LABORATORIUM 13

- Analiza skupień
- Optymalna liczba skupień
- Klastrowanie hierarchiczne
- Redukcja wymiaru

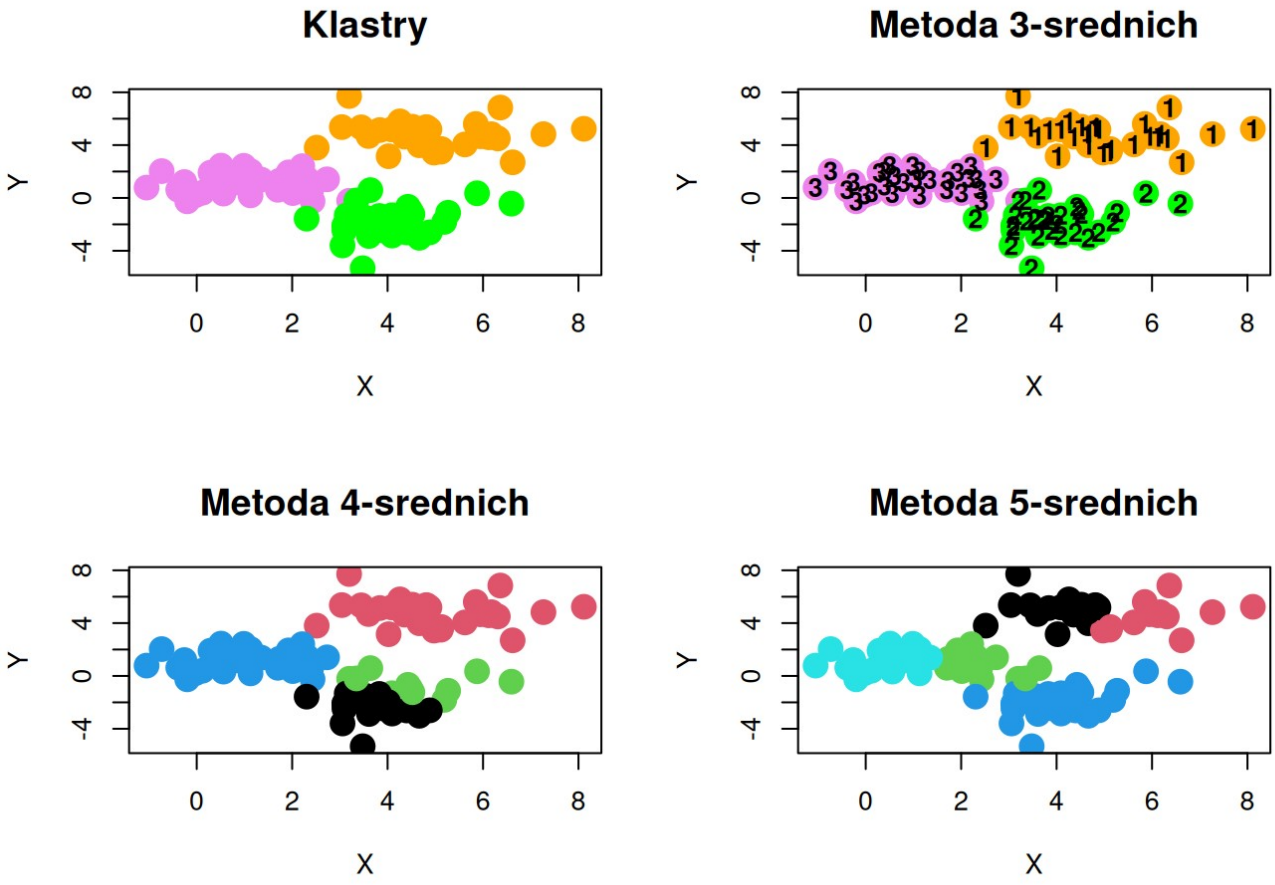
Analiza skupień

Jedną z metod klasyfikacji bez nadzoru jest analiza skupień (klastrów). Jej założenia są następujące:

- Dane jest n-elementowy zbiór obserwacji $\mathbf{x}_i$.
- Chcemy podzielić ten zbiór na K klastrów (skupień).
- Rozpatrujemy sumę kwadratów odległości $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$ pomiędzy wszystkimi parami punktów: $T = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}$.
- Sumę T możemy zapisać jako $T = W + B$, gdzie $W = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} d_{ij}$ jest sumą odległości wewnątrz skupień, a $B = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)\neq k} d_{ij}$ sumą odległości pomiędzy klastrami.
- Zadaniem algorytmu jest minimalizacja W (lub maksymalizacja B).

