



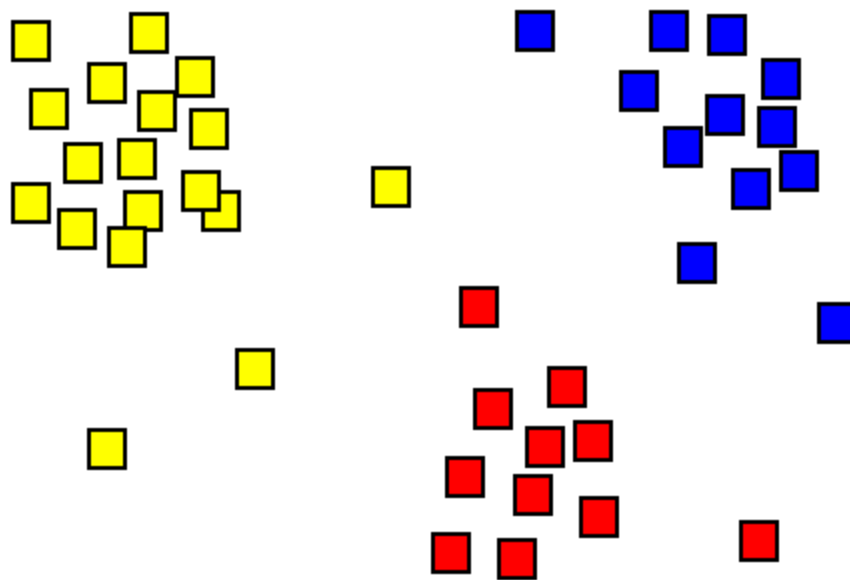
Кластеризация

9 октября 2014

Виноградова Светлана

Кластеризация— 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.



Меры расстояний

- ✓ Евклидово расстояние (геометрическое расстояние в многомерном пространстве):
- ✓ Квадрат евклидова расстояния
- ✓ Расстояние городских кварталов (манхэттенское расстояние)
- ✓ Матрицы корреляций

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

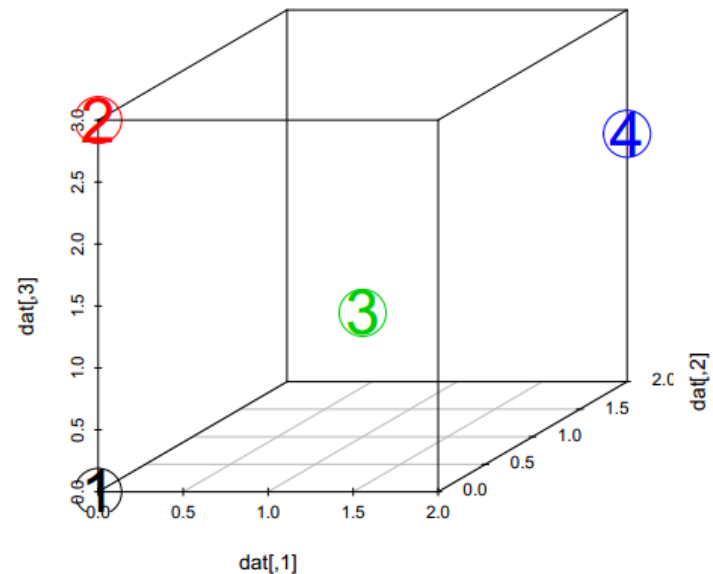
```
dist(x, method="euclidean", diag=FALSE, upper=FALSE)
```

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 3.000000 1.732051 3.464102
samp2 3.000000 0.000000 2.449490 3.000000
samp3 1.732051 2.449490 0.000000 1.732051
samp4 3.464102 3.000000 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.

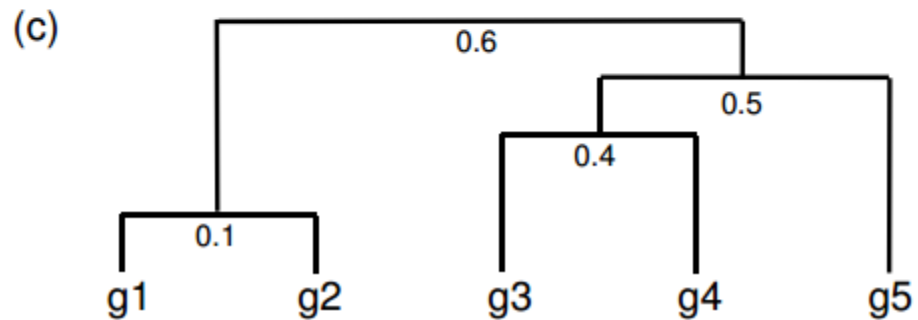
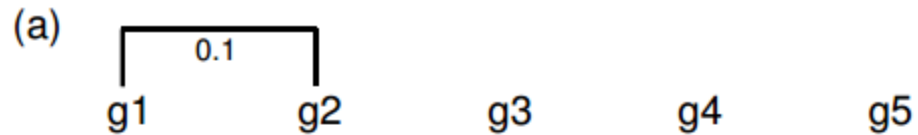
Алгоритмы иерархической кластеризации

