

Object detection with the DBScan algorithm

Détection d'objets avec l'algorithme DBScan

Exercice de programmation

Partie 2

CSI2510 Algorithmes et Structure de Données

Automne 2022

Professeur : Robert Laganier

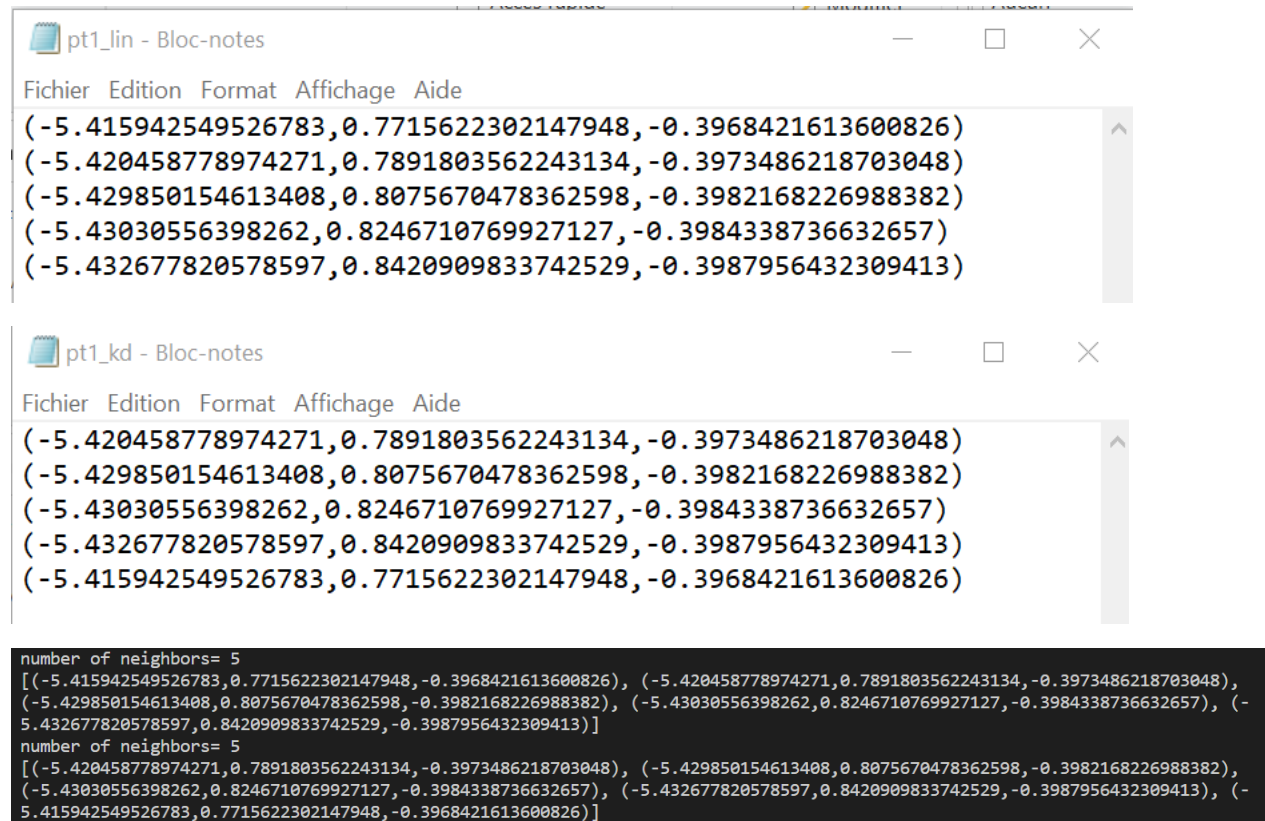
Prénom : Chada

Nom : Bendriss

Numéro d'étudiant : 300266679

Exp1

Point 1 :



