K means clustering

luis manuel

19/12/2020

K means clustering

K_Means_Clustering. (Las diapositivas para este y otros cursos de ciencia de datos se pueden encontrar en github https://github.com/DataScienceSpecialization/courses/. Si desea usarlas, debe descargarlas como un archivo zip y verlas localmente. Esta lección corresponde a 04_ExploratoryAnalysis / kmeansClustering.)

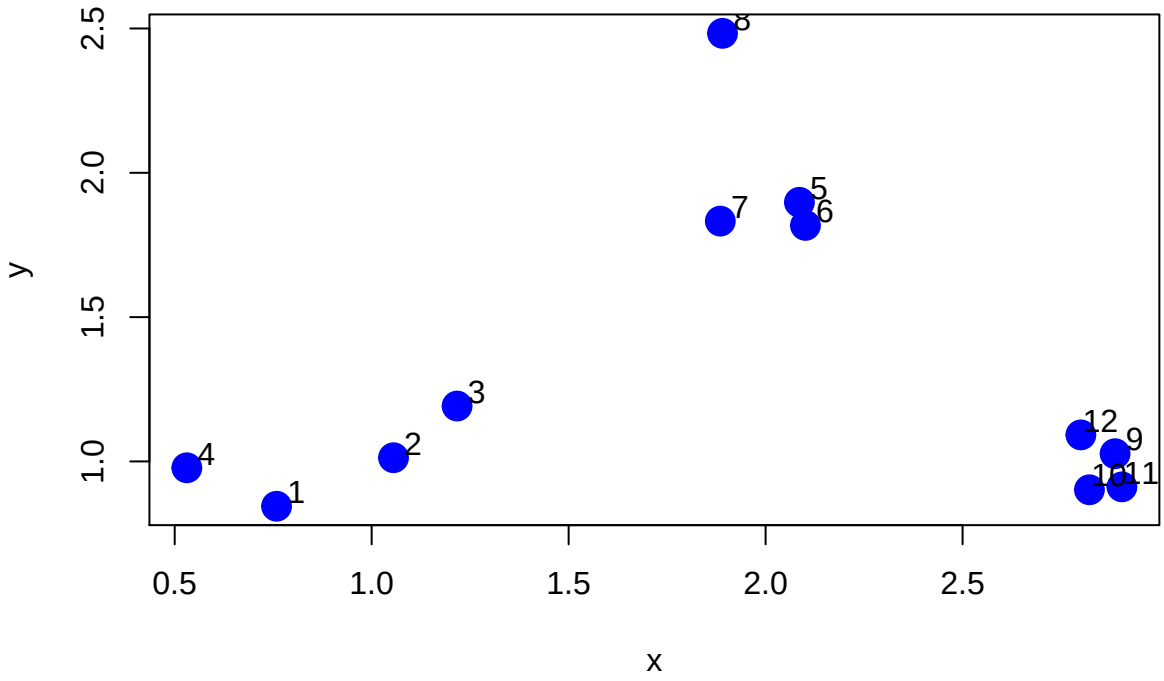
En esta lección, aprenderemos sobre la agrupación en clústeres de k-medias, otra forma sencilla de examinar y organizar datos multidimensionales. Al igual que con la agrupación jerárquica, esta técnica es más útil en las primeras etapas del análisis cuando intenta comprender los datos, por ejemplo, encontrar algún patrón o relación entre diferentes factores o variables.

La documentación de R nos dice que el método de k-medias "tiene como objetivo dividir los puntos en k grupos de manera que la suma de los cuadrados desde los puntos hasta los centros de los grupos asignados se minimice".

Dado que la agrupación organiza los puntos de datos que están cerca en grupos, asumiremos que hemos decidido una medida de distancia, por ejemplo, euclidiana.

Para ilustrar el método, usaremos estos puntos aleatorios que generamos, familiares para usted si ya ha pasado por la lección de agrupamiento jerárquico. Demostraremos el agrupamiento de k-medias en varios pasos, pero primero explicaremos la idea general.

```
set.seed(1234)
 x <- rnorm(12, rep(1:3, each = 4), 0.2)
 y <- rnorm(12, rep(c(1, 2, 1), each = 4), 0.2)
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
```



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 volver a calcular 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.

Entonces, el agrupamiento de k-medias requiere alguna métrica de distancia (digamos euclidiana), un número fijo hipotético de grupos y una suposición inicial en cuanto a los centroides del grupo. Como se describe, ¿qué produce este proceso?

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.