- ✓ 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**

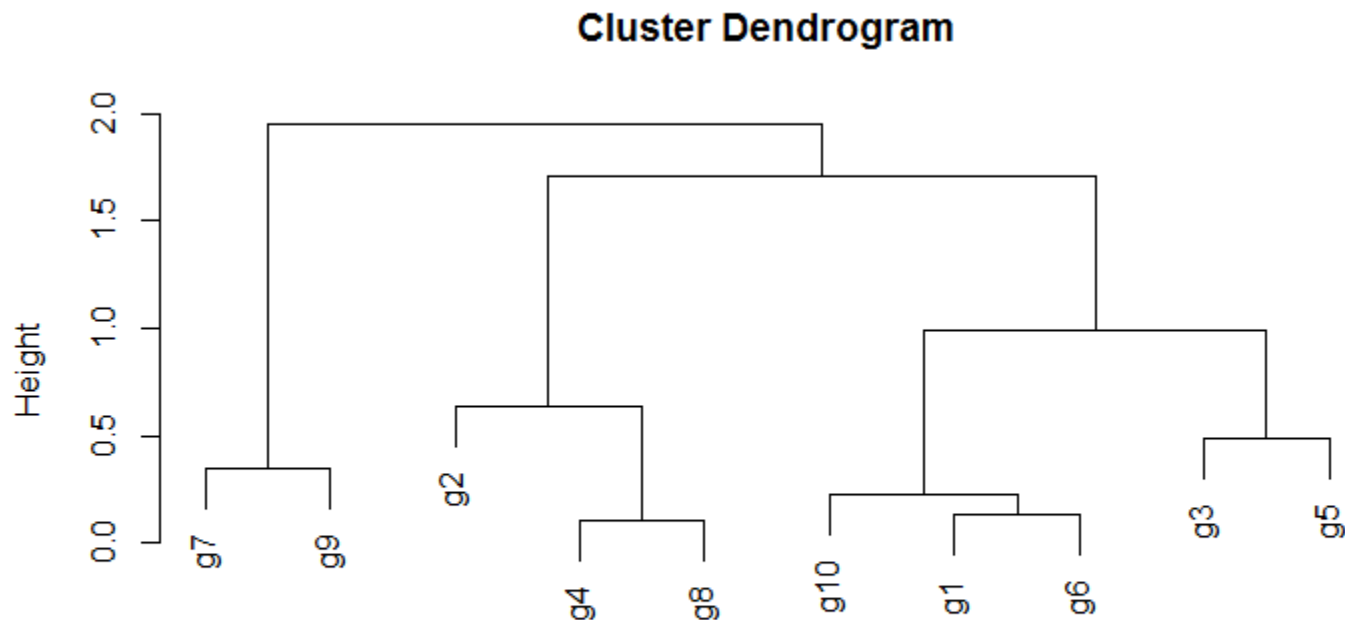
Результаты таких алгоритмов обычно представляют в виде дерева – дендрограммы. Классический пример такого дерева – классификация животных и растений.

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

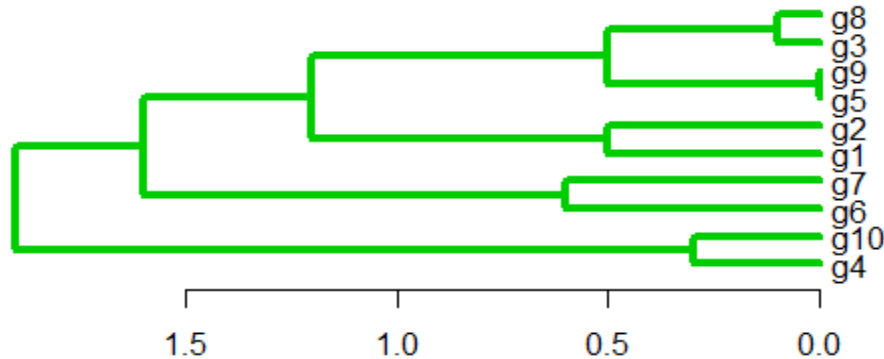



```
y <- matrix(rnorm(50), 10, 5, dimnames=list(paste("g", 1:10, sep=""), paste("t", 1:5, sep=""))) # генерируем данные
c <- cor(t(y), method="spearman"); d <- as.dist(1-c)
hr <- hclust(d, method = "complete", members=NULL)
plot(hr)
```



```
as.dist(1 - cor(t(y), method = "pearson"))
hclust (*, "complete")
```

```
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)
```

