Tutorial DBSCAN

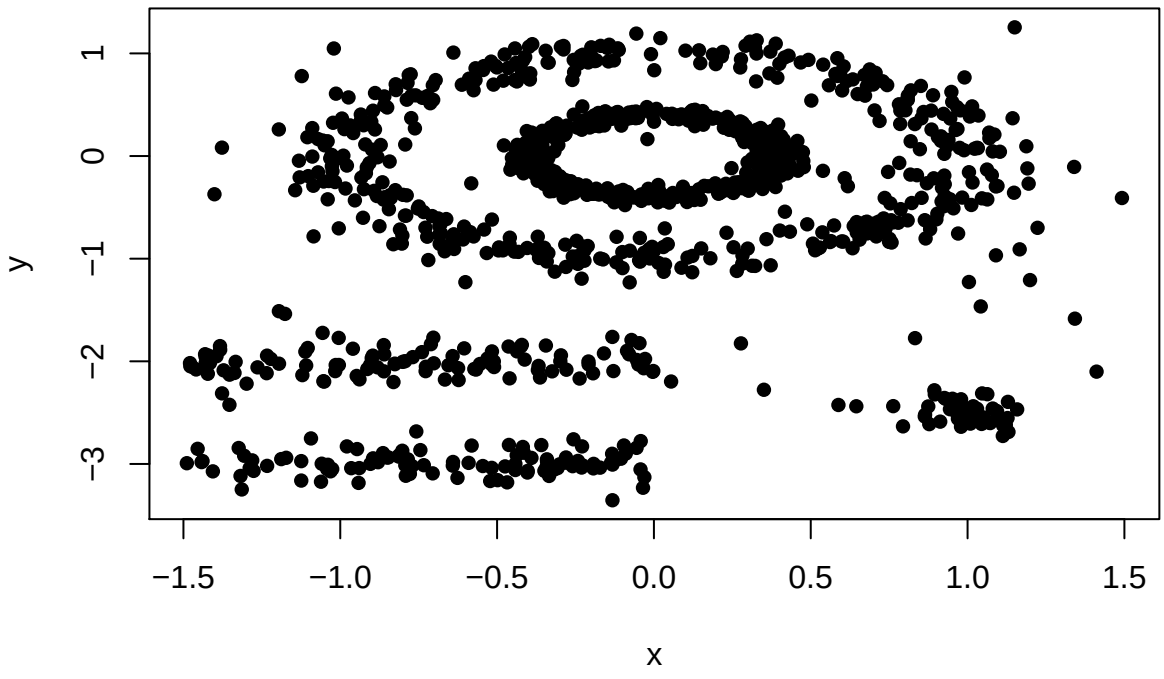
Tutorial ini adalah bagian dari mata kuliah Fundamen Sains Data, Informatika, UII.

Pada tutorial kali ini kita akan membanding hasil dari K-Means dan DBSCAN pada sebuah data set yang memiliki *odd shape* atau bentuk yang tidak biasa. Di sini kita akan menggunakan packge `factoextra`. Kita akan menggunakan data set `multishapes`, yang berisi 1100 observations dengan dua variabel (kita mengambil dua kolom pertama pada baris ke-3 dari script di bawah ini) yang merepresentasikan berbagai macam bentuk.

```
1 library(factoextra)
2 data("multishapes")
3 df <- multishapes[, 1:2]
```