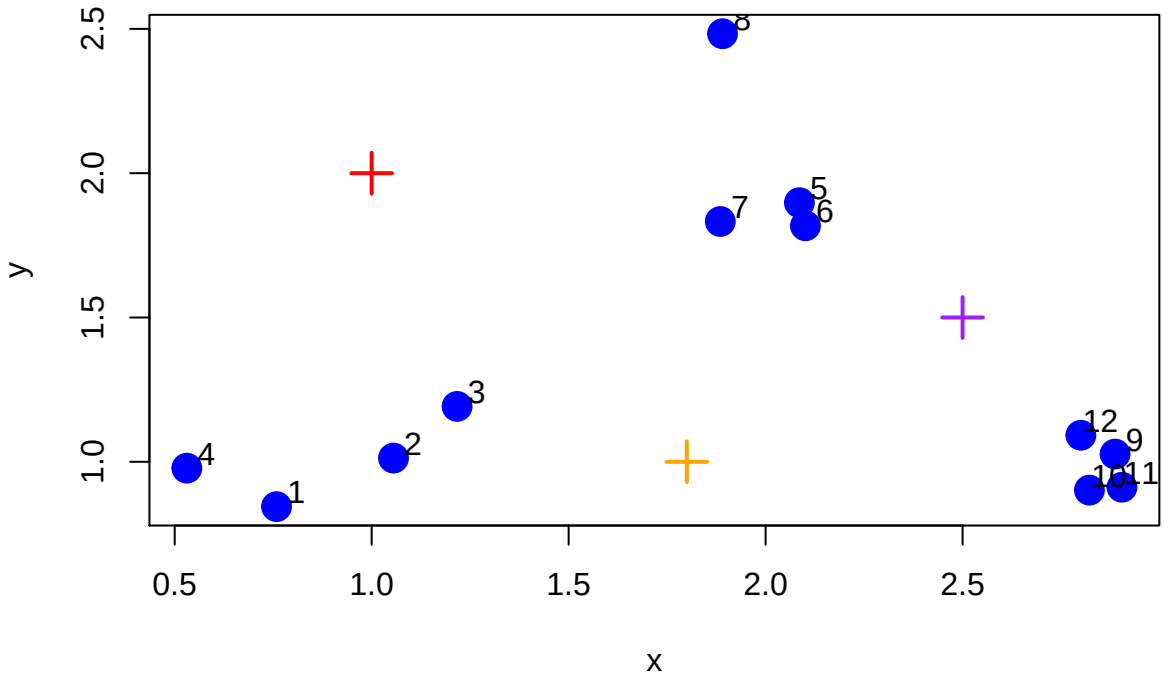
Ahora pasaremos por este proceso usando nuestros puntos aleatorios como nuestros datos. Las coordenadas de estos se almacenan en 2 vectores, xey. Observamos la pantalla y adivinamos que hay 3 grupos. Elegiremos 3 posiciones de centroides, una para cada grupo.

Hemos creado dos vectores de 3 largos para usted, cx y cy. Estos contienen respectivamente las coordenadas x e y para los 3 centroides propuestos. Para mayor comodidad, también los hemos almacenado en una matriz cmat de 2 por 3. Las coordenadas x están en la primera fila y las coordenadas y en la segunda. Mire cmat ahora.

```
cx<-c(1,1.8,2.5)
cy<-c(2,1,1.5)
cmat<-rbind(cx,cy)
```

Las coordenadas de estos puntos son (1,2), (1.8,1) y (2.5,1.5). Agregaremos estos centroides al gráfico de nuestros puntos. Haga esto llamando a los puntos de comando R con 6 argumentos. Los primeros 2 son cx y cy, y el tercero es col set igual a la concatenación de 3 colores, "rojo", "naranja" y "violeta". El cuarto argumento es pch establecido igual a 3 (un signo más), el quinto es cex establecido igual a 2 (expansión de carácter), y el final es lwd (ancho de línea) también establecido igual a 2.

```
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)
```



Vemos que el primer centroide (1,2) está en rojo. El segundo (1.8,1), a la derecha y debajo del primero, es naranja, y el centroide final (2.5,1.5), el más a la derecha, es violeta.

```
mdist<-function(x,y,cx,cy){
  distTmp <- matrix(NA,nrow=3,ncol=12)
  distTmp[1,] <- (x-cx[1])^2 + (y-cy[1])^2
  distTmp[2,] <- (x-cx[2])^2 + (y-cy[2])^2
  distTmp[3,] <- (x-cx[3])^2 + (y-cy[3])^2
  return(distTmp)
}
```

Hemos escrito una función para usted llamada mdist que toma 4 argumentos. Los vectores de puntos de

datos (xey) son los dos primeros y los dos vectores de coordenadas del centroide (cx y cy) son los dos últimos. Llame a mdist ahora con estos argumentos.

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

```
##          [,1]      [,2]      [,3]     [,4]      [,5]      [,6]      [,7]
## [1,] 1.392885 0.9774614 0.7000680 1.264693 1.1894610 1.2458771 0.8113513
## [2,] 1.108644 0.5544675 0.3768445 1.611202 0.8877373 0.7594611 0.7003994
## [3,] 3.461873 2.3238956 1.7413021 4.150054 0.3297843 0.2600045 0.4887610
##          [,8]      [,9]     [,10]     [,11]     [,12]
## [1,] 1.026750 4.5082665 4.5255617 4.8113368 4.0657750
## [2,] 2.208006 1.1825265 1.0540994 1.2278193 1.0090944
## [3,] 1.337896 0.3737554 0.4614472 0.5095428 0.2567247
```

