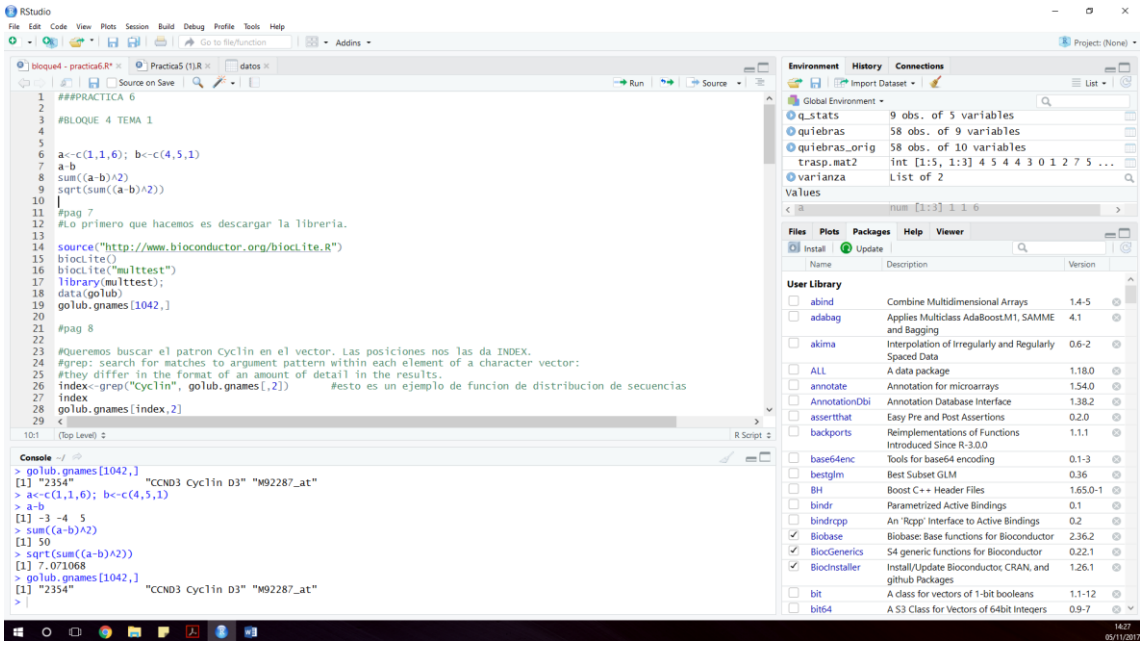
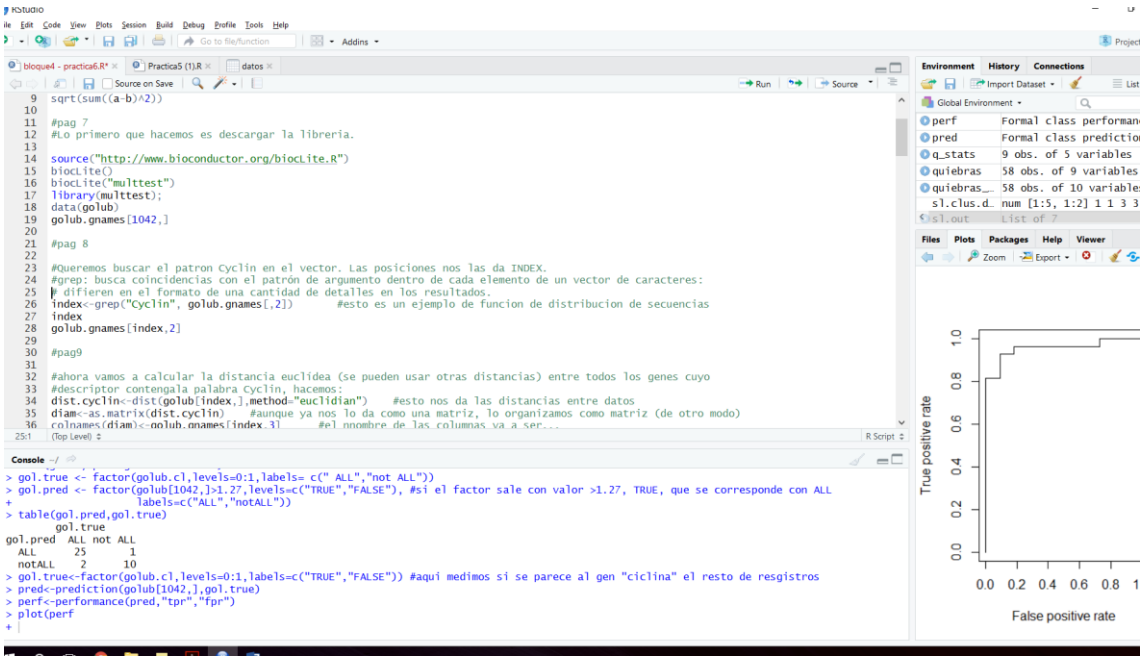
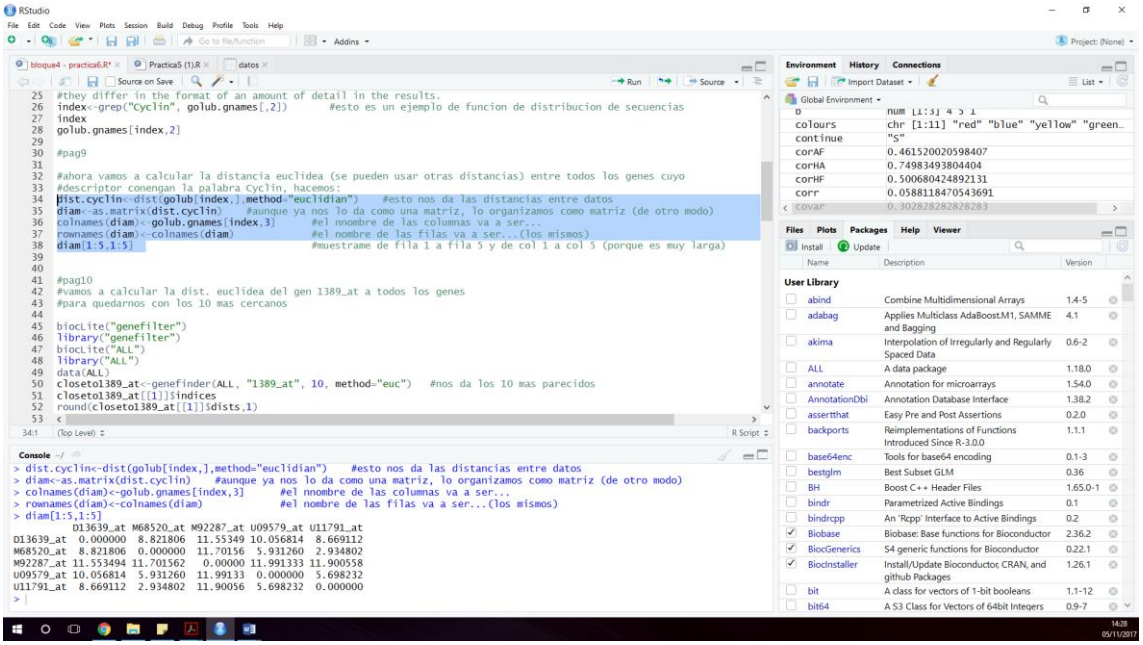
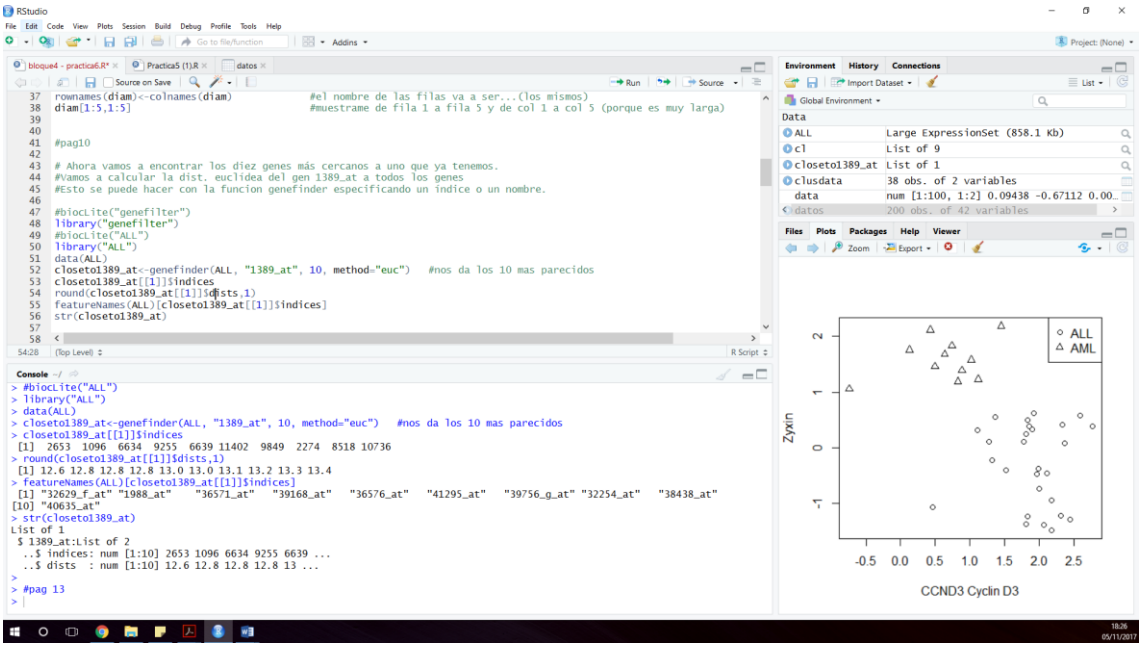
Programación con R. PRACTICA 6



