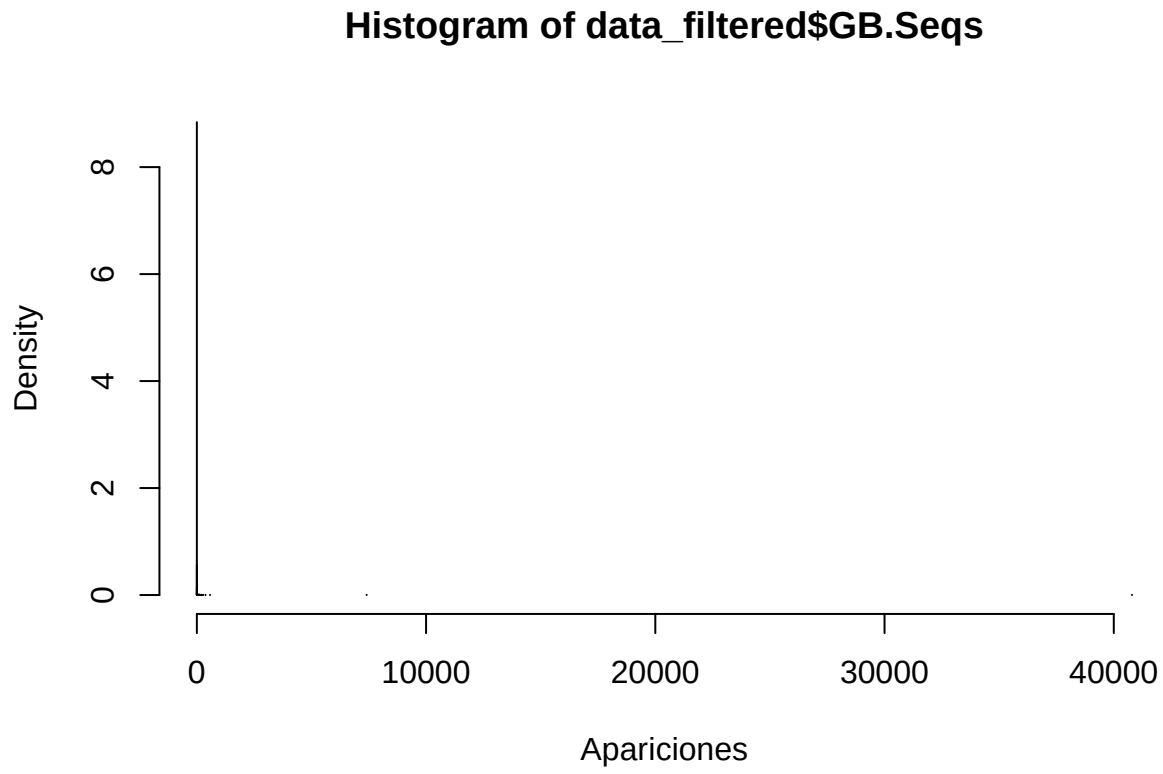
```r
data$GB.Seqs <- na.omit(as.numeric(as.character(data$GB.Seqs)))
data$Log.GB.Seqs <- log(data$GB.Seqs + 1)
data$Bin.GB.Seqs <- numeric(length(data$GB.Seqs))
data <- na.omit(data)
data_neg <- data[data$CVTOT == -1,]
data_filtered <- data[data$CVTOT != -1,]
# Visualización de los hist de data_filtered
sum(data_filtered$GB.Seqs == 0)
```

```
## [1] 2677
```

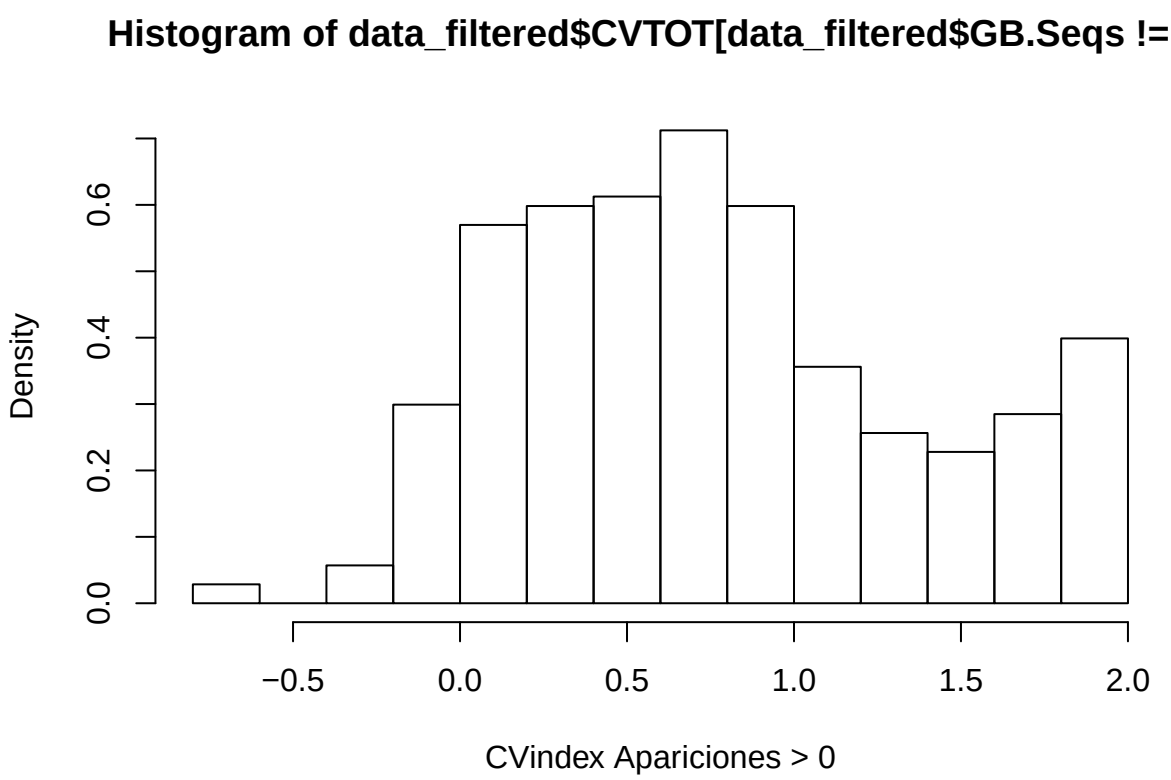
```r
sum(data_filtered$GB.Seqs != 0)
```

```
## [1] 351
```

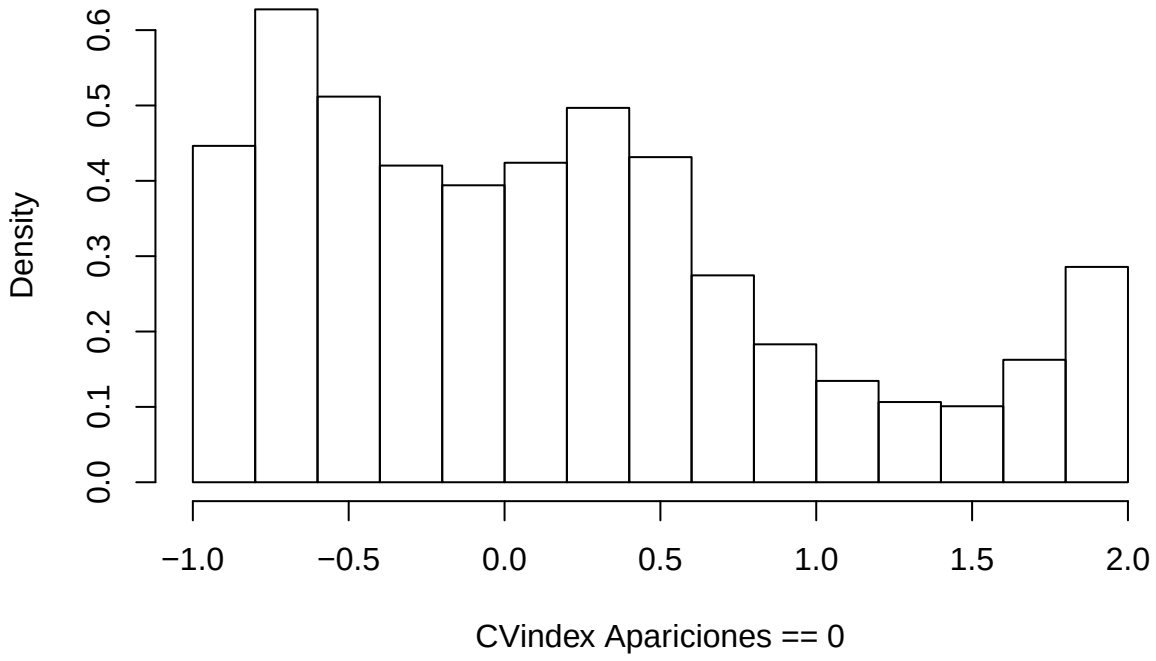
```r
hist(data_filtered$GB.Seqs, xlab = 'Apariciones', breaks = 'fd', freq = F)
```




```r
hist(data_filtered$CVTOT[data_filtered$GB.Seqs != 0],
     breaks = 'fd', freq = F, xlab = 'CVindex Apariciones > 0')
```




```
hist(data_filtered$CVTOT[data_filtered$GB.Seqs == 0],
     breaks = 'fd', freq = F, xlab = 'CVindex Apariciones == 0')
```




```
summary(data_neg)
```

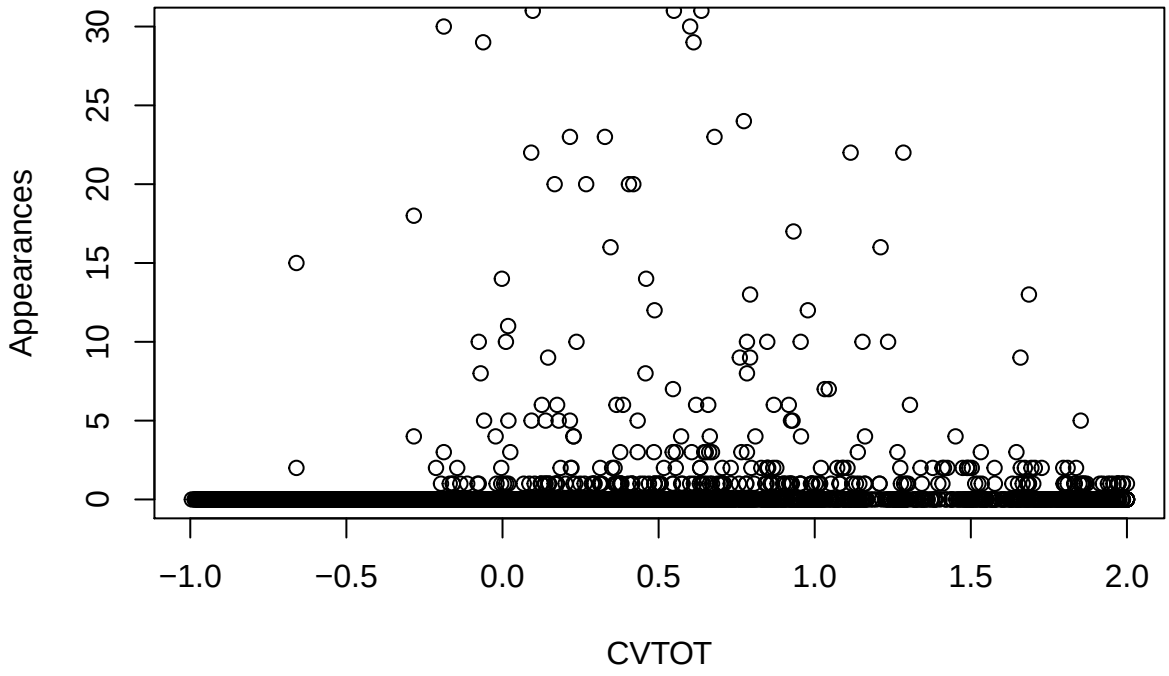
```
##        X           Position        CVTOT        CVMIT        GB.Seqs
##  Min.   :  30   Min.   :1673   Min.   :-1          :621   Min.   :   0.00
##  1st Qu.:1046   1st Qu.:2061   1st Qu.:-1   --     :  0   1st Qu.:   0.00
##  Median :2040   Median :2356   Median :-1   -0.001 :  0   Median :   1.00
##  Mean   :1887   Mean   :2393   Mean   :-1   -0.002 :  0   Mean   :  52.69
##  3rd Qu.:2728   3rd Qu.:2789   3rd Qu.:-1   -0.003 :  0   3rd Qu.:   7.00
##  Max.   :3759   Max.   :3229   Max.   :-1   -0.004 :  0   Max.   :3978.00
##                                             (Other):  0
##   Log.GB.Seqs      Bin.GB.Seqs
##  Min.   :0.0000   Min.   :0
##  1st Qu.:0.0000   1st Qu.:0
##  Median :0.6931   Median :0
##  Mean   :1.3685   Mean   :0
##  3rd Qu.:2.0794   3rd Qu.:0
##  Max.   :8.2888   Max.   :0
##
```

```
summary(data_filtered)
```

```
##        X             Position          CVTOT             CVMIT
##  Min.   :   0.0   Min.   :   1.0   Min.   :-0.9950   2      :  31
##  1st Qu.: 926.8   1st Qu.: 910.8   1st Qu.:-0.4783   -0.075 :  21
##  Median :1832.5   Median :1772.5   Median : 0.1535   -0.027 :  20
##  Mean   :1870.8   Mean   :1657.0   Mean   : 0.2226   -0.055 :  18
##  3rd Qu.:2826.2   3rd Qu.:2403.2   3rd Qu.: 0.6993   -0.082 :  17
```

```
##  Max.   :3672.0   Max.   :3167.0   Max.   : 2.0000   --      :  13
##                                                      (Other):2908
##     GB.Seqs          Log.GB.Seqs       Bin.GB.Seqs
##  Min.   :    0.00   Min.   : 0.0000   Min.   :0
##  1st Qu.:    0.00   1st Qu.: 0.0000   1st Qu.:0
##  Median :    0.00   Median : 0.0000   Median :0
##  Mean   :   17.78   Mean   : 0.1858   Mean   :0
##  3rd Qu.:    0.00   3rd Qu.: 0.0000   3rd Qu.:0
##  Max.   :40793.00   Max.   :10.6163   Max.   :0
##
```

```
plot(data_filtered$CVTOT,data_filtered$GB.Seqs,pch=1,ylim = c(0,30),
     xlab="CVTOT",ylab="Appearances")
```




```
x <- unique(data_filtered$CVTOT)
y <- data_filtered$GB.Seqs
```

Análisis paramétricos:

Repetimos los experimentos realizados para el caso original pero tras haber eliminado los casos con valores asignados de -1 en el índice de conservación (CVTOT).

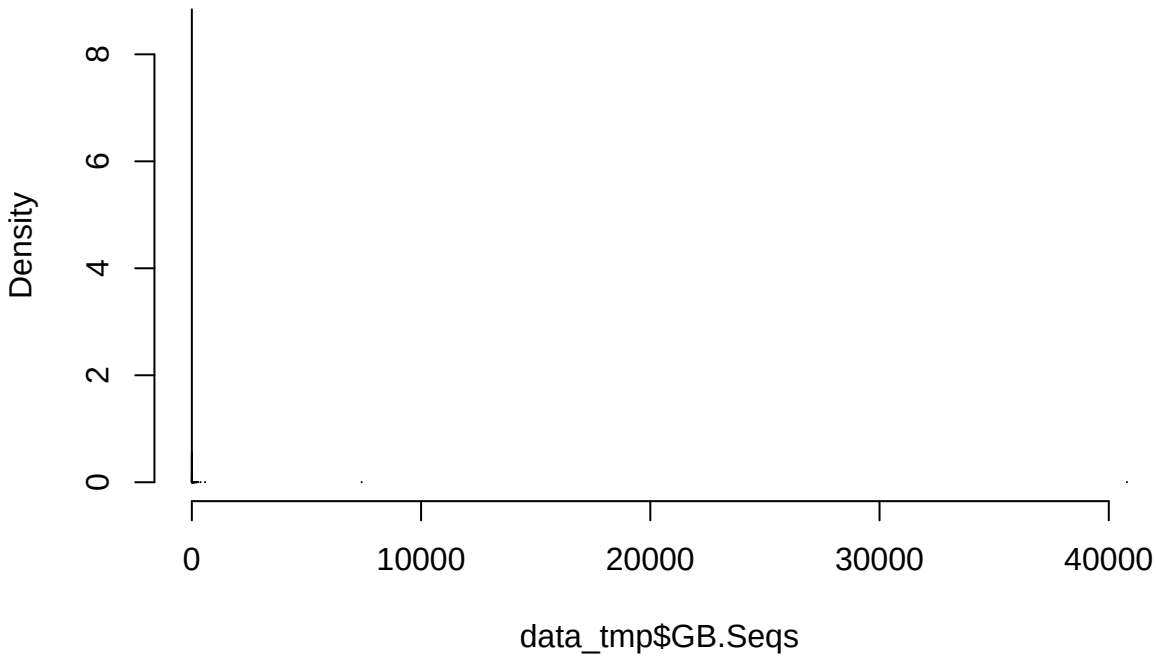
- Filtrado de datos superiores: 0%, 1% y 5%
- Ajustes:
  - Poisson
  - QuasiPoisson (estimación del parámetro de sobredispersión)
  - Zeros Inflados:

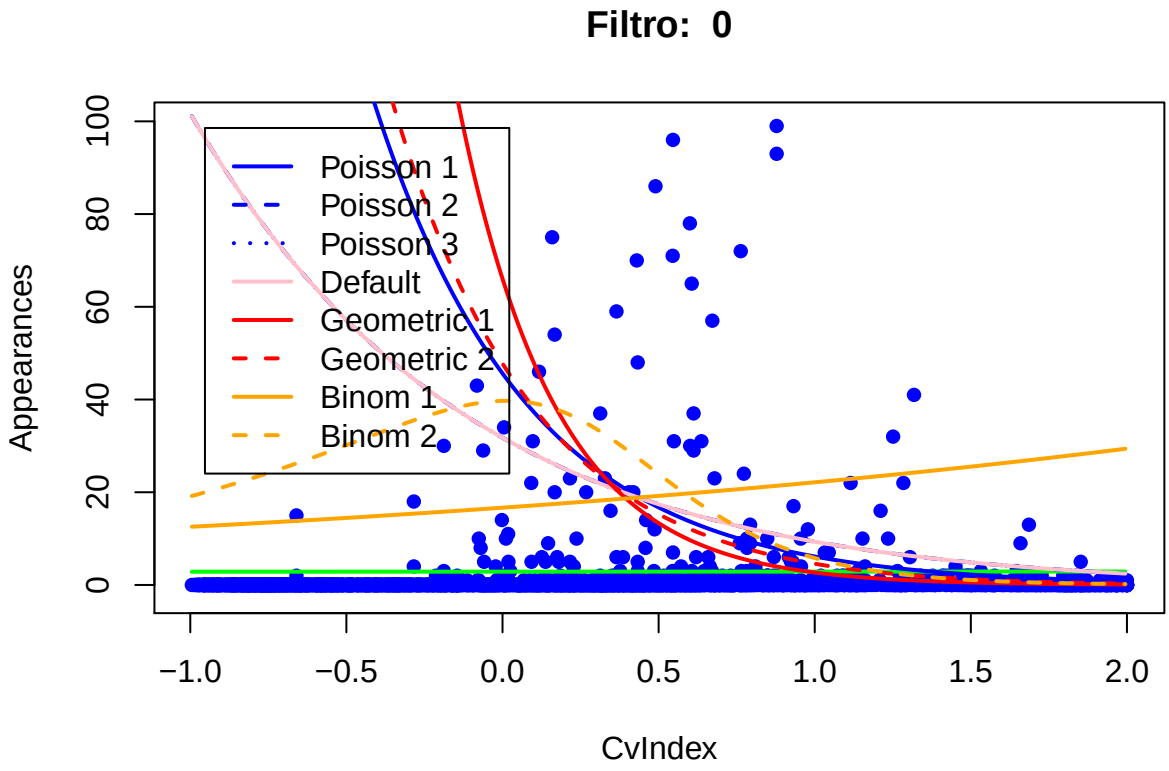
        * Poisson
        * Geométrica
        * Link: logit
    - Binomial Negativa

```
## Conditional variance - modo univariante pensar en el modo multi
con_var <- function(df, residuos, col_y, col_x, new_data, FUN = glm, ...)
{
  df$con_var_up <- df[,col_y]+residuos^2
  df$con_var_down <- df[,col_y] - residuos^2
  print(length(df[,col_x]))
  print(length(df$con_var_up))
  model_adjust_up <- FUN(df['con_var_up'] ~ df[,col_x], ...)
  model_adjust_down <- FUN(df['con_var_down'] ~ df[,col_x], ...)
  return(c(sqrt(predict(model_adjust_up, newdata = new_data)),
           sqrt(predict(model_adjust_down, newdata= new_data))))
}

## Estudios paramétricos:
### GLM
## Poisson
## Geometrica
## BN
# Poisson + ZerosInflated
## Caso 1: Datos completos
## Caso 2: Datos filtrados al 1% superior
## Caso 3: Datos filtrados al 5$ superior
l_a <- c(0, 0.01, 0.05)
for (a in l_a)
{
  # Eliminamos datos fuera de lugar
  lower_bound <- quantile(y, 0.00);lower_bound
  upper_bound <- quantile(y, 1 - a);upper_bound

  outlier_ind <- which(y < lower_bound | y > upper_bound)
  if (!is.na(outlier_ind[1]))
    data_tmp <- data_filtered[-outlier_ind,]
  else
    data_tmp <- data_filtered

  # Comprobamos datos
  summary(data_tmp)
  head(data_tmp)
  dim(data_tmp)
  # Analisis visual
  #windows()
  hist(data_tmp$GB.Seqs, freq = F, breaks = 'fd')

  #windows()
  plot(data_tmp['CVTOT'],data_tmp['GB.Seqs'],
       pch=16,col="blue",ylim=c(-2,100),xlab="CvIndex",ylab="Appearances",main=paste("Filtro: ",a))
  ## Regresión paramétrica - poisson (sistema de recuento, con valores > 0, makes sense) - sin tocar Os
  RegZero_s_inf <- glm(GB.Seqs ~ CVTOT , data = data_tmp,family = poisson)
  RegZero_s_inf_theta <- glm(GB.Seqs ~ CVTOT,  data = data_tmp, family = quasipoisson(link = "log"))
```

