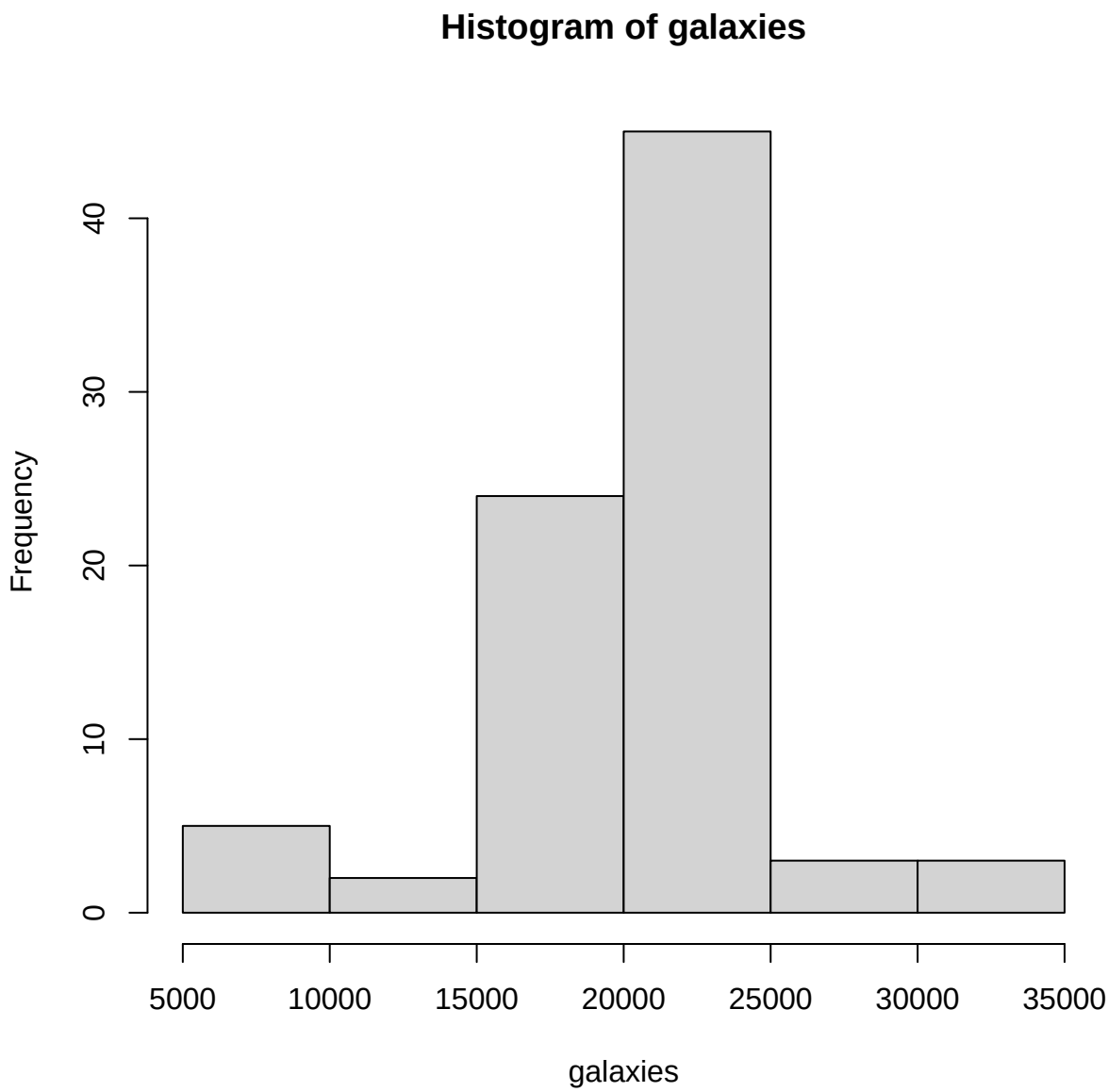
Chapter 1. Kernel density estimation

1.1. Histogram

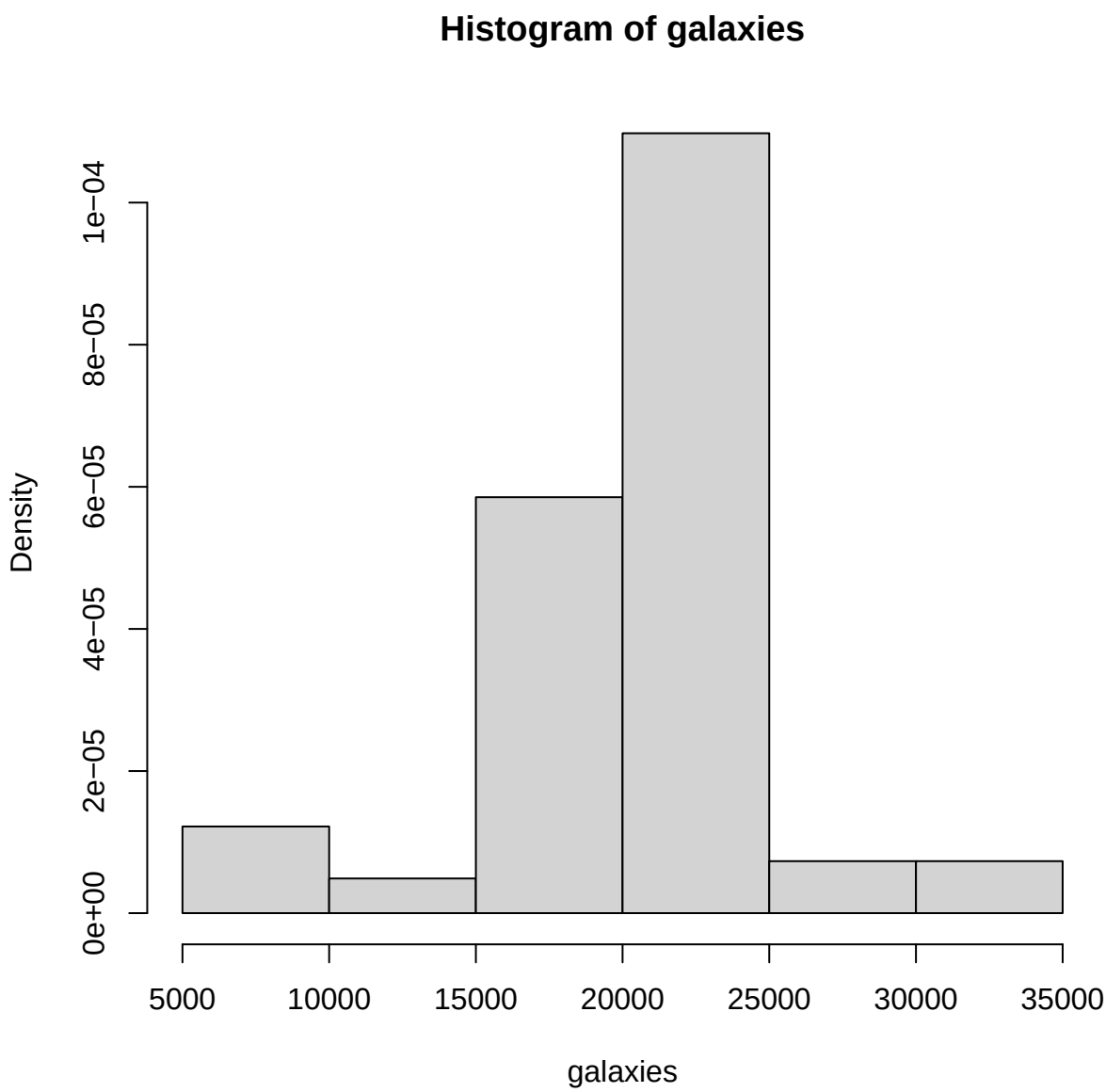
1.1.1. Real world data

```
library('sm')
library(MASS)
data(galaxies)
?galaxies
hist(galaxies)
```

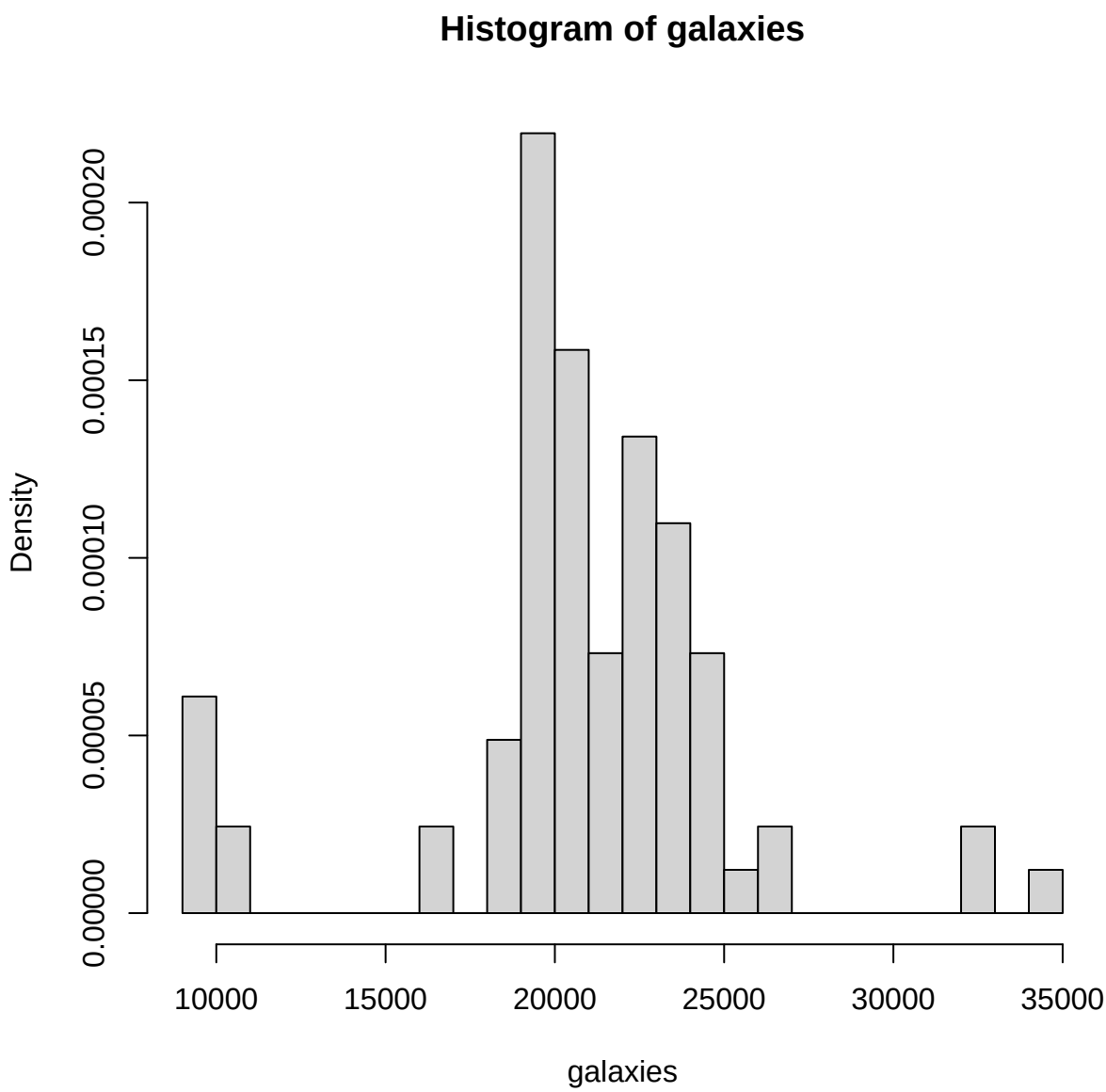
1.1.1.1. Galaxies



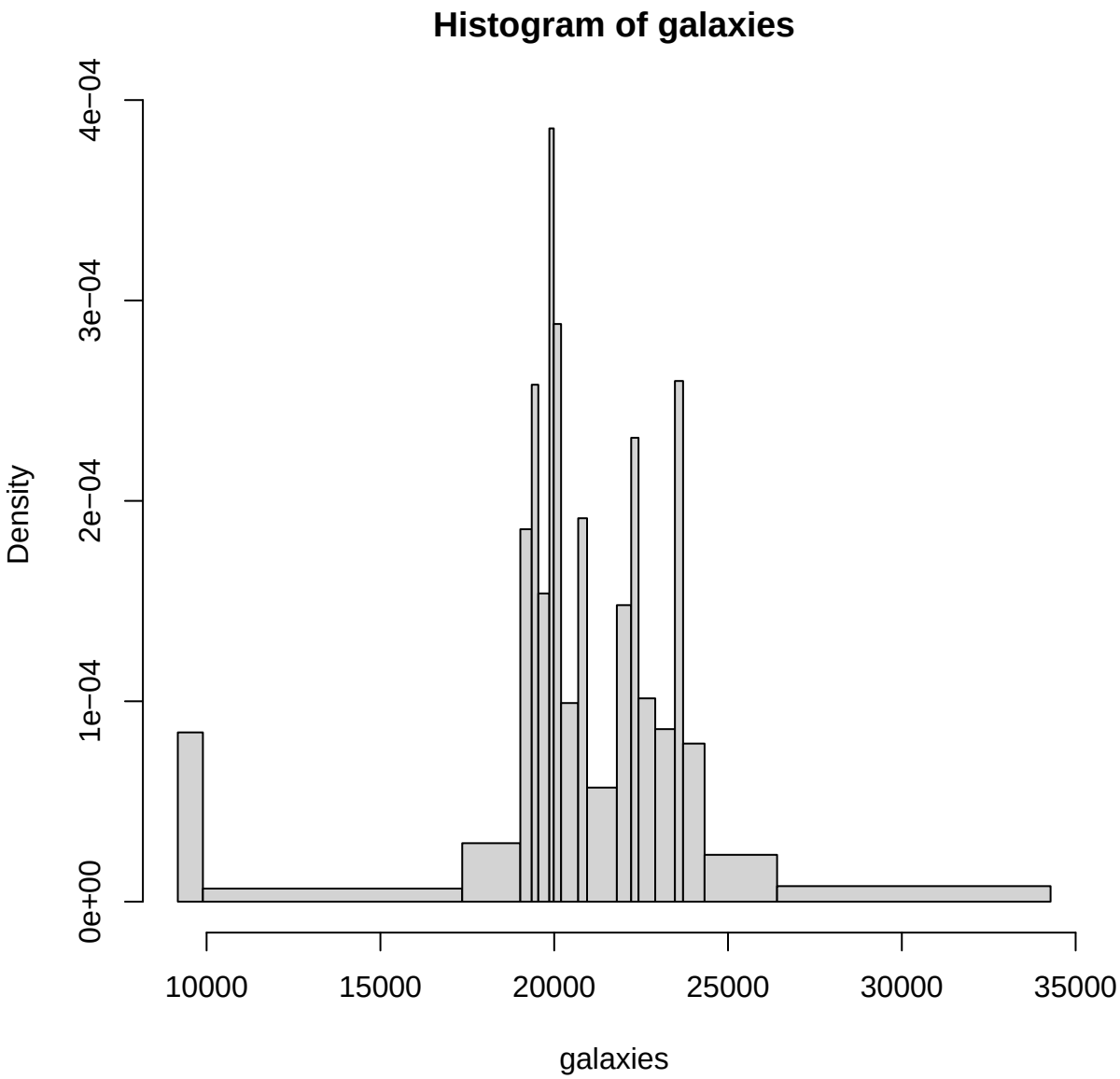

```
hist(galaxies,freq=F)
```




```r
hist(galaxies,freq=F,nclass=20)
```

