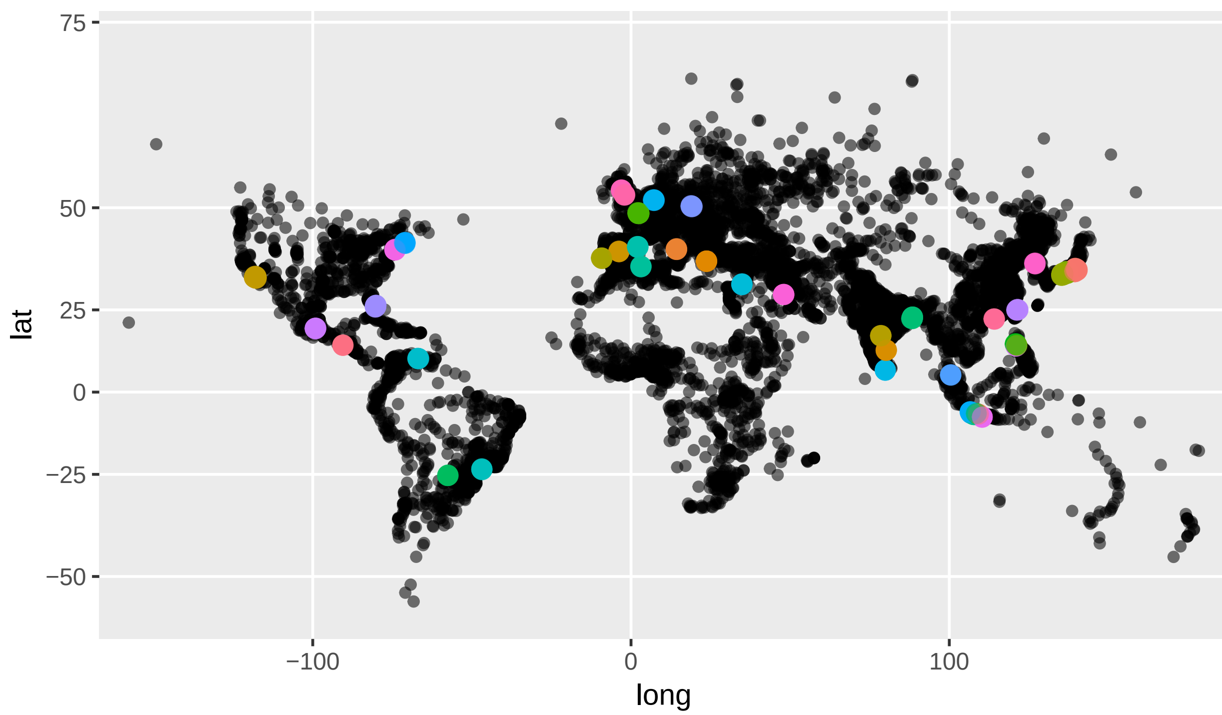
Exercise Sheet 5

1. Use DBSCAN to find clusters from nearby large cities around the world (metropolitan areas). A city with at least 50,000 inhabitants is considered large. The ϵ-neighborhood of a city contains all adjacent cities with a Euclidean distance of 0.15 or less in latitude and longitude. A city is considered a core object of a conurbation if at least 8 cities are located in its ϵ-neighborhood. For clustering, use the `maps::world.cities`dataset. Answer the following questions:
   a) How many clusters, core objects, border objects and noise objects are found by DBSCAN?
   b) How many cities does the largest cluster contain and in which country are the cities of the largest cluster located?
   c) Which three countries have the most cities in clusters?
   d) Are the Indian cities `Rajendranagar` und `Rajpur` (directly) density-reachable or density-connected?
   e) Are `Essen` und `Castrop-Rauxel` (directly) density-reachable or density-connected?
   f) Which cities are density-reachable from Bochum, but not directly density-reachable?

```r
library(tidyverse)
library(purrr)
library(maps)
library(dbscan)
library(stringr)
library(magrittr)
library(leaflet)

data("world.cities")
world.cities <- world.cities %>% filter(pop >= 50000)

minPts <- 8
eps <- 0.15

clusters <- dbscan(select(world.cities, lat, long), minPts = minPts, eps = eps)
world.cities$cluster <- clusters$cluster
# Mark objects within a cluster with `1` and
# objects outside a cluster with `-1` (noise).
world.cities$type <- ifelse(clusters$cluster > 0, 1, -1)
# Mark border objects with `0` -> Thus all
# objects with `type == 1` are core objects.
world.cities$type[which(! clusters$cluster ==
                          dbscan(select(world.cities, lat, long),
                                 minPts = minPts, eps = eps,
                                 borderPoints = F)$cluster)] <- 0


# Two alternative representations
groups  <- world.cities %>% filter(cluster != 0)
noise  <- world.cities %>% filter(cluster == 0)
```

```
# (1)
ggplot(world.cities, aes(x = long, y = lat, alpha = 0.5)) +
  geom_point(aes(fill = "grey"), noise) +
  geom_point(aes(colour = as.factor(cluster)), groups,
             size = 3) +
  coord_map() +
  theme(legend.position = "none")
```




```
# # (2)
# leaflet(world.cities %>%
#           filter(cluster>0)) %>%
#   addTiles() %>%
#   addMarkers(~long, ~lat, popup = ~name)

# a)
# cluster number:
max(world.cities$cluster)
```

```
## [1] 41
```

```
# number of core objects, border objects, noise:
tibble(
  num_core = sum(world.cities$type == 1),
  num_border = sum(world.cities$type == 0),
```

```r
  num_noise = sum(world.cities$type == -1)
)
```

```
## # A tibble: 1 x 3
##   num_core num_border num_noise
##      <int>      <int>     <int>
## 1      457        294      8127
```

```r
# b)
cluster_sizes <- world.cities %>% filter(cluster > 0) %>%
  count(cluster, sort = T) %>% ungroup()
cluster_sizes %>% slice(1)
```

```
## # A tibble: 1 x 2
##   cluster     n
##     <int> <int>
## 1       1   103
```

```r
world.cities %>% filter(cluster == cluster_sizes$cluster[1]) %$%
  table(.$country.etc)
```

```
##
## Japan
##   103
```

```r
# c)
world.cities %>% filter(cluster > 0) %>% count(country.etc, sort = T)
```

```
## # A tibble: 26 x 2
##    country.etc     n
##    <chr>       <int>
##  1 Japan         189
##  2 USA           100
##  3 Indonesia      70
##  4 India          63
##  5 Philippines    39
##  6 France         32
##  7 Spain          24
##  8 Greece         23
##  9 Taiwan         20
## 10 UK             16
## # ... with 16 more rows
```

```r
# d)
print("d")
```

```
## [1] "d"
```

```r
world.cities %>% filter(name %in% c("Rajendranagar", "Rajpur"))
```

```
##             name country.etc    pop   lat  long capital cluster type
```

```
## 1 Rajendranagar       India 182541 17.29 78.39       0       8     0
## 2        Rajpur       India 478518 22.44 88.44       0      17     0
```

```r
# Neither, since they are in different clusters.
#(Actually, it should still be checked whether there is a chain
# of core objects connecting the two cities with each other.)
# (Theoretically, it is possible that objects located in different
# clusters are densely reachable when a border object is in the
# eps neighborhood of core objects of different clusters.)
# ...is not the case here...

# e)
print("e")
```

```
## [1] "e"
```

```r
world.cities %>% filter(country.etc == "Germany", cluster > 0)
```

```
##              name country.etc    pop   lat long capital cluster type
## 1          Bochum     Germany 384208 51.48 7.20       0      25    1
## 2  Castrop-Rauxel     Germany  77660 51.55 7.31       0      25    0
## 3           Essen     Germany 596204 51.47 7.00       0      25    0
## 4    Gelsenkirchen    Germany 267673 51.51 7.11       0      25    1
## 5         Gladbeck     Germany  76720 51.58 6.98       0      25    0
## 6        Hattingen     Germany  56355 51.41 7.18       0      25    0
## 7           Herne     Germany 171360 51.54 7.21       0      25    0
## 8          Herten     Germany  64936 51.59 7.21       0      25    0
## 9  Recklinghausen     Germany 121791 51.61 7.19       0      25    0
## 10         Witten     Germany 100706 51.44 7.34       0      25    0
```

```r
# density-reachable, since both cities belong to the same cluster,
# but are only border objects

# f)
print("f")
```

```
## [1] "f"
```

```r
bochum_coord <- world.cities %>% filter(name == "Bochum") %$% c(.$lat[1], .$long[1])
bochum_cluster <- world.cities %>% filter(name == "Bochum") %$% .$cluster[1]

world.cities %>%
  mutate(dist_bochum = sqrt((lat-bochum_coord[1])^2 + (long-bochum_coord[2])^2)) %>%
  filter(dist_bochum > eps, cluster == bochum_cluster)
```

```
##       name country.etc    pop   lat long capital cluster type dist_bochum
## 1    Essen     Germany 596204 51.47 7.00       0      25    0   0.2002498
## 2 Gladbeck     Germany  76720 51.58 6.98       0      25    0   0.2416609
```

