

# Introduction to Machine Learning

## NPFL 054

<http://ufal.mff.cuni.cz/course/npfl054>

Barbora Hladká  
hladka@ufal.mff.cuni.cz

Martin Holub  
holub@ufal.mff.cuni.cz

Charles University,  
Faculty of Mathematics and Physics,  
Institute of Formal and Applied Linguistics

## Outline

- **Basic data analysis**
  - data for the Movie recommendation task
  - data for the Verb Pattern Recognition task
- **Clustering**
  - USArrest data set

# Movie recommendation task (MOV)

Predict the user's rating for a given movie

	Toy Story (1995)	Star Wars (1977)	Some Like It Hot (1959)
Peter	?	5	4
Paul	2	5	?
Mary	2	4	?

E.g., predict Mary's rating for the *Some Like it Hot* movie

# MOV – Available data

- About users

	age	gender	occupation	zip code
Peter	19	M	student	58644
Mary	50	F	healthcare	60657

- About movies

title	action	...	IMDb rating	director
Toy Story	0	...	8.3	John Lasseter
Some Like It Hot	0	...	8.3	Billy Wilder
Star Wars	1	...	8.7	George Lucas

# MOV – Getting examples

- Create a database of movies to be rated by users
- Set up a rating scale allowing users to rate movies
- Record users' ratings
- Typically, the dataset of ratings is sparse.  
So do some pruning, like require a minimum of twenty ratings per user

## Basic statistics

<b>number of ratings</b>	100,000
<b>number of movies</b>	1,682
<b>number of users</b>	943

- Data comes from the MovieLens datasets
  - for more details, go to the course web page

# MOV – Data representation

	1	2	3	4	5-8	9-33
vote id	MOVIE	USER	RATING	TIMESTAMP	user features	movie features
1	1	1	5	1997-09-23 00:02:38	24 M technician 85711	Toy Story (1995) ...
...	...	...	...	...	...	...
100,000	1682	916	3	...	...	...

See the feature description mov.pdf at  
<https://ufal.mff.cuni.cz/course/npfl054/materials>

# MOV – Loading the data into R

```
# http://ufal.mff.cuni.cz/course/NPFL054/materials
> source("load-mov-data.R")
> names(examples)
[1] "movie"          "user"           "rating"
[4] "timestamp"      "age"            "gender"
...
[31] "directors"      "writers"        "stars"

> names(movies)
[1] "movie"          "title"          "release_date"
[4] "genre_action"   "genre_adventure" "genre_animation"
...
[22] "imdb_url"       "imdb_rating"    "directors"
[25] "writers"        "stars"

> names(users)
[1] "user"          "age"            "gender"         "occupation" "zip"
> names(votes)
[1] "user"          "movie"          "rating"         "timestamp"
```



## Machine learning process

- ① Formulating the task (e.g. predict user's rating for a given movie)
- ② Getting data (e.g. MOV data)
  - **Data analysis**
- ③ Building predictor
- ④ Evaluation

**Deeper understanding the task by statistical view on the data**  
**We exploit the data in order to make prediction of the target value.**

- Build intuition and understanding for both the task and the data
- Ask questions and search for answers in the data
  - **What values do we see?**
  - **What associations do we see?**
- Do plotting and summarizing

## We focus on

- Recap of methods for basic data exploration
- Analyzing distributions of values
- Analyzing association between features
- Analyzing association between features and target value

# Methods for basic data exploration

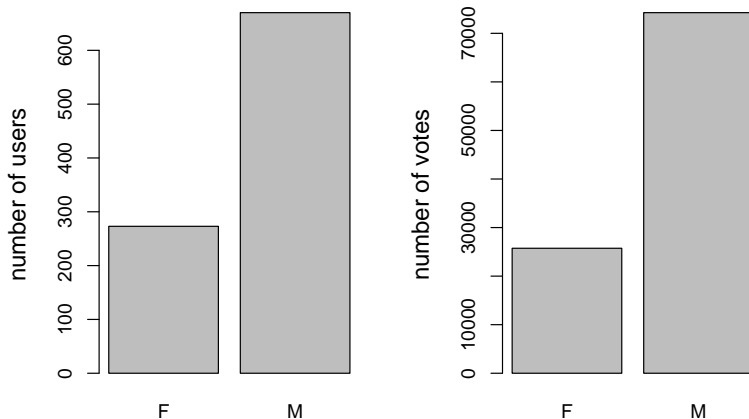
**Frequency tables** display the frequency of categorical feature values.

```
# frequency of voting men and women  
> table(examples$gender)  
  F    M  
25740 74260
```

# Methods for basic data exploration

**Bar plots** visualize frequency tables

**Barplot (barplot-gender.R)**



# Methods for basic data exploration

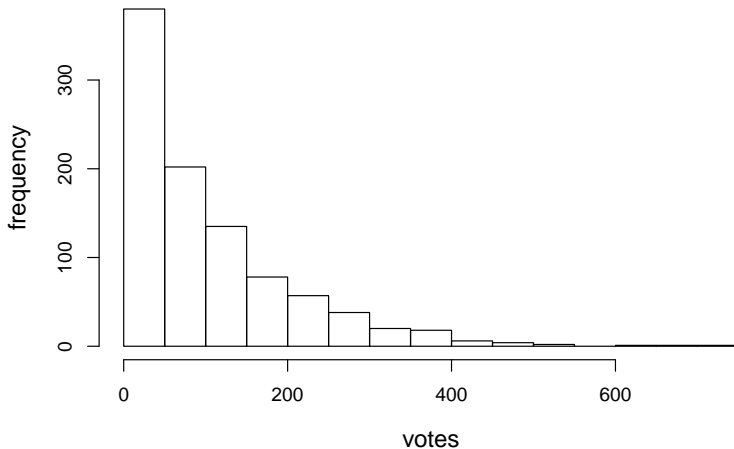
**Histograms** visualize distribution of feature values.

