Spectral clustering

Przemysław Kaleta

January 23, 2019

W tym pliku wykonamy pierwsze testy dla naszego nowo zaimplementowanego algorytmu. Wykonamy wizualizację otrzymanych skupień i obliczymy indeksy dla otrzymanych predykcji. Na koniec stworzymy i zapiszemy wygenerowane ręcznie zbiory danych.

Funkcje do testów

Stwórzmy najpierw funkcje, które posłużą nam do wczytywania i analizowania danych.

```
read_data <- function(benchmark, dataset){
  matrix_file_name <- paste(dataset, ".data.gz", sep="")
  labels_file_name <- paste(dataset, ".labels0.gz", sep="")
  matrix_path <- file.path("..", "benchmarks", benchmark, matrix_file_name)
  labels_path <- file.path("..", "benchmarks", benchmark, labels_file_name)
  X <- as.matrix(read.table(matrix_path))
  Y <- as.matrix(read.table(labels_path))
  return(list(X=X, Y=Y))
}


plot_data <- function(X, Y, title=""){
  plot(X[, 1], X[, 2], col=unlist(Y), pch=20)
  title(title)
}


test_spectral_single <- function(benchmark, dataset, M=20, k=NULL, scale=FALSE, plot=TRUE){
  data <- read_data(benchmark, dataset)
  X <- data$X
  if(scale){
    X <- scale(X)
  }
  Y <- data$Y
  if(is.null(k)){
    k = length(unique(unlist(Y)))
  }
  set.seed(42)  # because kmeans in spectral clustering randomly initializes centers
  Y_pred <- spectral_clustering(X, k, M)
  if(plot){
    plot_data(X, Y_pred, paste(paste(benchmark, dataset, sep="/"), ": spectral ", sep=""))
  }
  print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
  #return(Y_pred)
}


test_hclust <- function(benchmark, dataset, method="complete", k=NULL, scale=FALSE, plot=TRUE){
```

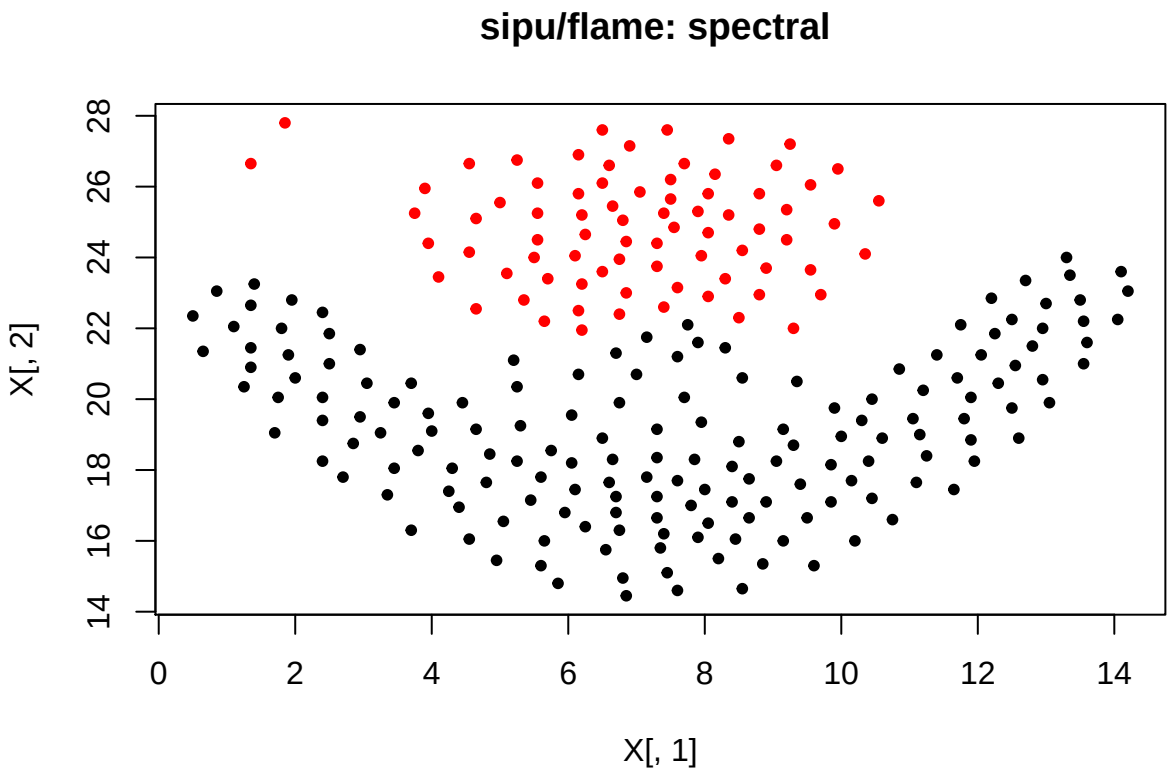
```
  data <- read_data(benchmark, dataset)
  X <- data$X
  if(scale){
    X <- scale(X)
  }
  Y <- data$Y
  if(is.null(k)){
    k = length(unique(unlist(Y)))
  }

  hc <- hclust(dist(X), method)
  Y_pred <- cutree(hc, k=k)
  if(plot){
    plot_data(X, Y_pred, paste(paste(benchmark, dataset, sep="/"), ": hclust ", method, sep=""))
  }
  print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
  #return(Y_pred)
}


test_genie <- function(benchmark, dataset, k=NULL, scale=FALSE, plot=TRUE){
  data <- read_data(benchmark, dataset)
  X <- data$X
  if(scale){
    X <- scale(X)
  }
  Y <- data$Y
  if(is.null(k)){
    k = length(unique(unlist(Y)))
  }

  hc <- hclust2(dist(X))
  Y_pred <- cutree(hc, k=k)
  if(plot){
    plot_data(X, Y_pred, paste(paste(benchmark, dataset, sep="/"), ": genie", sep=""))
  }
  print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
  #return(Y_pred)
}
```