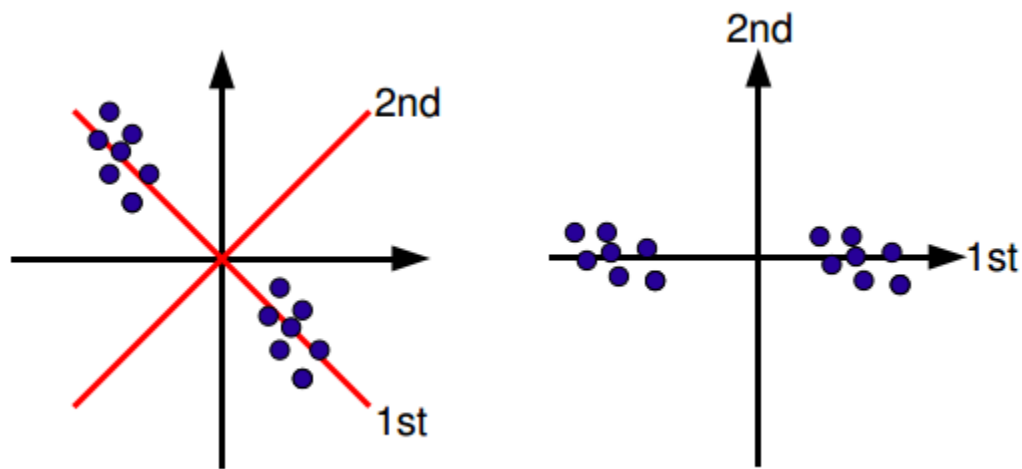
```
"((g7:0.174726218492745,g9:0.174726218492745):0,((g2:0.26766991502818,  
(g4:0.0523033203513996,g8:0.0523033203513996):0.26766991502818):0,  
((g10:0.0444120439304247,(g1:0.0680271558561211,g6:0.0680271558561211):  
0.0444120439304247):0,(g3:0.243351630808965,g5:0.243351630808965):0):0);"
```

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.



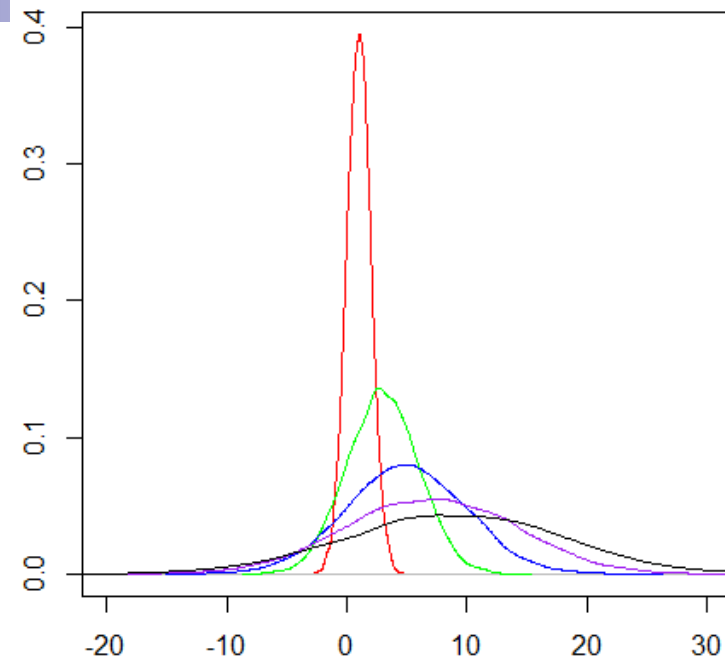
Principal Component	1 st	2 nd	3 rd	Other
Proportion of Variance	62%	34%	3%	rest