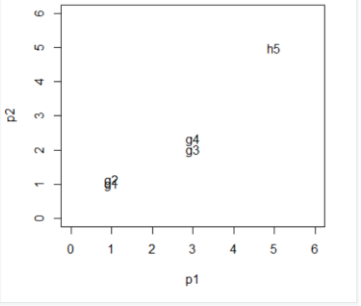



#they differ in the format of an amount of detail in the results.
index<-grep("Cyclin", golub.gnames[,2]) #esto es un ejemplo de funcion de distribucion de secuencias
index
golub.gnames[index,2]
#pag9
#ahora vamos a calcular la distancia euclidea (se pueden usar otras distancias) entre todos los genes cuyo
#descriptor conengan la palabra Cyclin, hacemos:
dist.cyclin<-dist(golub[index,],method="euclidian") #esto nos da las distancias entre datos
diam<-as.matrix(dist.cyclin) #aunque ya nos lo da como una matriz, lo organizamos como matriz (de otro modo)
colnames(diam)<-golub.gnames[index,3] #el nnombre de las columnas va a ser...
rownames(diam)<-colnames(diam) #el nombre de las filas va a ser...(los mismos)
diam[1:5,1:5] #muestrame de fila 1 a fila 5 y de col 1 a col 5 (porque es muy larga)
#pag10
#vamos a calcular la dist. euclidea del gen 1389_at a todos los genes
#para quedarnos con los 10 mas cercanos
biocLite("genefilter")
library("genefilter")
biocLite("ALL")
library("ALL")
data(ALL)
closeto1389_at<-genefinder(ALL, "1389_at", 10, method="euc") #nos da los 10 mas parecidos
closeto1389_at[[1]]$indices
round(closeto1389_at[[1]]$dists,1)
> dist.cyclin<-dist(golub[index,],method="euclidian") #esto nos da las distancias entre datos
> diam<-as.matrix(dist.cyclin) #aunque ya nos lo da como una matriz, lo organizamos como matriz (de otro modo)
> colnames(diam)<-golub.gnames[index,3] #el nnombre de las columnas va a ser...
> rownames(diam)<-colnames(diam) #el nombre de las filas va a ser...(los mismos)
> diam[1:5,1:5]
D13639_at M68520_at M92287_at U09579_at U11791_at
D13639_at 0.000000 8.821806 11.55349 10.056814 8.669112
M68520_at 8.821806 0.000000 11.70156 5.931260 2.934802
M92287_at 11.553494 11.701562 0.00000 11.991333 11.900558
U09579_at 10.056814 5.931260 11.99133 0.000000 5.698232
U11791_at 8.669112 2.934802 11.90056 5.698232 0.000000
>
Data
ALL Large ExpressionSet (858.1 Kb)
cl List of 9
closeto1389_at List of 1
clusdata 38 obs. of 2 variables
data num [1:100, 1:2] 0.09438 -0.67112 0.00...
colours chr [1:11] "red" "blue" "yellow" "green...
continue "s"
corAF 0.461520020598407
corHA 0.74983493804404
corHF 0.500680424892131
corr 0.0588118470543691
covar 0.302828282828283
User Library
abind Combine Multidimensional Arrays 1.4-5
adabag Applies Multiclass AdaBoost.M1, SAMME and Bagging 4.1
akima Interpolation of Irregularly and Regularly Spaced Data 0.6-2
ALL A data package 1.18.0
annotate Annotation for microarrays 1.54.0
AnnotationDbi Annotation Database Interface 1.38.2
assertthat Easy Pre and Post Assertions 0.2.0
backports Reimplementations of Functions Introduced Since R-3.0.0 1.1.1
base64enc Tools for base64 encoding 0.1-3
bestglm Best Subset GLM 0.36
BH Boost C++ Header Files 1.65.0-1
bindr Parametrized Active Bindings 0.1
bindrcpp An 'Rcpp' Interface to Active Bindings 0.2
Biobase Biobase: Base functions for Bioconductor 2.36.2
BiocGenerics S4 generic functions for Bioconductor 0.22.1
BiocInstaller Install/Update Bioconductor, CRAN, and github Packages 1.26.1
bit A class for vectors of 1-bit booleans 1.1-12
bit64 A S3 Class for Vectors of 64bit Integers 0.9-7