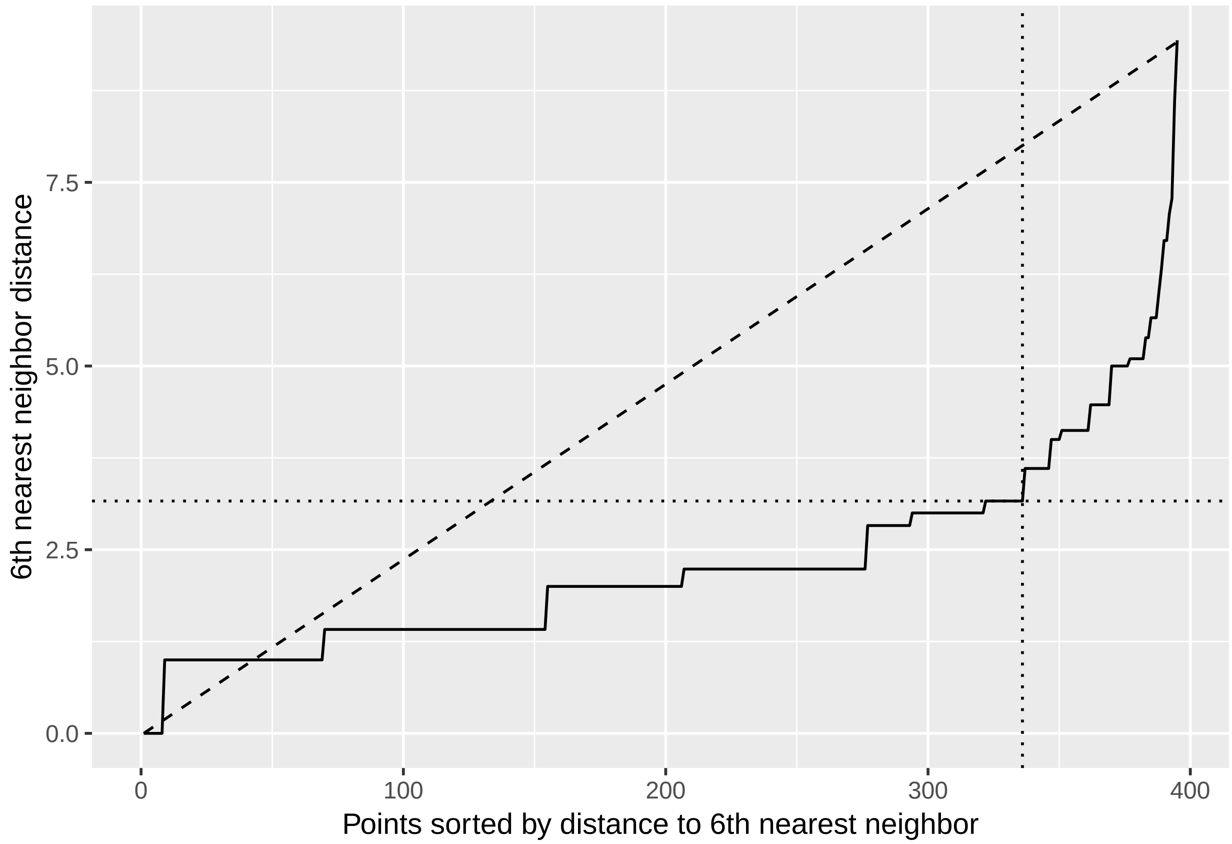
```r
# Essen und Gladbeck
```

2. Given again be the dataset from task 2 of task sheet 3. This time use DBSCAN with

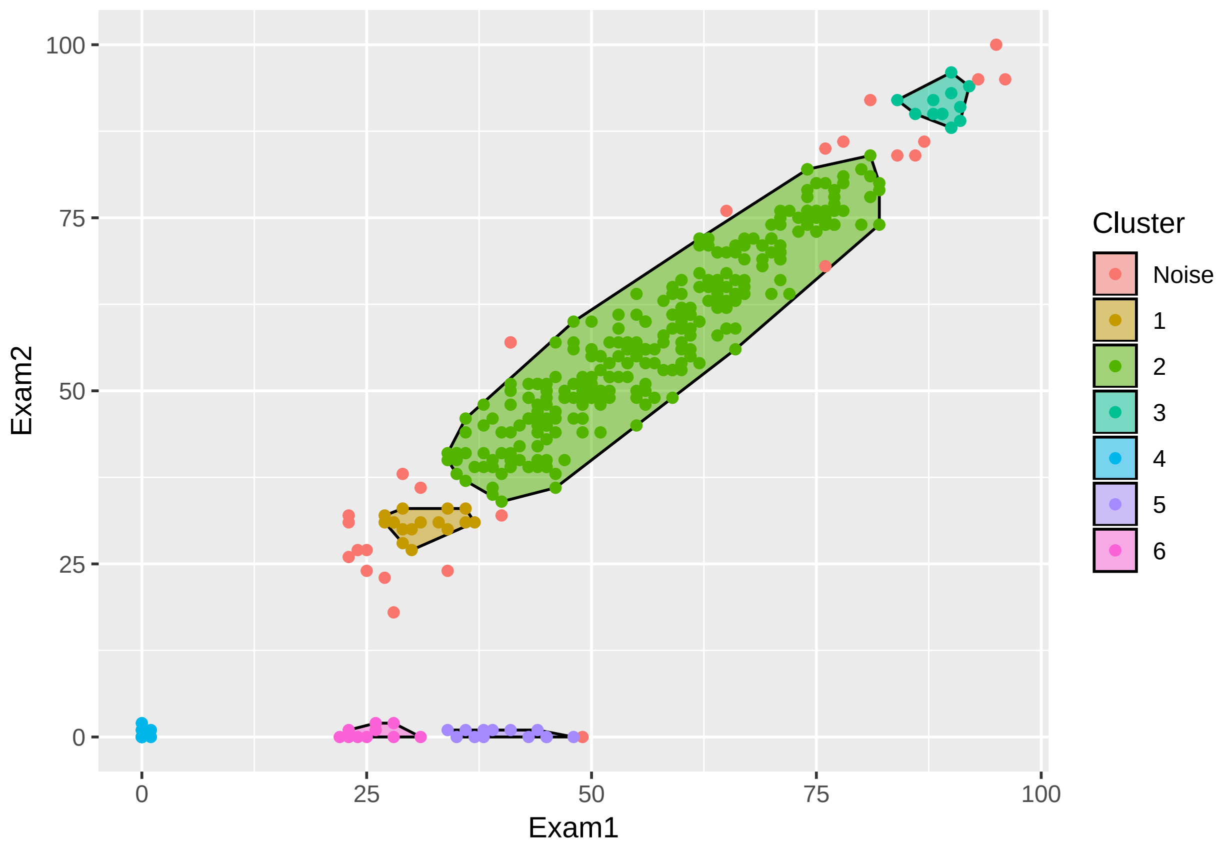
$minPts = 6$ for clustering. First determine a *suitable* value for ϵ. Display the clustering in a scatter plot. Highlight cluster assignments and noise points in color. Compare and discuss the clustering of DBSCAN with the clustering of k-Means.

```r
library(readr)
library(dplyr)
library(stringr)
library(forcats)
library(tibble)
library(tidyr)
library(ggplot2)
library(purrr)
student <- read_csv(str_c(dirname(getwd()), "/Ex_5/clustering-student-mat.csv"))
```

```r
k <- 6
# Create Distance Matrix
dm <- student %>% dist() %>% as.matrix() %>% as_tibble()
# Convert distance matrix so that all distances are stored in one variable
# For each instance select the `k` nearest distance and sort data frame by distance
# `ll`: Calculate straight between smallest and largest 6-dist
dm <- dm %>%
  mutate(id = row_number()) %>%
  gather(id2, dist, -id) %>%
  group_by(id) %>%
  arrange(dist) %>%
  slice(k) %>%
  ungroup() %>%
  arrange(dist) %>%
  mutate(no = row_number()) %>%
  mutate(ll = dist[1] + (no-1)*((dist[n()] - dist[1])/n()))

# Determine minimum distance between straight line and 6-dist.
dm_opt <- dm %>%
  mutate(lldist = ll - dist) %>%
  arrange(-lldist) %>%
  slice(1)

ggplot(dm, aes(x = no, y = dist)) +
  geom_line() +
  geom_line(aes(y = ll), linetype = 2) +
  geom_vline(xintercept = dm_opt$no, linetype = 3) +
  geom_hline(yintercept = dm_opt$dist, linetype = 3) +
  labs(x = str_c("Points sorted by distance to ", k, "th nearest neighbor"),
       y = str_c(k, "th nearest neighbor distance"))
```

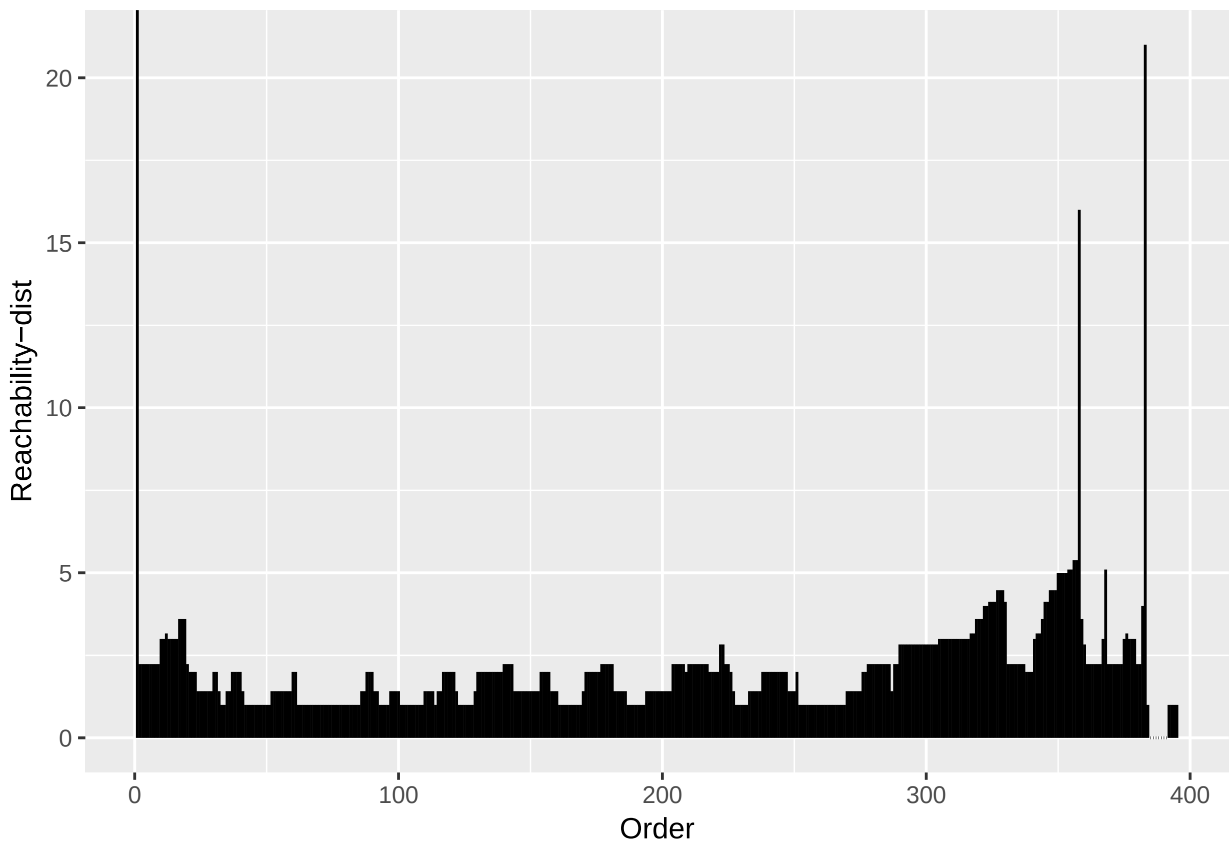



```r
minPts <- k
eps <- dm_opt$dist[1]

library(dbscan)
clu <- dbscan(student, minPts = minPts, eps = eps)

student_clu <- student %>%
  bind_cols(., tibble(Cluster = clu$cluster)) %>%
  mutate(Cluster = factor(Cluster)) %>%
  mutate(Cluster = fct_recode(Cluster, "Noise" = "0"))

student_hull <- student_clu  %>%
  split(.$Cluster) %>%
  purrr::map(~ slice(., chull(.$Exam1, .$Exam2))) %>%
  do.call("rbind", .)

ggplot(student_clu, aes(Exam1, Exam2, color = Cluster, fill = Cluster)) +
  geom_polygon(data = student_hull %>% filter(!Cluster == "Noise"), alpha = .5, color = "black"
  geom_point(pch = 21) +
  scale_fill_discrete(drop = F) +
  scale_color_discrete(drop = F)
```



