Занятие 9
Кластеризация

10 ноября 2017

Елена Ставровская

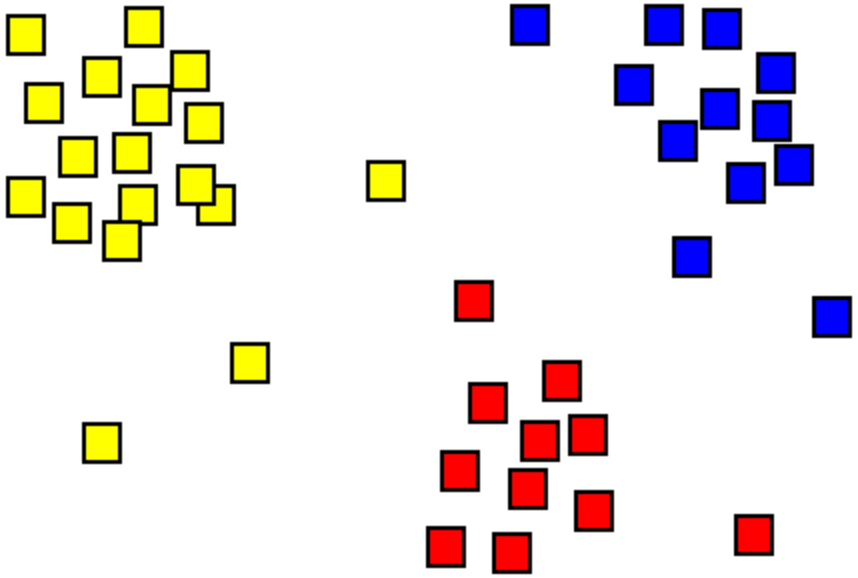
Overview

- 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.

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

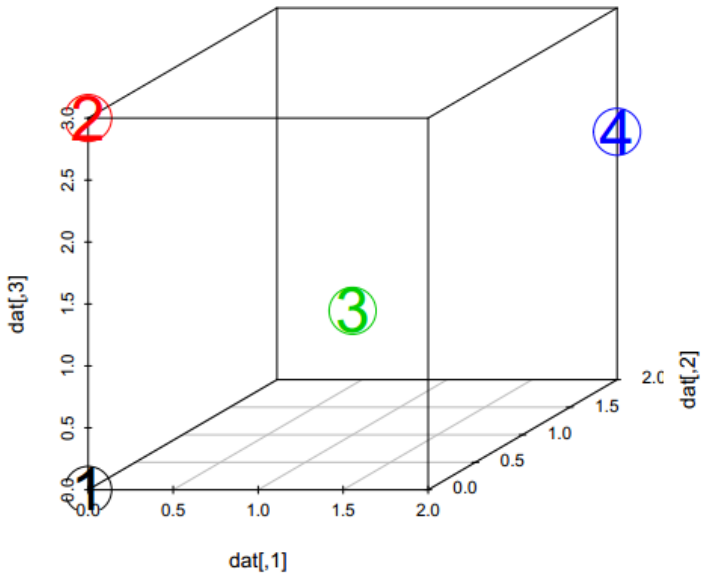
- ✓ Матрицы корреляций

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

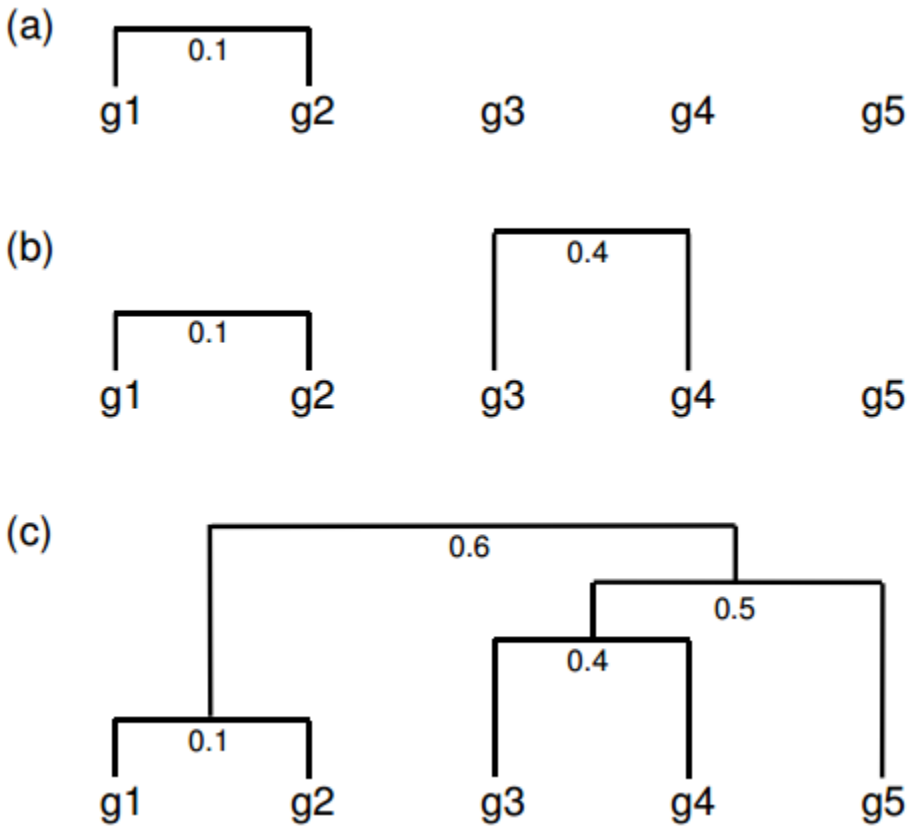
- ✓ Нисходящие алгоритмы: вначале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. top-down

- ✓ 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.

