Pràctica 8: Clústers no jeràrquics.

•Mètodes basats en centroides: $K-means$

Dins de $K-means$ hi ha diversos algoritmes (Hartigan and Wong, Lloyd, McQueen,...). Referències a l'ajuda de la funció kmeans() de R. Per a cada algoritme, l'inici és aleatori -si no s'especifica un conjunt de centres com argument de la funció-, a l'ajuda de la funció ho explica. Ho aplicarem en un exemple amb l'opció de centres aleatoris a l'inici, però fixant la llavor per obtenir tots el mateix resultat. A la pràctica, convé aplicar la funció diverses vegades. Si sempre surt la mateixa classificació, perfecte. Si l'inici fa canviar suficientment la classificació, aleshores convé analitzar què passa i considerar una solució més 'robusta', per exemple, assignar cada cas al clúster al qual més vegades s'ha asignat. Això no és fàcil, perquè el número del clúster depén de cada realització (!). D'altra banda, pot ser que l'algoritme no convergeixi per algunes llavors o mai. En aquest cas, caldrà provar altres procediments.

Les dades `pottery` contenen la composició química de 48 peces de ceràmica romano-britànica determinada per espectrofotometria (Tubb et al., 1980). Aplicarem a `pottery` dos mètodes no-jereràquics diferents: el mètode de les k-mitjanes i el mètode EM (cas Gaussià). L'objectiu és obtenir una classificació de les restes ceràmiques en base a les similituds i diferències en la seva composició química, i veure quants subgrups de peces hi ha.

1. Instal·la i carrega la llibreria `HSAUR2` que conté les dades `pottery`. Guarda les dades en un dataframe `pots`. Per a les variables numèriques de l'arxiu `pots`, estandarditza les dades de la manera següent: divideix cada variable pel seu *rang*. Substitueix les variables de `pots` per les estandarditzades sense canviar el nom del fitxer.

```
require(HSAUR2)
# ?pottery
pots.num<-pottery[1:9]
rangs <- apply(pots.num,2,function(x){max(x)-min(x)})
# idem: rangs <- sapply(pots.num, function(x){max(x)-min(x)}) # prova-ho!
# idem: rangs <- sapply(pots.num, function(x){diff(range(x))})
pots.num<- sweep(pots.num, 2, rangs, FUN = "/")
pots<- data.frame(pots.num,pottery[10])
```

2. Per a cada nombre de clústers, $k = 1, \ldots, 6$, aplica la funció `kmeans(...,k)` i guarda els resultats en uns outputs `KM1`, ..., `KM6`. ((*Nota:* Es pot usar `for()`, però aquí es mostra com fer-ho amb lapply() o `sapply()`, variants de `apply()` aplicables a llistes o vectors que aquí hem aplicat al vector **1:6**.)) Quin objectes obtenim?

```
RNGkind(sample.kind = "Rounding") # per assegurar que set.seed dóna el mateix (podria dependre de la versió de R, entre altres
set.seed(213) # proveu primer aquesta llavor i despres repetiu tot l'exercici amb una altra llavor, podeu trobar solucions dif
# ATENCIÓ: la solució que obteniu al vostre ordinador podria ser diferent de la que he obtingut jo
llista <- lapply(1:6,function(k){kmeans(pots.num,k)})   k es troba entre 1 i 6 i apliquem la funció a aquests valors de k
names(llista) <- paste0("KM",1:6)   KM1: 1 clúester. Solució trivial. Número de clúesters 1 per a tots els objectes. Suma de quadrats entre clúesters= 0% no hi ha diferències entre clústers. Within total. Entre clusters variabilitat nul·la.
list2env(llista,globalenv()) # recupera els elements de la llista com a objectes independents al workspace
KM1; KM2; KM3; # etc                KM2: Mitjanes pels dos clústers. Suma quadrats dins de clúster s'ha reduït. La resta between.
names(KM3)                          KM3: Suma quadrats dins s'ha reduit encara més. Entre clúesters s'ha incrementat arribant al 38.8%.
                                    Variabilitat total la que teniem en KM1 entre clústers.
KM3$clusters     # classe, què és?
KM3$centers      # ? sabries escriure la fórmula?
KM3$totss        # ? sabries escriure la fórmula?   Suma quadrats totals
KM3$withinss     # ? qué és aquest vector ? Sabries escriure la fórmula del primer element del vector (per exemple)?
KM3$tot.withinss # ? relació amb l'anterior?   Suma de les sumes de quadrats dins de clústers
KM3$betweenss    # ? fórmula?   Suma quadrats entre
KM3$size         # ?   Número objectes de cada clúester
KM3$iter         # ?   Número de interacions
```