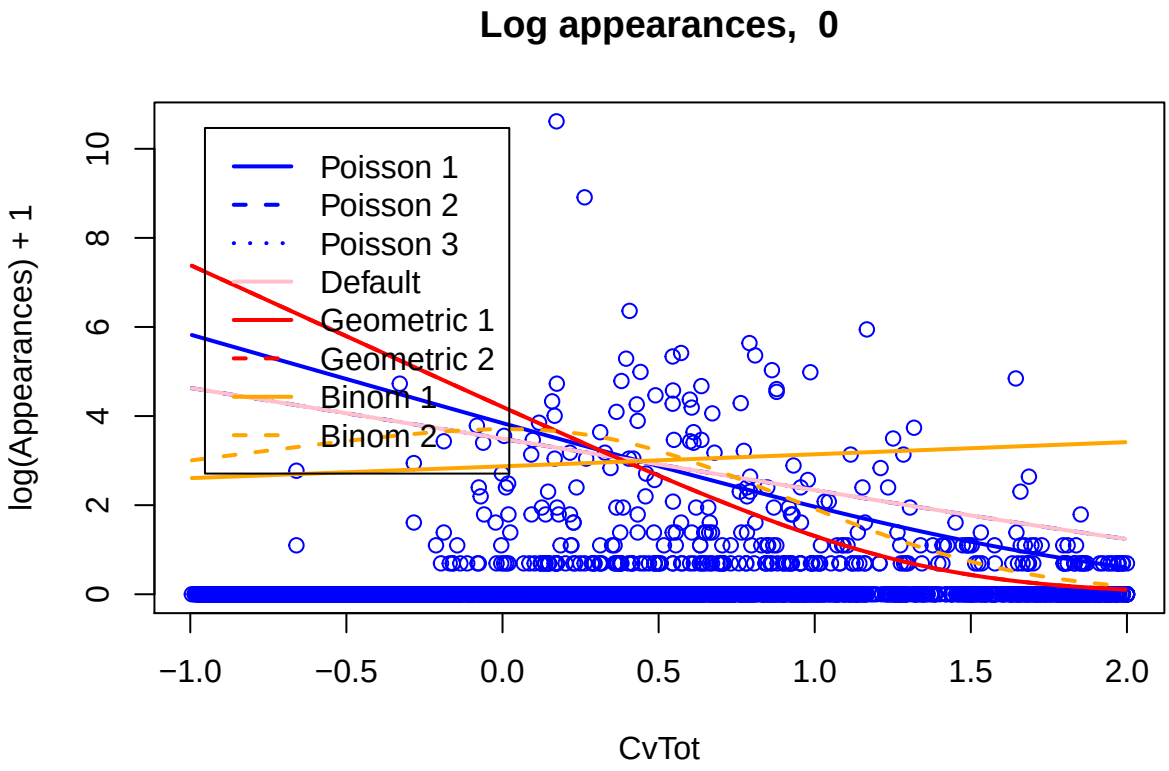
```r
summary(RegZero_s_inf_theta)
summary(RegZero_s_inf)
x<-seq(min(data_tmp[,'CVTOT']),max(data_tmp[,'CVTOT']),by=0.01)
Pr_s_inf<-predict(RegZero_s_inf, newdata=data.frame("CVTOT"=x),se.fit=TRUE)
lines(x,Pr_s_inf$fit,col="green",type="l",lwd=2)
# El ajuste va malament
f1 <- formula(GB.Seqs ~ CVTOT | 1 ) # Los 0's son estructurales
f2 <- formula(GB.Seqs ~ CVTOT | CVTOT) # Los 0s dependen de CVTOT
f3 <- formula(GB.Seqs ~ CVTOT)
RegZero_inf_sP_c1 <- zeroinfl(f1 ,data= data_tmp, dist="poisson", link = "logit")
RegZero_inf_sP_c2 <- zeroinfl(f2, data = data_tmp, dist="poisson", link = "logit")
RegZero_inf_sP_c3 <- zeroinfl(f3, data = data_tmp, dist="poisson", link = "logit")
RegZero_inf_sP_def <- zeroinfl(f3, data = data_tmp)
RegZero_inf_sP_geo1 <- zeroinfl(f1, data = data_tmp, dist = "geometric")
RegZero_inf_sP_geo2 <- zeroinfl(f2, data = data_tmp, dist = "geometric")
RegZero_inf_sP_ngbin_1 <- zeroinfl(f1, data = data_tmp, dist = "negbin")
RegZero_inf_sP_ngbin_2 <- zeroinfl(f2, data = data_tmp, dist = "negbin")
summary(RegZero_inf_sP_c1)
summary(RegZero_inf_sP_c2)
summary(RegZero_inf_sP_c3)
summary(RegZero_inf_sP_def)
summary(RegZero_inf_sP_geo1)
summary(RegZero_inf_sP_geo2)
summary(RegZero_inf_sP_ngbin_1)
summary(RegZero_inf_sP_ngbin_2)
Pr_inf_sP_c1<-predict(RegZero_inf_sP_c1,
                      newdata=data.frame("CVTOT"=x),se.fit=TRUE, type = "response")
Pr_inf_sP_c2<-predict(RegZero_inf_sP_c2,
                      newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_c3<-predict(RegZero_inf_sP_c3,
                      newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_def<-predict(RegZero_inf_sP_def,
                       newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_geo1<-predict(RegZero_inf_sP_geo1,
                       newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_geo2<-predict(RegZero_inf_sP_geo2,
                       newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_ngbin_1<-predict(RegZero_inf_sP_ngbin_1,
                       newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
Pr_inf_sP_ngbin_2<-predict(RegZero_inf_sP_ngbin_2,
                       newdata=data.frame("CVTOT"=x),se.fit=TRUE,type = "response")
# Poisson - blue
# Geometric - red
# Default - pink
# NegBin - orange
lines(x,Pr_inf_sP_c1,col="blue",type="l",lwd=2,lty=1)
lines(x,Pr_inf_sP_c2,col="blue",type="l",lwd=2,lty=2)
lines(x,Pr_inf_sP_c3,col="blue",type="l",lwd=2,lty=3)
lines(x,Pr_inf_sP_def,col="pink",type="l",lwd=2,lty=1)
# Curiosamente estas dos no van tan mal
lines(x,Pr_inf_sP_geo1,col="red",type="l",lwd=2,lty=1)
lines(x,Pr_inf_sP_geo2,col="red",type="l",lwd=2,lty=2)
# BinomNeg
```

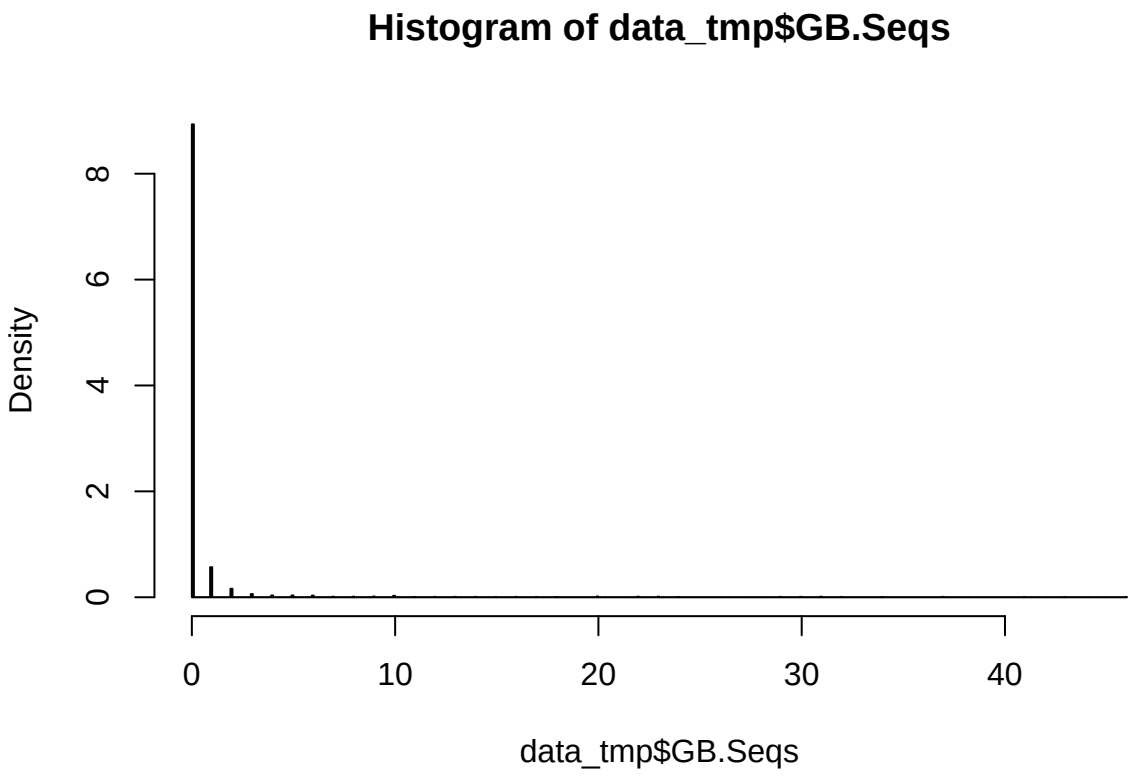
```
  lines(x,Pr_inf_sP_ngbin_1,col="orange",type="l",lwd=2,lty=1)
  lines(x,Pr_inf_sP_ngbin_2,col="orange",type="l",lwd=2,lty=2)
  legend("topleft", c("Poisson 1","Poisson 2","Poisson 3","Default"
    ,"Geometric 1","Geometric 2", "Binom 1", "Binom 2"),
         lty=c(1,2,3,1,1,2,1,2),col=c(rep("blue",3),"pink",rep("red",2),rep("orange",2)),
         ncol=1, lwd=rep(2,7), inset = .05)
  # Comparación de los modelos
  print(AIC(RegZero_inf_sP_c1,RegZero_inf_sP_c2,RegZero_inf_sP_c3,
      RegZero_inf_sP_def,RegZero_inf_sP_geo1,RegZero_inf_sP_geo2,RegZero_inf_sP_ngbin_1,
      RegZero_inf_sP_ngbin_2))
  # Una de alternativa suele ser aplicar el logaritmo de los datos
  data_tmp$GB.Seqs.log <- as.numeric(log(data_tmp$GB.Seqs + 1))
  summary(data_tmp$GB.Seqs.log)
  #windows()
  plot(data_tmp$CVTOT, data_tmp$GB.Seqs.log, col = "blue",
       xlab = "CvTot",ylab = "log(Appearances) + 1", main = paste("Log appearances, ",a))
  lines(x,log(Pr_inf_sP_c1+1),col="blue",type="l",lty =1, lwd = 2)
  lines(x,log(Pr_inf_sP_c2+1),col="blue",type="l",lty = 2, lwd = 2)
  lines(x,log(Pr_inf_sP_c3+1),col="blue",type="l",lty = 3, lwd = 2)
  lines(x, log(Pr_inf_sP_def+1), col="pink", type="l", lty = 1,lwd = 2)
  lines(x,log(Pr_inf_sP_geo1+1),col="red",type="l",lty = 1,lwd = 2)
  lines(x,log(Pr_inf_sP_geo1+1),col="red",type="l",lty = 2,lwd = 2)
  lines(x,log(Pr_inf_sP_ngbin_1+1),col="orange",type="l",lty = 1, lwd = 2)
  lines(x,log(Pr_inf_sP_ngbin_2+1),col="orange",type="l",lty = 2, lwd = 2)
  legend("topleft", c("Poisson 1","Poisson 2","Poisson 3","Default"
    ,"Geometric 1","Geometric 2", "Binom 1", "Binom 2"),
         lty=c(1,2,3,1,1,2,1,2),col=c(rep("blue",3),"pink",rep("red",2),rep("orange",2)),
         ncol=1, lwd=rep(2,7), inset = .05)
}
```

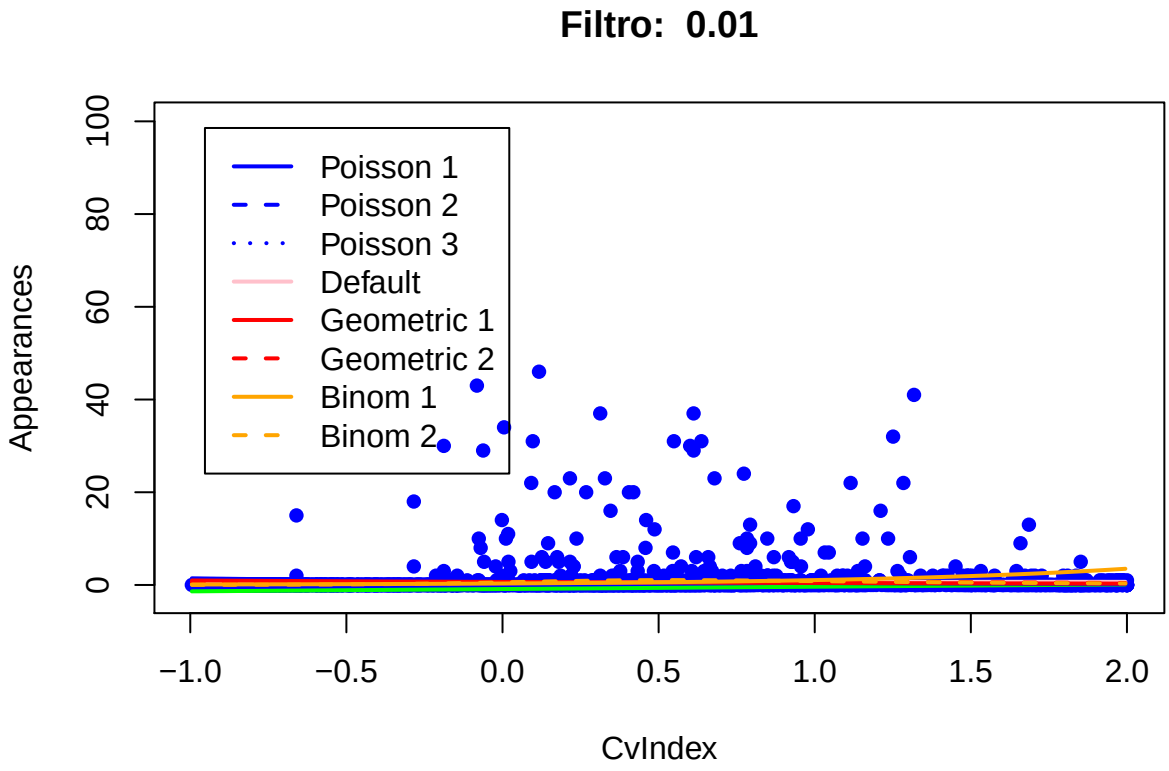

Histogram of data_tmp$GB.Seqs
Density
0
2
4
6
8
0
10000
20000
30000
40000
data_tmp$GB.Seqs



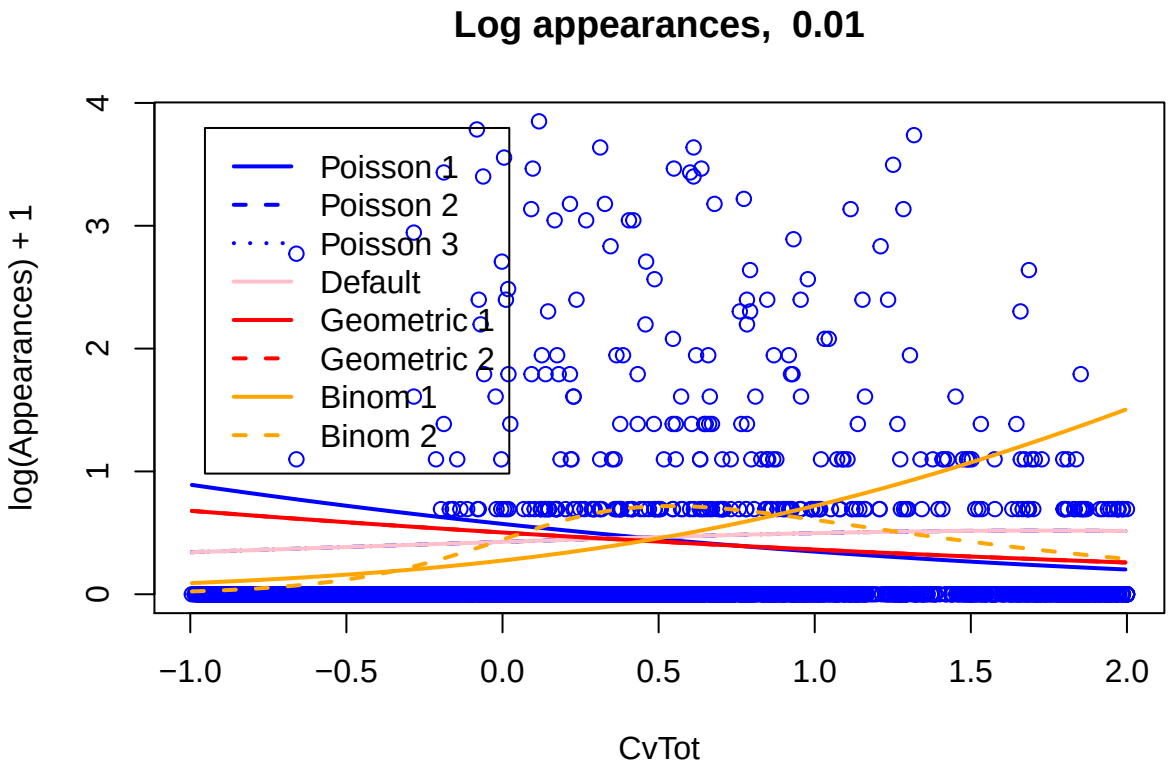

```
##                          df        AIC
## RegZero_inf_sP_c1         3 458073.741
## RegZero_inf_sP_c2         4 457906.137
## RegZero_inf_sP_c3         4 457906.137
## RegZero_inf_sP_def        4 457906.137
## RegZero_inf_sP_geo1       3   5619.575
## RegZero_inf_sP_geo2       4   5345.575
## RegZero_inf_sP_ngbin_1    4   4491.607
## RegZero_inf_sP_ngbin_2    5   4048.273
```

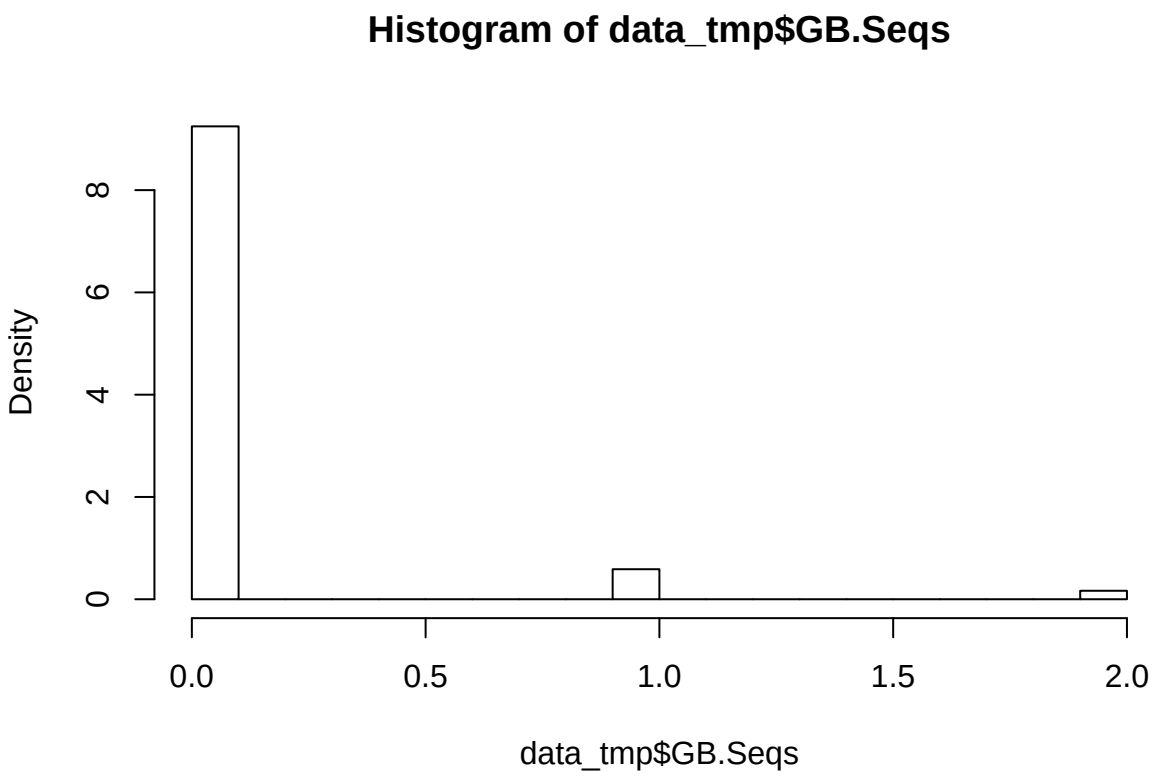

Log appearances, 0
Poisson 1
Poisson 2
Poisson 3
Default
Geometric 1
Geometric 2
Binom 1
Binom 2
log(Appearances) + 1
0
2
4
6
8
10
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
CvTot

