Exercice Régression

Rachid Sahli

23 janvier 2025

Exercice 1

On génère un jeu de données Y qui est composé d'une tendance linéaire et périodique avec un bruit aléatoire.

Rappel :

Régressogramme : C'est une méthode non paramétrique utilisée pour estimer la fonction de régression d'une variable dépendante en fonction d'une variable indépendante, sans faire d'hypothèses spécifiques sur la forme de la relation entre les variables.

```
#Données simulées

n<-2000 # 2000 points
t<-1:n # vecteur allant de 1 à n
e<-0.2*rnorm(n) # bruit aléatoire
f<-2+3*t/n+ sin(2*pi*t/n) # fonction de régression
Y<-e+f # réponse observée
plot(t/n, Y, pch=".", xlab="x", main="Nuage de point et Régressogramme")

newt<-cut(t/n,20) # creation de bins (20)
levels(newt) #les modalites de cette nouvelle variable
```

```
##  [1] "(-0.000499,0.0505]" "(0.0505,0.1]"       "(0.1,0.15]"
##  [4] "(0.15,0.2]"         "(0.2,0.25]"         "(0.25,0.3]"
##  [7] "(0.3,0.35]"         "(0.35,0.4]"         "(0.4,0.45]"
## [10] "(0.45,0.5]"         "(0.5,0.55]"         "(0.55,0.6]"
## [13] "(0.6,0.65]"         "(0.65,0.7]"         "(0.7,0.75]"
## [16] "(0.75,0.8]"         "(0.8,0.85]"         "(0.85,0.9]"
## [19] "(0.9,0.95]"         "(0.95,1]"
```

```
table(newt)
```

```
## newt
## (-0.000499,0.0505]       (0.0505,0.1]         (0.1,0.15]         (0.15,0.2]
##                100                100                100                100
##         (0.2,0.25]         (0.25,0.3]         (0.3,0.35]         (0.35,0.4]
##                100                100                100                100
##         (0.4,0.45]         (0.45,0.5]         (0.5,0.55]         (0.55,0.6]
##                100                100                100                100
##         (0.6,0.65]         (0.65,0.7]         (0.7,0.75]         (0.75,0.8]
##                100                100                100                100
##         (0.8,0.85]         (0.85,0.9]         (0.9,0.95]           (0.95,1]
##                100                100                100                100
```

```r
meany<-tapply(Y,newt,mean) # calcul des moyennes pour chaque bin
xseq1<-seq(0,1,by=1/20) # vecteur de 20 valeurs entre 0 et 1
length(xseq1)
```

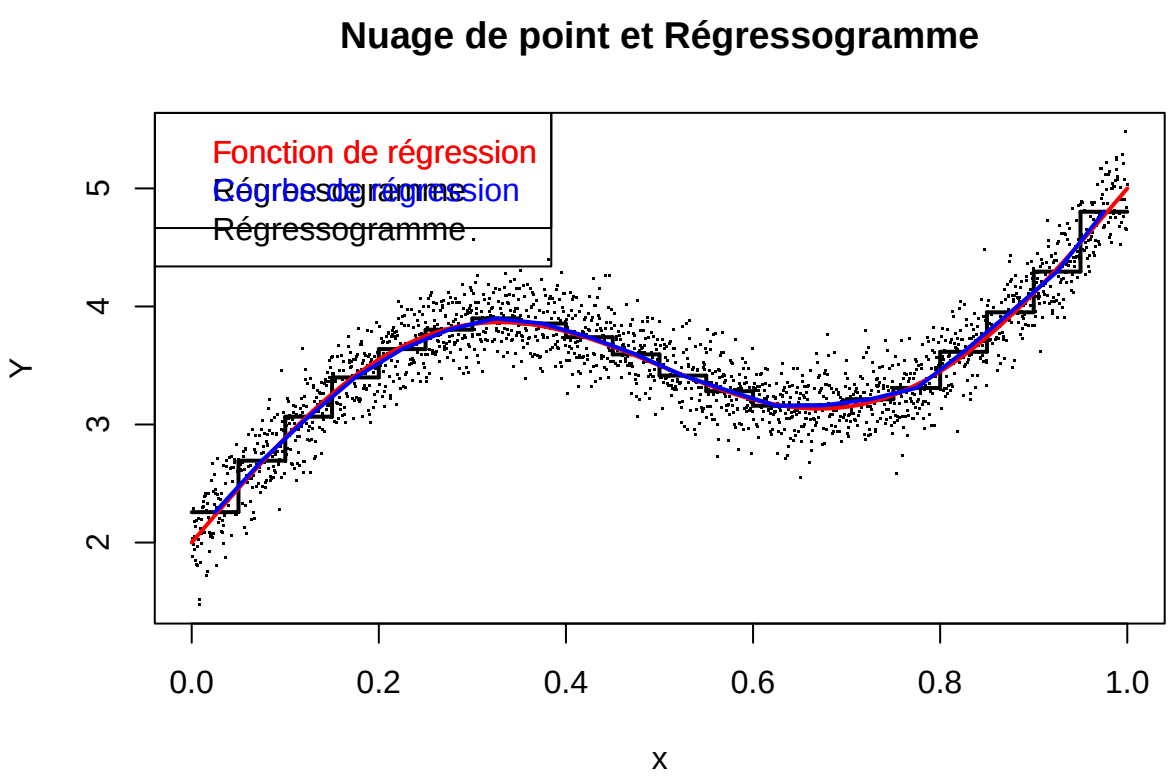
```
## [1] 21
```

```r
length(meany)
```

```
## [1] 20
```

```r
for(i in 1:20){lines(xseq1[i:(i+1)], c(meany[i], meany[i]), lwd=2)} # lignes horizontales regress
for(i in 1:19){lines(c(xseq1[i+1],xseq1[i+1]),meany[i:(i+1)], lwd=2)} # lignes verticales regress
grille<-seq(0,1,by=0.01)
lines(grille, 2+3*grille+ sin(2*pi*grille), col="red", lwd=2)
legend("topleft", legend=c("Fonction de régression", "Régressogramme"),
       text.col=c("red", "black"))

xseq11<-head(xseq1,20)
xseq12<-tail(xseq1,20)
xseq2<-(xseq11+xseq12)/2
lines(xseq2, meany, col="blue", lwd=2)
legend("topleft", legend=c("Fonction de régression", "Courbe de régression","Régressogramme"),
       text.col=c("red", "blue","black"))
```