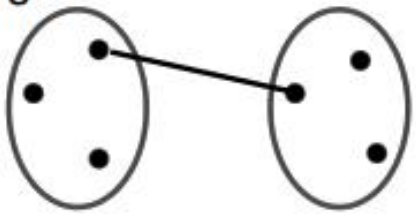
Bottom-up

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?

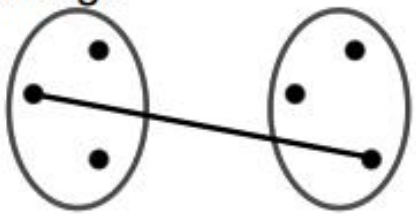
Single Linkage



Одиночная связь (расстояния ближайшего соседа)

Результирующие кластеры имеют тенденцию объединяться в цепочки.

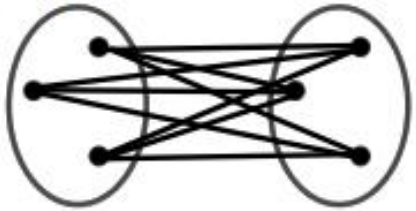
Complete Linkage



Полная связь (расстояние наиболее удаленных соседей)

работает очень хорошо, когда объекты происходят из отдельных групп

Average Linkage



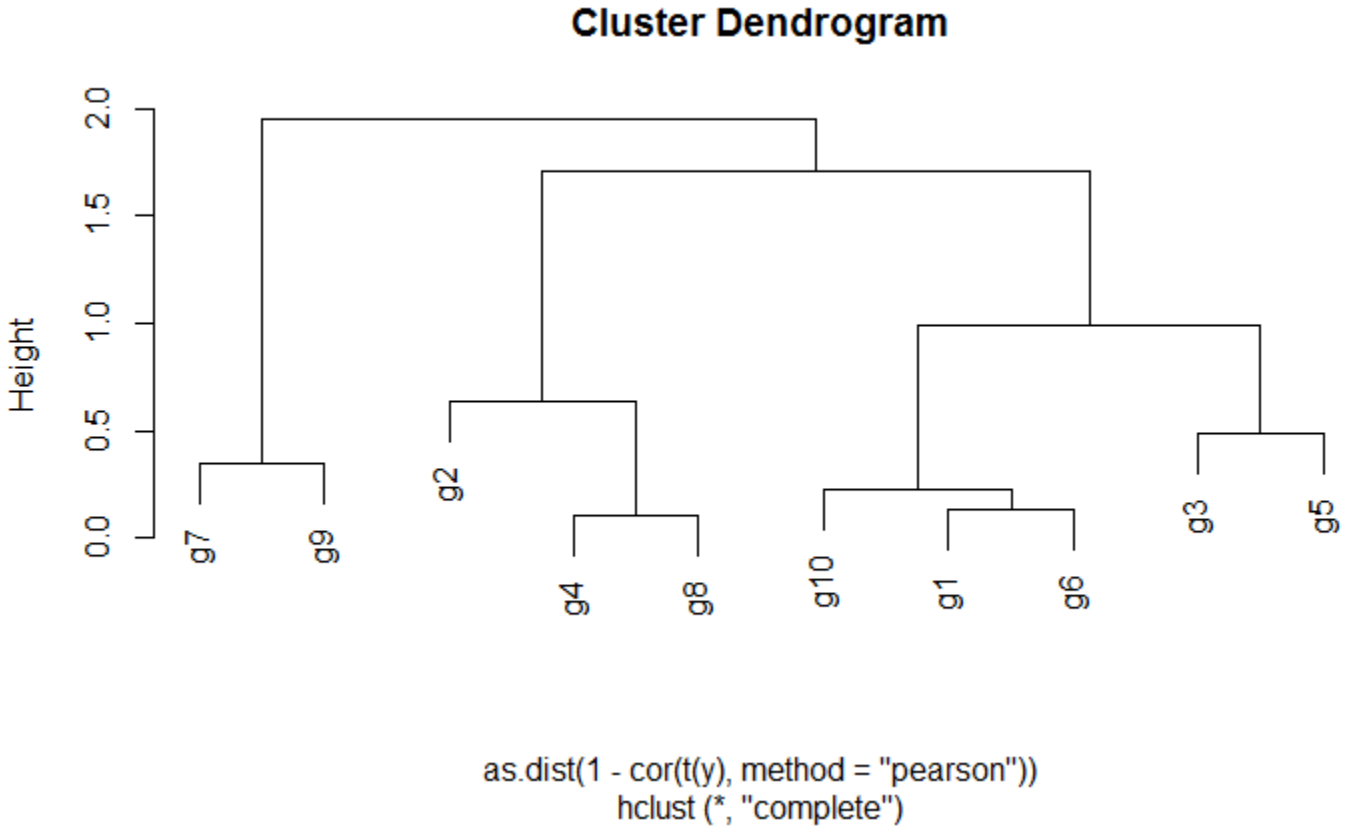
Взвешенное (**WPGMA** - *weighted* pair group method with averaging)