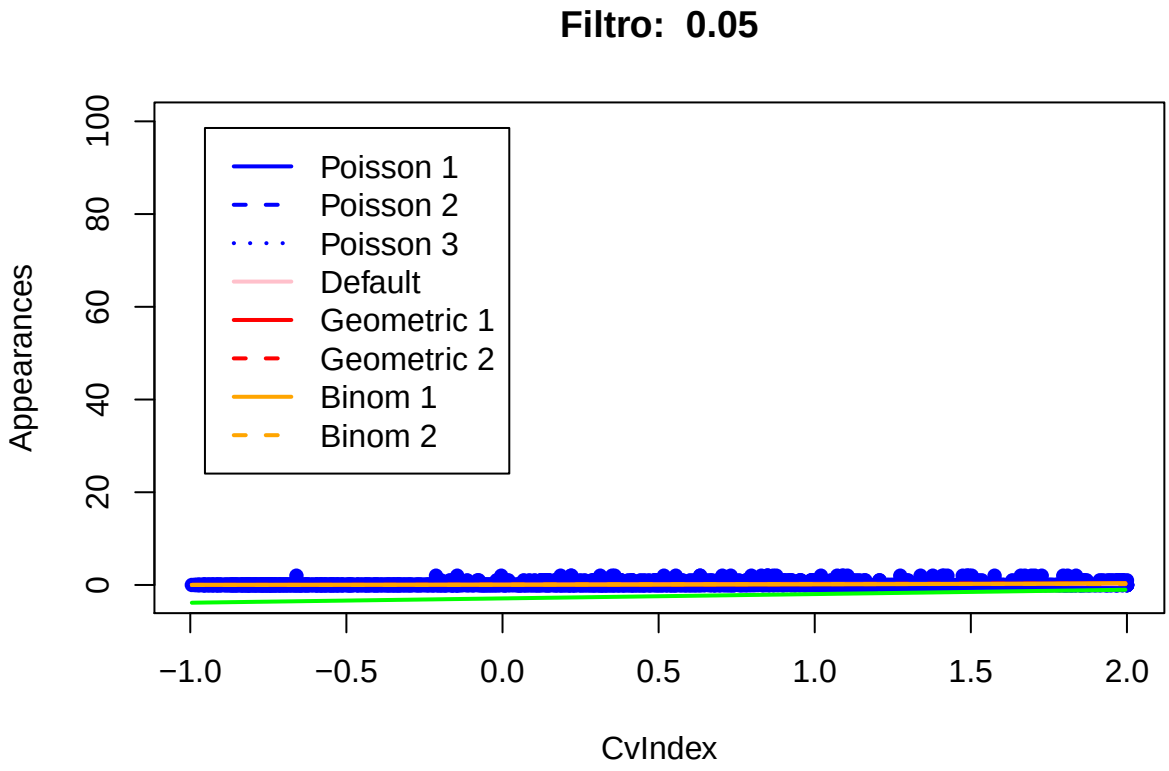

Histogram of data_tmp$GB.Seqs
Density
0
2
4
6
8
0
10
20
30
40
data_tmp$GB.Seqs



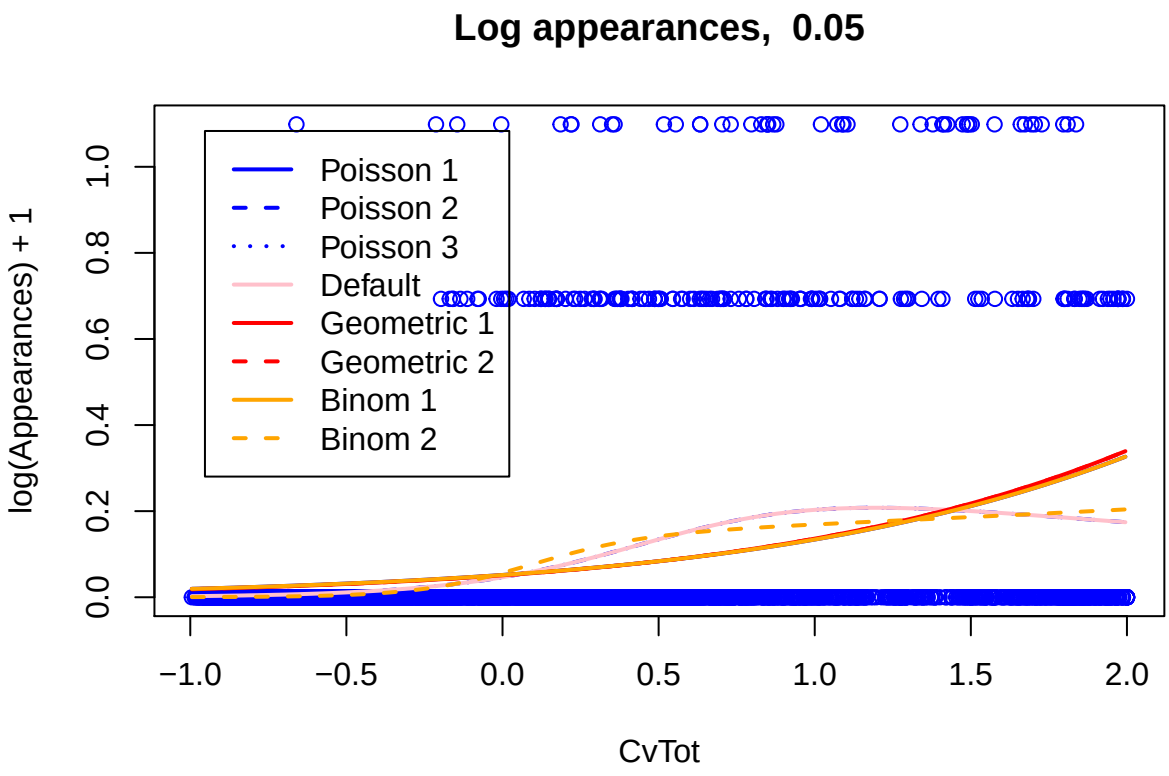

```
##                          df      AIC
## RegZero_inf_sP_c1         3 5233.026
## RegZero_inf_sP_c2         4 5049.318
## RegZero_inf_sP_c3         4 5049.318
## RegZero_inf_sP_def        4 5049.318
## RegZero_inf_sP_geo1       3 3595.306
## RegZero_inf_sP_geo2       4 3338.562
## RegZero_inf_sP_ngbin_1    4 3352.398
## RegZero_inf_sP_ngbin_2    5 3110.015
```


Log appearances, 0.01
log(Appearances) + 1
CvTot
Poisson 1
Poisson 2
Poisson 3
Default
Geometric 1
Geometric 2
Binom 1
Binom 2
0
1
2
3
4
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Histogram of data_tmp$GB.Seqs
Density
0
2
4
6
8
0.0
0.5
1.0
1.5
2.0
data_tmp$GB.Seqs




```
##                          df      AIC
## RegZero_inf_sP_c1         3 1654.677
## RegZero_inf_sP_c2         4 1596.103
## RegZero_inf_sP_c3         4 1596.103
## RegZero_inf_sP_def        4 1596.103
## RegZero_inf_sP_geo1       3 1658.441
## RegZero_inf_sP_geo2       4 1635.717
## RegZero_inf_sP_ngbin_1    4 1656.678
## RegZero_inf_sP_ngbin_2    5 1589.449
```



GAMLSS

```
## Modelos paramétricos via GAMLSS (Generalize Aditive models for location scale and shape)
l_a <- c(0.0)
for (a in l_a){
  # Eliminamos datos fuera de lugar
  lower_bound <- quantile(y, 0.00);lower_bound
  upper_bound <- quantile(y, 1 - a);upper_bound

  outlier_ind <- which(y < lower_bound | y > upper_bound)
  if (!is.na(outlier_ind[1])){
    data_tmp <- data_filtered[-outlier_ind,]
  }else{
    data_tmp <- data_filtered}
  ## Estudio de las distribuciones
  #windows()
  op <- par(mfrow = c(2, 3))
  dPO <- histDist(data_tmp$GB.Seqs, "PO",
                  main = "PO", trace = FALSE, xlim=c(0,20))     # Poisson
  dNBI <- histDist(data_tmp$GB.Seqs, "NBI",
                   main = "NBI", trace = FALSE,xlim=c(0,20))    # Binomial negativa tipo 1
  dPIG <- histDist(data_tmp$GB.Seqs, "PIG",
                   main = "PIG", trace = FALSE,xlim=c(0,20))    # Poisson inverse gaussian
  dSI <- histDist(data_tmp$GB.Seqs, "SICHEL",
                  main = "SICHEL", trace = FALSE, xlim=c(0,20)) # Sichel? Inverse Gaussian distribution
  dZIP <- histDist(data_tmp$GB.Seqs, "ZIP",
                   main = "ZIP", trace = FALSE, xlim=c(0,20))  # Zero inflated poisson
```

```r
dZIP2 <- histDist(data_tmp$GB.Seqs, "ZIP2",
                  main = "ZIP2", trace = FALSE, xlim=c(0,20)) # Zero inf. pois. 2
par(op)
AIC(dPO, dNBI, dPIG, dSI, dZIP, dZIP2)

## En base al AIC nos quedaríamos con dPIG que es un modelo de poisson pero descartariamos claramente
## inflados.

## A través del ajuste de gamlss ajustamos distintos modelos con diferentes familias de funciones
## Eliminapos PO, ZIP y ZIP2 dado que sus AIC están fuera de lugar en comparación a los otros 3
fam.gamlss <- c("NBI", "NBII", "PIG", "DEL", "SICHEL")
m.l <- m.q <- m.s <- m.lq <-list()
## Ajuste lineal con enlaces
for (i in 1:5) {
  m.l[[fam.gamlss[i]]] <- gamlss(GB.Seqs~ CVTOT, data=na.omit(data_tmp),
                                 family = fam.gamlss[i], n.cyc = 20, trace = FALSE)$aic
}
## Regresión polinómico de grado 2 con los enlaces
for (i in 1:5) {
  m.q[[fam.gamlss[i]]] <-GAIC(  gamlss(GB.Seqs ~ poly(CVTOT,2),
                                       data=data_tmp, family = fam.gamlss[i],
                                       n.cyc = 20, trace = FALSE))
}
## Smooth cubic splines (suavización via splines cúbicos)
for (i in 1:5) {
  m.s[[fam.gamlss[i]]] <-GAIC(  gamlss(GB.Seqs ~ cs(CVTOT),
                                       data=na.omit(data_tmp), family = fam.gamlss[i],
                                       n.cyc = 20, trace = FALSE))
}
print(unlist(m.l))
print(unlist(m.q))
print(unlist(m.s))

## Igual que se visualizaba anteriormente, los mejores resultados vienen de SICHEL y de PIG (Igual
# a como ocurre en la versión de Manuel)

# sigma.fo permite otorgar una formula para ajustar un modelo para el parámetro sigma (desviación)
m.ql <- list()
for (i in 1:5) {
  m.ql[[fam.gamlss[i]]] <- GAIC(gamlss(GB.Seqs ~ poly(CVTOT, 2),
                                       data=na.omit(data_tmp), sigma.fo = ~CVTOT,
                                       family = fam.gamlss[i], n.cyc = 60, trace = FALSE))
}
print(unlist(m.ql))

# Nuevamente apenas hay mejoría.
## En cuanto al "mejor" modelo:
##  PIG - Poisson Inverse Gaussian o SICHEL (no existen diferencias significativas en el AIC)
##  El ajuste vía smoothing spline - aunque el ajuste por regresión polinómico local (grado 2)
# tampoco queda descartado. - Nuevamente lo mismo que para Manuel.

## En base a esto vamos a evaluar los resultados via plot
# Lineal + ajuste de sd
```

```
  mL <- gamlss(GB.Seqs~CVTOT , data=na.omit(data_tmp), family =PIG,
               sigma.fo=~CVTOT, n.cyc = 20, trace = FALSE)
  # Cuadrático + ajuste de sd
  mC <- gamlss(GB.Seqs~CVTOT+I(CVTOT^2) , data=na.omit(data_tmp), family =PIG,
               sigma.fo=~CVTOT, n.cyc = 20, trace = FALSE)
  # Vía smoothing spline + ajuste de sd
  mS <- gamlss(GB.Seqs~cs(CVTOT) , data=na.omit(data_tmp), family =PIG,
               sigma.fo=~CVTOT, n.cyc = 20, trace = FALSE)

  # Obtenemos predicciones
  mL.p <- predict(mL, what = "mu", newdata=data.frame("CVTOT"=x),
                  type = "response", terms = NULL, se.fit = FALSE, data = na.omit(data_tmp))
  mC.p <- predict(mC, what = "mu", newdata=data.frame("CVTOT"=x),
                  type = "response", terms = NULL, se.fit = FALSE, data = na.omit(data_tmp))
  mS.p <- predict(mS, what = "mu", newdata=data.frame("CVTOT"=x),
                  type = "response", terms = NULL, se.fit = FALSE, data = na.omit(data_tmp))
  # Ploteamos
  #windows()
  par(mfrow = c(1,1))
  plot(data_tmp$CVTOT,data_tmp$GB.Seqs,pch=16,ylim=c(0,200), xlab="CVTOT",
      ylab="Appearances",main=paste("Filtro: ",a))
  lines(x,mL.p,col="red",type="l",lwd=2)
  lines(x,mC.p,col="blue",type="l",lwd=2)
  lines(x,mS.p,col="orange",type="l",lwd=2)
  legend("topright", c("PIG.Lineal","PIG.Cuadrático","PIG.Splines"),
         lty=c(2,2,2),col=c("red","blue","orange"), ncol=1, lwd=rep(2,3), inset = .05, cex=0.75)
}
```

```
## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in qnorm(uval): NaNs produced

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

## Warning in RS(): Algorithm RS has not yet converged

##      NBI     NBII      PIG      DEL   SICHEL
## 4489.614 4386.073 3912.581 4491.618 3914.292
##      NBI     NBII      PIG      DEL   SICHEL
## 4240.618 4244.760 3759.323 4242.630 3760.488
##        NBI       NBII        PIG        DEL     SICHEL
##   4165.663   4235.826   3755.538 158389.361   3756.902

## Warning in RS(): Algorithm RS has not yet converged
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

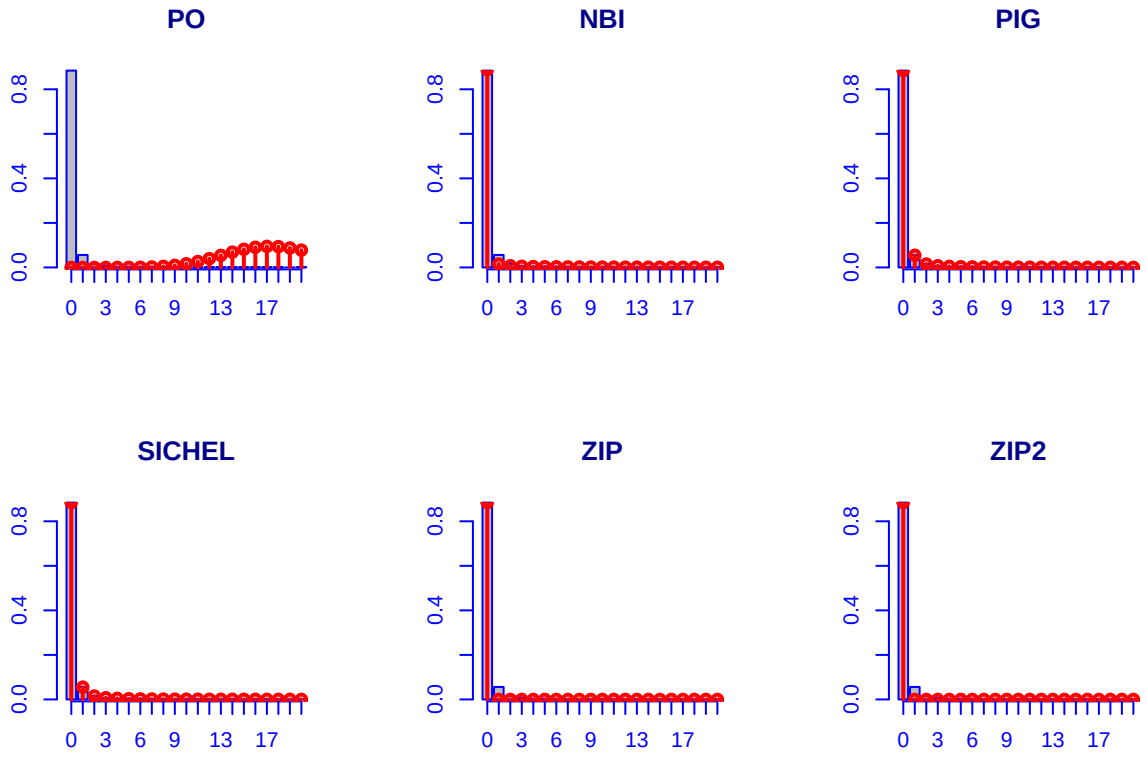
```
##      NBI     NBII      PIG      DEL   SICHEL
## 4052.020 4006.081 3760.435 4054.024 3761.977
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

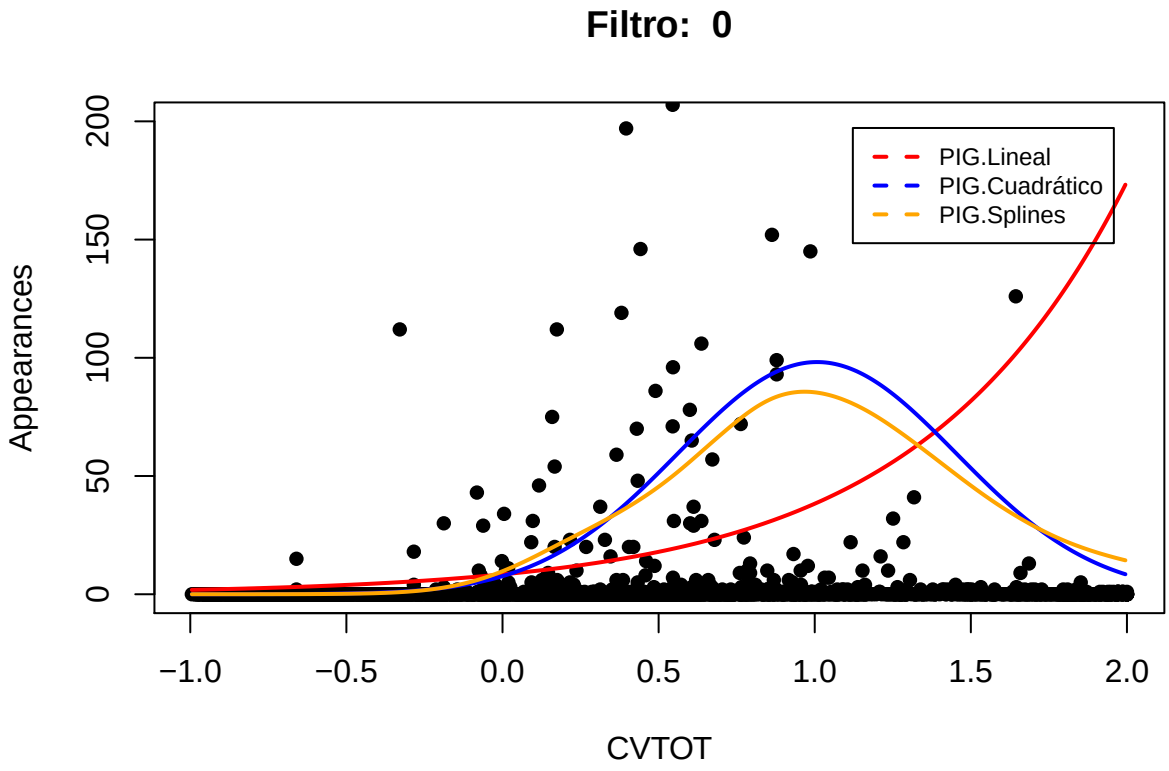
```
## Warning in RS(): Algorithm RS has not yet converged
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
## Warning in predict.gamlss(mS, what = "mu", newdata = data.frame(CVTOT = x), : There is a discrepancy
##  used to achieve 'safe' predictions
##
```




```
## new prediction
```



Regresión cuantil

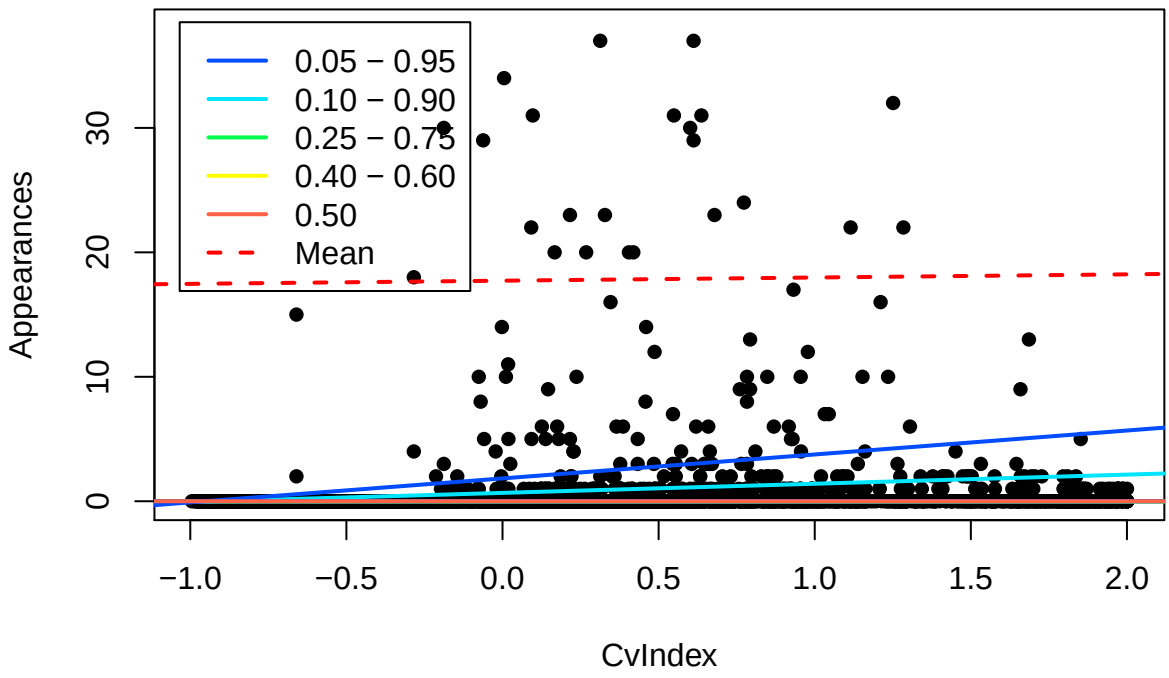
```
## Regresión quantil
# Vector de cuantiles seleccionados (siempre un número impar con la mediana en medio):
quant <- c(0.05, 0.10, 0.25, 0.40, 0.50, 0.60, 0.75, 0.90, 0.95)
L <- (length(quant)/2)+0.5

# Datos para figuras
quant2 <- as.character(format(quant,digits=2))
colores <- topo.colors(L)[L:1]
leg <- numeric(length=(L+1))
for ( i in 1:(L-1) )  leg[i] <- paste(quant2[i]," - ",quant2[2*L-i],sep="")
leg[L] <- "0.50"; leg[L+1] <- "Mean"

# linear model
lin.mod <- rq(GB.Seqs~ CVTOT, data=data_filtered, tau=quant ,method="br")
summary(lin.mod,se="boot")   # Tantos ceros implican que las regresiones cuantil tengan pendiente no si
```

```
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.05
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
```

```
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.1
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.25
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.4
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.5
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.6
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
```

```
##     method = "br")
##
## tau: [1] 0.75
##
## Coefficients:
##             Value Std. Error t value Pr(>|t|)
## (Intercept)   0     0        NaN     NaN
## CVTOT         0     0        NaN     NaN
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.9
##
## Coefficients:
##             Value    Std. Error t value  Pr(>|t|)
## (Intercept)  0.67107  0.04986   13.45918  0.00000
## CVTOT        0.73421  0.05468   13.42857  0.00000
##
## Call: rq(formula = GB.Seqs ~ CVTOT, tau = quant, data = data_filtered,
##     method = "br")
##
## tau: [1] 0.95
##
## Coefficients:
##             Value   Std. Error t value Pr(>|t|)
## (Intercept) 1.83387 0.46269    3.96351 0.00008
## CVTOT       1.92431 0.48888    3.93614 0.00008
```

```
plot(data_filtered$CVTOT,data_filtered$GB.Seqs,pch=16,col=1,ylim=c(0,38),xlab="CvIndex",ylab="Appearanc
for (i in 1:(L-1))
  {
  abline(rq(GB.Seqs~ CVTOT, data=data_filtered,tau=quant[L-i] ,method="br"),col=colores[i], lty=1, lwd =
  abline(rq(GB.Seqs~ CVTOT, data=data_filtered,tau=quant[L+i] ,method="br"),col=colores[i], lty=1,lwd =1
  }
abline(rq(GB.Seqs~ CVTOT, data=data_filtered,tau=quant[L] ,method="br"),col="tomato",lty=1,lwd=2)
abline(lm(GB.Seqs~ CVTOT, data=data_filtered),col="red", lty=2, lwd = 2)
legend("topleft",leg, col=c(colores[(L-1):1],"tomato","red"), lty=c(rep(1,L),2), inset=0.025,lwd = 2)
```



Los resultados son bastante esperables sobretodo teniendo en cuenta la cantidad de 0's que tienen los datos de partida. Como se puede ver salvo para casos de cuantiles muy extremos 0.05-0.95 o 0.1-0.9, no se ve una pendiente significativa.