Wstępne testy algorytmów

W tej części przyjrzymy się, jak radzą sobie poszczególne algorytmy. W szczególności zobaczymy jak działa algorytm spectral clustering, który zaimplementowaliśmy oraz porównamy jego działanie z innymi na różnych zbiorach danych.

Przyjrzyjmy się najpierw czy algorytm produkuje jakieś sensowne wyniki. Załadujemy dane ze zbioru sipu/flame. Spójrzmy jak działa.

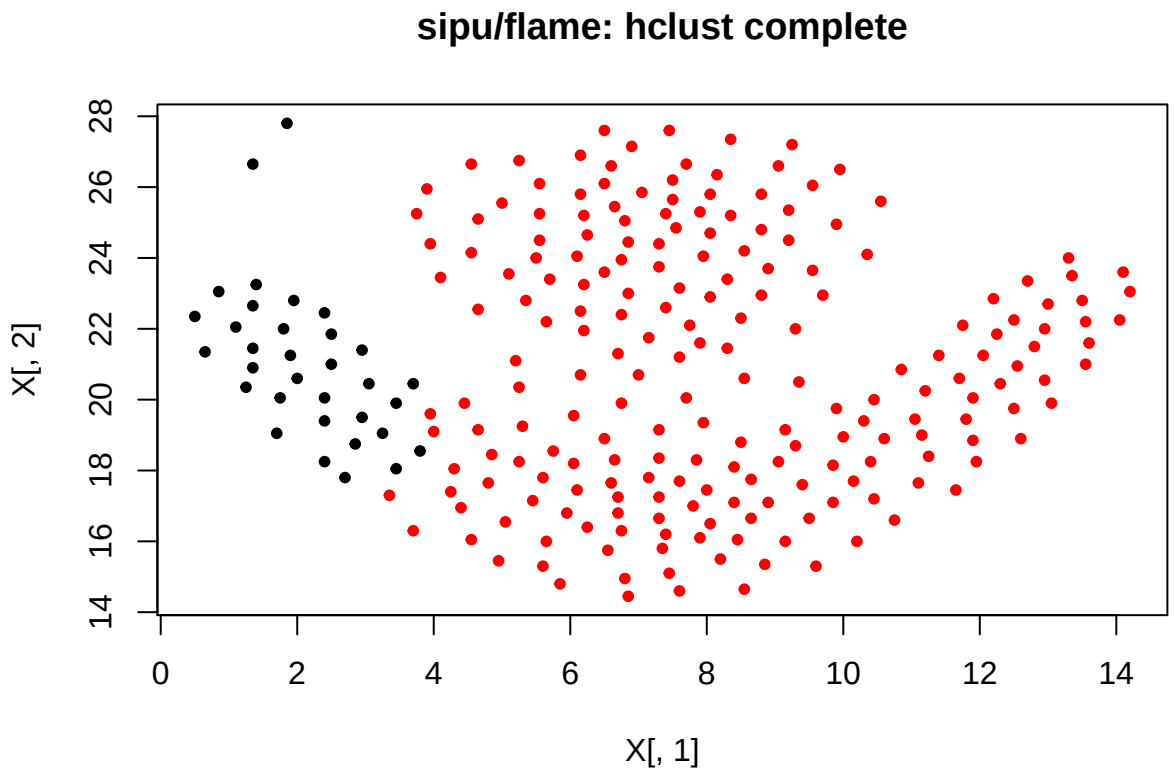
```
test_spectral_single("sipu", "flame", M=20)
```




```
## [1] "FM: 0.927249445551185  AR: 0.838160673829826"
```

