Lab 8 - Dealing with Difficulties for Clustering

Loading libraries and data

```
#install.packages("mclust")
#install.packages("Rmixmod")
#install.packages("teigen")
library(mclust)
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(Rmixmod)
```

```
## Loading required package: Rcpp
```

```
## Rmixmod v. 2.1.5 / URI: www.mixmod.org
```

```
library(teigen)
library(prabclus)# NNclean() function
```

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

Simulating data with noise

$p(y_i) = \sum_{g=1}^{G} \tau_g f_g(y_i|\theta_g)$ with $\sum_{g=1}^{G} \tau_g = 1$

$f_g(y_i|\theta_g)$ is the density function of $N(\mu_g, \Sigma_g)$

μ_g

$\Sigma_g = \lambda_g D_g A_g D_g'$

- D_g is the matrix of eigenvectors of Σ_g
- $A_g = \text{diag}\{A_{1.g}, \cdots, A_{d,g}\}$ is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_g in descending order
- λ_g is the associated constant of proportionality

Characterization of the model:

- the first letter: "E" if λ_g is equal across clusters; "V" otherwise
- the second letter: "E" if A_g is equal across clusters (specifically, "I" if $A_g = I$); "V" otherwise
- the third letter: "E" if D_g is equal across clusters (specifically, "I" if $A_g = I$);; "V" otherwise

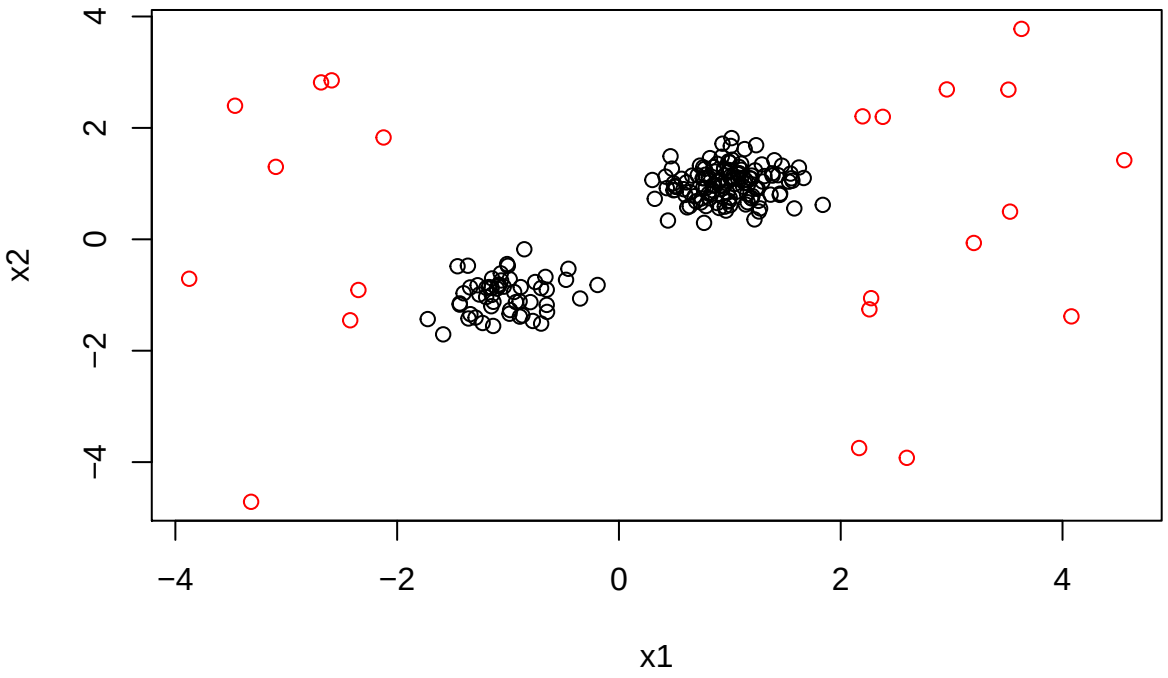
```
simdata1 <- sim(modelName="EII",n=180,
parameters=list(pro=c(.67,.33),mean=matrix(c(1,1,-1,-1),
ncol=2),variance=list(sigmasq=0.09)))
# pro: \tau_g
x1N <- rnorm (60,sd=2); x2N <- rnorm (60,sd=2)# noise
```