Análisis no paramétricos

Análisis via curvas spline:

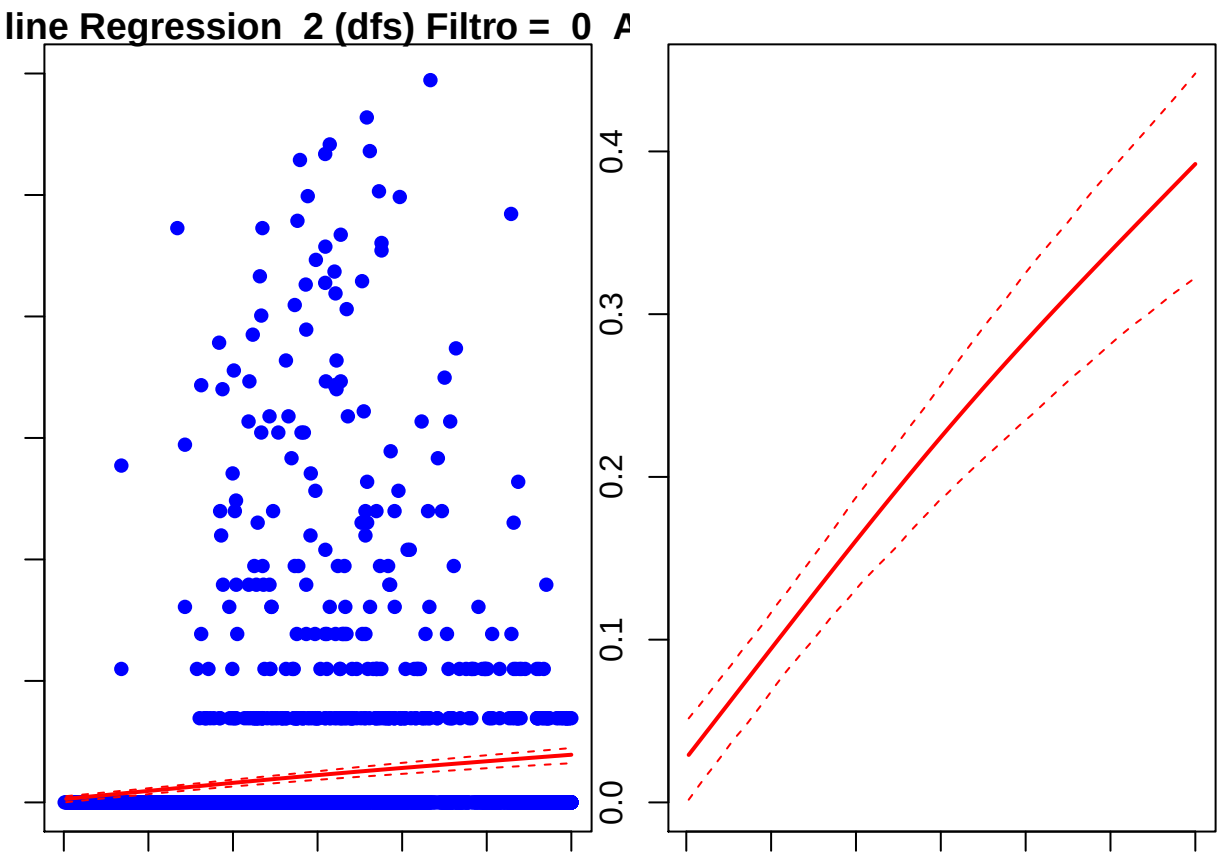
```
splines_adjust<-function(DataSub,columna_y, columna_x, df,m,alpha,B,mainT, type = 'regular') {
  x_l <- DataSub[,columna_x]
  y_l <- DataSub[,columna_y]
  if (type == 'b' || type == 'p'){
    ind_x <- sort(x_l, index.return = T)$ix
    x_l <- x_l[ind_x]
    y_l <- y_l[ind_x]
  }
    # Fit spline to data, with cross-validation to pick lambda
  if (type == 'regular'){
      spl<-smooth.spline(x=x_l,y=y_l,df=df)
  }
  if (type == 'b')
  {
    nodos <- seq(min(x_l), max(x_l), by = m)
    x_diseno <- bs(x = x_l, knots = nodos)
    spl <- lm(y_l~x_diseno)
  }
  if (type == 'p')
```

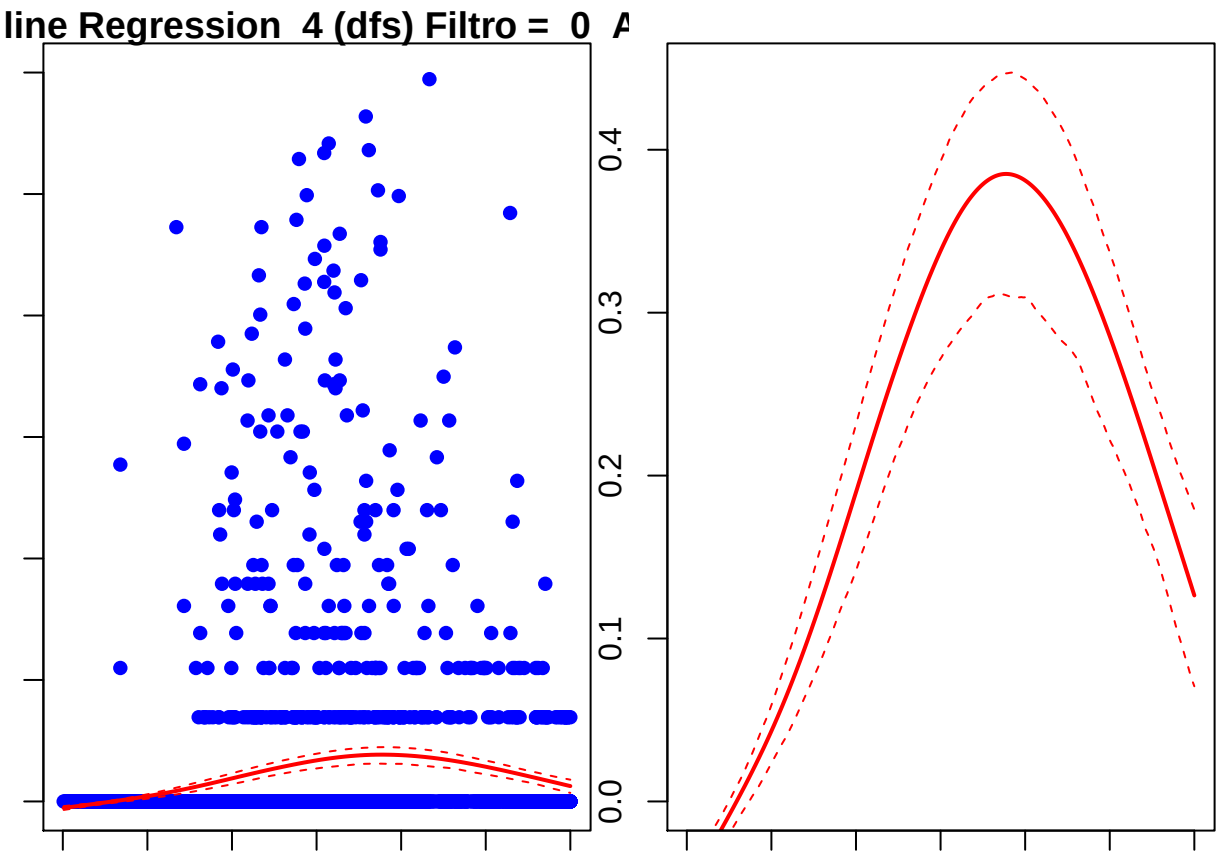
```
  {
    spl <- gam(y_l ~ s(x_l, k=df, bs = "cr"))
    #gam.check(spl)
  }
    Grid <- seq(from=min(x_l),to=max(x_l),length.out=m)
    if (type == 'regular'){
      Main<-predict(spl,x=Grid)$y
      Estim<-matrix(nrow=m,ncol=B)}
    if (type == 'b' || type == 'p'){
      Grid <- x_l
      Main<-predict(spl)
      Estim <- matrix(nrow = dim(DataSub)[1], ncol = B)}
    for (i in 1:B) {
        DataRes<-DataSub[sample(dim(DataSub)[1],replace=TRUE),c(columna_x,columna_y)]
        x_loc <- DataRes[,columna_x]
    y_loc <- DataRes[,columna_y]
        if (type == 'b' || type == 'p')
        {
          ind_x <- sort(x_loc, index.return = T)$ix
      x_loc <- x_loc[ind_x]
      y_loc <- y_loc[ind_x]
        }
        # Fit spline to data, with cross-validation to pick lambda
        if (type == 'regular')
          spl <- smooth.spline(x=x_loc,y=y_loc,df=df)
        if (type == 'b'){
          nodos <- seq(min(x_loc), max(x_loc), by = m)
      x_loc_diseno <- bs(x = x_loc, knots = nodos)
      spl <- lm(y_loc~x_diseno)
        }
    if (type == 'p')
    {
      spl <- gam(y_loc ~ s(x_loc, k=df, bs = "cr"))
    }
        if (type == 'regular'){
          Grid <- seq(from=min(x_loc),to=max(x_loc),length.out=m)
          Estim[,i]<-predict(spl,x=Grid)$y}
        if (type == 'b' || type == 'p'){
          Estim[,i]<-predict(spl)}
    }
    Lower<-2*Main-apply(Estim,1,quantile,probs=1-alpha/2)
    Upper<-2*Main-apply(Estim,1,quantile,probs=alpha/2)
    plot(x_l,y_l,pch=16,col="blue",ylim=c(0,6),xlab="CvIndex",ylab="log(Appearances + 1)",
        main=mainT)
    lines(Grid,Main,col="red",lwd=2)
    lines(Grid,Lower,col="red",lty=2)
    lines(Grid,Upper,col="red",lty=2)
    plot(Grid,Main,col="red",type="l",lwd=2,ylim = c(0,max(Upper)),xlab="CvIndex",ylab="log(Appearances
    lines(Grid,Lower,col="red",lty=2)
    lines(Grid,Upper,col="red",lty=2)
    return(spl)}
############################################################
```

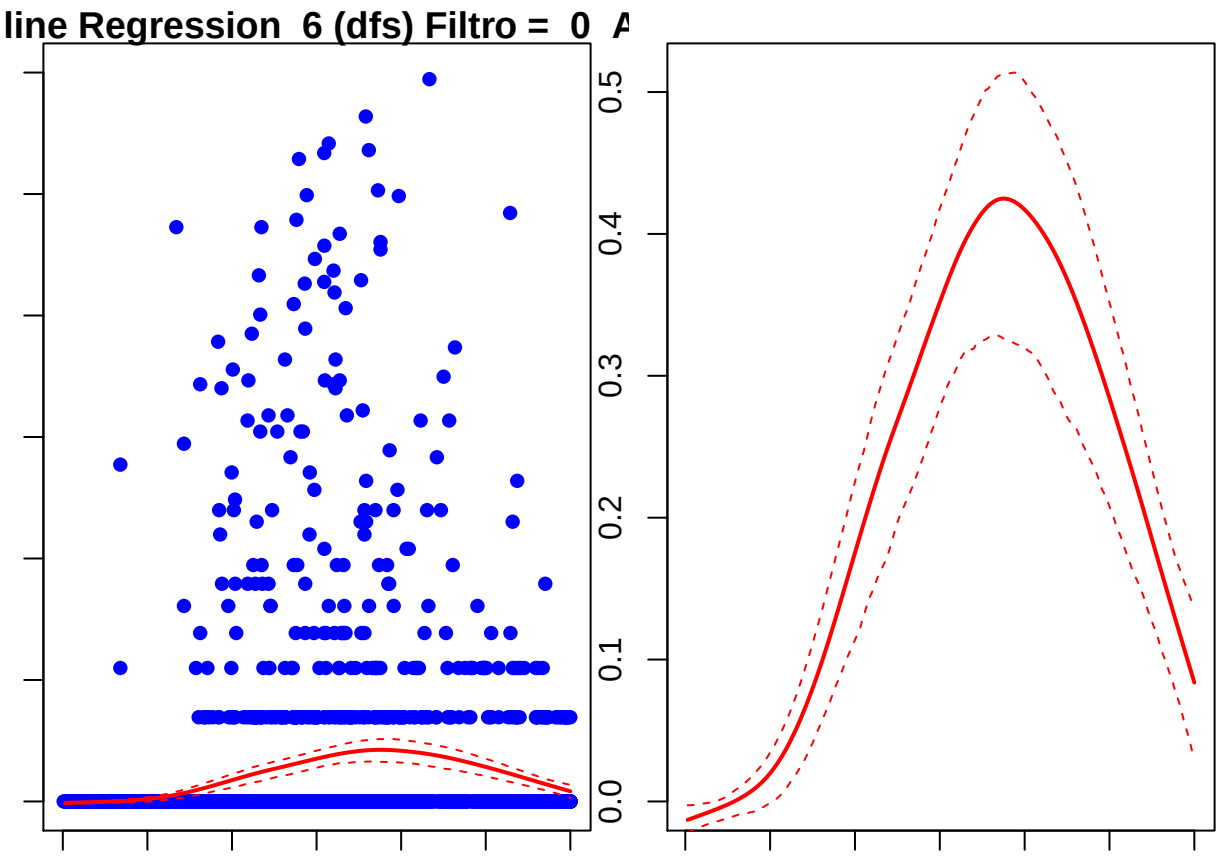
Estudios a través de curvas Spline: * Smoothing spline: * Problema: selección de los grados de libertad, en

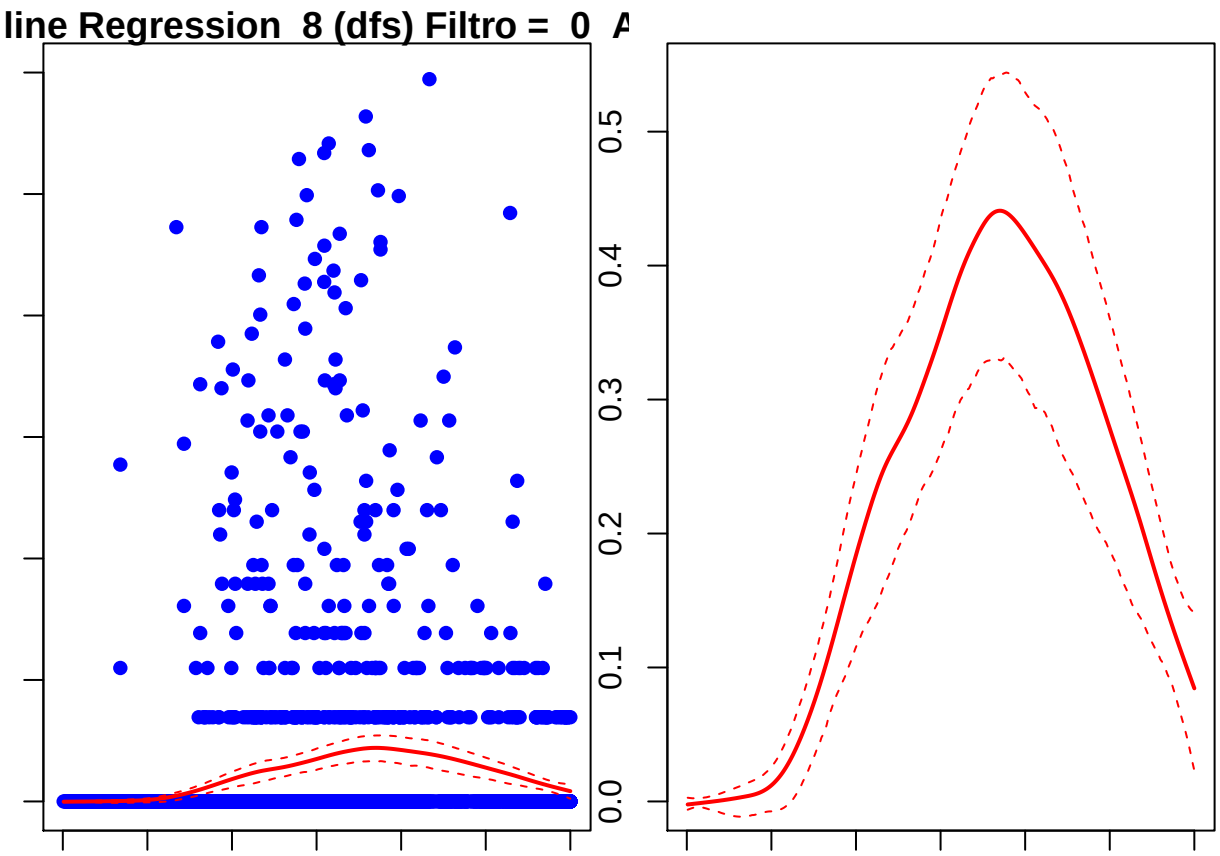
este caso no parece muy problemático dado que con 4 ya obtenemos un buen ajuste y su incremento solo implica un crecimiento del sobreajuste y una menor suavidad de la curva.