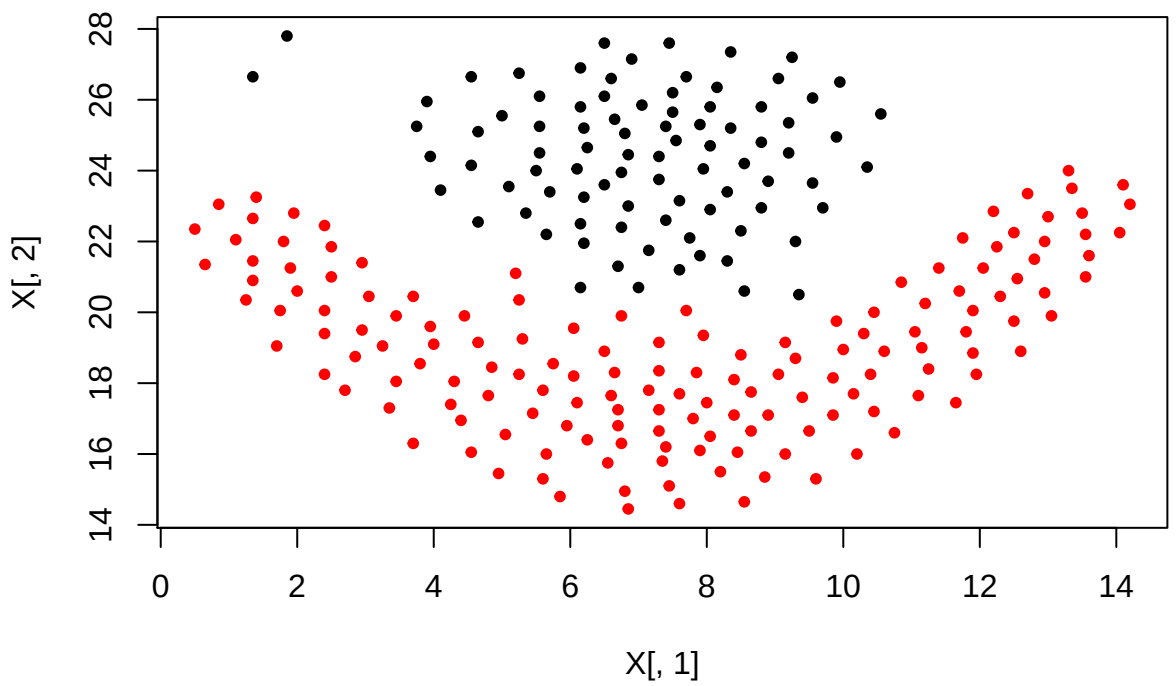
Jak widzimy nie jest źle. Żeby móc wyciągnąć więcej wniosków z wyników porównamy działanie z innymi algorytmami. Następnie przyjrzymy się też wynikom miar jakości analizy skupień indeksowi Fowlkesa–Mallowsa (FM) oraz skorygowanemu indeksowi Randa (AR).

```
test_hclust("sipu", "flame")
```




```
## [1] "FM: 0.623036389947208  AR: -0.0422301552686084"
```

```
test_genie("sipu", "flame")
```




```
## [1] "FM: 1  AR: 1"
```