Optimisation de l'estimation d'une fonction f en testant différentes granularités de partitionnement des données. En ajusant le nombre de partitions m et en calculant l'erreur quadratique moyenne (EQM) pour chaque partition, l'objectif est de trouver le nombre de partitions qui minimise l'erreur.

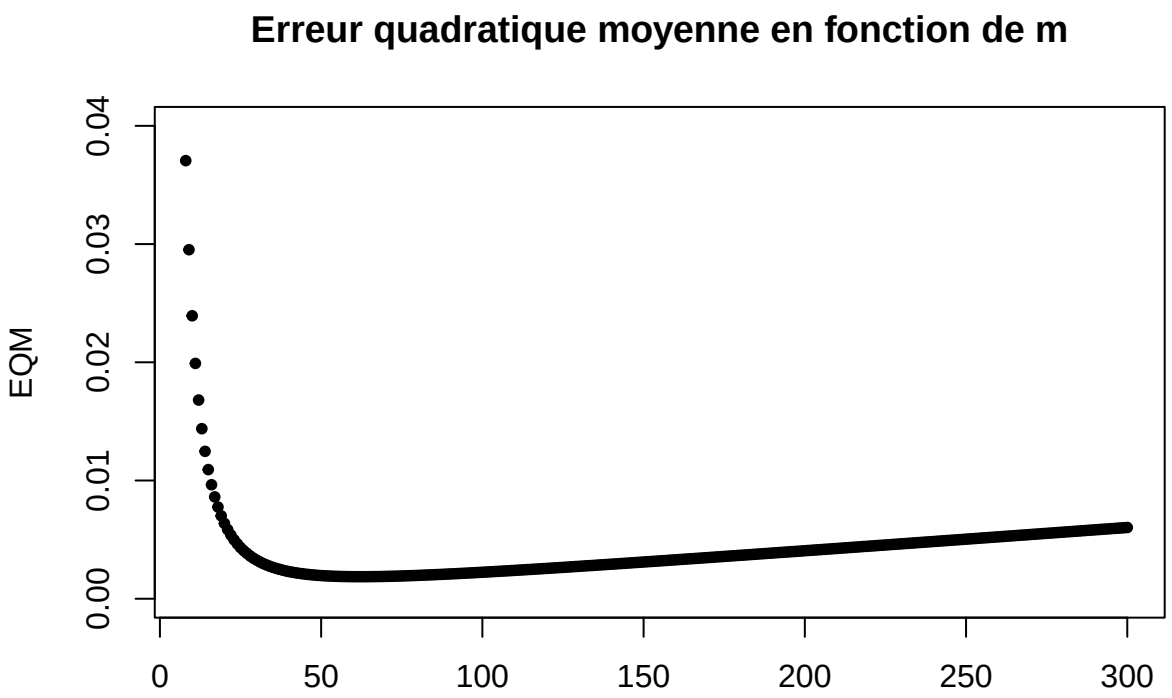
```r
#Visualisation de l'écart à f (inconnu si f est inconnue)

N<-300 # 300 itérations
```

```r
EQM<-vector(mode = "numeric", length = N) # Vecteur vide pour EQM
EQM[1]<-mean((f-mean(f))^2)+(0.04)/n # Calcul EQM
for(i in 2:N) # Calcul pour les i partitions
{
  newt<-cut(t/n,i)
  F<-as.factor(newt)
  reg<-lm(f~F)
  EQM[i]<-mean(reg$residuals^2)+(0.04*i)/n
}
which.min(EQM) # partition qui minimise l'EQM
```

```
## [1] 62
```

```r
plot(EQM, ylim=c(0,0.04), xlim=c(10,300),pch=20, main="Erreur quadratique moyenne en fonction de m", xla
```




```r
#Sélection automatique de partition avec capushe

library(capushe) # Import library
```

```
## Warning: package 'capushe' was built under R version 4.2.3
```

```
## Loading required package: MASS
```

```r
N<-100
n<-2000
t<-1:n
e<-0.2*rnorm(n)
f<-2+3*t/n+ sin(2*pi*t/n)
Y<-e+f
C1<-2:N
C1<-as.vector(C1)
C4<-vector(mode = "numeric", length = N-1)
for(i in 1:(N-1))
{
```

```
  newt<-cut(t/n,i+1)
  F<-as.factor(newt)
  reg<-lm(Y~F)
  C4[i]<-sum(reg$residuals^2)
}
M<-cbind(C1,C1,C1,C4)
summary(capushe(M))
```

```
## Warning in DDSE(data, pct = pct, point = point, psi.rlm = psi.rlm, scoef =
## scoef): Some elements in Kappa are negative
```

```
##
## Model selected by DDSE: 57
##
## Corresponding Slope interval: [0.0624;0.0823](32.3% of points)
## Selected model complexity: 57
##
##
## Model selected by Djump: 57
##
## Estimated penalty constant: 0.0676
## Selected model complexity: 57
```

```
capushe(M)@Djump
```

```
## Warning in DDSE(data, pct = pct, point = point, psi.rlm = psi.rlm, scoef =
## scoef): Some elements in Kappa are negative
```

```
## [1] "57"
```

```
S<-(capushe(M)@Djump)@model
```

```
## Warning in DDSE(data, pct = pct, point = point, psi.rlm = psi.rlm, scoef =
## scoef): Some elements in Kappa are negative
```

```
S<-as.numeric(S)
newt<-cut(t/n,S)
meany<-tapply(Y,newt,mean)
xseq1<-seq(0,1,by=1/S)
length(xseq1)
```

```
## [1] 58
```

