Кластеризация

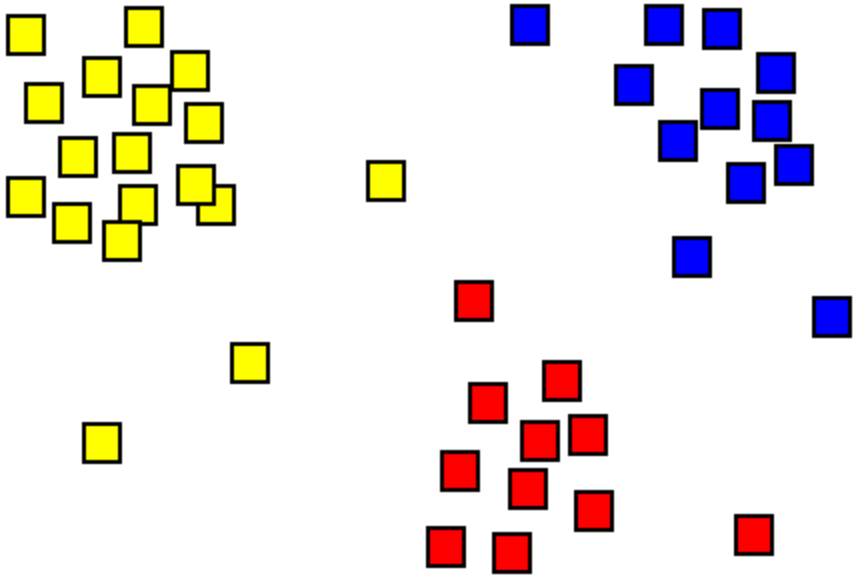
Overview

- Понятие кластеризации
- Меры расстояний
- Классификация алгоритмов

Кластеризация (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами.

Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны.

Например:



Меры расстояний

- ✓ Евклидово расстояние

  Наиболее распространенная функция расстояния. Представляет собой геометрическое расстояние в многомерном пространстве:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x'_i)^2}$$

- ✓ Квадрат евклидова расстояния

  Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x'_i)^2$$

- ✓ Расстояние городских кварталов (манхэттенское расстояние)

Влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат):

$$\rho(x, x') = \sum_{i}^{n} |x_i - x'_i|$$

- ✓ Расстояние Чебышева

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате.

$$\rho(x, x') = \max(|x_i - x'_i|)$$

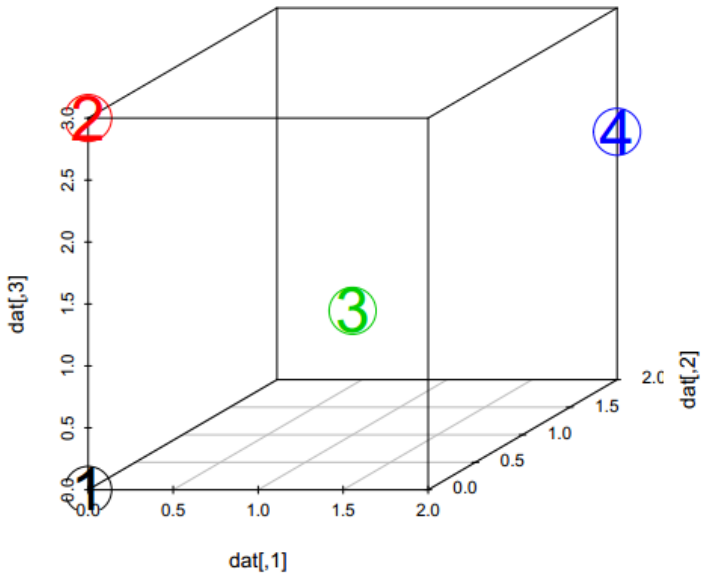
- ✓ Матрицы корреляций

В R есть встроенная функция dist() для подсчета расстояний между строками матрицы:

dist(x, method="euclidean", diag=FALSE, upper=FALSE)

Методы:
"euclidean", "maximum", "manhattan", "canberra", "binary" или "minkowski"

```
dat <- rbind(samp1=c(0,0,0), samp2=c(0,0,3), samp3=c(1,1,1),
samp4=c(2,2,2))
dat
      [,1] [,2] [,3]
samp1    0    0    0
samp2    0    0    3
samp3    1    1    1
samp4    2    2    2
```



```
> dist(dat, method="euclidian")
         samp1    samp2    samp3
samp2 3.000000
samp3 1.732051 2.449490
samp4 3.464102 3.000000 1.732051

> as.matrix(dist(dat, method="euclidian"))
         samp1   samp2    samp3    samp4
samp1 0.000000 3.00000 1.732051 3.464102
samp2 3.000000 0.00000 2.449490 3.000000
samp3 1.732051 2.44949 0.000000 1.732051
samp4 3.464102 3.00000 1.732051 0.000000
```

Какие бывают алгоритмы кластеризации

- ✓ Иерархические алгоритмы строят систему вложенных разбиений.

На выходе мы получаем дерево кластеров, корнем которого является вся выборка, а листьями — наиболее мелкие кластера.

- ✓ Неиерархические («плоские») алгоритмы строят одно разбиение объектов на кластеры.

Алгоритмы иерархической кластеризации

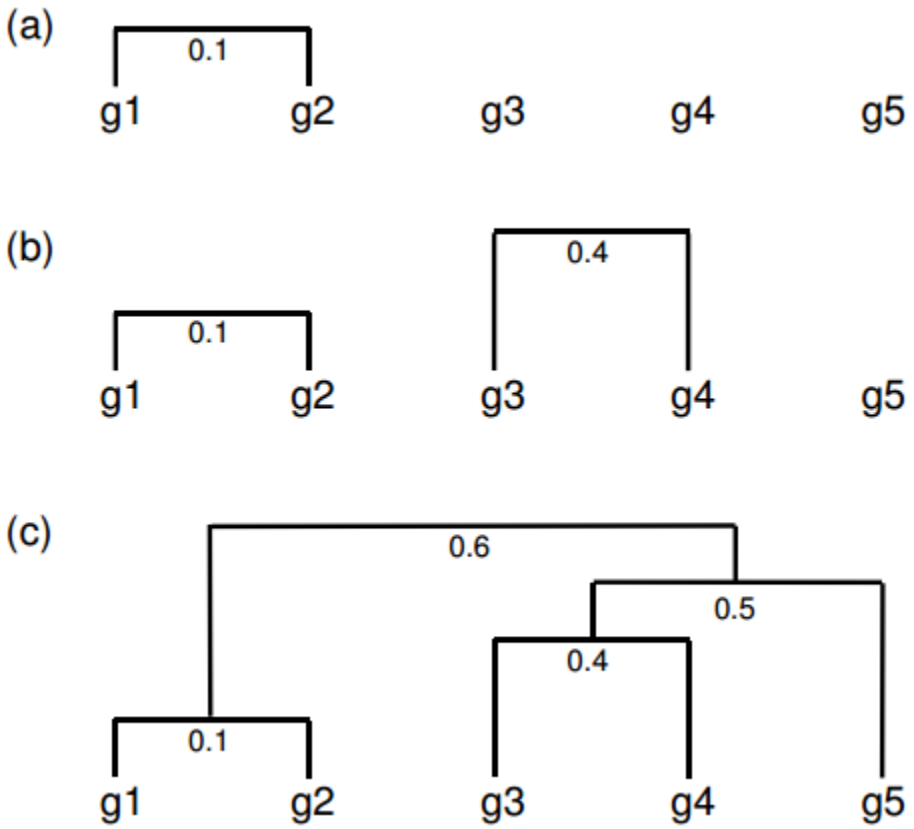
- ✓ Нисходящие алгоритмы: вначале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. top-down

