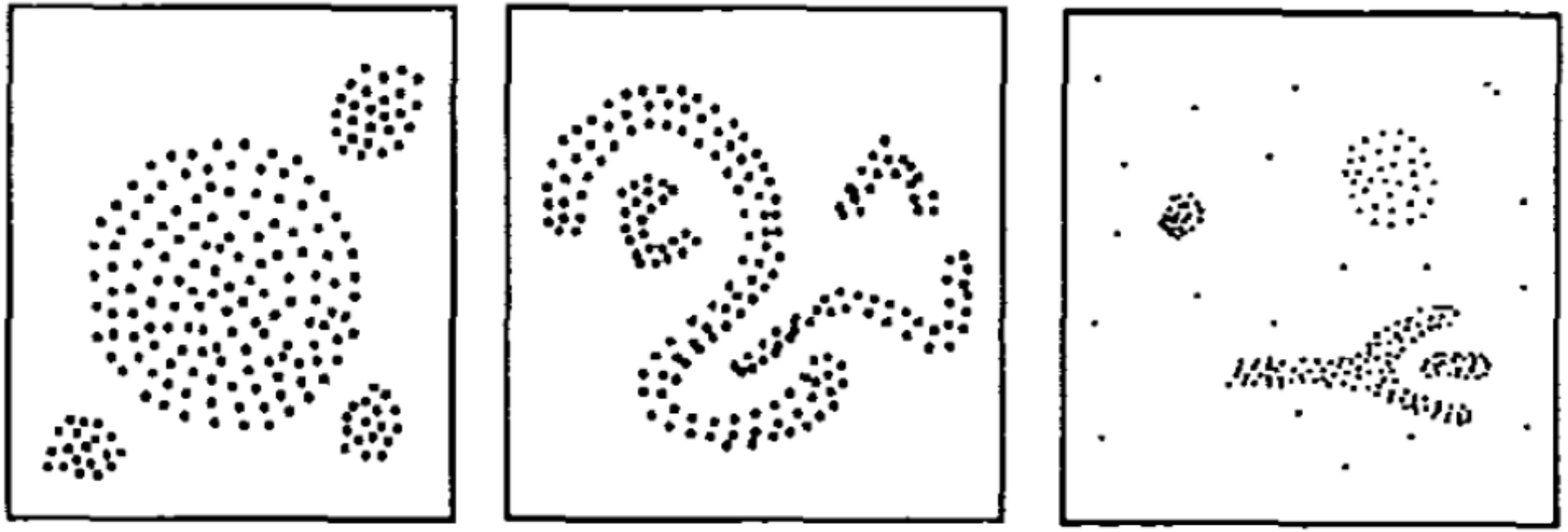
Modelos de Soporte No Supervisado

DBSCAN: Density-Based Clustering

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

- DBSCAN (Agrupamiento espacial basado en densidad y aplicación con ruido), es un algoritmo de clustering basado en la densidad, introducido por Ester et al. 1996
- Se utilizará para identificar clusters en un conjunto de datos que contenga ruido y valores atípicos.
- La idea básica detrás del enfoque de agrupamiento basado en la densidad se deriva de un método de agrupamiento intuitivo.
- Los clusters son regiones densas en el espacio de datos, separadas por regiones de menor densidad de puntos.
- El algoritmo DBSCAN se basa en esta noción intuitiva de "clusters" y "ruido".
- La idea clave es que para cada punto de un clúster, el vecindario de un determinado radio debe contener al menos un número mínimo de puntos.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)