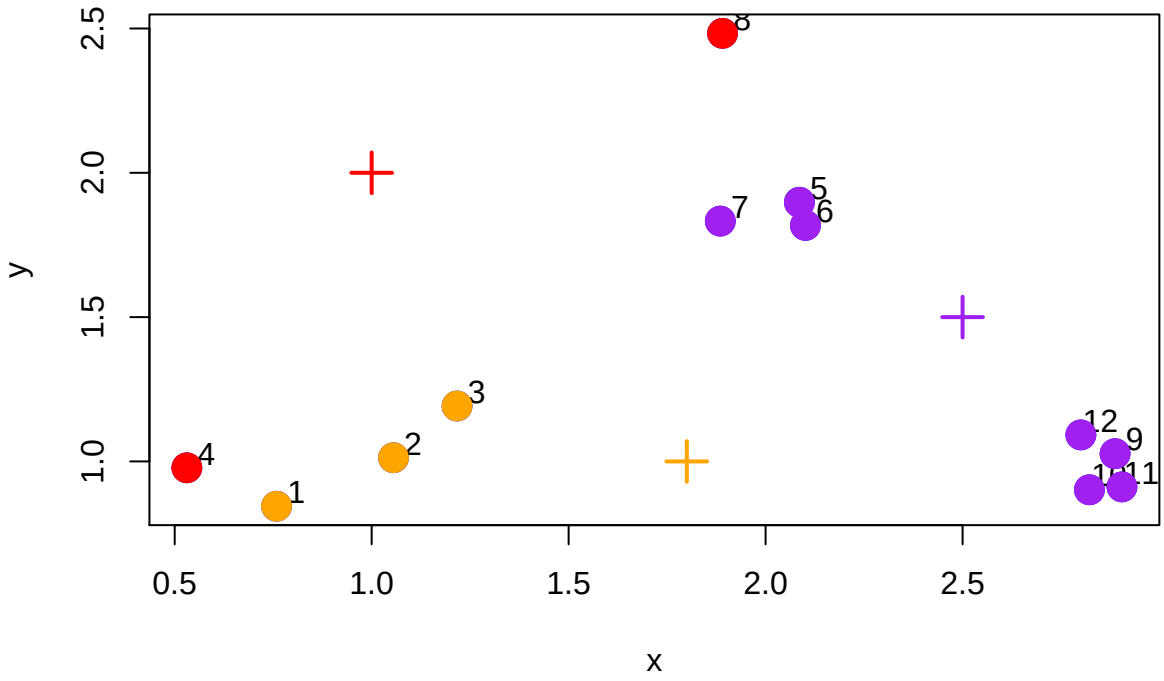
Hemos almacenado estas distancias en la matriz distTmp para usted. Ahora tenemos que asignar un grupo a cada punto.

R tiene una función práctica which.min que puede aplicar a TODAS las columnas de distTmp con una llamada. Simplemente llame a la función R aplicar con 3 argumentos. El primero es distTmp, el segundo es 2, que significa las columnas de distTmp, y el tercero es which.min, la función que desea aplicar a las columnas de distTmp. Pruebe esto ahora.

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

Hemos almacenado el vector de colores del clúster ("rojo", "naranja", "púrpura") en la matriz cols1 para usted y también hemos almacenado las asignaciones del clúster en la matriz newClust. Coloreemos los 12 puntos de datos de acuerdo con sus asignaciones. Nuevamente, use los puntos de comando con 5 argumentos. Los primeros 2 son x e y. El tercero es pch establecido en 19, el cuarto es cex establecido en 2 y el último, col está establecido en cols1 [newClust].

```
cols1<-c("red"    , "orange", "purple")
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust])
```



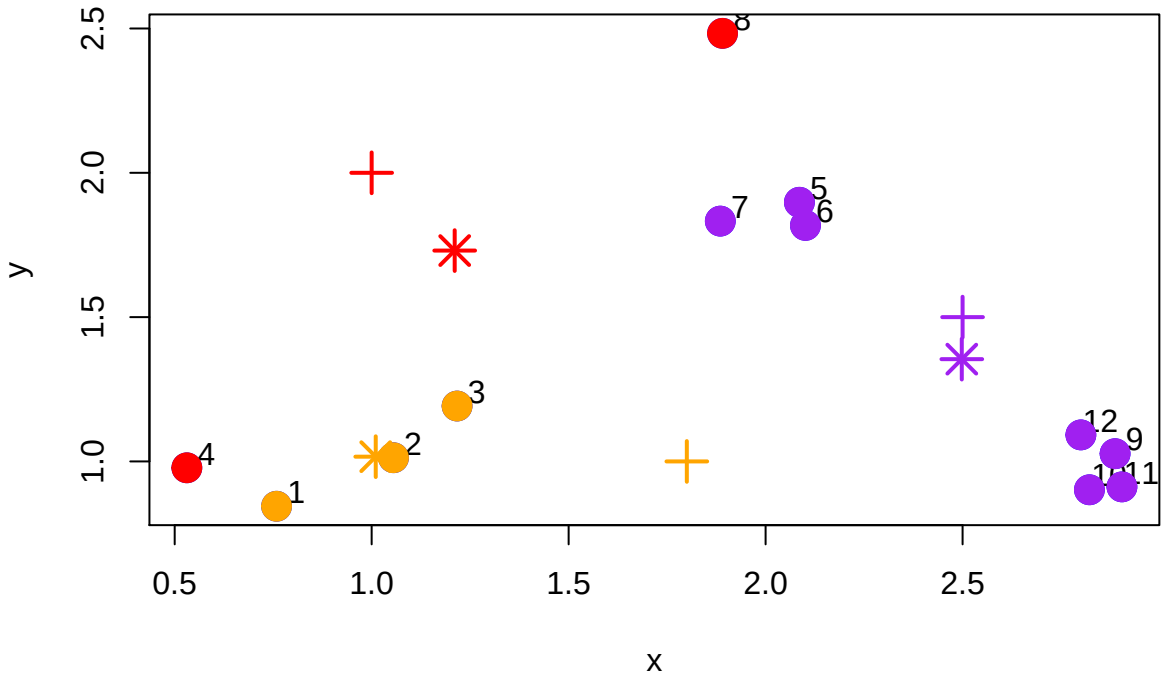
Ahora tenemos que recalcular nuestros centroides para que sean el promedio (centro de gravedad) del grupo de puntos asignados a ellos. Tenemos que hacer las coordenadas xey por separado. Primero haremos la coordenada x. Recuerde que los vectores xey tienen las coordenadas respectivas de nuestros 12 puntos de datos.