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=""))))
```

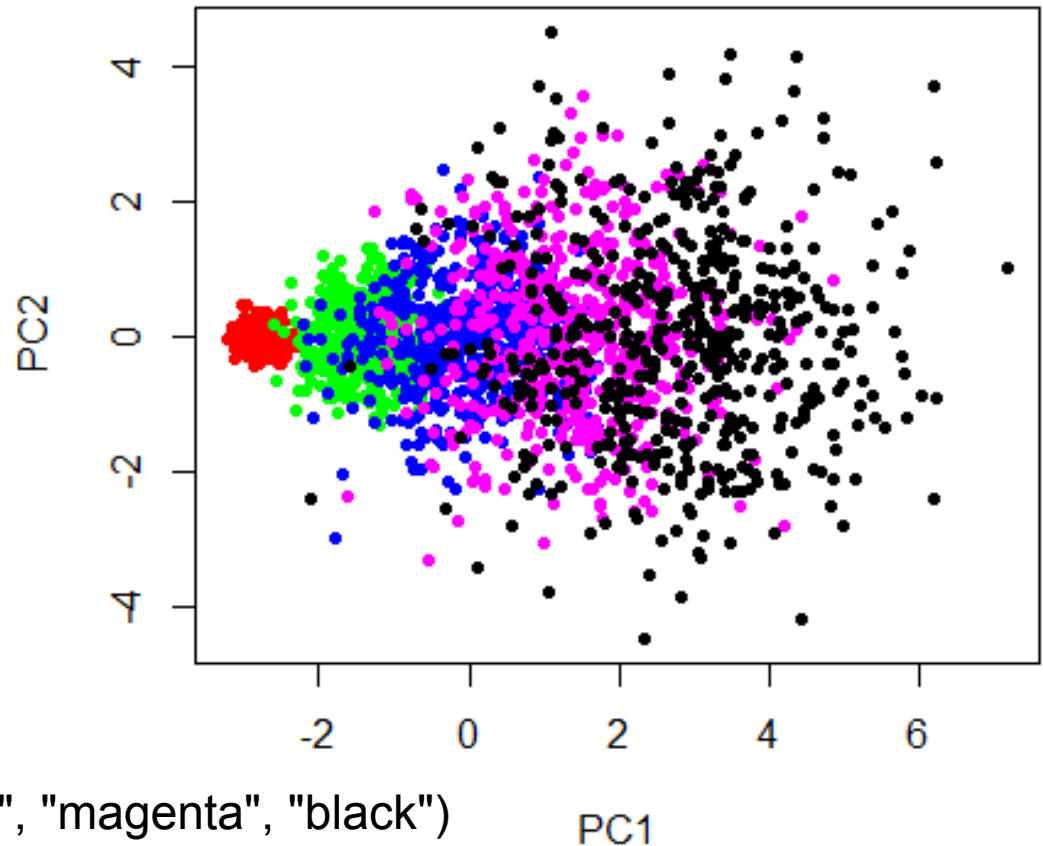
генерируем матрицу, 2500 строк и 20 столбцов

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