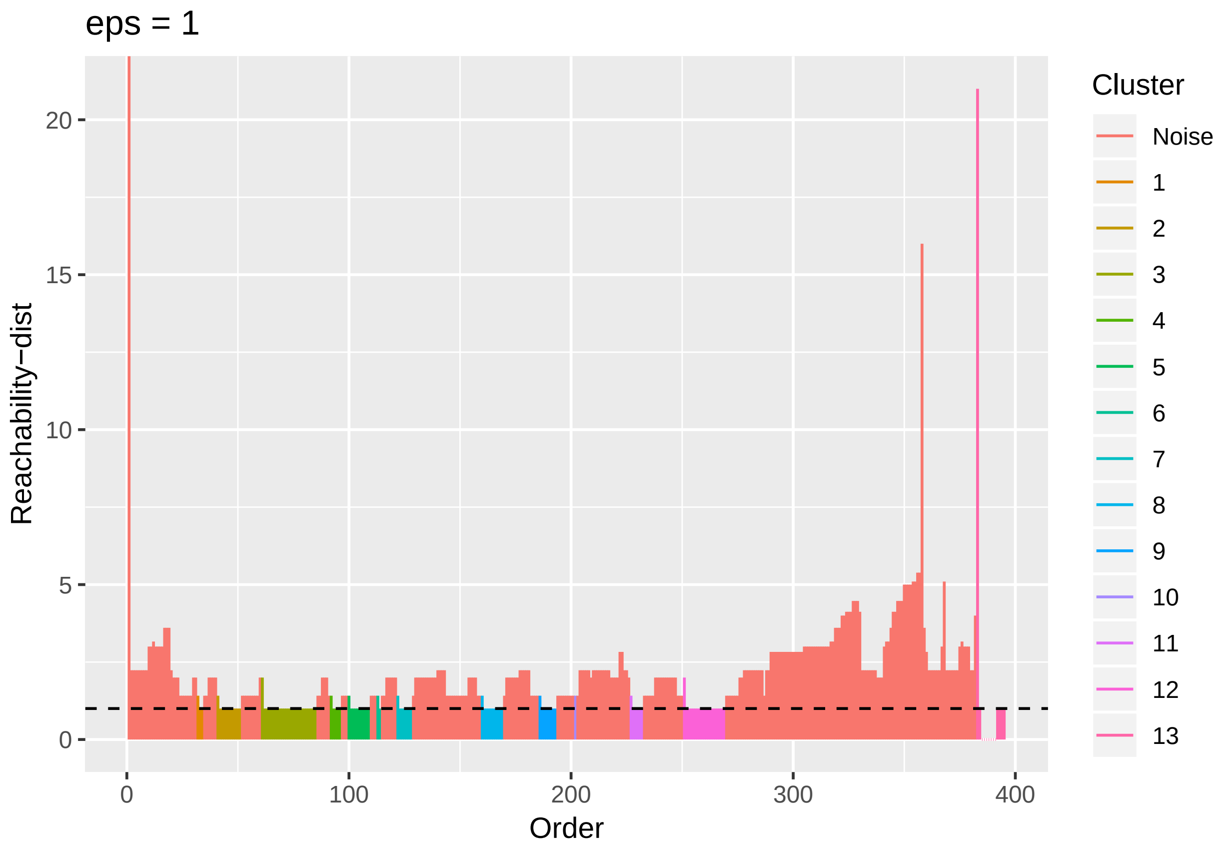
3. Given again, be the dataset from task 2 of task sheet 3. Use OPTICS to create a density reachability diagram for $minPts = 6$. Extract a clustering for each $\textit{reachability-dist} = \{1, 1.5, \ldots, 5\}$ and display the result in a scatter plot, respectively. Highlight cluster assignments and noise points in color. Evaluate the change of the clustering result with increasing threshold for *reachability-dist* regarding the number of clusters as well as the number of core, border, and noise points.

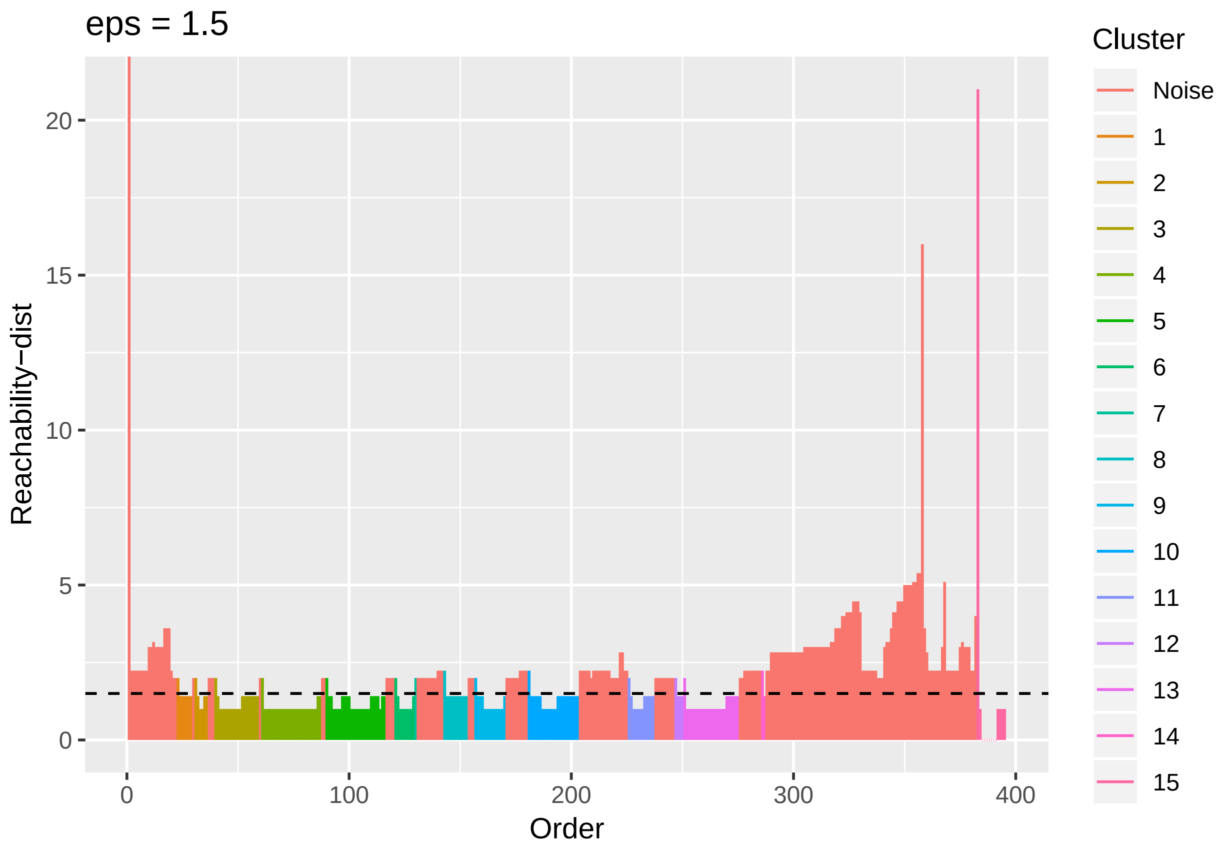
```
k <- 6
optics_clu <- optics(student, eps = Inf, minPts = k)

student_clu <- student %>%
  bind_cols(., tibble(R_Dist = optics_clu$reachdist))
student_clu <- student_clu[optics_clu$order,]
student_clu$Order <- 1:nrow(student_clu)

ggplot(student_clu, aes(x = Order, xend = Order, y = 0, yend = R_Dist)) +
  geom_segment() +
  labs(y = "Reachability-dist")
```

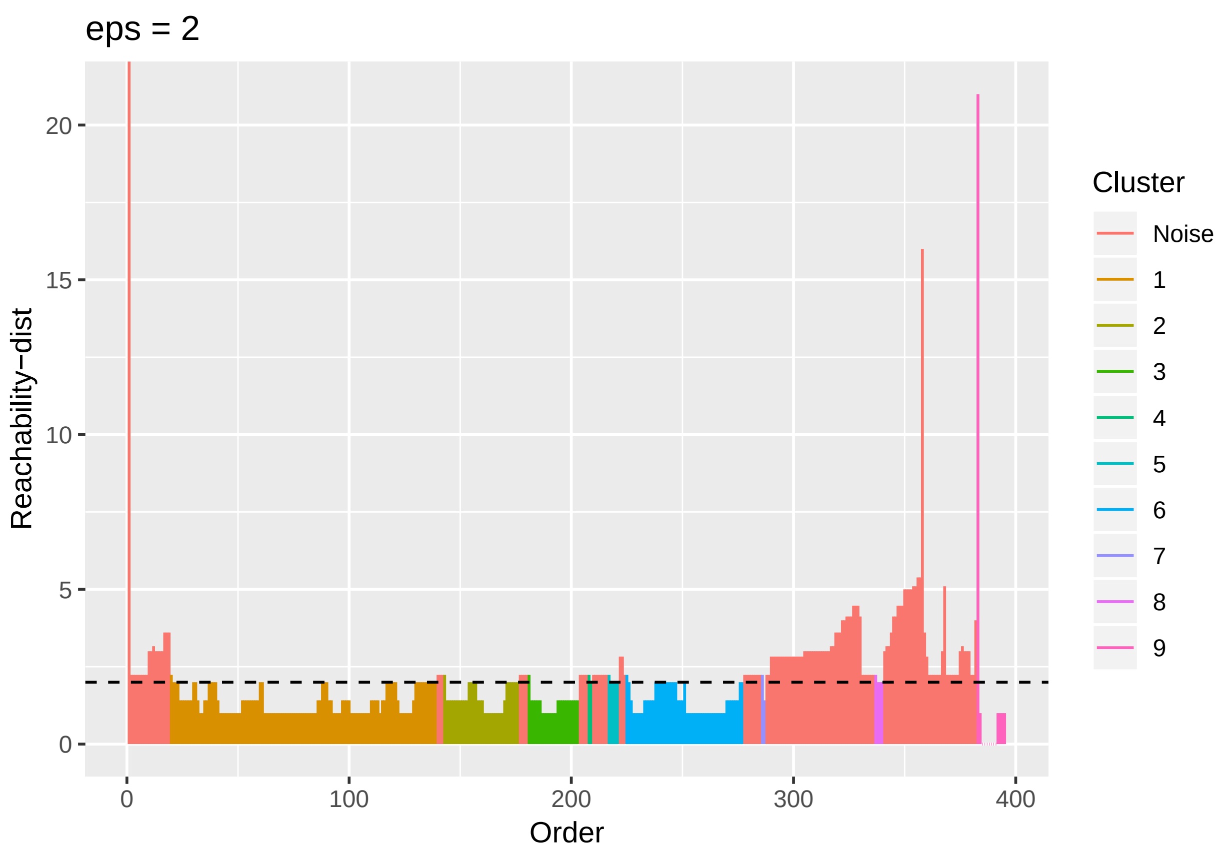



```
eps <- seq(1,5, by = 0.5)

reach_plot_list <- vector("list", length(eps))
clust_plot_list <- vector("list", length(eps))

for(i in seq_along(eps)) {
  clu <- extractDBSCAN(optics_clu, eps_cl = eps[i])

  student_clu <- student %>%
    bind_cols(., tibble(Cluster = clu$cluster, R_Dist = optics_clu$reachdist)) %>%
    mutate(Cluster = factor(Cluster)) %>%
    mutate(Cluster = fct_recode(Cluster, "Noise" = "0"))
  student_clu <- student_clu[optics_clu$order,]
  student_clu$Order <- 1:nrow(student_clu)


  reach_plot_list[[i]] <- ggplot(student_clu, aes(x = Order, xend = Order, y = 0, yend = R_Dis
    geom_segment(aes(color = Cluster)) +
    geom_hline(yintercept = eps[i], linetype = 2) +
    labs(y = "Reachability-dist", title = str_c("eps = ", eps[i]))

  student_hull <- student_clu  %>%
    split(.$Cluster) %>%
```