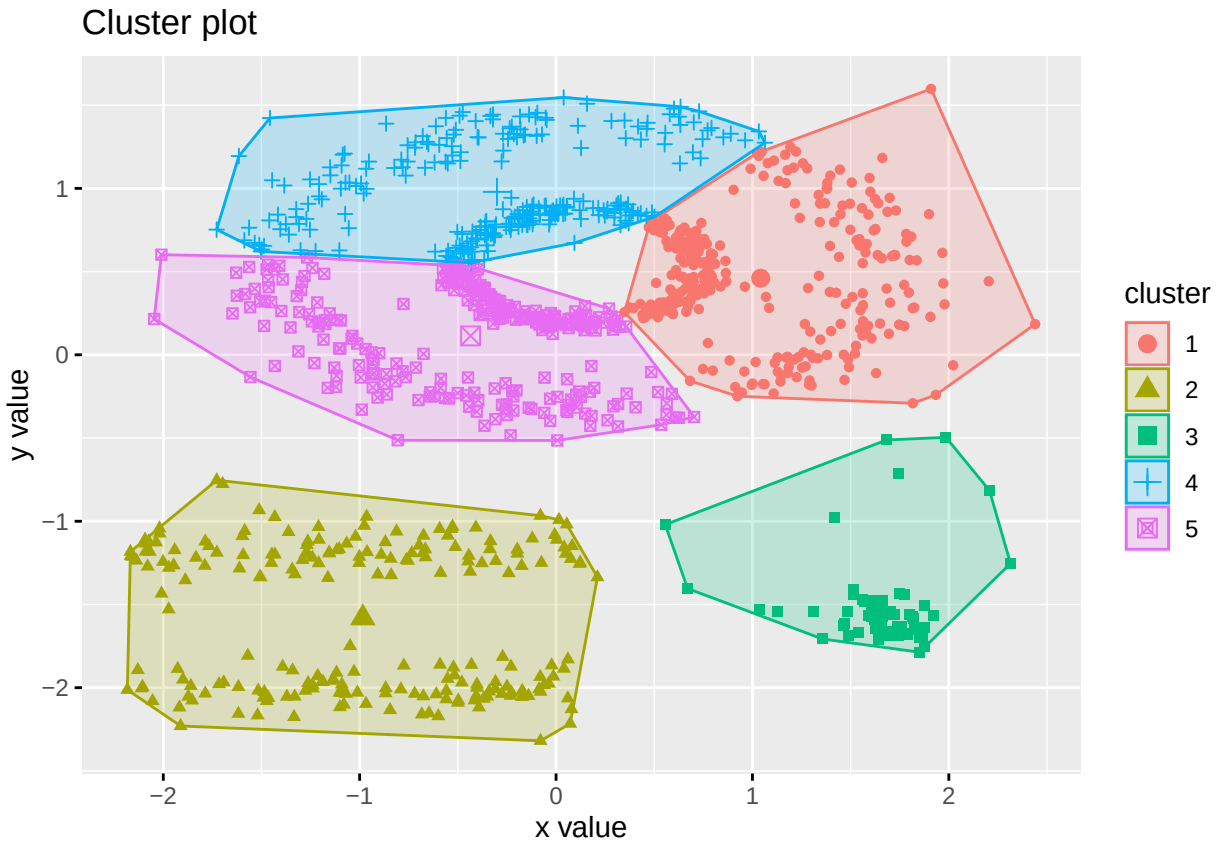
Selanjutya kita plot data set tersebut, untuk memperoleh gambaran. Maksud dari `pch=16` adalah tipe simbol ke-16 untuk sebuah data point; di sini, kita memilih simbol sebuah lingkaran solid untuk sebuah data point.

```
plot(df, pch=16)
```



Berdasarkan plot data set di atas, secara intuisi kita melihat ada sekitar 5 cluster. Mari kita coba aplikasi K-Means dengan 5 cluster.

```
set.seed(123)
km.res <- kmeans(df, 5, nstart = 25)
fviz_cluster(km.res, df, frame = FALSE, geom = "point")
```