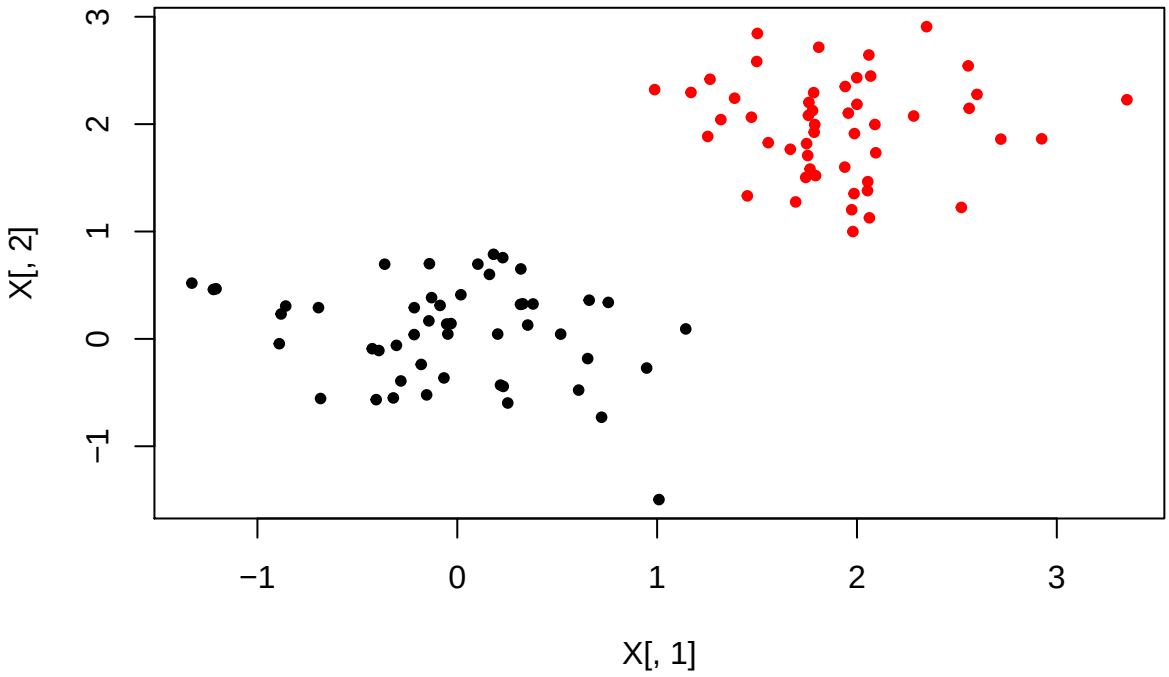
Testy na własnych zbiorach danych

Stworzymy teraz kilka prostych, przykładowych zbiorów danych, na których będziemy mogli przetestować działanie algorytmu. Pierwszy z nich wygenrujemy z rozkładów normalnych o różnych parametrach (parametr rozkładu będzie decydował o klasie). Kolejne będą wygenerowane na podstawie wymyślonych kształtów w 2D. Wszystkie te zbiory zapiszemy w folderze `datasets`.