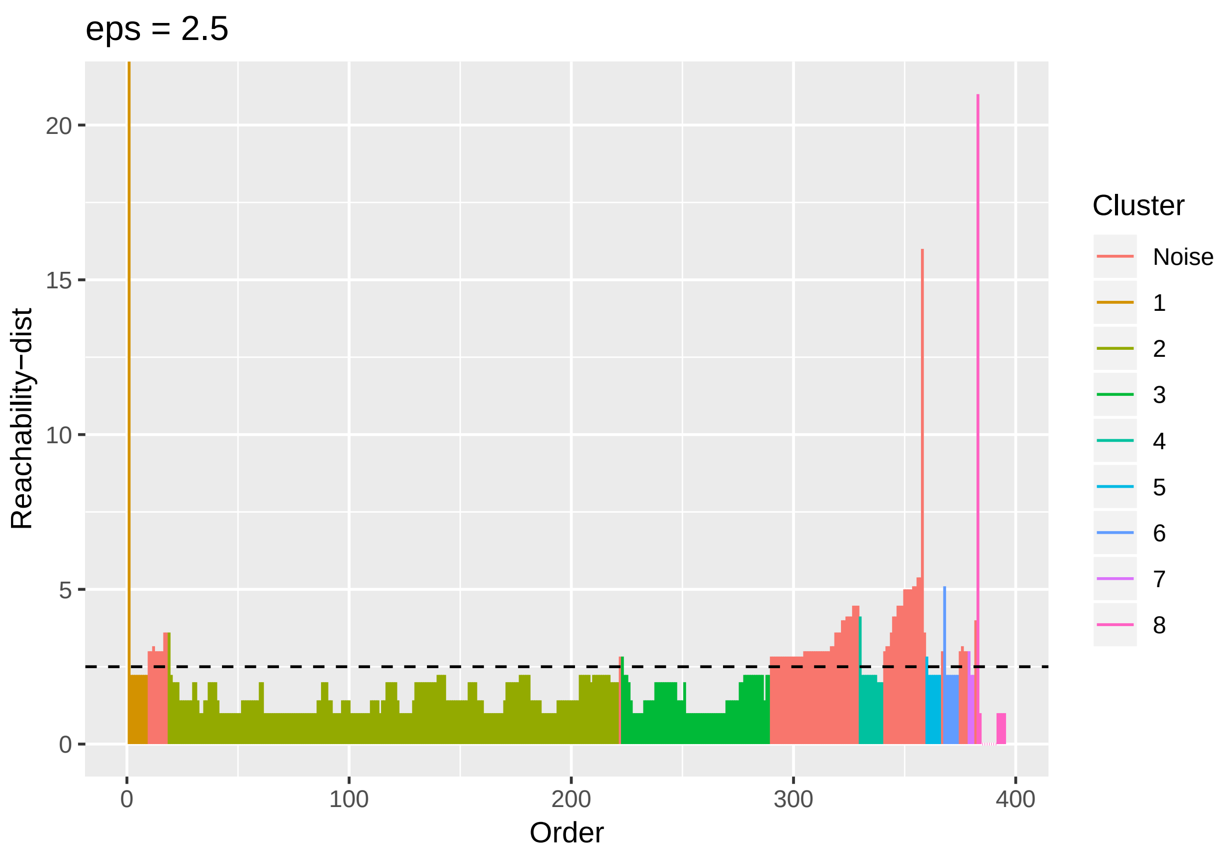
```
    purrr::map(~ slice(., chull(.$Exam1, .$Exam2))) %>%
    do.call("rbind", .)

  clust_plot_list[[i]] <- ggplot(student_clu, aes(Exam1, Exam2, color = Cluster, fill = Cluste
    geom_polygon(data = student_hull %>% filter(!Cluster == "Noise"), alpha = .5, color = "blac
    geom_point(pch = 21) +
    scale_fill_discrete(drop = F) +
    scale_color_discrete(drop = F) +
    labs(title = str_c("eps = ", eps[i]))
}

reach_plot_list %>% walk(print)
```

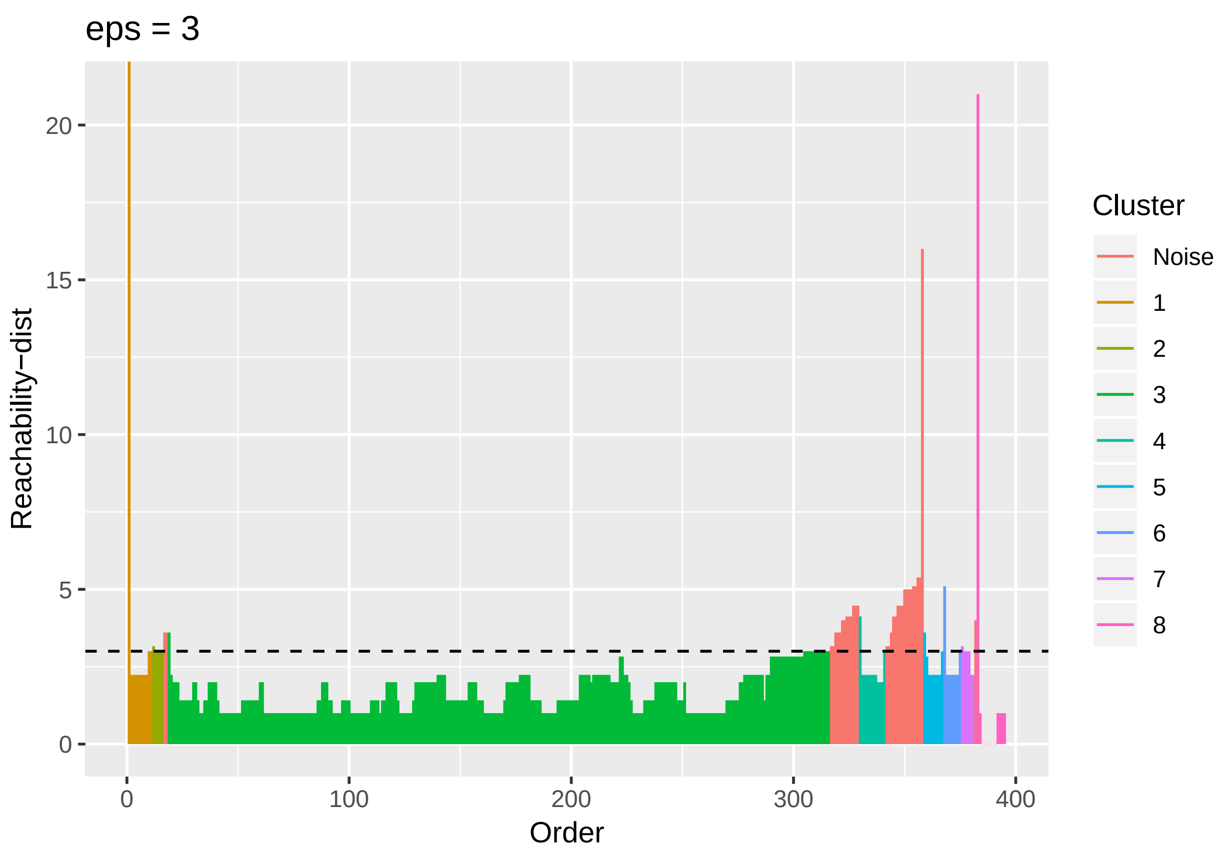



eps = 1.5
Reachability−dist
0
5
10
15
20
0
100
200
300
400
Order
Cluster
Noise
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