Remove noise points that are close to the cluster centers

```
dist1 <- rep (NA ,60); dist2 <- rep (NA ,60)
for (k in 1:60) dist1[k] <- sqrt ((x1N[k]-1)^2 + (x1N[k]-1)^2)# distance to the 1st cluster
for (k in 1:60) dist2[k] <- sqrt ((x1N[k]+1)^2 + (x1N[k]+1)^2)# distance to the 2nd cluster
close <- dist1<1.4 | dist2<1.4
noise <- cbind (x1N,x2N)
noise <- noise[!close ,]# remove noise points that are close to the cluster centers
Nnoise <- dim (noise)[[1]]
simdata <- rbind (simdata1[,c(2,3)],noise)
n<-dim(simdata)[[1]]
```

Plotting noisy data

```
plot (simdata , type="n")
points (simdata[1:180,])
points (simdata[181:n,], col="red")# noise points
```



Noise detection and clustering with Mclust

- add an additional component to the mixture model to represent the outliers, typically a uniform component.

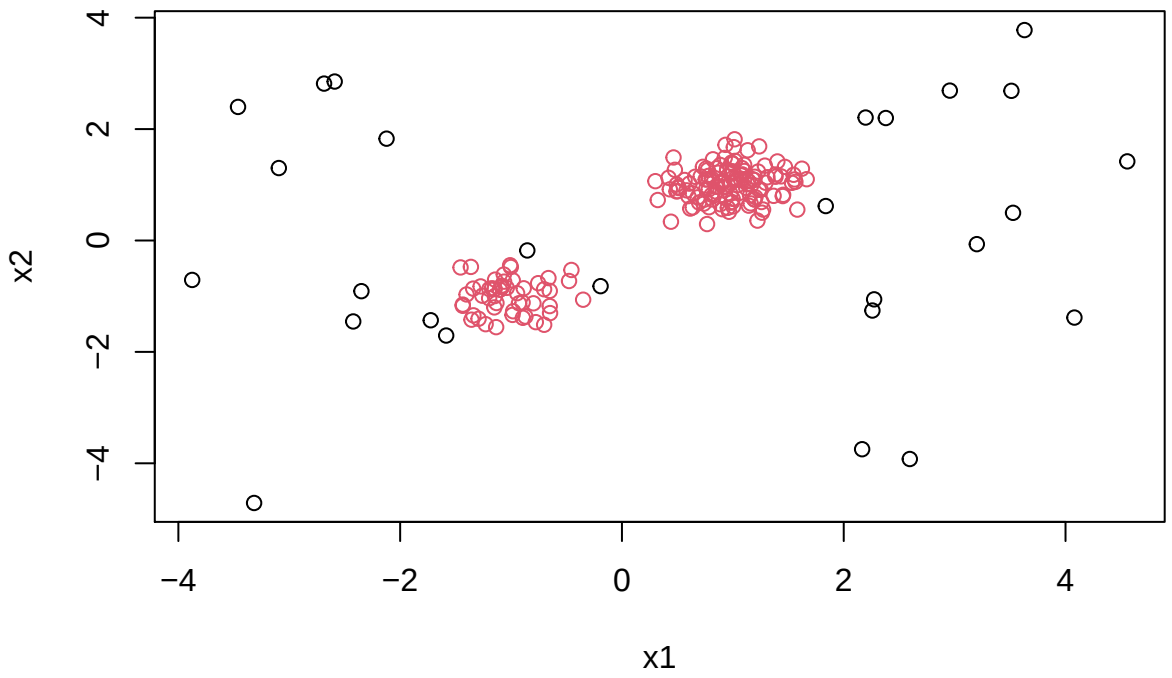
```
NNclean.out <- NNclean (scale(simdata), k=12)
MclustN <- Mclust(simdata , initialization = list(noise = (
NNclean.out$z==0)))# z: 1 means cluster, 0 means noise
```