Podemos usar la función R tapply que aplica "una función sobre una matriz irregular". Esto significa que a cada elemento de la matriz se le asigna un factor y la función se aplica a subconjuntos de la matriz (identificados por el vector de factor). Esto nos permite aprovechar el factor vector newClust que calculamos. Llame a tapply ahora con 3 argumentos, x (los datos), newClust (la matriz de factores) y mean (la función a aplicar).

```
newCx<-tapply(x,newClust,mean)
newCy<-tapply(y,newClust,mean)
```

Ahora que tenemos nuevas coordenadas xy nuevas y para los 3 centroides, podemos trazarlos. Hemos almacenado las coordenadas para usted en las variables newCx y newCy. Utilice los puntos de comando R con estos como los primeros 2 argumentos. Además, utilice los argumentos col set igual a cols1, pch igual a 8, cex igual a 2 y lwd también igual a 2.

```
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust])
points(newCx,newCy,col=cols1,pch=8,cex=2,lwd=2)
```

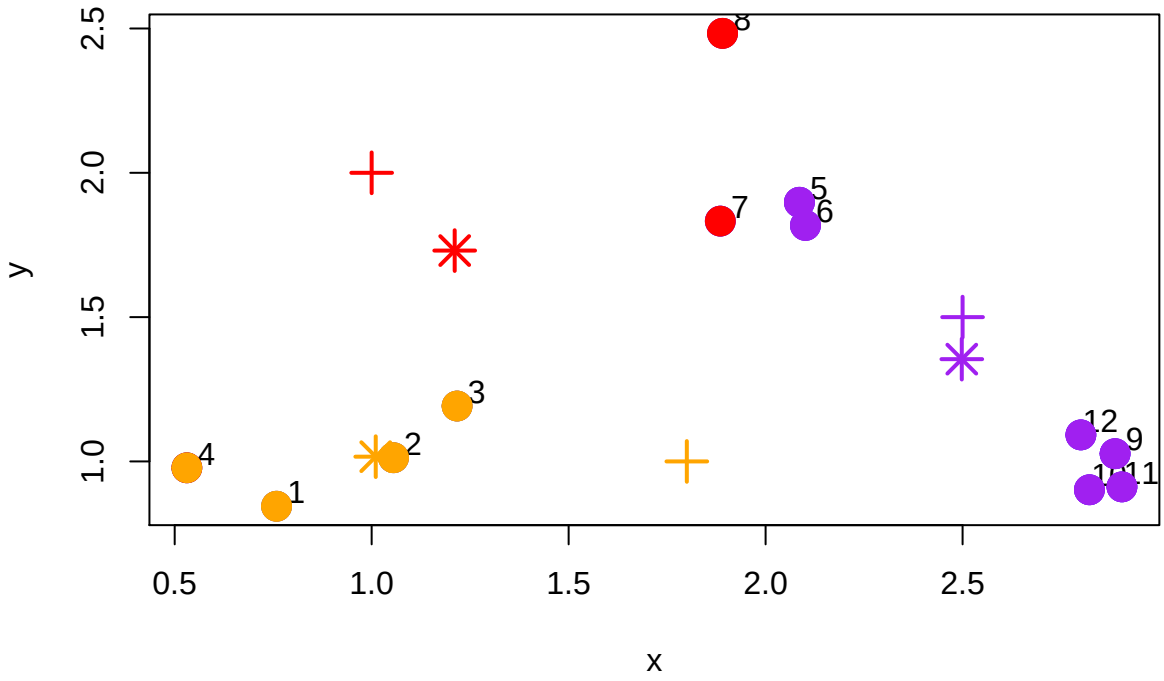


Vemos cómo los centroides se han acercado a sus respectivos grupos. Esto es especialmente cierto mdist con los 4 argumentos x, y, newCx y newCy. Esto nos permitirá reasignar los puntos de datos a nuevos clústeres si es necesario.

```
distTmp2<-mdist(x,y,newCx,newCy)
newClust2<-apply(distTmp2,2,which.min)
```

Hemos almacenado las nuevas asignaciones de clústeres en un vector de factores llamado newClust2. Utilice los puntos de la función R para cambiar el color de los puntos con sus nuevas asignaciones. Nuevamente, hay 5 argumentos, xey son primero, seguidos de pch establecido en 19, cex en 2 y cols1 [newClust2].

```
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust])
points(newCx,newCy,col=cols1,pch=8,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust2])
```