3. Extrau les sumes de quadrats *within (totals)* i guarda-les en within `wss1`, ..., `wss6`, i en un vector `wss` amb totes juntes.

```
# sumes de quadrats within, les obtinc totes de la llista
wllista <- lapply(llista,function(KM){KM[[5]]})  # per què [[5]]? perquè doble corxet?   El que volem nosaltres és tot.withinss està al número 5
names(wllista) <- paste0("wss",1:6)                Anirà a la llista dins de KM i d'aquí treurà el 5 (doble corxet). Amb un corxet et donarà una llista
wllista
list2env(wllista,globalenv())
wss3  # els tenim a l'espai de treball
wss <- unlist(wllista)   Unlist: el objectes no el volem com una llista
class(wss)
```

4. Comprova que el valor de `wss1` és igual a $(n-1)\,(Var(X_1) + \ldots + Var(X_p))$. Comprova (raonament o demostració teòrica) que aquesta fórmula coincideix amb la fórmula de **WSS** (diapositives de teoria) quan hi ha un únic clúster ($k = 1$). Raona teòricament, sense fer càlculs numèrics, que $\mathbf{TSS} = WSS$ si $k = 1$. *Nota:* Recorda que la suma de quadrats total és sempre constant independentment del nombre de clústers.

```r
n <- nrow(pots.num)
TSS <- (n-1)*sum(apply(pots.num,2,"var"))
# TSS; # wss1;
all.equal(wss1,TSS,check.attributes=F)
```

Dona TRUE

5. Calcula les sumes de quadrats between `bss` per diferència $\mathbf{BSS = TSS - }WSS$. Com les obtindries directament a partir de `KM1`, ..., `KM6` ?

```r
bss <- TSS - wss
names(bss) <- paste0("bss",1:6)
bss
# les obtinc totes de la llista
bllista <- lapply(llista,function(KM){KM[[6]]})  # per què [[5]]? perquè doble corxet?
names(bllista) <- paste0("bss",1:6)
bllista
list2env(bllista,globalenv())
bss3 # els tenim a l'espai de treball
bss <- unlist(bllista)
class(bss)
```

6. Representa les sumes de quadrats `wssk` i `bssk` respecte de k al mateix gràfic (línies, type="b"). Per què decreix `wssk` quan k creix? I inversament per a `bssk`?
7. Representa `wss` respecte de , $k = 1, \ldots, 6$ en una gràfica anomenada *de sedimentació* i interpreta-la convenientment, raonant quin és el nombre de cústers més recomanable.

```r
# Grafs 6 i 7
par(mfrow=c(1,2),cex=.7,cex.main=.7, cex.axis=.6, cex.lab=.6)
plot(1:6, wss, type = "b", lwd=2, ylim=c(0,wss[1]),
 xlab = "Nombre de clústers", ylab = "Sumes de qudrats within i between")
points(1:6, bss, type = "b",lwd = 2,col="blue")
legend("topright",legend=c("WSS","BSS"), col=c(1,4),lwd=2)
#
df<-data.frame(k=1:6,bss,wss) # data.frame auxiliar
plot(wss~k,data=df, type = "b",lwd=2,ylim=c(0,wss[1]))
# text(bss~wss,cex=.6,data=df,labels=k)
```

Aquesta gràfica (com tots els resultats anteriors) pot dependre de la llavor escollida !