Plotting results

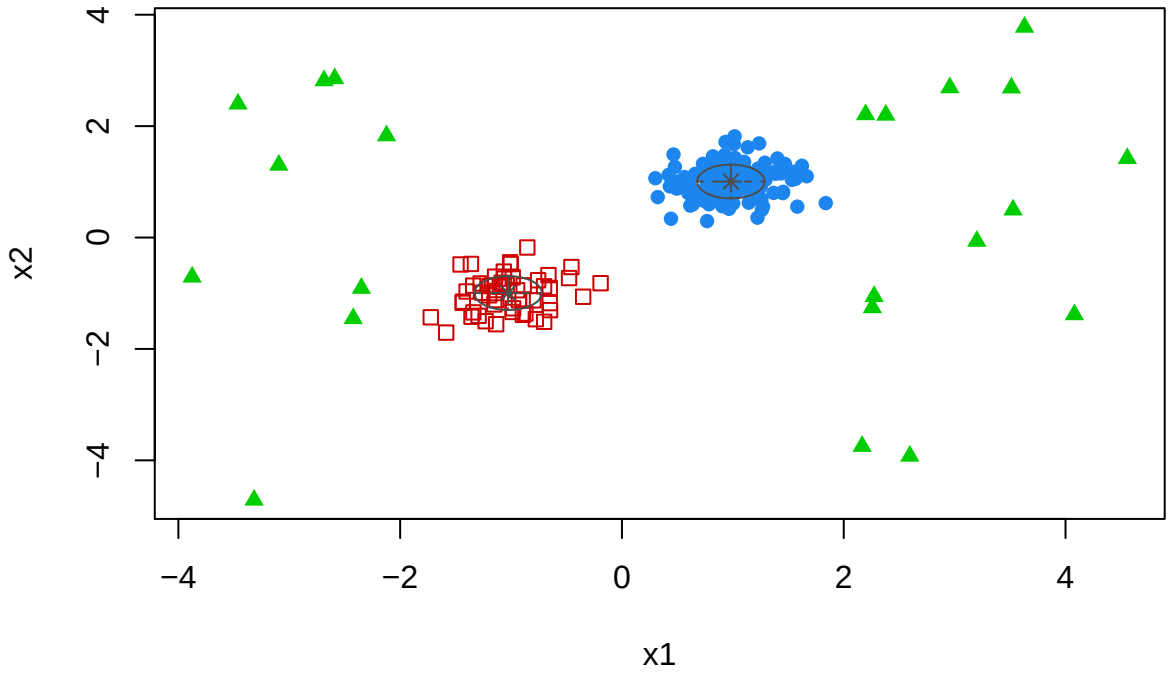
The size of circles corresponds to their uncertainty. The finally identified outliers are shown as gray dots.

$\text{Uncer}_{\text{i}} = 1 - \max_{g=1,\cdots,G} \hat{z}_{i,g}$

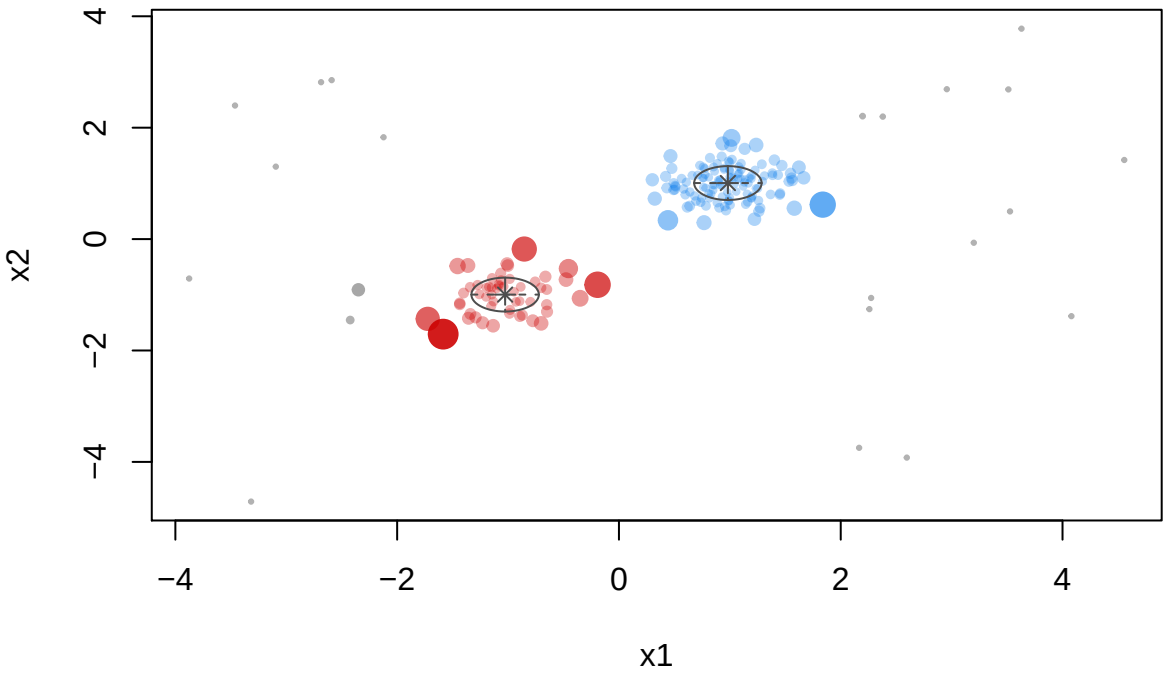
```
plot(simdata , col=1+NNclean.out$z)# Initialization using nearest neighbor cleaning (black)
```




```
mclust2Dplot(simdata , parameters=MclustN$parameters , z=MclustN$z,
what = "classification", PCH=20)# classification using model-based clustering
```




```
plot(MclustN , what="uncertainty")# uncertainty plot
```