- ✓ Восходящие алгоритмы: вначале помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере. bottom-up

Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы. Классический пример такого дерева – классификация животных и растений.

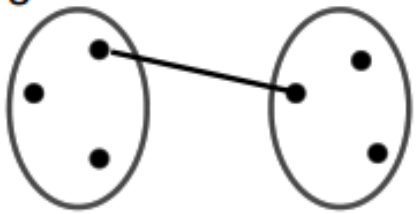
Bottom-up

1. Найти 2 кластера (или объекта) с наименьшим расстоянием между ними
2. Объединить их в один кластер
3. Пересчитать расстояния между кластерами
4. Вернуться к шагу 1



Как вычислять «расстояния» между кластерами?

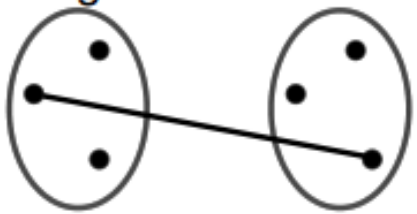
Single Linkage



Одиночная связь (расстояния ближайшего соседа)

Результирующие кластеры имеют тенденцию объединяться в цепочки.

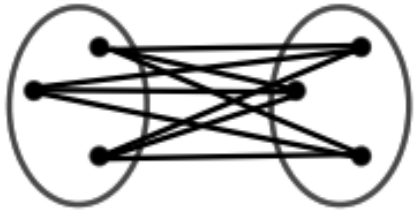
Complete Linkage



Полная связь (расстояние наиболее удаленных соседей)

работает очень хорошо, когда объекты происходят из отдельных групп

Average Linkage



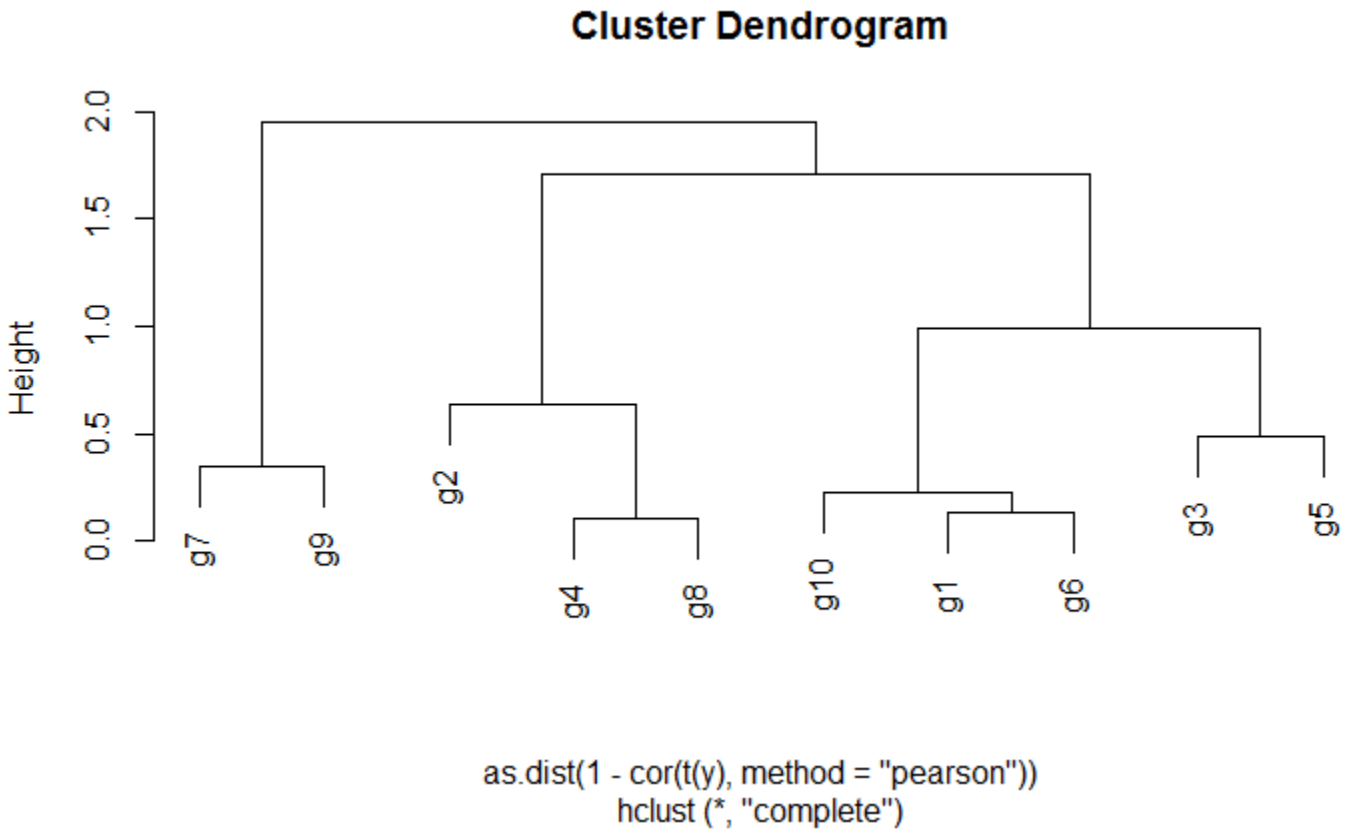
Взвешенное (**WPGMA** - *weighted* pair group method with averaging)