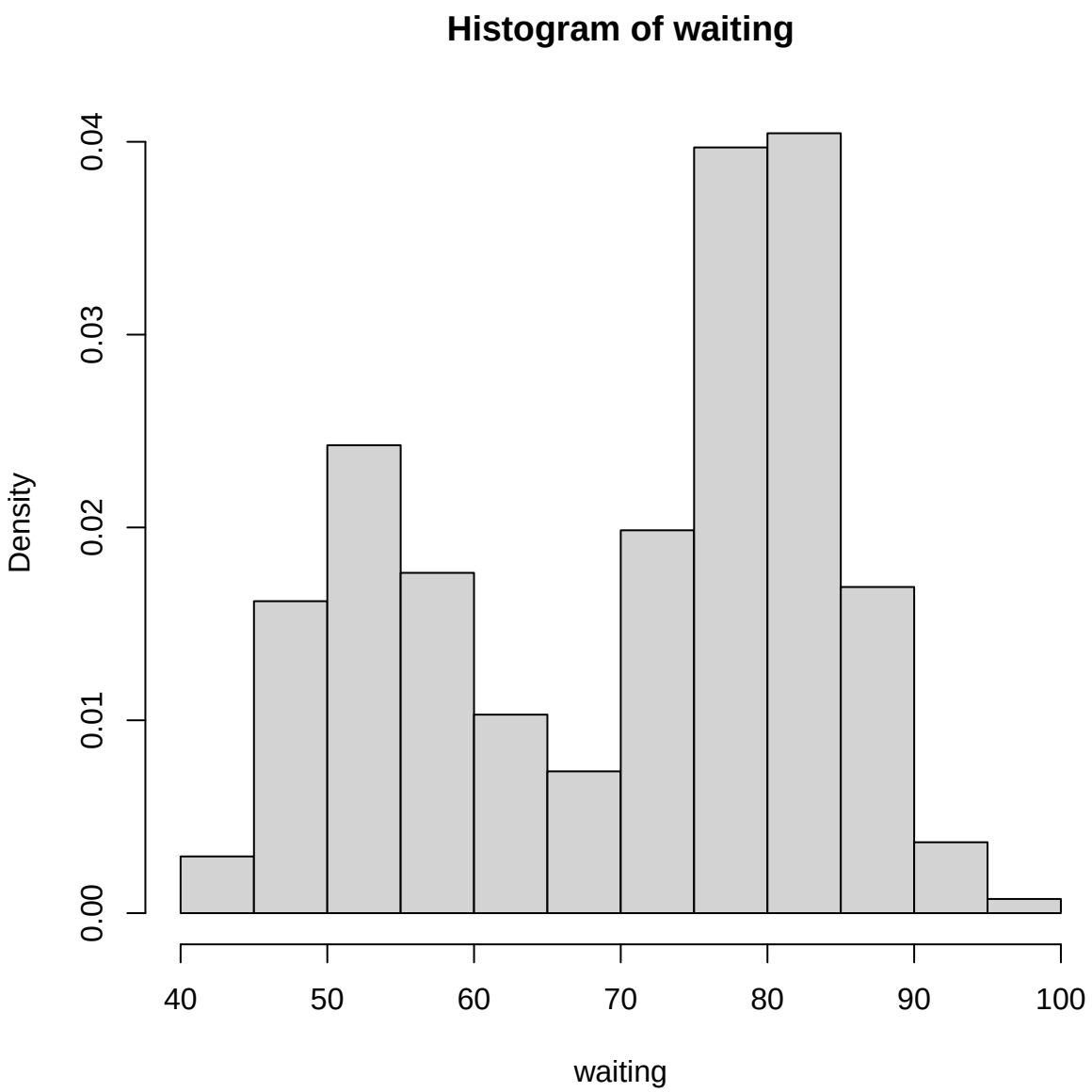



```r
hist(galaxies,breaks=quantile(galaxies,seq(0,1,len=20)))
```

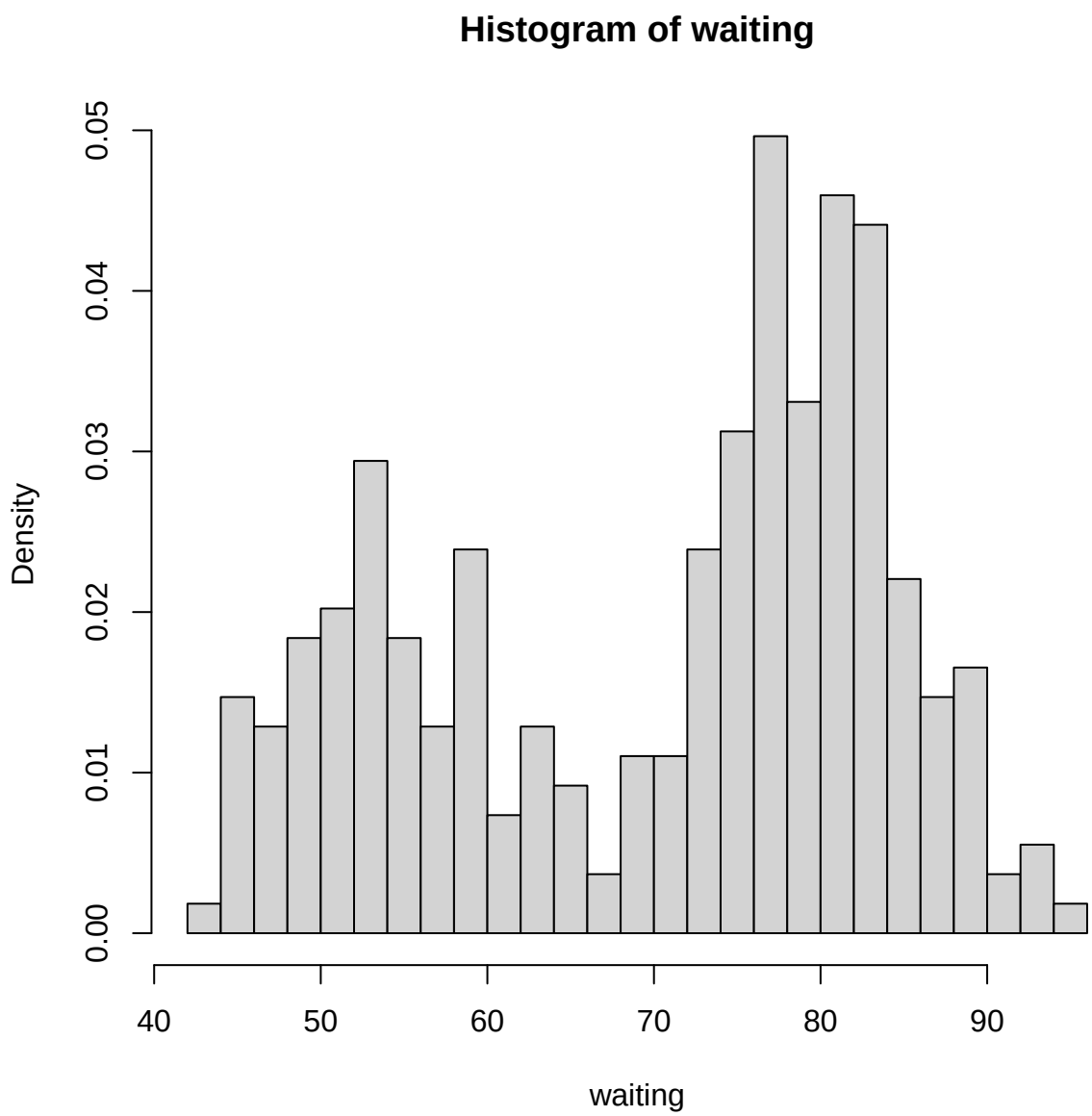



```
data(faithful)
attach(faithful)
?faithful
hist(waiting,freq=F)
```

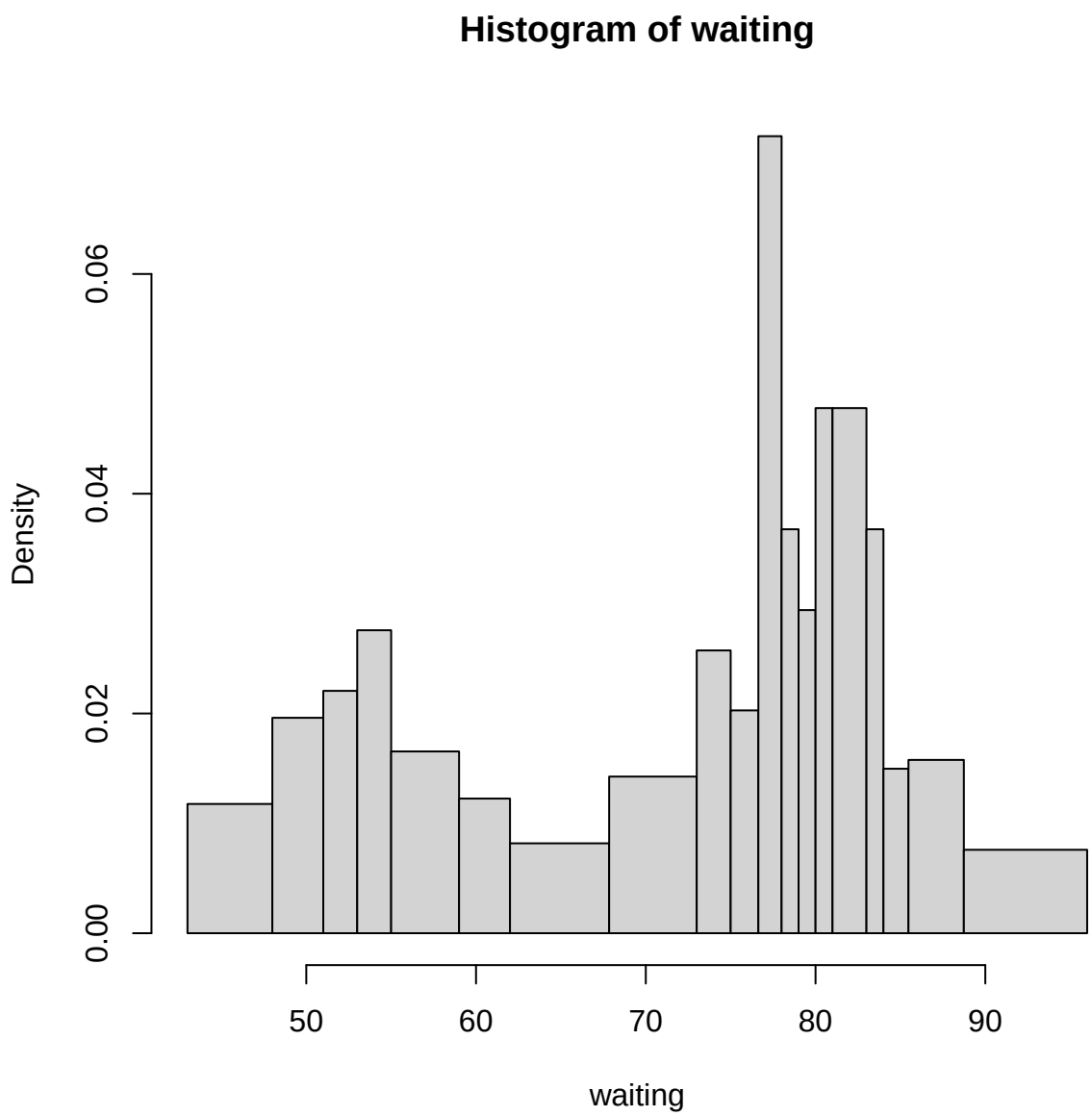
1.1.1.2. Faithful



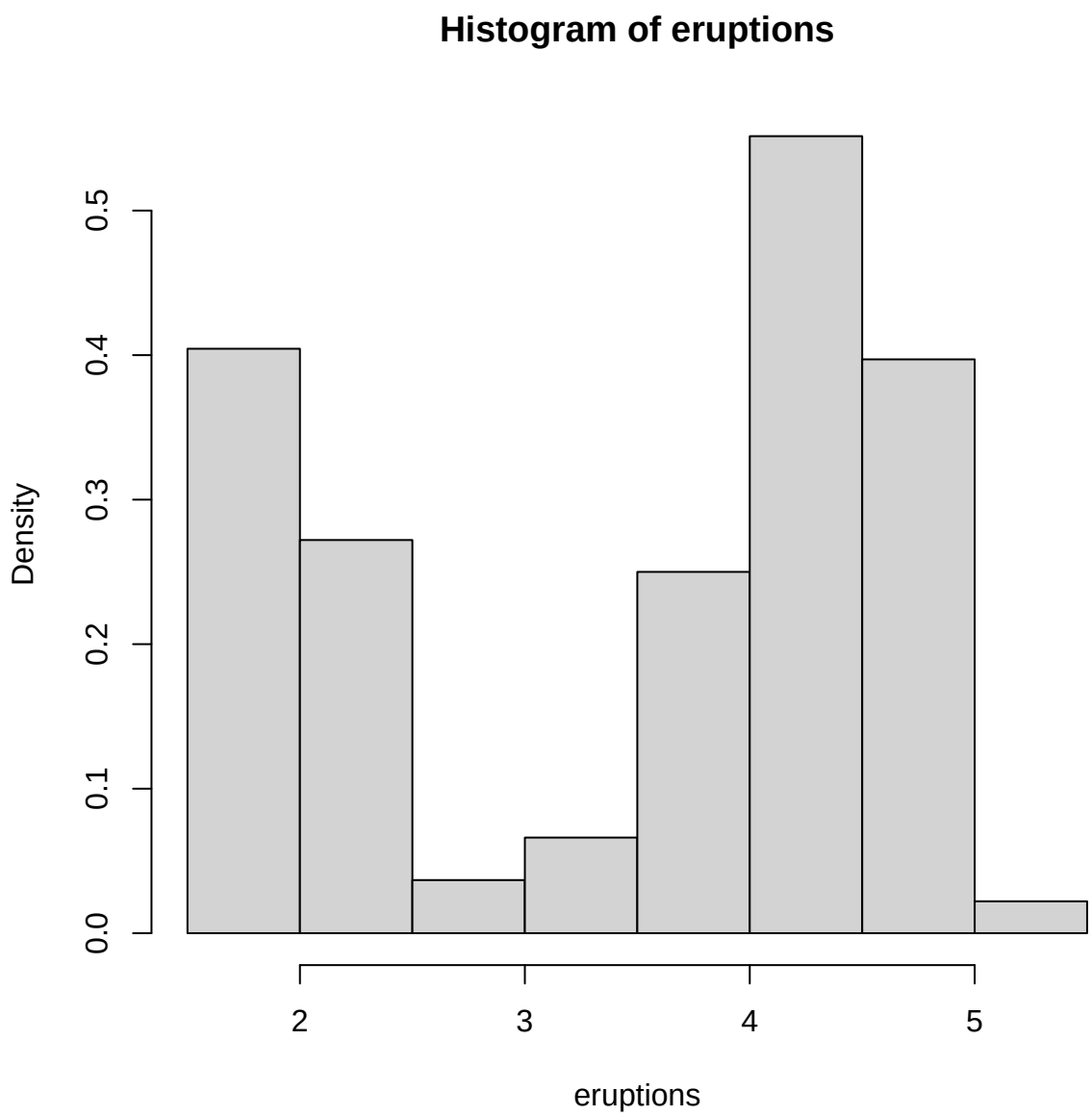

```r
hist(waiting,freq=F,nclass=20)
```