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

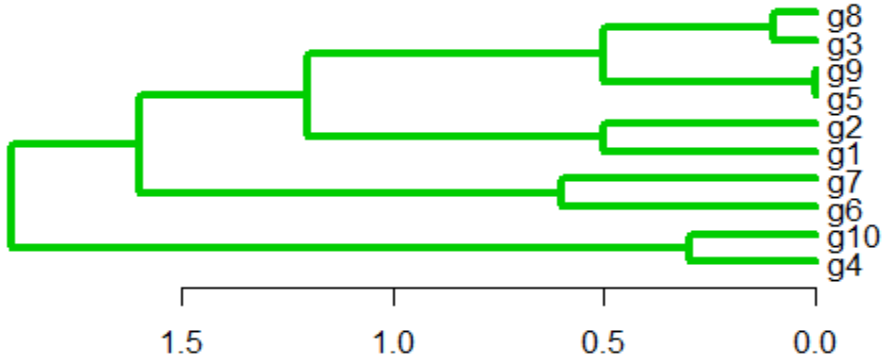
или невзвешенное (**UPGMA** - *unweighted* PGMA) попарное среднее:

(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:

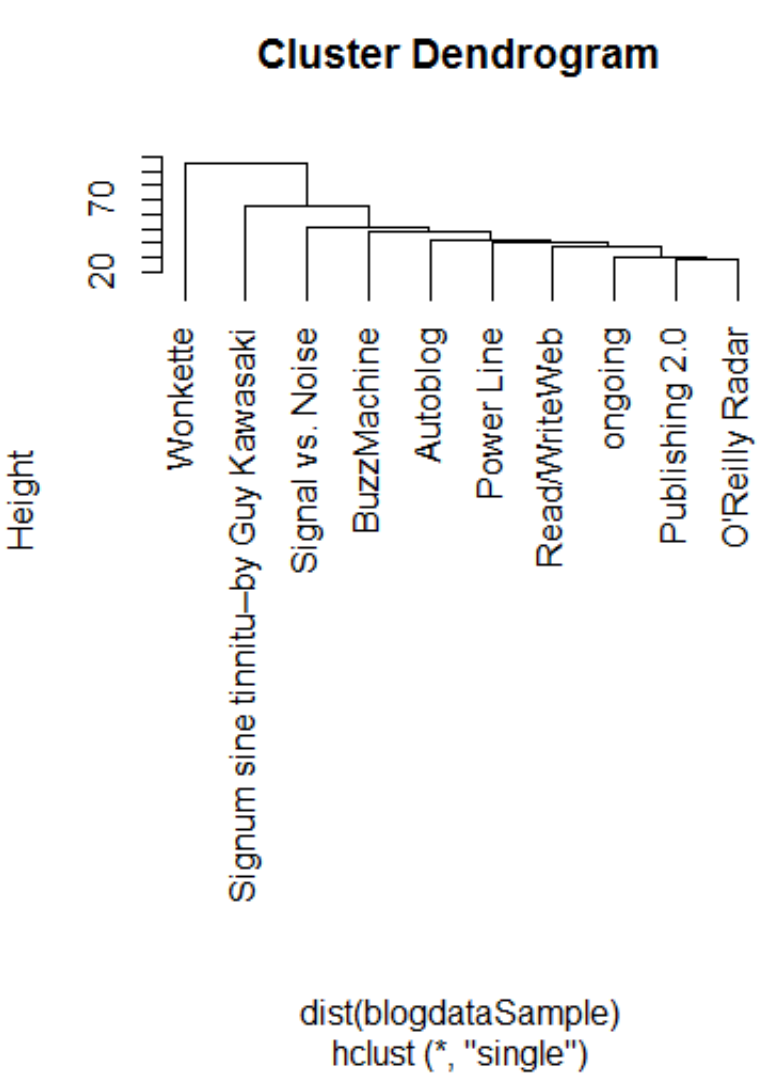
library(ctc); hc2Newick(hr)

"((g7:0.174726218492745,g9:0.174726218492745):0,((g2:0.26766991502818,(g4:0.05230
33203513996,g8:0.0523033203513996):0.26766991502818):0,((g10:0.044412043930424
7,(g1:0.0680271558561211,g6:0.0680271558561211):0.044412043930424 7):0,(g3:0.2433
51630808965,g5:0.243351630808965):0):0):0);"