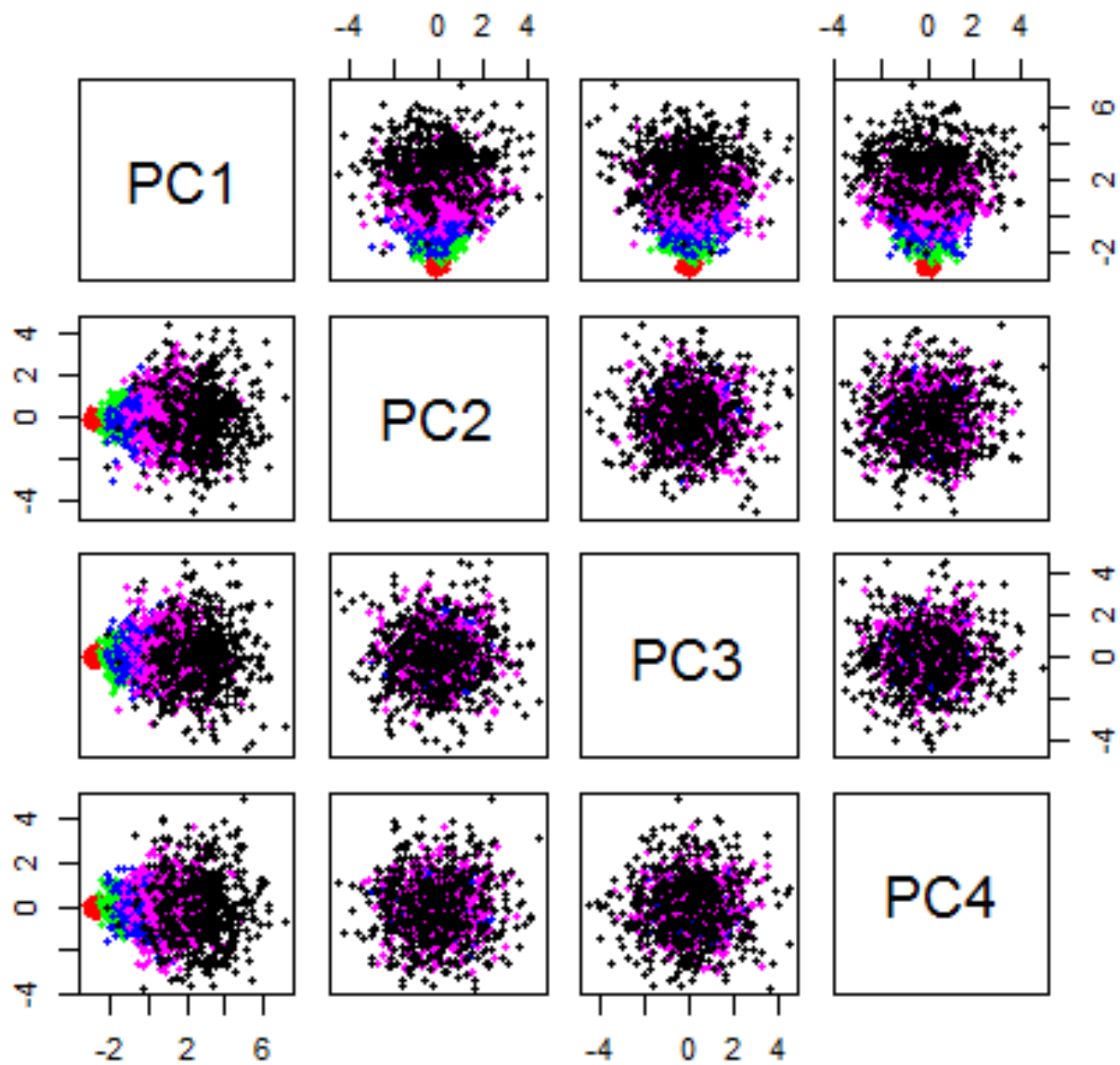


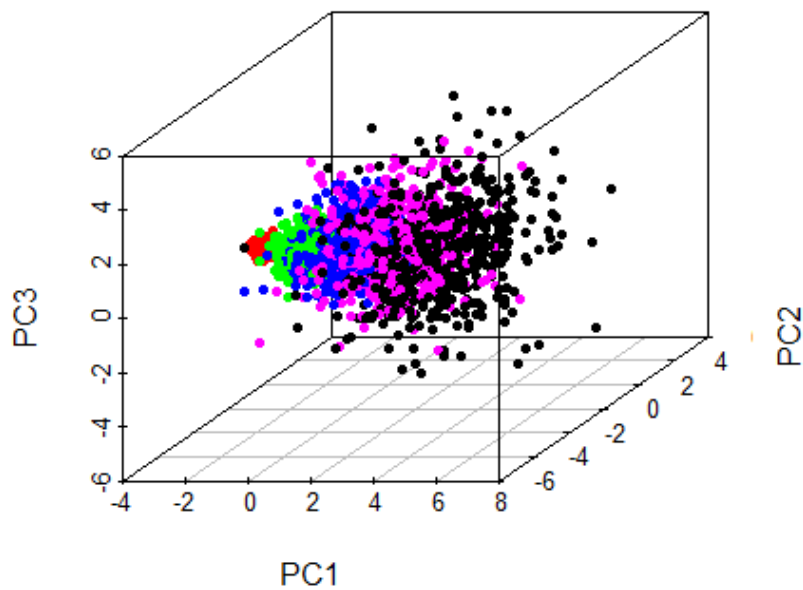
```
summary(pca) # Выводит на экран summary для всех компонент
summary(pca)$importance[, 1:6] # для первых шести компонент