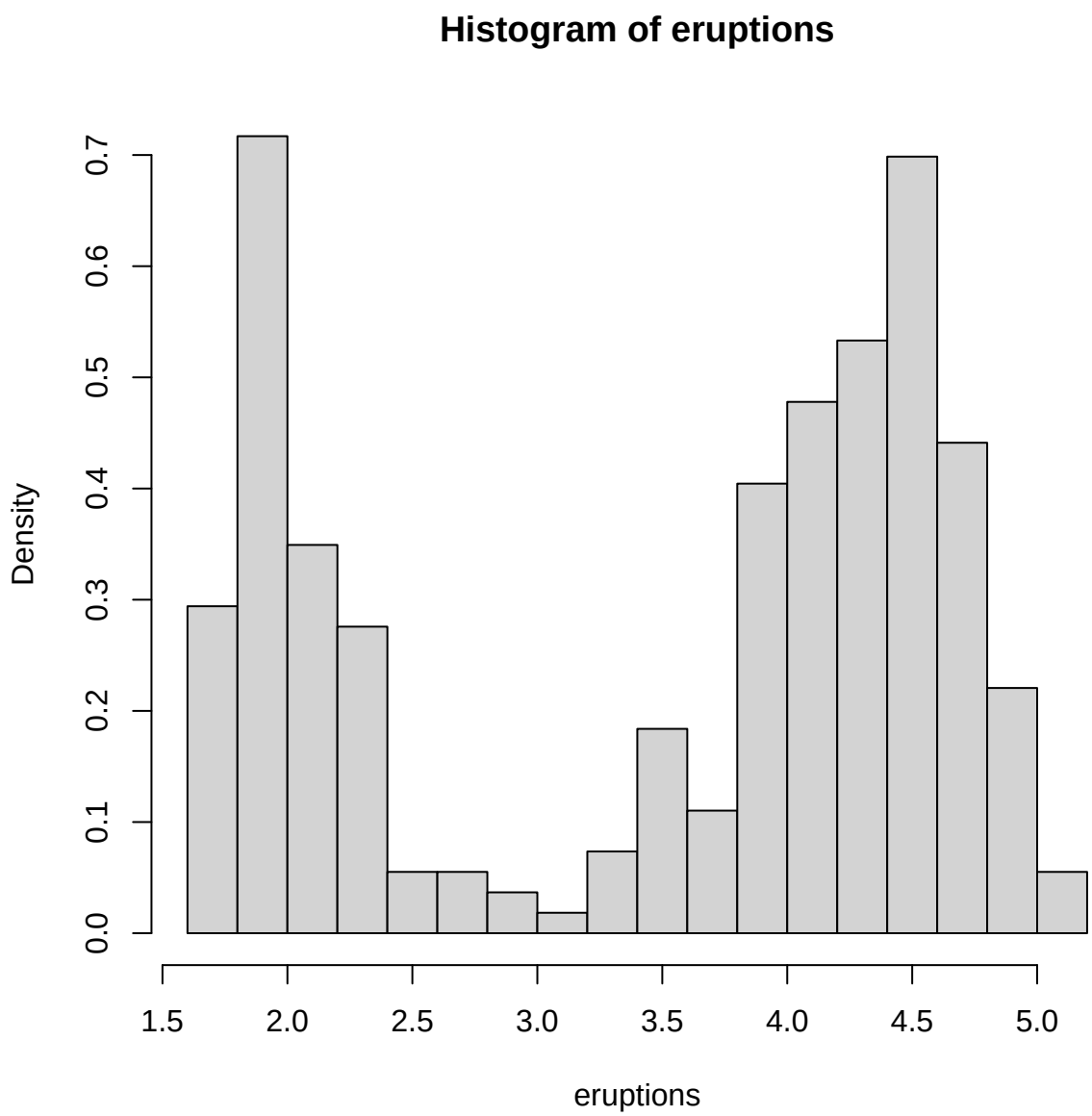

```r
hist(waiting,breaks=quantile(waiting,seq(0,1,len=20)))
```



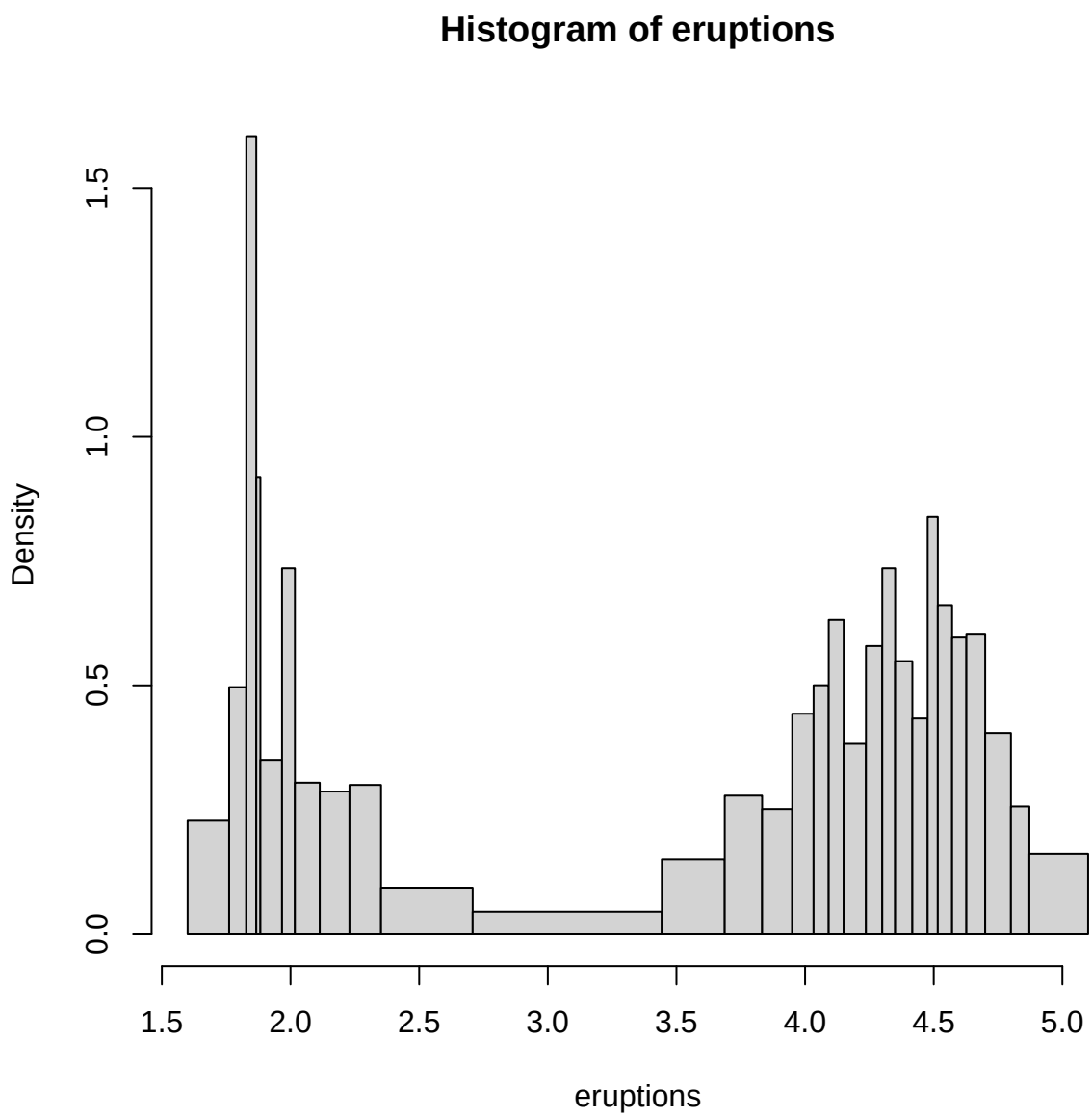

```
hist(eruptions,freq=F)
```




```r
hist(eruptions,freq=F,nclass=20)
```




```r
hist(eruptions,breaks=quantile(eruptions,seq(0,1,len=30)))
```