database 1 database 2 database 3

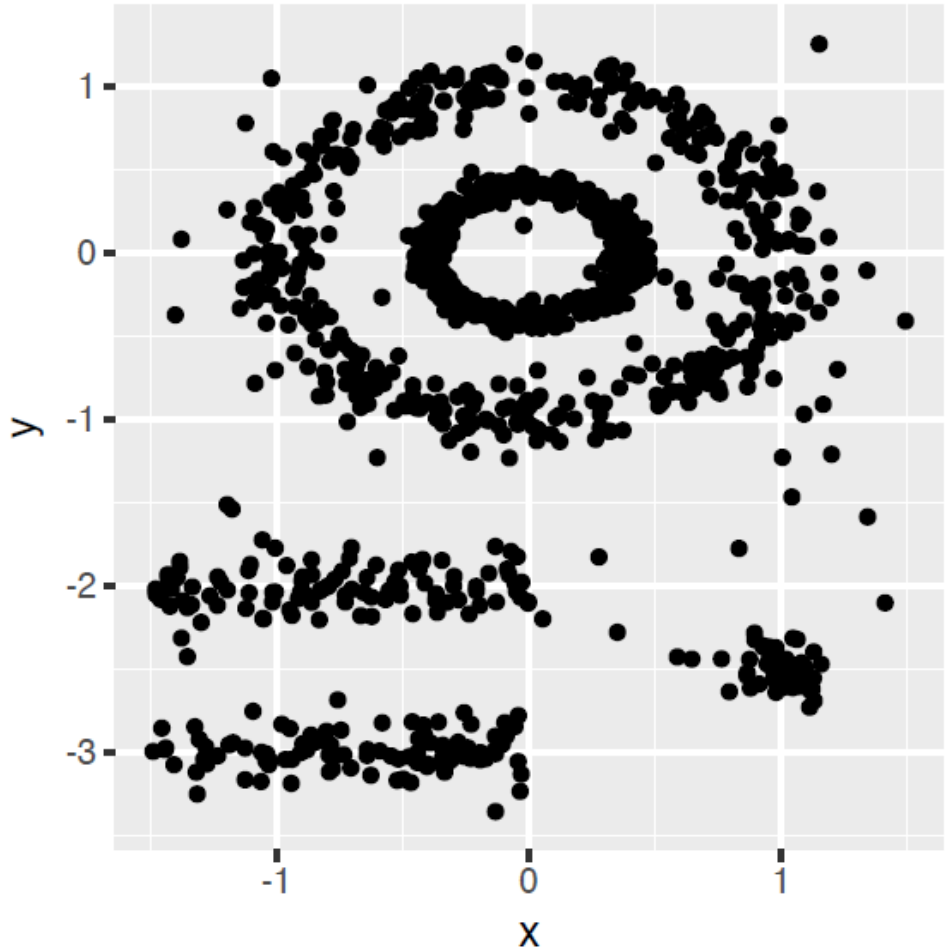
(From Ester et al. 1996)

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

¿Por qué DBSCAN?

- Los métodos de partición (K-means, PAM clustering) y cluster jerárquicos son adecuados para encontrar grupos de forma esférica o clústeres convexos.
- En otras palabras, funcionan bien solo para clusters compactos y bien separados.
- Pero son severamente afectados por la presencia de ruido y valores atípicos en los datos.
- Lamentablemente, los datos de la vida real pueden contener: i) grupos de formas arbitrarias (grupos de formas ovales, lineales y "S"); ii) muchos valores atípicos y ruido.
- La figura siguiente muestra un conjunto de datos que contiene clusters y valores atípicos / ruidos no convexos.
- Se usa el conjunto de datos simulados multishapes [en el paquete factoextra].

DBSCAN (Density-Based Spatial Clustering and Application with Noise)



DBSCAN (Density-Based Spatial Clustering and Application with Noise)

El gráfico anterior contiene 5 clústeres y valores atípicos, que incluyen:

- 2 clústeres ovales
- 2 clústeres lineales
- 1 grupo compacto

- Dados estos datos, el algoritmo k-means tiene dificultades para identificar estos grupos con formas arbitrarias.
- Para ilustrar esta situación, el siguiente código R calcula k-means en el conjunto de datos multishapes.
- La función fviz_cluster () [factoextra package] se usa para visualizar los clusters.