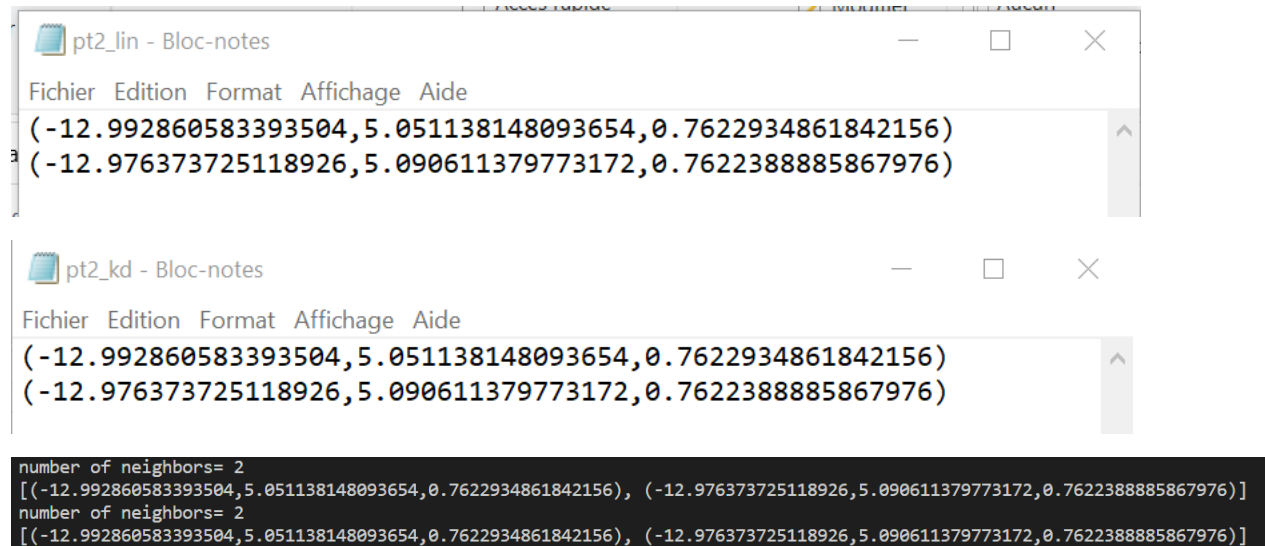
```
pt1_lin - Bloc-notes
Fichier Edition Format Affichage Aide
(-5.415942549526783,0.7715622302147948,-0.3968421613600826)
(-5.420458778974271,0.7891803562243134,-0.3973486218703048)
(-5.429850154613408,0.8075670478362598,-0.3982168226988382)
(-5.43030556398262,0.8246710769927127,-0.3984338736632657)
(-5.432677820578597,0.8420909833742529,-0.3987956432309413)

pt1_kd - Bloc-notes
Fichier Edition Format Affichage Aide
(-5.420458778974271,0.7891803562243134,-0.3973486218703048)
(-5.429850154613408,0.8075670478362598,-0.3982168226988382)
(-5.43030556398262,0.8246710769927127,-0.3984338736632657)
(-5.432677820578597,0.8420909833742529,-0.3987956432309413)
(-5.415942549526783,0.7715622302147948,-0.3968421613600826)

number of neighbors= 5
[(-5.415942549526783,0.7715622302147948,-0.3968421613600826), (-5.420458778974271,0.7891803562243134,-0.3973486218703048),
(-5.429850154613408,0.8075670478362598,-0.3982168226988382), (-5.43030556398262,0.8246710769927127,-0.3984338736632657), (-
5.432677820578597,0.8420909833742529,-0.3987956432309413)]
number of neighbors= 5
[(-5.420458778974271,0.7891803562243134,-0.3973486218703048), (-5.429850154613408,0.8075670478362598,-0.3982168226988382),
(-5.43030556398262,0.8246710769927127,-0.3984338736632657), (-5.432677820578597,0.8420909833742529,-0.3987956432309413), (-
5.415942549526783,0.7715622302147948,-0.3968421613600826)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

Point 2 :



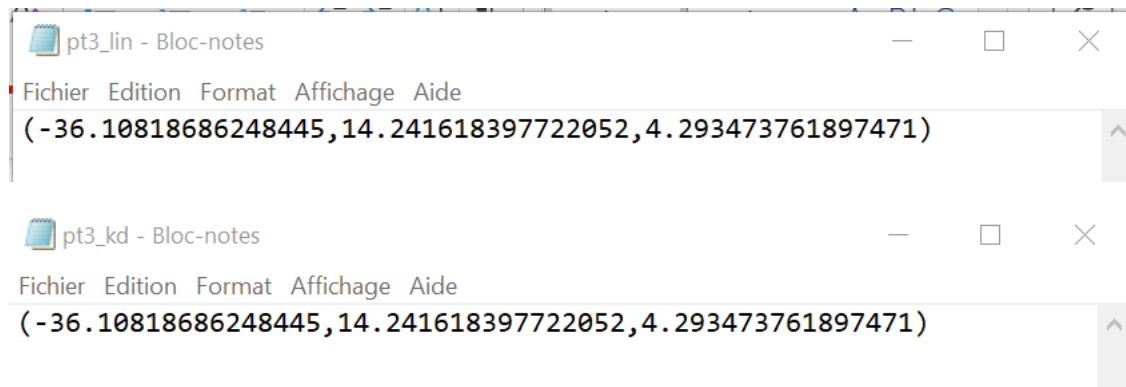
```
pt2_lin - Bloc-notes
Fichier Edition Format Affichage Aide
(-12.992860583393504,5.051138148093654,0.7622934861842156)
(-12.976373725118926,5.090611379773172,0.7622388885867976)