```

	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))])
```





```
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

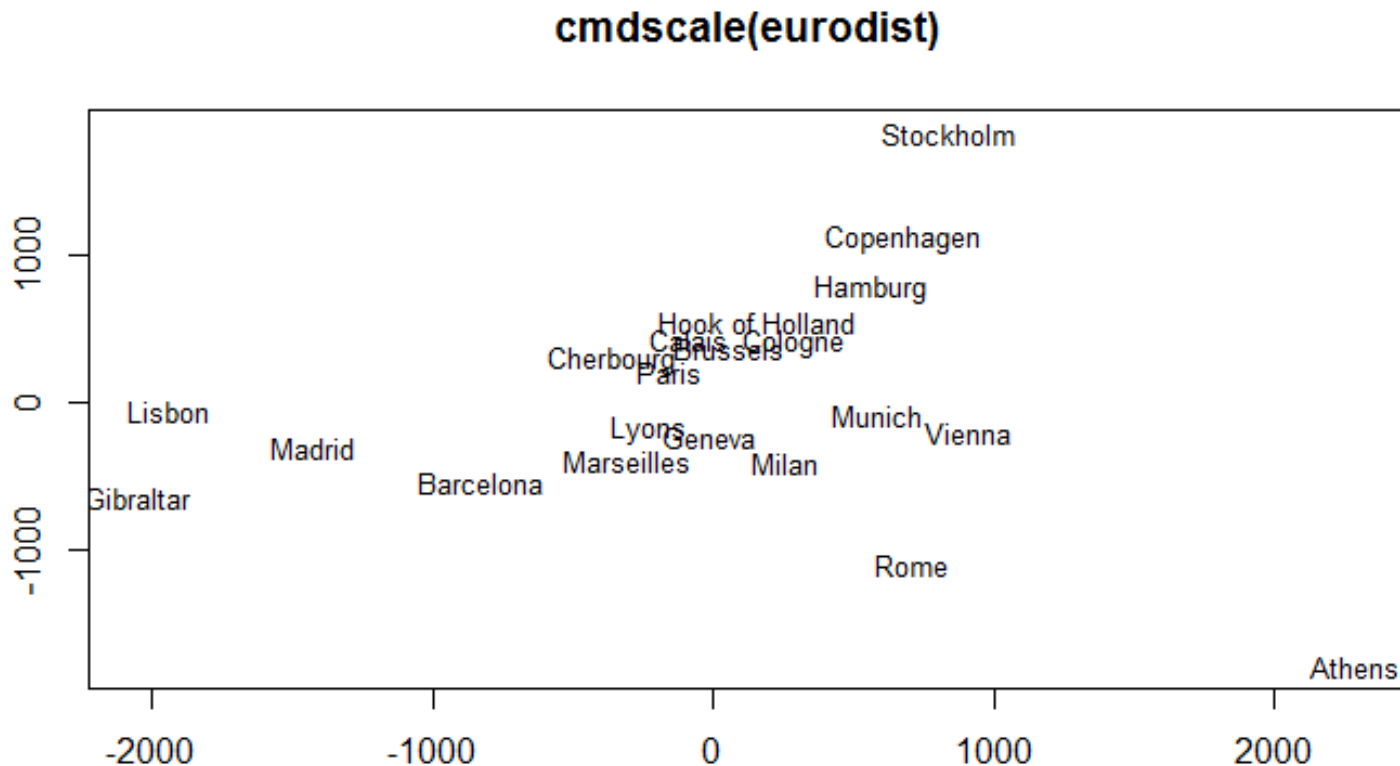
```
loc <- cmdscale(eurodist)
```