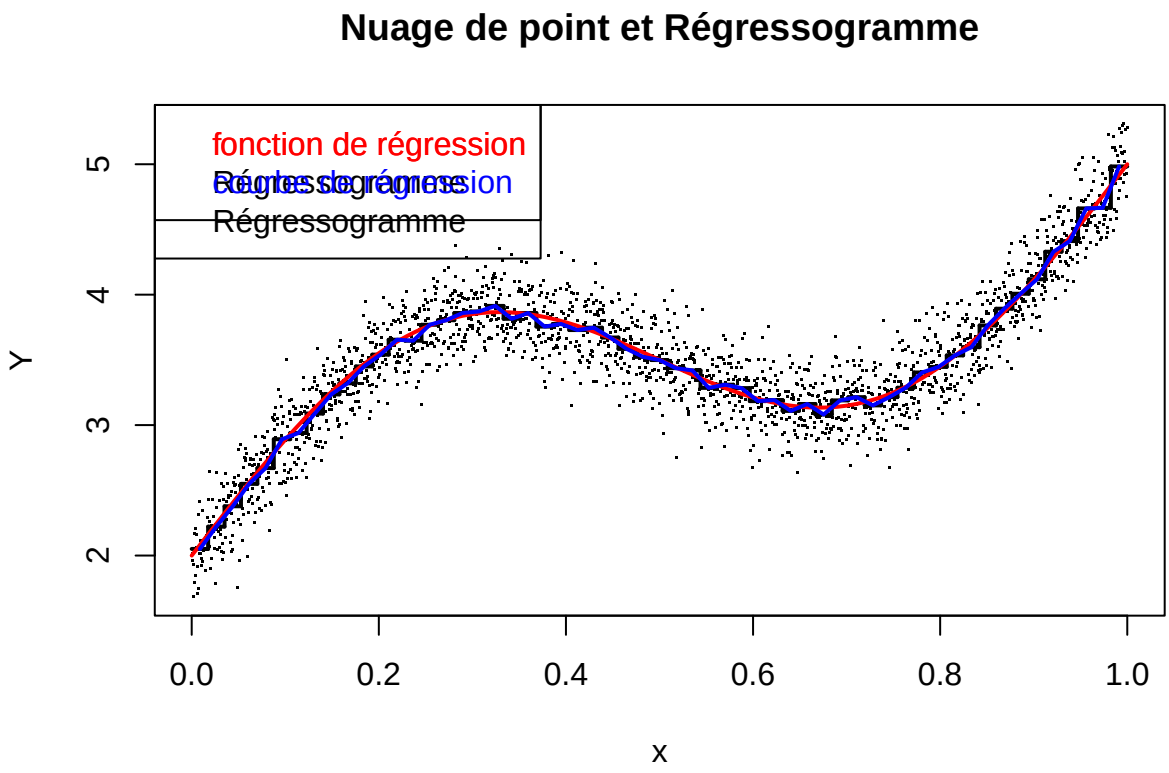
```
length(meany)
```

```
## [1] 57
```

```
## Représentation graphique du Regressogramme sélectionné avec Capushe

plot(t/n, Y, pch=".", xlab="x", main="Nuage de point et Régressogramme")
for(i in 1:S){lines(xseq1[i:(i+1)], c(meany[i], meany[i]), lwd=2)}
for(i in 1:(S-1)){lines(c(xseq1[i+1],xseq1[i+1]),meany[i:(i+1)], lwd=2)}
grille<-seq(0,1,by=0.01)
lines(grille, 2+3*grille+ sin(2*pi*grille), col="red", lwd=2)
legend("topleft", legend=c("fonction de régression", "Régressogramme"),
       text.col=c("red", "black"))
## Ajout de la courbe de régression
xseq11<-head(xseq1,S)
```

```r
xseq12<-tail(xseq1,S)
xseq2<-(xseq11+xseq12)/2
lines(xseq2, meany, col="blue", lwd=2)
legend("topleft", legend=c("fonction de régression", "courbe de régression","Régressogramme"),
       text.col=c("red", "blue","black"))
```