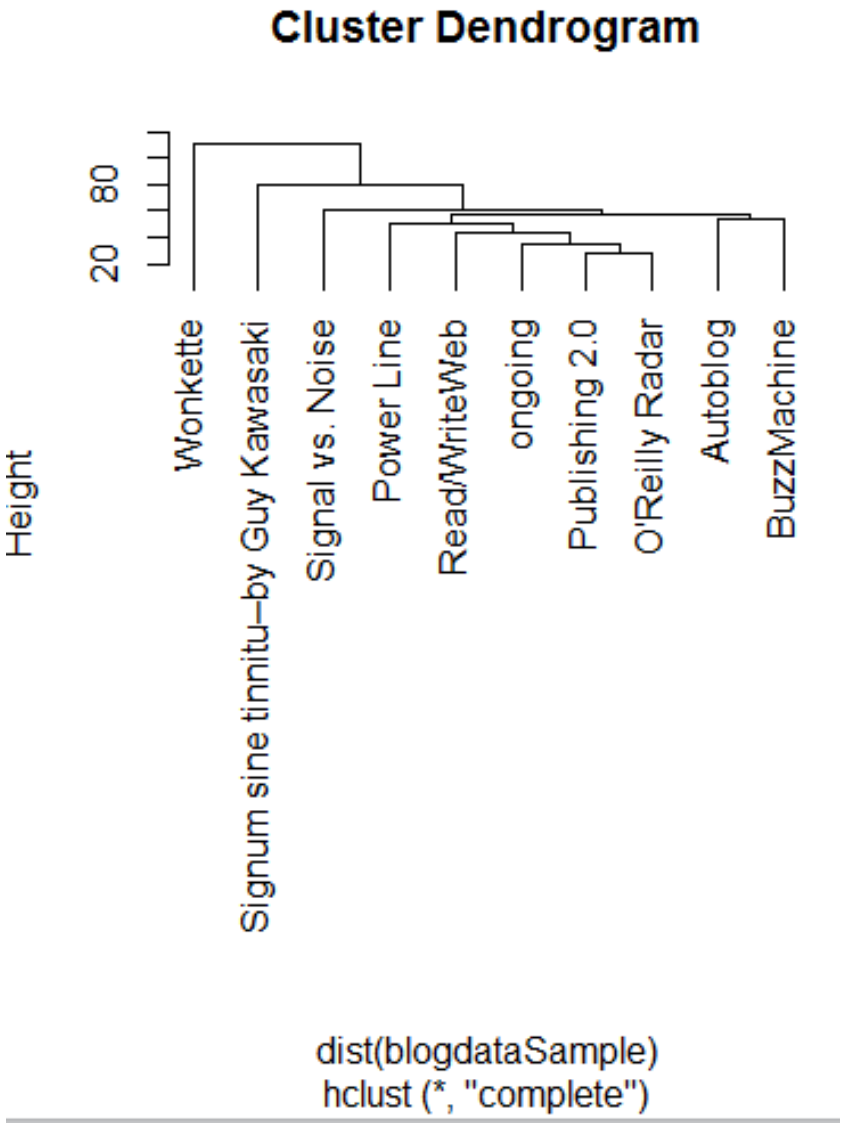
Анализ частоты встречаемости слов в блогах

```
blogdata<-read.csv("blogdata.csv")
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



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



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

Еще полезные команды

- ✓ cutree(tree, k = NULL, h = NULL)

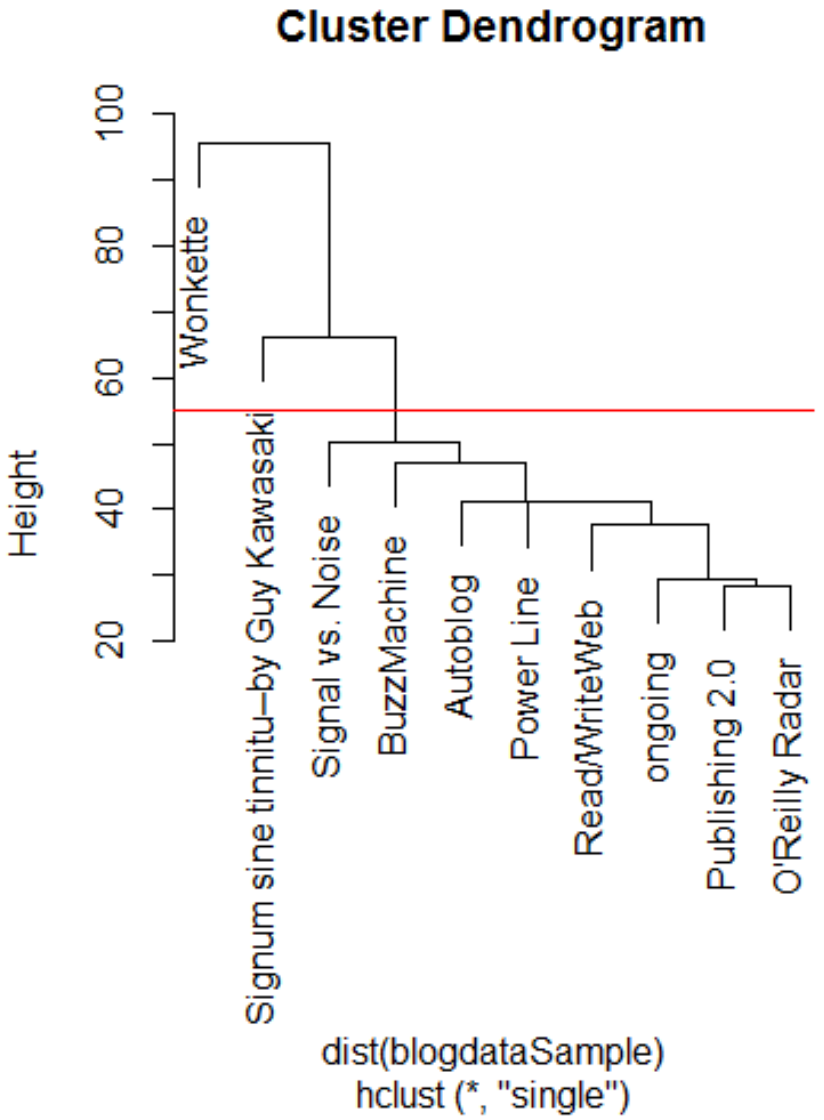
Обрезает дендрограмму так, чтобы получилось k кластеров или по определенной высоте h

- ✓ 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")
```