8. Aplica `kmeans()` amb el valor escolllit i guarda el resultat en `pots.km`. Interpreta els clústers comparant les seves mitjanes amb una gràfica de línies per il·lustrar-ho. Comenta els resultats. Nota: Paletes de colors Fes una taula de contingència entre les variables `kiln` i `cluster`. Interpreta la taula.

```r
k<-3
pots.km<-kmeans(pots[1:9], k)
centr<-pots.km$centers
p<-ncol(centr)
par(mfrow=c(1,1),cex=.7,cex.main=.7, cex.axis=.6, cex.lab=.6)
etiq<-colnames(centr)
matplot(1:p,t(centr),type="o",pch=19,lty=1:k,lwd=2,col=rainbow(k),axes=F,xlab="variables",ylab="mitjanes",cex=.7)
axis(2,seq(0,2,by=0.25),las=1)
axis(1,1:p,labels=etiq,line=1,cex.axis=.6)
legend("topright",legend=1:k,lty=1:k,col=rainbow(k))
table(pots.km$cluster,pots$kiln)
```

9. Aplica components principals amb cinc components (valor per defecte) a les variables numèriques i amb `cluster` com a qualitativa. Interpreta els resultats.

```r
pots2<-data.frame(pots[-10],clus=as.factor(pots.km$cluster))
require(FactoMineR)
pc2<-PCA(pots2,scale.unit=F,quali.sup=10,graph=F)
par(mfrow=c(2,2),cex=.7,cex.main=.7, cex.axis=.6, cex.lab=.6)
plot(pc2,choix="var",graph.type = "classic")
plot(pc2,habillage=10,col.hab=rainbow(k),graph.type = "classic")
plot(pc2,axes=c(1,3),habillage=10,col.hab=rainbow(k),graph.type = "classic")
plot(pc2,axes=c(2,3),habillage=10,col.hab=rainbow(k),graph.type = "classic")
```

Comenta la gràfica de components principals, en general i aprofundint en els clusters en particular: et sembla **vàlida** la solució amb 3 clústers? Justifica la resposta.

La validació sembla bona.

El clúster 2 (el verd) és el que representa que té més MgO, K20, MnO. Clúster 3 (blau) té més BaO, Fe2O3

Ha d'haver-hi cohèrencia entre tots els gràfics.

•Mètodes basats en models: Experança i maximització (*EM*) - Cas Gaussià

9. Reanalitza les dades pottery amb el mètode EM (*Model-Based clustering*) amb la distribució Gaussiana, usant la funció `Mclust()` de la llibreria `mclust`.
10. Explora el codi següent i els resultats, buscant les ajudes que facin falta.

```
require(mclust)
pots.em<-Mclust(pots.num,G=1:5,verbose=FALSE)
print(pots.em)
summary(pots.em)

plot(pots.em, what = "BIC", col = "black", ylab = "-BIC")
pots.em3<-Mclust(pots[1:9], G=3, modelNames="VVI",verbose=FALSE)
emclus<-pots.em3$classification
table(emclus)
table(emclus, pots.km$cluster)

mitjanes<-pots.em3$parameters$mean
colnames(mitjanes)<-unique(emclus)
mitjanes

clPairs(pots[1:5], classification = emclus, symbols = 1:3, col = "black") ## algunes variables

pots3<-data.frame(pots[-10],clus=as.factor(emclus))
require(FactoMineR)
pc3<-PCA(pots3,quali.sup=10,graph=F)
par(mfrow=c(1,2),cex=.7,cex.main=.7, cex.axis=.6, cex.lab=.6)
plot(pc3,choix="var",graph.type = "classic")
plot(pc3,choix="ind",habillage=10,graph.type = "classic")
```

Com més alt millor. Si surt mirar-me el video min 35.

Aquesta taula compara les dues solucions KM (les clumnes) i EM (les files).

	1	2	3
1	0	0	20
2	0	14	1
3	10	0	0

El que abans era clúster 1 ara és 3...

Ref: Everitt-Hothorn.

Mclust:
https://journal.r-project.org/archive/2016/RJ-2016-021/RJ-2016-021.pdf