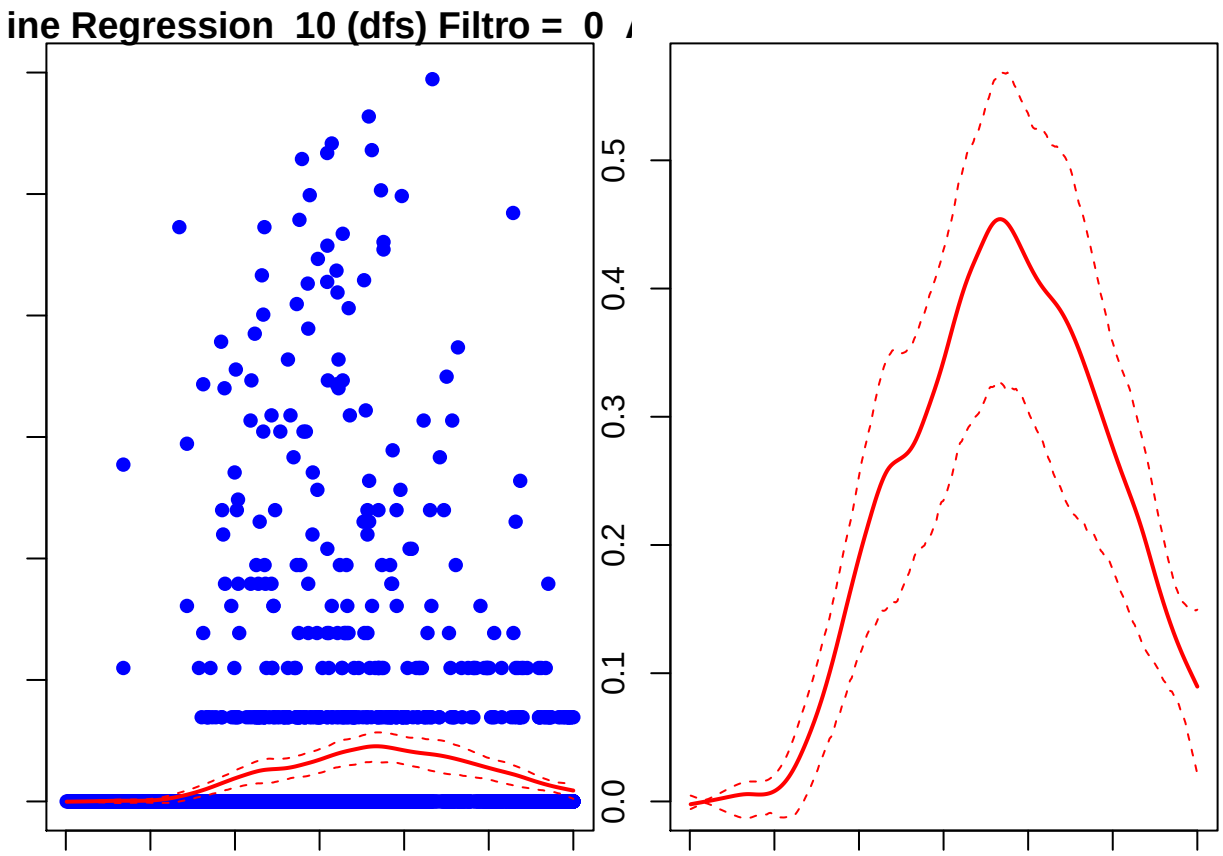
```
par(mar=c(1,1,1,1))
## Modelos no paramétricos:
#### Splines.
l_a <- c(0.0, 0.01, 0.05)
for (a in l_a)
{
  # Eliminamos datos fuera de lugar
  lower_bound <- quantile(y, 0.00);lower_bound
  upper_bound <- quantile(y, 1 - a);upper_bound

  outlier_ind <- which(y < lower_bound | y > upper_bound)
  if (!is.na(outlier_ind[1])){
    data_tmp <- data_filtered[-outlier_ind,]
  }else{
    data_tmp <- data_filtered}
  dfs <- seq(2, 10, by = 2)
  #windows()
  par(mfrow=c(1,2))
  for (df in dfs)
  {
    splines_adjust(data_tmp, 'Log.GB.Seqs', 'CVTOT', df,300,0.025,1000,paste("Smoothing Spline Regressi
  }
}
```

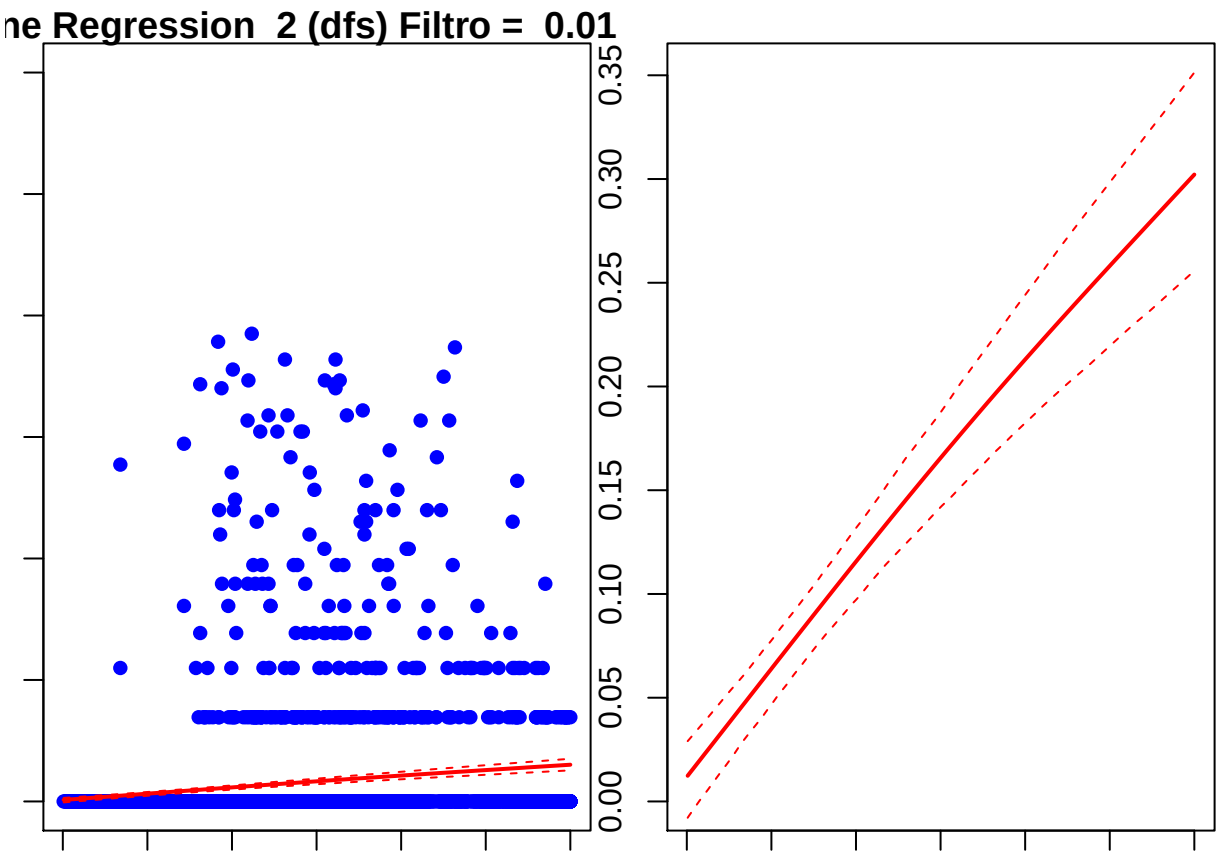


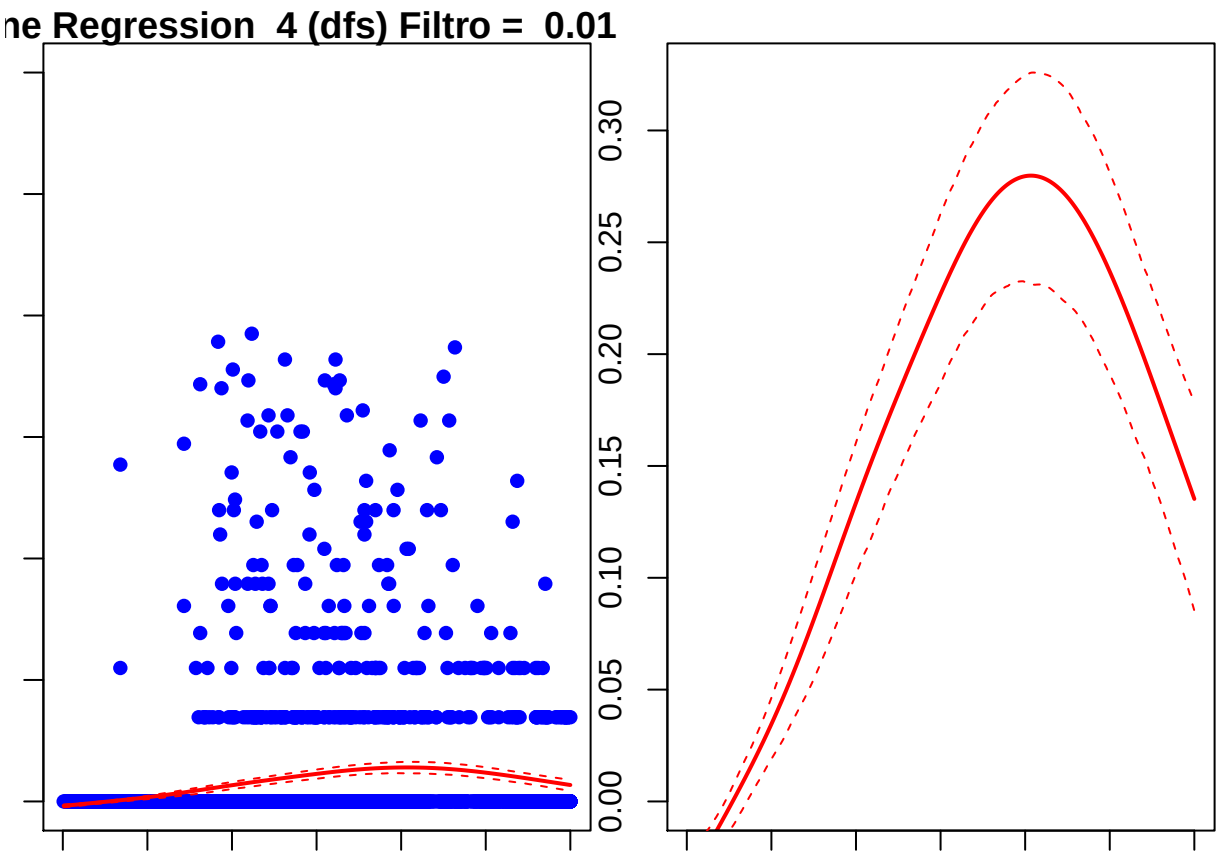

line Regression 4 (dfs) Filtro = 0 A
0.0
0.1
0.2
0.3
0.4

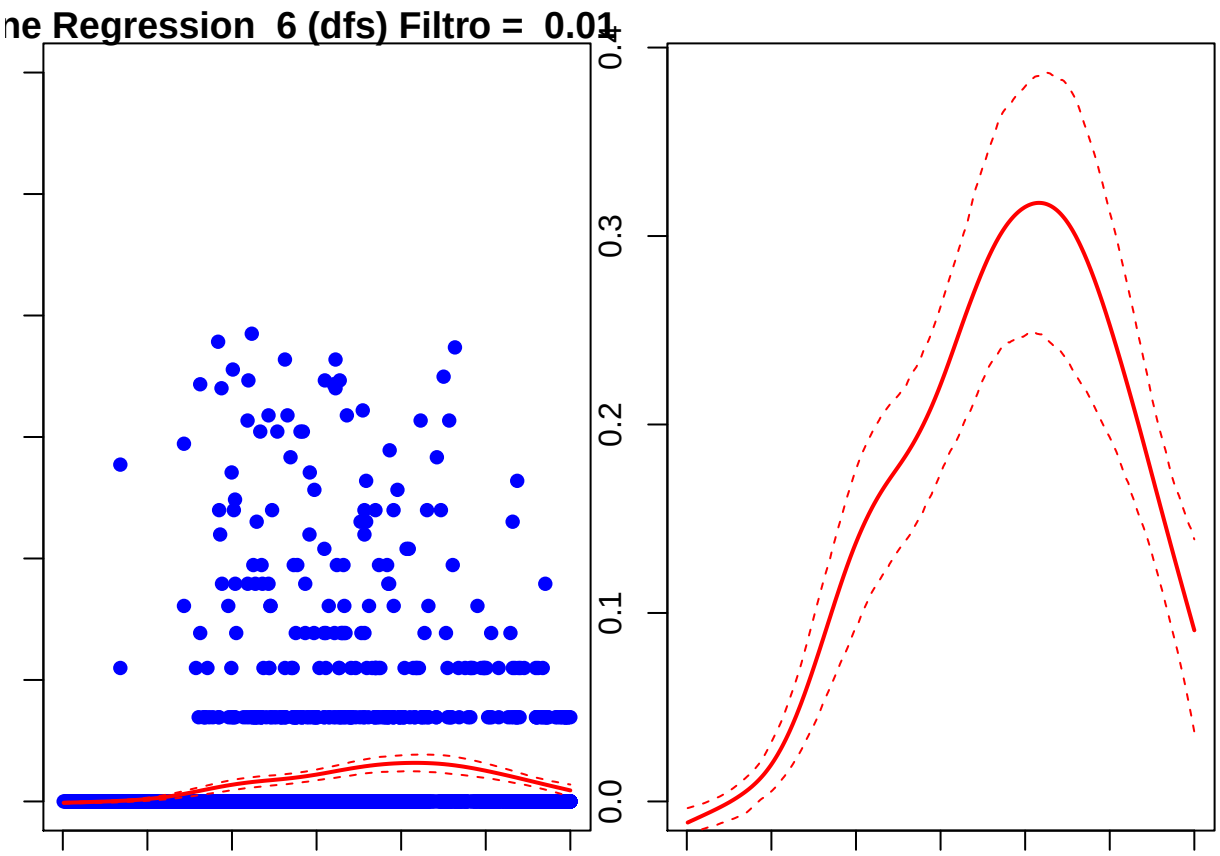

line Regression 6 (dfs) Filtro = 0 A
0.0
0.1
0.2
0.3
0.4
0.5

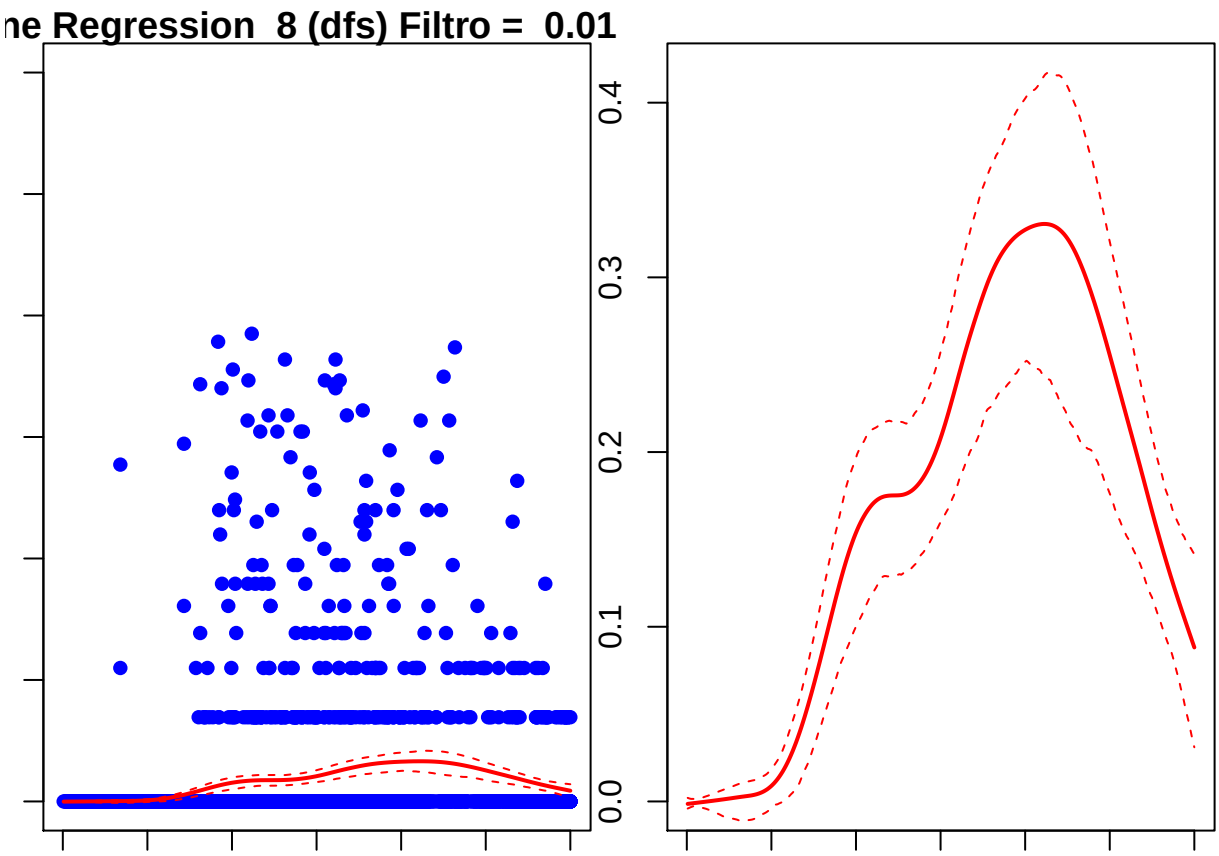

line Regression 8 (dfs) Filtro = 0 A
0.0
0.1
0.2
0.3
0.4
0.5

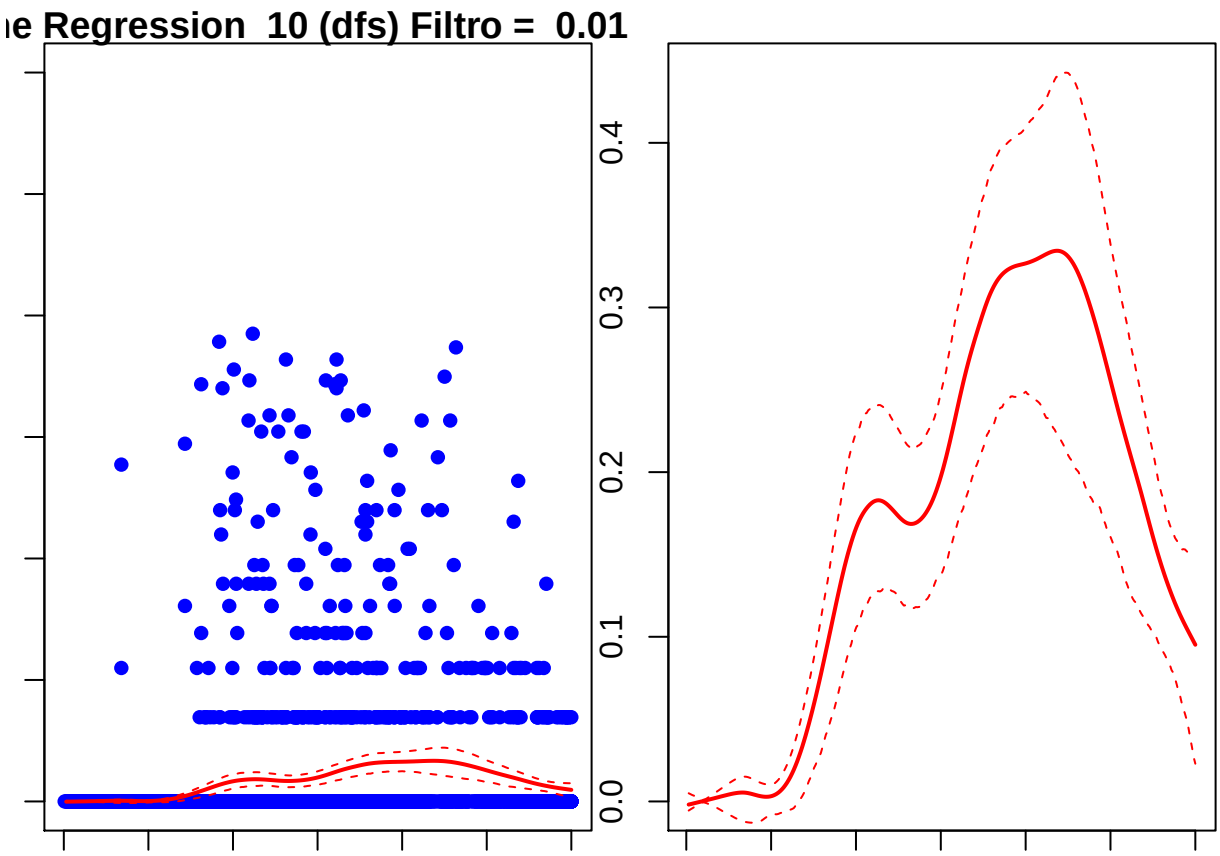

ine Regression 10 (dfs) Filtro = 0
0.0
0.1
0.2
0.3
0.4
0.5

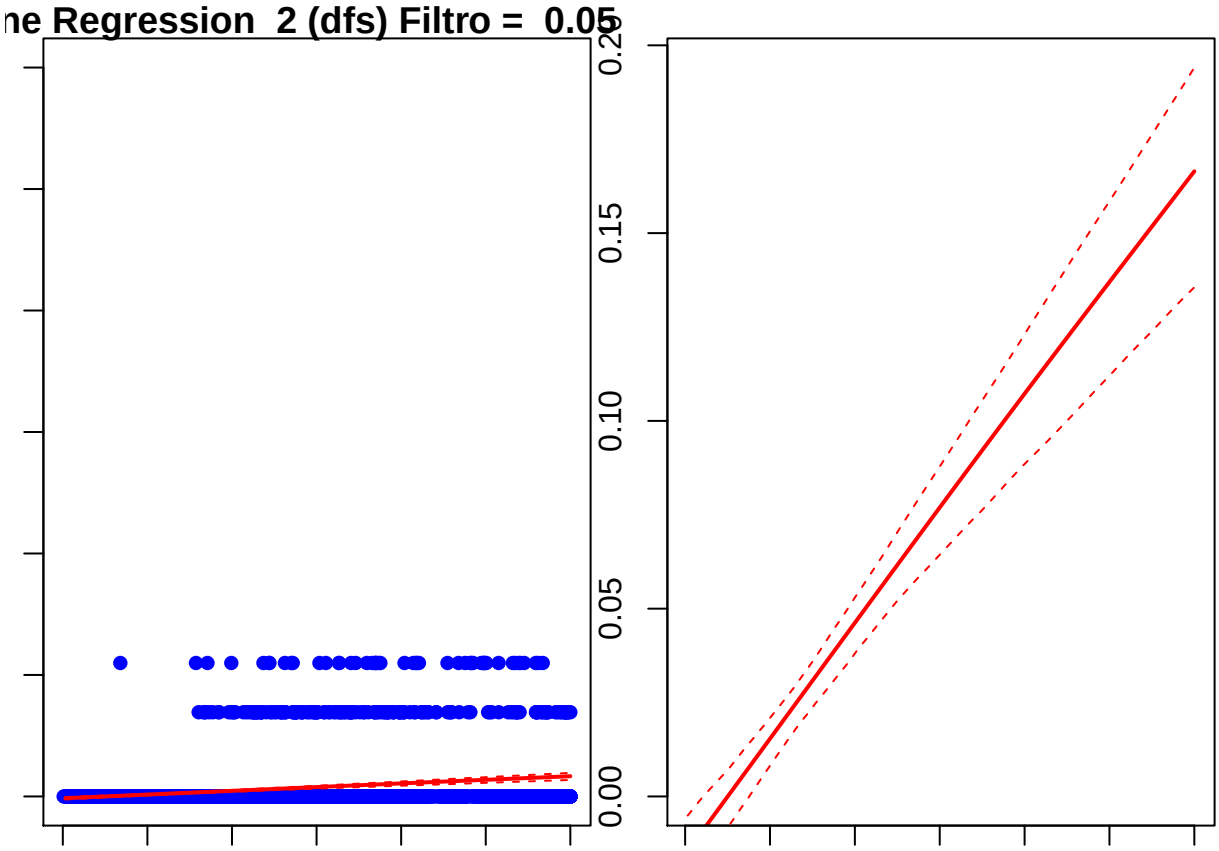

ne Regression 2 (dfs) Filtro = 0.01
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35