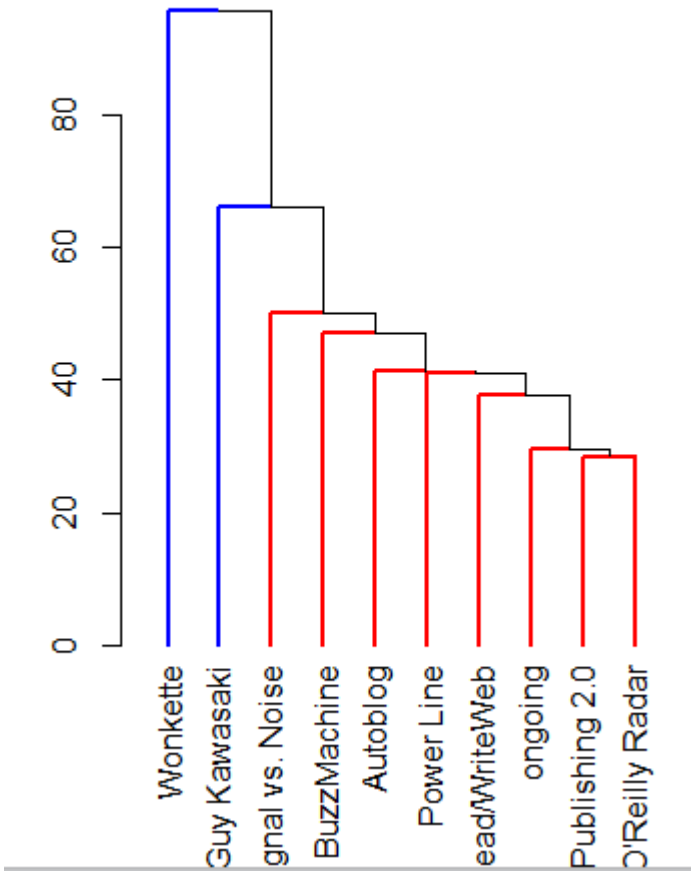


Дополнительные функции:

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)