pt2_kd - Bloc-notes
Fichier Edition Format Affichage Aide
(-12.992860583393504,5.051138148093654,0.7622934861842156)
(-12.976373725118926,5.090611379773172,0.7622388885867976)

number of neighbors= 2
[(-12.992860583393504,5.051138148093654,0.7622934861842156), (-12.976373725118926,5.090611379773172,0.7622388885867976)]
number of neighbors= 2
[(-12.992860583393504,5.051138148093654,0.7622934861842156), (-12.976373725118926,5.090611379773172,0.7622388885867976)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

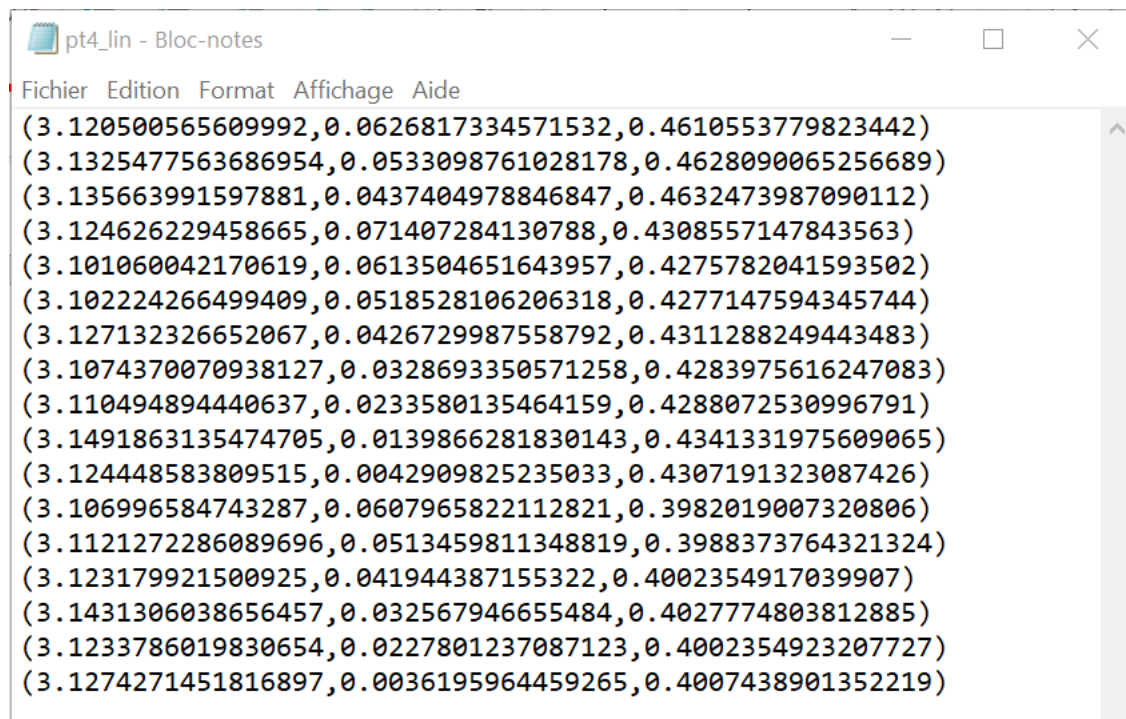
Point 3 :



```
number of neighbors= 1  
[(-36.10818686248445,14.241618397722052,4.293473761897471)]  
number of neighbors= 1  
[(-36.10818686248445,14.241618397722052,4.293473761897471)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

Point 4 :