W praktyce algorytm metody k-średnich inicjalizuje środki K skupień (gdzie K jest parametrem) i w kolejnych iteracjach przypisuje najbliższe punkty do tych skupień oraz wyznacza nowe skupienia itd. aż w układzie nie będzie zachodzić żadna zmiana. Funkcją implementującą taki algorytm w języku R jest `kmeans()`.

```
library(MASS)
set.seed(111)

n <- 30
sigma <- matrix(c(1,0,0,1),2,2)

mu1 <- c(5,5)
mu2 <- c(1,1)
mu3 <- c(4,-2)

kolory <- c(rep("orange", n), rep("violet", n), rep("green", n))

clust1 <- mvrnorm(n, mu1, sigma)
clust2 <- mvrnorm(n, mu2, sigma)
clust3 <- mvrnorm(n, mu3, sigma)

all_points <- rbind (clust1, clust2, clust3)

xrange <- range (all_points[,1])
yrange <- range (all_points[,2])

par(mfrow = c(2,2))

plot(all_points, col=kolory, pch=19, cex=2, xlab="X", ylab="Y", xlim=xrange, ylim=yrange)
title("Klastry", cex.main=1.4, font=2)

cl <- kmeans(all_points, 3)

plot(all_points, col=kolory, pch=19, cex=2, xlab="X", ylab="Y", xlim=xrange, ylim=yrange)
text(all_points[,1], all_points[,2], cl$cluster, font=2)
title("Metoda 3-srednich", cex.main=1.4, font=2)

cl1 <- kmeans(all_points, 4)

plot(all_points, col=cl1$cluster, pch=19, cex=2, xlab="X", ylab="Y", xlim=xrange, ylim=yra
         nge)
title("Metoda 4-srednich", cex.main=1.4, font=2)

cl2 <- kmeans(all_points, 5)

plot(all_points, col=cl2$cluster, pch=19, cex=2, xlab="X", ylab="Y", xlim=xrange, ylim=yra
         nge)
title("Metoda 5-srednich", cex.main=1.4, font=2)
```