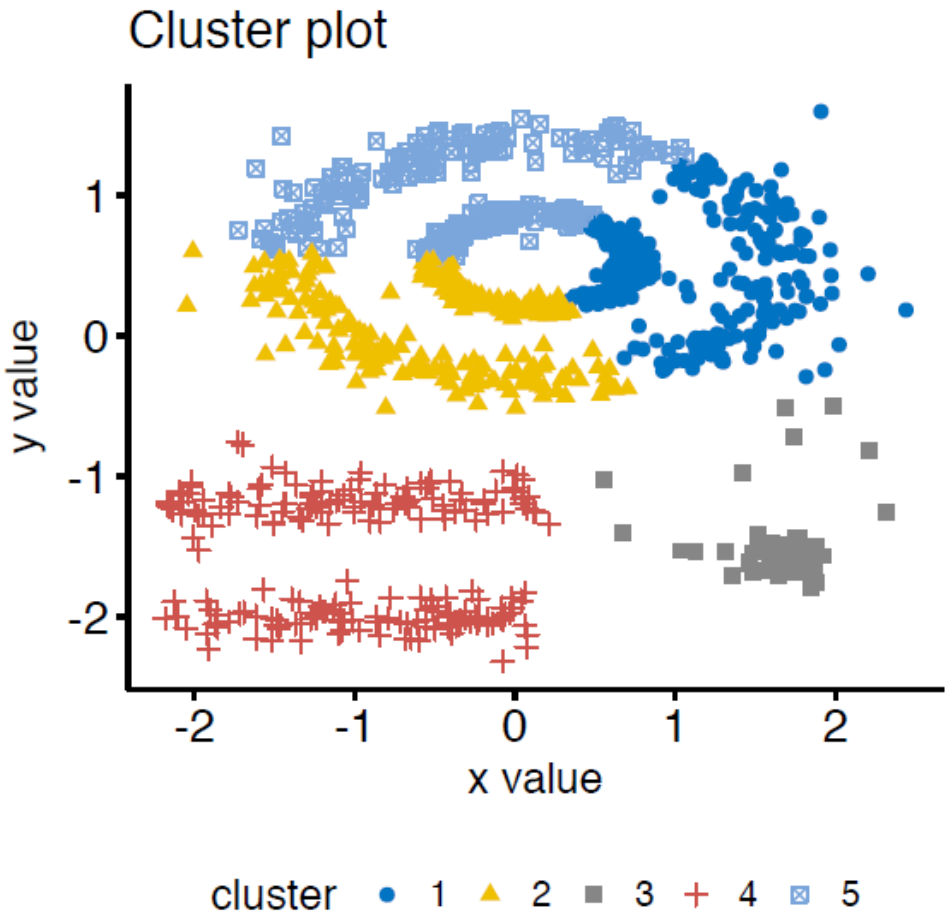
DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Primero, instale factoextra: install.packages ("factoextra"); luego calcule y visualice

k-means clustering usando el conjunto de datos multishapes:

```
library(factoextra)
data("multishapes")
df <- multishapes[, 1:2]
set.seed(123)
km.res <- kmeans(df, 5, nstart = 25)
fviz_cluster(km.res, df, geom = "point",
ellipse= FALSE, show.clust.cent = FALSE,
palette = "jco", ggtheme = theme_classic())
```

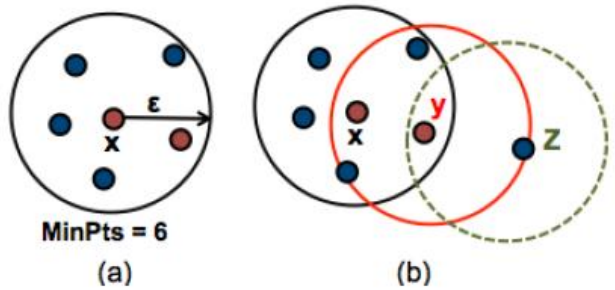
DBSCAN (Density-Based Spatial Clustering and Application with Noise)



Sabemos que hay 5 cinco grupos en los datos, pero se puede ver que el método k-means identificar incorrectamente los 5 grupos.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Algoritmo



- El objetivo es identificar regiones densas, que se pueden medir por el número de objetos cerca de un punto dado
- Se requieren dos parámetros importantes para DBSCAN: épsilon ("eps") y mínimo de puntos ("MinPts").
- El parámetro eps define el radio de vecindad alrededor un punto x.
- El parámetro MinPts es el número mínimo de vecinos dentro del radio "eps".
- Cualquier punto x en el conjunto de datos, con un conteo mayor o igual a MinPts, es marcado como un punto central.
- Decimos que x es el punto de frontera, si el número de sus vecinos es menos que MinPts, pero pertenece al k-neighborhood de algún punto central z.
- Finalmente, si un punto no es ni un núcleo ni un punto de frontera, entonces se llama punto de ruido.
- La figura muestra los diferentes tipos de puntos (núcleo, borde y puntos atípicos) usando MinPts = 6. Aquí x es un punto central porque los vecinos k(x) = 6, y es un borde porque tiene (y-vecinos) <MinPts, pero pertenece a la vecindad del punto central x. Finalmente, z es un punto de ruido.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Comenzamos definiendo 3 términos, necesarios para comprender el algoritmo DBSCAN:

- Densidad directa alcanzable: un punto "A" es directamente accesible desde otra densidad el punto "B" si: i) "A" está en la "vecindad" de "B" y ii) "B" es un punto central.
- Densidad alcanzable: un punto "A" es la densidad alcanzable desde "B" si hay un conjunto de los puntos centrales que van desde "B" a "A.
- Densidad conectada: dos puntos "A" y "B" están conectados a densidad si hay un punto central "C", tal que tanto "A" como "B" son densidades alcanzables desde "C".
- Un grupo basado en densidad se define como un grupo de puntos conectados por densidad.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

El algoritmo de clustering basado en densidad (DBSCAN) funciona de la siguiente manera:

1. Para cada punto xi, calcule la distancia entre xi y los otros puntos. Encuentra todos los puntos vecinos dentro de la distancia eps del punto inicial (xi). Cada punto, con un conteo vecino mayor o igual que MinPts, se marca como punto central o visitado.