Exercie 2

On travaille sur le jeu de données Eucalyptus. Nous allons nous concentrer sur les variables hauteur (ht) et circonférence (circ).

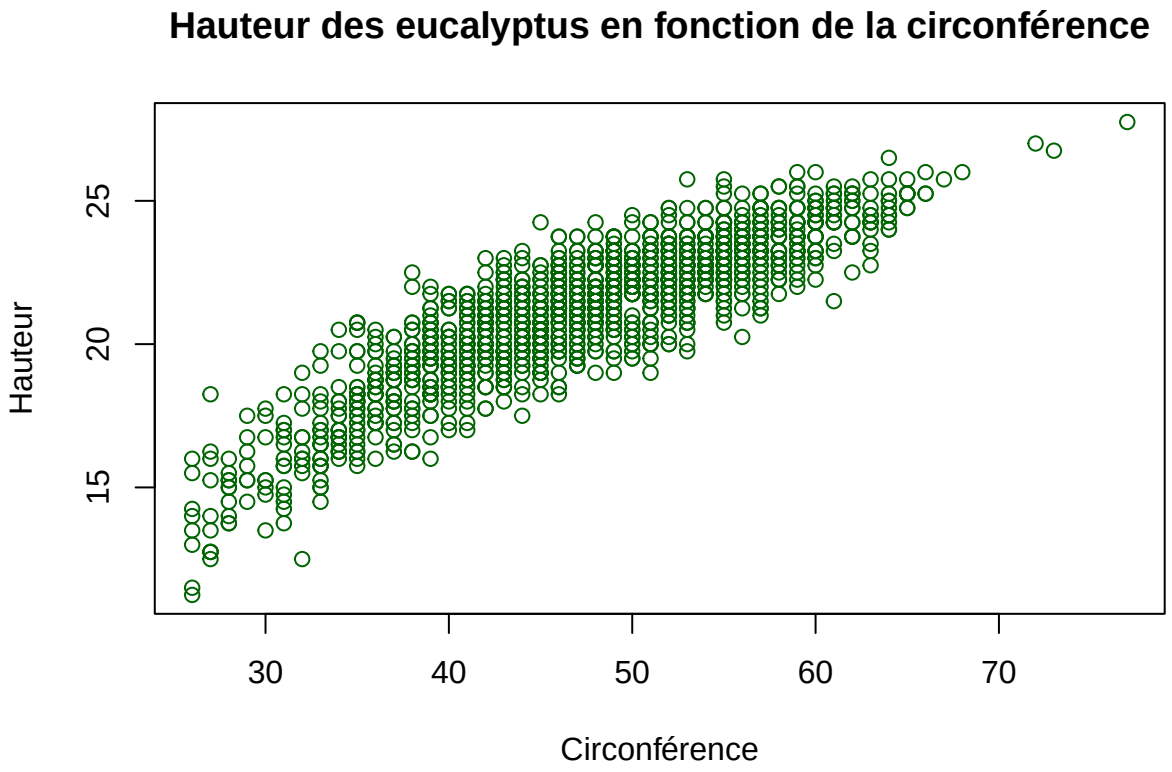
```r
head(euca) # Aperçu des données
```

```
##      ht circ bloc  clone
## 1 18.25   36    1 L2-123
## 2 19.75   42    1 L2-123
## 3 16.50   33    1 L2-123
## 4 18.25   39    1 L2-123
## 5 19.50   43    1 L2-123
## 6 16.25   34    1 L2-123
```