Loading libraries

```
#install.packages("covRobust")
library (covRobust)
```

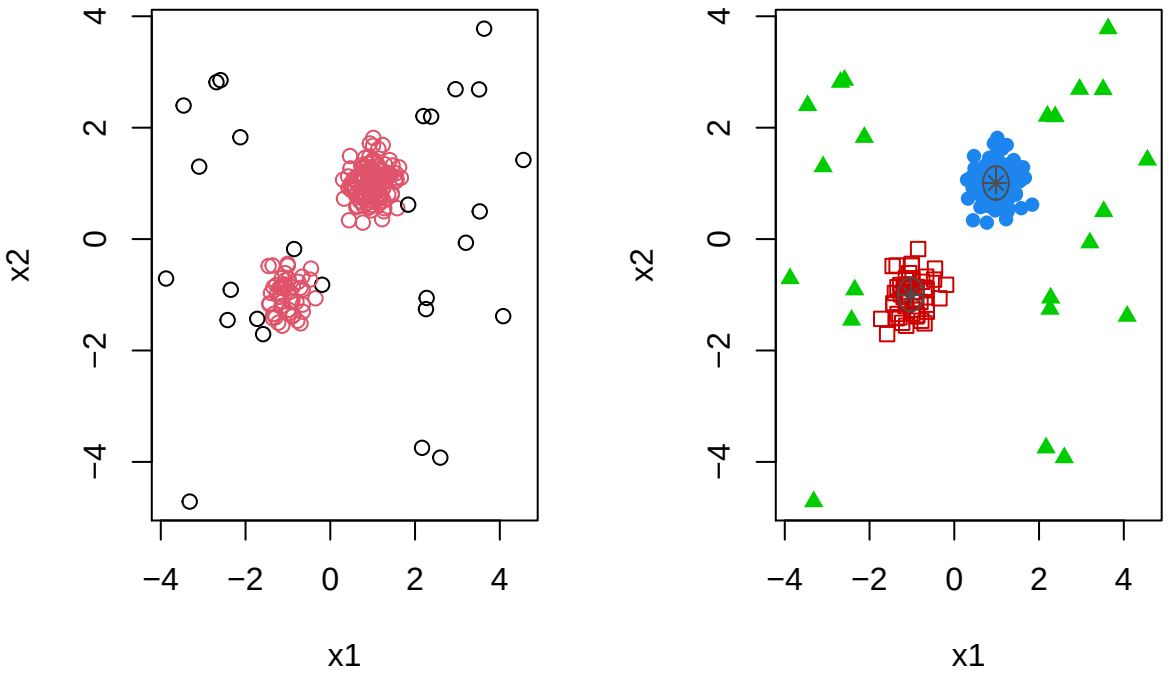
Robust covariance estimation and clustering (using the simulated data of the previous listing)