```
# MDS для географических расстояний между европейскими городами
```

```
plot(loc[,1], -loc[,2], type="n", xlab="", ylab="", main="cmdscale(eurodist)")
```

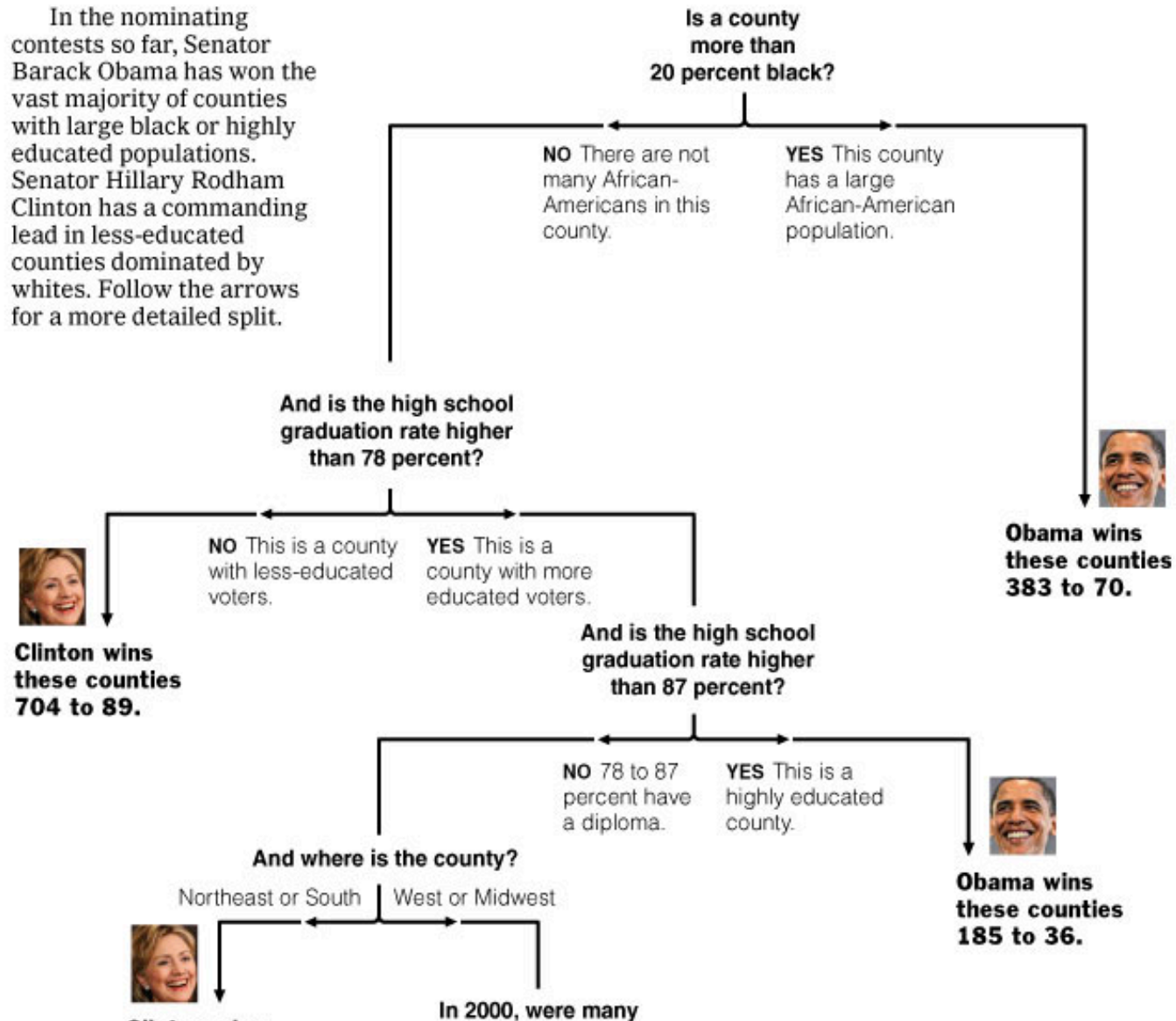
```
text(loc[,1], -loc[,2], rownames(loc), cex=0.8)
```

```
# строим график, минус нужен для того, чтобы перевернуть график в нужную ориентацию
```



Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.





Базовый алгоритм

1. Находим параметр, по которому группа разделяется лучше всего
2. Делим данные на 2 группы (листья), которые разделяются по узлу
3. Внутри каждой группы снова находим параметр, разделяющий эту группу лучше всего
4. Продолжаем, пока листья не окажутся достаточно маленькими или «чистыми»

Measures of impurity

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \text{ in Leaf } m} \mathbb{1}(y_i = k)$$

Misclassification Error:

$$1 - \hat{p}_{mk(m)}$$

Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \times \hat{p}_{mk'} = \sum_{k=1}^K \hat{p}_{mk} (1 - \hat{p}_{mk})$$

Cross-entropy or deviance:

$$- \sum_{k=1}^K \hat{p}_{mk} \ln \hat{p}_{mk}$$

Пример

```
> data(iris)
```

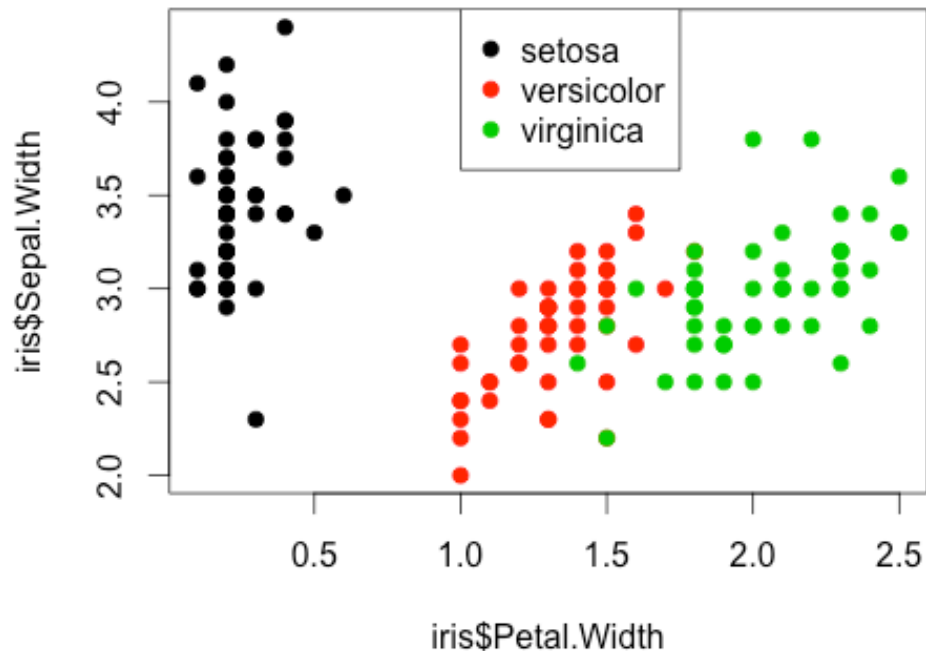
```
> names(iris)
```