Traçage du nuage de points

Dans un premier temps, nous représentons le nuage de points de la hauteur des eucalyptus en fonction de la circonférence.

```r
plot(euca$circ, euca$ht, col = "darkgreen", ylab = "Hauteur", xlab = "Circonférence",
     main = "Hauteur des eucalyptus en fonction de la circonférence")
```



Calcul du régressogramme

On calcule ensuite le régressogramme avec une partition de taille 16.

```
newx<-cut(euca$circ,16) # découpe en 16 bins
table(newx) # répartition par partition
```

```
## newx
## (25.9,29.2] (29.2,32.4] (32.4,35.6] (35.6,38.8] (38.8,41.9] (41.9,45.1]
##          35          33          72          83         126         223
## (45.1,48.3] (48.3,51.5] (51.5,54.7] (54.7,57.9] (57.9,61.1] (61.1,64.2]
##         190         182         176         149         107          37
## (64.2,67.4] (67.4,70.6] (70.6,73.8] (73.8,77.1]
##          12           1           2           1
```

```
meany<-tapply(euca$ht,newx,mean) # calcul des moyennes de chaque partition
xseq1<-seq(min(euca$circ),max(euca$circ),
           by =(max(euca$circ)-min(euca$circ))/16) # born des bins dans une séquence
length(xseq1) # taille de la séquence
```

```
## [1] 17
```

```
length(meany) # nombre de moyennes
```

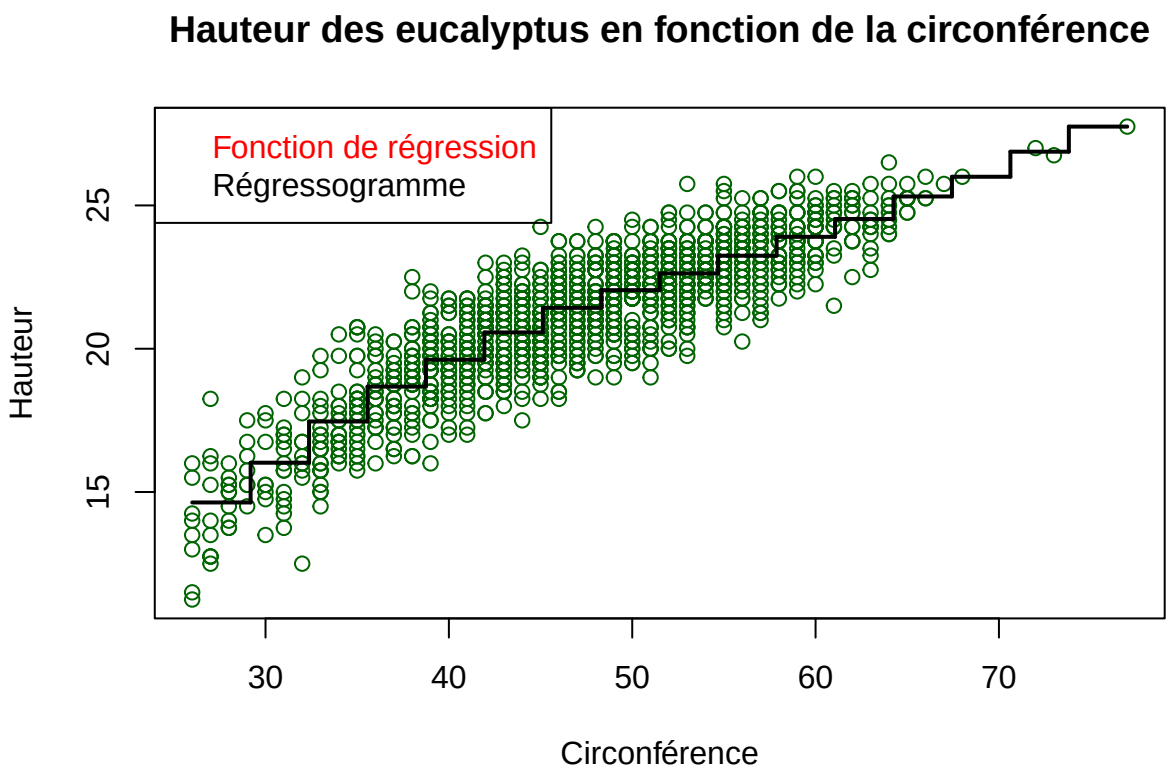
```
## [1] 16
```

```
plot(euca$circ, euca$ht, col = "darkgreen", ylab = "Hauteur", xlab = "Circonférence",
     main = "Hauteur des eucalyptus en fonction de la circonférence") # repésentation graphique

for(i in 1:16){lines(xseq1[i:(i+1)], c(meany[i], meany[i]), lwd=2)} # lignes horizontales
for(i in 1:15){lines(c(xseq1[i+1],xseq1[i+1]),meany[i:(i+1)], lwd=2)} # lignes verticales
xseq11<- head(xseq1,16)
```

```r
xseq12<-tail(xseq1,16)
xseq2<-(xseq11+xseq12)/2 # calcul du milieu de chaque bins

grille<-seq(0,1,by=0.01) # grille pour la fonction de regression
lines(grille, 2+3*grille+ sin(2*pi*grille), col="red", lwd=2)
legend("topleft", legend=c("Fonction de régression", "Régressogramme"),
       text.col=c("red", "black"))
```