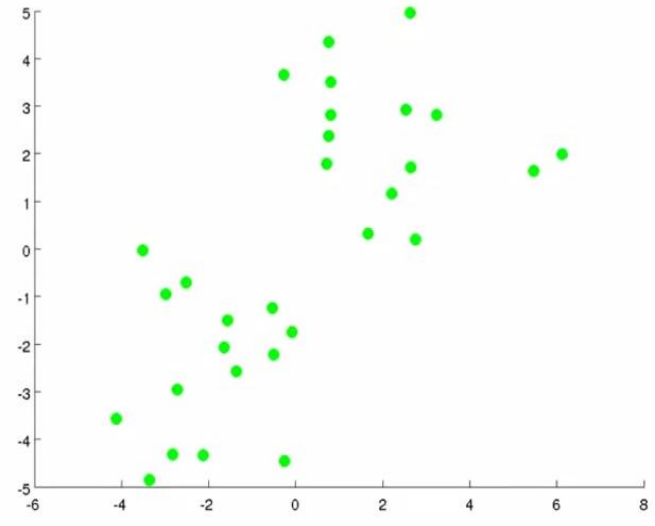
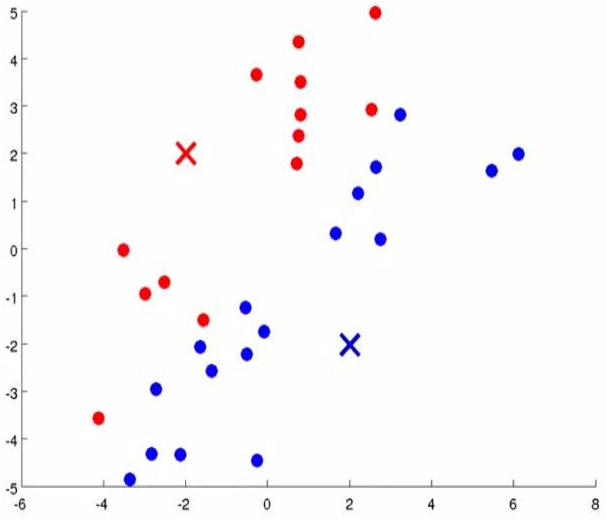
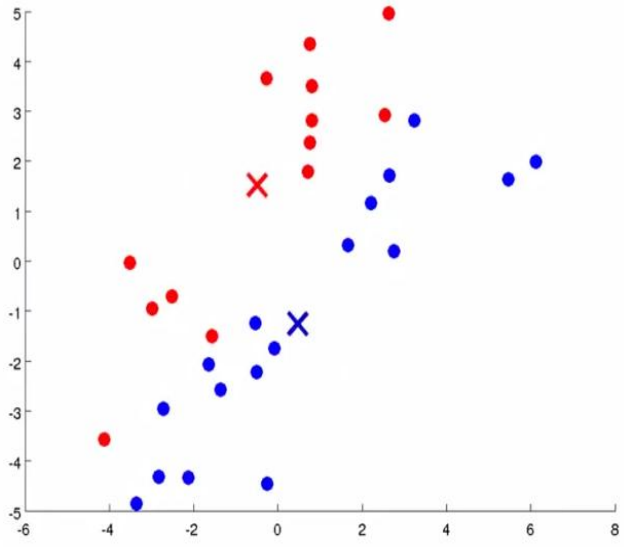
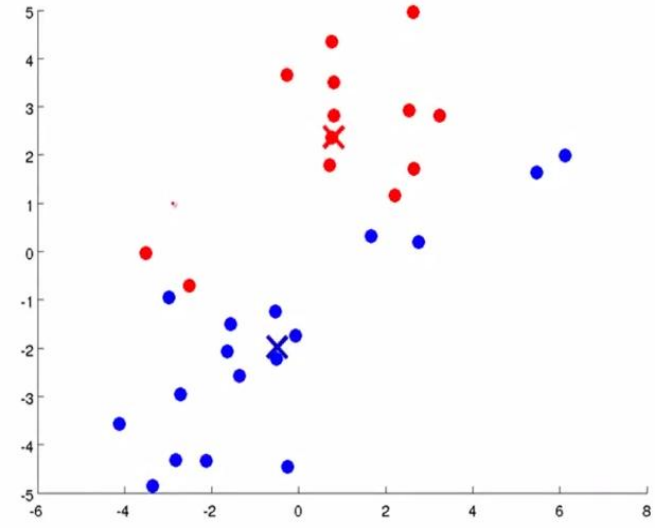
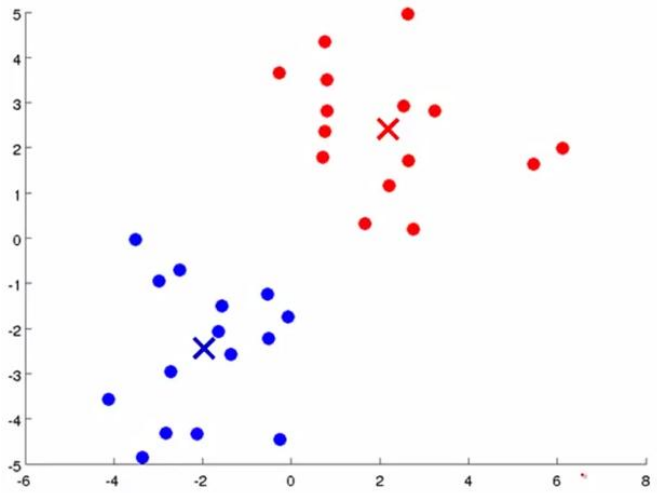
Неиерархическая кластеризация