(AB) *и* ***C+(DE)*** *= (55 + 90) / 2 = 72.5*

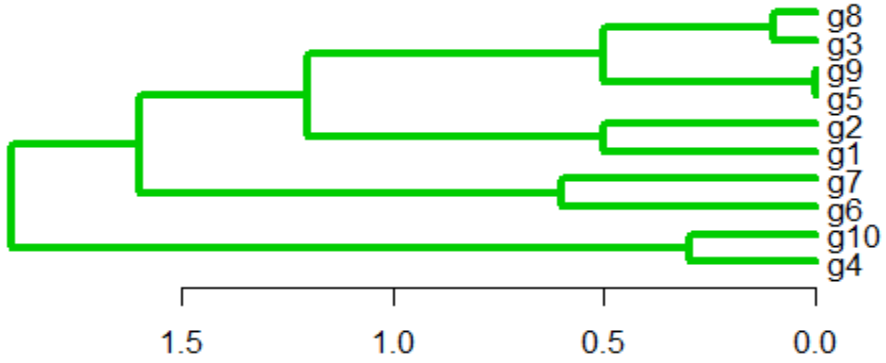
или невзвешенное (**UPGMA** - *unweighted* PGMA) попарное среднее:

(AB) *и* ***C+(DE)*** *= (55 + **2**x90) / 3 = 78.33*

```
y <- matrix(rnorm(50), 10, 5, dimnames=list(paste("g", 1:10, sep=""), paste("t", 1:5, sep=""))) # генерируем данные
c <- cor(t(y), method="spearman"); d <- as.dist(1-c)
hr <- hclust(d, method = "complete", members=NULL)
plot(hr)
```



plot(as.dendrogram(hr), edgePar=list(col=3, lwd=4), horiz=T)




```
str(as.dendrogram(hr))
--[dendrogram w/ 2 branches and 10 members at h = 1.9]
  |--[dendrogram w/ 2 branches and 2 members at h = 0.3]
  |   |--leaf "g4"
  |   `--leaf "g10"
  `--[dendrogram w/ 2 branches and 8 members at h = 1.6]
     |--[dendrogram w/ 2 branches and 2 members at h = 0.6]
     |   |--leaf "g6"
     |   `--leaf "g7"
     `--[dendrogram w/ 2 branches and 6 members at h = 1.2]
        |--[dendrogram w/ 2 branches and 2 members at h = 0.5]
        |   |--leaf "g1"
        |   `--leaf "g2"
        `--[dendrogram w/ 2 branches and 4 members at h = 0.5]
           |--[dendrogram w/ 2 branches and 2 members at h = 2.22e-16]
           |   |--leaf "g5"
           |   `--leaf "g9"
           `--[dendrogram w/ 2 branches and 2 members at h = 0.1]
              |--leaf "g3"
              `--leaf "g8"
```

Выдача в виде скобочной структуры:

library(ctc); hc2Newick(hr)

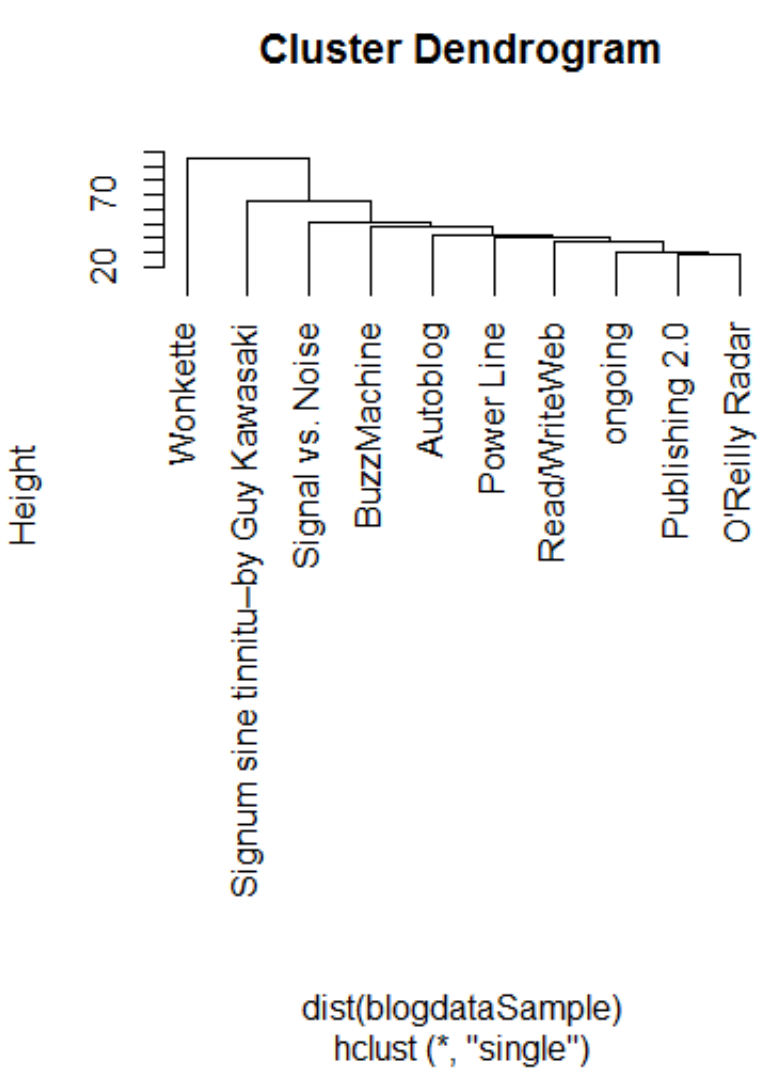
"((g7:0.174726218492745,g9:0.174726218492745):0,((g2:0.26766991502818,
(g4:0.0523033203513996,g8:0.0523033203513996):0.26766991502818):0,
((g10:0.0444120439304247,(g1:0.0680271558561211,g6:0.0680271558561211):
0.0444120439304247):0,(g3:0.243351630808965,g5:0.243351630808965):0):0):0);"