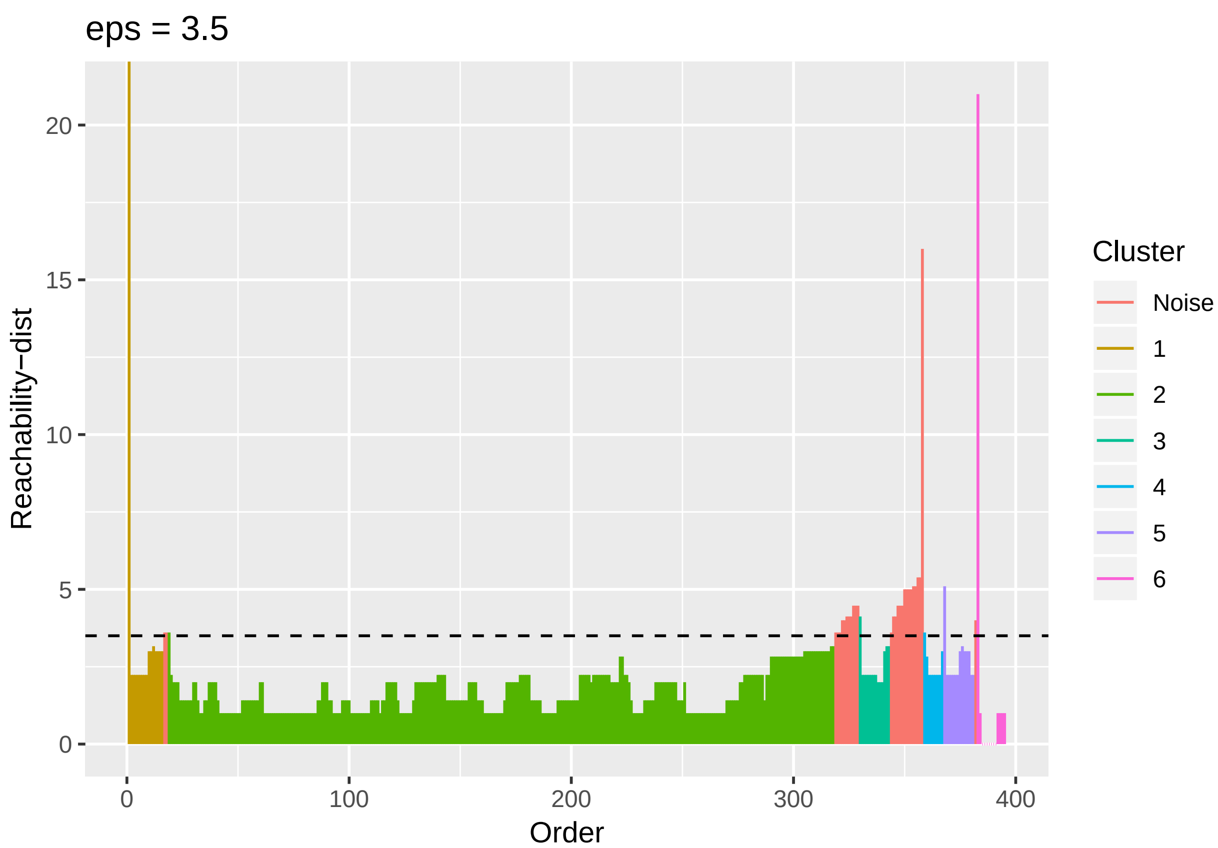


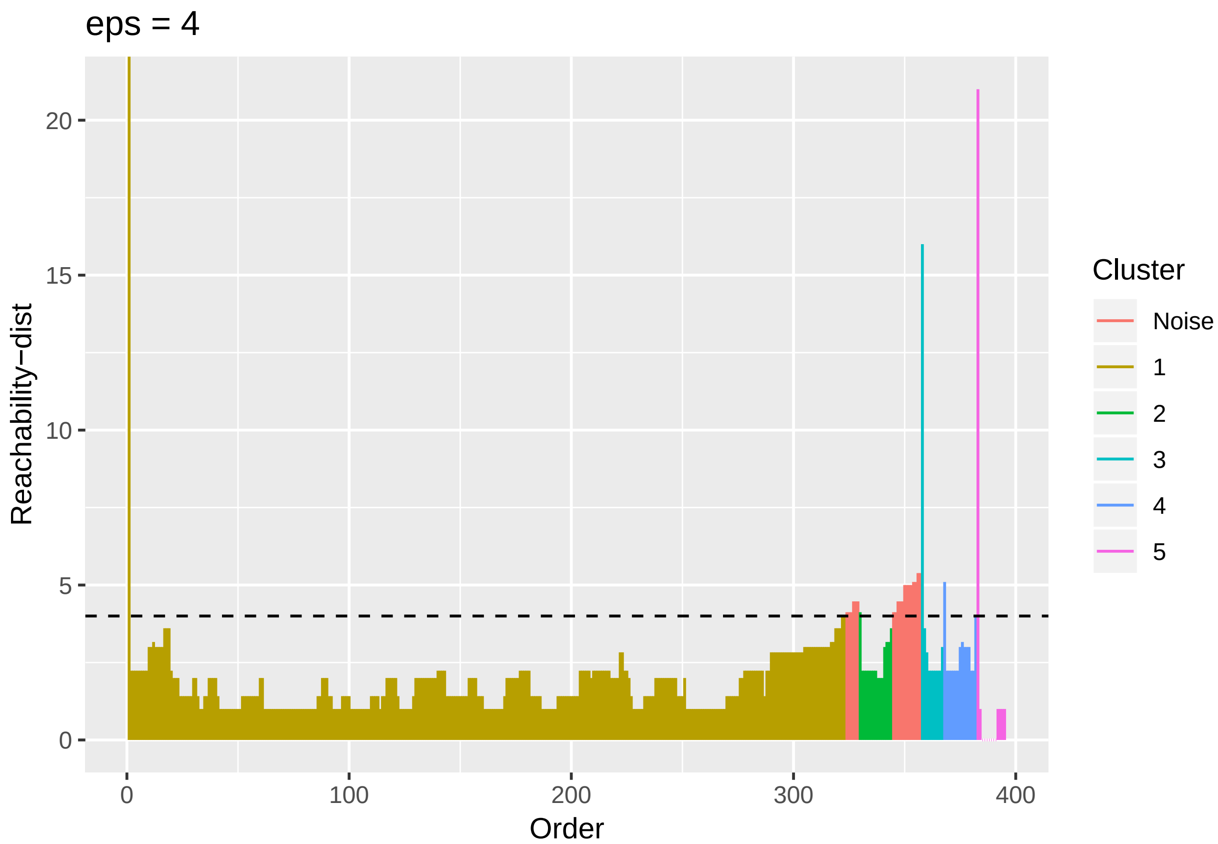
eps = 2
Reachability−dist
20
15
10
5
0
0
100
200
300
400
Order
Cluster
Noise
1
2
3
4
5
6
7
8
9


eps = 2.5
Reachability−dist
0
5
10
15
20
0
100
200
300
400
Order
Cluster
Noise
1
2
3
4
5
6
7
8

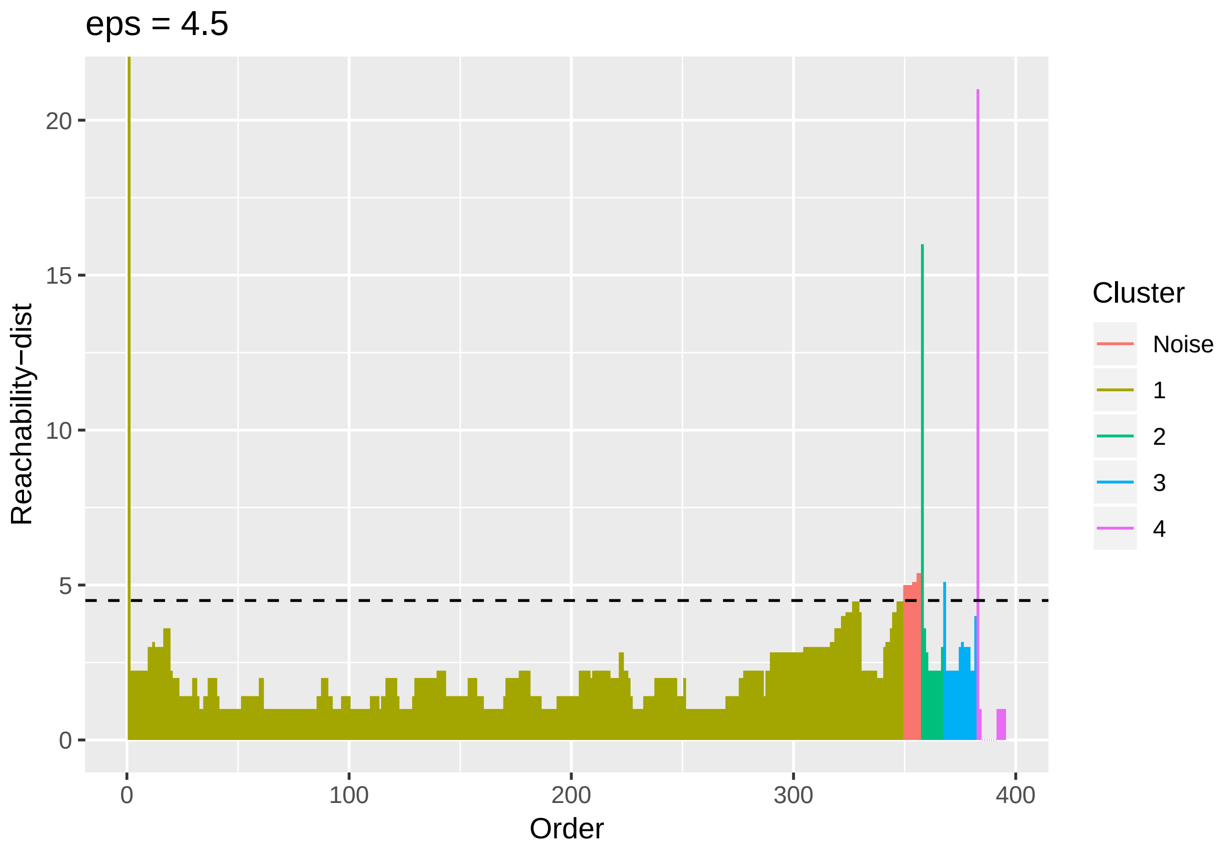


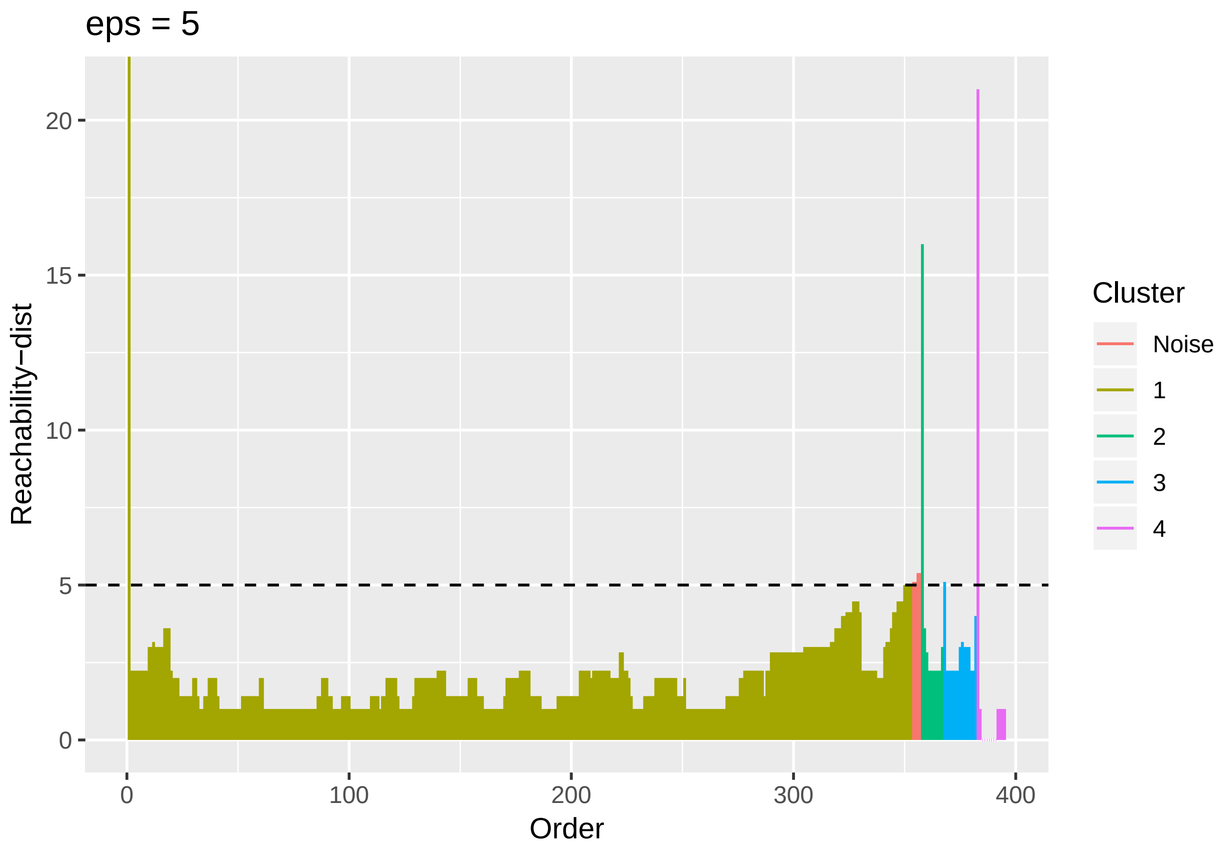
eps = 3
Reachability−dist
Order
0
5
10
15
20
0
100
200
300
400
Cluster
Noise
1
2
3
4
5
6
7
8


eps = 3.5
Reachability−dist
20
15
10
5
0
0
100
200
300
400
Order
Cluster
Noise
1
2
3
4
5
6