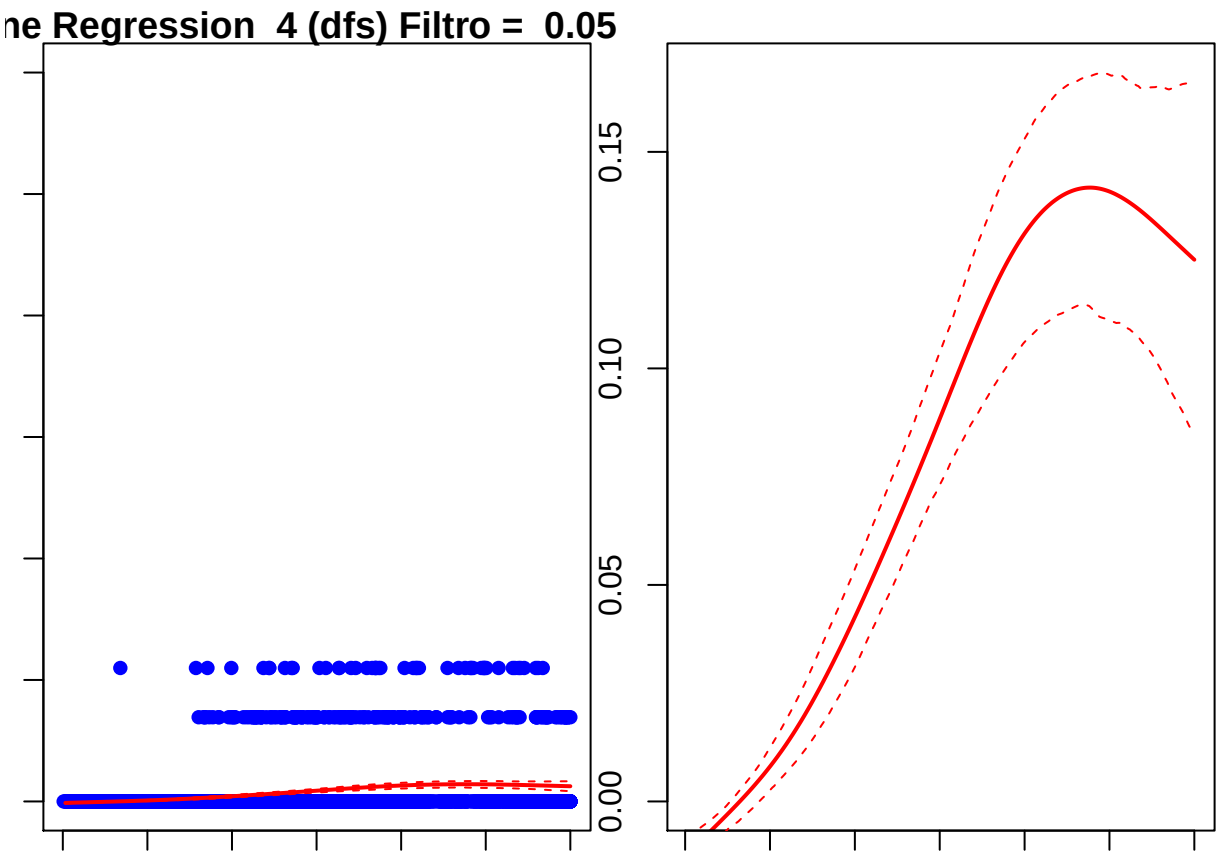

ne Regression 4 (dfs) Filtro = 0.01
0.00
0.05
0.10
0.15
0.20
0.25
0.30

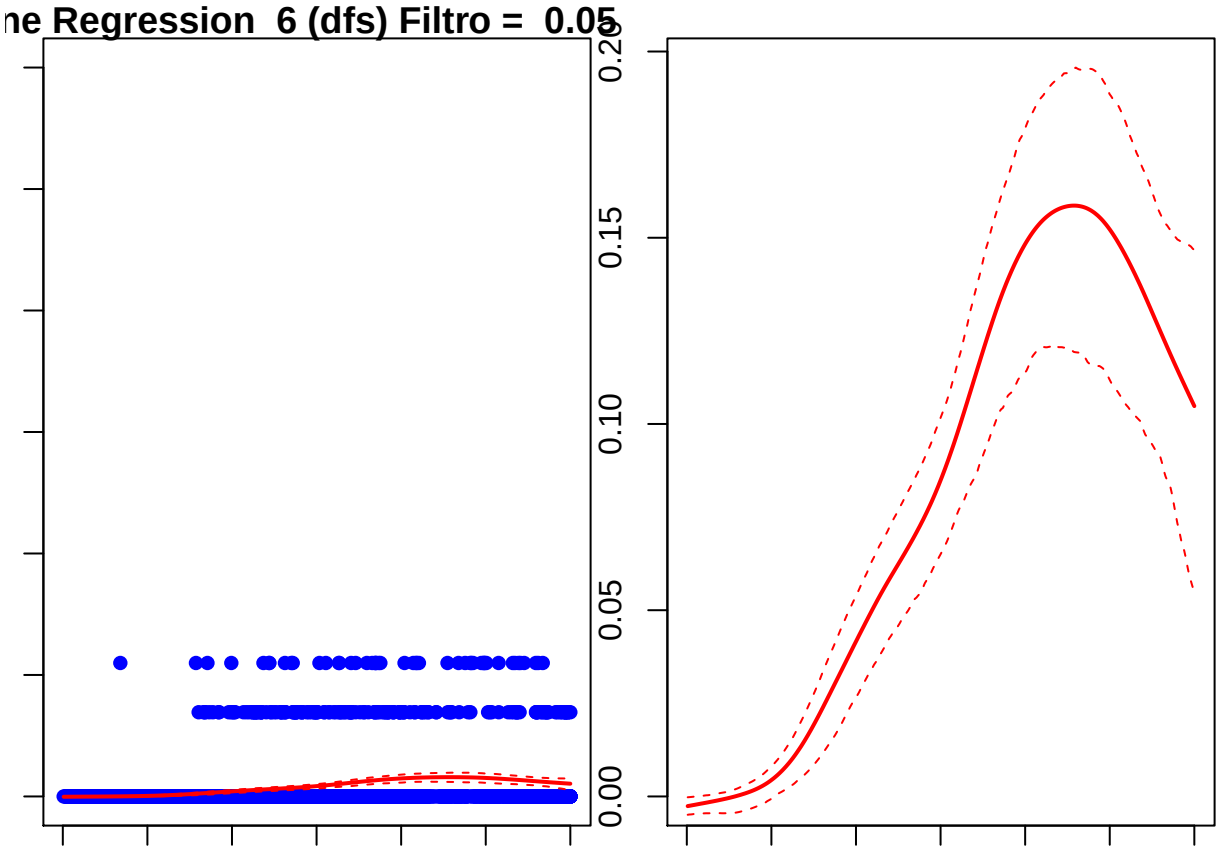

ne Regression 6 (dfs) Filtro = 0.01
0.0
0.1
0.2
0.3
0.4

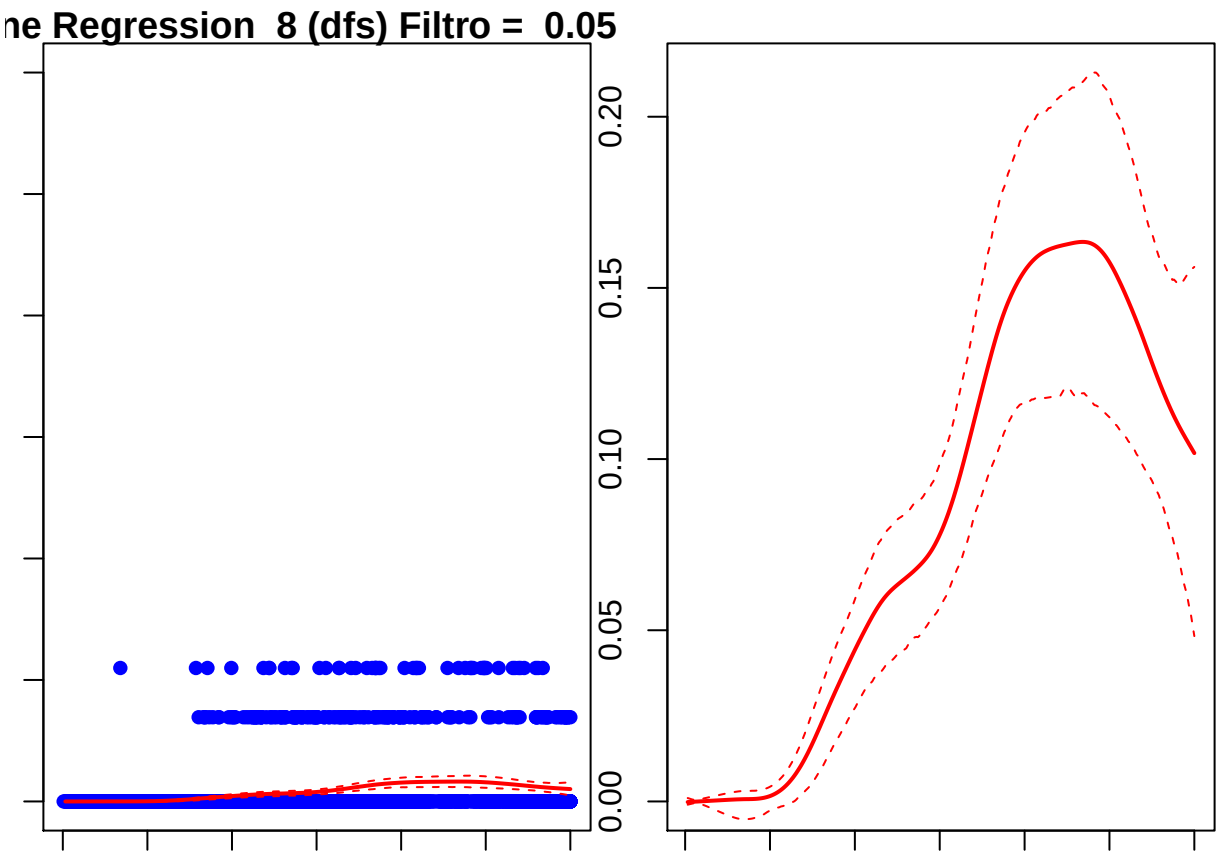

ne Regression 8 (dfs) Filtro = 0.01
0.0
0.1
0.2
0.3
0.4

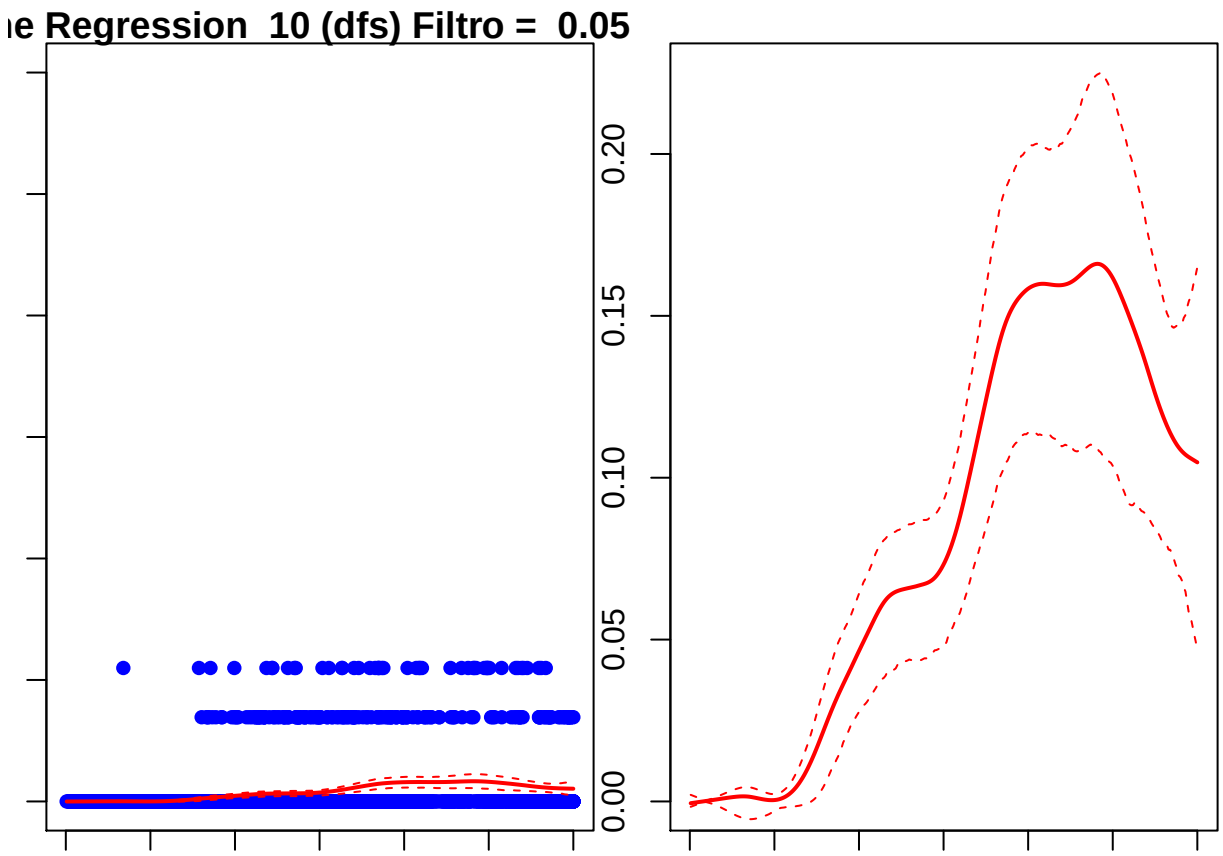

ıe Regression 10 (dfs) Filtro = 0.01
0.0
0.1
0.2
0.3
0.4


ne Regression 2 (dfs) Filtro = 0.05
0.00
0.05
0.10
0.15
0.20


ne Regression 4 (dfs) Filtro = 0.05
0.00
0.05
0.10
0.15

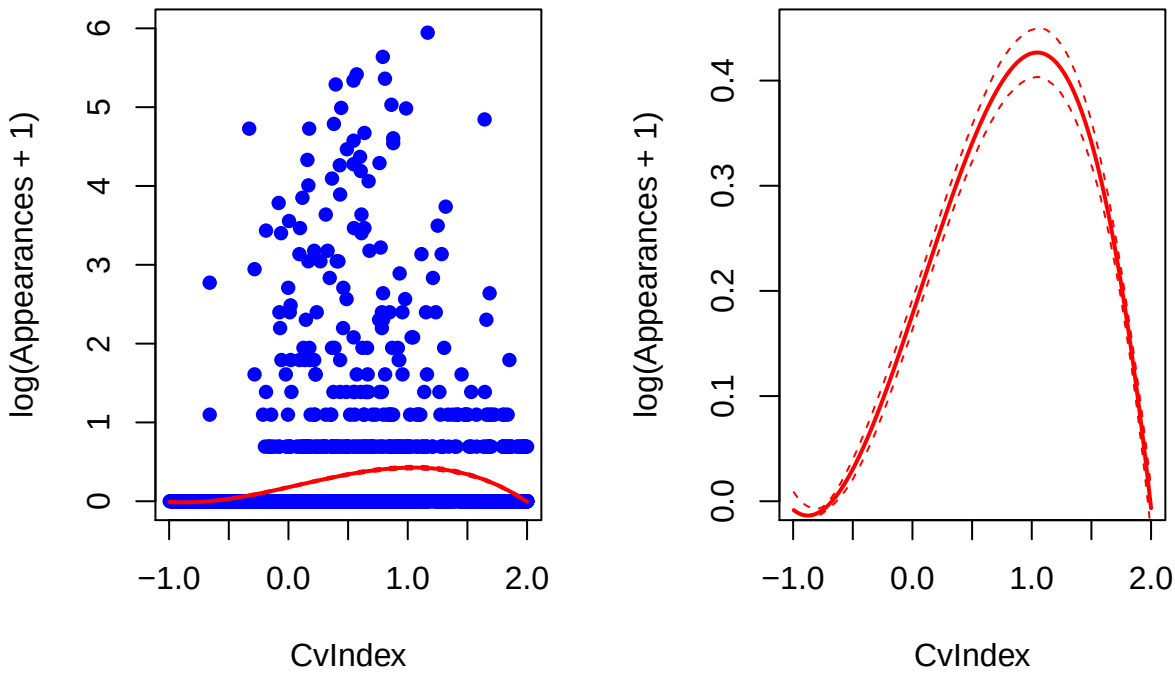

ne Regression 6 (dfs) Filtro = 0.05
0.00
0.05
0.10
0.15
0.20

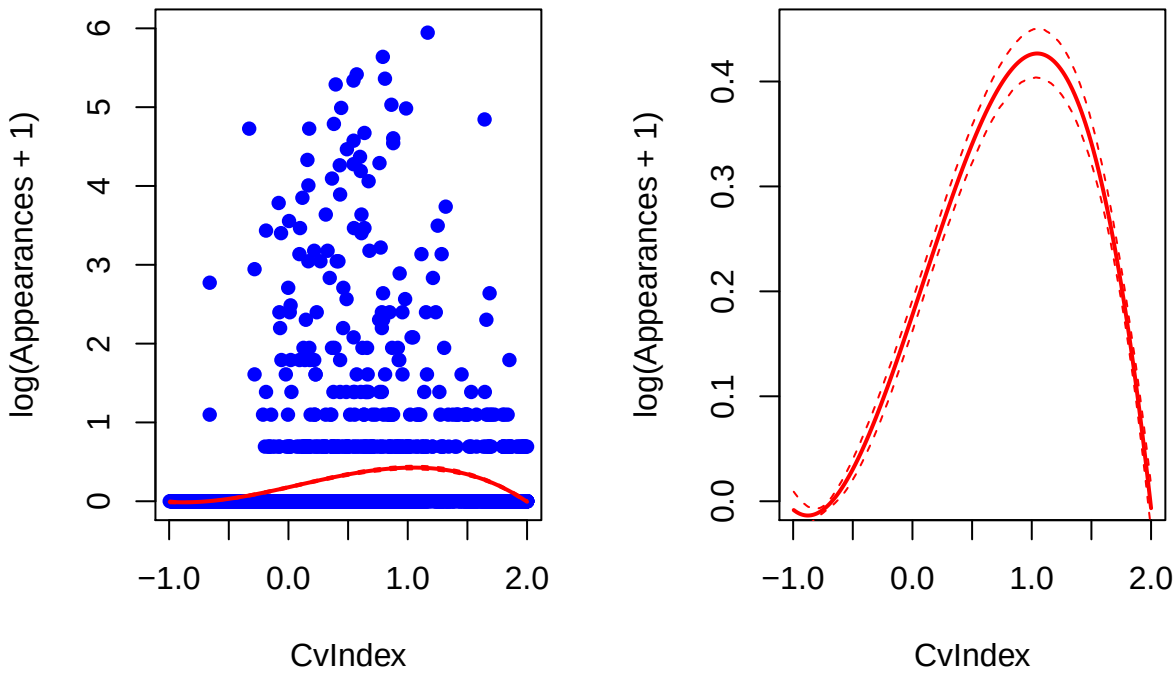

ne Regression 8 (dfs) Filtro = 0.05
0.00
0.05
0.10
0.15
0.20

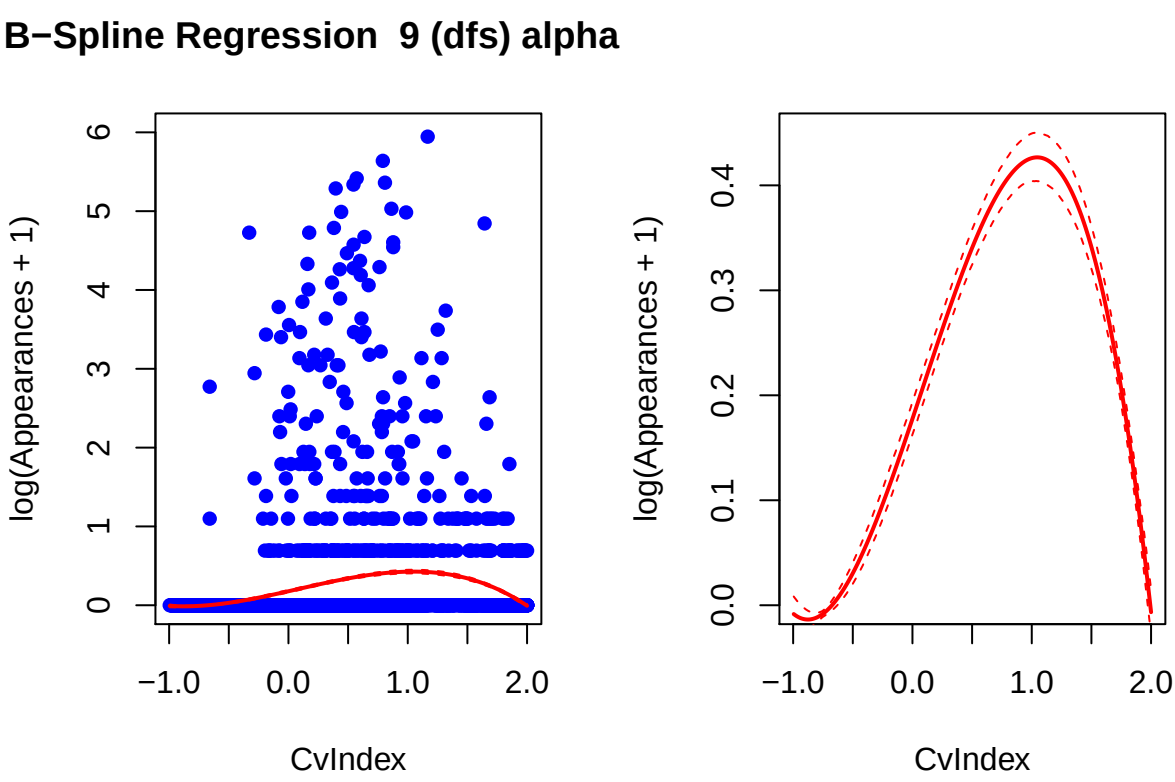


- B-Splines - Se puede ver claramente que el ajuste es altamente similar al obtenido con el smoothing spline, a mayores incrementar los df's o filtrar outliers no implica grandes cambios.