On observe une tendance linéaire croissante entre les deux variables.

Comparaison des ajustements

Sur notre précédent graphique, nous ajustons un polynome de degré 3 et nous rajoutons la courbe de régression. L'objectif est de comparer les ajustements. Cela nous permettera de trouver la meilleure courbe polynomiale qui représente les données de ton graphique.

```r
## Comparaison avec ajustement polynomial

plot(euca$ht~euca$circ,xlab="Circonference",
     ylab="Hauteur",
     main="Hauteur en fonction de la circonférence", pch=".")
poly1<-lm(euca$ht~euca$circ)
poly2<-lm(euca$ht~euca$circ+ I(euca$circ^2))
poly3<-lm(euca$ht~euca$circ+I(euca$circ^2)+I(euca$circ^3))
poly4<-lm(euca$ht~euca$circ+I(euca$circ^2)+I(euca$circ^3)+I(euca$circ^4))
AIC(poly1)
```

```
## [1] 4578.454
```

```r
AIC(poly2)
```

```
## [1] 4440.56
```

```
AIC(poly3)
```

```
## [1] 4412.794
```

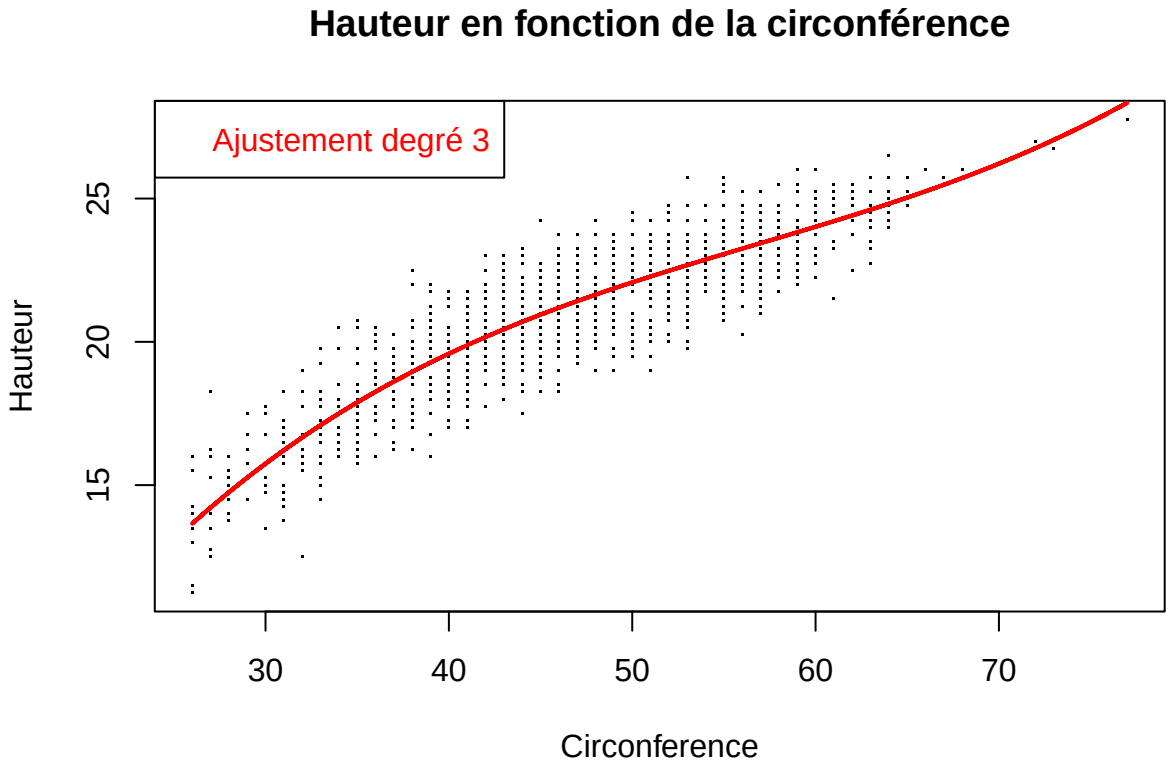
```
AIC(poly4)
```

```
## [1] 4414.737
```

```
poly3$coefficients
```

```
##    (Intercept)      euca$circ I(euca$circ^2) I(euca$circ^3)
##  -1.191514e+01   1.486965e+00  -2.285298e-02   1.341817e-04
```

```
grille<-seq(min(euca$circ),max(euca$circ),by=0.001)
pred3=poly3$coefficients[1]+ poly3$coefficients[2]*grille+
  poly3$coefficients[3]*(grille)^2+poly3$coefficients[4]*(grille)^3
lines(grille,pred3, col="red", lwd=2)
legend("topleft", legend="Ajustement degré 3",  text.col="red")
```




```
#Sélection automatique de partition avec capushe (Eucalyptus)
N<-50
n<-length(euca$circ)
C1<-2:N
C1<-as.vector(C1)
C4<-vector(mode = "numeric", length = N-1)
for(i in 1:(N-1))
{
  newx<-cut(euca$circ,i+1)
  F<-as.factor(newx)
  reg<-lm(euca$ht~F)
```

```
  C4[i]<-sum(reg$residuals^2)
}
M<-cbind(C1,C1,C1,C4)
summary(capushe(M))
```

```
## Warning in Djump(data, scoef = scoef, Careajump = Careajump, Ctresh = Ctresh):
## There are several maximum jump
```

```
##
## Model selected by DDSE: 26
##
## Corresponding Slope interval: [1.22;1.63](28.6% of points)
## Selected model complexity: 26
##
##
## Model selected by Djump: 26
##
## Estimated penalty constant: 1.26
## Selected model complexity: 26
```

```
capushe(M)@Djump
```

```
## Warning in Djump(data, scoef = scoef, Careajump = Careajump, Ctresh = Ctresh):
## There are several maximum jump
```

```
## [1] "26"
```

```
S<-(capushe(M)@Djump)@model
```

```
## Warning in Djump(data, scoef = scoef, Careajump = Careajump, Ctresh = Ctresh):
## There are several maximum jump
```

