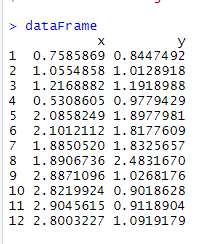
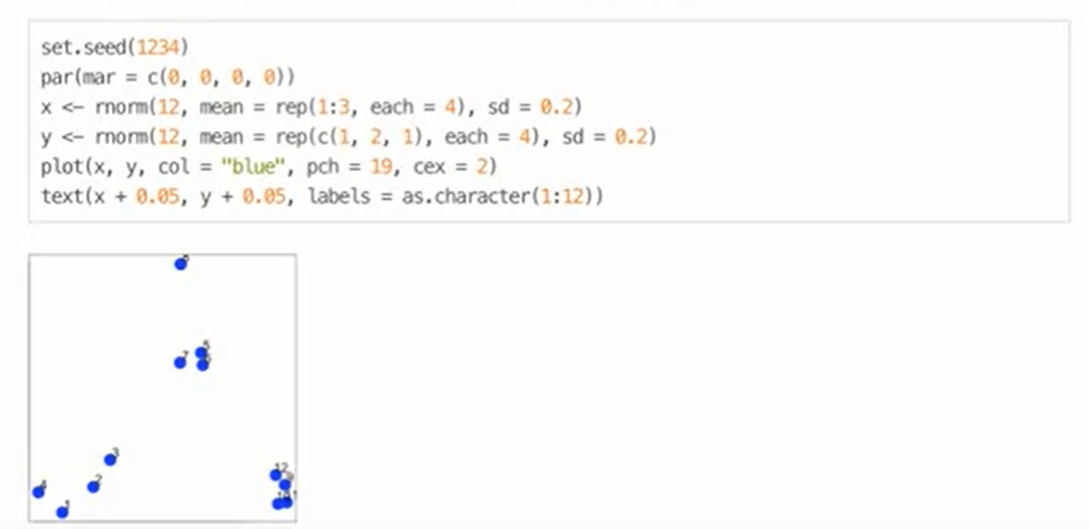
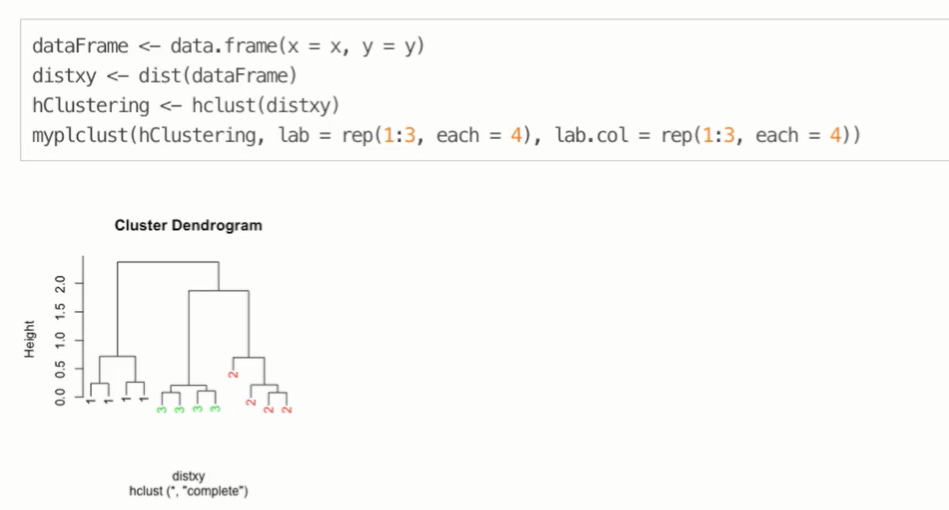
**Hierarchical Clustering**

****

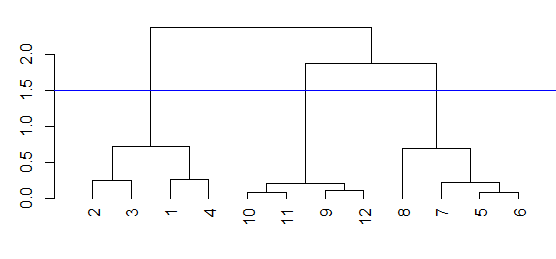




hc <- hclust(distxy)#distxy data frame de distancia por pares

plot(as.dendrogram(hc))

abline(h=1.5,col="blue")

****

Observe que las alturas verticales de las líneas y el etiquetado de la escala en el borde izquierdo dan alguna indicación de distancia.