- the nearest neighbor cleaning method tends to identify more points as outliers than are actually present.
- the nearest neighbor variance estimation method (NNVE) adds simulated noise to the data before applying the nearest neighbor cleaning method, tending to reduce the number of false positives identified in the original data.

```
nnve.out <- cov.nnve (scale(simdata))
simdata.MclustN.NNVE <- Mclust (simdata,initialization=list(noise
=(nnve.out$ classification==0)))
```

Plotting results

```
par(mfrow=c(1,2))
plot (simdata , col=1+nnve.out$classification)# initialization of assignment of outliers via NNVE
mclust2Dplot(simdata , parameters=simdata.MclustN.NNVE$parameters ,
z=simdata.MclustN.NNVE$z,
what = "classification", PCH=20)# classification by model-based clustering initialized with NNVE.
```




```
par(mfrow=c(1,1))
```

Loading libraries

```
#install.packages("tclust")
```

Another approach to deal with noisy data:

- remove observations identified as outliers at the estimation stage.
- a robust version of the EM algorithm for mixture models
- the tclust model assumes that the data at hand contains $n\alpha$ "spurious" observations

```
#install.packages("MBCbook")
library(tclust)
library(MBCbook)
```

```
## Loading required package: mvtnorm
```

```
##
## Attaching package: 'mvtnorm'
```

```
## The following object is masked from 'package:mclust':
##
##     dmvnorm
```

Data simulation (class1 is for outliers)

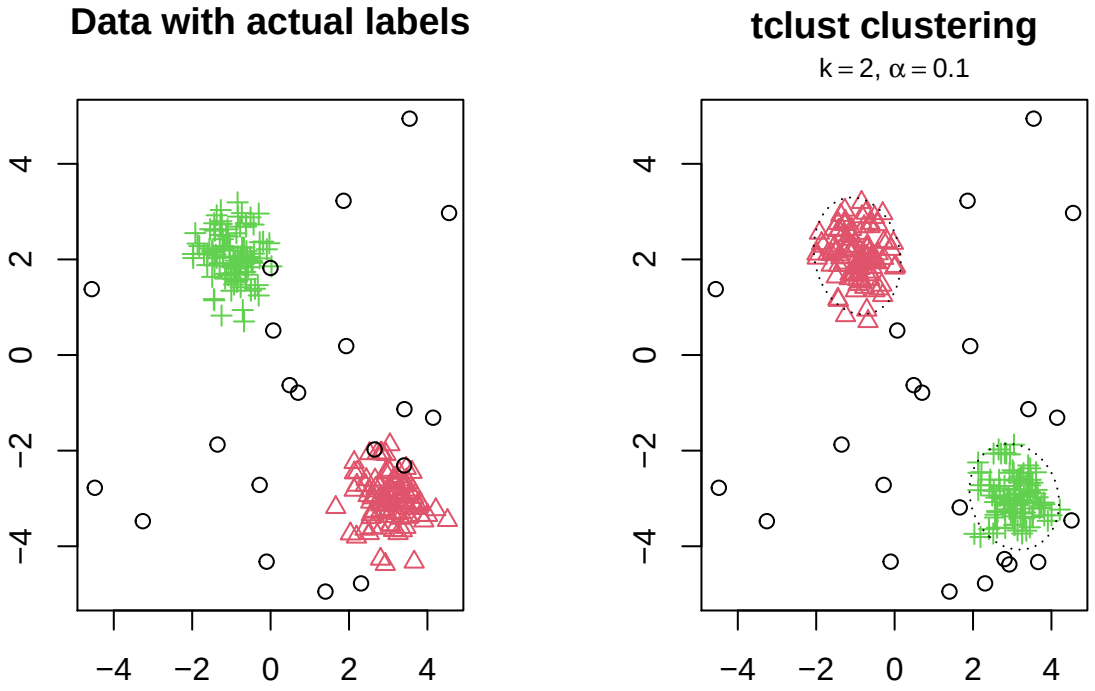
```
n=200;alpha=0.1
m1 = c(-1 ,2); m2 = 3*c(1,-1)
S1 = 0.25*diag(2); S2 = 0.25 * diag(2)
X=rbind(mvrnorm(n/2,m1,S1),mvrnorm(n/2,m2,S2),
matrix(runif(alpha*n*2,-5,5),ncol=2))
cls = rep(3:1,c(n/2,n/2,alpha*n))
```

tclust with the actual G and percentage of outliers

```
out <- tclust(X,k=2,alpha=alpha)
```