Анализ частоты встречаемости слов в блогах

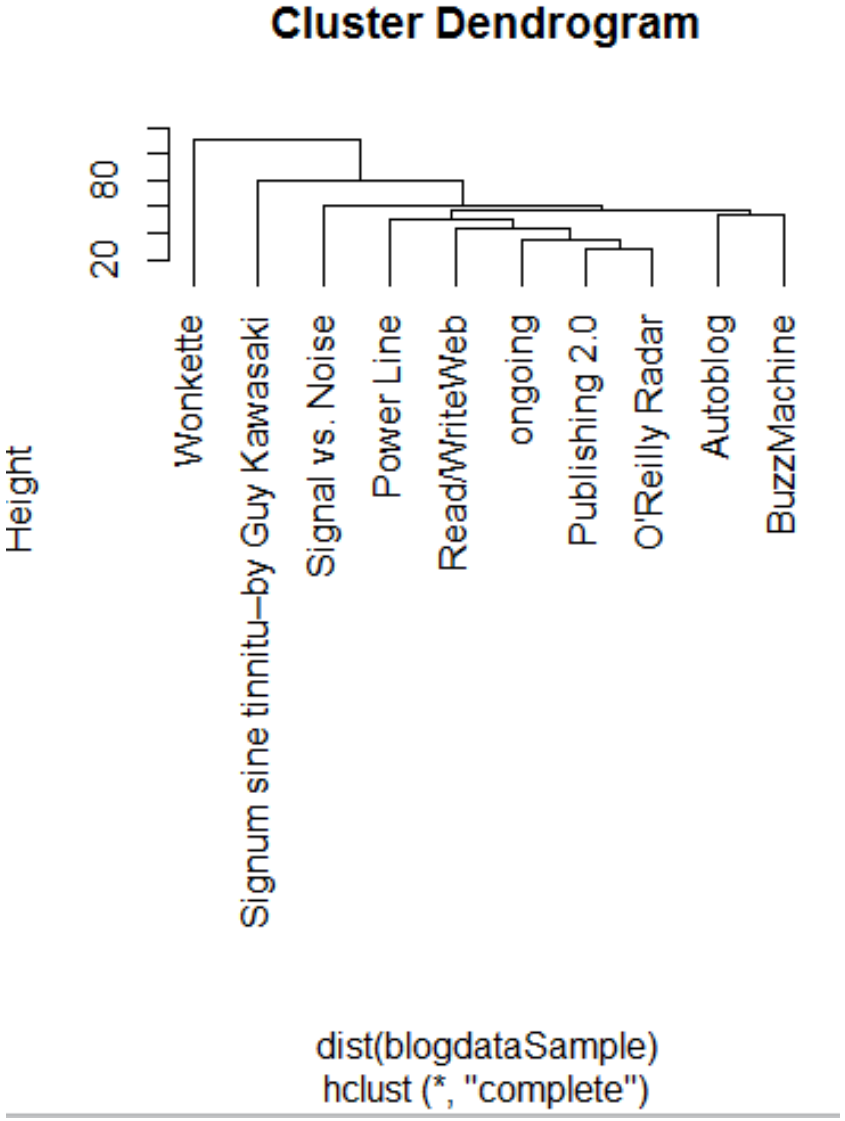
```
idx <- sample(1:dim(blogdata)[1], 10)
blogdataSample <- blogdata[idx,]
blogdataSample$Blog <- NULL
hc <- hclust(dist(blogdataSample), method="single")
plot(hc, hang=-1, labels=blogdata$Blog[idx])
```

	Blog	china	kids	music	yahoo	want	wrong	service	tech	saying
8	GigaOM	6	0	0	2	1	0	3	1	0
63	456 Berea Street	0	0	0	0	10	3	1	0	0
3	Publishing 2.0	0	0	7	4	0	1	3	6	0
88	Derek Powazek	0	1	0	0	0	0	0	1	0
14	Online Marketing Report	0	0	0	3	0	0	0	0	0
66	TechEBlog	1	0	4	0	1	0	0	0	0
71	Dave Shea's mezzoblue	0	0	0	0	0	1	0	0	0
27	Gizmodo	2	0	6	1	0	0	0	2	0
79	Joi Ito's Web	0	0	4	2	0	0	3	0	0
65	Pharyngula	0	0	0	0	0	0	0	0	0

Данные можно скачать: http://kiwitobes.com/clusters/blogdata.txt



hc <- hclust(dist(blogdataSample), method="single")



hc <- hclust(dist(blogdataSample), method="complete")

Еще полезные команды

- ✓ cutree(tree, k = NULL, h = NULL)

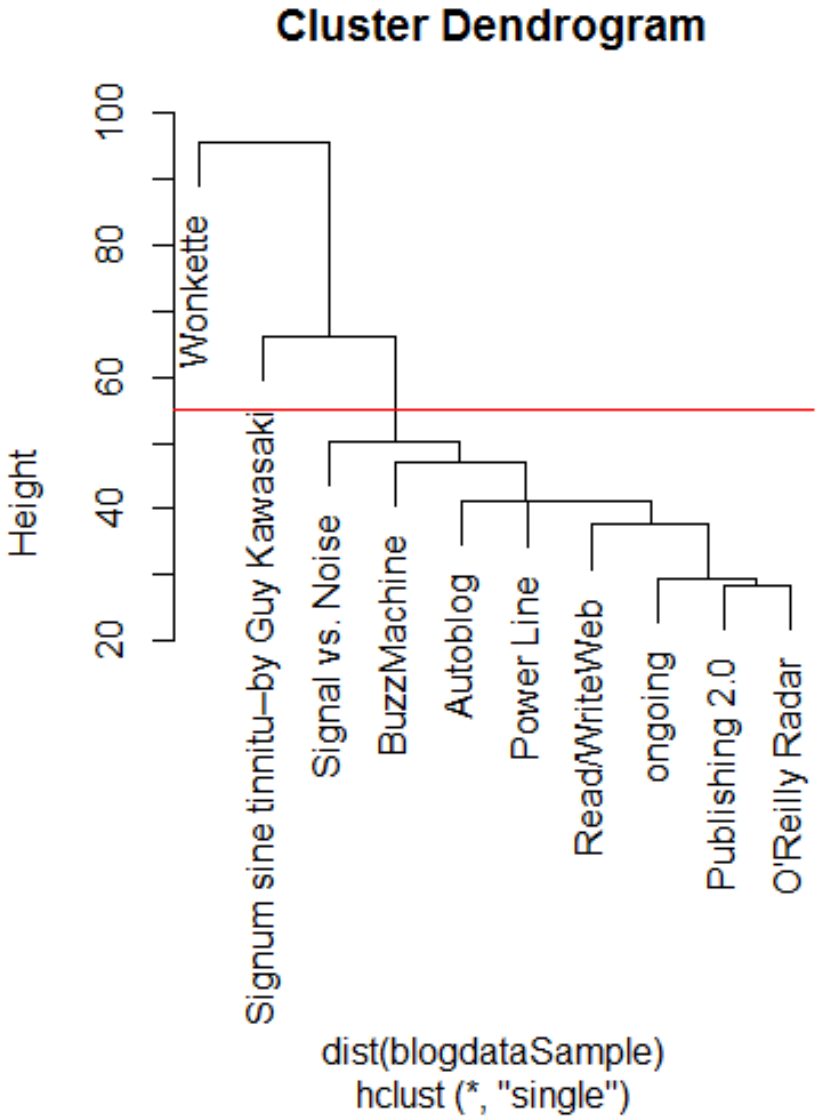
Обрезает дендрограмму так, чтобы получилось k кластеров или по определенной высоте h

- ✓ heatmap()

Строит heatmap, сверху и слева – дендрограмма для кластеризации по столбцам и по строкам.