rownames(diam)<-colnames(diam) #el nombre de las filas va a ser...(los mismos)
diam[1:5,1:5] #muestrame de fila 1 a fila 5 y de col 1 a col 5 (porque es muy larga)
#pag10
Ahora vamos a encontrar los diez genes más cercanos a uno que ya tenemos.
#Vamos a calcular la dist. euclidea del gen 1389_at a todos los genes
#Esto se puede hacer con la funcion genefinder especificando un indice o un nombre.
#biocLite("genefilter")
library("genefilter")
#biocLite("ALL")
library("ALL")
data(ALL)
closeto1389_at<-genefinder(ALL, "1389_at", 10, method="euc") #nos da los 10 mas parecidos
closeto1389_at[[1]]$indices
round(closeto1389_at[[1]]$dists,1)
featureNames(ALL)[closeto1389_at[[1]]$indices]
str(closeto1389_at)
> #biocLite("ALL")
> library("ALL")
> data(ALL)
> closeto1389_at<-genefinder(ALL, "1389_at", 10, method="euc") #nos da los 10 mas parecidos
> closeto1389_at[[1]]$indices
[1] 2653 1096 6634 9255 6639 11402 9849 2274 8518 10736
> round(closeto1389_at[[1]]$dists,1)
[1] 12.6 12.8 12.8 12.8 13.0 13.0 13.1 13.2 13.3 13.4
> featureNames(ALL)[closeto1389_at[[1]]$indices]
[1] "32629_f_at" "1988_at" "36571_at" "39168_at" "36576_at" "41295_at" "39756_g_at" "32254_at" "38438_at"
[10] "40635_at"
> str(closeto1389_at)
List of 1
$ 1389_at:List of 2
..$ indices: num [1:10] 2653 1096 6634 9255 6639 ...
..$ dists : num [1:10] 12.6 12.8 12.8 12.8 13 ...
>
> #pag 13
ALL
AML
Zyxin
CCND3 Cyclin D3

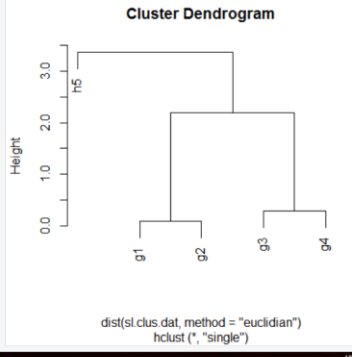
```
closeto1389_at[[1]]$indices
round(closeto1389_at[[1]]$dists,1)
featureNames(ALL)[closeto1389_at[[1]]$indices]
str(closeto1389_at)

#pag 13

#Mínima distancia:
#considerando tantos clústeres como nodos, calculamos la matriz de distancia entre cada par de clústeres. Luego
#unimos en un clúster los clústeres más cercanos y calculamos el centroide, que será la referencia para la
#siguiente matriz de distancias. Si el número de clústers es K parar, si no volvemos al paso 2.

names<-list(c("g1","g2","g3","g4","h5"),c("p1","p2")) #valores de genes de 2 personas p1 y p2
sl.clus.dat<-matrix(c(1,1,1,1.1,3,2,3,2.3,5,5),ncol=2,byrow=TRUE,dimnames=names)
sl.clus.dat
#el siguiente grafico Calcula las distancias entre los genes y realiza un análisis de enlace único
#(single linkage)
plot(sl.clus.dat,type="n",xlim=c(0,6), ylim=c(0,6))
text(sl.clus.dat,labels=row.names(sl.clus.dat))
print(dist(sl.clus.dat,method="euclidian"),digits=3)
```

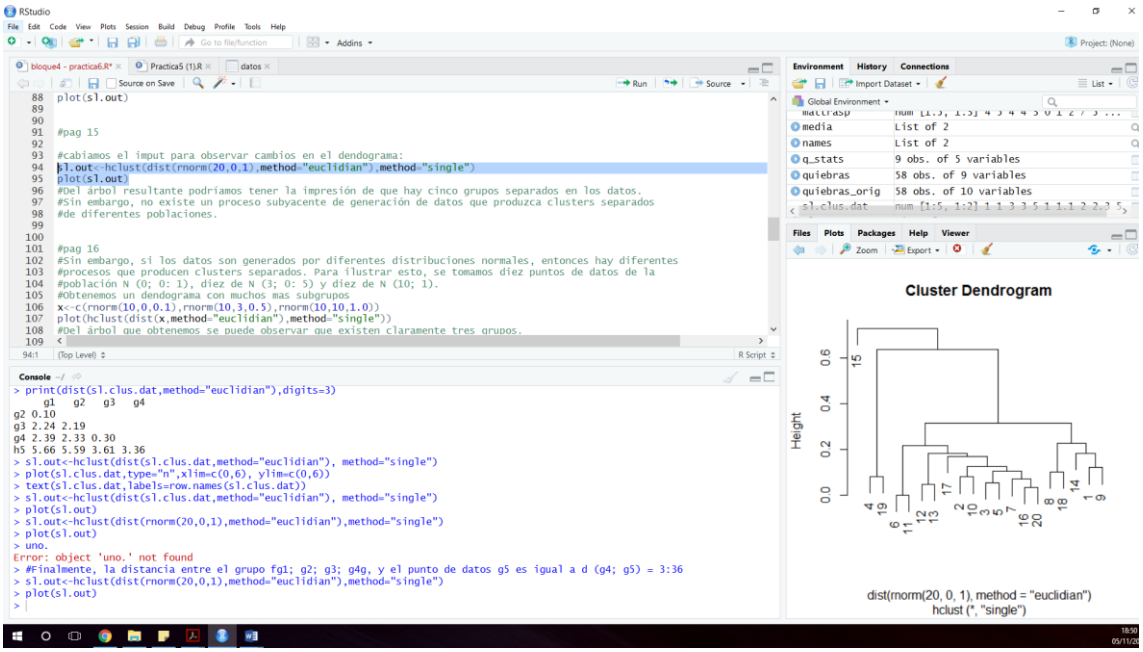
```
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> print(dist(sl.clus.dat,method="euclidian"),digits=3)
    g1   g2   g3   g4
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> print(dist(sl.clus.dat,method="euclidian"),digits=3)
    g1   g2   g3   g4
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.clus.dat,type="n",xlim=c(0,6), ylim=c(0,6))
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.out)
>
```