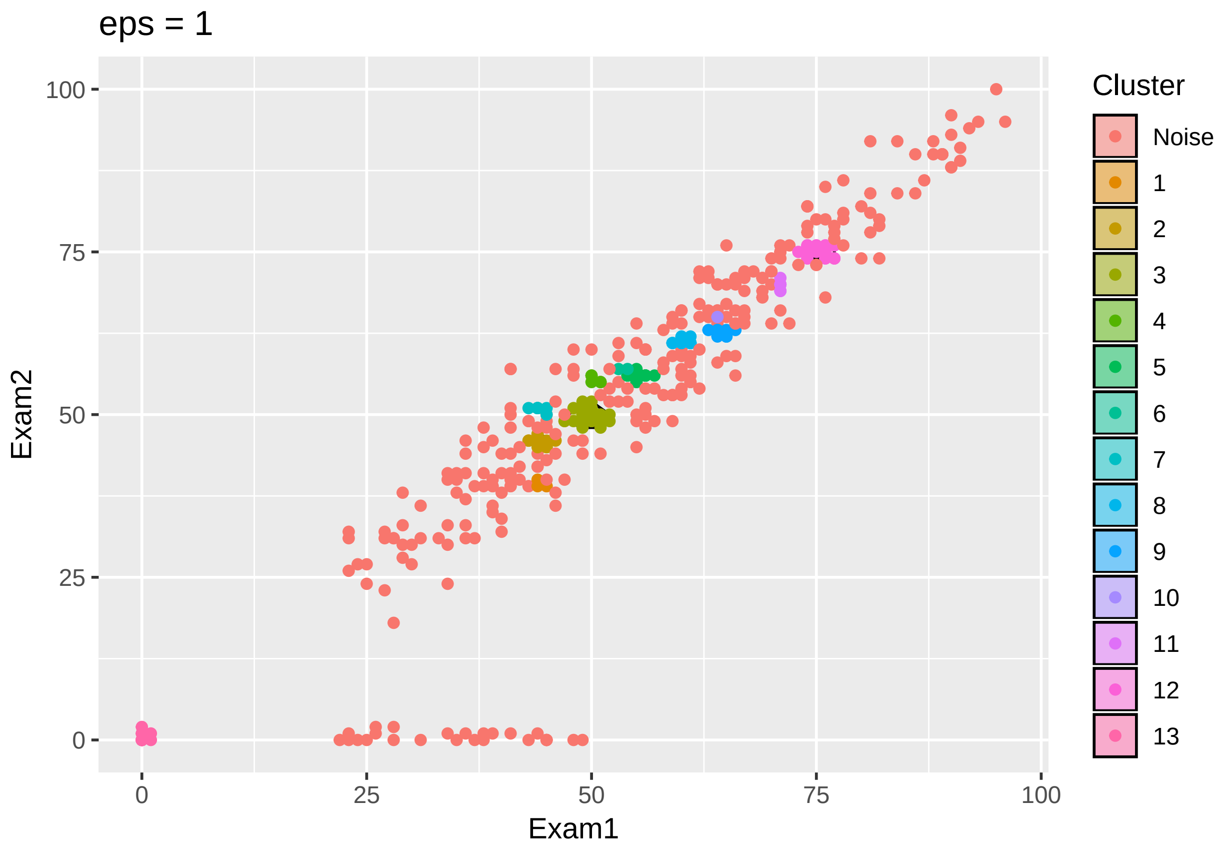


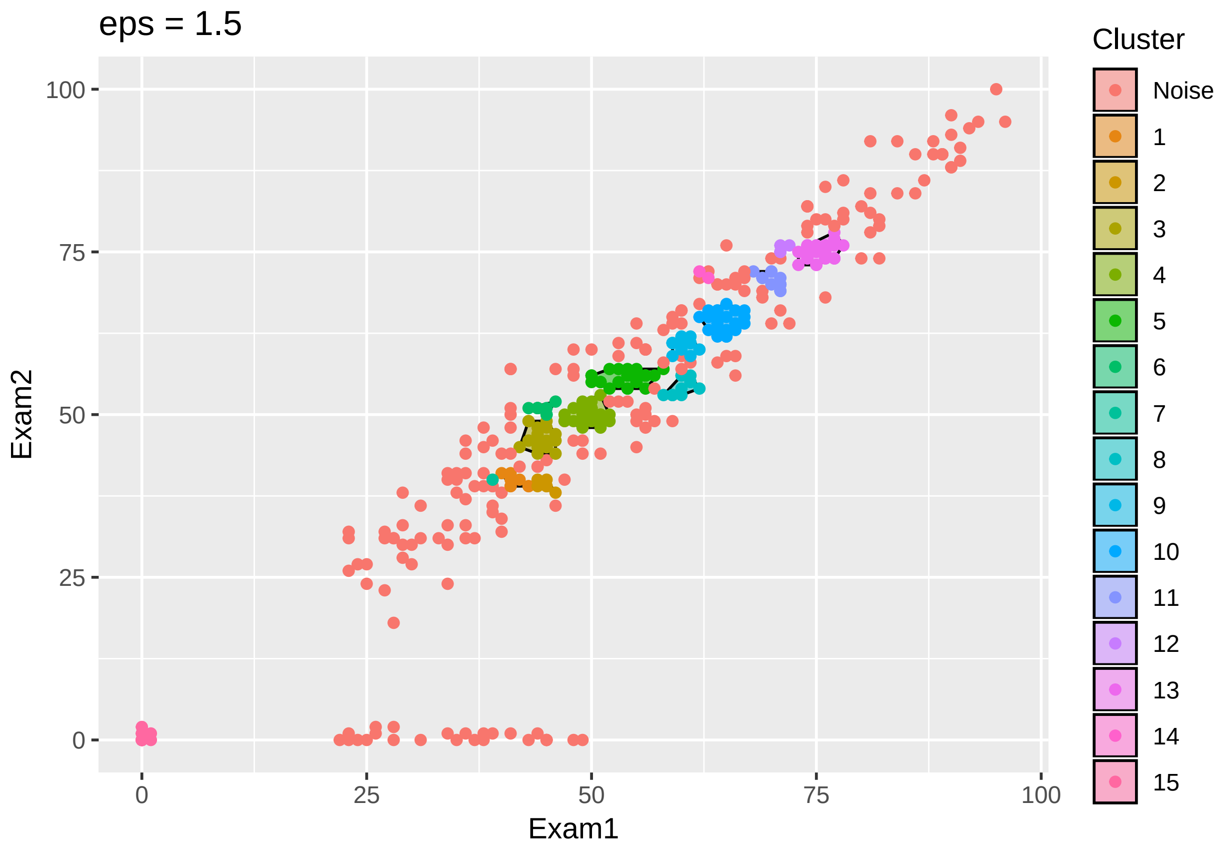
eps = 4
Reachability−dist
Order
0
5
10
15
20
0
100
200
300
400
Cluster
Noise
1
2
3
4
5

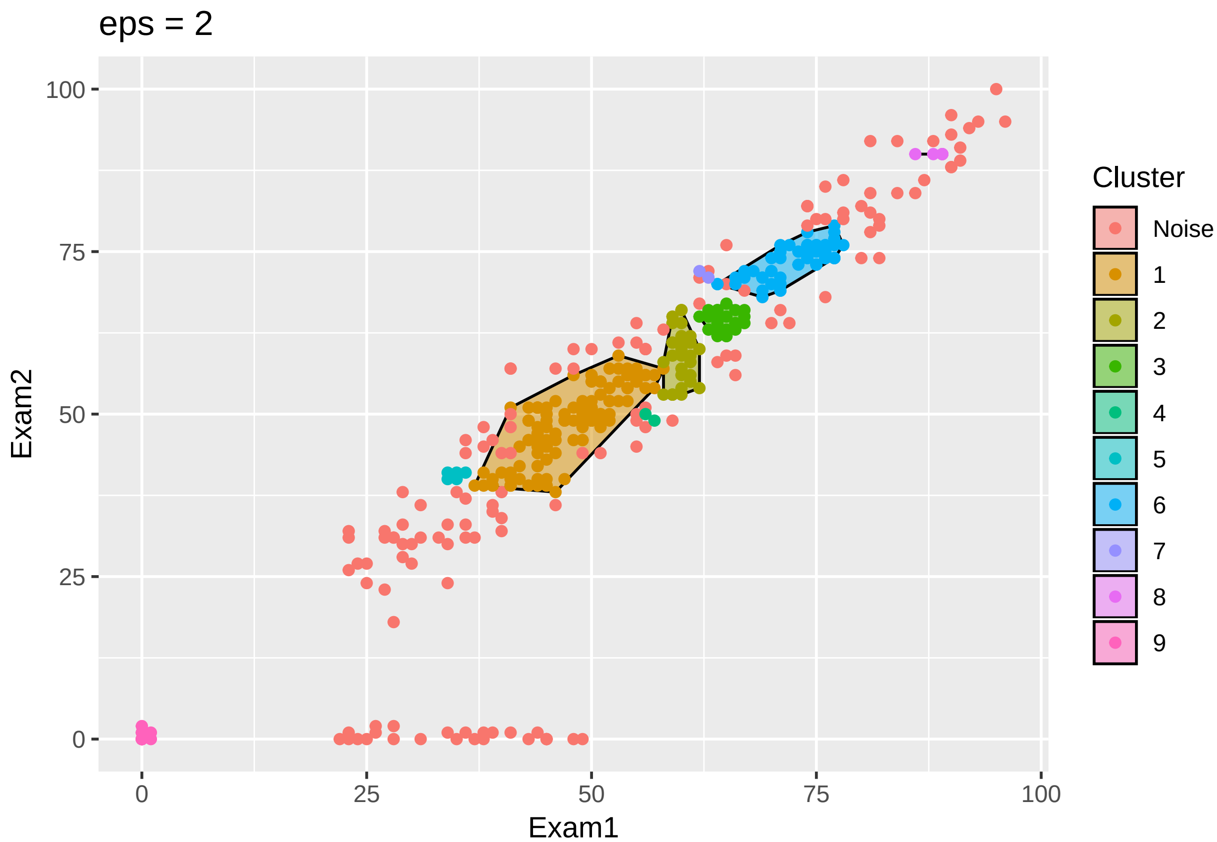

eps = 4.5
Reachability−dist
Order
0
5
10
15
20
0
100
200
300
400
Cluster
Noise
1
2
3
4