Обрезаем дерево

```
mycl <- cutree(hc, k=3)
plot(hc); abline(h=55, col="red")
```

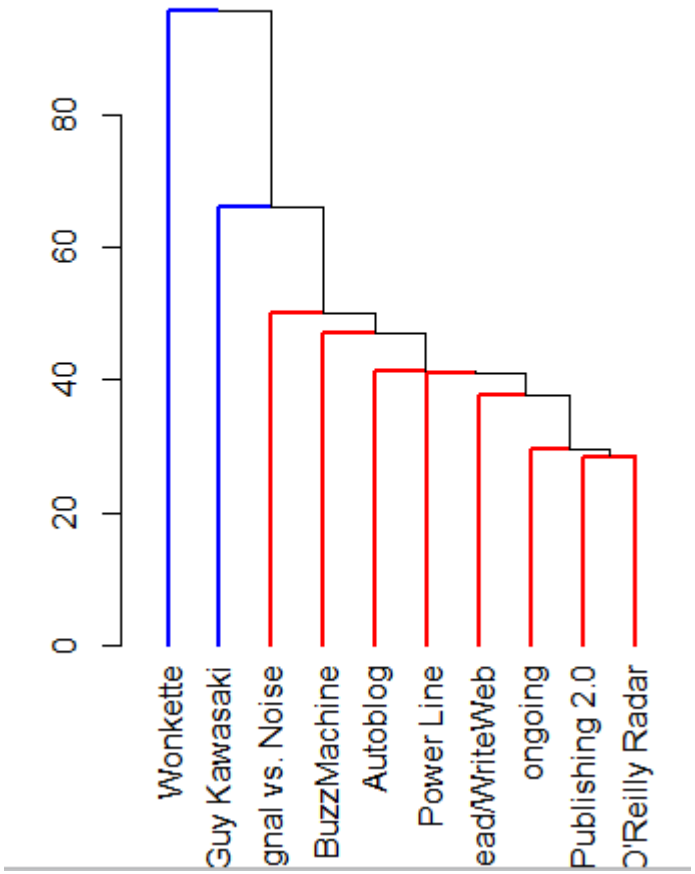


Дополнительные функции:

library(dynamicTreeCut)
help(cutreeDynamic)

Красим ветки

```
dend_colored <- dendrapply(as.dendrogram(hc), dendroCol,
keys=c("Wonkette","Signum sine tinnitu--by Guy Kawasaki"), xPar="edgePar",
bgr="red", fgr="blue", lwd=2, pch=20)
plot(dend_colored)
```



source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/dendroCol.R")

Важно

- Даже для полностью случайных данных на выходе будет кластеризация
- Алгоритм «жадный», без итеративности, выборы, сделанные на ранних этапах будут сильно влиять на итог
- Дендрограмма не уникальна для каждой кластеризации: левые и правые ветви можно менять местами
- В R функция hclust() помещает более «плотные» кластеры на левую часть дендрограммы (облегчает восприятие)

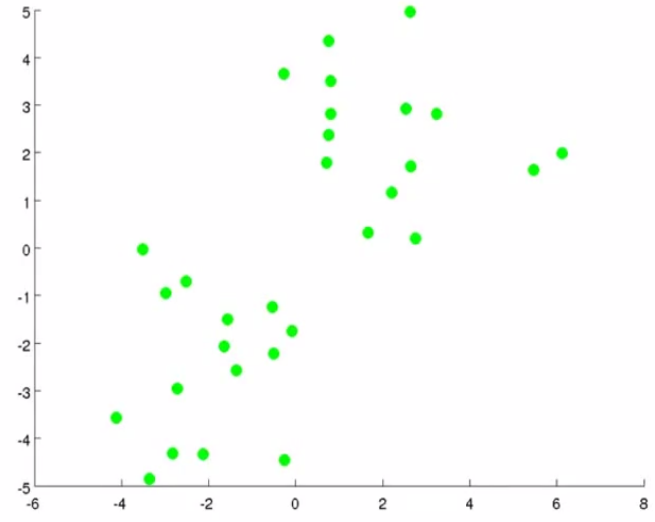
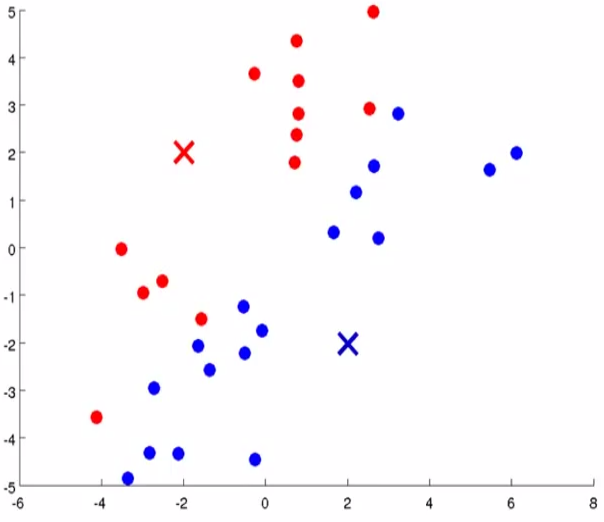
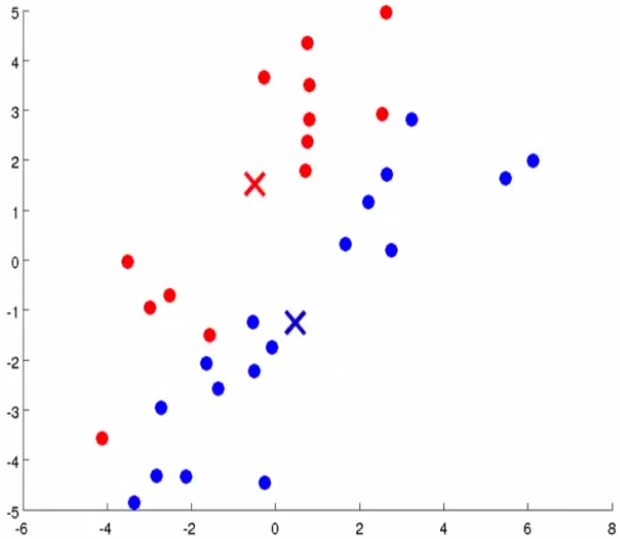
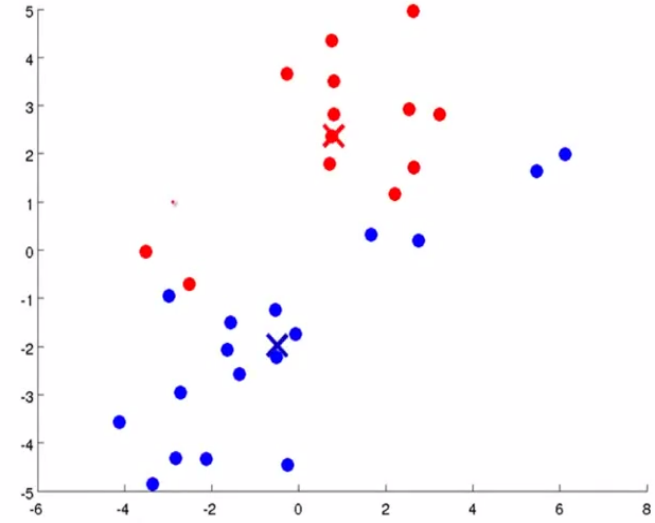
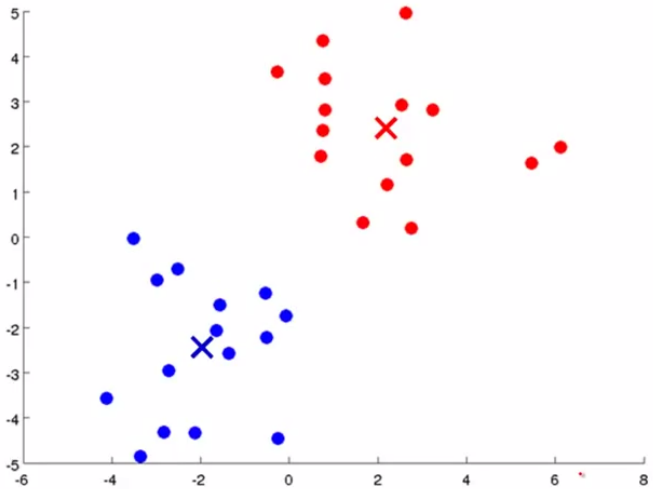
Неиерархическая кластеризация