Vemos que esta línea azul cruza 3 líneas verticales y esto nos dice que usando la distancia 1.5 (unidades no especificadas) nos da 3 grupos (1 a 4), (9 a 12) y (5 a 8). A esto lo llamamos un "corte" de nuestro dendrograma.

al cortar a .4 tenemos 5 grupos, lo que indica que esta distancia es lo suficientemente pequeño como para dividir nuestra agrupación original de puntos

Entonces, la cantidad de clústeres en sus datos depende de dónde dibuje la línea

si está tratando de medir una distancia entre dos grupos, tome la mayor distancia entre los pares de puntos en esos dos grupos.

La segunda forma de medir una distancia entre dos conglomerados que solo haremos a mención se llama vinculación promedio. Primero, calcula un punto "promedio" en cada cúmulo (considérelo el centro de gravedad del cúmulo). Haces esto por calcular las coordenadas xey medias (promedio) de los puntos en el grupo.

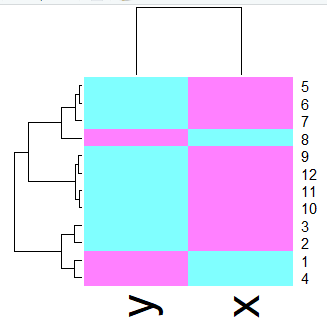
Luego, calcula las distancias entre el promedio de cada grupo para calcular el distancia entre clústeres

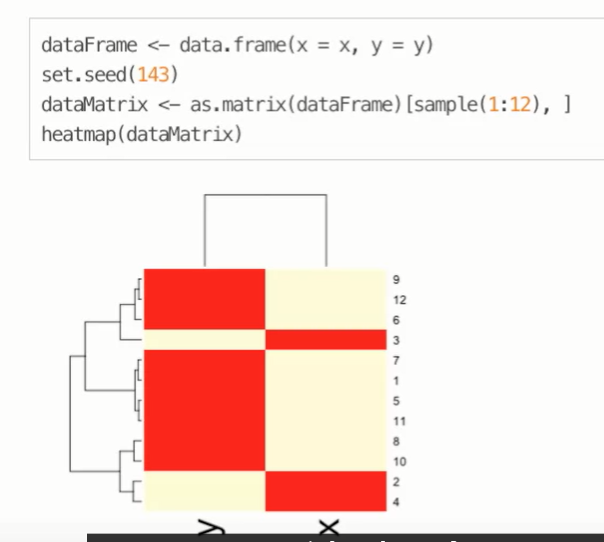
El último método de visualización de datos que mencionaremos en esta lección se refiere al calor mapas

un mapa de calor es "una representación gráfica de datos donde los valores individuales contenidos en una matriz se representan como colores. ... Los mapas de calor se originan en pantallas 2D de los valores en una matriz de datos. Los valores más grandes fueron representados por pequeños cuadrados grises oscuros o negros (píxeles) y los valores más pequeños por cuadrados más claros ".

No diremos demasiado sobre este tema, pero existe un tutorial conciso muy agradable sobre cómo crear mapas de calor en R en <http://sebastianraschka.com/Articles/heatmaps_in_r.html#clustering>

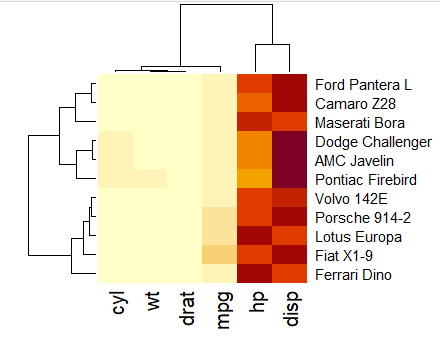
heatmap(dataMatrix,col=cm.colors(25))

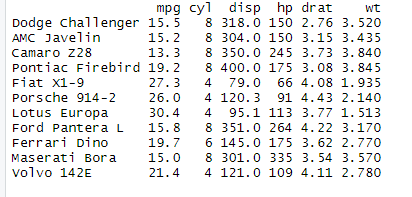




Dataset mtcars

heatmap(mt)





**K-means**

Como dijimos, k-means es un enfoque de partición que requiere que primero adivine cuántos clústeres tiene (o desea). Una vez que fija este número, crea aleatoriamente un "centroide" (un punto fantasma) para cada grupo y asigna cada punto u observación en su conjunto de datos al centroide al que está más cerca. Una vez que a cada punto se le asigna un centroide, reajusta la posición del centroide convirtiéndola en el promedio de los puntos asignados.