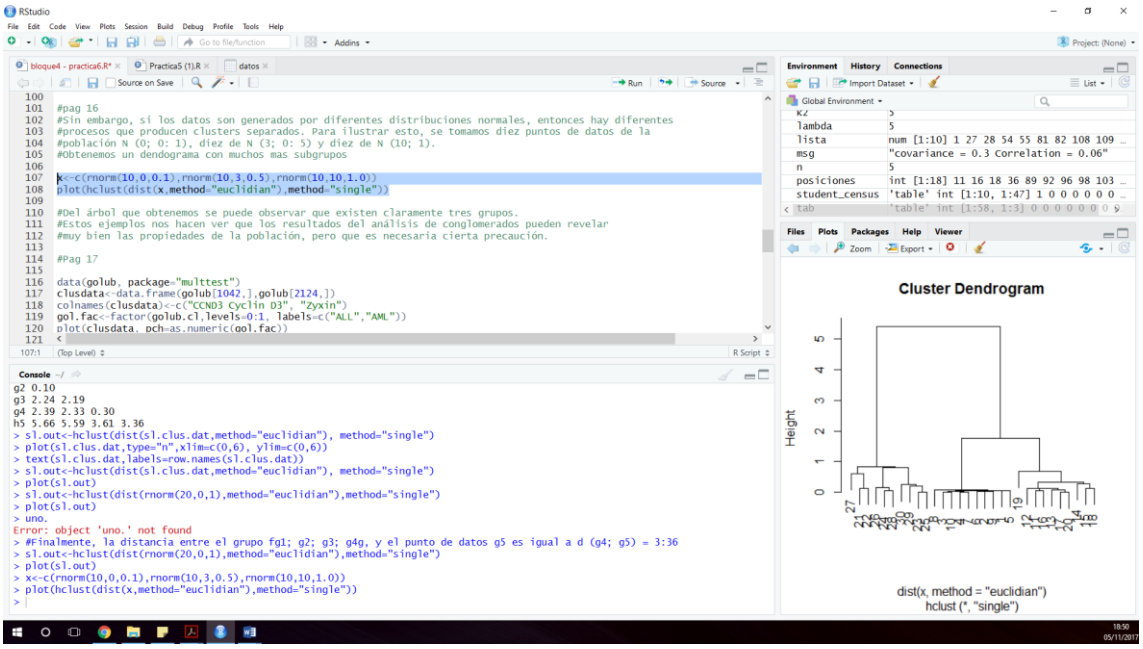



```
############pag 122 Krijnen
#Al comienzo, cada punto de datos se ve como un clúster separado. Entonces los dos puntos más cercanos (genes) de
#la matriz de distancia euclidiana son g1 y g2, teniendo d (g1; g2) = 0:10.
#Estos dos puntos de datos se fusionan en un grupo, digamos I = fg1; g2g.
#En el gráfico, esto se ilustra mediante la línea horizontal a la altura 0.10 en el árbol.
#Los otros tres puntos de datos g3; g4; g5 se ven como tres grupos diferentes. A continuación,
#la distancia mínima entre conglomerados se puede leer desde la matriz de distancia euclidiana.
#El más pequeño es d (g3; g4) = 0:30, así que el nuevo grupo J = fg3; g4g corresponde a la línea horizontal
#a la altura 0.30.
#Ahora hay tres clusters, I, J y K = fx5g. Desde la matriz de distancia euclidiana, se puede observar que
#la distancia entre el grupo I y J es de 2.19. Por lo tanto, el clúster I y J se fusionaron en uno.
#Finalmente, la distancia entre el grupo fg1; g2; g3; g4g, y el punto de datos g5 es igual a d (g4; g5) = 3:36

sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
plot(sl.out)


#pag 15

#cabiamos el imput para observar cambios en el dendograma:
```

```
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> print(dist(sl.clus.dat,method="euclidian"),digits=3)
    g1   g2   g3   g4
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> print(dist(sl.clus.dat,method="euclidian"),digits=3)
    g1   g2   g3   g4
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.clus.dat,type="n",xlim=c(0,6), ylim=c(0,6))
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.out)
>
```

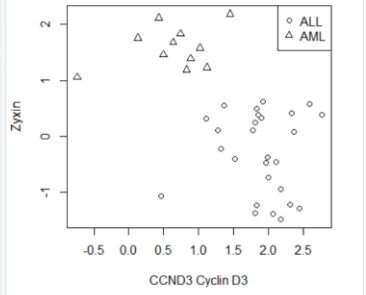


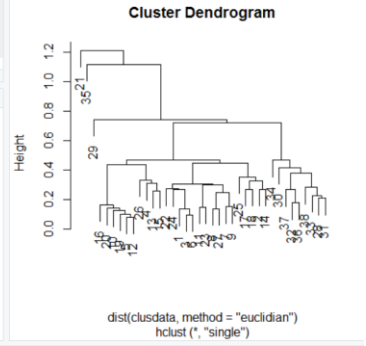

RStudio
plot(sl.out)
#pag 15
#cabiamos el imput para observar cambios en el dendograma:
sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
plot(sl.out)
#Del árbol resultante podríamos tener la impresión de que hay cinco grupos separados en los datos.
#Sin embargo, no existe un proceso subyacente de generación de datos que produzca clusters separados
#de diferentes poblaciones.
#pag 16
#Sin embargo, si los datos son generados por diferentes distribuciones normales, entonces hay diferentes
#procesos que producen clusters separados. Para ilustrar esto, se tomamos diez puntos de datos de la
#población N (0; 0: 1), diez de N (3; 0: 5) y diez de N (10; 1).
#Obtenemos un dendograma con muchos mas subgrupos
x<-c(rnorm(10,0,0.1),rnorm(10,3,0.5),rnorm(10,10,1.0))
plot(hclust(dist(x,method="euclidian"),method="single"))
#Del árbol que obtenemos se puede observar que existen claramente tres grupos.
> print(dist(sl.clus.dat,method="euclidian"),digits=3)
g1 g2 g3 g4
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.clus.dat,type="n",xlim=c(0,6), ylim=c(0,6))
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.out)
> sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
> plot(sl.out)
> uno.
Error: object 'uno.' not found
> #Finalmente, la distancia entre el grupo fg1; g2; g3; g4g, y el punto de datos g5 es igual a d (g4; g5) = 3:36
> sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
> plot(sl.out)
Cluster Dendrogram
Height
dist(rnorm(20, 0, 1), method = "euclidian")
hclust (*, "single")