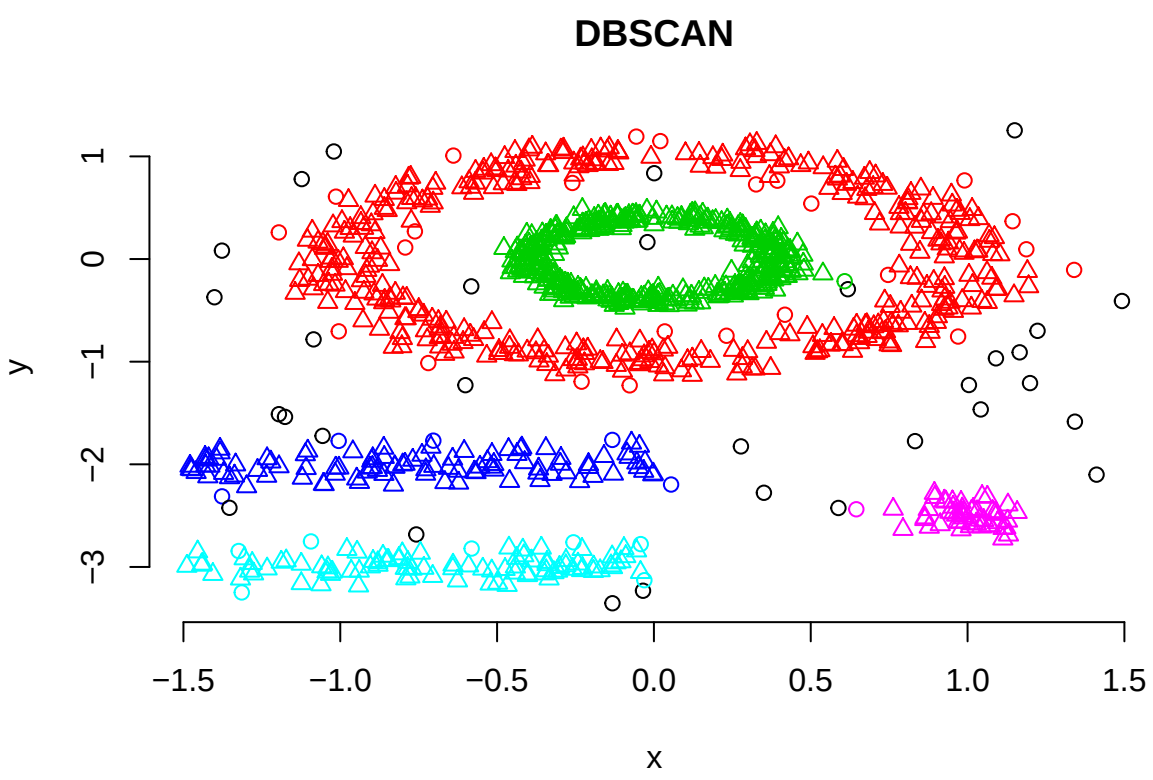


Dapat kita lihat, K-Means memberikan 5 cluster yang berbeda dengan apa yang kita bayangkan. Di sini, adalah sebuah contoh di mana sebaiknya kita tidak menggunakan K-Means pada data yang memiliki bentuk sebaran yang tidak lazim.

Selanjutnya, mari kita aplikasikan DBSCAN pada data yang sama. Ingat pada DBSCAN, kita tidak perlu menentukan jumlah cluster di awal.