2. Para cada punto central, si aún no está asignado a un clúster, crea un nuevo clúster. Encuentre recursivamente todos sus puntos conectados de densidad y asígnelos al mismo grupo como el punto central.

3. Itere a través de los puntos no visitados restantes en el conjunto de datos. Los puntos que no pertenecen a ningún clúster se tratan como valores atípicos o ruido.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Ventajas

1. A diferencia de K-means, DBSCAN no requiere que el usuario especifique el número de clusters que se generarán

2. DBSCAN puede encontrar cualquier forma de clúster. El clúster no tiene que ser circular.

3. DBSCAN puede identificar valores atípicos

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Estimación de parámetros

- MinPts: cuanto mayor sea el conjunto de datos, mayor será el valor de minPts, minPts se deben elegir al menos 3.

- Epsilon: El valor para epsilon se puede elegir utilizando un gráfico de distancia k, trazando el distancia al k = minPts vecino más cercano. Buenos valores de épsilon están donde la gráfica muestra una curva fuerte.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Ejemplo DBSCAN

- Aquí, usaremos el paquete R “fpc” para calcular DBSCAN. También es posible usar el paquete “dbscan”, que proporciona una reimplementación más rápida del algoritmo DBSCAN en comparación con el paquete fpc.

- También usaremos el paquete factoextra para visualizar clusters.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Primero, instale los paquetes de :

install.packages("fpc")

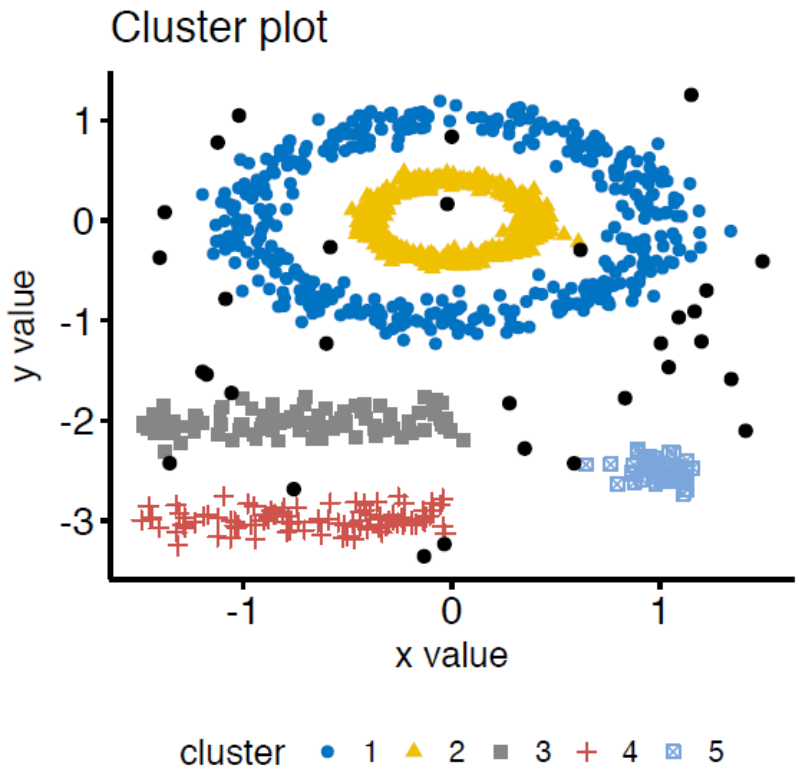
install.packages("dbscan")

install.packages("factoextra")

El siguiente código R estima y visualiza DBSCAN usando el conjunto de datos multishapes [paquete factoextra R]:

```
# Abrimos los datos multishapes
data("multishapes", package = "factoextra")
df <- multishapes[, 1:2]
# Estimamos DBSCAN
library("fpc")
set.seed(123)
db <- fpc::dbscan(df, eps = 0.15, MinPts = 5)
```

DBSCAN (Density-Based Spatial Clustering and Application with Noise)




```
# Graficamos los resultados
        library("factoextra")
        fviz_cluster(db, data = df, stand = FALSE,
        ellipse = FALSE, show.clust.cent = FALSE,
        geom = "point",palette = "jco", ggtheme = theme_classic())
```

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Tenga en cuenta que la función fviz_cluster () usa diferentes símbolos de puntos para los puntos centrales (es decir, puntos semilla) y puntos fronterizos.