#Plottingresults

```
par(mfrow=c(1,2))
plot(X,col=cls,pch=cls,xlab='',ylab='',
main="Data with actual labels")# the simulated data with the actual group label
# (outliers are the black circles)
plot(out,xlab='',ylab='',main='tclust clustering')# the tclust solution,
```




```
# which gives results similar to the actual group labels
par(mfrow=c(1,1))
```

Clustering for various G and alpha (using data of the previous example)

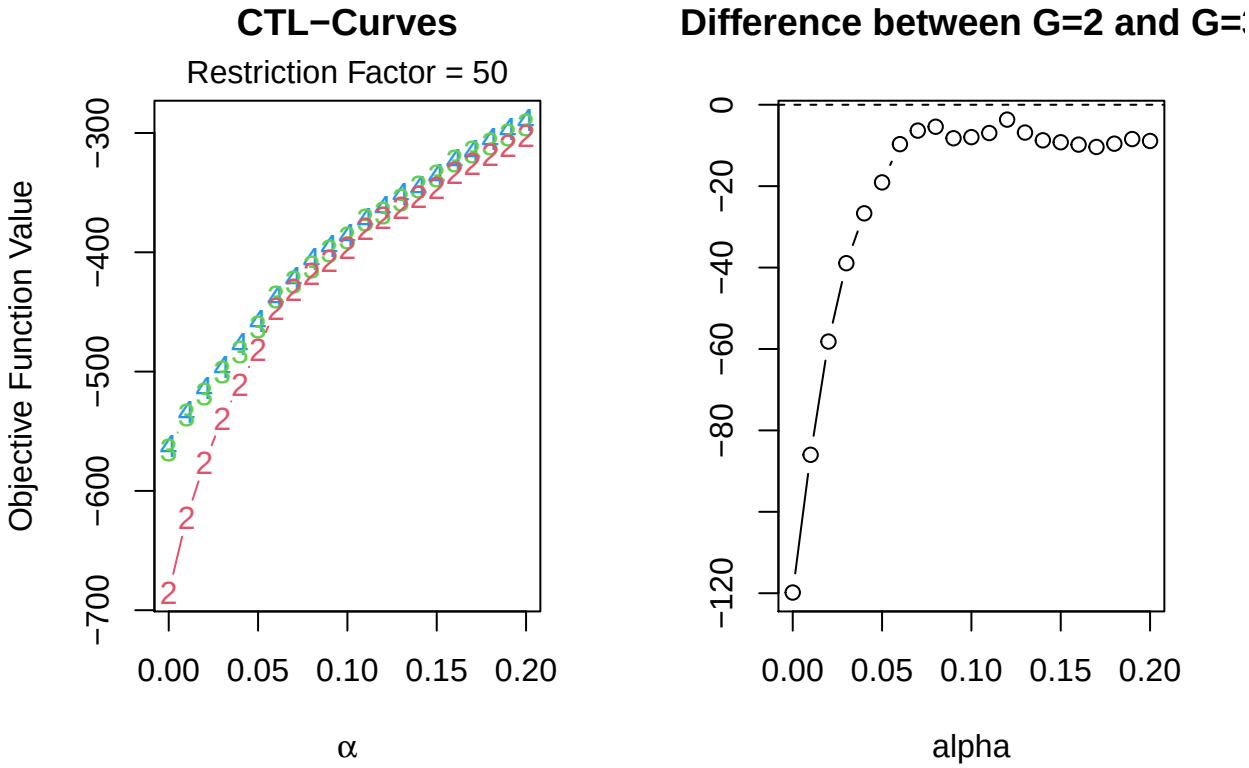
```
out <- ctlcurves(X, k=2:4, alpha=seq(0,0.2,len=21))
```

```
## Depending on arguments x, k and alpha, this function needs some time to compute.
## (Remove this message by setting "trace = 0")
```

Plotting results

```
par(mfrow=c(1,2))
plot(out)
```

```
plot(seq (0, 0.2, len = 21),out$obj[1,] - out$obj[2,], type='b',
xlab='alpha',ylab='',main='Difference between G=2 and G=3')
abline(h=0,lty=2)
```




```
par(mfrow=c(1,1))
```

Appropriate values for G and α are respectively 2 and 0.09.