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

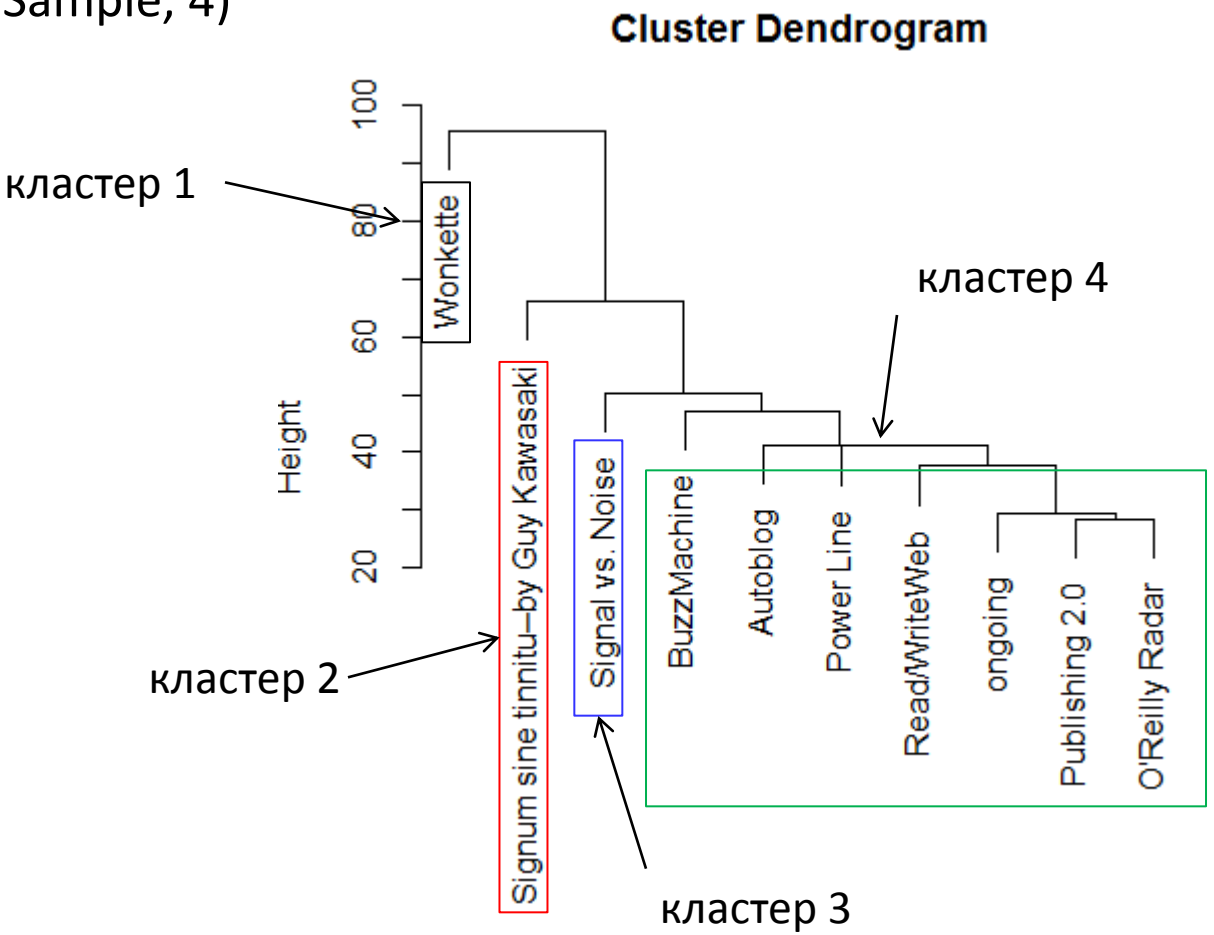
K-means

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.

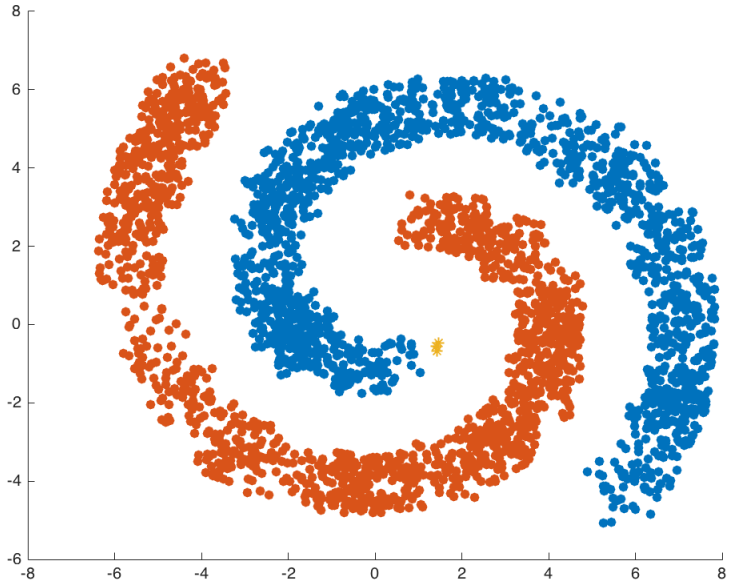
km <- kmeans(blogdataSample, 4)




```
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
```