- ✓ K-Means
- ✓ Principal Component Analysis
- ✓ Multidimensional Scaling
- ✓ Biclustering
- ✓ Many Additional Techniques

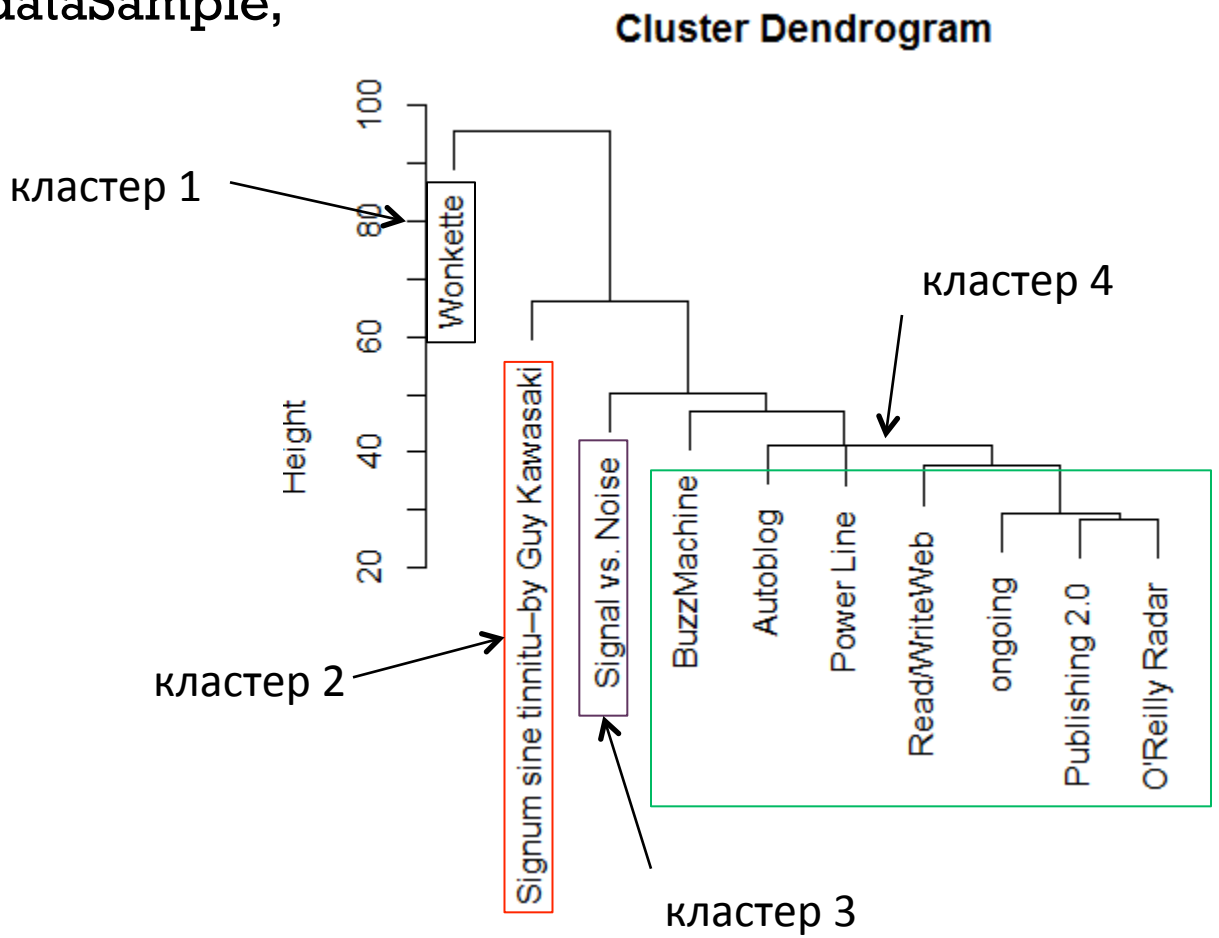
K-means

1. Выбираем количество кластеров (k)
2. Случайно разделяем точки на k кластеров
3. Рассчитываем «центр» для каждого кластера
4. Рассчитываем расстояния от каждой точки до каждого центра
5. Помещаем точку в кластер, к центру которого она ближе всего
6. Повторяем до тех пор, пока точки не перестанут перемещаться между кластерами

kmeans() из stats package,
kcca() из flexclust package
trimkmeans() из trimcluster package.

```
km <- kmeans(blogdataSample,
4)
```

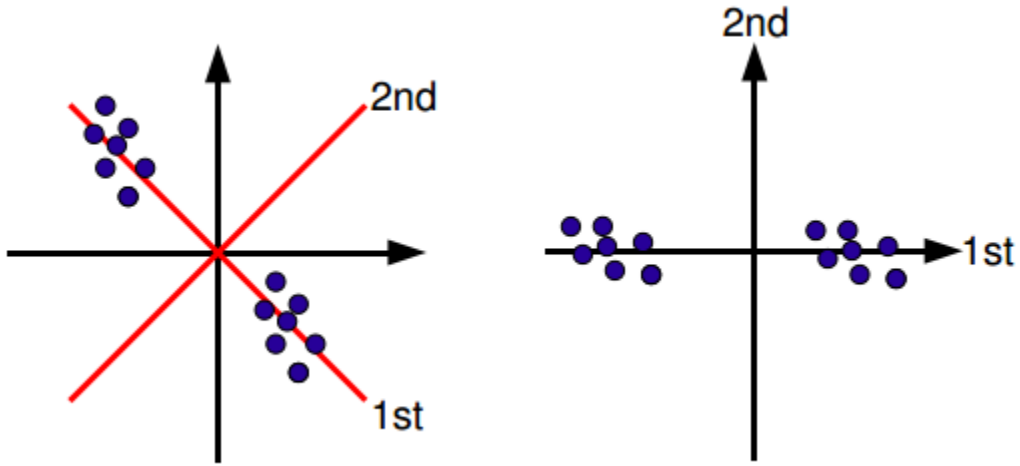



```
Clustering vector:
                ongoing                  Wonkette
                      4                         1
               Autoblog            Read/WriteWeb
                      4                         4
          Signal vs. Noise           Publishing 2.0
                      3                         4
        O'Reilly Radar  Signum sine tinnitu--by Guy Kawasaki
                      4                         2
            BuzzMachine                 Power Line
                      4                         4
```