Add a new feature `VOTES` for the number of votes of the users

```
# get the number of votes for each user
> v <- as.data.frame(table(examples$user))
> users$votes <- v$Freq
> min(users$votes)
[1] 20
> max(users$votes)
[1] 737
```

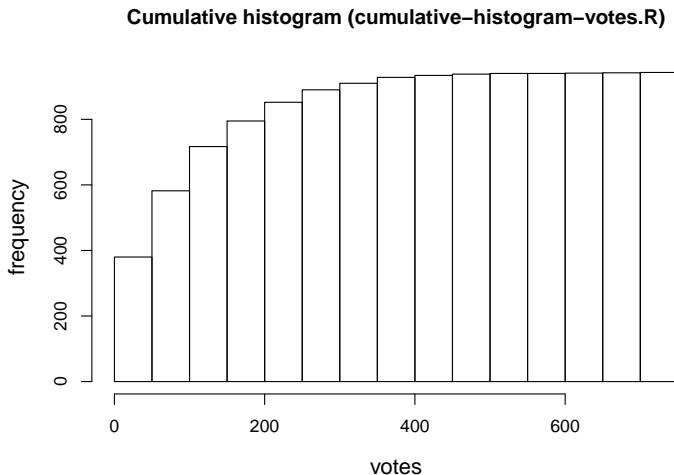
# Methods for basic data exploration

Histogram (histogram-votes.R)



# Methods for basic data exploration

**Cumulative histograms** visualize cumulative frequencies.





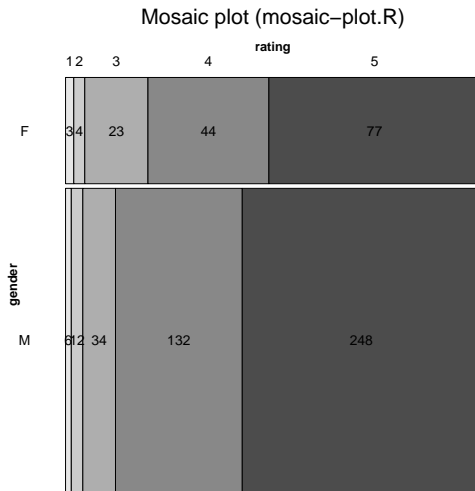
# Methods for basic data exploration

**Contingency tables** display the frequency of values for combination of two categorical features.

```
> # Star Wars ratings
> movie <- subset(examples, movie == 50); nrow(movie) # 583
> # construct contingency table for gender and rating
> ct <- table(movie$gender, movie$rating)
> margin.table(ct)          # total sum
[1] 583
> addmargins(ct)            # adds marginal sums by default
      1    2    3    4    5 Sum
F      3    4   23   44   77 151
M      6   12   34  132  248 432
Sum     9   16   57  176  325 583
> round(prop.table(ct),3)    # prop.table generates proportions
      1      2      3      4      5
F 0.005 0.007 0.039 0.075 0.132
M 0.010 0.021 0.058 0.226 0.425
```

# Methods for basic data exploration

**Mosaic plots** visualize contingency tables.



# Methods of basic data exploration

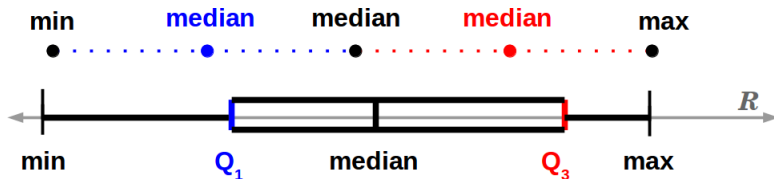
## Measures of center and variation

```
> min(users$votes);max(users$votes)
[1] 20
[1] 737
> mean(users$votes)
[1] 106.4
> median(users$votes)
[1] 65
> summary(users$votes) # five-number summary
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   20     33     65    106    148    737

> sd(users$votes) # standard deviation
[1] 100.93
```

# Methods of basic data exploration

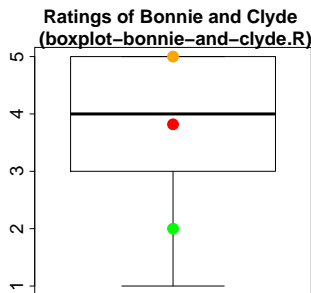
**Box-and-whiskers plots** visualize five-number summaries.



- $Q_3 - Q_1 = \text{interquartile range}$

# Methods of basic data exploration

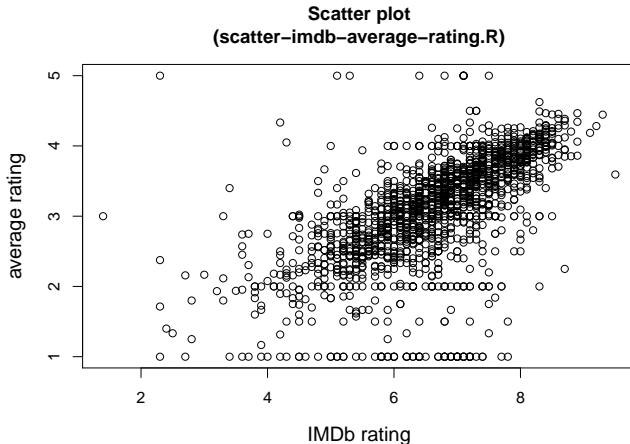
## Box-and-whiskers plots



- the sample of 122 ratings  
1 2 3 4 5  
2 7 35 45 33
- $\text{median}(\mathbf{x}) = 4$
- $\bar{x} = 3.82$
- $sd(\mathbf{x}) = 0.95$
- the bottom whisker is much longer than the top whisker
- the sample is skewed to the left
- Peter's rating is in green and Mary's rating in orange