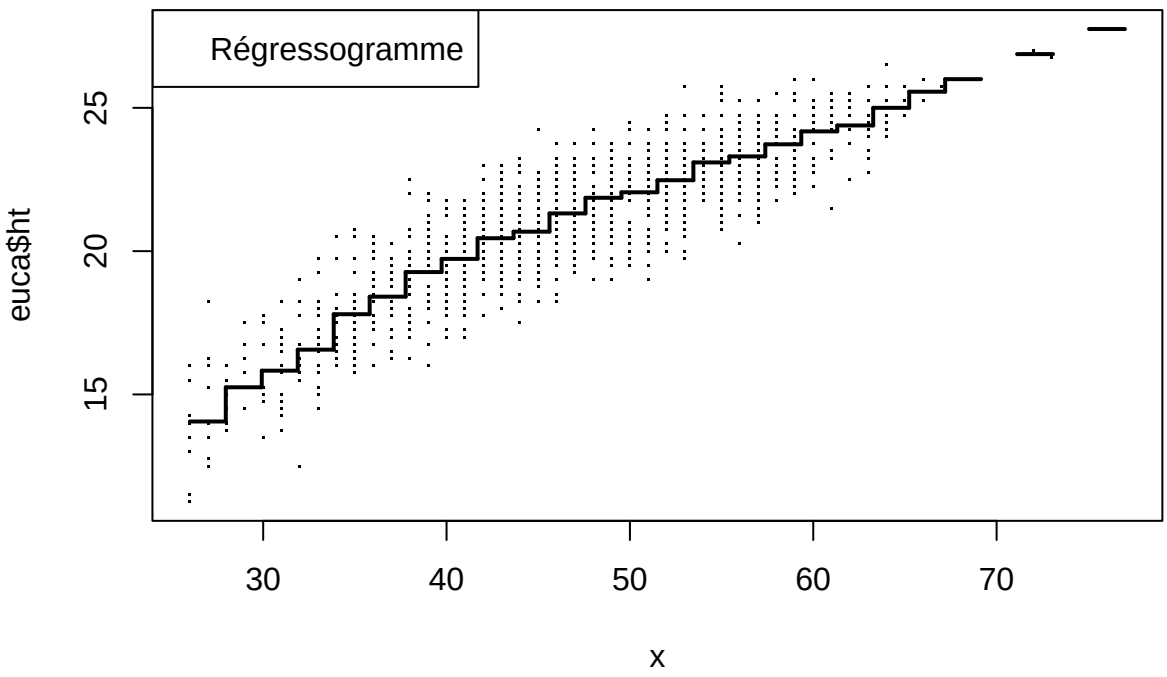
```
S<-as.numeric(S)
newx<-cut(euca$circ,S)
meany<-tapply(euca$ht,newx,mean)
xseq1<-(max(euca$circ)-min(euca$circ))*seq(0,1,by=1/S)+min(euca$circ)
length(xseq1)
```

```
## [1] 27
```