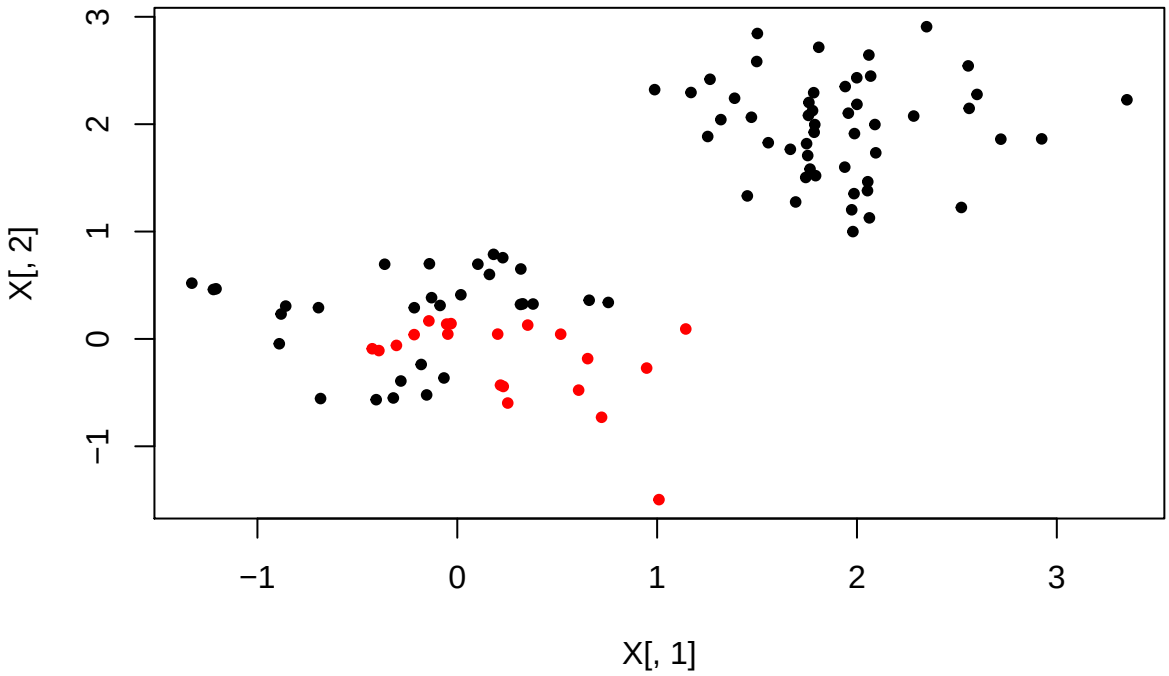
```
# My own datasets
random_dataset <- function(mi1, mi2, sig, n){
  x <- rnorm(n, mi1[1], sig)
  y <- rnorm(n, mi1[2], sig)
  X1 <- cbind(x, y)
  Y1 <- rep(1, n)

  x <- rnorm(n, mi2[1], sig)
  y <- rnorm(n, mi2[2], sig)
  X2 <- cbind(x, y)
  Y2 <- rep(2, n)

  X <- rbind(X1, X2)
  Y <- c(Y1, Y2)
  return(list(X=X, Y=Y))
}

get_heart <- function(n, x_change=0, y_change=0){
  t <- seq(0, 6.29, length.out=n)
  x <- 16 * sin(t)^3
  y <- 13 * cos(t) - 5 * cos(2*t) - 2 * cos(3*t) - cos(4*t)
  return(cbind(x + x_change, y + y_change))
}

save_hearts <- function(n, n_hearts){
  X <- matrix(, nrow=0, ncol = 2)
  Y <- c()

  for(i in 1:n_hearts){
    X1 <- get_heart(n, 20*i, 20*i)
    Y1 <- rep(i, nrow(X1))

    X <- rbind(X, X1)
    Y <- c(Y, Y1)
  }
  file_name <- paste("hearts", n_hearts, sep="_")
  write.table(X, file=file.path("datasets", paste(file_name, "data", sep=".")), row.names=FALSE, col.nar
  write.table(Y, file=file.path("datasets", paste(file_name, "labels0", sep=".")), row.names=FALSE, col
  return(list(X=X, Y=Y))
}
```

Zobaczmy jak wyglądają nasze zbiory i jakie wyniki osiąga na nich spectral clustering.

```
data <- random_dataset(c(0, 0), c(2, 2), 0.5, 50)
X <- data$X
Y <- data$Y
plot_data(X, Y)
```




```
Y_pred <- spectral_clustering(X, 2, 20)
plot_data(X, Y_pred)
```



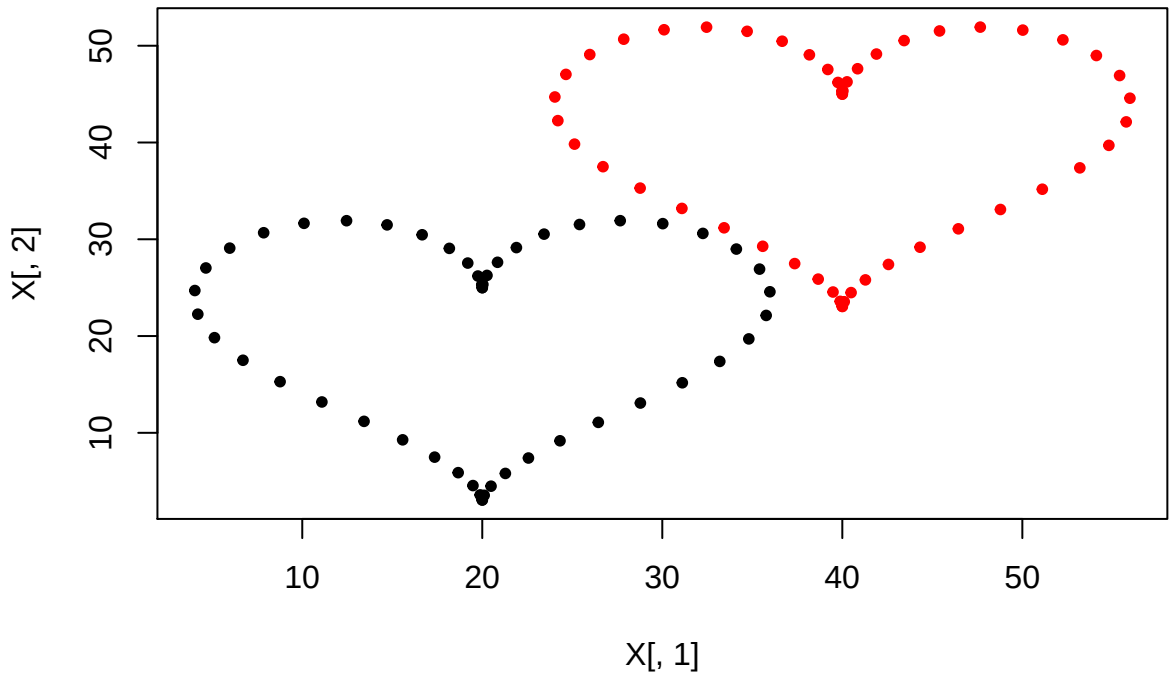

```
print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
```

```
## [1] "FM: 0.645752777311899  AR: 0.154534363562424"
```