Los puntos negros corresponden a valores atípicos.

Se puede ver que DBSCAN funciona mejor para estos conjuntos de datos y puede identificar el conjunto correcto de clusters en comparación con los algoritmos de k-medias.

El resultado de la función fpc :: dbscan () se puede mostrar de la siguiente manera:

- **print**(db)

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

dbscan Pts=1100 MinPts=5 eps=0.15

0 1 2 3 4 5

border 31 24 1 5 7 1

seed 0 386 404 99 92 50

total 31 410 405 104 99 51

- En la tabla anterior, los nombres de columna son número de clúster. El clúster 0 corresponde a valores atípicos (puntos negros en la trama DBSCAN).
- La función print.dbscan () muestra una estadística de la cantidad de puntos que pertenecen a los conglomerados que son semillas y puntos fronterizos.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)

```
# Miembros del Cluster, los outliers son codificados como 0
# Muestra aleatoria
db$cluster[sample(1:1089, 20)]
## [1] 1 3 2 4 3 1 2 4 2 2 2 2 2 1 4 1 1 1 0
```

El algoritmo DBSCAN requiere que los usuarios especifiquen los valores de eps óptimos y el parámetro MinPts. En el código R anterior, usamos eps = 0.15 y MinPts = 5.

Una limitación de DBSCAN es que es sensible a la elección de epsilon, en particular si los conglomerados tienen diferente densidades

Si épsilon es demasiado pequeño, los clústeres más dispersos se definirán como ruido.

Si es demasiado grande, los grupos más densos pueden fusionarse. Esto implica que, si hay grupos con diferentes densidades locales, entonces un solo valor puede no ser suficiente.

Una pregunta natural es: ¿Cómo definir el valor óptimo de epsilon?

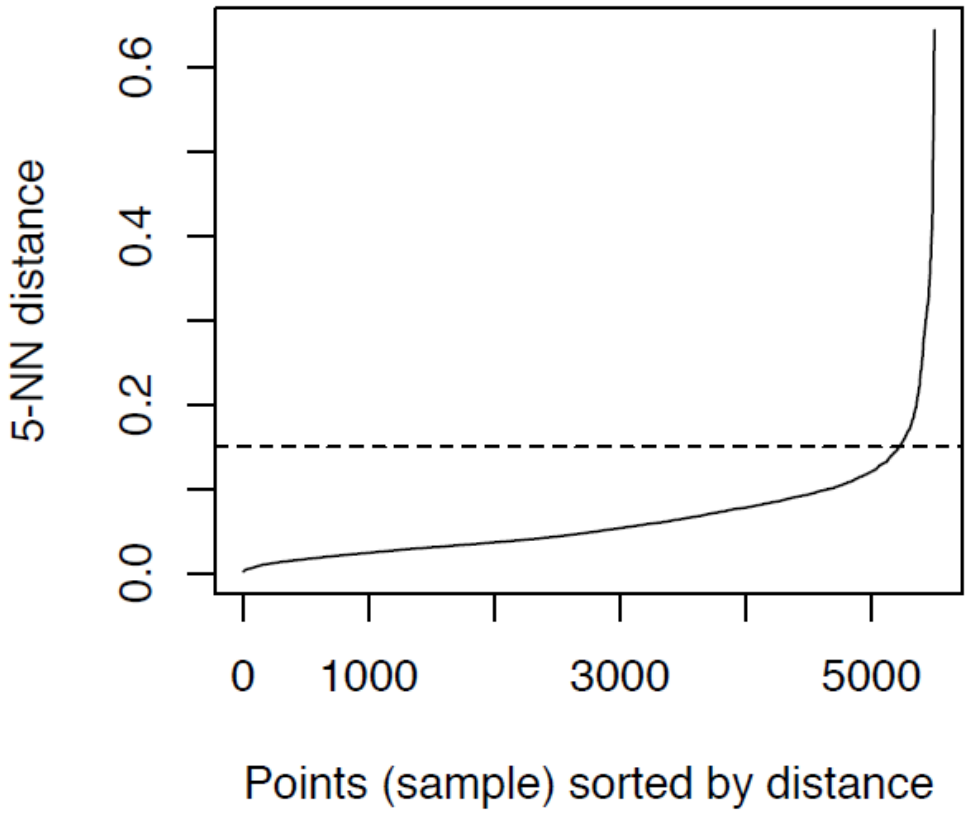
DBSCAN (Density-Based Spatial Clustering and Application with Noise)