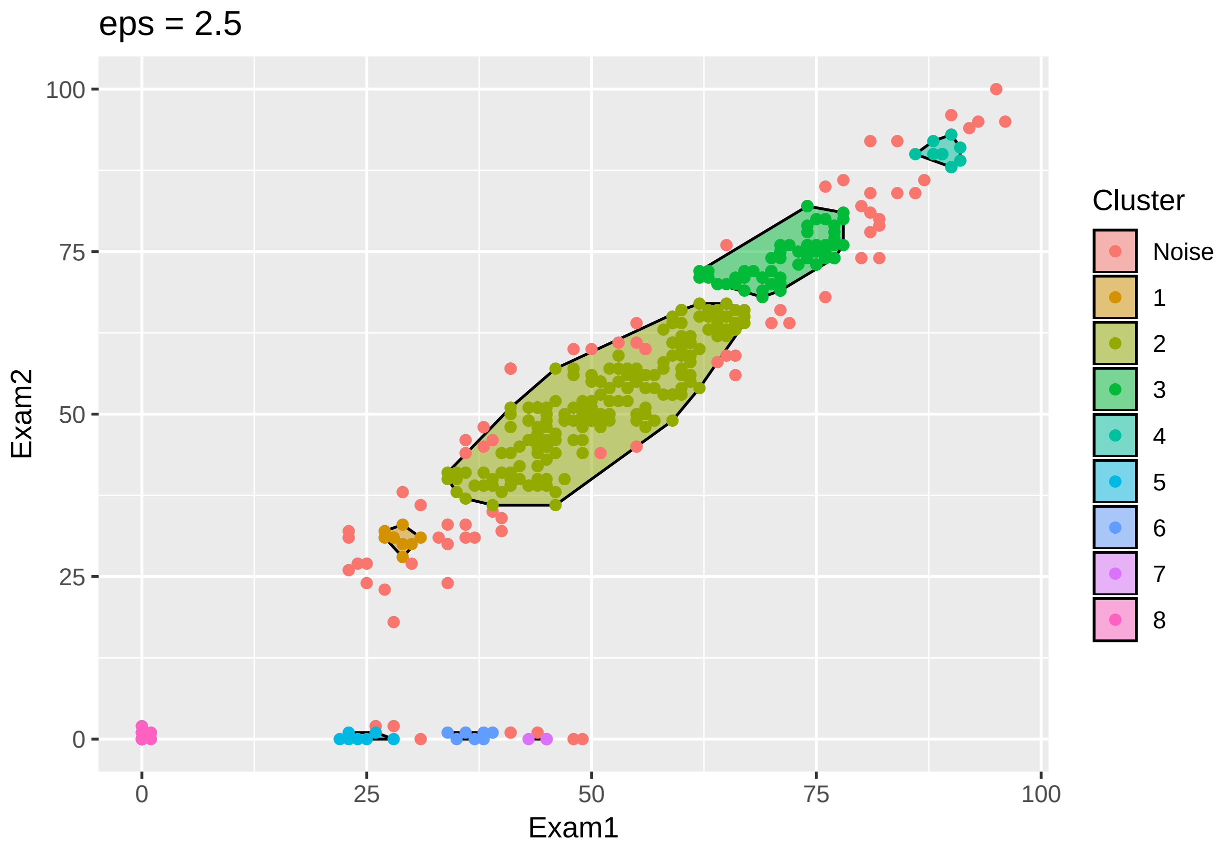



```
clust_plot_list %>% walk(print)
```

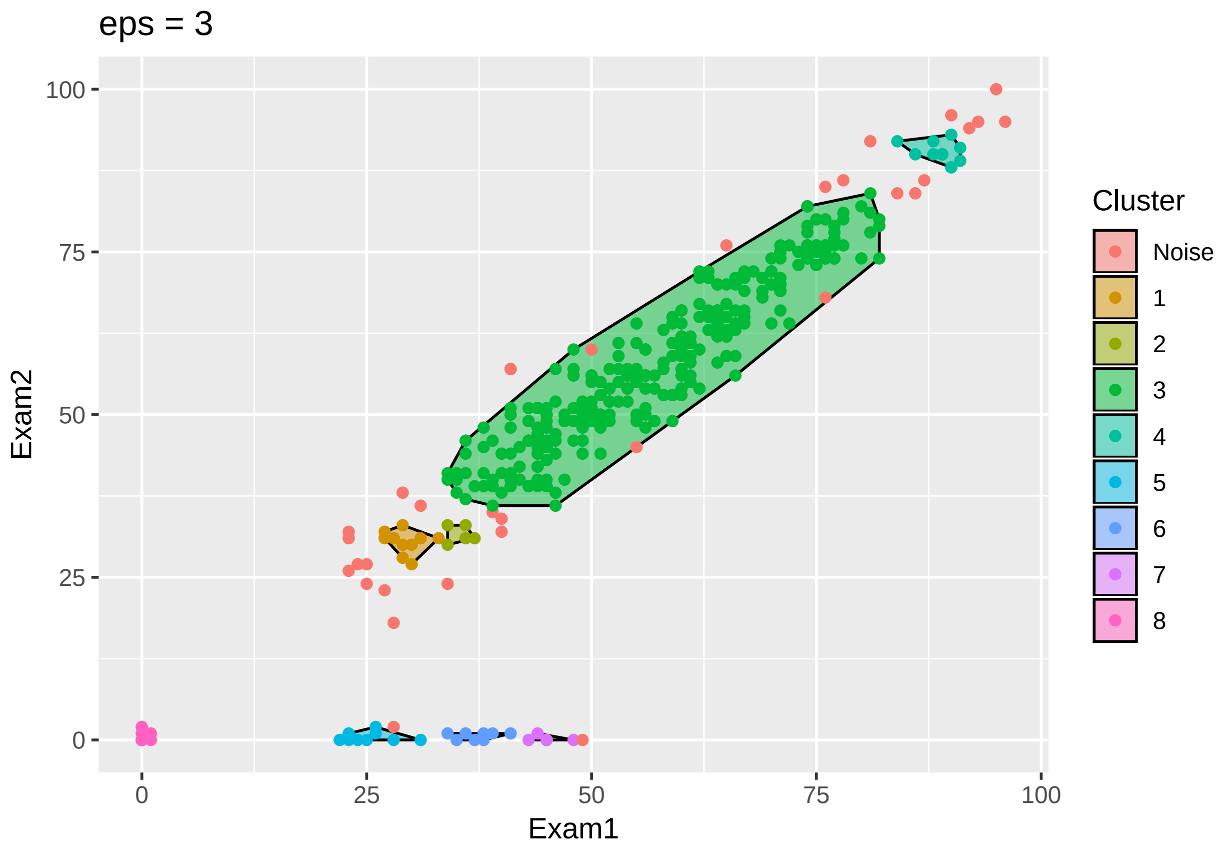

eps = 1
Exam2
Exam1
0
25
50
75
100
Cluster
Noise
1
2
3
4
5
6
7
8
9
10
11
12
13

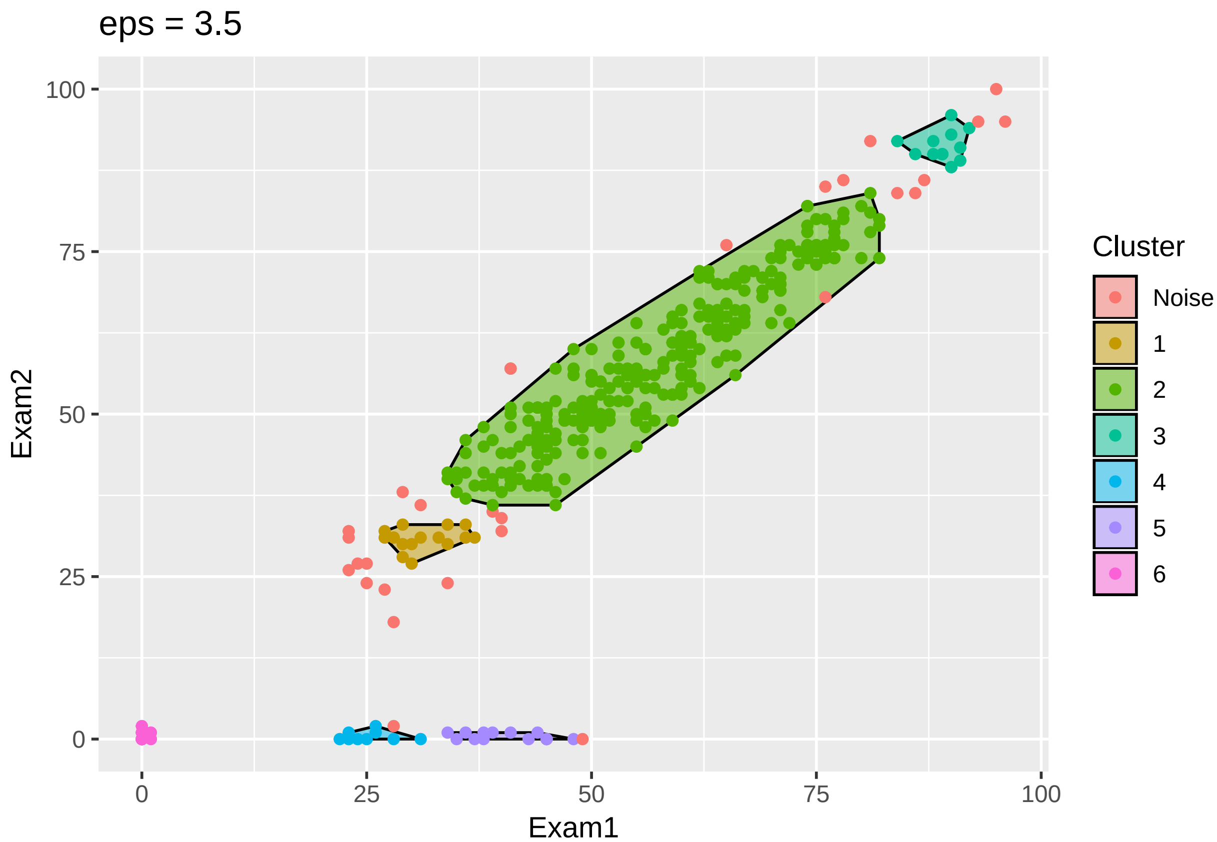

eps = 1.5
Cluster
Noise
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Exam2
Exam1
100
75
50
25
0
0
25
50
75
100