```
[1] "Sepal.Length" "Sepal.Width" "Petal.Length"  
"Petal.Width" "Species"
```

```
> table(iris$Species)
```

```
setosa versicolor virginica  
  50      50      50
```

```
> plot(iris$Petal.Width,iris$Sepal.Width,pch=19,col=as.numeric(iris
$Species))
> legend(1,4.5,legend=unique(iris
$Species),col=unique(as.numeric(iris$Species)),pch=19)
```





```
> install.packages("tree")
> library(tree)
> tree1 <- tree(Species ~ Sepal.Width + Petal.Width,data=iris)
> summary(tree1)
```

Classification tree:

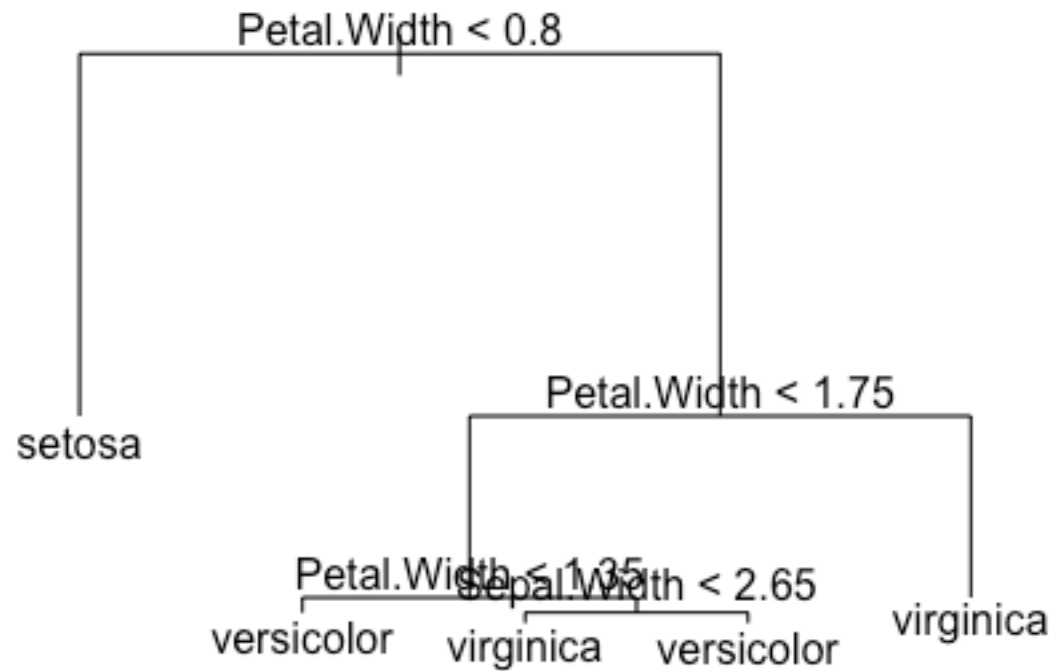
```
tree(formula = Species ~ Sepal.Width + Petal.Width, data = iris)
```

Number of terminal nodes: 5

Residual mean deviance: 0.204 = 29.57 / 145

Misclassification error rate: 0.03333 = 5 / 150

Построим дерево и подпишем листья



Еще один способ посмотреть на классификацию

```
> plot(iris$Petal.Width,iris$Sepal.Width,pch=19,col=as.numeric(iris$Species))  
> partition.tree(tree1,label="Species",add=TRUE)  
> legend(1.75,4.5,legend=unique(iris$Species),col=unique(as.numeric(iris  
$Species)),pch=19)
```