Método para determinar el valor de epsilon óptimo:

- *El método propuesto aquí consiste en calcular las distancias del vecino k-más cercano en una matriz de puntos.*
- *La idea es calcular, el promedio de las distancias de cada punto a su k vecinos más cercano*
- *El valor de k será especificado por el usuario y corresponde a MinPts.*
- *A continuación, estas distancias k se trazan en orden ascendente. El objetivo es determinar la "rodilla", que corresponde al parámetro épsilon óptimo.*
- *Una rodilla corresponde a un umbral donde se produce un cambio brusco a lo largo de la distancia k en la curva.*
- La función kNNdistplot () [en el paquete dbscan] se puede usar para dibujar la distancia k:

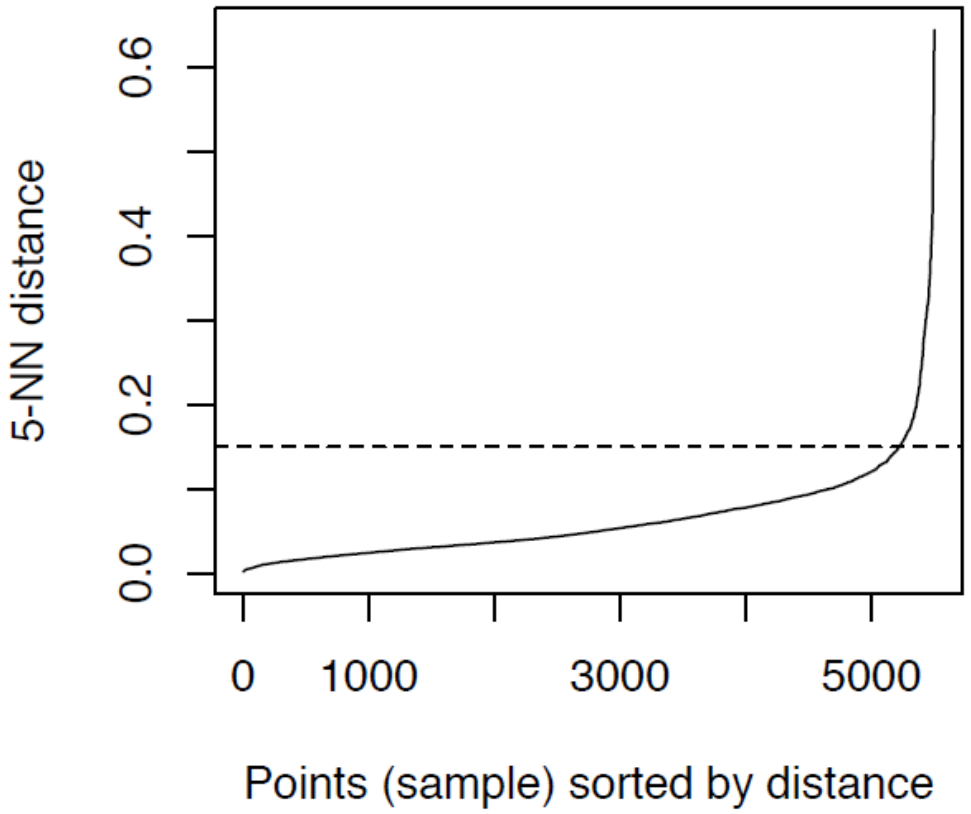
```
dbscan::kNNdistplot(df, k = 5)
abline(h = 0.15, lty = 2)
```

DBSCAN (Density-Based Spatial Clustering and Application with Noise)



Se puede ver que el valor de epsilon óptimo está alrededor de una distancia de 0.15.

DBSCAN (Density-Based Spatial Clustering and Application with Noise)



Se puede ver que el valor de eps óptimo está alrededor de una distancia de 0.15.

Predicciones de conglomerados con el algoritmo DBSCAN

La función predict.dbscan (object, data, newdata) [en el paquete fpc] se puede usar para predecir los clusters para nuevos datos.

Para más detalles, lea la documentación (? predict.dbscan).