Aplikasi DBSCAN Di sini kita akan menggunakan fungsi `dbscan()` dari package `fpc`, dengan parameter $\epsilon = 5$ dan $minPts = 5$.

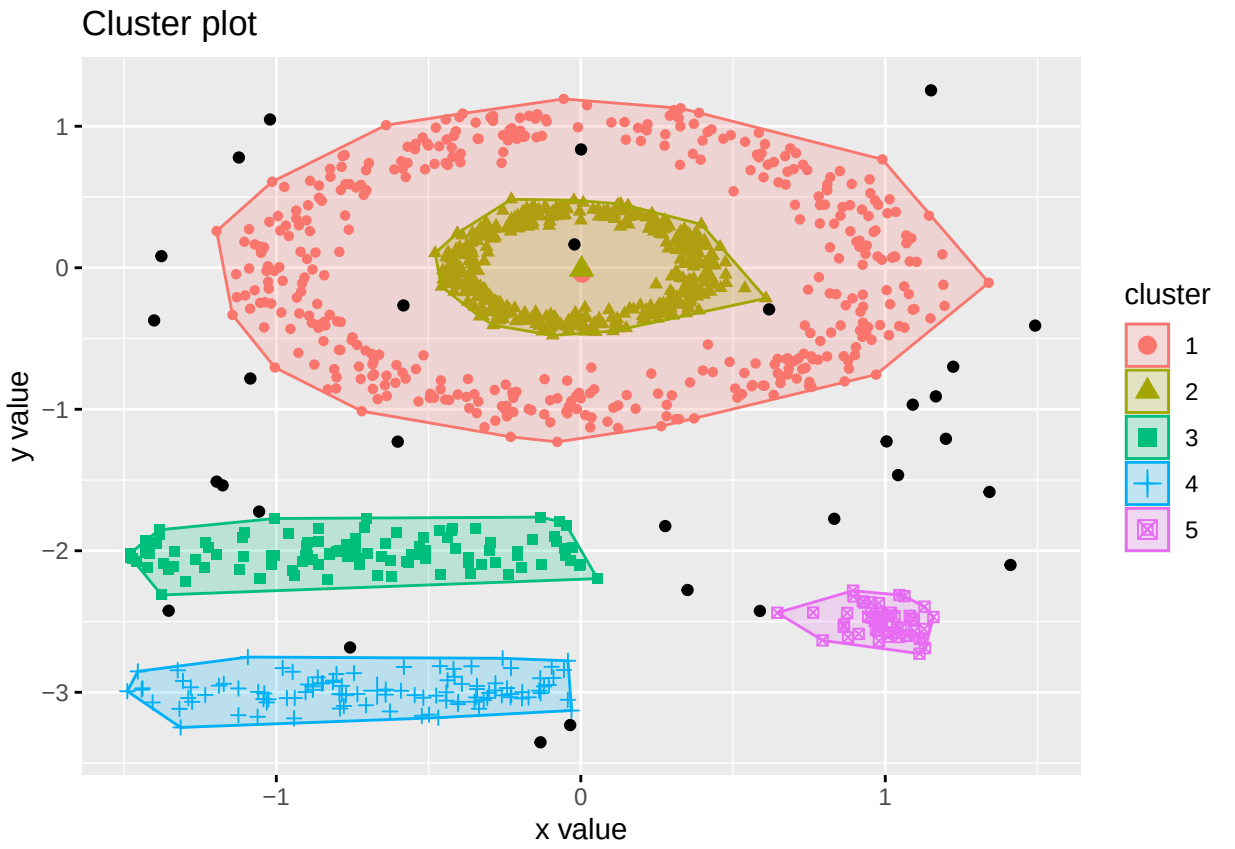
```
library("fpc")
set.seed(123)
db <- dbscan(df, eps = 0.15, MinPts = 5)
plot(db, df, main = "DBSCAN", frame = FALSE)
```