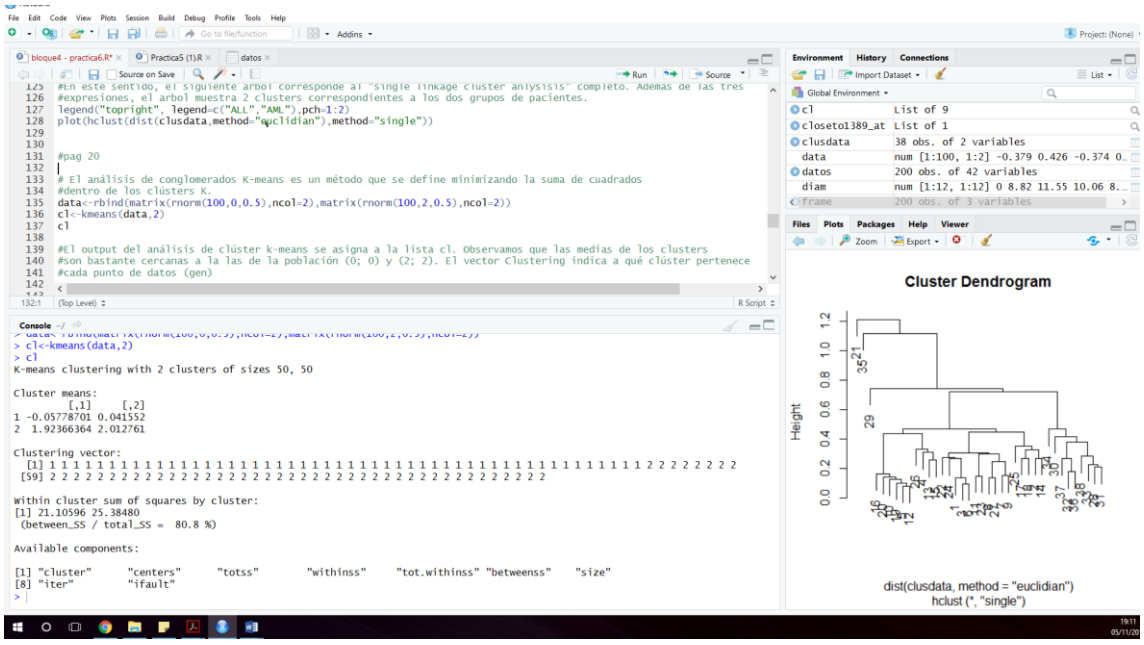

RStudio
#pag 16
#Sin embargo, si los datos son generados por diferentes distribuciones normales, entonces hay diferentes
#procesos que producen clusters separados. Para ilustrar esto, se tomamos diez puntos de datos de la
#población N (0; 0: 1), diez de N (3; 0: 5) y diez de N (10; 1).
#Obtenemos un dendograma con muchos mas subgrupos
x<-c(rnorm(10,0,0.1),rnorm(10,3,0.5),rnorm(10,10,1.0))
plot(hclust(dist(x,method="euclidian"),method="single"))
#Del árbol que obtenemos se puede observar que existen claramente tres grupos.
#Estos ejemplos nos hacen ver que los resultados del análisis de conglomerados pueden revelar
#muy bien las propiedades de la población, pero que es necesaria cierta precaución.
#Pag 17
data(golub, package="multtest")
clusdata<-data.frame(golub[1042,],golub[2124,])
colnames(clusdata)<-c("CCND3 Cyclin D3", "Zyxin")
gol.fac<-factor(golub.cl,levels=0:1, labels=c("ALL","AML"))
plot(clusdata, pch=as.numeric(gol.fac))
g2 0.10
g3 2.24 2.19
g4 2.39 2.33 0.30
h5 5.66 5.59 3.61 3.36
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.clus.dat,type="n",xlim=c(0,6), ylim=c(0,6))
> text(sl.clus.dat,labels=row.names(sl.clus.dat))
> sl.out<-hclust(dist(sl.clus.dat,method="euclidian"), method="single")
> plot(sl.out)
> sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
> plot(sl.out)
> uno.
Error: object 'uno.' not found
> #Finalmente, la distancia entre el grupo fg1; g2; g3; g4g, y el punto de datos g5 es igual a d (g4; g5) = 3:36
> sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
> plot(sl.out)
> x<-c(rnorm(10,0,0.1),rnorm(10,3,0.5),rnorm(10,10,1.0))
> plot(hclust(dist(x,method="euclidian"),method="single"))
Cluster Dendrogram
Height
dist(x, method = "euclidian")
hclust (*, "single")

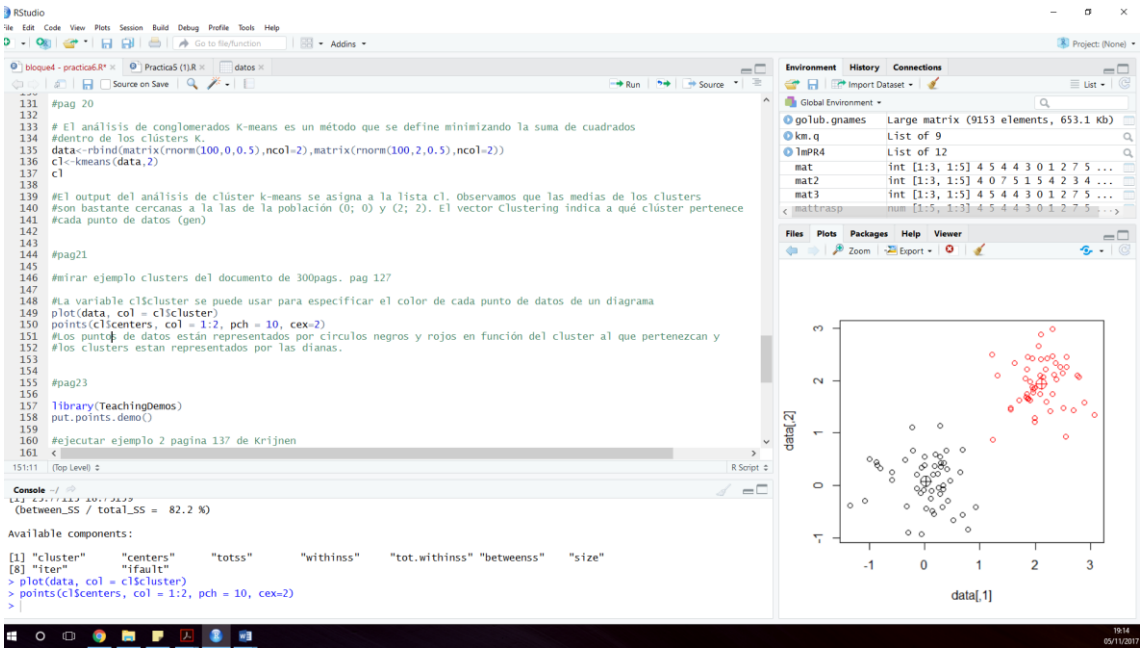
```
#muy bien las propiedades de la población, pero que es necesaria cierta precaución.

#Pag 17

#El siguiente gráfico muestra que los grupos de pacientes ALL y AML difieren con respecto a los valores de
#expresión genética "CCND3 Cyclin D3" and "Zyxin"

data(golub, package="multtest")
clusdata<-data.frame(golub[1042,],golub[2124,])
colnames(clusdata)<-c("CCND3 Cyclin D3", "Zyxin")
gol.fac<-factor(golub.cl,levels=0:1, labels=c("ALL","AML"))
plot(clusdata, pch=as.numeric(gol.fac))

#En este sentido, el siguiente árbol corresponde al "single linkage cluster anlysisis" completo. Ademas de las tres
#expresiones, el arbol muestra 2 clusters correspondientes a los dos grupos de pacientes.
legend("topright", legend=c("ALL","AML"),pch=1:2)
plot(hclust(dist(clusdata,method="euclidian"),method="single"))


#pag 20
```