```
pt4_kd - Bloc-notes
Fichier Edition Format Affichage Aide
(3.106996584743287,0.0607965822112821,0.3982019007320806)
(3.1121272286089696,0.0513459811348819,0.3988373764321324)
(3.123179921500925,0.041944387155322,0.4002354917039907)
(3.1233786019830654,0.0227801237087123,0.4002354923207727)
(3.1274271451816897,0.0036195964459265,0.4007438901352219)
(3.1431306038656457,0.032567946655484,0.4027774803812885)
(3.124626229458665,0.071407284130788,0.4308557147843563)
(3.101060042170619,0.0613504651643957,0.4275782041593502)
(3.102224266499409,0.0518528106206318,0.4277147594345744)
(3.1074370070938127,0.0328693350571258,0.4283975616247083)
(3.110494894440637,0.0233580135464159,0.4288072530996791)
(3.124448583809515,0.0042909825235033,0.4307191323087426)
(3.127132326652067,0.0426729987558792,0.4311288249443483)
(3.1491863135474705,0.0139866281830143,0.4341331975609065)
(3.120500565609992,0.0626817334571532,0.4610553779823442)
(3.1325477563686954,0.0533098761028178,0.4628090065256689)
(3.135663991597881,0.0437404978846847,0.4632473987090112)
```

```
number of neighbors= 17
[(3.120500565609992,0.0626817334571532,0.4610553779823442), (3.1325477563686954,0.0533098761028178,0.4628090065256689), (3.135663991597881,0.0437404978846847,0.4632473987090112), (3.124626229458665,0.071407284130788,0.4308557147843563), (3.101060042170619,0.0613504651643957,0.4275782041593502), (3.102224266499409,0.0518528106206318,0.4277147594345744), (3.127132326652067,0.0426729987558792,0.4311288249443483), (3.1074370070938127,0.0328693350571258,0.4283975616247083), (3.110494894440637,0.0233580135464159,0.4288072530996791), (3.1491863135474705,0.0139866281830143,0.4341331975609065), (3.124448583809515,0.0042909825235033,0.4307191323087426), (3.106996584743287,0.0607965822112821,0.3982019007320806), (3.1121272286089696,0.0513459811348819,0.3988373764321324), (3.123179921500925,0.041944387155322,0.4002354917039907), (3.1431306038656457,0.032567946655484,0.4027774803812885), (3.1274271451816897,0.0036195964459265,0.4007438901352219)]
number of neighbors= 17
[(3.106996584743287,0.0607965822112821,0.3982019007320806), (3.1121272286089696,0.0513459811348819,0.3988373764321324), (3.123179921500925,0.041944387155322,0.4002354917039907), (3.1233786019830654,0.0227801237087123,0.4002354923207727), (3.1274271451816897,0.0036195964459265,0.4007438901352219), (3.1431306038656457,0.032567946655484,0.4027774803812885), (3.124626229458665,0.071407284130788,0.4308557147843563), (3.101060042170619,0.0613504651643957,0.4275782041593502), (3.102224266499409,0.0518528106206318,0.4277147594345744), (3.1074370070938127,0.0328693350571258,0.4283975616247083), (3.110494894440637,0.0233580135464159,0.4288072530996791), (3.124448583809515,0.0042909825235033,0.4307191323087426), (3.127132326652067,0.0426729987558792,0.4311288249443483), (3.1491863135474705,0.0139866281830143,0.4341331975609065), (3.120500565609992,0.0626817334571532,0.4610553779823442), (3.1325477563686954,0.0533098761028178,0.4628090065256689), (3.135663991597881,0.0437404978846847,0.4632473987090112)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

Point 5 :