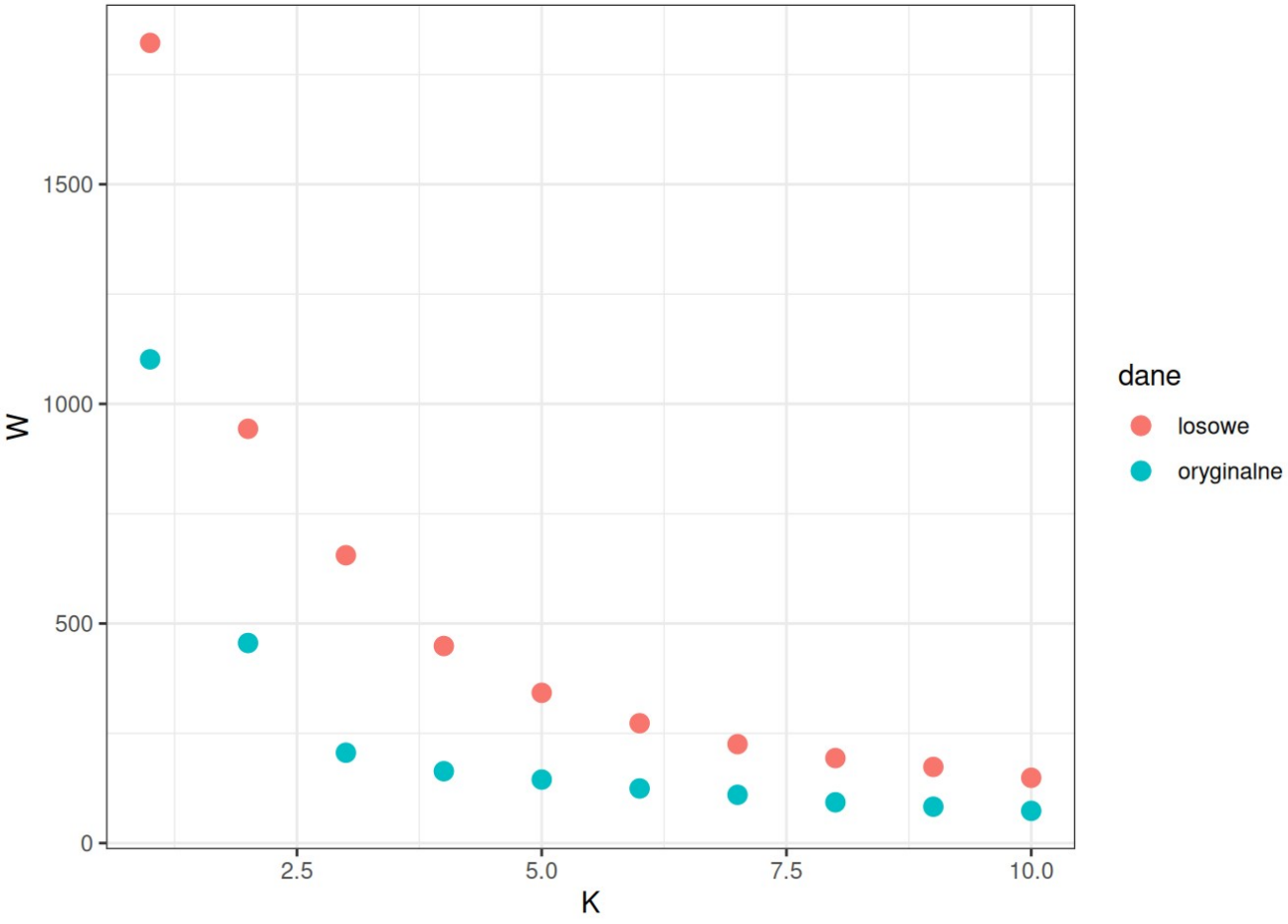


Optymalna liczba skupień

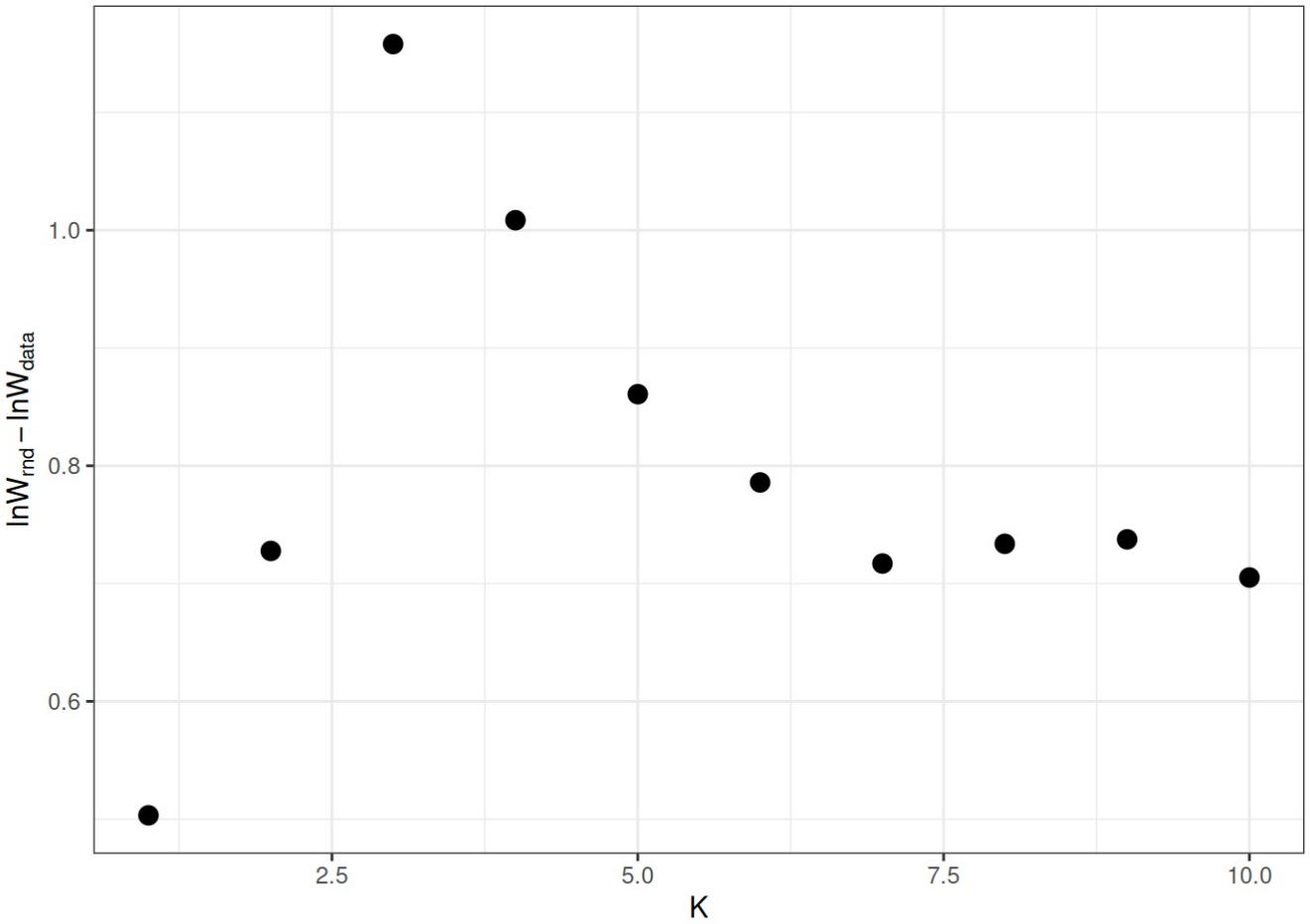
Kluczowym elementem algorytmu k-średnich jest wybór właściwej liczby grup. Samo porównywanie wartości W jest niewystarczające, gdyż będzie ona zawsze malała wraz z liczbą klastrów K. Do wyznaczenia optymalnej liczby skupień K^* wykorzystuje się **statystykę odstępu** (*gap statistics*) $GS = \ln\langle W_{rand}\rangle - \ln\langle W_{data}\rangle$, gdzie $\langle W_{rand}\rangle$ jest uśrednioną po wielu realizacjach sumą odległości wewnątrz skupień obliczoną dla danych losowych z rozkładu jednorodnego o granicach wyznaczonych przez skrajne współrzędne z danych. Wtedy optymalna liczba klastrów dana jest wzorem

$$K^* = \arg\max_K GS(K)$$

```
random_points <- function(n, K, xrange, yrange) {
  x <- runif (n, xrange[1], xrange[2])
  y <- runif (n, yrange[1], yrange[2])
  cl <- kmeans (cbind (x,y), K)
  return (cl$tot.withinss)
}

K <- 1:10
W.rand <- sapply (K, function(k) mean (sapply (1:20, function(i) random_points (3*n, k, xr
         ange, yrange))))
W.data <- sapply (K, function(k) mean (sapply (1:20, function(i) kmeans(all_points, k)$to
         t.withinss)))

W.df <- data.frame (K = K,
                    W.rand = W.rand,
                    W.data = W.data,
                    GS = log(W.rand/W.data))

library(ggplot2)
g <- ggplot(W.df)

g + theme_bw() +
  geom_point ( aes (x=K, y=W.rand, color="losowe"), size=3) +
  geom_point ( aes (x=K, y=W.data, color="oryginalne"), size=3) +
  labs (x = "K", y="W", color="dane")
```



```
g + theme_bw() +
  geom_point ( aes (x=K, y=GS), size=3) +
  labs (x="K", y=expression(lnW[rnd]-lnW[data]))
```