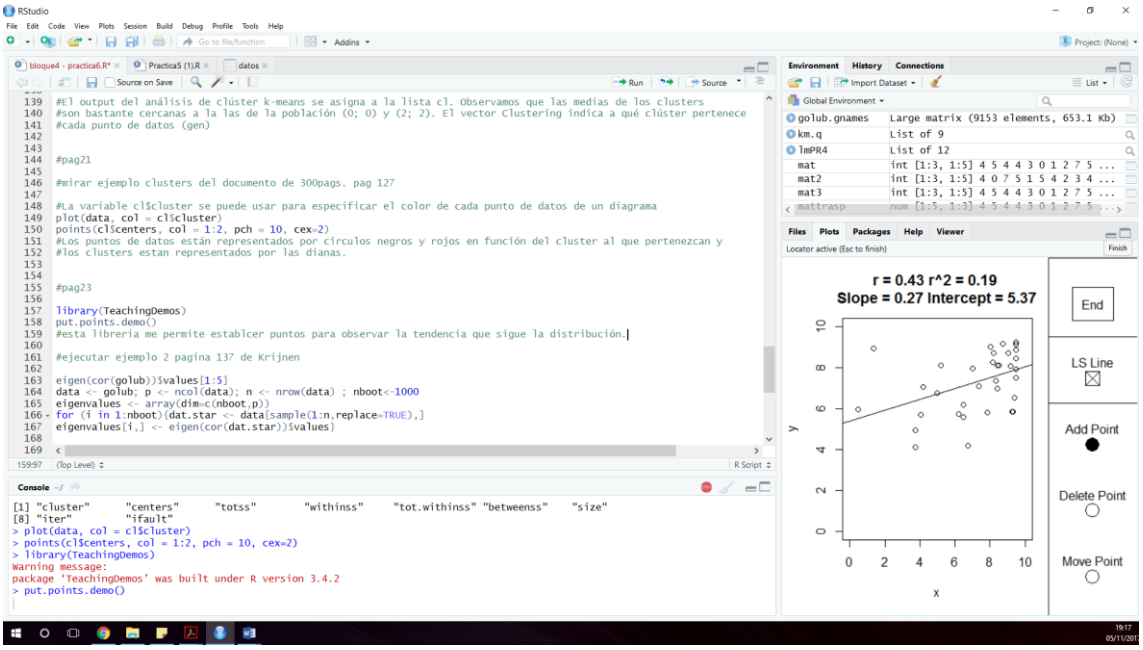
```
Error: object 'uno.' not found
> #Finalmente, la distancia entre el grupo fg1; g2; g3; g4g, y el punto de datos g5 es igual a d (g4; g5) = 3:36
> sl.out<-hclust(dist(rnorm(20,0,1),method="euclidian"),method="single")
> plot(sl.out)
> x<-c(rnorm(10,0,0.1),rnorm(10,3,0.5),rnorm(10,10,1.0))
> plot(hclust(dist(x,method="euclidian"),method="single"))
> data(golub, package="multtest")
> clusdata<-data.frame(golub[1042,],golub[2124,])
> colnames(clusdata)<-c("CCND3 Cyclin D3", "Zyxin")
> gol.fac<-factor(golub.cl,levels=0:1, labels=c("ALL","AML"))
> plot(clusdata, pch=as.numeric(gol.fac))
> data(golub, package="multtest")
> clusdata<-data.frame(golub[1042,],golub[2124,])
> colnames(clusdata)<-c("CCND3 Cyclin D3", "Zyxin")
> gol.fac<-factor(golub.cl,levels=0:1, labels=c("ALL","AML"))
> plot(clusdata, pch=as.numeric(gol.fac))
> legend("topright", legend=c("ALL","AML"),pch=1:2)
> plot(hclust(dist(clusdata,method="euclidian"),method="single"))
>
```