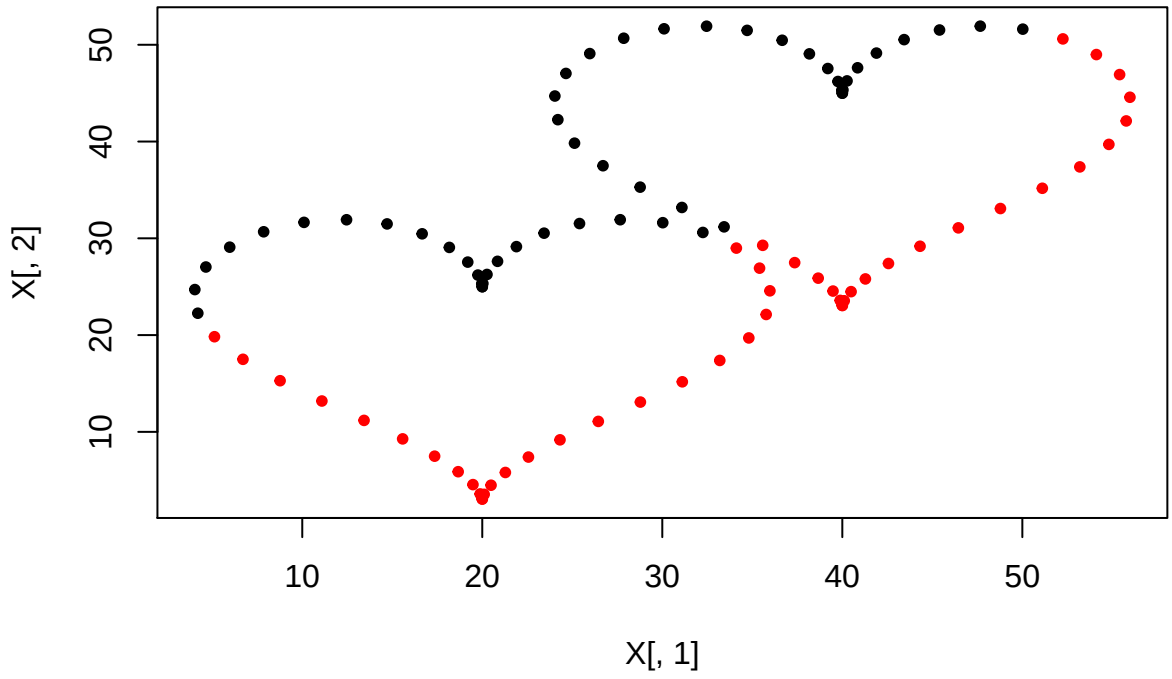
Algorytm jak widzimy nie odróżnia tak dobrze dwóch rozkładów, ale przynajmniej wyniki są dość sensowne. Zapiszmy nasz zbiór danych i przejdźmy do innego.

```
write.table(X, file=file.path("datasets", "normal.data"), row.names=FALSE, col.names=FALSE)
write.table(Y, file=file.path("datasets", "normal.labels"), row.names=FALSE, col.names=FALSE)
```

```
data <- save_hearts(50, 2)
X <- data$X
Y <- data$Y
plot_data(X, Y)
```




```
Y_pred <- spectral_clustering(X, 2)
plot_data(X, Y_pred)  # Calkiem niezle
```