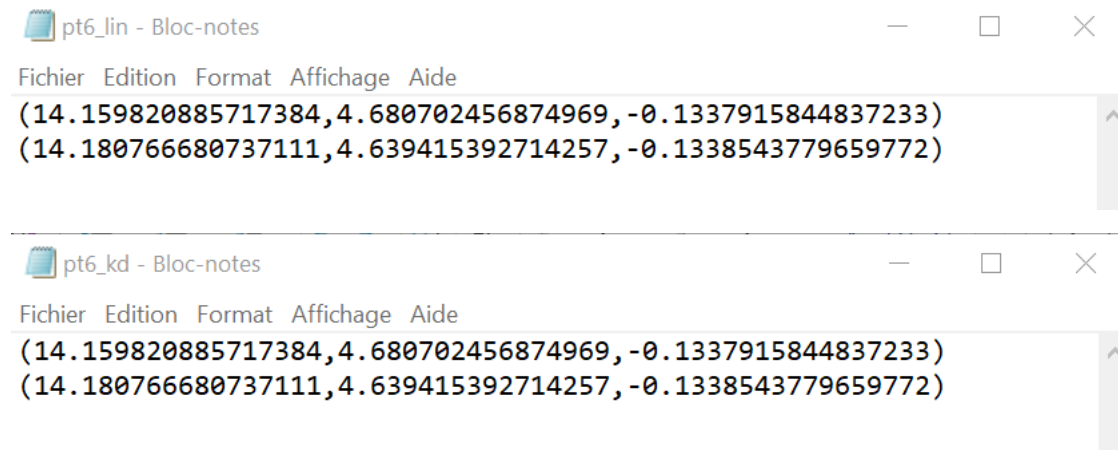
```
pt5_lin - Bloc-notes
Fichier Edition Format Affichage Aide
(11.597053489523276,3.032865894391464,1.8696242228185609)
(11.580473933555549,2.9906018684790574,1.8654633424019456)

pt5_kd - Bloc-notes
Fichier Edition Format Affichage Aide
(11.597053489523276,3.032865894391464,1.8696242228185609)
(11.580473933555549,2.9906018684790574,1.8654633424019456)

number of neighbors= 2
[(11.597053489523276,3.032865894391464,1.8696242228185609), (11.580473933555549,2.9906018684790574,1.8654633424019456)]
number of neighbors= 2
[(11.597053489523276,3.032865894391464,1.8696242228185609), (11.580473933555549,2.9906018684790574,1.8654633424019456)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

Point 6



```
number of neighbors= 2  
[(14.159820885717384,4.680702456874969,-0.1337915844837233), (14.180766680737111,4.639415392714257,-0.1338543779659772)]  
number of neighbors= 2  
[(14.159820885717384,4.680702456874969,-0.1337915844837233), (14.180766680737111,4.639415392714257,-0.1338543779659772)]
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes

D'après l'analyse ci-dessus, on peut donc conclure que les deux méthodes donnent le même résultat.

Exp2

1^{er} fichier

```
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 lin 0.5 Point_Cloud_1.csv 10
Average time: 270761.4579817752
time required for rangeQuery in nanoseconds: 802266200
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 kd 0.5 Point_Cloud_1.csv 10
Average time: 41682.92271346608
time required for rangeQueryKD in nanoseconds: 123506500
```

Pour le 1^{er} fichier, on peut donc conclure que la méthode kd est en moyenne 6,49 fois plus rapide que la méthode lin

2eme fichier

```
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 lin 0.5 Point_Cloud_2.csv 10
Average time: 289721.9205037387
time required for rangeQuery in nanoseconds: 1472366800
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 kd 0.5 Point_Cloud_2.csv 10
Average time: 82114.04958677686
time required for rangeQueryKD in nanoseconds: 417303600
```

Pour le 2^{-ème} fichier, on peut donc conclure que la méthode kd est en moyenne 5,52 fois plus rapide que la méthode lin

3eme fichier

```
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 lin 0.5 Point_Cloud_3.csv 10
Average time: 329656.3136907399
time required for rangeQuery in nanoseconds: 1488068600
PS C:\Users\chada\Desktop\uotttaw\2 Fall 2022\CSI 2510\Lab\CBProg2 - Copie\Prog2> java Exp2 kd 0.5 Point_Cloud_3.csv 10
Average time: 65712.02924235711
time required for rangeQueryKD in nanoseconds: 296624100
```

Pour le 3^{-ème} fichier, on peut donc conclure que la méthode kd est en moyenne 5 fois plus rapide que la méthode lin

Conclusion, les résultats expérimentaux conformes avec les résultats théoriques. Donc l'arbre k-d donne de meilleurs résultats.