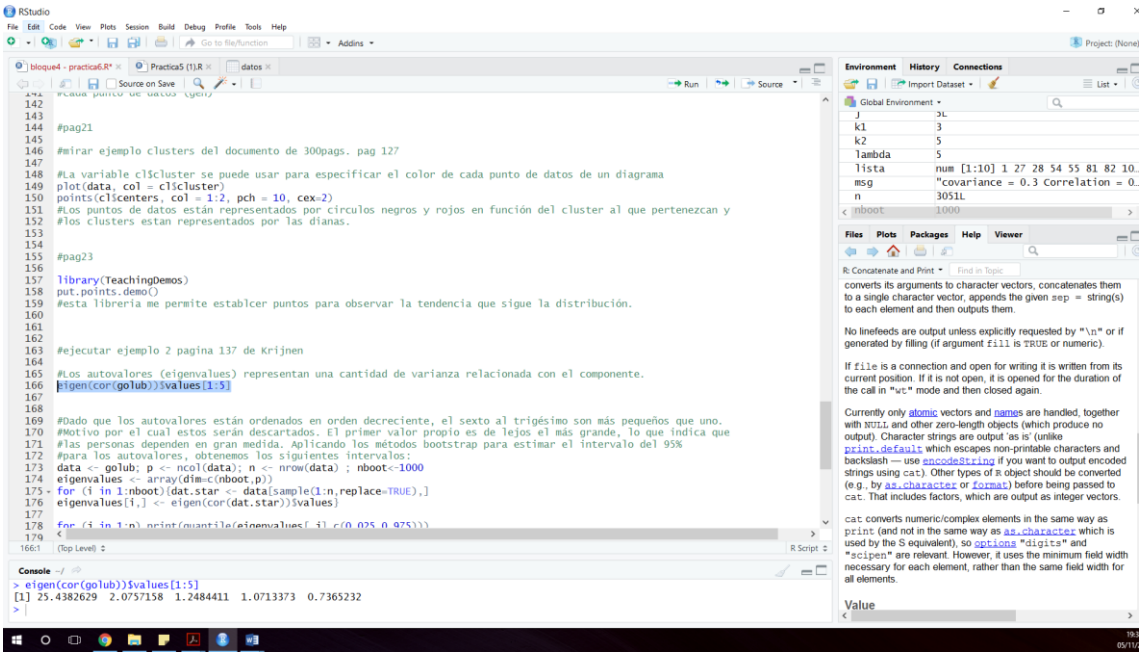


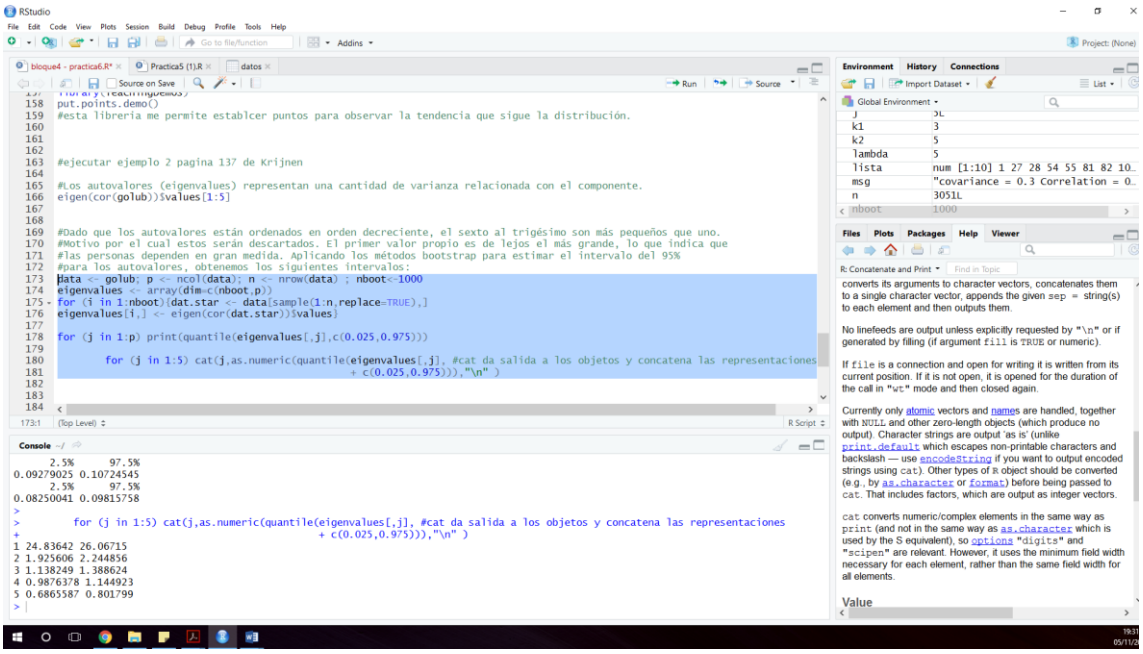


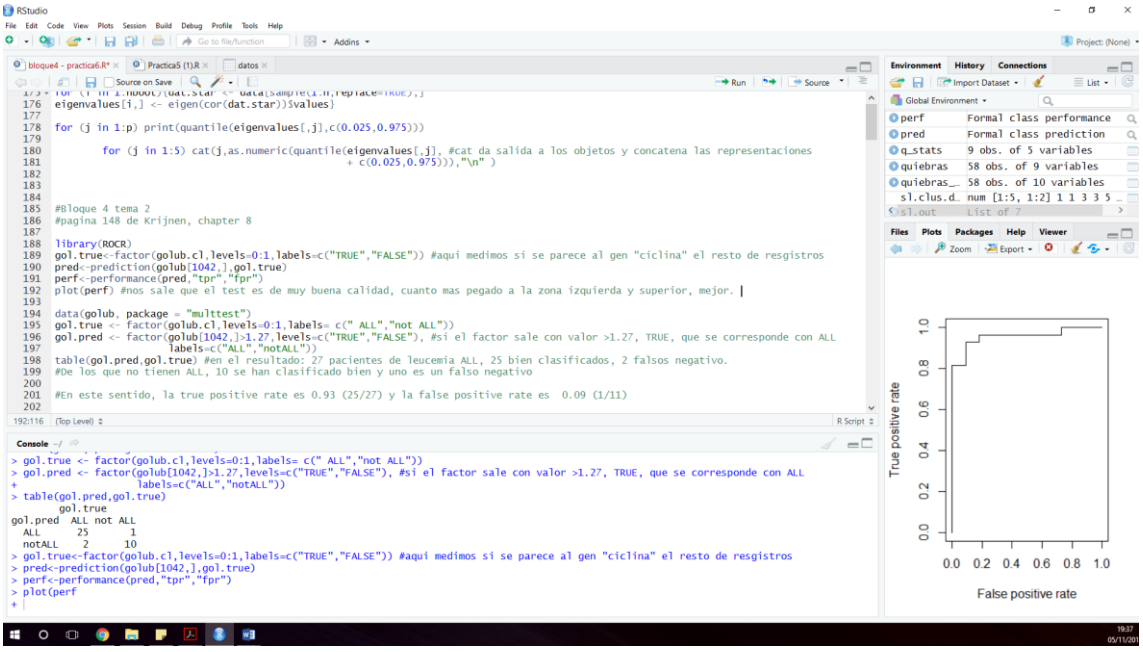

#En este sentido, el siguiente arbol corresponde al "single linkage cluster anlysisis" completo. Además de las tres
#expresiones, el arbol muestra 2 clusters correspondientes a los dos grupos de pacientes.
legend("topright", legend=c("ALL","AML"),pch=1:2)
plot(hclust(dist(clusdata,method="euclidian"),method="single"))
#pag 20
El análisis de conglomerados K-means es un método que se define minimizando la suma de cuadrados
#dentro de los clústers K.
data<-rbind(matrix(rnorm(100,0,0.5),ncol=2),matrix(rnorm(100,2,0.5),ncol=2))
cl<-kmeans(data,2)
cl
#El output del análisis de clúster k-means se asigna a la lista cl. Observamos que las medias de los clusters
#son bastante cercanas a la las de la población (0; 0) y (2; 2). El vector Clustering indica a qué clúster pertenece
#cada punto de datos (gen)
> cl<-kmeans(data,2)
> cl
K-means clustering with 2 clusters of sizes 50, 50
Cluster means:
[,1] [,2]
1 -0.05778701 0.041552
2 1.92366364 2.012761
Clustering vector:
[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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[59] 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Within cluster sum of squares by cluster:
[1] 21.10596 25.38480
(between_SS / total_SS = 80.8 %)
Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size"
[8] "iter" "ifault"
Cluster Dendrogram
Height
dist(clusdata, method = "euclidian")
hclust (*, "single")


#pag 20
El análisis de conglomerados K-means es un método que se define minimizando la suma de cuadrados
#dentro de los clústers K.
data<-rbind(matrix(rnorm(100,0,0.5),ncol=2),matrix(rnorm(100,2,0.5),ncol=2))
cl<-kmeans(data,2)
cl
#El output del análisis de clúster k-means se asigna a la lista cl. Observamos que las medias de los clusters
#son bastante cercanas a la las de la población (0; 0) y (2; 2). El vector Clustering indica a qué clúster pertenece
#cada punto de datos (gen)
#pag21
#mirar ejemplo clusters del documento de 300pags. pag 127
#La variable cl$cluster se puede usar para especificar el color de cada punto de datos de un diagrama
plot(data, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 10, cex=2)
#Los puntos de datos están representados por circulos negros y rojos en función del cluster al que pertenezcan y
#los clusters estan representados por las dianas.
#pag23
library(TeachingDemos)
put.points.demo()
#ejecutar ejemplo 2 pagina 137 de Krijnen
(between_SS / total_SS = 82.2 %)
Available components:
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size"
[8] "iter" "ifault"
> plot(data, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 10, cex=2)
data[,1]
data[,2]


RStudio
#El output del análisis de clúster k-means se asigna a la lista cl. Observamos que las medias de los clusters
#son bastante cercanas a la las de la población (0; 0) y (2; 2). El vector Clustering indica a qué clúster pertenece
#cada punto de datos (gen)
#pag21
#mirar ejemplo clusters del documento de 300pags. pag 127
#La variable cl$cluster se puede usar para especificar el color de cada punto de datos de un diagrama
plot(data, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 10, cex=2)
#Los puntos de datos están representados por circulos negros y rojos en función del cluster al que pertenezcan y
#los clusters estan representados por las dianas.
#pag23
library(TeachingDemos)
put.points.demo()
#esta libreria me permite establcer puntos para observar la tendencia que sigue la distribución.
#ejecutar ejemplo 2 pagina 137 de Krijnen
eigen(cor(golub))$values[1:5]
data <- golub; p <- ncol(data); n <- nrow(data) ; nboot<-1000
eigenvalues <- array(dim=c(nboot,p))
for (i in 1:nboot){dat.star <- data[sample(1:n,replace=TRUE),]
eigenvalues[i,] <- eigen(cor(dat.star))$values}
Console
[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size"
[8] "iter" "ifault"
> plot(data, col = cl$cluster)
> points(cl$centers, col = 1:2, pch = 10, cex=2)
> library(TeachingDemos)
Warning message:
package 'TeachingDemos' was built under R version 3.4.2
> put.points.demo()
Environment History Connections
golub.gnames Large matrix (9153 elements, 653.1 Kb)
km.q List of 9
lmPR4 List of 12
mat int [1:3, 1:5] 4 5 4 4 3 0 1 2 7 5 ...
mat2 int [1:3, 1:5] 4 0 7 5 1 5 4 2 3 4 ...
mat3 int [1:3, 1:5] 4 5 4 4 3 0 1 2 7 5 ...
mattrasp num [1:5, 1:3] 4 5 4 4 3 0 1 2 7 5 ...
Files Plots Packages Help Viewer
Locator active (Esc to finish)
Finish
r = 0.43 r^2 = 0.19
Slope = 0.27 Intercept = 5.37
End
LS Line
Add Point
Delete Point
Move Point
x
y