# Methods of basic data exploration

**Scatter plots** display values of two numerical features.



# What values do we see

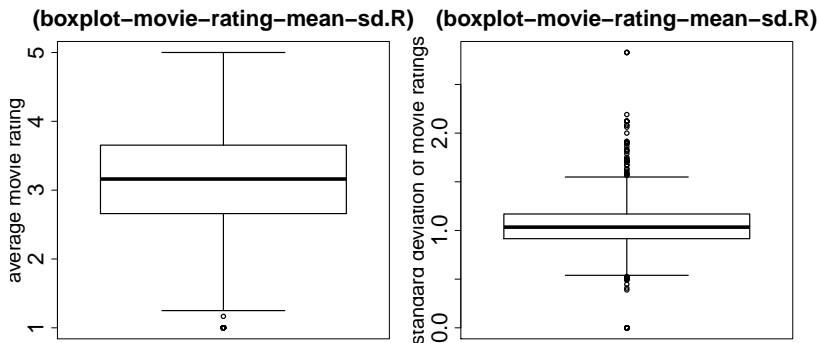
## Analyzing distributions of values

**Box-and-whiskers plots** as indicators of

- centrality
- spread
- symmetry
- tail length

# Analyzing distributions of values

- **Boxplots** are of a great importance to detect outliers and extreme values
- **Outlier** (**Extreme value**) is an observation that is distant from other observations, typically if it falls more than  $1.5 (3) * (Q_3 - Q_1)$  above  $Q_3$  or below  $Q_1$ .





# Analyzing distributions of values

**Boxplots are of a great importance** to detect outliers and extreme values

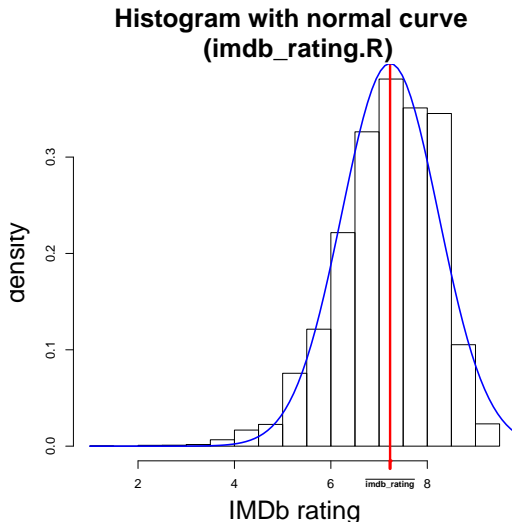
```
> boxplot <- boxplot(tapply(votes$rating, votes$movie, sd))
# analyze outliers
> boxplot$out[1:2]
      247      314
1.788854 0.000000
>
> subset(votes, movie == 247) # Turbo: A Power Rangers Movie
      user movie rating      timestamp
38147   38   247      5 1998-04-13 03:04:20
38148    1   247      1 1997-09-26 04:40:19
38149  374   247      1 1997-12-01 01:35:22
38150  222   247      1 1997-11-05 08:29:58
38151  782   247      1 1998-04-02 08:48:20

> movies[movies$movie == 247,]
247 Turbo: A Power Rangers Movie (1997) 28-Mar-1997 ...
```

# Analyzing distributions of values

## Analyzing imdb\_rating

- What kind of probability distribution characterizes the IMDb ratings?



# Analyzing distributions of values

## Analyzing `imdb_rating`

**Does `imdb_rating` follow a normal distribution?**

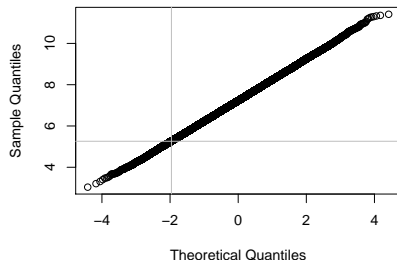
- Visualize the distribution using a quantile-quantile plot
- Use a distribution test

# Analyzing distributions of values

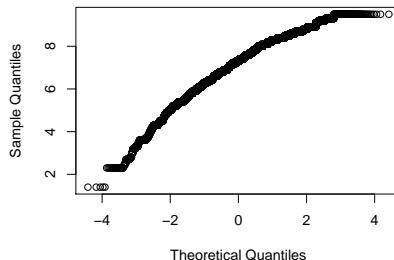
## Analyzing imdb\_rating

Visualize the distribution using a quantile-quantile plot

Normal Q-Q Plot



IMDb Q-Q plot(imdb\_rating.R)



- **Draw a conclusion:** `imdb_rating` does not follow a normal distribution.

# Association between feature and target value

## Numerical features

**Pearson correlation coefficient** is a measure of the linear relationship between features

- **For a population**

$$-1 \leq \rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \leq +1$$

- perfect negative correlation if  $\rho = -1$
- perfect positive correlation if  $\rho = +1$
- not linear relationship if  $\rho = 0$

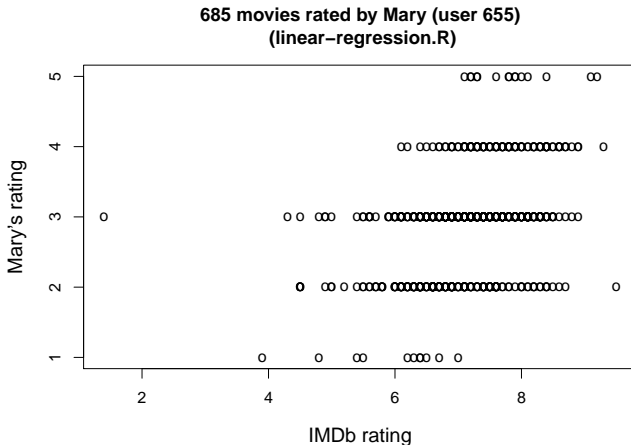
- **For a sample**

$$-1 \leq r_{X,Y} = \frac{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{s_X s_Y} \leq +1$$

# Association between feature and target value

## Numerical features

### Association between Mary's ratings and the IMDb ratings



# Association between feature and target value

## Numerical features

$r(\text{Peter's rating, imdb\_rating})$	0,51
$r(\text{Paul's rating, imdb\_rating})$	0,44
$r(\text{Mary's rating, imdb\_rating})$	0,37

# Association between features

## Numerical features

$r(\text{Peter's rating, Mary's rating})$	0,29
$r(\text{Peter's rating, Paul's rating})$	0,29
$r(\text{Paul's rating, Mary's rating})$	0,24



# Association between feature and target value

## Categorical features

### Pearson's $\chi^2$ test

This test compares observed frequencies  $O_{ij}$  with theoretical frequencies  $E_{ij}$  that we would expect in case of statistical independence of  $X$  and  $Y$ . Test statistic  $\chi^2 = \sum_{i=1}^r \sum_{j=1}^s \frac{(O_{ij}-E_{ij})^2}{E_{ij}}$  follows a  $\chi^2$  distribution with  $(r-1)(s-1)$  degrees of freedom when the null hypothesis is true ( $r/s$  is the number of rows/columns in the contingency table).