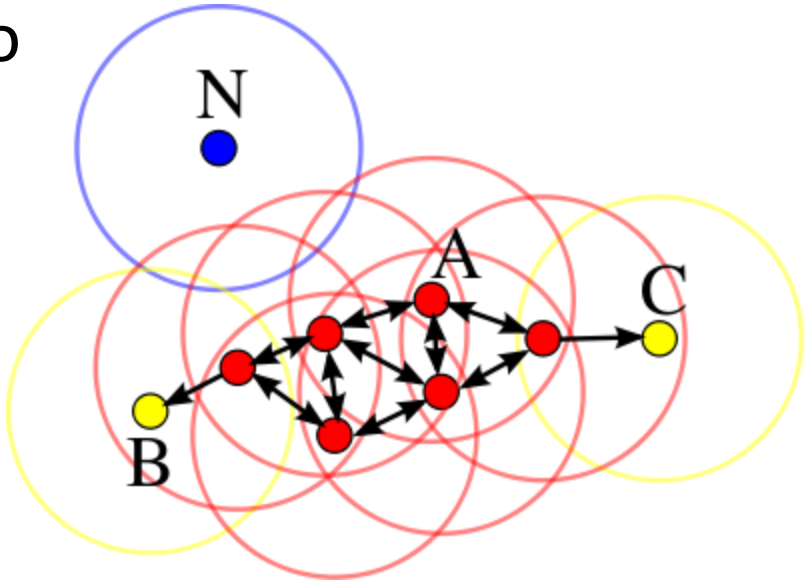
DBSCAN

- Позволяет кластеризовать сгустки точек произвольной формы
- Не требует задавать число кластеров
- Устойчив к шуму
- Плохо работает с кластерами разной плотности



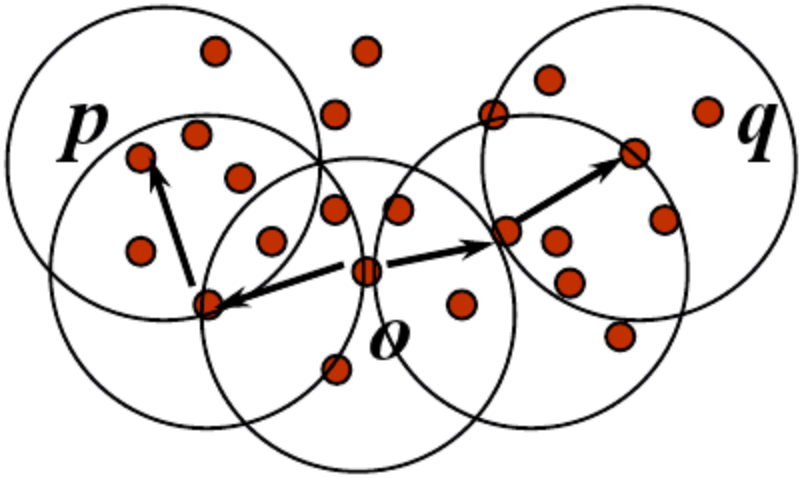
DBSCAN

- Красные – core, минимум **n** соседей (обычно от 3 до 9)
- Желтые – border, имеют среди соседей core
- Синие – noise, не имеют соседей
- Еще один параметр – **eps**, радиус для определения соседства



DBSCAN

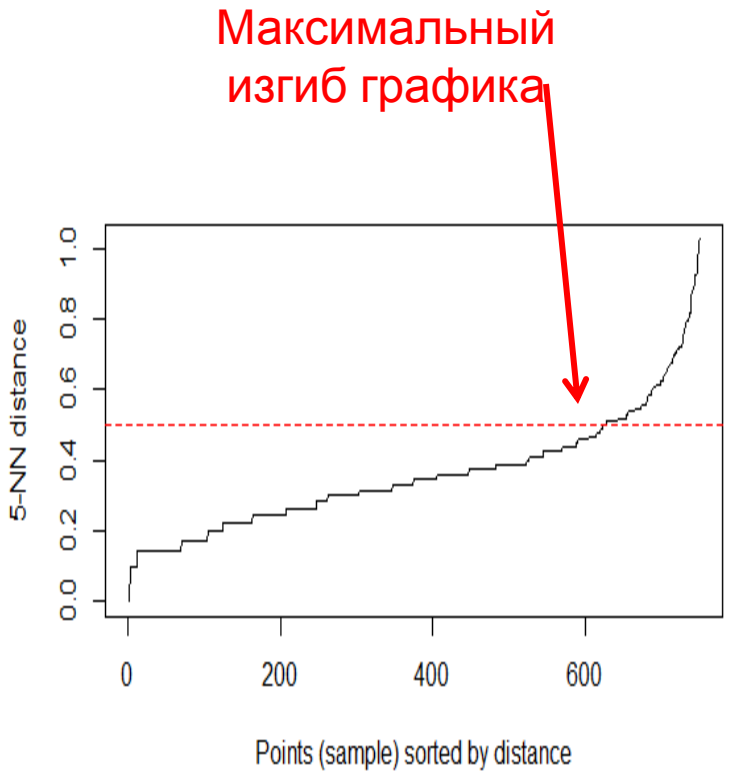
- Алгоритм перебирает все точки и выделяет связные компоненты



DBSCAN

Подбираем параметр eps

```
> library(dbscan)
> data(iris)
> iris <- as.matrix(iris[,1:4])
> kNNdistplot(iris, k = 5)
> abline(h=.5, col = "red", lty=2)
```



DBSCAN

```
> res <- dbscan(iris, eps = .5, minPts =
      5)
> res
DBSCAN clustering for 150 objects.
Parameters: eps = 0.5, minPts = 5
The clustering contains 2 cluster(s) and
      17 noise points.
0          1          2
17         49         84
>pairs(iris, col = res$cluster + 1L)
```