RStudio
#pag21
#mirar ejemplo clusters del documento de 300pags. pag 127
#La variable cl$cluster se puede usar para especificar el color de cada punto de datos de un diagrama
plot(data, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 10, cex=2)
#Los puntos de datos están representados por circulos negros y rojos en función del cluster al que pertenezcan y
#los clusters estan representados por las dianas.
#pag23
library(TeachingDemos)
put.points.demo()
#esta libreria me permite establcer puntos para observar la tendencia que sigue la distribución.
#ejecutar ejemplo 2 pagina 137 de Krijnen
#Los autovalores (eigenvalues) representan una cantidad de varianza relacionada con el componente.
eigen(cor(golub))$values[1:5]
#Dado que los autovalores están ordenados en orden decreciente, el sexto al trigésimo son más pequeños que uno.
#Motivo por el cual estos serán descartados. El primer valor propio es de lejos el más grande, lo que indica que
#las personas dependen en gran medida. Aplicando los métodos bootstrap para estimar el intervalo del 95%
#para los autovalores, obtenemos los siguientes intervalos:
data <- golub; p <- ncol(data); n <- nrow(data) ; nboot<-1000
eigenvalues <- array(dim=c(nboot,p))
for (i in 1:nboot){dat.star <- data[sample(1:n,replace=TRUE),]
eigenvalues[i,] <- eigen(cor(dat.star))$values}
Console
> eigen(cor(golub))$values[1:5]
[1] 25.4382629 2.0757158 1.2484411 1.0713373 0.7365232
Environment History Connections
k1 3
k2 5
lambda 5
lista num [1:10] 1 27 28 54 55 81 82 10..
msg "covariance = 0.3 Correlation = 0..
n 3051L
nboot 1000
Files Plots Packages Help Viewer
R: Concatenate and Print
converts its arguments to character vectors, concatenates them to a single character vector, appends the given sep = string(s) to each element and then outputs them.
No linefeeds are output unless explicitly requested by "\n" or if generated by filling (if argument fill is TRUE or numeric).
If file is a connection and open for writing it is written from its current position. If it is not open, it is opened for the duration of the call in "wt" mode and then closed again.
Currently only atomic vectors and names are handled, together with NULL and other zero-length objects (which produce no output). Character strings are output 'as is' (unlike print.default which escapes non-printable characters and backslash — use encodeString if you want to output encoded strings using cat). Other types of R object should be converted (e.g., by as.character or format) before being passed to cat. That includes factors, which are output as integer vectors.
cat converts numeric/complex elements in the same way as print (and not in the same way as as.character which is used by the S equivalent), so options "digits" and "scipen" are relevant. However, it uses the minimum field width necessary for each element, rather than the same field width for all elements.
Value


RStudio
put.points.demo()
#esta libreria me permite establcer puntos para observar la tendencia que sigue la distribución.
#ejecutar ejemplo 2 pagina 137 de Krijnen
#Los autovalores (eigenvalues) representan una cantidad de varianza relacionada con el componente.
eigen(cor(golub))$values[1:5]
#Dado que los autovalores están ordenados en orden decreciente, el sexto al trigésimo son más pequeños que uno.
#Motivo por el cual estos serán descartados. El primer valor propio es de lejos el más grande, lo que indica que
#las personas dependen en gran medida. Aplicando los métodos bootstrap para estimar el intervalo del 95%
#para los autovalores, obtenemos los siguientes intervalos:
data <- golub; p <- ncol(data); n <- nrow(data) ; nboot<-1000
eigenvalues <- array(dim=c(nboot,p))
for (i in 1:nboot){dat.star <- data[sample(1:n,replace=TRUE),]
eigenvalues[i,] <- eigen(cor(dat.star))$values}
for (j in 1:p) print(quantile(eigenvalues[,j],c(0.025,0.975)))
for (j in 1:5) cat(j,as.numeric(quantile(eigenvalues[,j], #cat da salida a los objetos y concatena las representaciones
+ c(0.025,0.975))),"\n")
2.5% 97.5%
0.09279025 0.10724545
2.5% 97.5%
0.08250041 0.09815758
1 24.83642 26.06715
2 1.925606 2.244856
3 1.138249 1.388624
4 0.9876378 1.144923
5 0.6865587 0.801799


RStudio
#Bloque 4 tema 2
#pagina 148 de Krijnen, chapter 8
library(ROCR)
gol.true<-factor(golub.cl,levels=0:1,labels=c("TRUE","FALSE")) #aqui medimos si se parece al gen "ciclina" el resto de resgistros
pred<-prediction(golub[1042,],gol.true)
perf<-performance(pred,"tpr","fpr")
plot(perf) #nos sale que el test es de muy buena calidad, cuanto mas pegado a la zona izquierda y superior, mejor.
data(golub, package = "multtest")
gol.true <- factor(golub.cl,levels=0:1,labels= c(" ALL","not ALL"))
gol.pred <- factor(golub[1042,]>1.27,levels=c("TRUE","FALSE"), #si el factor sale con valor >1.27, TRUE, que se corresponde con ALL
labels=c("ALL","notALL"))
table(gol.pred,gol.true) #en el resultado: 27 pacientes de leucemia ALL, 25 bien clasificados, 2 falsos negativo.
#De los que no tienen ALL, 10 se han clasificado bien y uno es un falso negativo
#En este sentido, la true positive rate es 0.93 (25/27) y la false positive rate es 0.09 (1/11)
> gol.true <- factor(golub.cl,levels=0:1,labels= c(" ALL","not ALL"))
> gol.pred <- factor(golub[1042,]>1.27,levels=c("TRUE","FALSE"), #si el factor sale con valor >1.27, TRUE, que se corresponde con ALL
+ labels=c("ALL","notALL"))
> table(gol.pred,gol.true)
gol.true
gol.pred ALL not ALL
ALL 25 1
notALL 2 10
> gol.true<-factor(golub.cl,levels=0:1,labels=c("TRUE","FALSE")) #aqui medimos si se parece al gen "ciclina" el resto de resgistros
> pred<-prediction(golub[1042,],gol.true)
> perf<-performance(pred,"tpr","fpr")
> plot(perf
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0