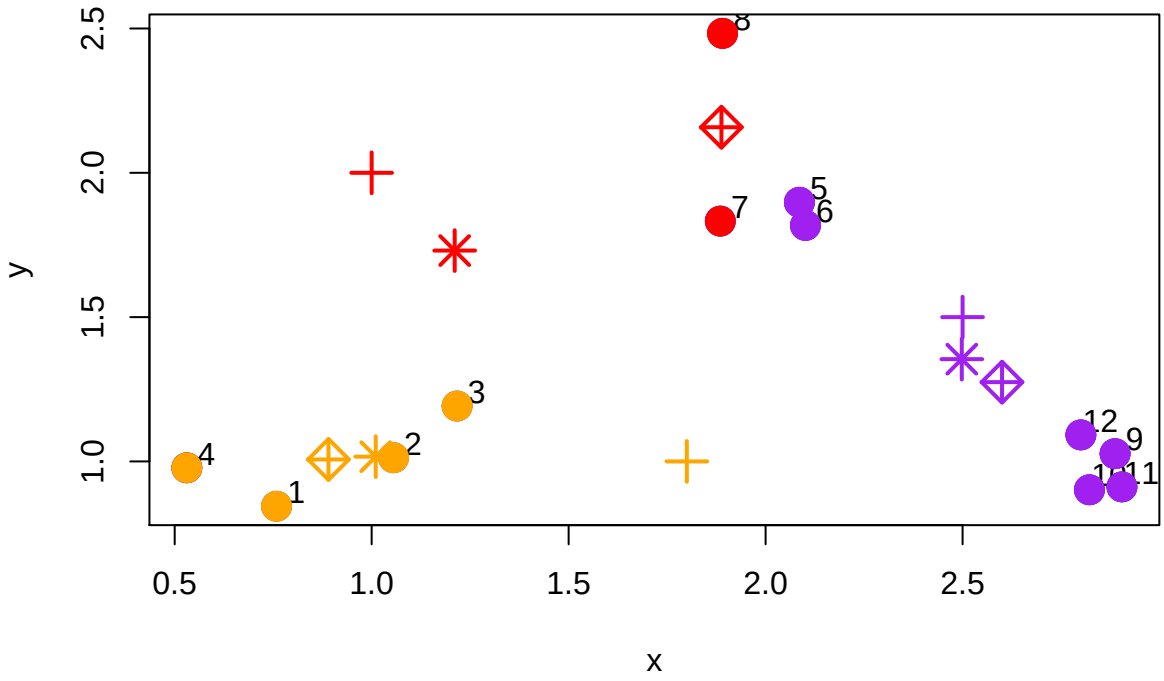


Ahora use tapply para encontrar la coordenada x del nuevo centroide. Recuerde que hay 3 argumentos, x, newClust2 y mean.

```
finalCx<-tapply(x,newClust2,mean)
finalCy<-tapply(y,newClust2,mean)
```

Hemos almacenado estas coordenadas para usted en las variables finalCx y finalCy. Grafique estos nuevos centroides usando la función de puntos con 6 argumentos. Los 2 primeros son finalCx y finalCy. El argumento col debe ser igual a cols1, pch debe ser igual a 9, cex 2 y lwd 2.

```
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
points(cx,cy,col=c("red","orange","purple"),pch=3,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust])
points(newCx,newCy,col=cols1,pch=8,cex=2,lwd=2)
points(x,y,pch=19,cex=2,col=cols1[newClust2])
points(finalCx,finalCy,col=cols1,pch=9,cex=2,lwd=2)
```



Debería ser obvio que si continuamos con este proceso, los puntos 5 a 8 se volverían rojos, mientras que los puntos 1 a 4 permanecerían naranjas y los puntos 9 a 12 morados.

Ahora que ha pasado por un ejemplo paso a paso, se sentirá aliviado al saber que R proporciona un comando para hacer todo este trabajo por usted. Como era de esperar, se llama kmeans y, aunque tiene varios parámetros, solo mencionaremos cuatro. Estos son x, (la matriz numérica de datos), centros, iter.max y nstart. El segundo de estos (centros) puede ser un número de grupos o un conjunto de centroides iniciales. El tercero, iter.max, especifica el número máximo de iteraciones por recorrer, y nstart es el número de inicios aleatorios que desea probar si especifica centros como un número.