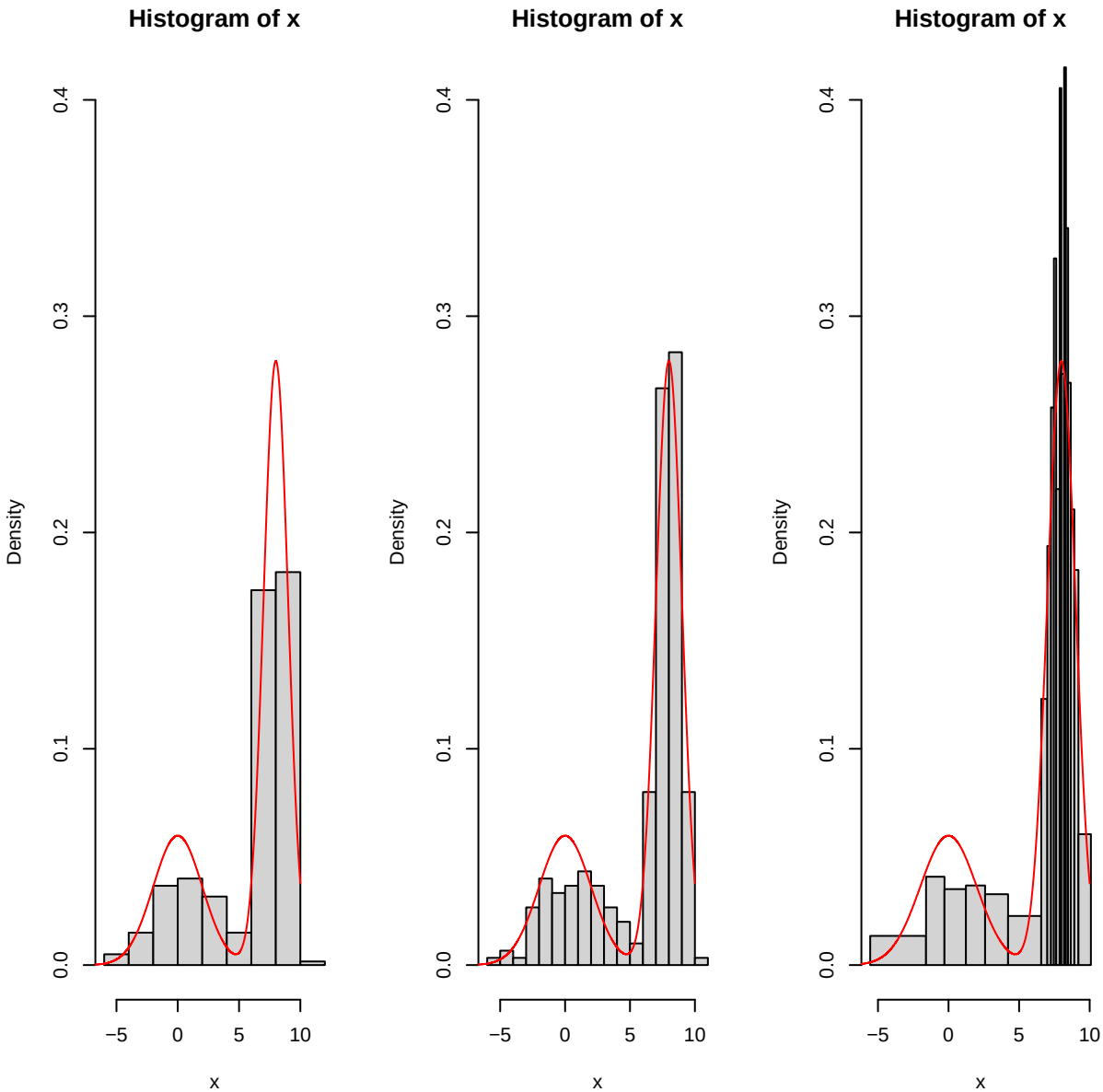


1.1.2. Simulation of a mixing model

$$f_X(x) = \alpha f_{\mathcal{N}(0,4)}(x) + (1-\alpha) f_{\mathcal{N}(8,1)}(x)$$

```
rmixing=function(n,alpha,l0,l1,p0,p1)
# Generate data from a mixing model
{
  z=rbinom(n,1,alpha)
  f1=eval(parse(text=paste('r',l1,'(',paste(c(n,p1),collapse=','),')',sep='')))
  f0=eval(parse(text=paste('r',l0,'(',paste(c(n,p0),collapse=','),')',sep='')))
  x=z*f1+(1-z)*f0
  return(x=x)
}

dmixing=function(t,alpha,l0,l1,p0,p1)
# draw the density of the mixing model
{
```

```r
  res=alpha*eval(parse(text=paste('d',l1,'(t,',paste(p1,collapse=','),')',sep='')))+(1-alpha)*eval(pars
}

#Example
n=300
alpha=0.3
l0='norm'
p0=c(8,1)
l1='norm'
p1=c(0,2)
s=seq(-10,10,0.001)

x=rmixing(n,alpha,l0,l1,p0,p1)

#### histogram
par(mfrow=c(1,3))
hist(x,freq=F,ylim=c(0,0.4))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
hist(x,freq=F,ylim=c(0,0.4),nclass=20)
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
hist(x,breaks=quantile(x,seq(0,1,len=20)),ylim=c(0,0.4))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```



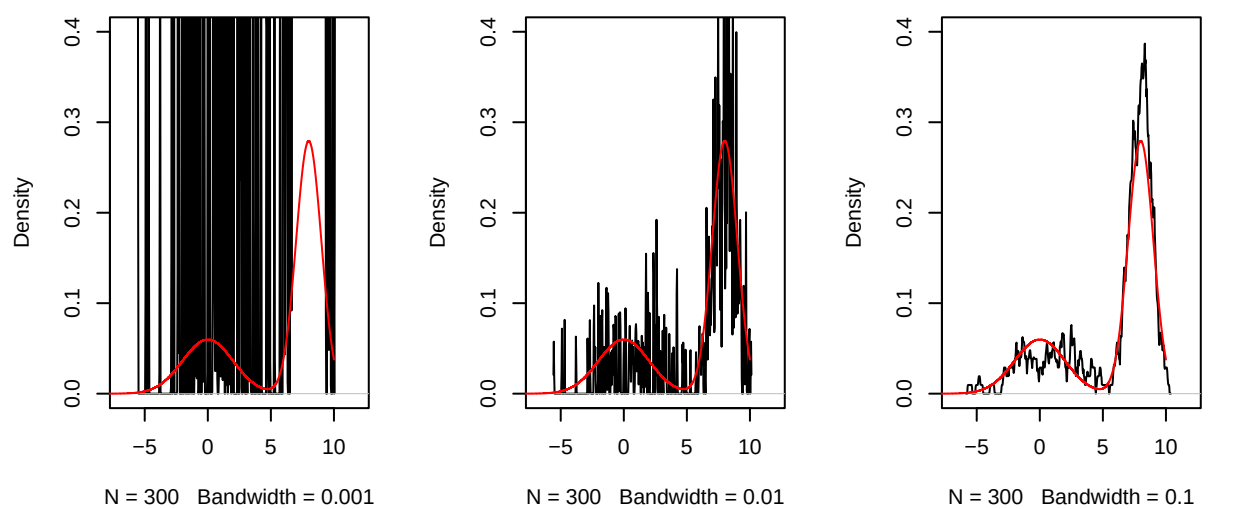
1.2. Moving window estimator

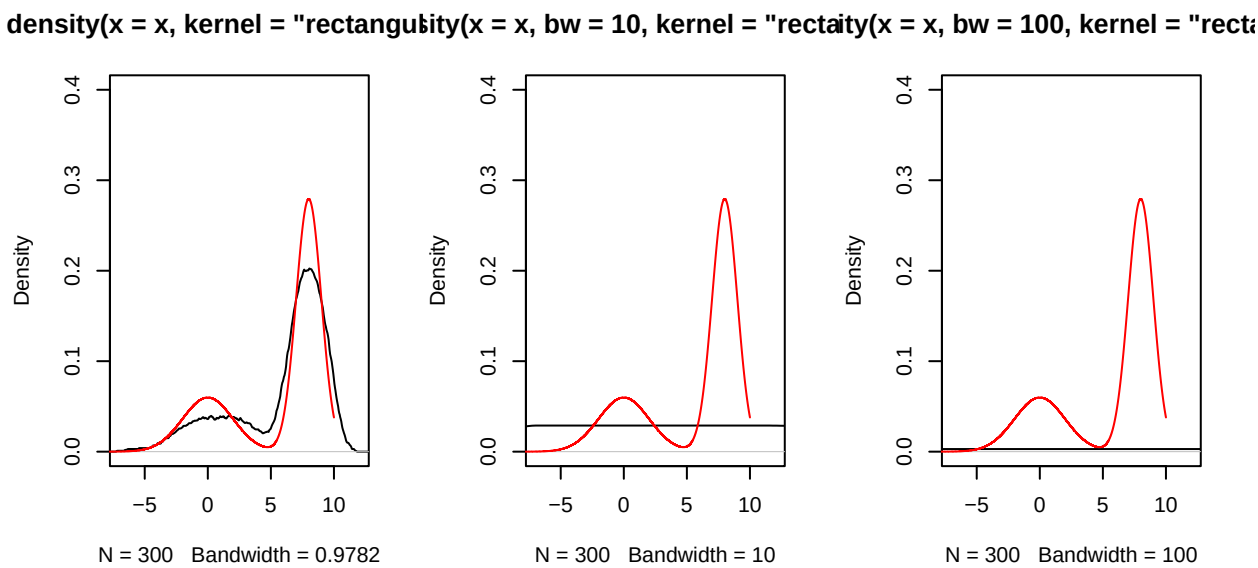
1.2.1. On the simulated mixing model

$$f_X(x) = \alpha f_{\mathcal{N}(0,4)}(x) + (1-\alpha) f_{\mathcal{N}(8,1)}(x)$$

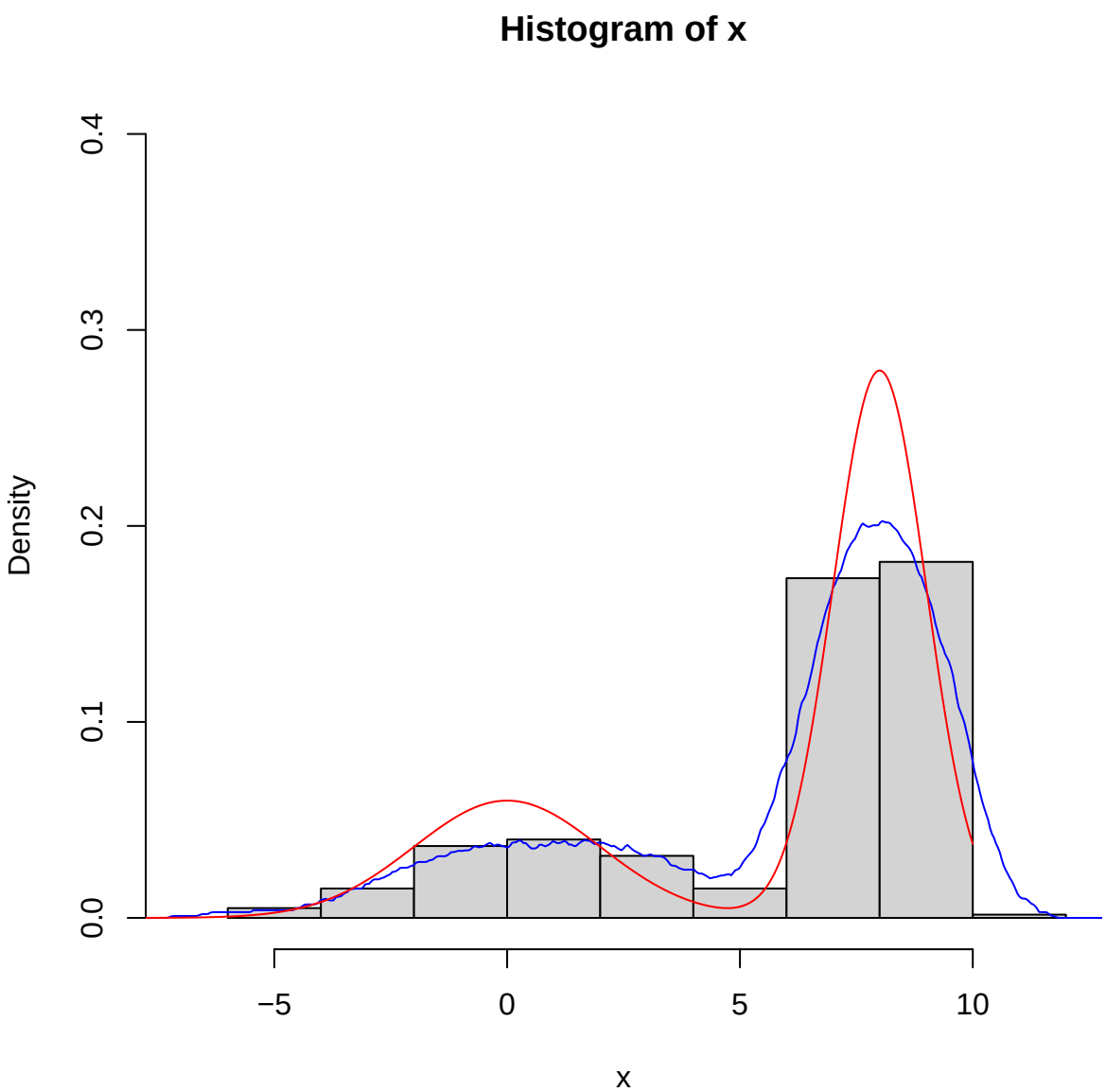
```
par(mfrow=c(2,3))
plot(density(x,bw=0.001,kernel='rectangular'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=0.01,kernel='rectangular'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=0.1,kernel='rectangular'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,kernel='rectangular'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=10,kernel='rectangular'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```

```r
plot(density(x,kernel='rectangular',bw=100),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```



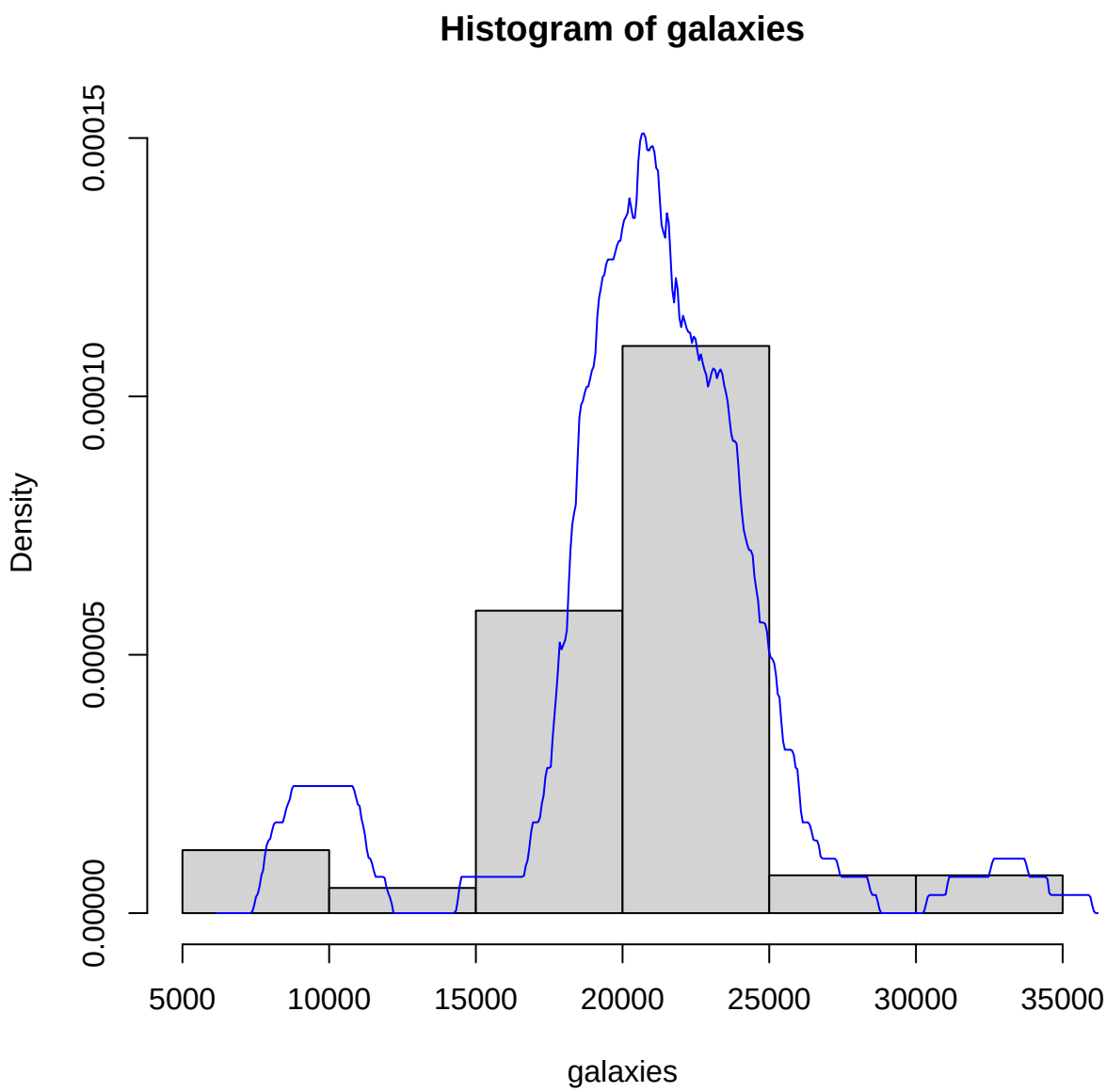



```r
par(mfrow=c(1,1))
hist(x,freq=F,ylim=c(0,0.4),xlim=c(-7,12))
lines(density(x,kernel='rectangular'),col='blue')
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```