Метод главных компонент
(*Principal component analysis, PCA*)

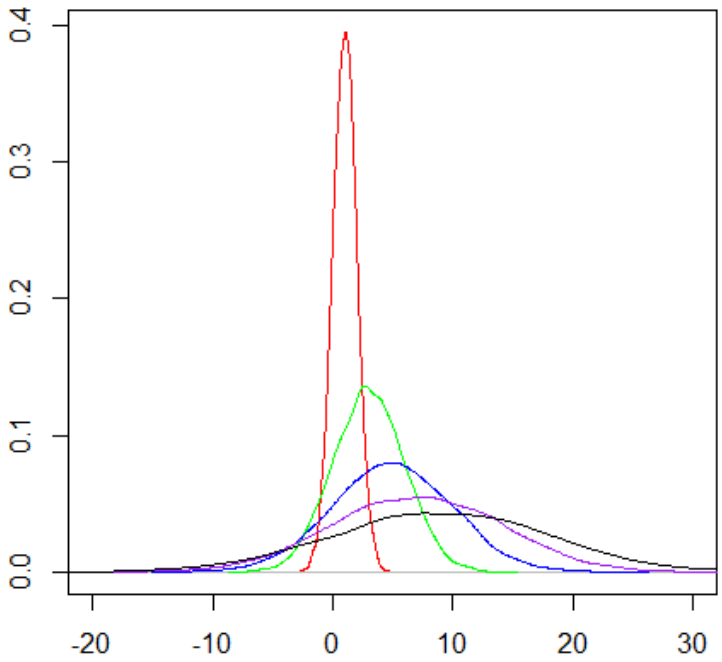
- ✓ один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации
- ✓ Преобразование некоторого набора возможно скоррелированных переменных в набор линейно нескоррелированных переменных – главных компонент
- ✓ Число главных компонент всегда меньше или равно количеству начальных переменных
- ✓ Первая компонента имеет наибольшую возможную дисперсию (т.е. объясняет вариабильность данных настолько, насколько это возможно)
- ✓ Каждая следующая компонента имеет наибольшую дисперсию при условии, что она ортогональна предыдущим

PCA – по сути ортогональное линейное преобразование, переводящее данные в новую систему координат.



Principal Component	**1st**	**2nd**	**3rd**	**Other**
Proportion of Variance	62%	34%	3%	rest

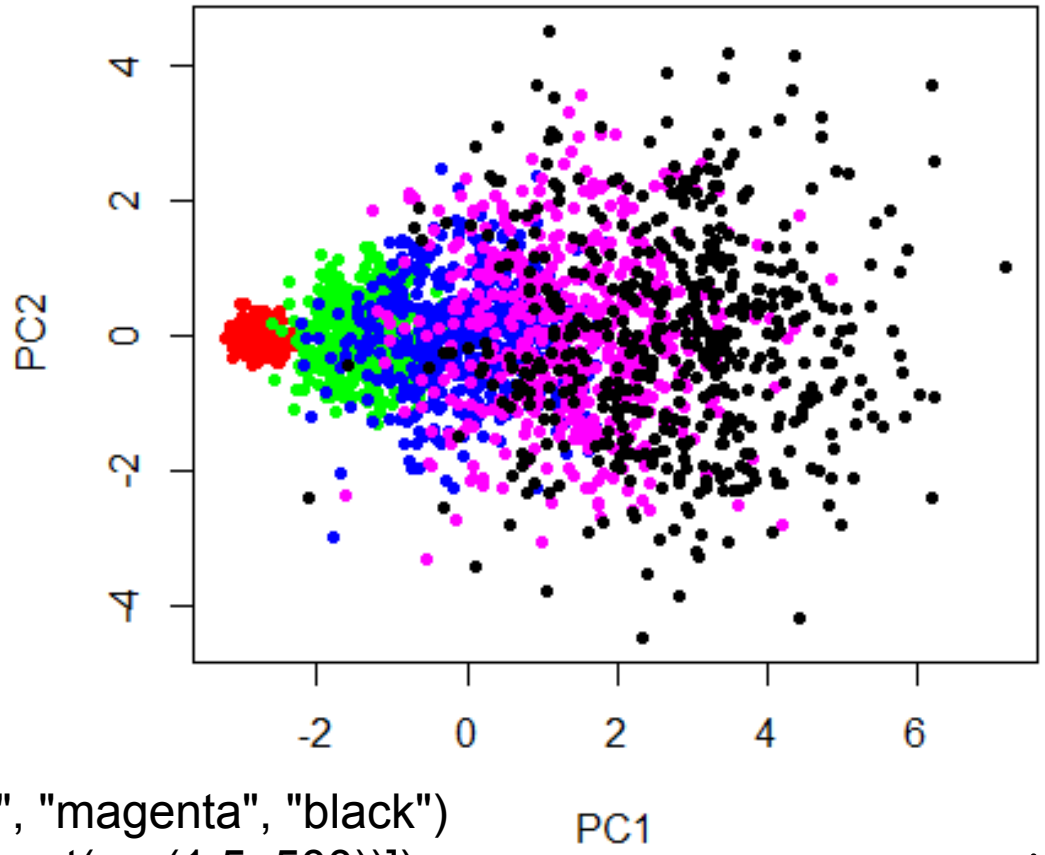
1st and 2nd principal components explain 96% of variance.