Exp3

Le code est implémentée dans la méthode main de la classe DBScan. Dans la classe DBScan se trouve les deux méthodes findClusters() et findClustersKD() pour lin et kd respectivement.

```
java DBScan Point_Cloud_1.csv 0.96 4
range query 7969
java DBScan Point_Cloud_1.csv 0.96 4
range query Kd 3823

java DBScan Point_Cloud_2.csv 0.96 4
range query 25316

java DBScan Point_Cloud_2.csv 0.96 4
range query Kd 10947

java DBScan Point_Cloud_3.csv 0.96 4
range query 11505

java DBScan Point_Cloud_3.csv 0.96 4
range query Kd 5689
```

```
java DBScan Point_Cloud_1.csv 0.96 4
number of clusters : 36
The size of all clusters found, from the largest one to the smallest one
9800
8033
2484
1730
1705
1445
725
655
583
342
328
324
268
215
192
182
138
104
93
55
25
22
20
18
16
15
14
14
13
13
13
8
5
4
4
We can now conclude that the noise has : 4 points
range query 9339
```

```
java DBScan Point_CloudKD_1.csv 0.96 4
number of clusters : 36
9800
8033
2484
1730
1705
1445
725
655
583
342
328
324
268
215
192
182
138
104
93
55
25
22
20
18
16
15
14
14
13
13
13
8
5
4
4
We can now conclude that the noise has : 4 points
range query Kd 5747
```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes. Par conséquent, la méthode kd est 1,62 plus rapide que la méthode lin. Conclusion, les résultats expérimentaux conformes avec les résultats théoriques.


```

java DBScan Point_CloudLIN_2.csv 0.96 4
number of clusters : 34
28939
13258
2949
1691
860
698
665
437
355
199
125
77
57
55
53
43
39
30
27
25
24
24
21
20
19
19
18
17
17
16
15
10
10
8
We can now conclude that the noise has : 8 points
range query 27828

```

```

java DBScan Point_CloudKD_2.csv 0.96 4
number of clusters : 34
The size of all clusters found, from the largest one to the smallest one
28939
13258
2949
1691
860
698
665
437
355
199
125
77
57
55
53
43
39
30
27
25
24
24
21
20
19
19
18
17
17
16
15
10
10
8
We can now conclude that the noise has : 8 points
range query Kd 13800

```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes. Par conséquent, la méthode kd est en moyenne 2,01 plus rapide que la méthode lin. Conclusion, les résultats expérimentaux conformes avec les résultats théoriques.


```

java DBScan Point_CloudLIN_3.csv 0.96 4
number of clusters : 46
The size of all clusters found, from the largest one to the smallest one
13401
10525
5253
5181
2978
2596
1173
862
363
312
297
240
239
237
209
192
137
93
92
75
65
62
61
40
39
39
32
31
28
28
23
23
22
20
20
19
17
17
16
15
14
13
12
12
11
6
We can now conclude that the noise has : 6 points
range query 22401

```

```

java DBScan Point_CloudKD_3.csv 0.96 4
number of clusters : 46
The size of all clusters found, from the largest one to the smallest one
13401
10525
5253
5181
2978
2596
1173
862
363
312
297
240
239
237
209
192
137
93
92
75
65
62
61
40
39
39
32
31
28
28
23
23
22
20
20
19
17
17
16
15
14
13
12
12
11
6
We can now conclude that the noise has : 6 points
range query Kd 11091

```

On remarque que le nombre de clusters de la méthode « lin » est le même que celui de la méthode « kd ». De plus, la liste des points voisins est la même pour les deux méthodes. Par conséquent, la méthode kd est en moyenne 2,01 plus rapide que la méthode lin. Conclusion, les résultats expérimentaux conformes avec les résultats théoriques.

```

Conclusion
time lin for file 1 with lin method 9339
time lin for file 1 with kd method 5747
time lin for file 2 with lin method 27828
time lin for file 2 with kd method 13800
time lin for file 3 with lin method 22401
time lin for file 3 with kd method 11091

```

Conclusion, les résultats expérimentaux conformes avec les résultats théoriques.