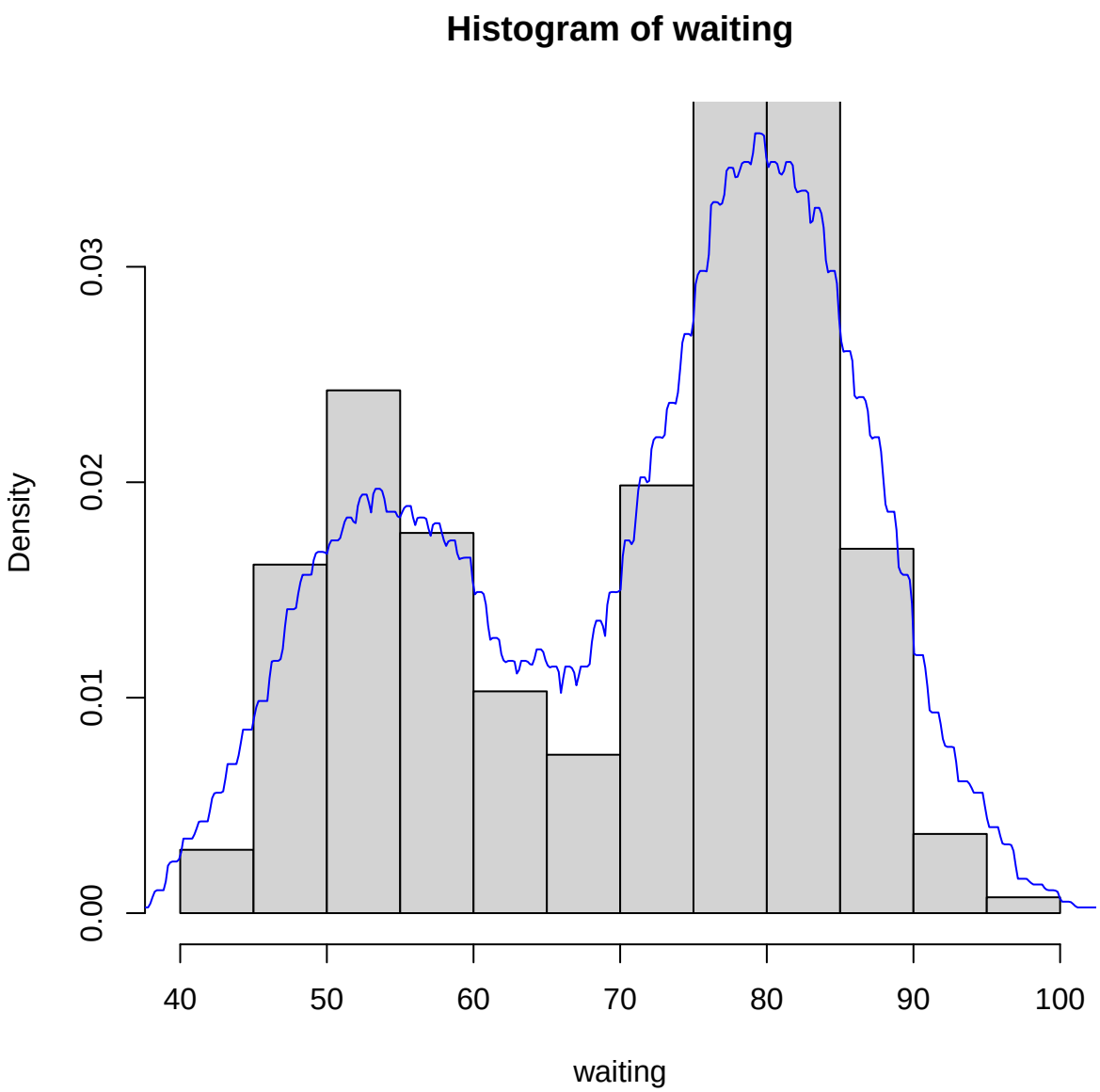


1.2.2. Come back to real world data

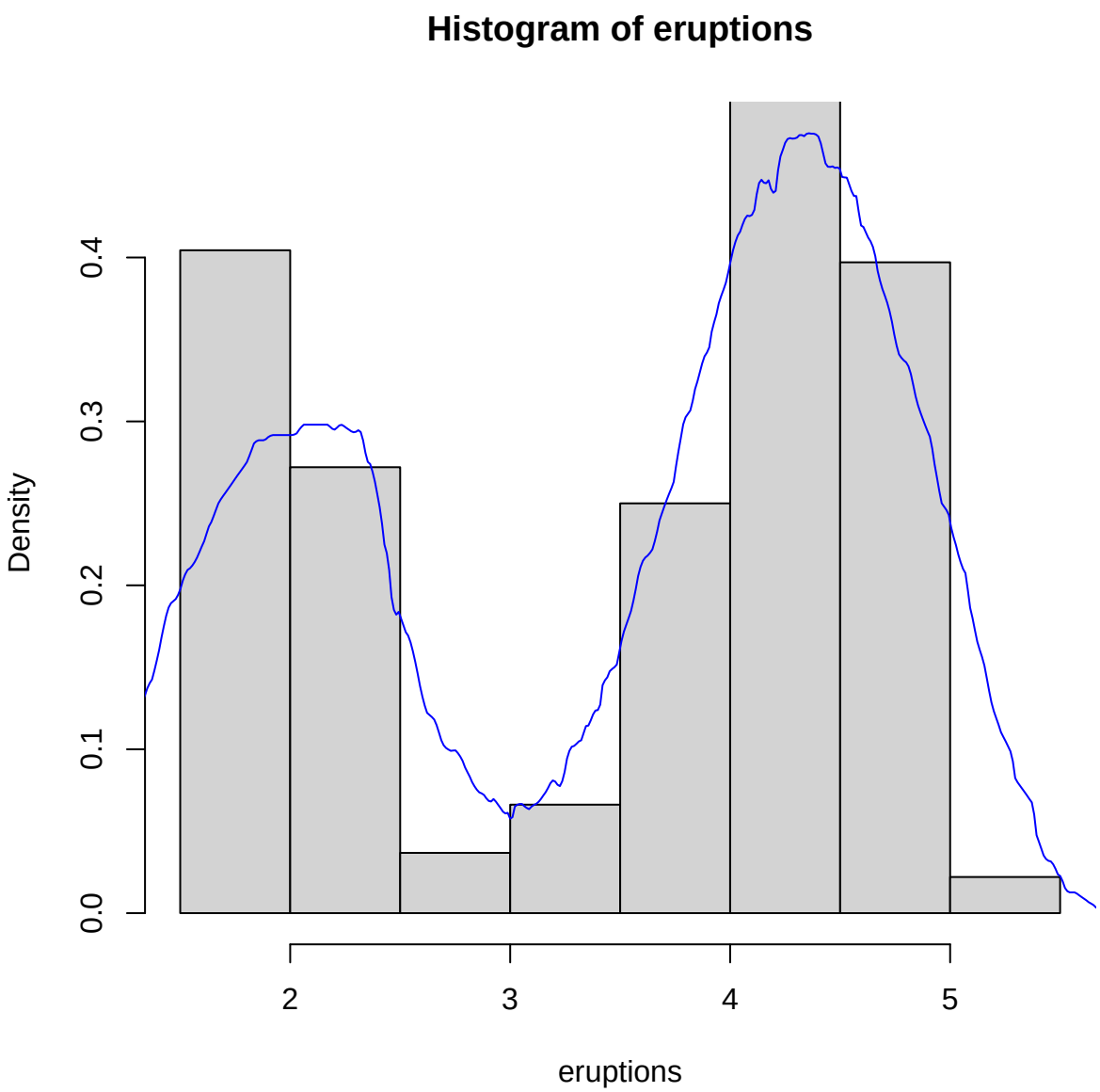
```
# Galaxies
hist(galaxies,freq=F,ylim=range(density(galaxies,kernel='rectangular')$y))
lines(density(galaxies,kernel='rectangular'),col='blue')
```




```r
# Faithful
hist(waiting,freq=F,ylim=range(density(waiting,kernel='rectangular')$y))
lines(density(waiting,kernel='rectangular'),col='blue')
```




```
hist(eruptions,freq=F,ylim=range(density(eruptions,kernel='rectangular')$y))
lines(density(eruptions,kernel='rectangular'),col='blue')
```



1.3. Kernel estimator

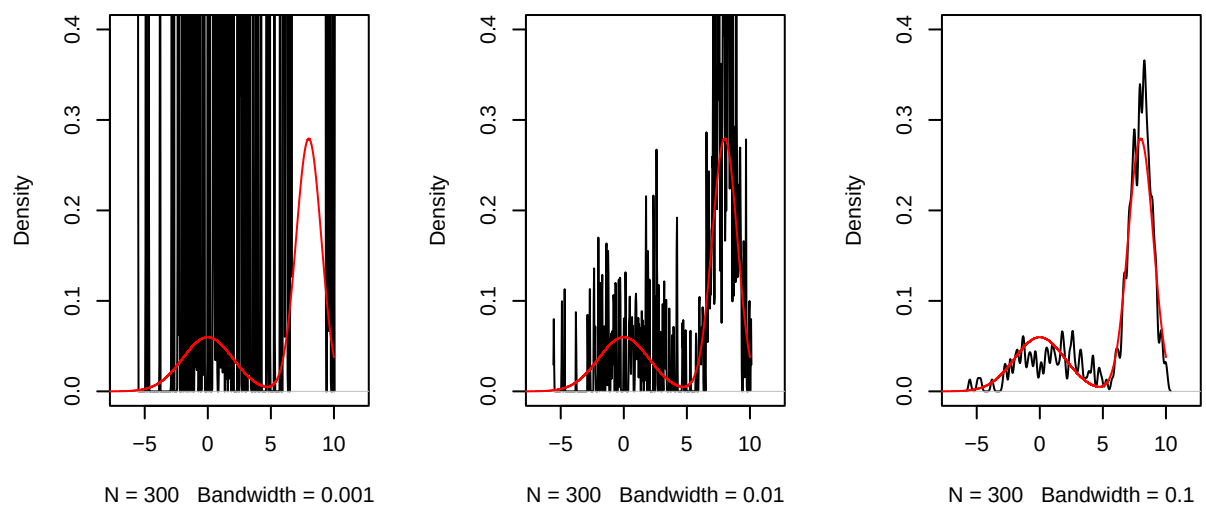
1.3.1. On the simulated mixing model

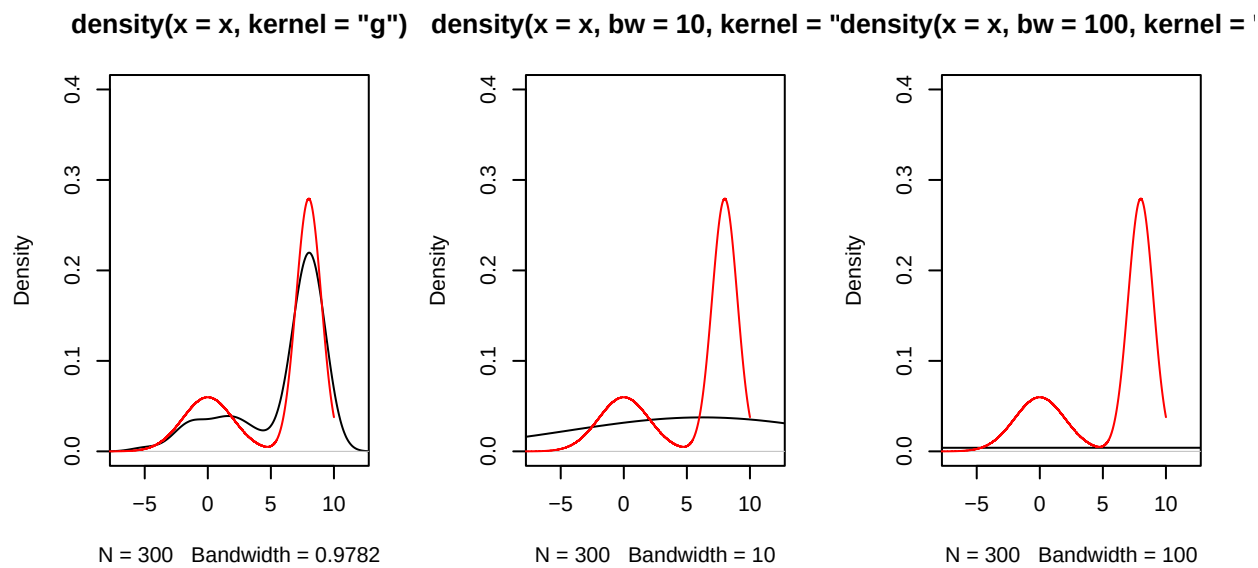
$$f_X(x) = \alpha f_{\mathcal{N}(0,4)}(x) + (1-\alpha) f_{\mathcal{N}(8,1)}(x)$$

```
par(mfrow=c(2,3))
plot(density(x,bw=0.001,kernel='g'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=0.01,kernel='g'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=0.1,kernel='g'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,kernel='g'),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=10,kernel='g'),ylim=c(0,0.4),xlim=c(-7,12))
```

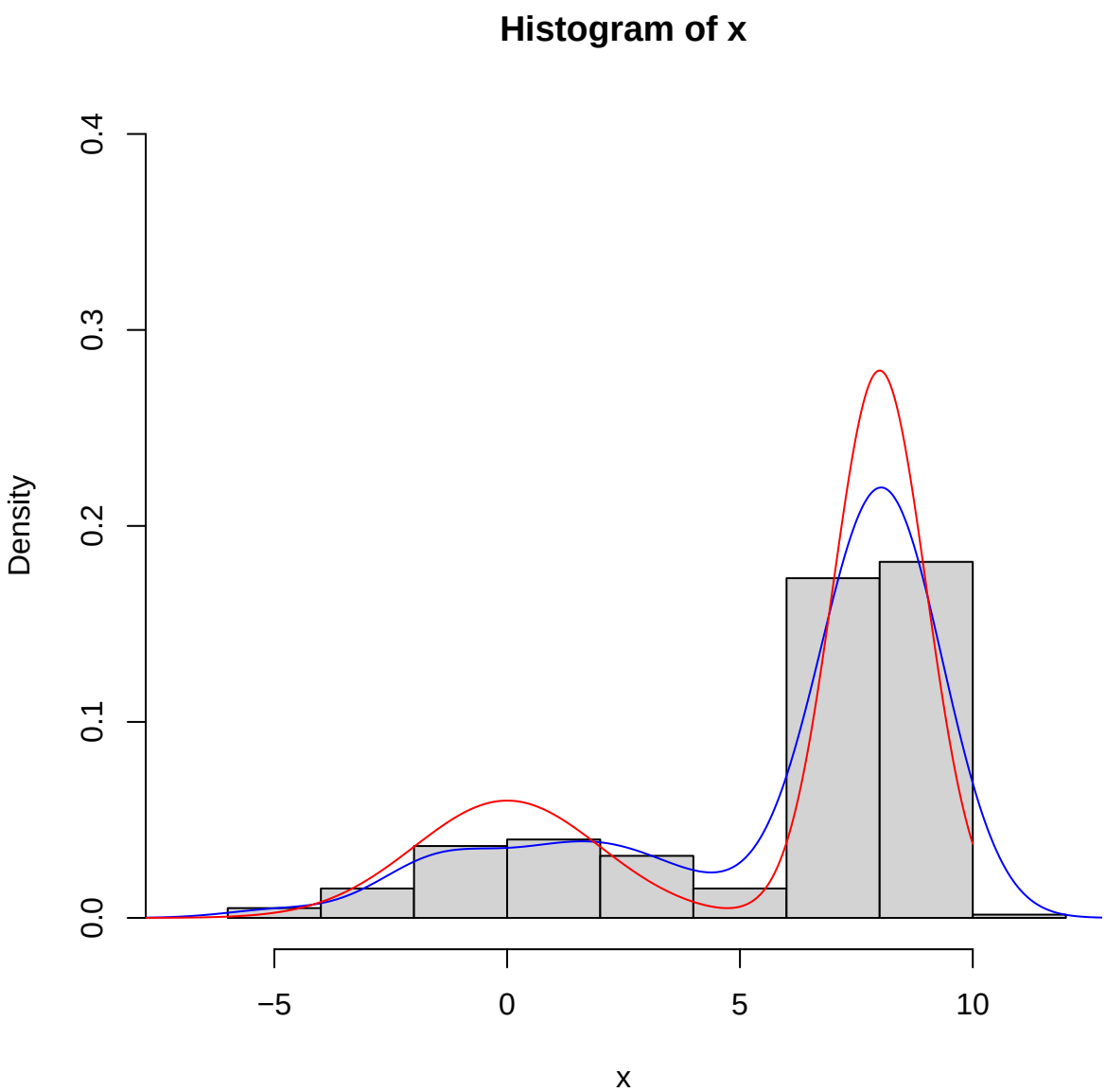
```
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,kernel='g',bw=100),ylim=c(0,0.4),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```

Effect of h value



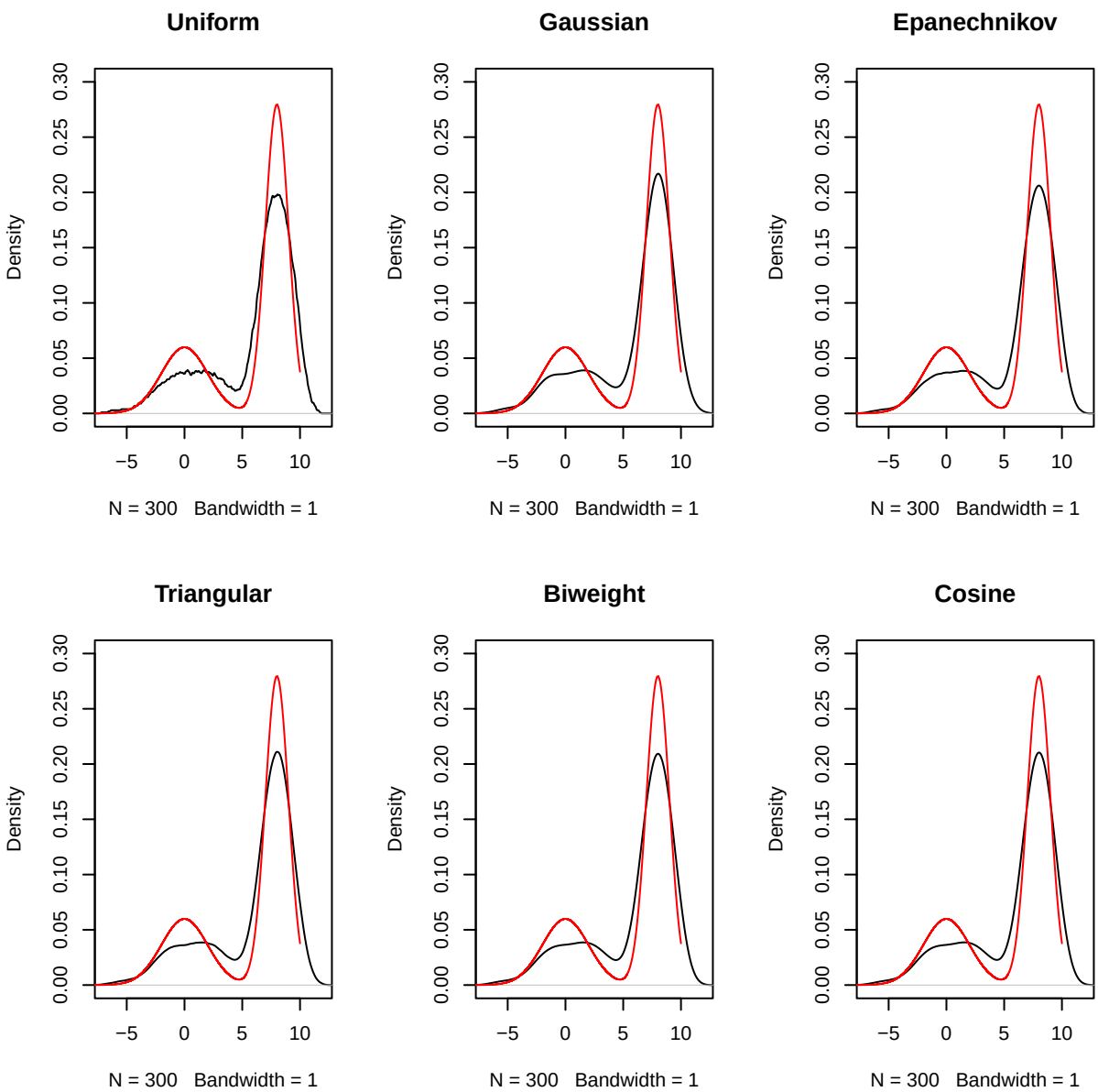



```
par(mfrow=c(1,1))
hist(x,freq=F,ylim=c(0,0.4),xlim=c(-7,12))
lines(density(x,kernel='g'),col='blue')
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```