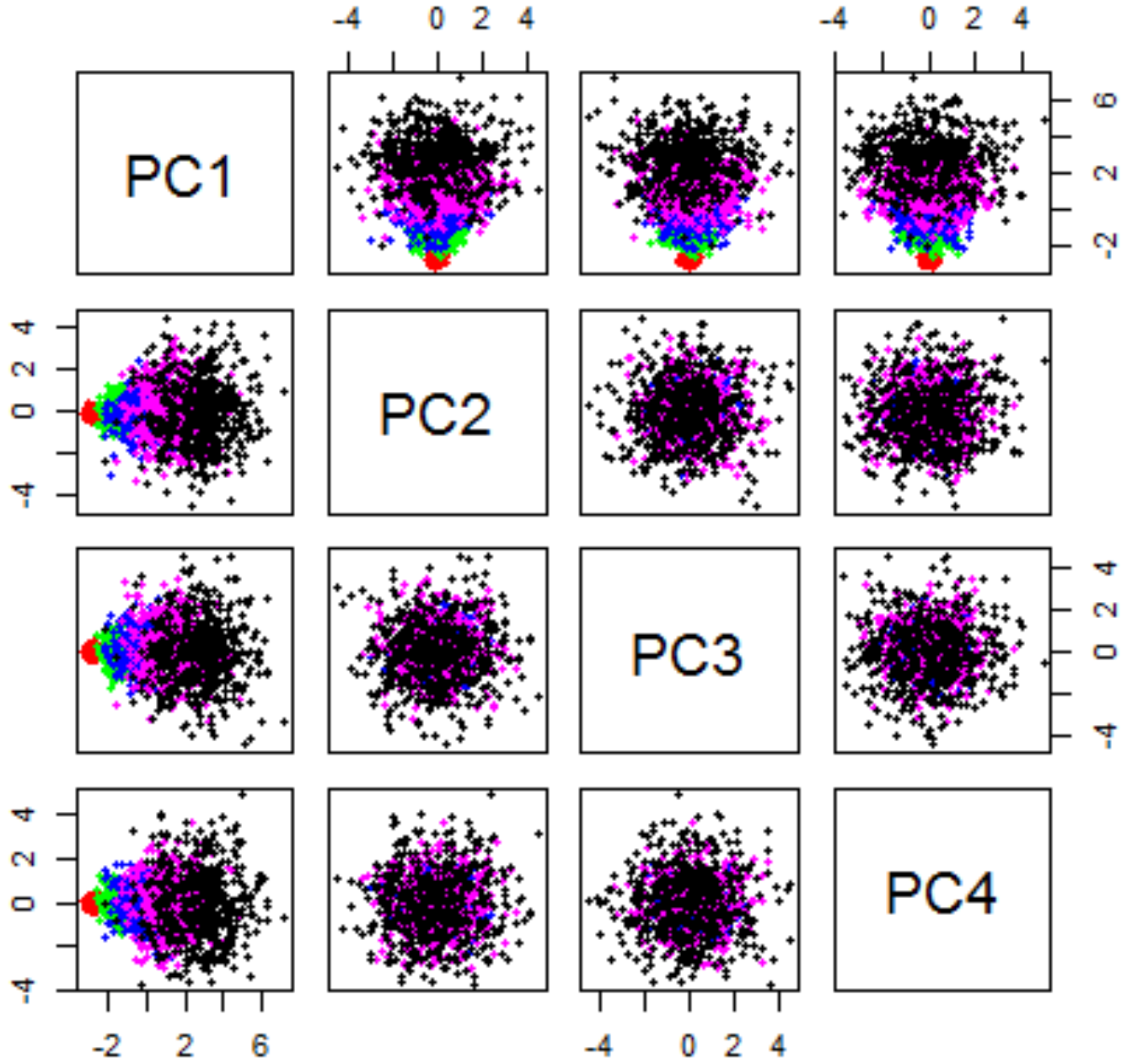

```
z1 <- rnorm(10000, mean=1, sd=1)
z2 <- rnorm(10000, mean=3, sd=3)
z3 <- rnorm(10000, mean=5, sd=5)
z4 <- rnorm(10000, mean=7, sd=7)
z5 <- rnorm(10000, mean=9, sd=9)
mydata <- matrix(c(z1, z2, z3, z4, z5), 2500, 20, byrow=T, dimnames=list(paste("R",
1:2500, sep=""), paste("C", 1:20, sep="")))
  # генерируем матрицу, 2500 строк и 20 столбцов

pca <- prcomp(mydata, scale=T)
```

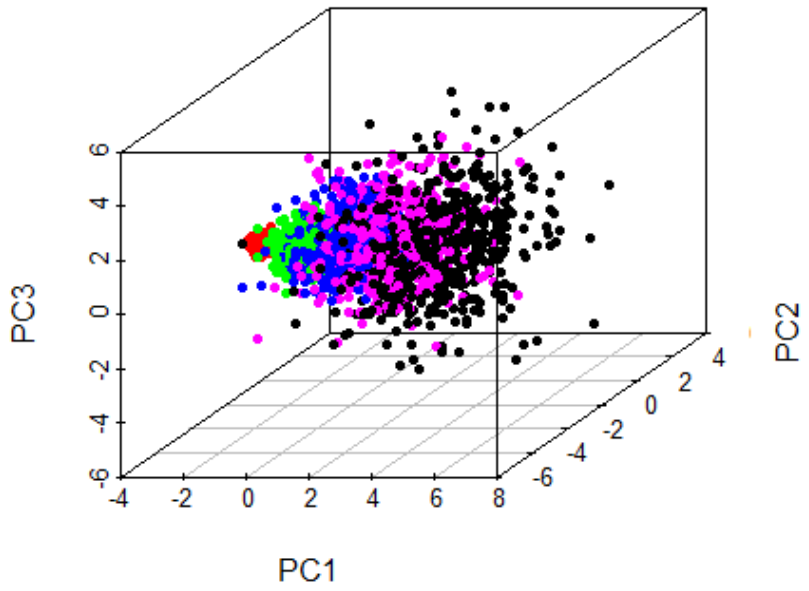
summary(pca) # Выводит на экран summary для всех компонент
summary(pca)$importance[, 1:6] # для первых шести компонент

```
                            PC1       PC2       PC3       PC4       PC5       PC6
Standard deviation     2.151885 0.9901776 0.9758579 0.9586941 0.9454844 0.9412403
Proportion of Variance 0.231530 0.0490200 0.0476100 0.0459500 0.0447000 0.0443000
Cumulative Proportion  0.231530 0.2805500 0.3281700 0.3741200 0.4188200 0.4631200
```



mycolors <- c("red", "green", "blue", "magenta", "black")
plot(pca$x, pch=20, col=mycolors[sort(rep(1:5, 500))])


PC1
PC2
PC3
PC4
-4 0 2 4
-4 0 2 4
-2 0 2 6
-4 0 2 4
-4 0 2 4
-4 0 2 4
-2 2 6
-4 0 2 4



pca$x[,1:3]

- здесь содержатся коэффициенты для перехода в новую систему координат, по первым 3 компонентам: можно использовать для дальнейшей кластеризации

Многомерное шкалирование

- ✓ На входе - попарные сходства/различия анализируемых объектов (матрица расстояний)

- ✓ На выходе числовые значения координат, которые приписываются каждому объекту в некоторой новой системе координат, с сохранением (насколько это возможно) попарных расстояний между объектами

```
loc <- cmdscale(eurodist)
# MDS для географических расстояний между европейскими городами
plot(loc[,1], -loc[,2], type="n", xlab="", ylab="", main="cmdscale(eurodist)")
text(loc[,1], -loc[,2], rownames(loc), cex=0.8)
# строим график, минус нужен для того, чтобы перевернуть график в нужную ориентацию
```