### Pearson contingency coefficient

$$0 < \sqrt{\frac{\chi^2}{n + \chi^2}} < 1$$

- perfect correlation if  $\rightarrow 1$
- no correlation if  $\rightarrow 0$

**Note:** Correction such that the coefficient can take values between 0 and 1 (which is not true if  $r \neq s$ ):  $\sqrt{\frac{\chi^2}{n + \chi^2}} / \sqrt{\frac{\min(r,s)-1}{\min(r,s)}}$

# Association between feature and target value

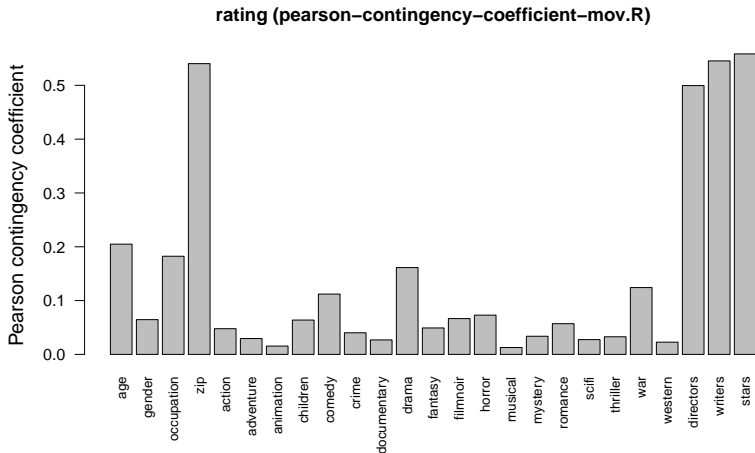
## Categorical features

### Association between gender and rating

		rating				
		1	2	3	4	5
gender	F	1894	2784	6784	8303	5975
	M	4216	8586	20361	25871	15226

# Association between feature and target value

## Categorical features



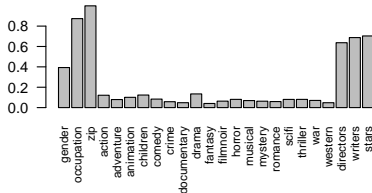
# Association between features

## Categorical features

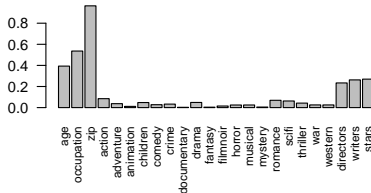
Pearson contingency coefficient

Pearson contingency coefficient

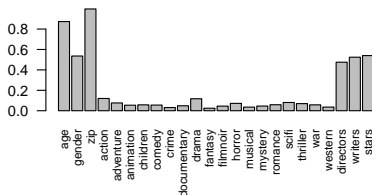
**age**  
(pearson–contingency–coefficient–mov.R)



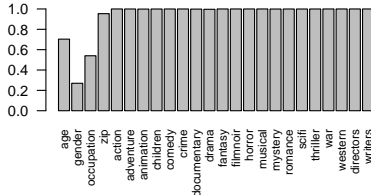
**gender**



**occupation**



**stars**



# Analyzing distributions of values

## Feature frequency

- **Feature frequency**

$$\text{fr}(A_j) = \#\{\mathbf{x}_i \mid x_i^j > 0\}$$

where  $A_j$  is the  $j$ -th feature,  $\mathbf{x}_i$  is the feature vector of the  $i$ -th instance, and  $x_i^j$  is the value of  $A_j$  in  $\mathbf{x}_i$ .

# Analyzing distributions of values

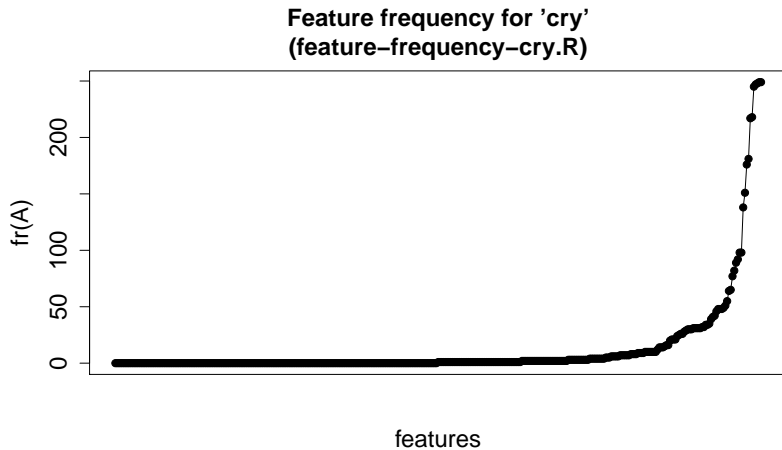
## Feature frequency – VPR data (cry)

```
> examples <- read.csv("cry.development.csv", sep="\t")
> c <- examples[, -c(1, ncol(examples))]
> nrow(examples)
[1] 250
> length(names(c)) # get the number of features
[1] 363
# compute feature frequency using the fr function (see feature-frequency-cry.R)
> ff <- apply(c, 2, fr) # apply fr to columns ('2') of c
> table(sort(ff))
```

0	1	2	3	4	5	6	7	8	9	10	12	14	15	16	20	
181	47	26	12	9	3	5	6	4	4	7	1	3	1	2	1	
21	24	25	26	28	29	30	31	32	34	35	39	41	42	46	48	49
3	1	1	2	1	1	3	5	2	2	1	1	1	1	1	3	1
51	55	64	65	77	82	89	92	98	138	151	176	181	217	218	245	
1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1
247	248	249														
1	1	2														