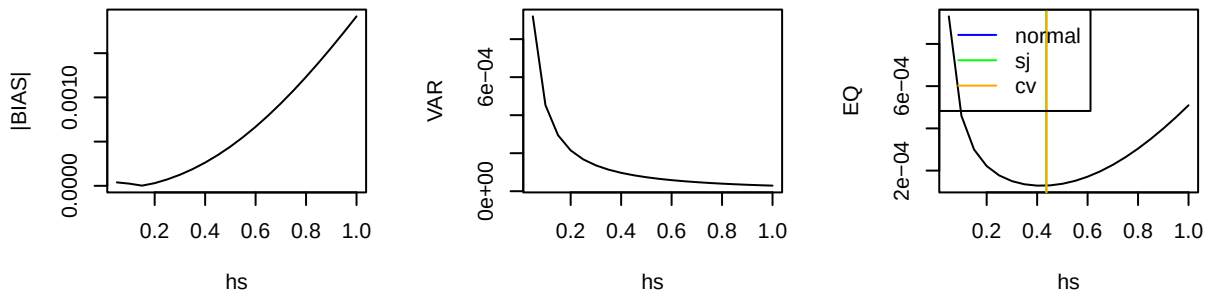

```
par(mfrow=c(2,3))
plot(density(x,bw=1,kernel='r'),main='Uniform',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=1,kernel='g'),main='Gaussian',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=1,kernel='e'),main='Epanechnikov',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=1,kernel='triangular'),main='Triangular',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,bw=1,kernel='b'),main='Biweight',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
plot(density(x,kernel='cosine',bw=1),main='Cosine',ylim=c(0,0.3),xlim=c(-7,12))
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')
```

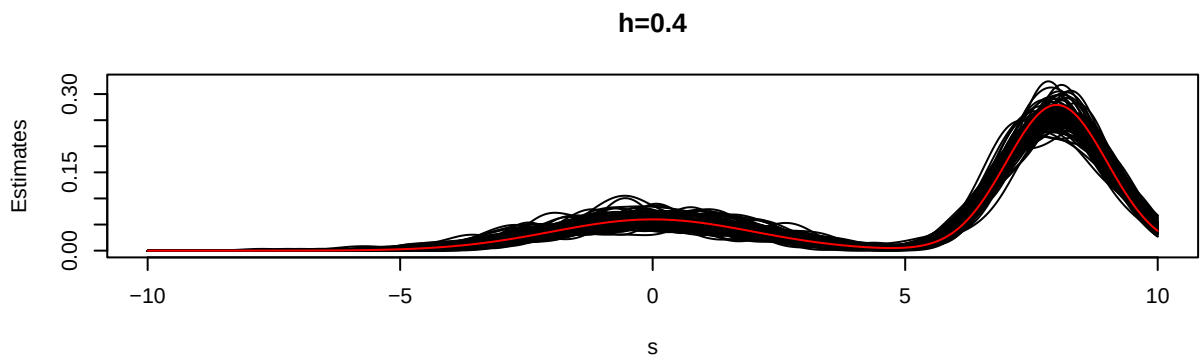
Effect of K value

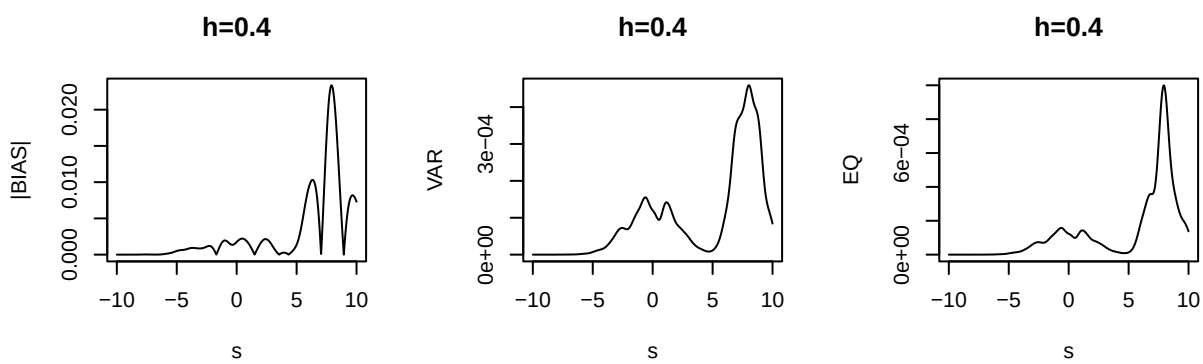



```
######## Quadratic loss
# number of simulations
J=100
hs=(1:20)/20
s=seq(-10,10,0.01)

h0=5
s0=1001


QUAD_LOSS=function(s,hs,J,n,alpha,l0,l1,p0,p1,h0,s0)
{
ls= length(s)
lh=length(hs)
EST=array(NA,c(J,ls,lh))
for (j in 1:J)
```

```
{
  x=rmixing(n,alpha,l0,l1,p0,p1)
  for (h in 1:lh)
     {
      EST[j,,h]=sm.density(x,h=hs[h],display='none',ylim=c(0,0.4),nbins=0,eval.points=s,poly.index=0)$es
     }
}
BIAS=apply(EST,c(2,3),mean)-dmixing(s,alpha,l0,l1,p0,p1)
VAR=apply(EST,c(2,3),var)
EQ=BIAS^2+VAR

nl=2
if (!is.null(s0)) nl=nl+1

layout(matrix(c(1:3,rep(4,3),5:(3*nl+1)),byrow=TRUE, ncol=3))
plot(hs,abs(apply(BIAS,2,mean)),type='l',ylab='|BIAS|')
plot(hs,apply(VAR,2,mean),type='l',ylab='VAR')
plot(hs,apply(EQ,2,mean),type='l',ylab='EQ')
abline(v=sm.density(x,method='normal',display="none",poly.index=0)$h,col='blue')
abline(v=sm.density(x,method='sj',display="none",poly.index=0)$h,col='green')
abline(v=sm.density(x,method='cv',display="none",poly.index=0)$h,col='orange')
legend(x="topleft",leg=c('normal','sj','cv'), col=c("blue","green","orange"),lty=1)

hopt=which(apply(EQ,2,mean)==min(apply(EQ,2,mean)))
if (is.null(h0)) h0=hopt
EST2=EST[,,h0]

plot(s,EST2[1,],type='l',ylab='Estimates',main=paste('h=',hs[h0],sep=''),ylim=range(EST2))
for (j in 1:J) lines(s,EST2[j,])
lines(s,dmixing(s,alpha,l0,l1,p0,p1),col='red')

if (!is.null(h0))
  {
  plot(s,abs(BIAS[,h0]),type='l',ylab='|BIAS|',main=paste('h=',hs[h0],sep=''))
  plot(s,VAR[,h0],type='l',ylab='VAR',main=paste('h=',hs[h0],sep=''))
  plot(s,EQ[,h0],type='l',ylab='EQ',main=paste('h=',hs[h0],sep=''))
  }
if (!is.null(s0))
  {
  plot(hs,abs(BIAS[s0,]),type='l',ylab='|BIAS|',main=paste('s=',s[s0],sep=''))
  plot(hs,VAR[s0,],type='l',ylab='VAR',main=paste('s=',s[s0],sep=''))
  plot(hs,EQ[s0,],type='l',ylab='EQ',main=paste('s=',s[s0],sep=''))
  }
return(list(BIAS=BIAS,VAR=VAR,EQ=EQ,hopt=hs[hopt]))
}

RES=QUAD_LOSS(s,hs,J,n,alpha,l0,l1,p0,p1,NULL,NULL)
```

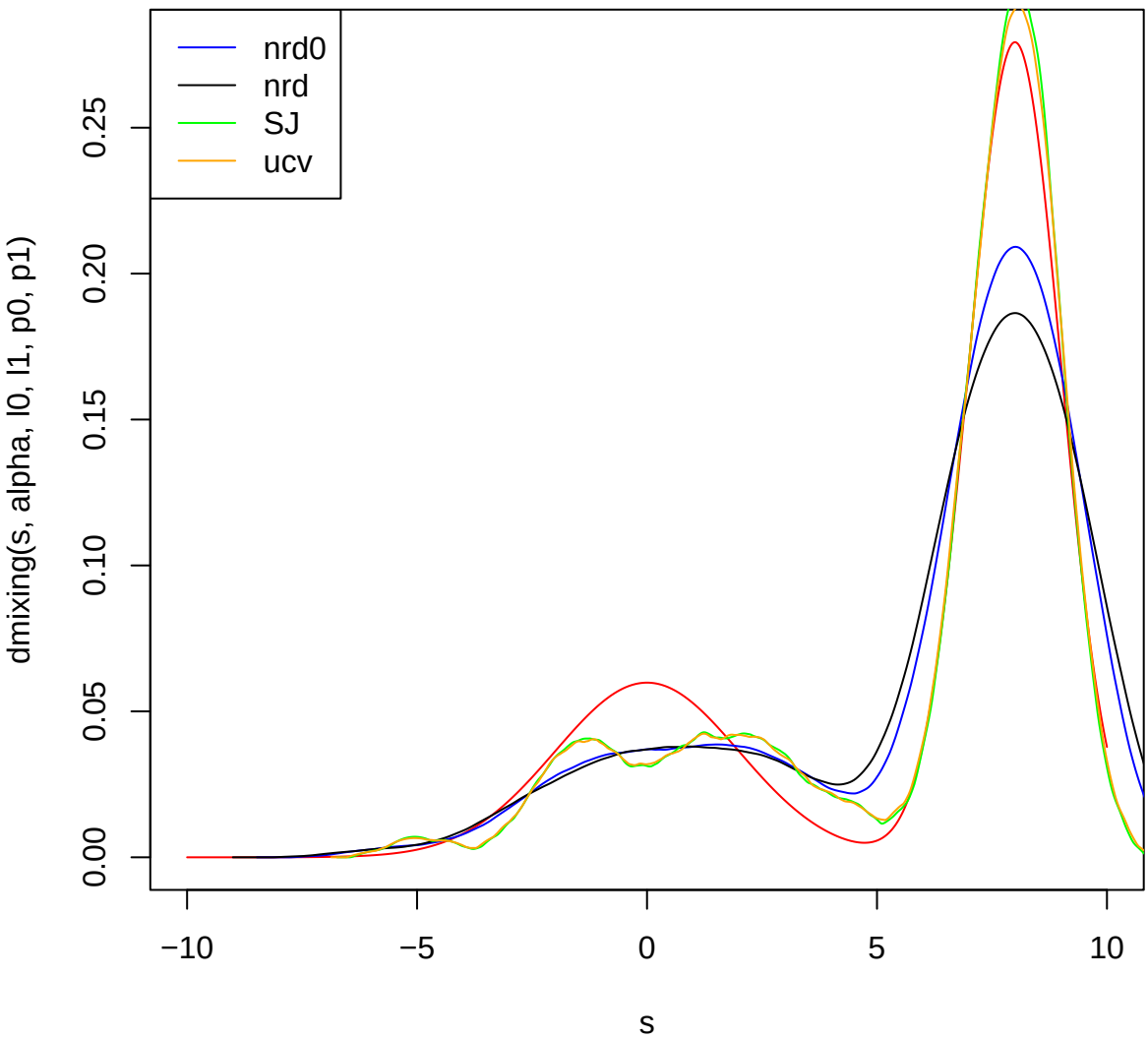




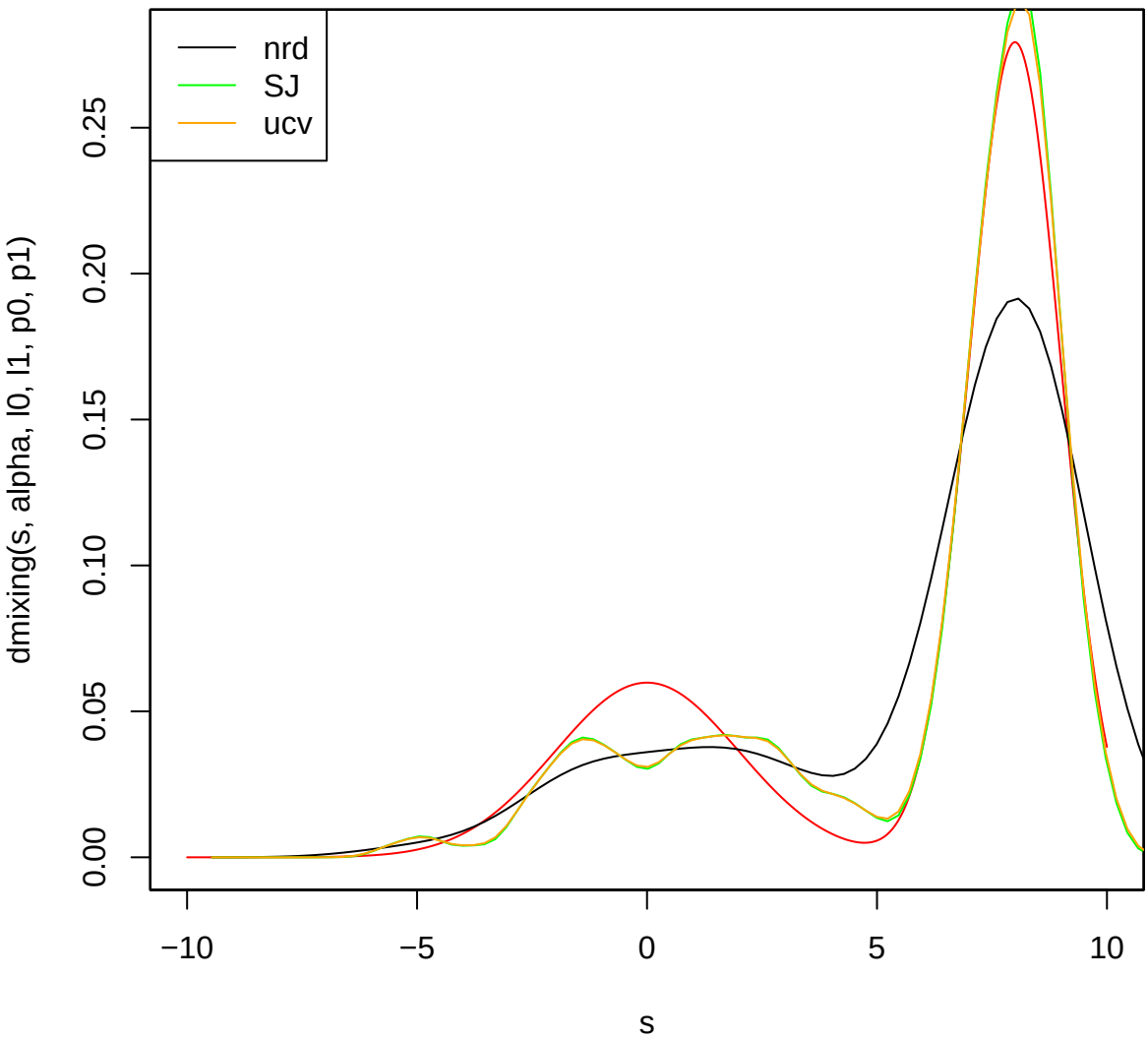



```
plot(s,dmixing(s,alpha,l0,l1,p0,p1),col='red',type='l')
lines(density(x,kernel='e'),col='blue')
lines(density(x,bw='nrd',kernel='e'))
lines(density(x,bw='SJ',kernel='e'),col='green')
lines(density(x,bw='ucv',kernel='e'),col='orange')
legend(x="topleft",leg=c('nrd0','nrd','SJ','ucv'), col=c("blue",'black',"green","orange"),lty=1)
```