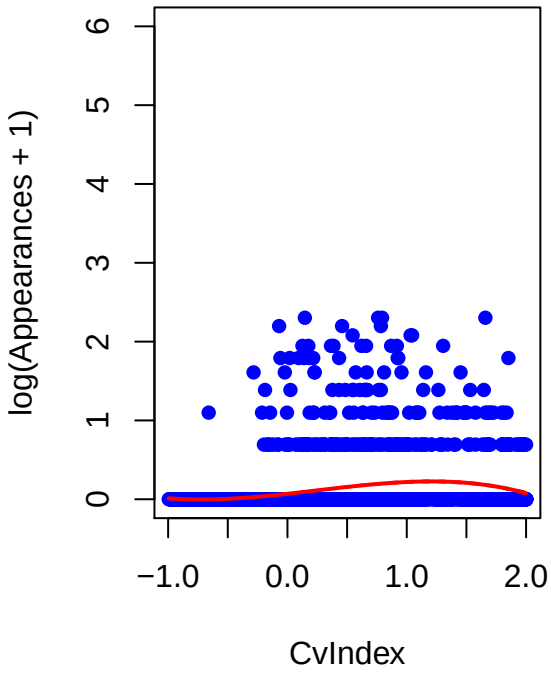
```
l_a <- c(0.0, 0.025,0.05)
for (a in l_a){
  # Eliminamos datos fuera de lugar
  lower_bound <- quantile(y, 0.00);lower_bound
  upper_bound <- quantile(y, 1 - a);upper_bound

  outlier_ind <- which(y < lower_bound | y > upper_bound)
  if (!is.na(outlier_ind[1])){
    data_tmp <- data_filtered[-outlier_ind,]
  }else{
    data_tmp <- data_filtered}
  dfs <- seq(3, 10, by = 3)
  #windows()
  par(mfrow=c(1,2))
  spls <- numeric(3)
  i <- 1
  for (df in dfs)
  {
    spls[i] <- splines_adjust(data_tmp, 'Log.GB.Seqs', 'CVTOT', df,6,0.5,1000,paste("B-Spline Regressio
    i <- i + 1
  }
}
```

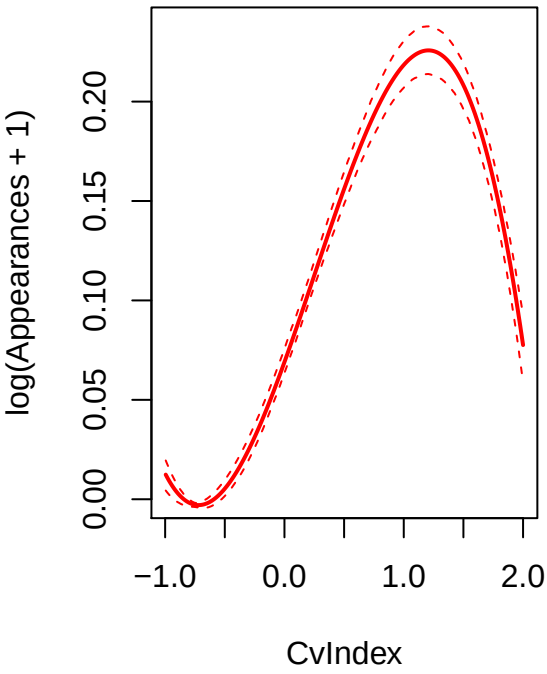

B−Spline Regression 3 (dfs) alpha
log(Appearances + 1)
0
1
2
3
4
5
6
−1.0
0.0
1.0
2.0
CvIndex
log(Appearances + 1)
0.0
0.1
0.2
0.3
0.4
−1.0
0.0
1.0
2.0
CvIndex


B−Spline Regression 6 (dfs) alpha
log(Appearances + 1)
0
1
2
3
4
5
6
−1.0
0.0
1.0
2.0
CvIndex
log(Appearances + 1)
0.0
0.1
0.2
0.3
0.4
−1.0
0.0
1.0
2.0
CvIndex

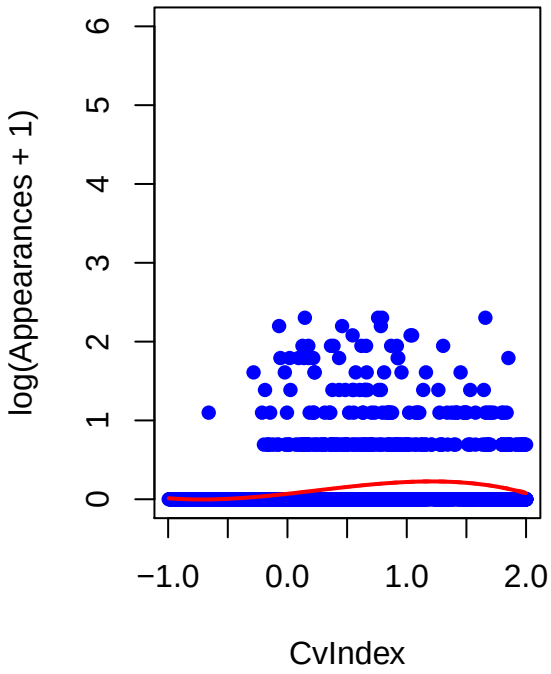

B−Spline Regression 9 (dfs) alpha
log(Appearances + 1)
0
1
2
3
4
5
6
−1.0
0.0
1.0
2.0
CvIndex
log(Appearances + 1)
0.0
0.1
0.2
0.3
0.4
−1.0
0.0
1.0
2.0
CvIndex

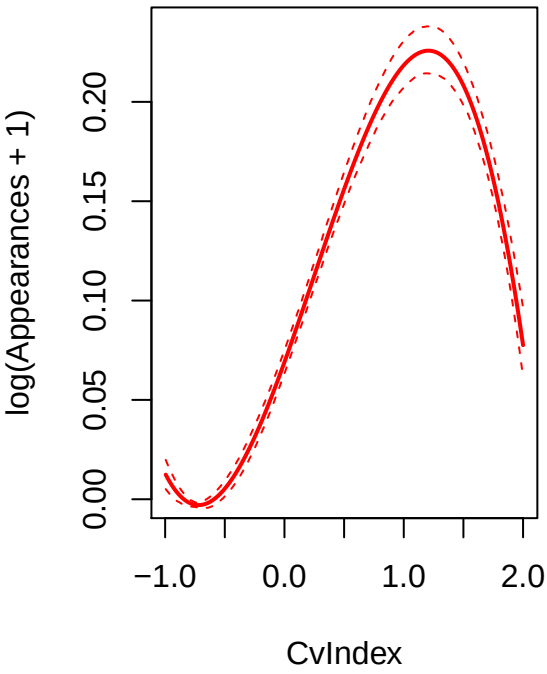
−Spline Regression 3 (dfs) alpha =



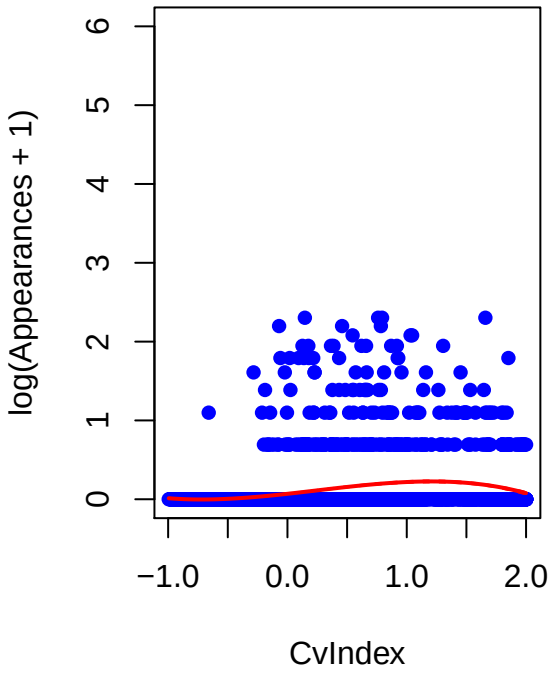


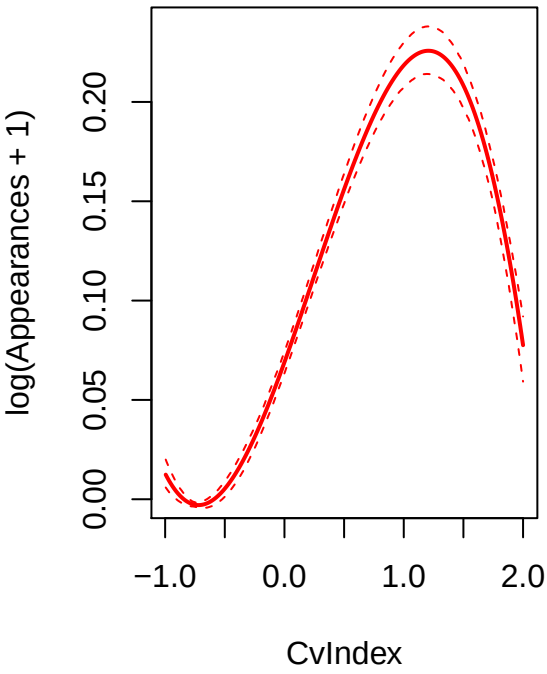
−Spline Regression 6 (dfs) alpha =

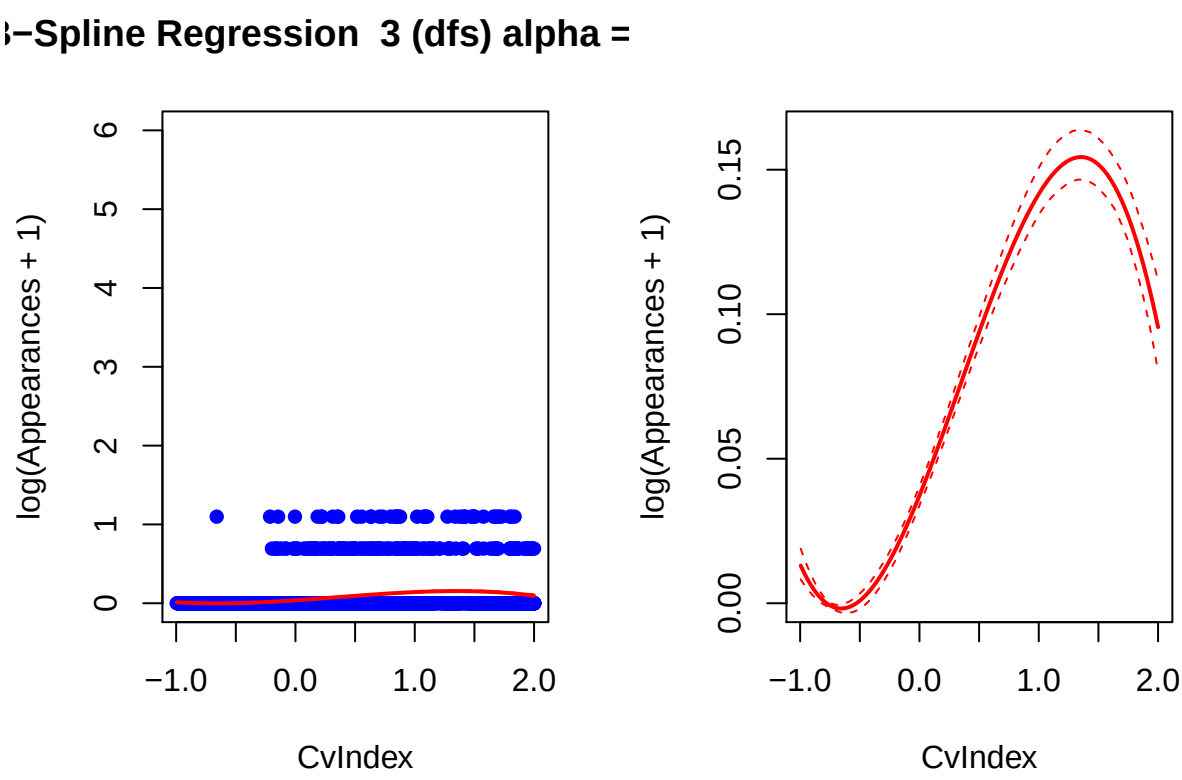




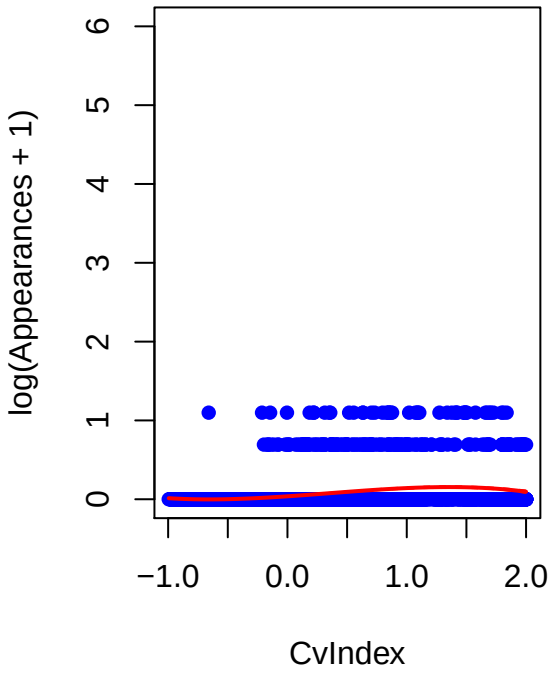
−Spline Regression 9 (dfs) alpha =

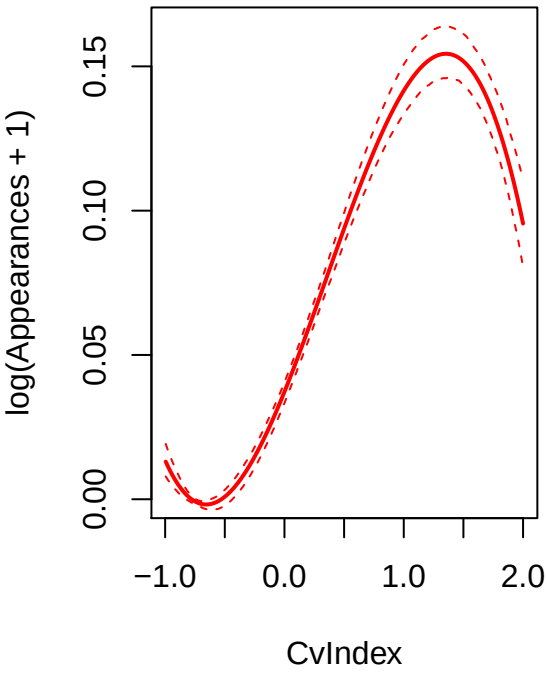


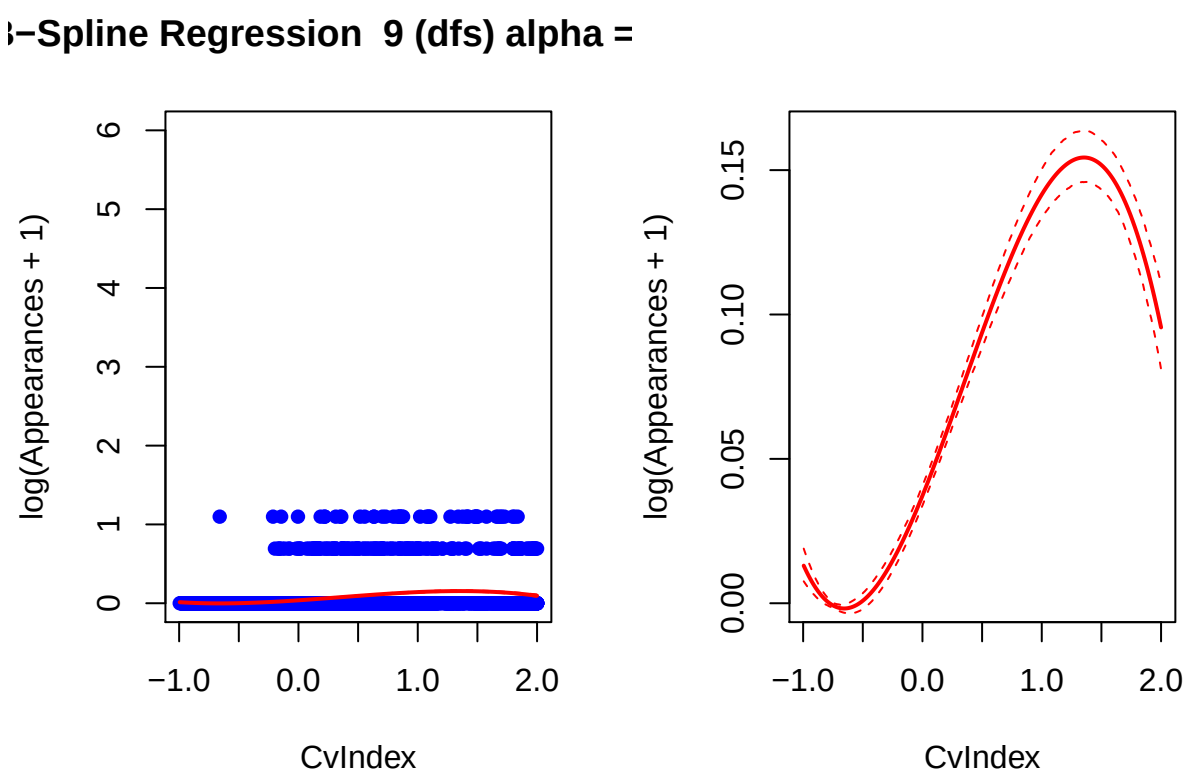



B−Spline Regression 3 (dfs) alpha =
log(Appearances + 1)
0
1
2
3
4
5
6
−1.0
0.0
1.0
2.0
CvIndex
log(Appearances + 1)
0.00
0.05
0.10
0.15
−1.0
0.0
1.0
2.0
CvIndex

s−Spline Regression 6 (dfs) alpha =





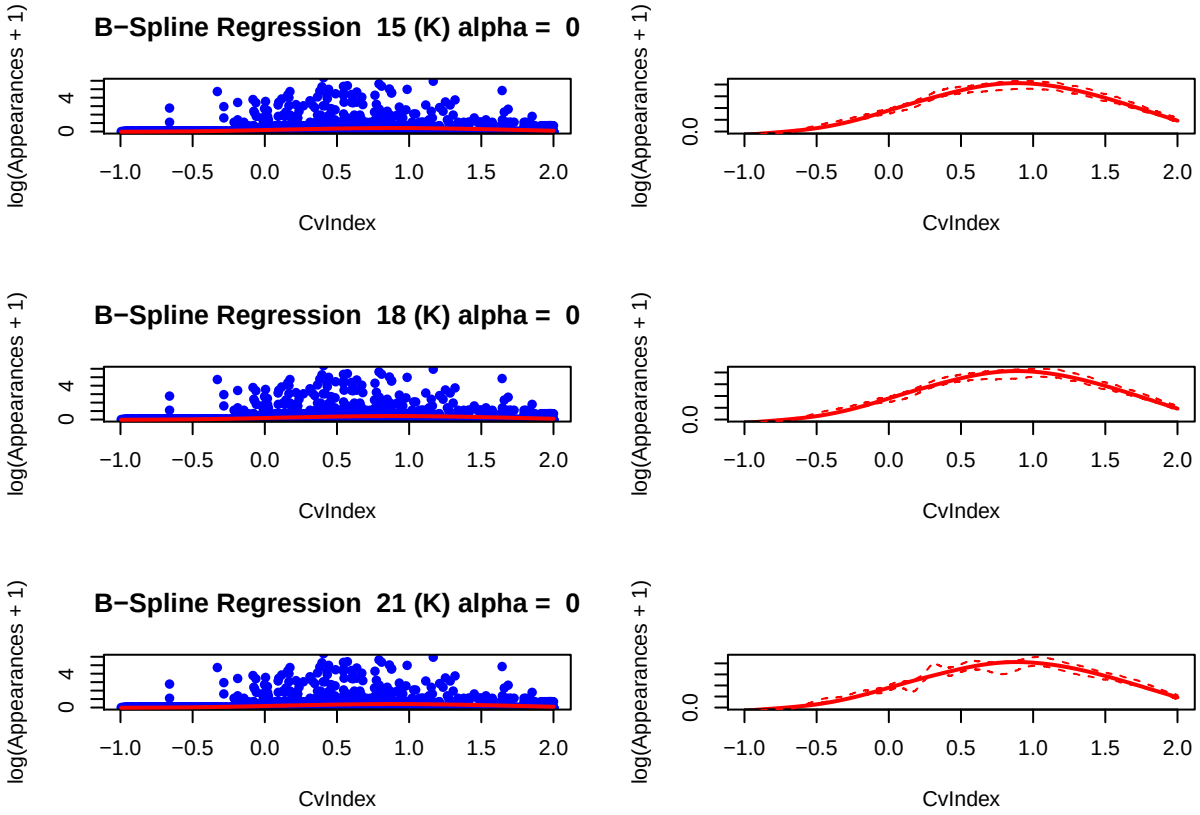


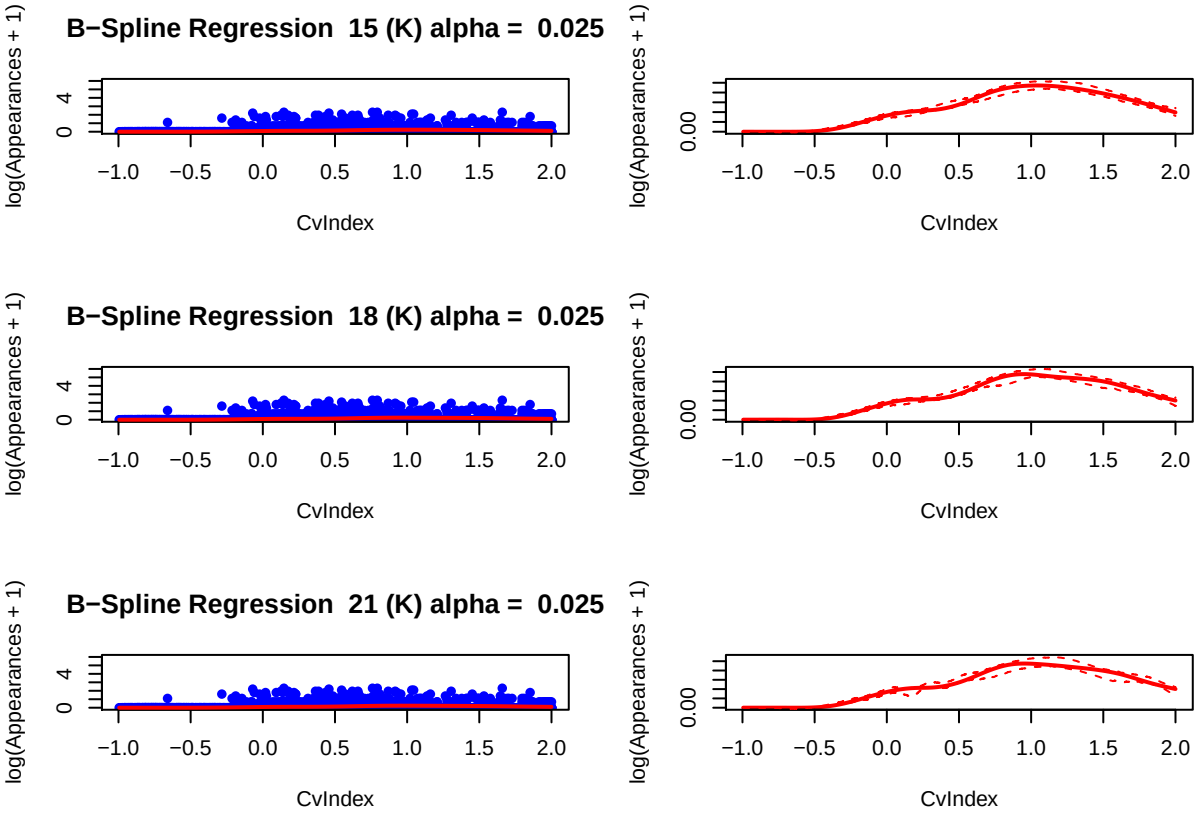
- P-Splines - los resultados de la evaluación del modelo vía gam.check demuestran que no es viable el P-spline pues no se cumplen las hipótesis de partida ni con imaginación, además en la mayor parte de casos el valor de k se encuentra extrañamente infraestimado (lo que no creo que sea una buena noticia)