Llame a kmeans ahora con 2 argumentos, dataFrame (que contiene las coordenadas xey de nuestros 12 puntos) y centros establecidos en 3.

```
dataFrame<-cbind(x,y)
kmObj<-kmeans(dataFrame,centers=3)
kmObj
```

```
## K-means clustering with 3 clusters of sizes 2, 8, 2
##
## Cluster means:
##           x         y
## 1 1.1361870 1.1023953
## 2 2.4220935 1.4954726
## 3 0.6447237 0.9113461
##
## Clustering vector:
##  [1] 3 1 1 3 2 2 2 2 2 2 2 2
```

```
##
## Within cluster sum of squares by cluster:
## [1] 0.02904713 3.96919665 0.03479993
##  (between_SS / total_SS =  64.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

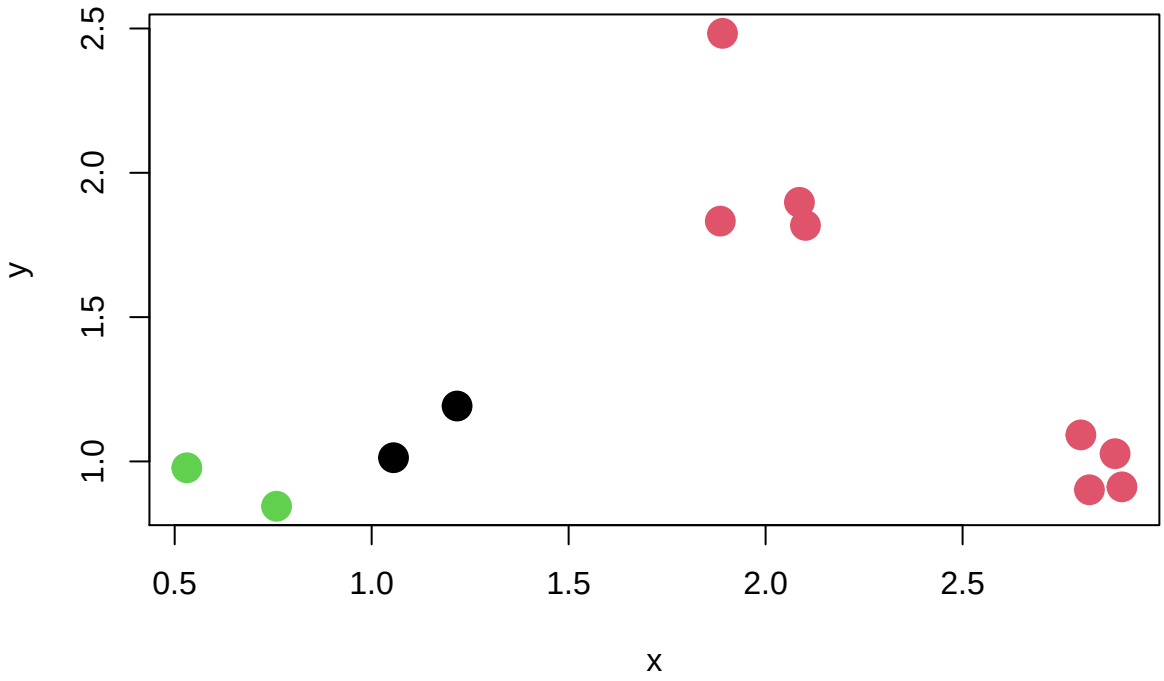
El programa devuelve la información de que los datos se agruparon en 3 grupos, cada uno de tamaño 4. También devuelve las coordenadas de las 3 medias del grupo, un vector llamado grupo que indica cómo se dividieron los 12 puntos en los grupos y la suma de cuadrados dentro de cada uno. racimo. También muestra todos los componentes disponibles devueltos por la función. Hemos almacenado estos datos para usted en un objeto kmeans llamado kmObj. Mire kmObj $ iter para ver cuántas iteraciones atravesó el algoritmo.