Automatic choice of h

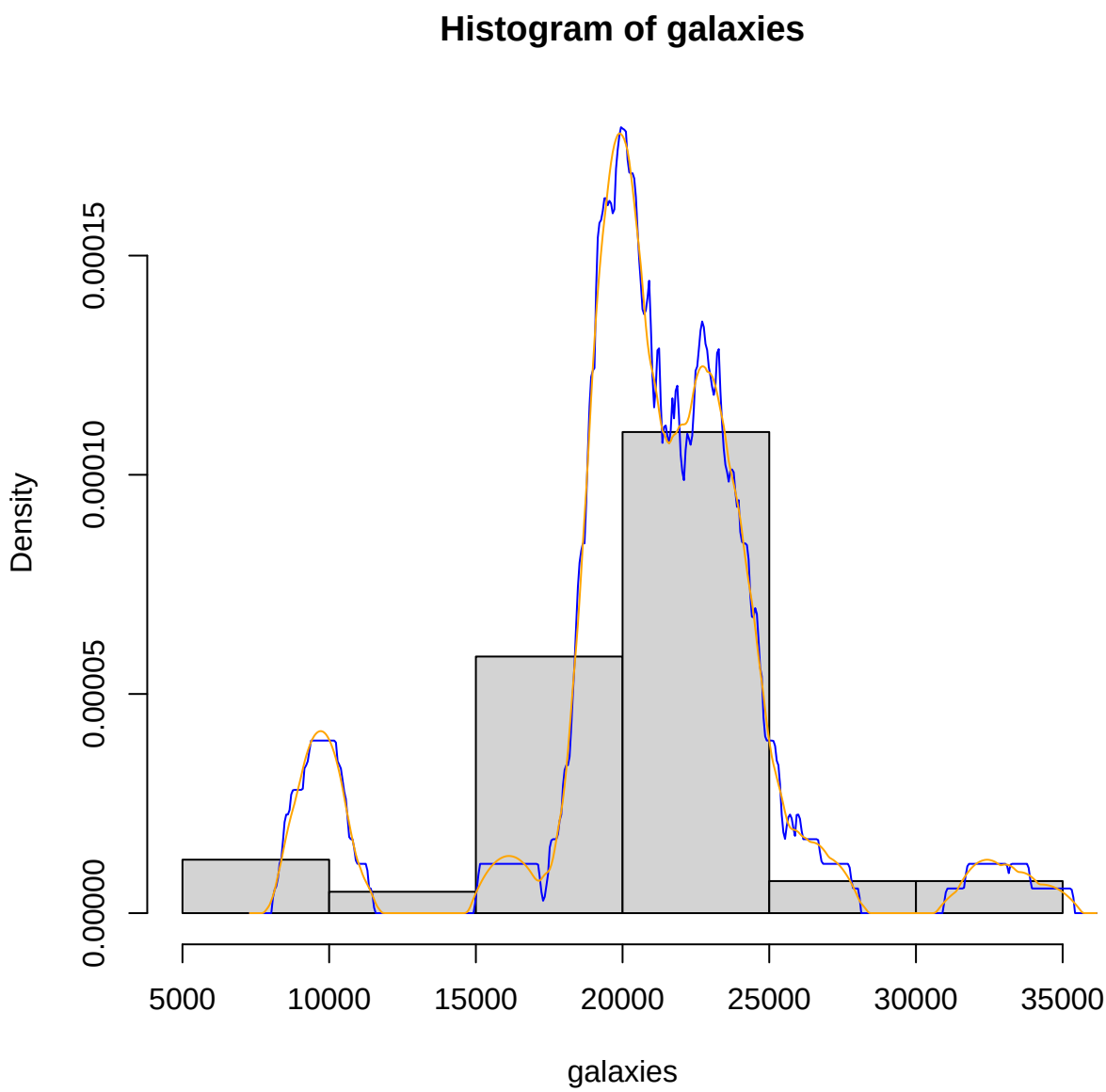



```
plot(s,dmixing(s,alpha,l0,l1,p0,p1),col='red',type='l')
sm.density(x,method='normal',add=T,poly.index=0,nbins=0)
sm.density(x,method='sj',col='green',add=T,poly.index=0,nbins=0)
sm.density(x,method='cv',col='orange',add=T,poly.index=0,nbins=0)
legend(x="topleft",leg=c('nrd','SJ','ucv'), col=c('black',"green","orange"),lty=1)
```

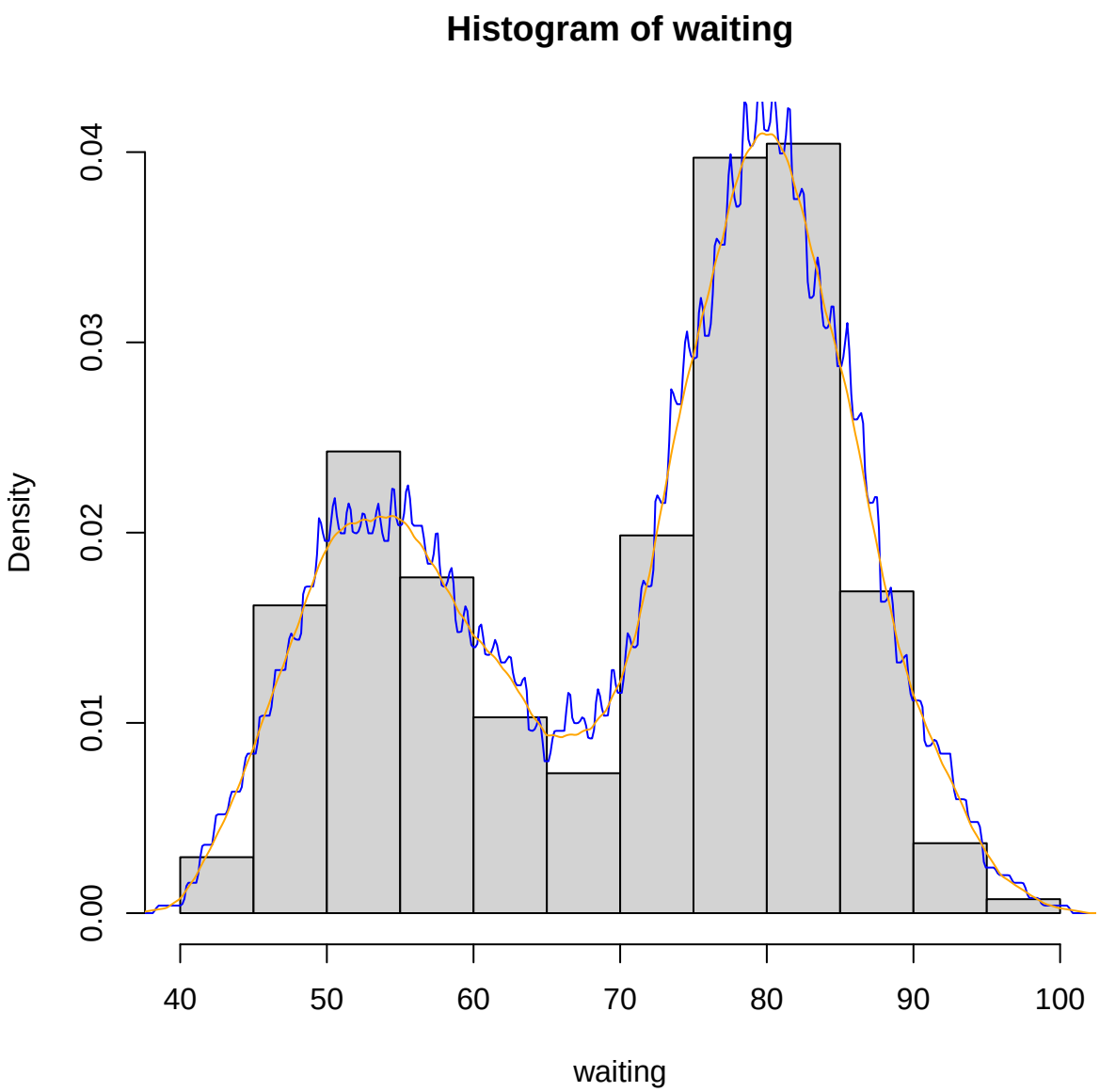


1.2.2. Come back to real world data

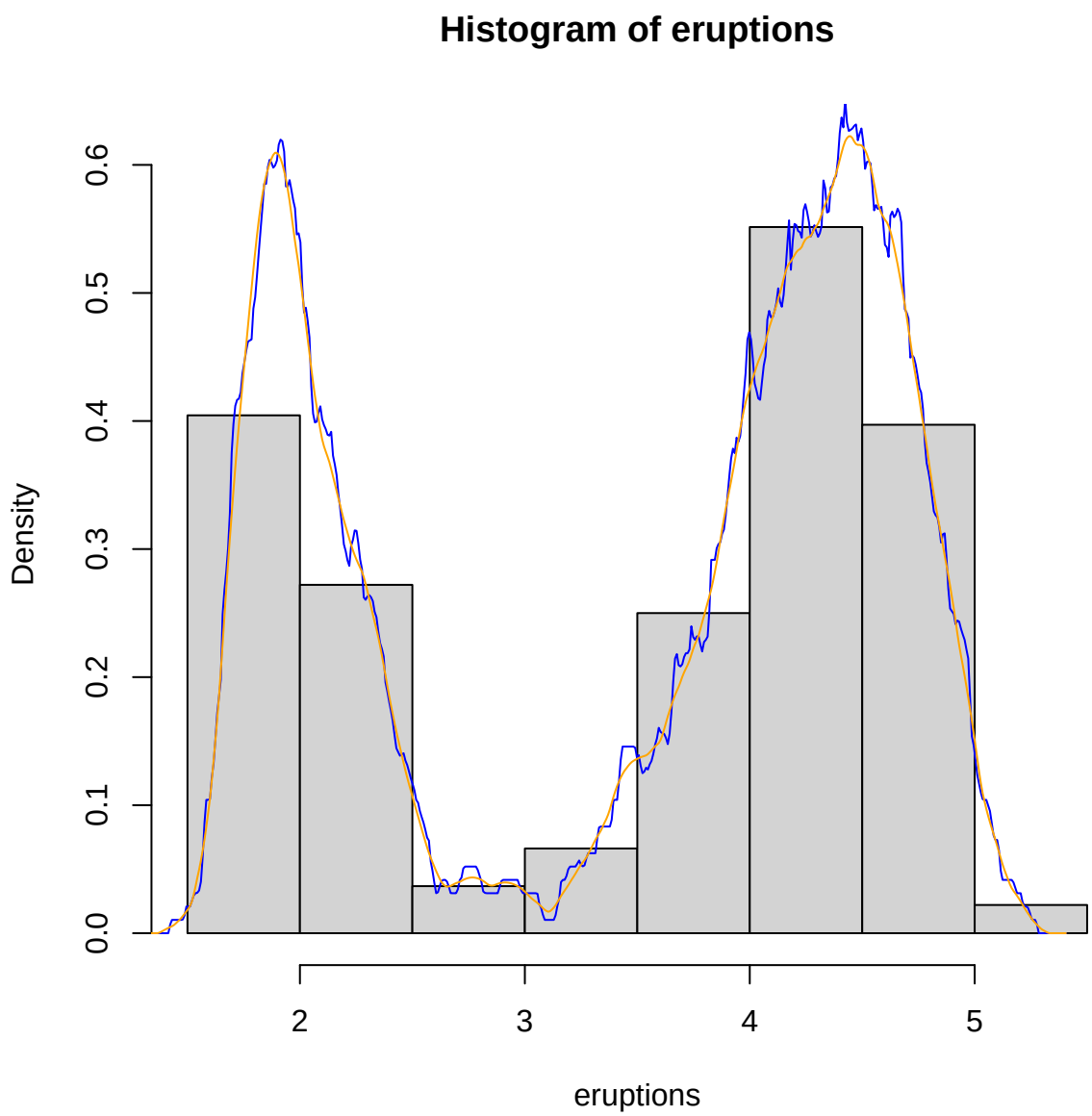
```
# Galaxies
hist(galaxies,freq=F,ylim=range(density(galaxies,kernel='e',bw='ucv')$y))
lines(density(galaxies,kernel='rectangular',bw='ucv'),col='blue')
lines(density(galaxies,kernel='e',bw='ucv'),col='orange')
```




```r
# Faithful
hist(waiting,freq=F,ylim=range(density(waiting,kernel='e',bw='ucv')$y))
lines(density(waiting,kernel='rectangular',bw='ucv'),col='blue')
lines(density(waiting,kernel='e',bw='ucv'),col='orange')
```




```
hist(eruptions,freq=F,ylim=range(density(eruptions,kernel='e',bw='ucv')$y))
lines(density(eruptions,kernel='rectangular',bw='ucv'),col='blue')
lines(density(eruptions,kernel='e',bw='ucv'),col='orange')
```



1.4. Applications

1.4.1. Mode estimation

Mode:

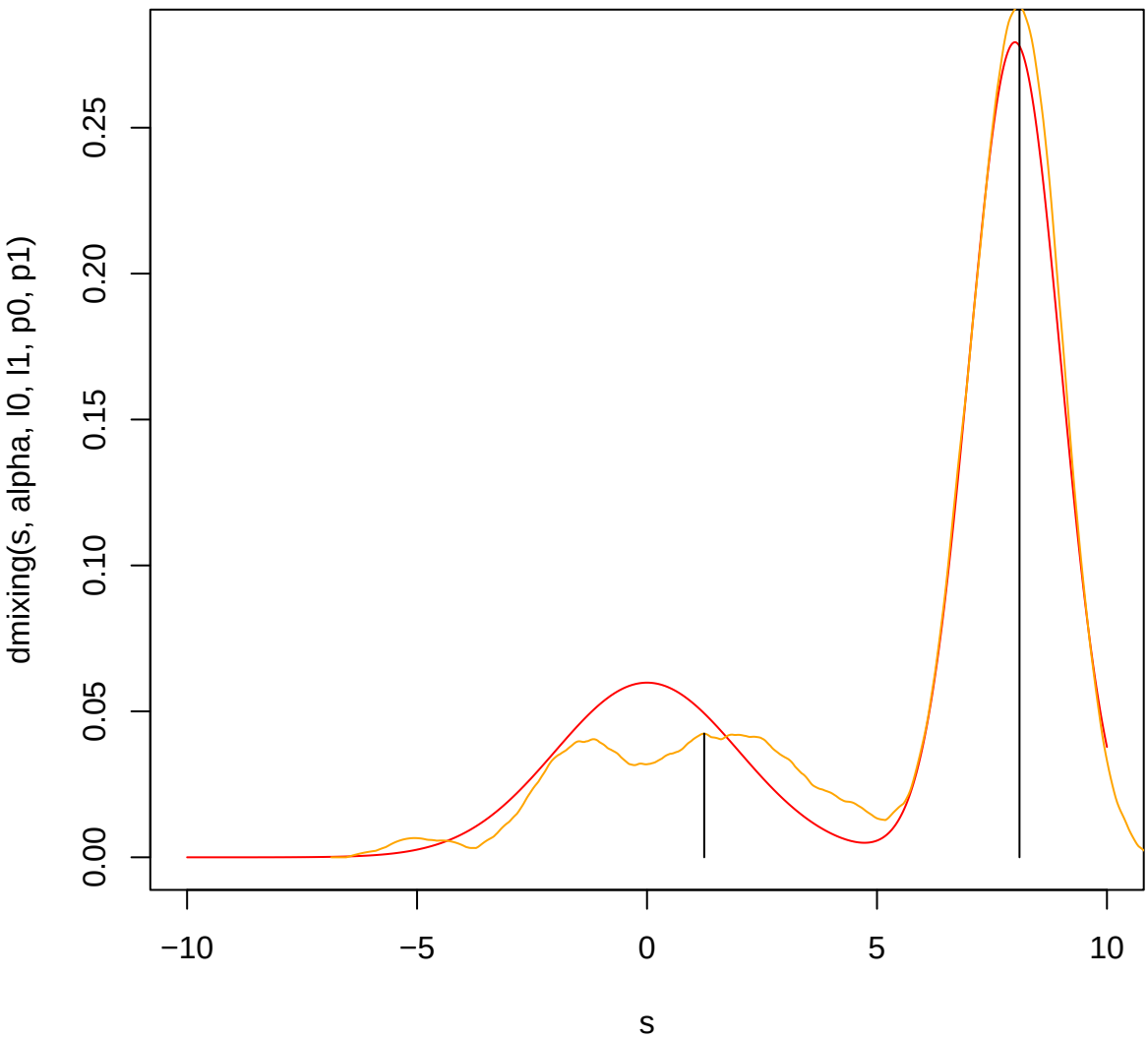
$$\theta_0 \in \arg\max_x f_X(x)$$

Estimateur à noyau du Mode:

$$\hat{\theta} \in \arg\max_x \hat{f}(x)$$

```
density.mode=function(x,a,b,M,bw='ucv',kernel='e',plot=T)
{
  disc=seq(a,b,length.out=M)
  dens=density(x,from=a,to=b,n=M,bw=bw,kernel=kernel)$y
  mod=disc[(order(dens))[M]]
```

```
  max=max(dens)
  if (plot) {plot(disc,dens,type='l')}
  return(list(mode=mod,max=max))
}

sm.mode=function(x,a,b,M,method='cv',plot=T)
{
  disc=seq(a,b,length.out=M)
  display="line"
  if (!plot) {display="none"}
  dens=sm.density(x,eval.points=disc,method=method,nbins=0,poly.index=0,display=display)$estimate
  mod=disc[(order(dens))[M]]
  max=max(dens)
  return(list(mode=mod,max=max))
}

plot(s,dmixing(s,alpha,l0,l1,p0,p1),col='red',type='l')
lines(density(x,bw='ucv',kernel='e'),col='orange')
re=density.mode(x,-10,10,1000,bw='ucv',kernel='e',F)
segments(re$mod,0,re$mod,re$max)
re1=density.mode(x,-10,3,1000,bw='ucv',kernel='e',F)
segments(re1$mod,0,re1$mod,re1$max)
```



1.4.1. Clustering from density level set

A point is said to belong to a cluster when the estimated density is above a given threshold l_0. The clusters corresponds to the connex components of such points.