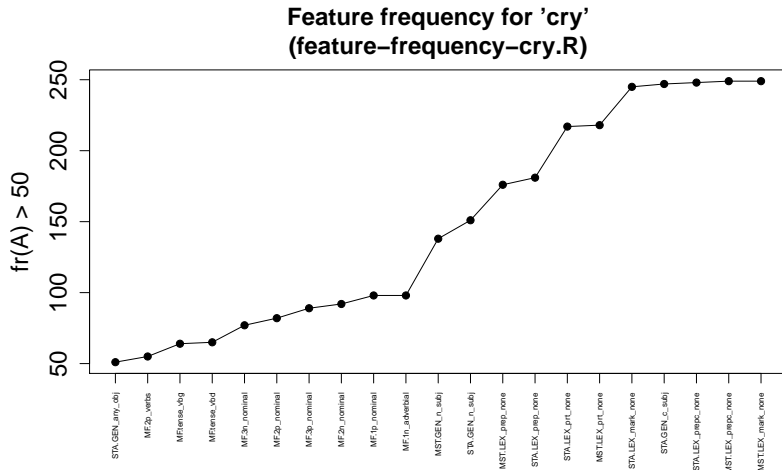
# Analyzing distributions of values

## Feature frequency – VPR data (cry)



# Analyzing distributions of values

## Feature frequency – VPR data (cry)





# Analyzing distributions of values

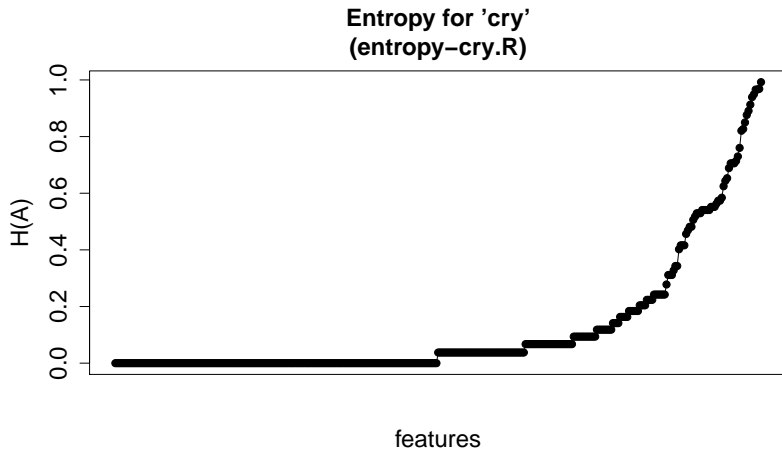
## Entropy – VPR data (cry)

```
# compute entropy using the entropy function (see entropy-cry.R)
> e <- apply(c, 2, entropy)
> table(sort(round(e,2)))
```

0	0.04	0.07	0.09	0.12	0.14	0.16	0.18	0.2	0.22	0.24	0.28	0.31	0.33
181	49	27	13	9	4	5	6	4	4	7	1	3	1
0.34	0.4	0.42	0.46	0.47	0.48	0.51	0.52	0.53	0.54	0.55	0.56	0.57	0.58
2	1	3	1	1	2	1	1	3	5	3	1	2	1
0.62	0.64	0.65	0.69	0.71	0.73	0.76	0.82	0.83	0.85	0.88	0.89	0.91	0.94
1	1	1	1	4	1	1	1	1	1	1	1	1	1
0.95	0.97	0.99											
1	3	1											

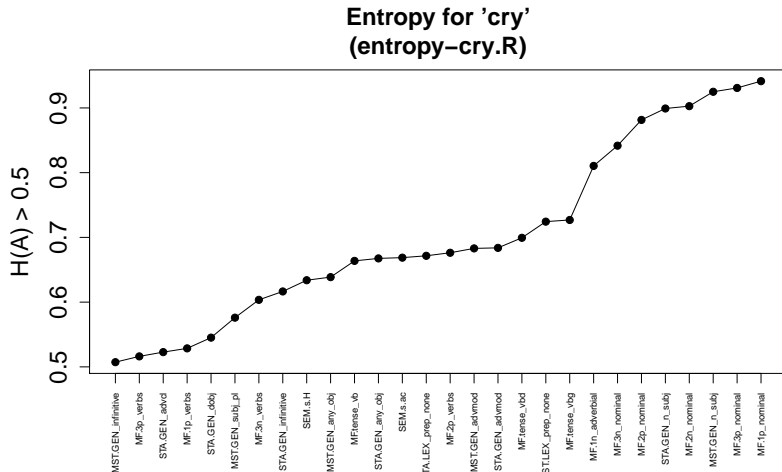
# Analyzing distributions of values

## Entropy – VPR data (cry)



# Analyzing distributions of values

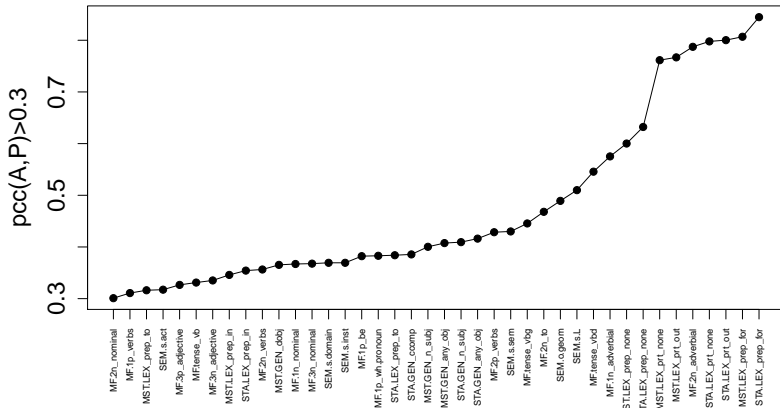
## Entropy – VPR data (cry)