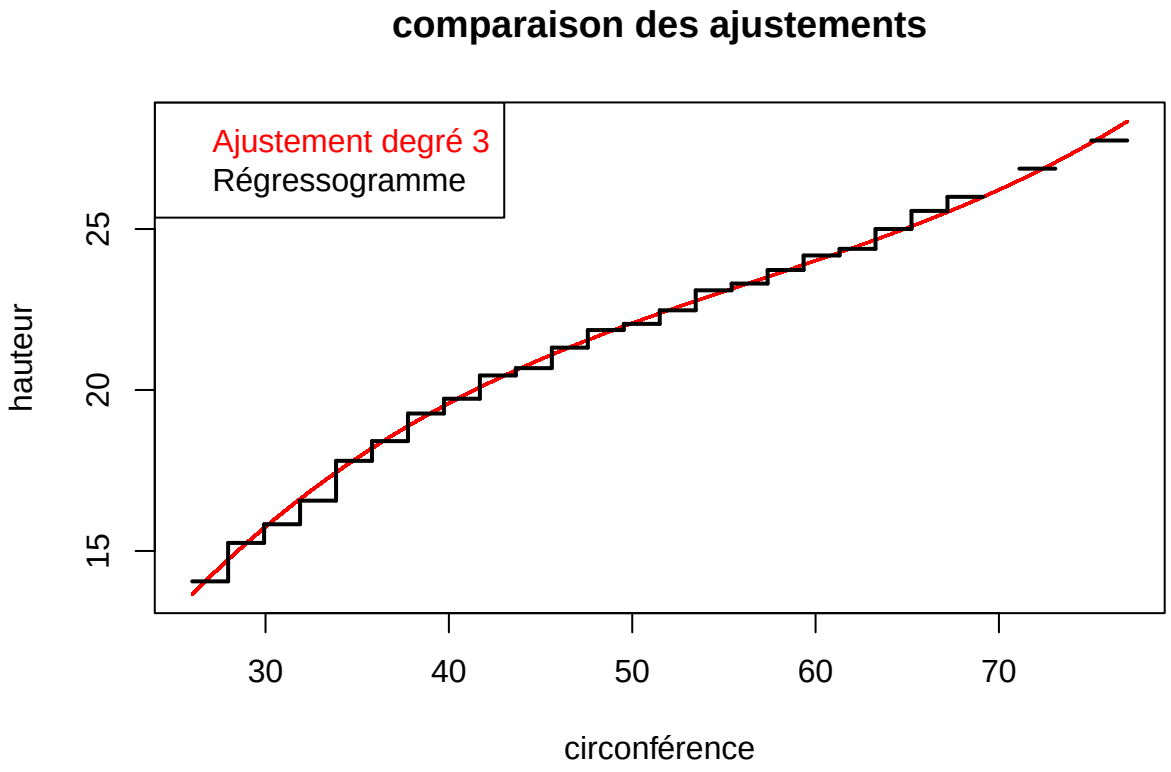
```
length(meany)
```

```
## [1] 26
```

```
plot(euca$circ, euca$ht, pch=".", xlab="x", main="Nuage de point et Régressogramme")
for(i in 1:S){lines(xseq1[i:(i+1)], c(meany[i], meany[i]), lwd=2)}
for(i in 1:(S-1)){lines(c(xseq1[i+1],xseq1[i+1]),meany[i:(i+1)], lwd=2)}
legend("topleft", legend=c( "Régressogramme"),
       text.col=c("black"))
```




```
## Comparaison du regressogramme sélectionné avec le polynôme de degré 3
plot(grille, pred3, pch=".", col="red", main="comparaison des ajustements", xlab="circonférence", ylab=
for(i in 1:S){lines(xseq1[i:(i+1)], c(meany[i], meany[i]), lwd=2)}
for(i in 1:(S-1)){lines(c(xseq1[i+1],xseq1[i+1]),meany[i:(i+1)], lwd=2)}
legend("topleft", legend=c("Ajustement degré 3",  "Régressogramme"),
       text.col=c("red",  "black"))
```


comparaison des ajustements
Ajustement degré 3
Régressogramme
hauteur
15
20
25
30
40
50
60
70
circonférence