Na powyższym wykresie $GS(K)$ wyraźnie widać, że $K^* = 3$. Policzmy macierz pomyłek i dokładność klasyfikacji dla tej liczby skupień.

```
library(caret)
cluster.predict <- as.factor (cl$cluster)
cluster.true <- as.factor (rep (c(1,3,2), each = n))
cluster.cm <- confusionMatrix (cluster.predict, cluster.true)
cluster.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 30  0  0
##          2  0 30  1
##          3  0  0 29
##
## Overall Statistics
##
##                Accuracy : 0.9889
##                  95% CI : (0.9396, 0.9997)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9833
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   1.0000   0.9667
## Specificity            1.0000   0.9833   1.0000
## Pos Pred Value         1.0000   0.9677   1.0000
## Neg Pred Value         1.0000   1.0000   0.9836
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.3333   0.3333   0.3222
## Detection Prevalence   0.3333   0.3444   0.3222
## Balanced Accuracy      1.0000   0.9917   0.9833
```

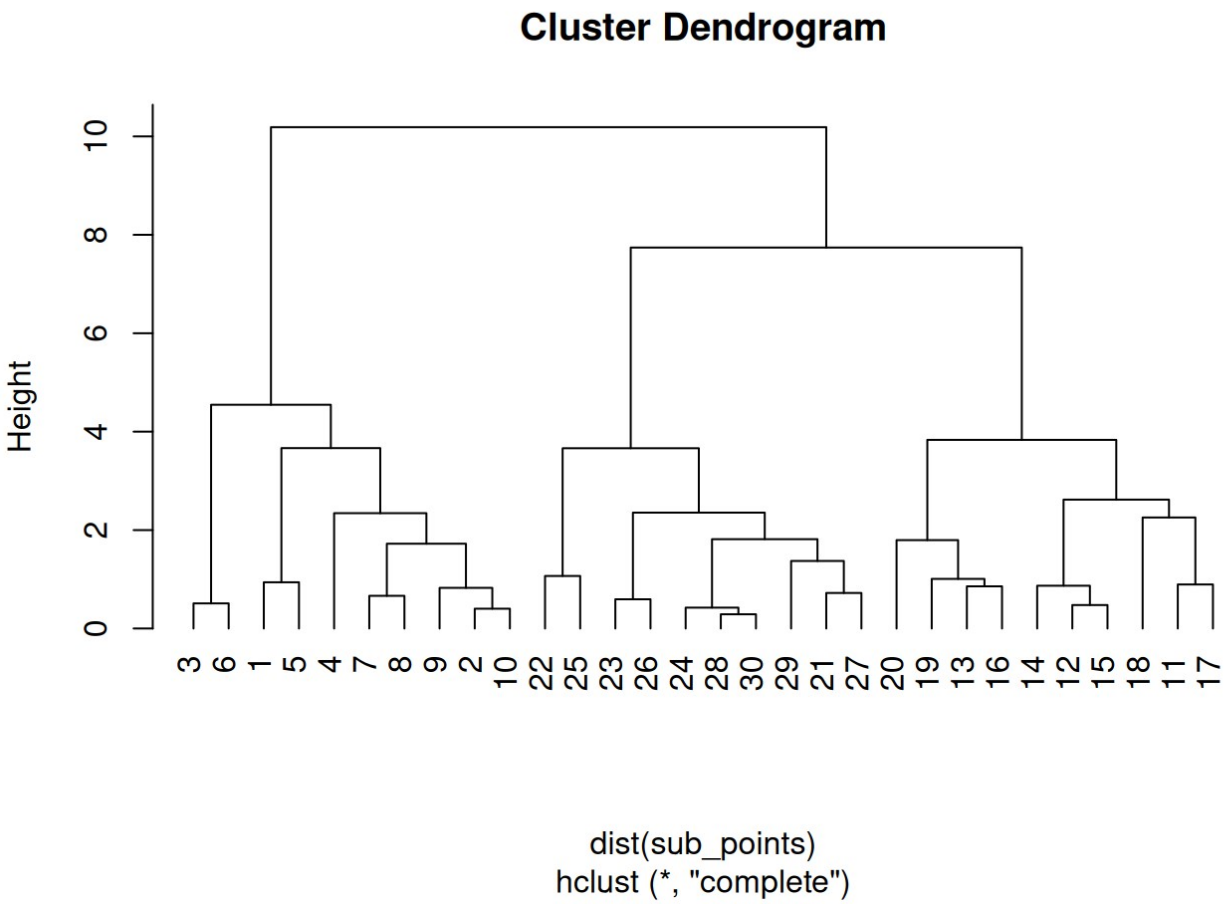
Klastrowanie hierarchiczne

Innym posobem wykonania analizy skupień jest wykorzystanie metod hierarchicznych. Ich ogólny opis można zawrzeć w następujących punktach:

- Opierają sie na pomiarze uogólnionej odmienności między dwoma dowolnymi zbiorami obserwacji.
- Nie wymagają z góry określenia liczby skupień.
- W pierwszym kroku metody algomeracyjnej tworzymy tyle skupień, ile jest obserwacji.
- W nastepnym kroku w jedno skupienie łączona jest para najmniej odległych obserwacji.
- Z kroku na krok skupień jest coraz mniej, aż w ostatnim powstaje cała próba w jednym skupieniu.
- W efekcie otrzymuje się nieskierowane drzewo (dendrogram).

Przy pomocy funkcji `hclust` wykonamy prosty dendrogram dla przerzedzonych danych z poprzednich przykładów

```
sub_points <- rbind (clust1[1:10,], clust2[1:10,], clust3[1:10,])
hc <- hclust (dist (sub_points))
plot(hc, hang = -1)
```



Kluczowe informacje dotyczące odległości, dla których tworzone są klastry oraz łączących się punktów są podane w polach `height` oraz `merge`. Ujemne wartości oznaczają łączone obserwacje, natomiast dodatnie oddają klastry połączone w podanym kroku algorytmu, np. `-24 1`, to połączenie obserwacji nr 17 z klastrem powstałym w pierwszym kroku. Wyboru optymalnej liczby skupień można dokonać metodą wizualną lub wykorzystać jakąś metodę heurystyczną.