# Association between feature and target value

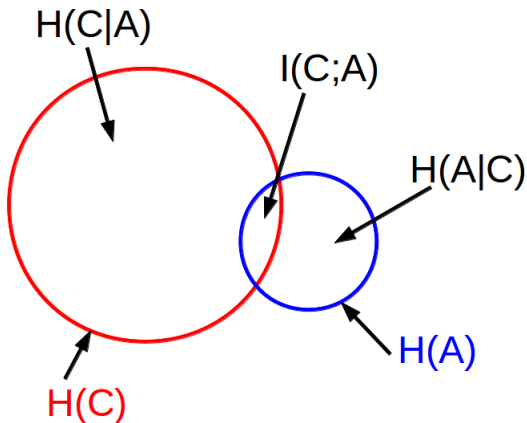
## Pearson contingency coefficient – VPR data (cry)

Pearson contingency coefficient for 'cry'  
(pearson-contingency-coefficient-vpr.R)



# Association between feature and target value

## Conditional entropy



# Association between feature and target value

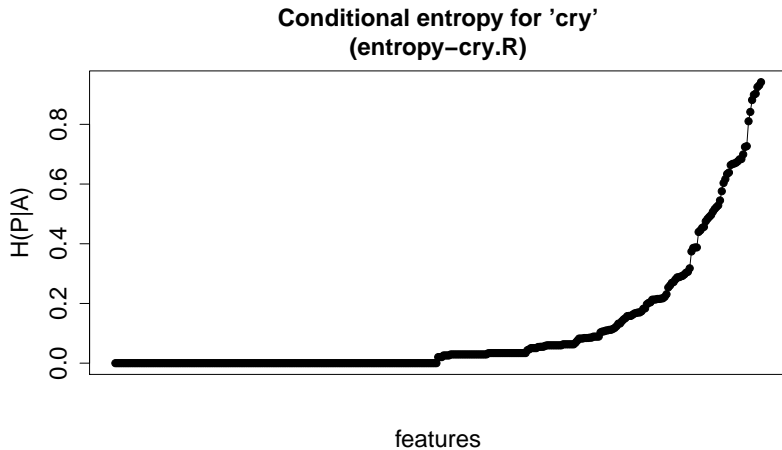
## Conditional entropy – VPR data (cry)

```
# compute conditional entropy using entropy.cond (see entropy-cry.R)
ce <- apply(c, 2, entropy.cond, y=examples$tp)
table(sort(round(ce,2))
```

0	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.11	0.12	0.13	0.14	0.15	0.16
181	3	47	1	7	19	1	7	6	1	7	2	2	1	2	4
0.17	0.18	0.2	0.21	0.22	0.23	0.25	0.26	0.27	0.28	0.29	0.3	0.31	0.32	0.37	0.39
5	2	3	3	5	1	1	1	2	1	4	2	1	1	1	3
0.44	0.45	0.46	0.48	0.49	0.5	0.51	0.52	0.53	0.55	0.58	0.6	0.62	0.63	0.64	0.66
1	2	1	2	1	1	1	2	1	1	1	1	1	1	1	1
0.67	0.68	0.7	0.72	0.73	0.81	0.84	0.88	0.9	0.92	0.93	0.94				
3	3	1	1	1	1	1	1	2	1	1	1				

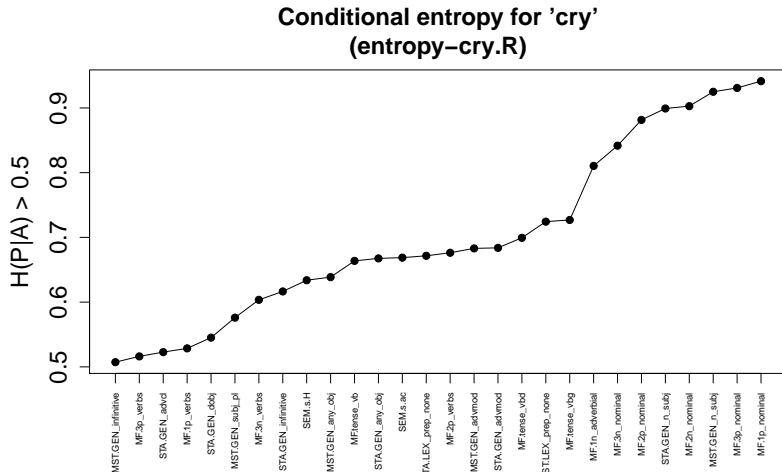
# Association between feature and target value

## Conditional entropy – VPR data (cry)



# Association between feature and target value

## Conditional entropy – VPR data (cry)





# What values do we see

## Analyzing distributions of values – VPR data (cry)

### Filter out ineffective features from the CRY data

```
> examples <- read.csv("cry.development.csv", sep="\t")
> n <- nrow(examples)
> ## remove id and target class tp
> c <- examples[,-c(1,ncol(examples))]

> ## remove features with 0s only
> c.1 <- c[, !lapply(c,fr) == 0 ]
> ## remove features with 1s only
> c.2 <- c.1[, !lapply(c.1,fr) == n ]
> ## remove column duplicates
> # unique removes duplicate rows
> c.effective <- data.frame(t(unique(t(as.matrix(c.2)))))

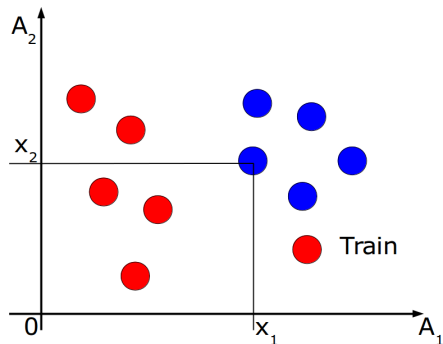
> ncol(c)                # get the number of input features
[1] 363
> ncol(c.effective)      # get the number of effective features
[1] 168
```

# Clustering

## Supervised vs. Unsupervised learning

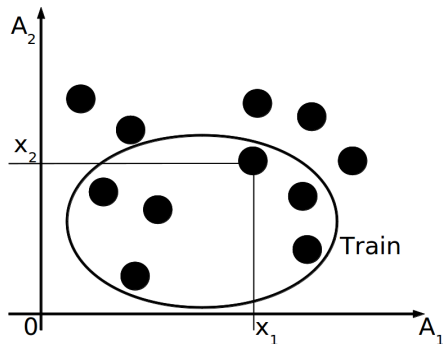
### Supervised learning

$$\text{Data} = \{\langle \mathbf{x}, y \rangle : \mathbf{x} \in X, y \in Y\}$$



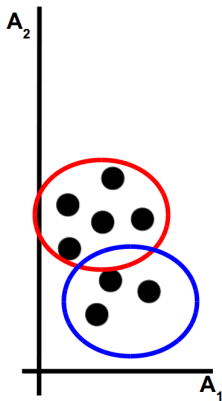
### Unsupervised learning

$$\text{Data} = \{\mathbf{x} : \mathbf{x} \in X\}$$



# Clustering

Clustering finds homogenous subgroups among the instances in the unlabeled data.