Dari hasil di atas, dapat kita lihat jika DBSCAN mampu menemukan 5 cluster yang kita bayangkan secara intuisi beserta data point yang sekiranya termasuk noise (simbol lingkaran tanpa warna).

Sebagai alternatif untuk plot, kita juga dapat menggunakan sebuah fungsi untuk plot cluster dari `factoexra`.

```
library("factoextra")
fviz_cluster(db, df, stand = FALSE, frame = FALSE, geom = "point")
```



Untuk melihat lebih detil dari hasil DBSCAN, kita dapat melihat lewat script,

```
print(db)
```

```
## 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
```

Kolom pada tabel di atas merepresentasikan cluster (dengan kolom 0 merepresentasikan noise). Baris `border` mengindikasikan jumlah *border point*, `seed` jumlah *core point*, `total` jumlah total (core + border) data point pada cluster tersebut.

Dapat kita lihat cluster 1 memiliki jumlah anggota terbanyak (410) dan cluster 5 jumlah anggota paling sedikit (51).

Untuk melihat indeks cluster dari setiap data point,

```
db$cluster
```

```
##   [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
##  [38] 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 0 1 1 1
##  [75] 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
## [112] 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
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
##  [186] 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
##  [223] 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
##  [260] 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
##  [297] 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
##  [334] 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
##  [371] 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 2 2 2 2 2 2 2
##  [408] 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
##  [445] 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
##  [482] 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
##  [519] 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
##  [556] 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
##  [593] 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
##  [630] 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
##  [667] 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
##  [704] 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
##  [741] 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
##  [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [815] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [852] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [889] 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 4 4 4 4 4 4 4 4 4 0
##  [926] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [963] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [1000] 4 1 5 1 2 0 0 0 0 0 1 0 2 0 2 0 1 1 4 0 0 3 0 0 2 0 0 1 0 0 0 0 1 2 0 0 0
## [1037] 1 1 1 1 0 1 3 0 3 0 0 0 0 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1074] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

Dari informasi di atas, kita dapat lihat setiap data point (dari 1 hingga ke 1100) masuk ke dalam cluster berapa; jika 0, dianggap sebagai noise.

Pendekatan untuk menentukan parameter ϵ dan $minPts$ Dalam menentukan $minPts$, ada anjuran yang berangkat dari pengalaman, salah satunya adalah dihitung dengan rumus $minPts = D + 1$, dengan D adalah jumlah variabel dalam sebuah data set yang akan dianalisis menggunakan DBSCAN.

Dalam menentukan ϵ, kita dapat menggunakan k-distance graph. Metode ini melihat jarak setiap data point dari k-th nearest neighbor (data point terdekat). Jarak-jarak yang telah dihitung, diurutkan dari besar ke kecil; kemudian kita tentukan threshold yang menjadi nilai ϵ pada bagian "lembah" atau bagian dari kurva di mana kurva mengalami pembelokan (*lembah*). Di sini kita akan menggunakan fungsi `kNNdistplot` dari package `dbscan`.

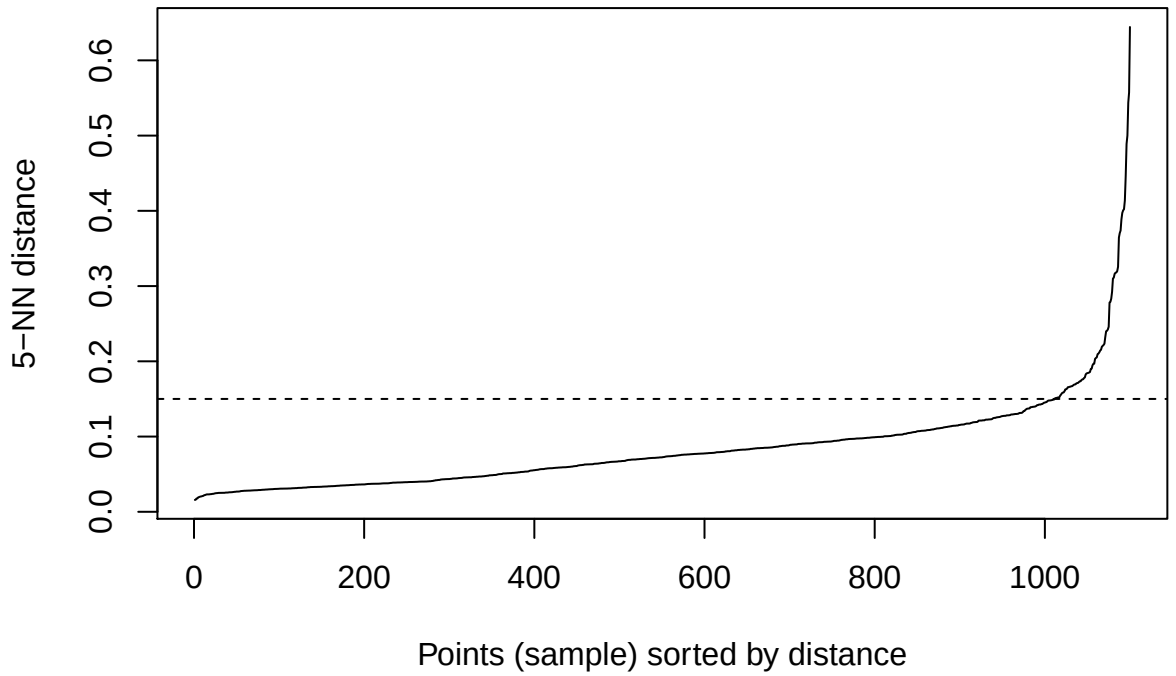
Masih menggunakan data set yang sama (`df`) dengan contoh di atas, kita ingin melihat k-distance graph dengan k=5.

```
library(dbscan)
```

```
##
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:fpc':
##
##     dbscan
```

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



Dari plot yang di dapat, kita dapat lihat bahwa lembah pada kurva adalah ketika jarak = 0.15. Nilai inilah yang kita jadikan sebagai nilai ϵ.

Alternatif lain dalam menentukan $minPts$ adalah dengan menghitung rata-rata dari nilai k-distance dari semua data point. Nilai rata-rata tersebut

Latihan Lakukan komputasi DBSCAN yang sama seperti di atas atau pada data lain yang Anda pilih sendiri, kemudian lihat perbedaan ketika

1. kita mengubah nilai ϵ ketika nilai $minPts = 5$ tetap
2. kita mengubah nilai $minPts$ ketika nilai $\epsilon = 0.15$ tetap

Jelaskan secara singkat, apa yang terjadi jika kita set $minPts$ semakin besar atau semakin kecil, dan yang terjadi ketika ϵ semakin besar atau kecil.

Referensi http://www.sthda.com/english/wiki/wiki.php?id_contents=7940