```
kmObj$iter
```

```
## [1] 1
```

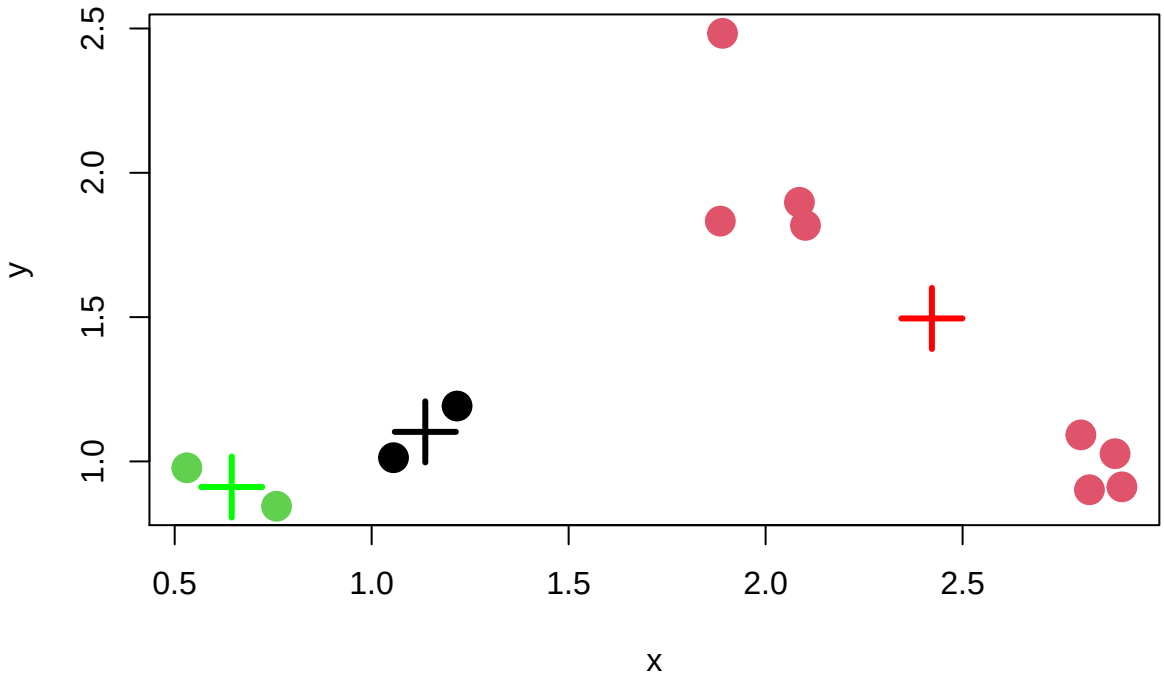
Dos iteraciones como hicimos antes. Solo queremos enfatizar cómo puede acceder a la información disponible para usted. Tracemos los puntos de datos codificados por colores de acuerdo con su grupo. Esto se almacenó en el clúster kmObj $. Ejecute la trama con 5 argumentos. Los datos, xey, son los dos primeros; el tercero, col se establece igual a kmObj $ cluster, y los dos últimos son pch y cex. El primero de estos debe establecerse en 19 y el último en 2.

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



Ahora agregue los centroides que están almacenados en los centros kmObj $. Usa la función de puntos con 5 argumentos. Los dos primeros son kmObj $ centres y col = c ("negro", "rojo", "verde"). Los últimos tres, pch, cex y lwd, deben ser iguales a 3.

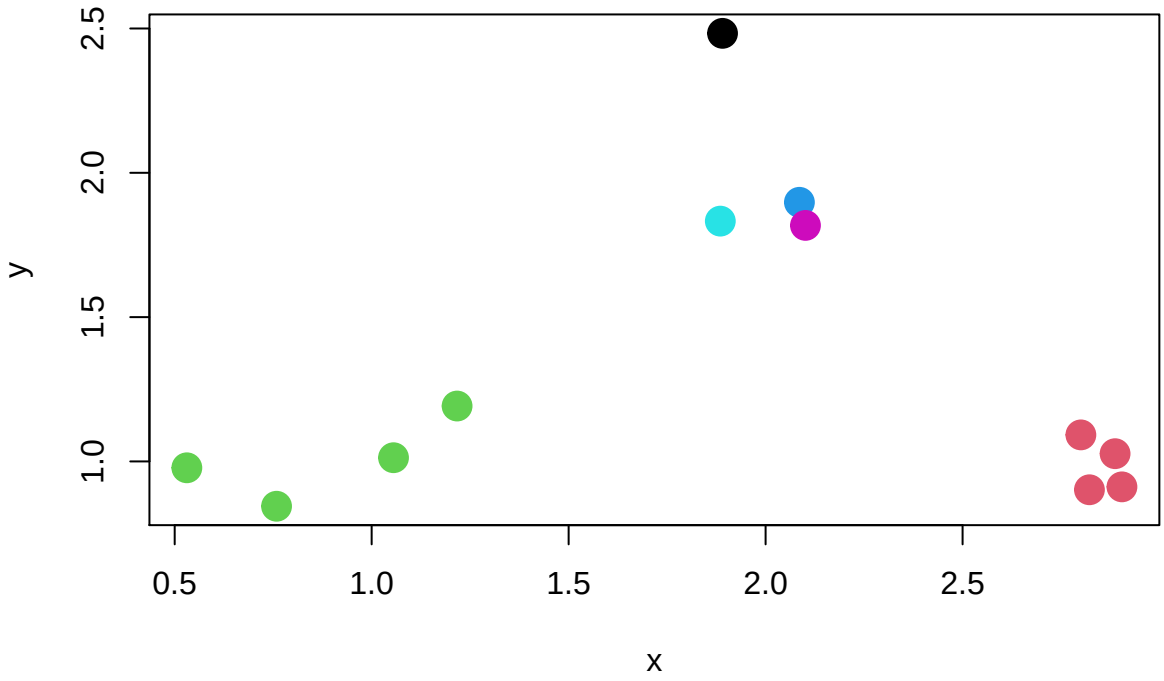
```
plot(x,y,col=kmObj$cluster,pch=19,cex=2)
points(kmObj$centers,col=c("black","red","green"),pch=3,cex=3,lwd=3)
```



¡Ahora para divertirse! Queremos mostrarle cómo se inicia la salida de la función kmeans (cuando solo solicita una serie de clústeres). Con inicios aleatorios, es posible que desee ejecutar la función varias veces para tener una idea de las relaciones entre sus observaciones. Llamaremos kmeans con los mismos puntos de datos (almacenados en dataFrame), pero pediremos 6 clústeres en lugar de 3.

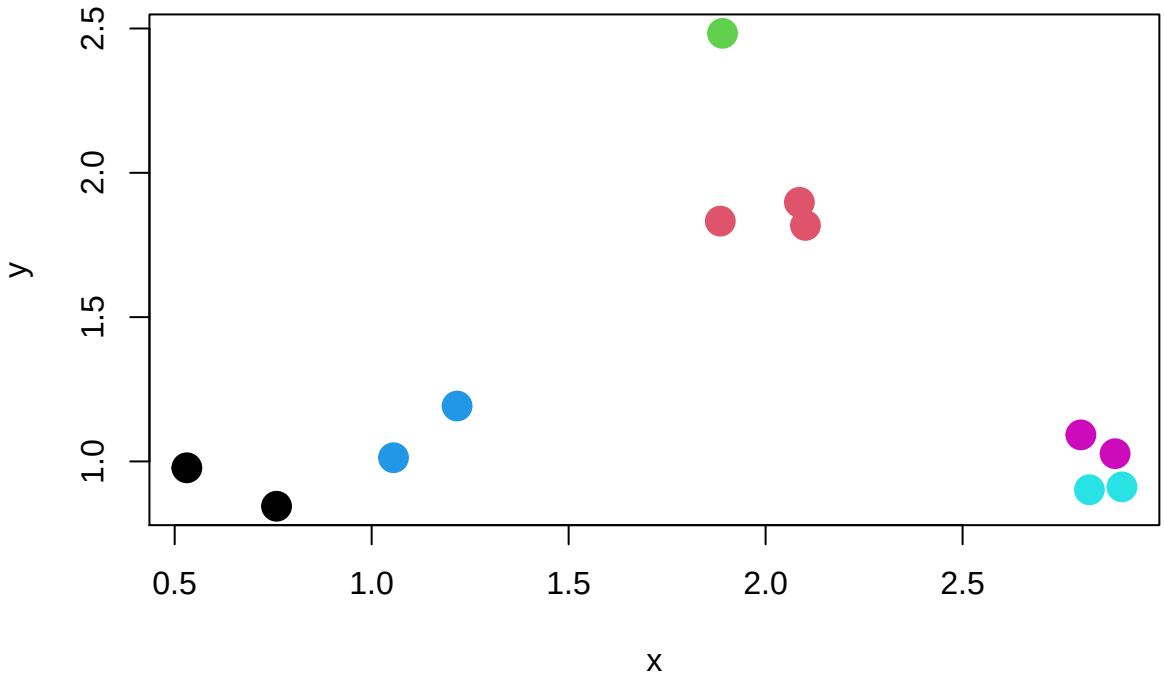
Trazaremos nuestros puntos de datos varias veces y cada vez cambiaremos el argumento col, lo que nos mostrará cómo la función R k significa que los agrupa. Entonces, llame a plot ahora con 5 argumentos. Los primeros 2 son x e y. El tercero es col set igual a la llamada kmeans (dataFrame, 6) $ cluster. Los dos últimos (pch y cex) se establecen en 19 y 2 respectivamente.