Level set corresponding to l_0

$$LS(l_0) := \{x, f_X(x) \geq l_0\}$$

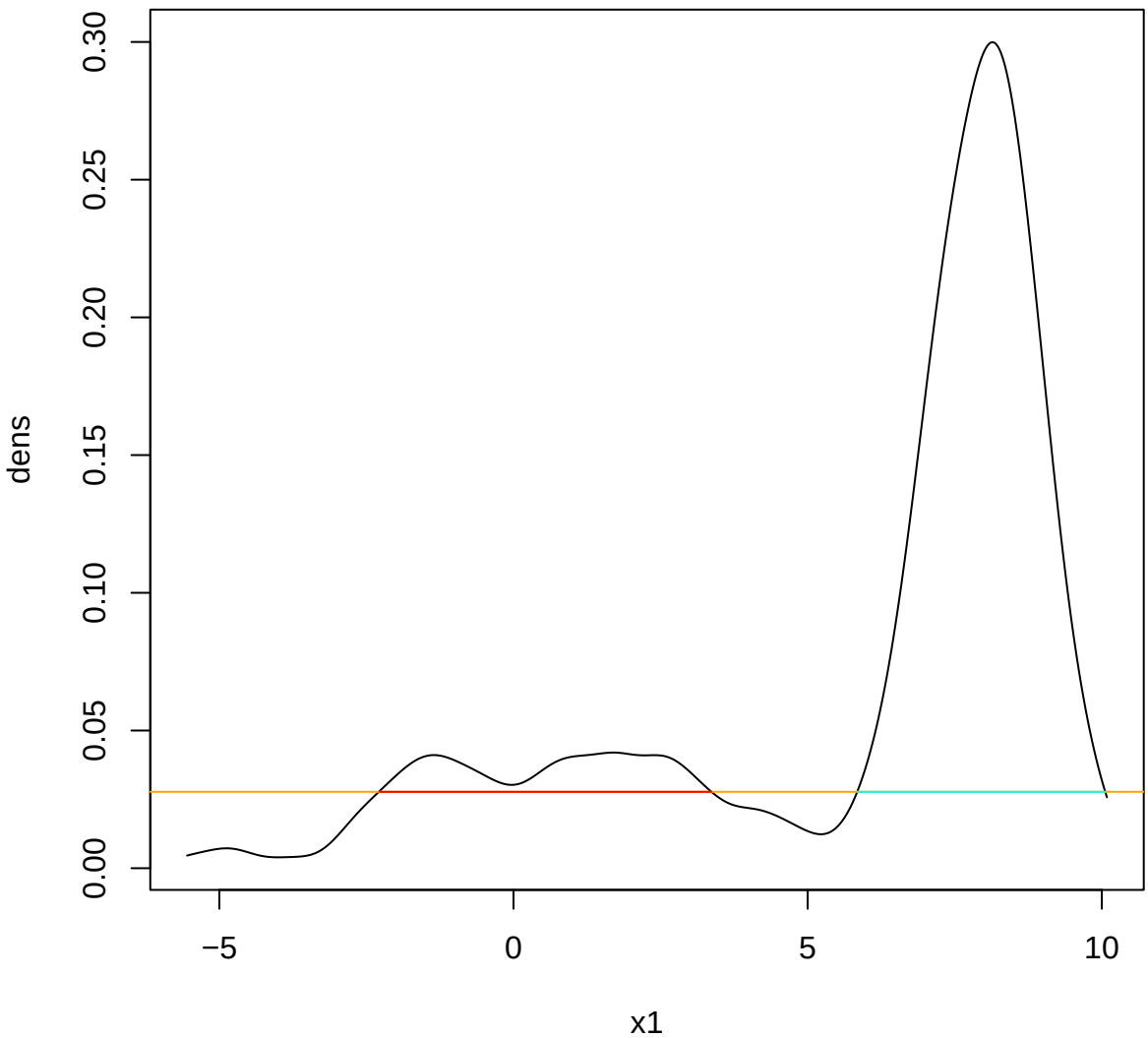
Estimated level set corresponding to l_0

$$ELS(l_0) := \{x, \hat{f}(x) \geq l_0\}$$

Clusters detected: the connex components of $ELS(l_0)$.

```
sm.clustering.level.sets=function(x,level=0.20,a=min(x),b=max(x),M=1000,method='sj',plot=T)
{
  disc=seq(a,b,length.out=M)
  n=length(x)
  o=order(x)
  x.and.disc=c(x,disc)
  #type=
```

```
  names(x.and.disc)=rep(c('data','disc'),c(n,M))
  x1=sort(x.and.disc)
  type1=names(x1)
  n1=length(x1)
  dens=sm.density(x,eval.points=x1,method=method,nbins=0,poly.index=0,display="none")$estimate
  adjusted.level=quantile(dens,level)
  is.over.level=dens>adjusted.level
  cluster=rep(0,M+n)
  k=1
  for (j in 1:(M+n))
    {
    if (j>1) {if (is.over.level[j]==0 & is.over.level[j-1]==1 ) {k=k+1}}
    if (is.over.level[j]==1) {cluster[j]=k}
    }
  if (plot) {plot(x1,dens,type='l')
             abline(adjusted.level,0,col='orange')}

  k=max(cluster)
  cluster.bounds=matrix(NA,2,k)
  palette=rainbow(k)
  for (j in 1:k)
    {
      cluster.bounds[,j]=range(x1[cluster==j])
      if (plot) { segments(cluster.bounds[1,j],adjusted.level,cluster.bounds[2,j],adjusted.level,col=pal
    }
  cluster.on.data=(cluster[type1=='data'])
  colnames(cluster.bounds)=paste('Cluster_',1:k)
  rownames(cluster.bounds)=c('min','max')
  #cluster.on.data[cluster.on.data==0]=NA
  return(list(cluster=cluster.on.data,cluster.bounds=cluster.bounds))
}
sm.clustering.level.sets(x,0.3)
```

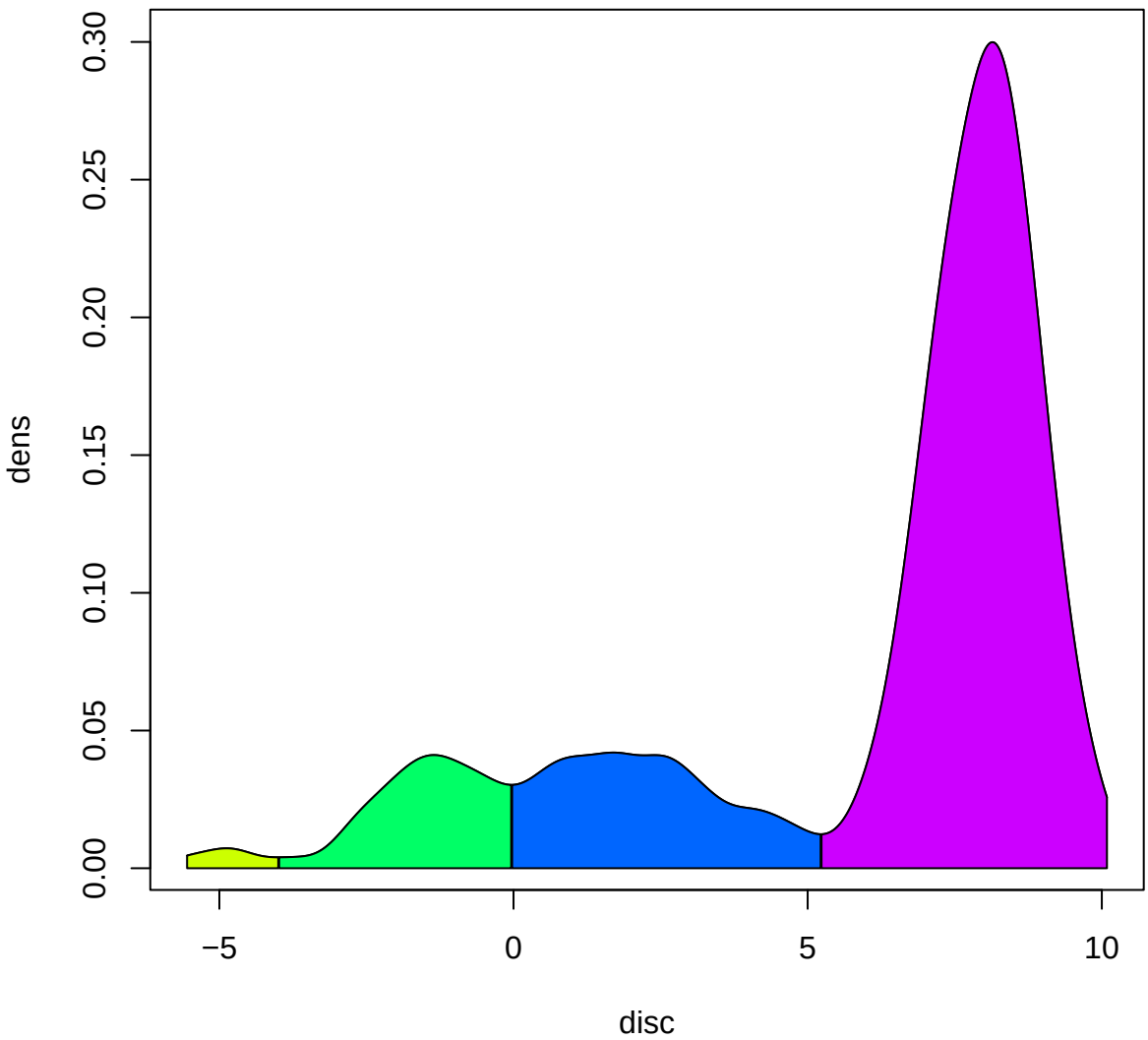



```
$cluster
  [1] 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[297] 2 2 2 0

$cluster.bounds
    Cluster_ 1 Cluster_ 2
min  -2.277448   5.845788
max   3.362373  10.056097
```

1.4.2. Clustering from density extrema

The clusters are located between the valley points of the estimated density. The parameter **tau** may be tuned to detect all the valley points or only the main ones.

```r
sm.clustering.extrema=function(x,tau=10,a=min(x),b=max(x),M=1000,method='sj',plot=T)
{
  disc=seq(a,b,length.out=M)
  n=length(x)
  dens=sm.density(x,eval.points= disc,method=method,nbins=0,poly.index=0,display="none")$estimate
  inc=rep(0,M)
  valleys=rep(0,M)
  k=1
  for (j in 1:M)
    {
    i1=max(1,j-tau)
    i2=min(M,j+tau)
    ratio=(dens[i2]-dens[i1])/(disc[i2]-disc[i1])
    if (ratio>0) {inc[j]=1}
    if (j>1) {if (inc[j]==0 & inc[j-1]==1 ) {k=k+1}}
    valleys[j]=k
    }
  if (plot) {plot(disc,dens,type='l')}

  k=max(valleys)
  thresholds=rep(NA,k)
  mins=rep(NA,k)
  for (j in 1:k)
    {
      candidates=disc[valleys==j]
      thresholds[j]=candidates[which.min(dens[valleys==j])]
      mins[j]=min(dens[valleys==j])
      #if (plot) {segments(thresholds[j],0,thresholds[j],mins[j],col='orange')}
  }
  n=length(x)
  cluster=rep(0,n)
  for (j in 1:k) {if (j==1) {cluster[x==thresholds[j]]=j}
cluster[x>thresholds[j]]=j}
  cluster.disc=rep(0,M)
  for (j in 1:k) {if (j==1) {cluster.disc[disc==thresholds[j]]=j}
                  cluster.disc[disc>thresholds[j]]=j}
  K=max(cluster.disc)
  palette=rainbow(K+1)
  if (plot) {for (j in 0:K) {if (sum(cluster.disc==j)!=0) {
  disc.j=disc[cluster.disc==j]
                  dens.j=dens[cluster.disc==j]
                  n.j=sum(cluster.disc==j)
                  polygon(c(disc.j[1],disc.j,disc.j[n.j]),c(0,dens.j,0),
        col = palette[j+1])}
  } }
  #cluster.on.data[cluster.on.data==0]=NA
  return(list(cluster=cluster,thresholds=thresholds))
}
sm.clustering.extrema(x,10)
```




```
$cluster
  [1] 4 4 4 2 4 4 4 2 3 4 4 4 3 4 4 4 2 3 4 4 4 4 3 4 4 4 4 4 4 3 2 4 4 4 3 4 4
 [38] 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 2 3 4 4 4 4 3 3 4 4 4 2 4 3 3 4 3 4 3 4
 [75] 4 4 4 3 4 1 4 4 4 4 4 3 4 2 4 4 2 4 4 3 4 4 3 3 3 4 4 4 4 4 4 4 3 4 2 4 4
[112] 4 4 2 4 4 4 4 2 3 4 4 2 3 4 4 4 4 4 4 4 4 4 4 2 4 4 4 3 2 3 4 4 2 4 3 4 4
[149] 4 1 4 2 4 4 3 4 4 4 4 4 4 4 4 4 3 4 4 4 2 3 4 2 3 4 4 4 4 2 4 3 3 4 4 2 4
[186] 3 4 2 4 4 4 4 4 2 4 2 3 4 4 4 4 4 4 3 4 2 4 4 4 4 2 4 4 4 3 4 4 3 4 4 4 3
[223] 4 4 4 4 4 4 4 4 4 4 4 4 3 4 2 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 4 2
[260] 4 3 2 4 4 3 4 4 3 2 2 4 4 4 3 4 4 4 4 3 1 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 3
[297] 4 4 4 4

$thresholds
[1] -5.54865512 -3.99913614 -0.03925428  5.21971924 10.08740010
```