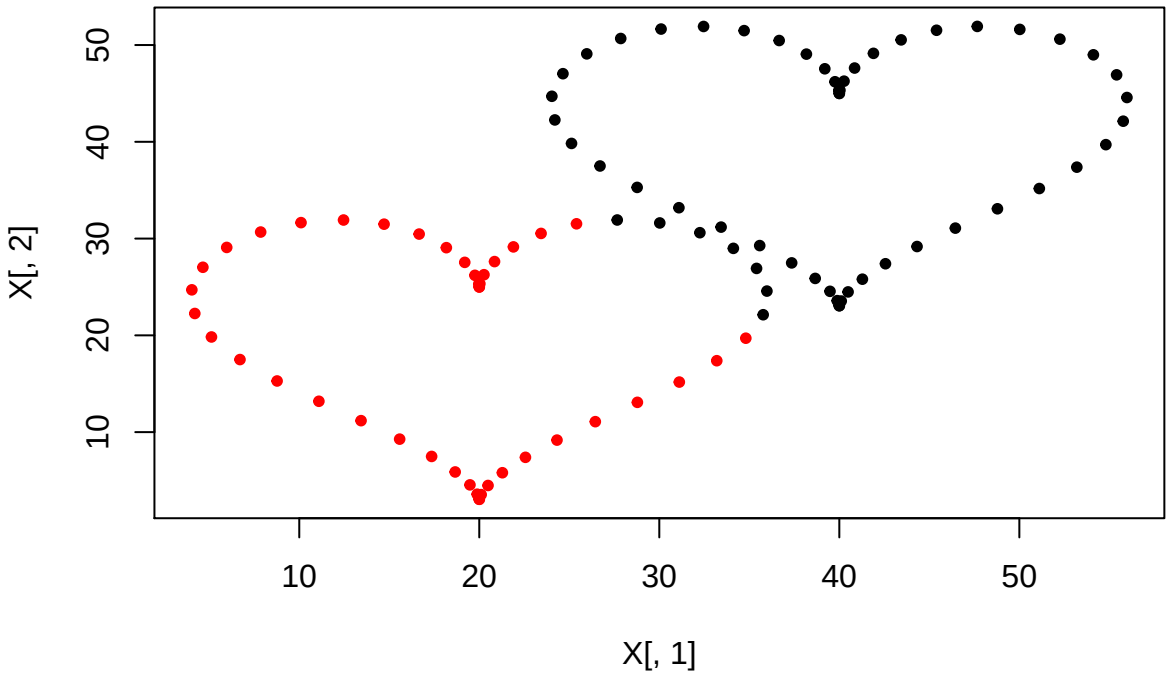

```
print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
```

```
## [1] "FM: 0.493474878673787  AR: -0.00372238820590676"
```

Ciekawą obserwacją może być tutaj fakt, że spectral clustering niekoniecznie wybiera takie skupienia jakie myśleliśmy ze wybierze, ale nie klasyfikuje też do jednego zbioru punktów będących blisko siebie. Znajduje całkiem sensowne skupienia punktów, w którym w jednej klasie są punkty dość od siebie odległe.

Wydaje mi się, że algorytm kmeans działa w przeciwny sposób i punkty w jednym skupieniu są raczej blisko siebie. Sprawdźmy to.

```
Y_pred <- kmeans(X, 2)$cluster
plot_data(X, Y_pred)
```




```
print(paste("FM:", FM_index(Y, Y_pred), " AR:", adjustedRandIndex(Y, Y_pred), sep=" "))
```

```
## [1] "FM: 0.86850084483954  AR: 0.736995458051247"
```

Okazuje się to prawdą. Pokazuje to, że cel w analizie skupień może być niejednoznaczny i przez skupienie każdy może uważać coś innego. Do różnych celów dobre mogą być różne algorytmy. KMeans będzie dobry wtedy, kiedy wiemy, że punkty należące do jednego skupienia będą blisko siebie w przestrzeni euklidesowej.

Zapiszmy na koniec nasze wygenerowane zbiory danych.

```
for(i in 2:10){
  save_hearts(50, i)
}
```