Una vez que haya reposicionado los centroides, debe recalcular la distancia de las observaciones a los centroides y reasignar cualquiera, si es necesario, al centroide más cercano a ellos. Nuevamente, una vez que se realizan las reasignaciones, reajuste las posiciones de los centroides en función de la nueva pertenencia al clúster. El proceso se detiene una vez que llega a una iteración en la que no se realizan ajustes o cuando ha alcanzado un número máximo predeterminado de iteraciones.

Cuando finaliza, el agrupamiento de k-medias devuelve una posición final del centroide de cada grupo, así como la asignación de cada punto de datos u observación a un grupo.

cmat

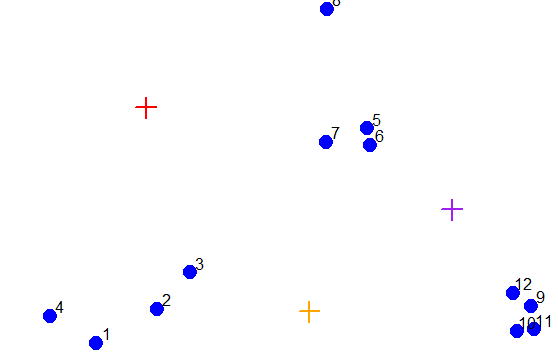
[,1] [,2] [,3]

[1,] 1 1.8 2.5

[2,] 2 1.0 1.5 //puntos iniciales del clustering, utilizando los mismos puntos del ejercicio anterior

##insertar codigo para graficar los puntos

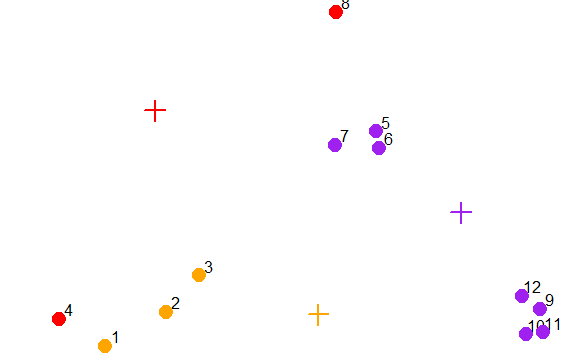
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)#grafica los cluster



distTmp <-mdist(x,y,cx,cy)

newClust<-apply(distTmp,2,which.min)

points(x,y,pch=19,cex=2,col=cols1[newClust])



newCx<-tapply(x,newClust,mean)

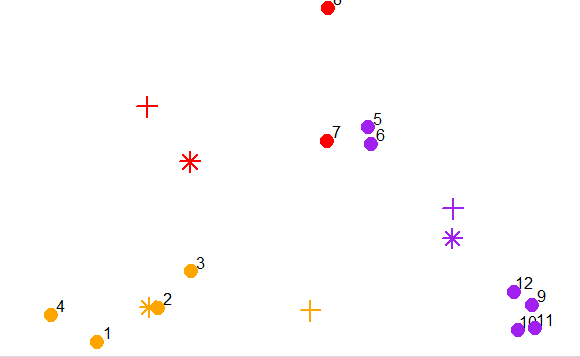
newCy<-tapply(y,newClust,mean)

points(newCx,newCy,col=cols1,pch=8,cex=2,lwd=2)

distTmp2<-mdist(x,y,newCx,newCy)

newClust2<- apply(distTmp2,2,which.min)

points(x,y,pch=19,cex=2,col=cols1[newClust2])



finalCx<-tapply(x,newClust,mean)

finalCy<-tapply(y,newClust,mean)

points(finalCx, finalCy,col=cols1,pch=9,cex=2,lwd=2)

distTmp3<-mdist(x,y,finalCx,finalCy)

newClust3<- apply(distTmp3,2,which.min)

points(x,y,pch=19,cex=2,col=cols1[newClust3])

todo esto se puede hacer con un comando en R

kmObj <-kmeans(dataFrame,centers=3)

plot(x,y,col=kmObj$cluster,pch=19,cex=2)



//el problema aquí es que tiene un inicio de cluster aleatorio por lo cual podria ser necesario ejecutar varias veces el comando para notar las relaciones de los datos