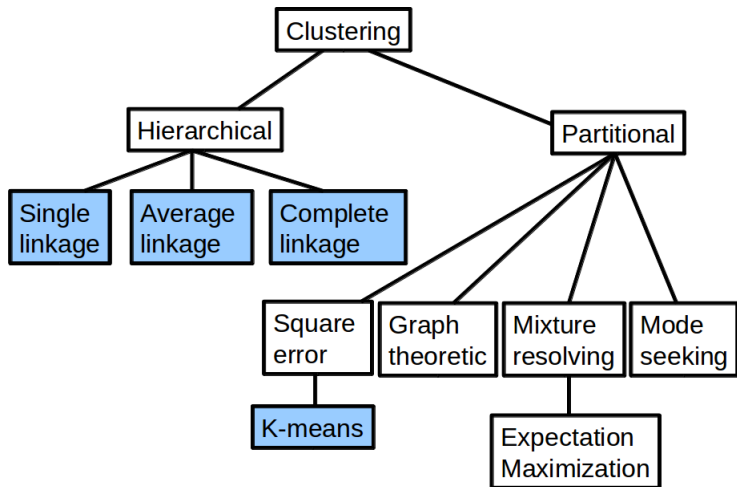


## Discovering structure

The most common criteria

- **Homogeneity**  
Objects within a same cluster should be similar each other
- **Separation**  
Objects in different clusters should be dissimilar from each other

# Clustering algorithms



Credits: (Kononenko, Kukar, 2007)

# Clustering algorithms

- $Data = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
- A set of  $k$  clusters  $C = \{C_1, C_2, \dots, C_k\}$  containing the indices of the instances
- $C_1 \cup \dots \cup C_k = \{1, 2, \dots, n\}$
- $C_j \cap C_i = \emptyset, \forall i \neq j$
- $i$ -th cluster centroid  $\mu(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$

# K-means algorithm

- **similarity function**  $\text{sim}_{ij} = \text{sim}(\mathbf{x}_i, \mathbf{x}_j)$

①  $\text{sim}_{ij} \geq 0$

②  $\text{sim}_{ii} = 1$

③  $\text{sim}_{ij} \leq \text{sim}_{ik} + \text{sim}_{kj}$

④  $\text{sim}_{ij} = \text{sim}_{ji}$

positivity

constancy of self-similarity

triangular inequality

symmetry

- **squared Euclidean distance**

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2^2 = \sum_{i=1}^m (u_i - v_i)^2$$

- **Loss function  $L(C_i)$ : Within-cluster variation**

$$L(C_i) = \frac{1}{|C_i|} \sum_{\mathbf{x}_l, \mathbf{x}_j \in C_i} \|\mathbf{x}_l - \mathbf{x}_j\|_2^2$$

- **Loss function  $L(C_1, \dots, C_K)$ : Total within-cluster variation**

$$L(C_1, \dots, C_K) = \sum_{i=1}^K L(C_i)$$

# K-means algorithm

## K-means optimization problem

$$\operatorname{argmin}_{C_1, \dots, C_K} L(C_1, \dots, C_K) = \operatorname{argmin}_{C_1, \dots, C_K} \sum_{i=1}^K L(C_i) \quad (1)$$

The number of all possible assignments of  $n$  instances into  $K$  clusters is

$$\frac{1}{K!} \sum_{k=1}^K (-1)^{K-k} \binom{K}{k} k^n$$

Computationally expensive :- ( But ...

- Prove

$$\frac{1}{C_k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C_k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = 2 \sum_{\mathbf{x} \in C_k} \|\mathbf{x} - \mu(C_k)\|_2^2$$



# K-means algorithm

- 1 Create clusters  $C_1^0, \dots, C_K^0$ :
  - randomly assign a number, from 1 to  $K$ , to each of the instance
- 2 while *total within-cluster variation does change* do
  - a) for all clusters  $C_i^t, i = 1, \dots, K$  do

$$\mu(C_i^t) = \frac{1}{|C_i^t|} \sum_{\mathbf{x} \in C_i^t} \mathbf{x}$$

- b) for all clusters  $C_i^t, i = 1, \dots, K$  do

$$C_i^{t+1} = \{\mathbf{x}; \|\mathbf{x} - \mu(C_i^t)\|_2^2 \leq \|\mathbf{x} - \mu(C_l^t)\|_2^2, \forall l \neq i\}$$

This algorithm finds a local rather than a global optimum.