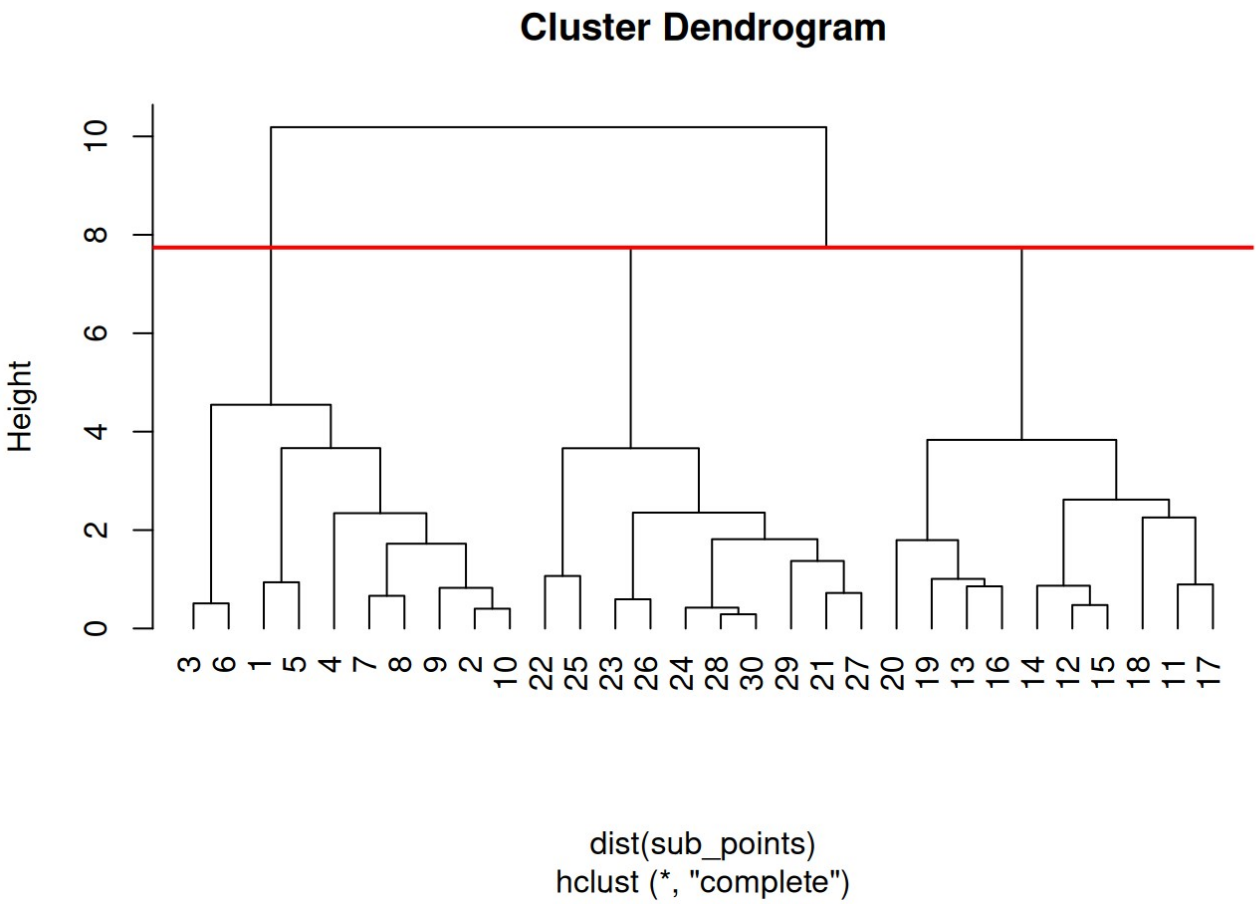
```
hc$height
```

```
##  [1]  0.2893393  0.4030210  0.4246878  0.4772576  0.5104020  0.5946472
##  [7]  0.6635294  0.7216255  0.8254000  0.8572473  0.8693177  0.8955508
## [13]  0.9395204  1.0086786  1.0677816  1.3723851  1.7232404  1.7962182
## [19]  1.8155090  2.2566573  2.3432081  2.3546992  2.6193467  3.6627558
## [25]  3.6655308  3.8337065  4.5464827  7.7416469 10.1880062
```

```
hc$merge
```

```
##       [,1] [,2]
##  [1,]  -28  -30
##  [2,]   -2  -10
##  [3,]  -24    1
##  [4,]  -12  -15
##  [5,]   -3   -6
##  [6,]  -23  -26
##  [7,]   -7   -8
##  [8,]  -21  -27
##  [9,]   -9    2
## [10,]  -13  -16
## [11,]  -14    4
## [12,]  -11  -17
## [13,]   -1   -5
## [14,]  -19   10
## [15,]  -22  -25
## [16,]  -29    8
## [17,]    7    9
## [18,]  -20   14
## [19,]    3   16
## [20,]  -18   12
## [21,]   -4   17
## [22,]    6   19
## [23,]   11   20
## [24,]   15   22
## [25,]   13   21
## [26,]   18   23
## [27,]    5   25
## [28,]   24   26
## [29,]   27   28
```

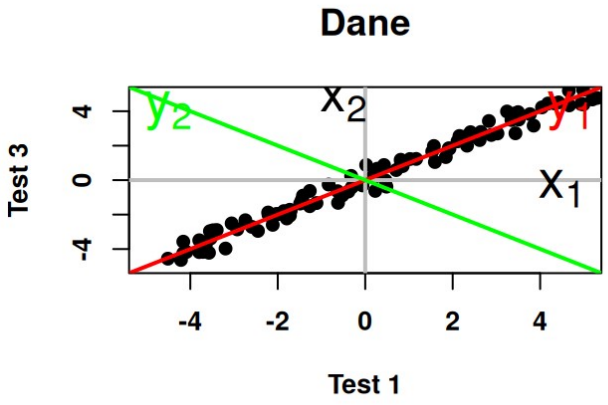
```
height.diff.max <- hc$height[which.max (diff (hc$height))+1]
plot (hc, hang = -1)
abline (h=height.diff.max, col="red", lwd=2)
```