```
plot(x,y,col=kmeans(dataFrame,6)$cluster,pch=19,cex=2)
```



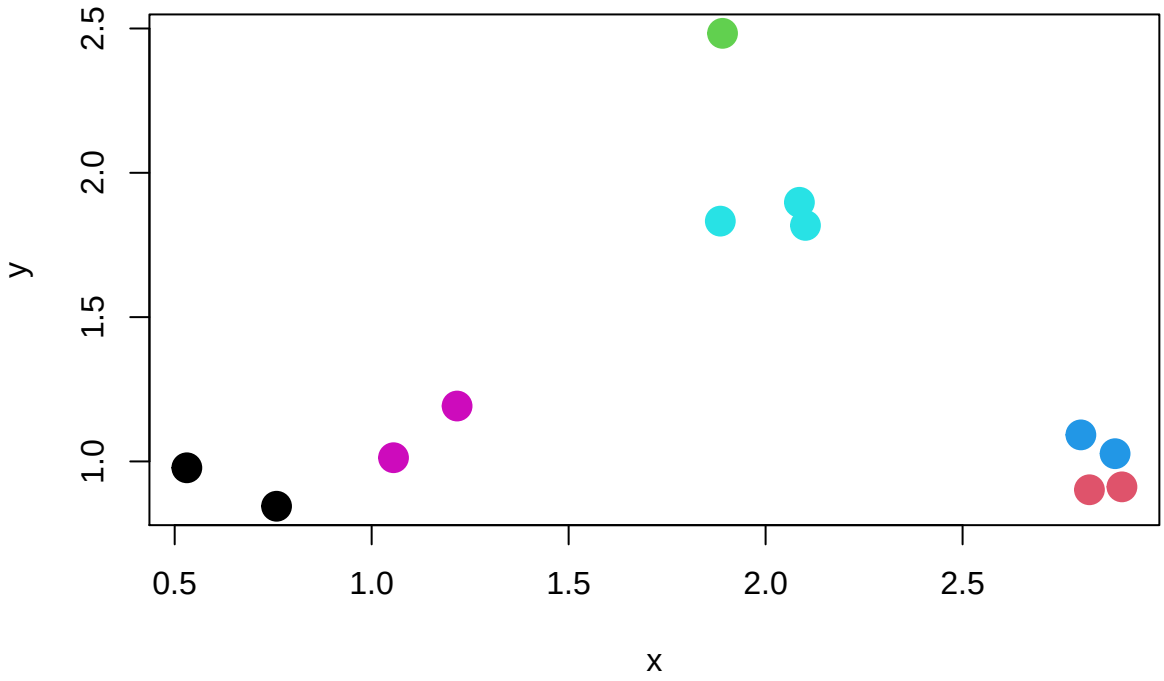
¿Ves cómo se agrupan los puntos? Ahora recuerde su último comando y vuelva a ejecutarlo.

```
plot(x,y,col=kmeans(dataFrame,6)$cluster,pch=19,cex=2)
```



¿Ves cómo ha cambiado la agrupación? Como dirían los Teletubbies, "¡Otra vez! ¡Otra vez!"

```
plot(x,y,col=kmeans(dataFrame,6)$cluster,pch=19,cex=2)
```



Entonces, la agrupación cambia con diferentes comienzos. ¿Quizás 6 son demasiados grupos?