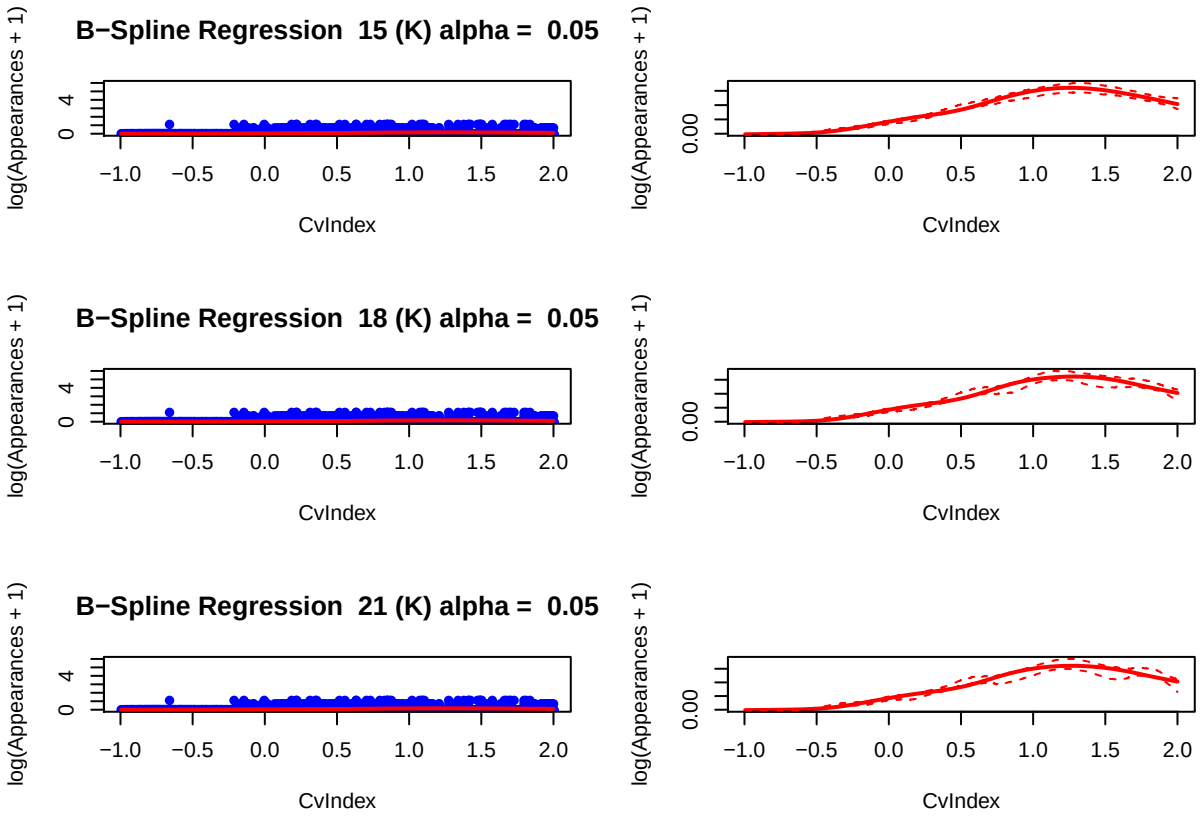
Por probar.

```
l_a <- c(0.0, 0.025,0.05)
for (a in l_a){
  # Eliminamos datos fuera de lugar
  lower_bound <- quantile(y, 0.00);lower_bound
  upper_bound <- quantile(y, 1 - a);upper_bound

  outlier_ind <- which(y < lower_bound | y > upper_bound)
  if (!is.na(outlier_ind[1])){
    data_tmp <- data_filtered[-outlier_ind,]
  }else{
    data_tmp <- data_filtered}
  dfs <- seq(15, 21, by = 3)
  #windows()
  par(mfrow=c(3,2))
  spls <- numeric(3)
  i <- 1
  for (df in dfs)
  {
    spls[i] <- splines_adjust(data_tmp, 'Log.GB.Seqs', 'CVTOT', df,6,0.5,1000,paste("B-Spline Regressio
    i <- i + 1
  }
```

```
}
```


B−Spline Regression 15 (K) alpha = 0
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
B−Spline Regression 18 (K) alpha = 0
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
B−Spline Regression 21 (K) alpha = 0
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex


B−Spline Regression 15 (K) alpha = 0.025
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.00
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
B−Spline Regression 18 (K) alpha = 0.025
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.00
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
B−Spline Regression 21 (K) alpha = 0.025
log(Appearances + 1)
0 4
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex
log(Appearances + 1)
0.00
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
CvIndex



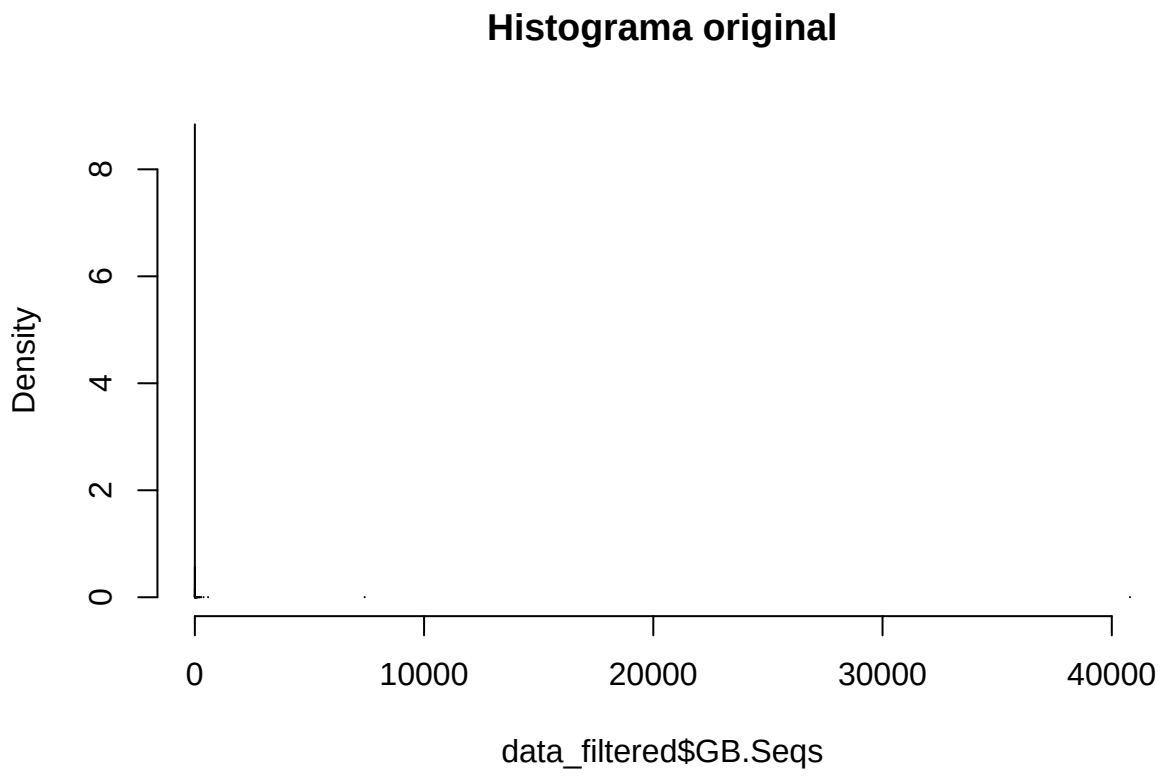
Regresión logística: paramétrica y no paramétrica.

A la vista de lo comentado por Antón en la reunión donde se dijo que el haplotipo determinaba el número de veces que un genoma era secuenciado. Es obvio por tanto que el número de apariciones en GenBank estará altamente correlado con mayor número de secuenciaciones. Por tanto, es probable que la magnitud de las ocurrencias en GenBank pueda no ser relevante y únicamente lo sea si aparece o no (discretizamos la variable GB.Seqs). * Opción A: poner a 1 si Apariciones > 0 * Opción B: poner a 1 si Apariciones $>$ filtro, con filtro > 1, a la vista de los resultados del histograma no parece sensato dado que eliminaríamos cerca del 95% de la información.

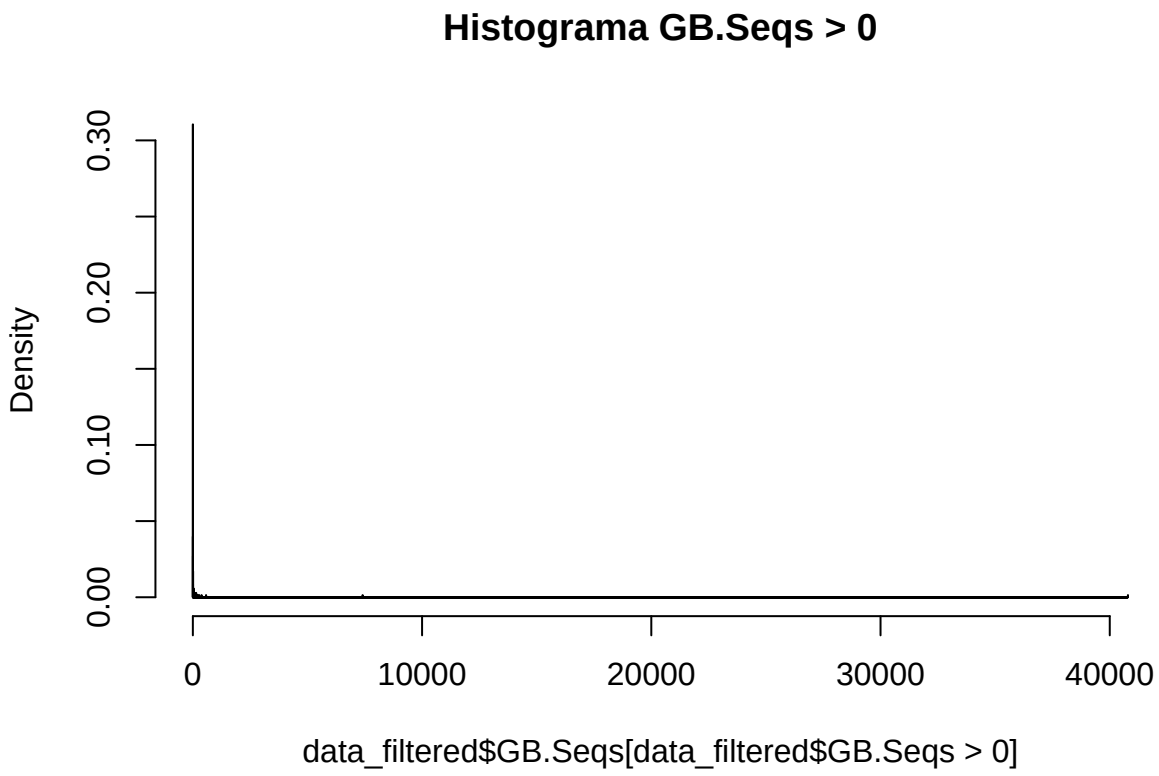
```
par(c(1,3))
```

```
## NULL
```

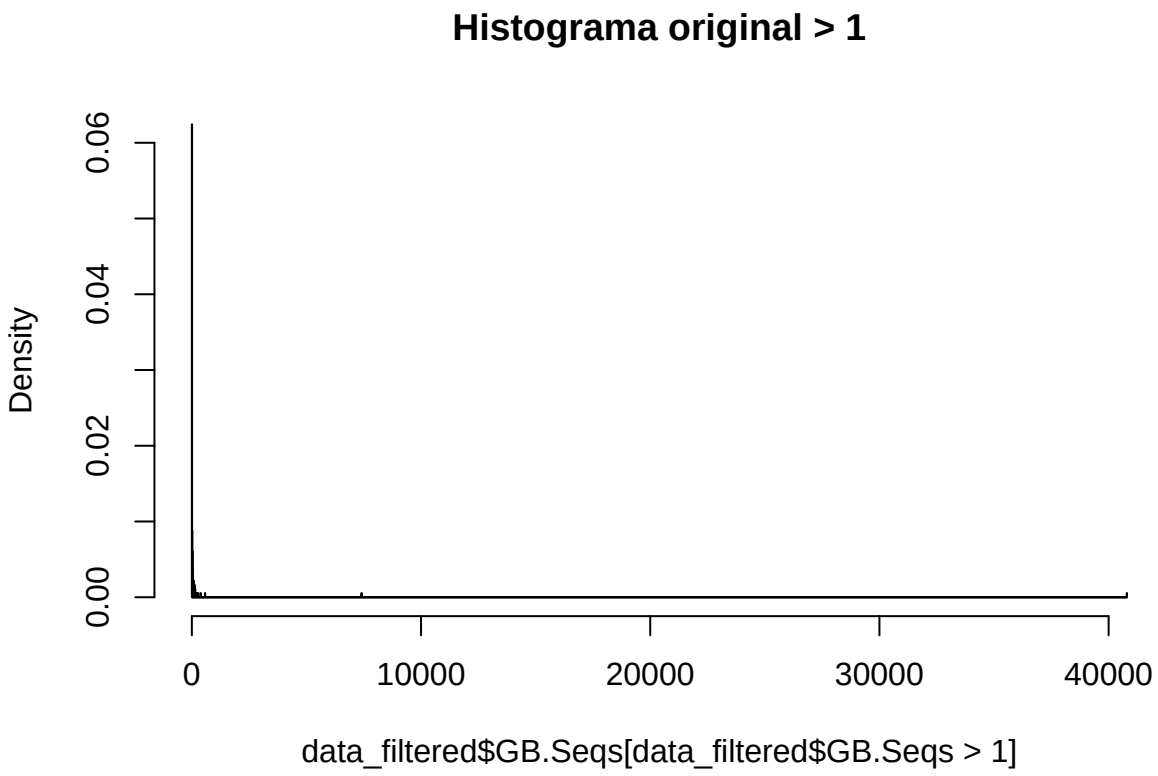
```
hist(data_filtered$GB.Seqs, breaks = 'fd', freq = F, main = 'Histograma original')
```




```
hist(data_filtered$GB.Seqs[data_filtered$GB.Seqs > 0], breaks = 'fd', freq = F, main = 'Histograma GB.S
```




```
hist(data_filtered$GB.Seqs[data_filtered$GB.Seqs > 1], breaks = 'fd', freq = F, main = 'Histograma orig
```

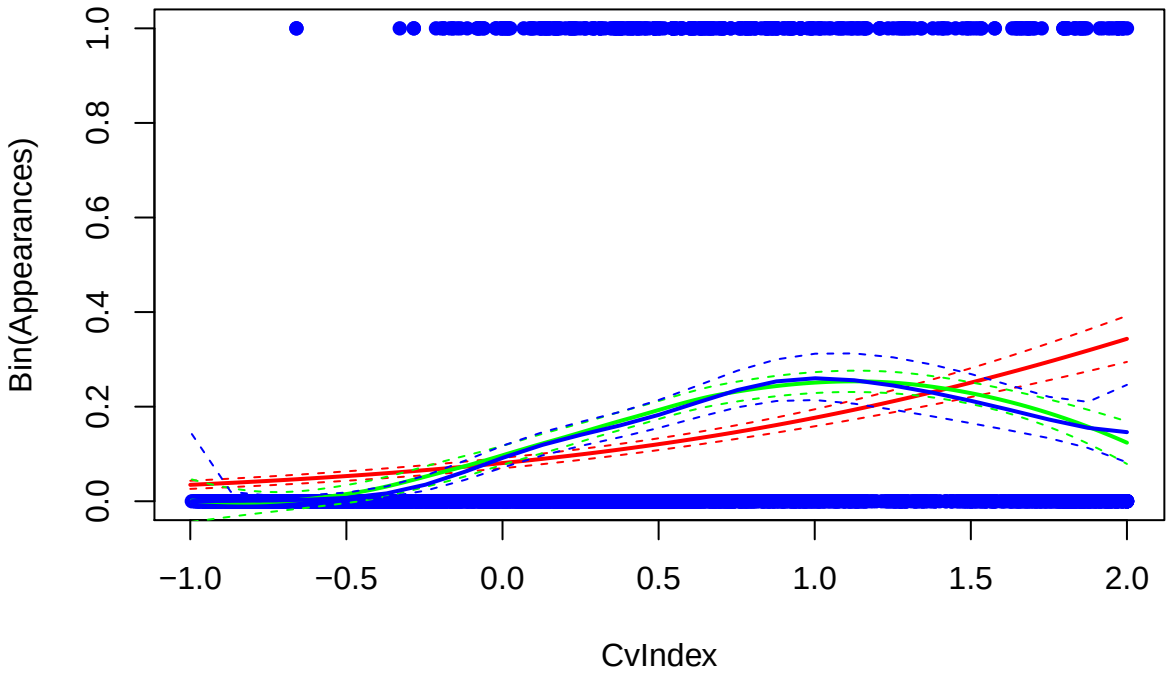



```r
data_filtered$Bin.GB.Seqs[data_filtered$GB.Seqs > 0] <- 1
# Regresiones logísticas
plot(data_filtered$CVTOT,data_filtered$Bin.GB.Seqs,pch=16,col="blue",ylim=c(0,1),xlab="CvIndex",ylab="Bi
# Regresión logística clásica
mod_logit <- glm(Bin.GB.Seqs~CVTOT, family = binomial(link = 'logit'), data = data_filtered)
summary(mod_logit)
```

```
##
## Call:
## glm(formula = Bin.GB.Seqs ~ CVTOT, family = binomial(link = "logit"),
##     data = data_filtered)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9173  -0.5103  -0.3874  -0.3015   2.4774
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.43228    0.07504  -32.41   <2e-16 ***
## CVTOT        0.89211    0.07014   12.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2172.4  on 3027  degrees of freedom
```

```
## Residual deviance: 2002.8  on 3026  degrees of freedom
## AIC: 2006.8
##
## Number of Fisher Scoring iterations: 5
```

```r
# Predictions
x_pred <- data.frame(CVTOT = seq(-1,2, length = 1000))
pred <- predict(mod_logit, x_pred,se.fit = T, type = 'response')
lines(x_pred$CVTOT, pred$fit, col = 'red', lwd = 2)
lines(x_pred$CVTOT, pred$fit + pred$se.fit*1.96, col = 'red', lty = 2)
lines(x_pred$CVTOT, pred$fit - pred$se.fit*1.96, col = 'red', lty = 2)

# Regresión loess (olvidamos el carácter dicotómico de la variable)
idx <- sort(data_filtered$CVTOT,index.return=TRUE)$ix
x_sort <- data_filtered$CVTOT[idx]
y_sort=data_filtered$Bin.GB.Seqs[idx]
lo <- loess(y_sort~x_sort)
pred_lo <- predict(lo, se = TRUE)
lines(x_sort,pred_lo$fit,col="green",lwd=2)
lines(x_sort,pred_lo$fit+pred_lo$se.fit*1.96,col="green",lwd=1, lty = 2)
lines(x_sort,pred_lo$fit-pred_lo$se.fit*1.96,col="green",lwd=1, lty = 2)

# Regresión logística local
h <- h.select(data_filtered$CVTOT, data_filtered$Bin.GB.Seqs)
log_loc <- sm.binomial(data_filtered$CVTOT,data_filtered$Bin.GB.Seqs,h=h,add=T,col="red",lwd=2,pch=28,y
lines(log_loc$eval.points,log_loc$estimate,type="l",col="blue",lwd=2)
lines(log_loc$eval.points,log_loc$upper,col="blue",lty=2)
lines(log_loc$eval.points,log_loc$lower,col="blue",lty=2)
```



A la vista de los resultados parece lógico decir que la regresión logística clásica obvia la caída final de probabilidad, pero por otro lado parece correlar en gran medida con los visto para los ajustes no paramétricos tipo, spline.