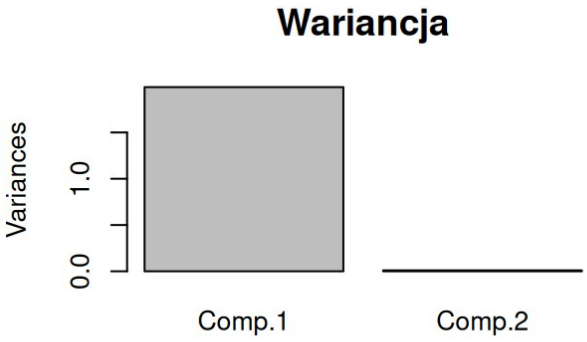


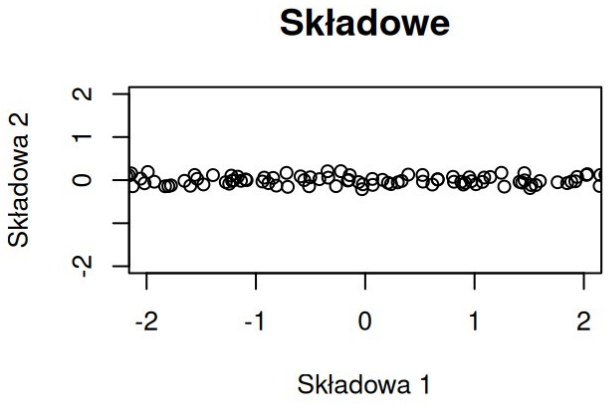
Redukcja wymiaru

Jednym z zadań eksploracji danych jest redukcja wymiaru, czyli określenie, które ze składowych wektora obserwacji są nieistotne lub też jakie inne kombinacje składowych mogą się okazać przydatne do dalszej analizy. Standartową metodą redukcji wymiaru jest **analiza składowych głównych** (**PCA** - *Principal Component Analysis*). Polega ona na znalezieniu nowego kierunku, który maksymalizuje wariancję zrzutowanych na niego obserwacji. Następnie szukamy kolejnego kierunku, również o jak największej wariancji, tyle, że ortogonalnego do poprzedniego itd. Okazuje się, że takie cechy odpowiadają wektorom własnym związanym z kolejnymi wartościami własnymi (począwszy od największej). W poniższym przykładzie mamy do czynienia z “zaszumioną” relacją y=x, alogrytm PCA (funkcja `princomp()`) wykrywa jako nowe kierunki wektory [1,1] i [-1,1] (pole `loadings`). Wartości oryginalnych danych zrzutowane na nowe kierunki otrzymamy za pomocą opcji `scores` , natomiast przeciążona funkcja `plot()` prezentuje wartości wariancji w kolejnych nowych kierunkach.

```
# Przykład 12.6
x <- seq(-5, 5, by=.1)
y <- x

eta <- runif(101, max = 1)
dzeta <- runif(101, max = 1)

x <- x + eta
y <- y + dzeta

par(mfrow = c(2,2))

plot(x, y, pch=19, xlab="Test 1", ylab="Test 3", font=2, font.lab=2, xlim=c(-5,5), ylim=c
         (-5,5))
abline(h=0, v=0, lwd=2, col="gray")
abline(0,1,lwd=2,col="red")
abline(0,-1,lwd=2,col="green")
text(4.5,-0.5,expression(x[1]),cex=2)
text(-0.5,4.5,expression(x[2]),cex=2)
text(4.7,4,expression(y[1]),cex=2, col="red")
text(-4.5,4,expression(y[2]),cex=2, col="green")
title("Dane", cex.main=1.4)

test <- data.frame(x, y)

test.pc <- princomp(~., cor=T, data=test)

plot(test.pc, main="")
title("Wariancja", cex.main=1.4)

plot(test.pc$scores, xlim=c(-2,2), ylim=c(-2,2), xlab="Składowa 1", ylab="Składowa 2")
title("Składowe", cex.main=1.4)
```


Dane
Test 3
Test 1
y2
x2
y1
x1
-4
-2
0
2
4


Wariancja
Variances
0.0
1.0
Comp.1
Comp.2


Składowe
Składowa 2
Składowa 1
-2
-1
0
1
2