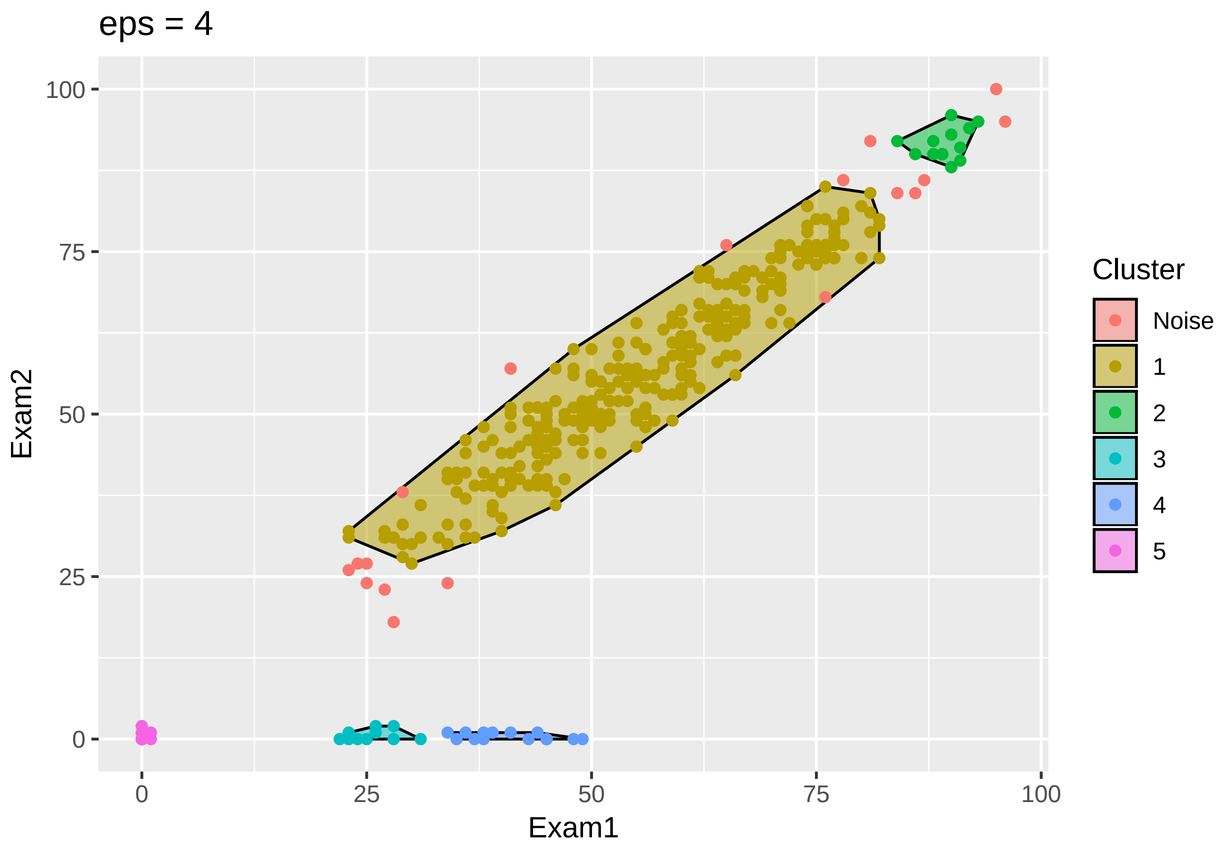

eps = 2
Exam2
Exam1
0
25
50
75
100
Cluster
Noise
1
2
3
4
5
6
7
8
9



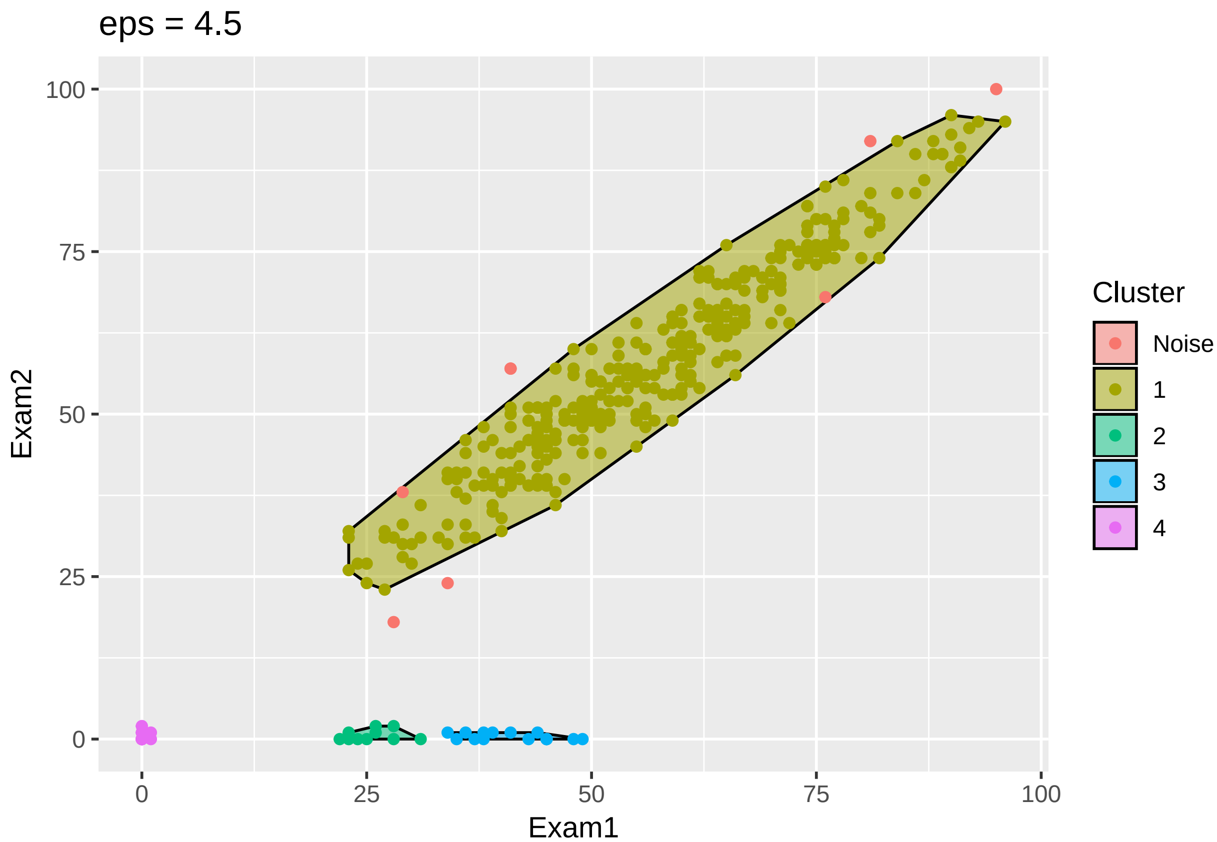
eps = 2.5
Exam2
100
75
50
25
0
0
25
50
75
100
Exam1
Cluster
Noise
1
2
3
4
5
6
7
8

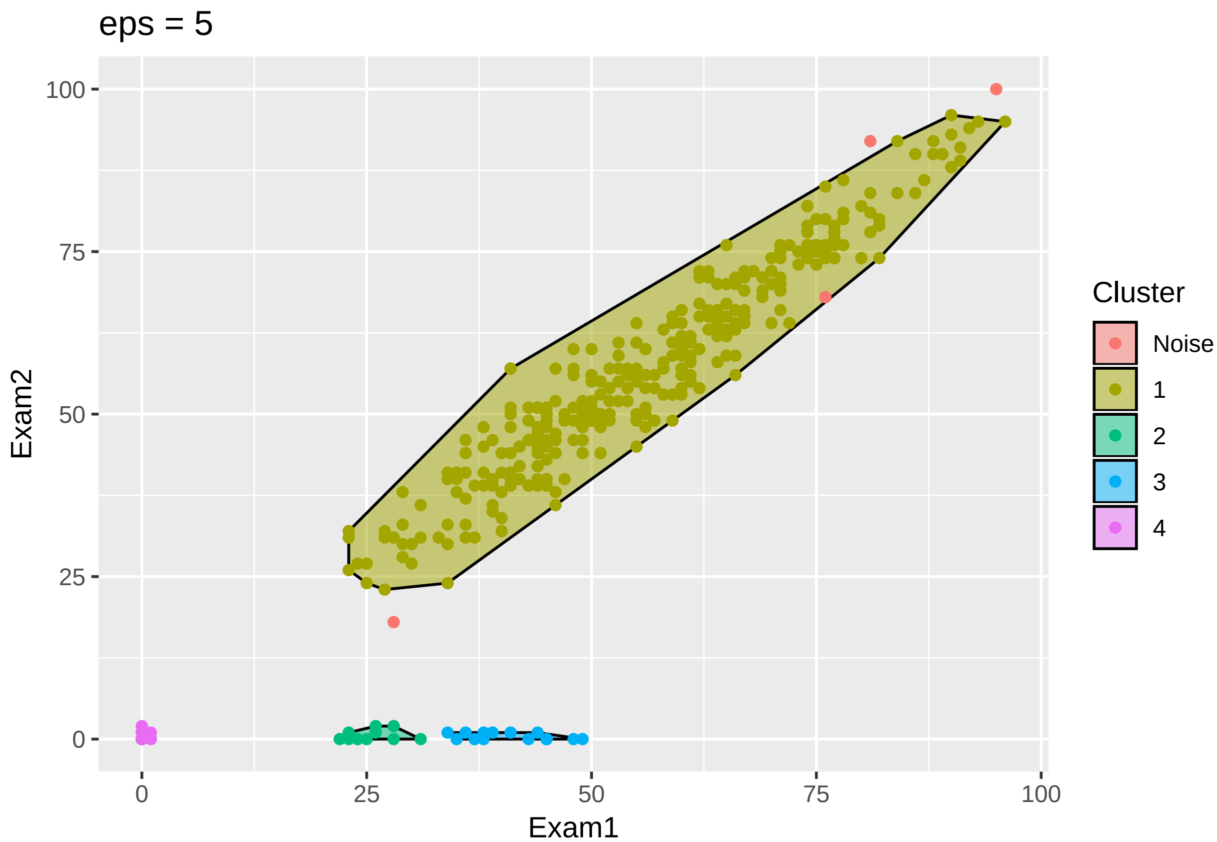

eps = 3
Exam2
Exam1
100
75
50
25
0
0
25
50
75
100
Cluster
Noise
1
2
3
4
5
6
7
8


eps = 3.5
Exam2
Exam1
100
75
50
25
0
0
25
50
75
100
Cluster
Noise
1
2
3
4
5
6



eps = 4
Exam2
100
75
50
25
0
Exam1
0
25
50
75
100
Cluster
Noise
1
2
3
4
5


eps = 4.5
Exam2
Exam1
Cluster
Noise
1
2
3
4
0
25
50
75
100



4. Using the example of the silhouette coefficient, discuss the strengths and weaknesses of internal quality measures. Why are they only conditionally suitable for the comparison between clusterings of different algorithms (e.g. K-Means and DBSCAN)? In which cases should they still be used?

- internal measures prefer clustering algorithms with a similar objective function.
- Silhouette coefficient (SC) will tend to yield poor values for non-spherical clustering algorithms.
- Reason: SC prefers low average distances to objects of the same cluster and high average distances to objects of other clusters.
- all internal measures define a prototype for an 'optimal' clustering, i.e. the quality measure only makes a statement about how well the clustering approaches this prototype, and not how good the intrinsic clustering is -> could lead to overfitting
- In which applications should internal mass be used?
  - Comparison of clusterings of the same type (partitioning)
  - Comparison of the same clustering algorithm with different parameter combinations -> parameter tuning
- Possible bias concerning the number of clusters e.g. in cohesion, sum of the quadratic distances to the centroid,... -> this measures is preferred by many clusters

---

Dataset for task 2 and 3:
http://isgwww.cs.uni-magdeburg.de/cv/lehre/VisAnalytics/material/exercise/datasets/clustering-student-mat.csv