# Dataset USArrests from the base R distribution

– statistics in arrests per 100,000 residents in each of the 50 US states in 1973

```
> attributes(USArrests)
$names #UrbanPop is the percent of the population living in urban areas
[1] "Murder"    "Assault"    "UrbanPop"    "Rape"
$class
[1] "data.frame"
$row.names
[1] "Alabama"      "Alaska"      "Arizona"      "Arkansas"
[5] "California"    "Colorado"    "Connecticut"  "Delaware"
[9] "Florida"      "Georgia"     "Hawaii"       "Idaho"
[13] "Illinois"     "Indiana"     "Iowa"         "Kansas"
[17] "Kentucky"     "Louisiana"   "Maine"        "Maryland"
[21] "Massachusetts" "Michigan"    "Minnesota"    "Mississippi"
[25] "Missouri"     "Montana"     "Nebraska"     "Nevada"
[29] "New Hampshire" "New Jersey"  "New Mexico"   "New York"
[33] "North Carolina" "North Dakota" "Ohio"         "Oklahoma"
[37] "Oregon"       "Pennsylvania" "Rhode Island"  "South Carolina"
[41] "South Dakota"  "Tennessee"   "Texas"        "Utah"
[45] "Vermont"      "Virginia"    "Washington"   "West Virginia"
[49] "Wisconsin"     "Wyoming"
```

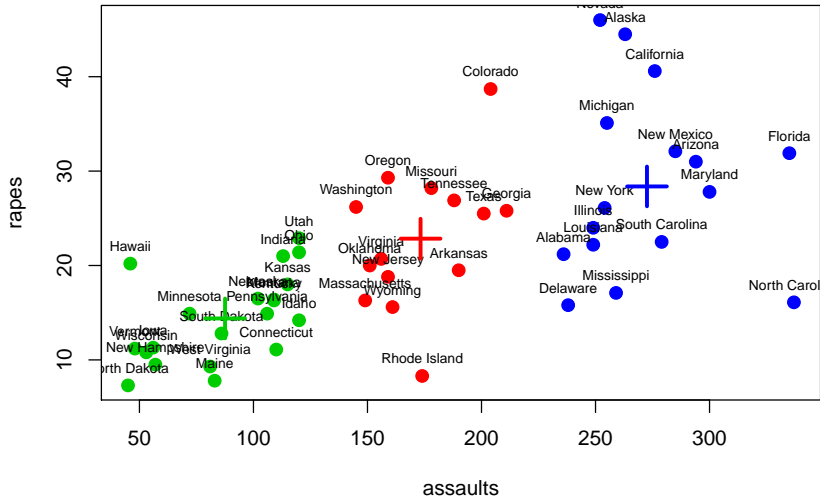
# K-means algorithm with the USArrests data

```
> str(USArrests)
'data.frame': 50 obs. of  4 variables:
 $ Murder   : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault  : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop : int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape     : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 ...

> d <- USArrests
> examples <- d[,c(2,4)]
> km.3 <- kmeans(examples, 3, nstart=20)
> km.3$tot.withinss
[1] 38435.53
> km.3$withinss
[1] 15847.167 7109.191 15479.168
```

# K-means algorithm with the USArrests data

K-means, K = 3



# K-means algorithm with the USArrests data

Multiple initial cluster assignments (nstart)

```
> km.1 <- kmeans(examples,6,nstart=1)
> km.1$tot.withinss
[1] 13711.87
>
> km.20 <- kmeans(examples,6,nstart=20)
> km.20$tot.withinss
[1] 10282.92
```

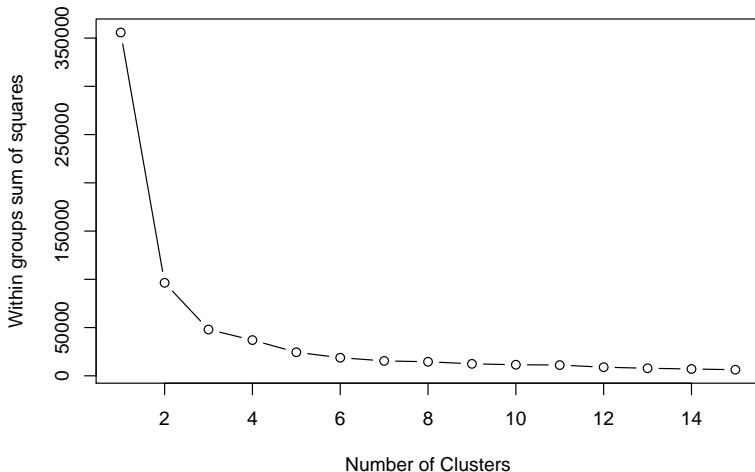
# K-means algorithm with the USArrests data

## Which $K$ to choose?

```
> km.3 <- kmeans(d, 3, nstart=20)
> km.3$tot.withinss
[1] 47964.27
> km.3$withinss
[1] 9136.643 19263.760 19563.863
>
> wssplot(d) # to be presented in the lab
```

# K-means algorithm with the USArrests data

Which  $K$  to choose?



# K-means algorithm

## Remarks

- K-means algorithm converges to a local optimum of the loss function  $L$ . It takes less than  $\leq K^n$  iterations.
- The results depend on the initial clusters. The standard solution is to try a number of different starting points (see `nstart` in R).
- It can happen that the set of instances closest to  $\mu(C_i)$  is empty, so that  $C_i$  cannot be updated. This is an annoyance that must be handled in an implementation.
- The results depend on the metric used to measure similarity.
- The results depend on the value of  $K$ .



# Hierarchical clustering methods

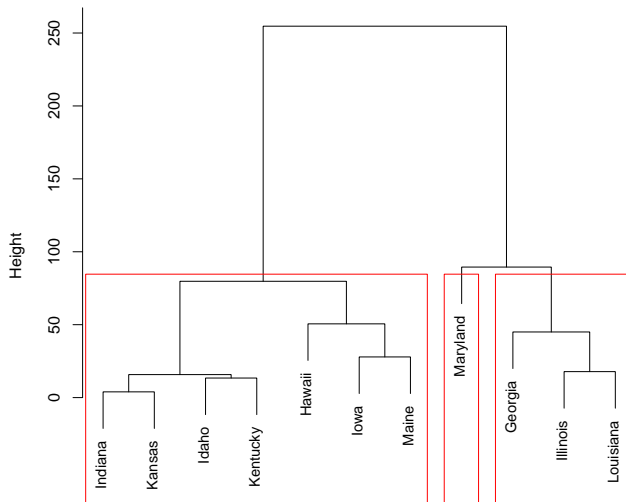
- do not require specification of the number of clusters.
- do produce tree-based representation of the instances, called **dendrogram**

# Dendrogram

is a rooted binary tree where

- the root node represents an input data set  $Data$ ,  $|Data| = n$
- the internal nodes represent the groups of instances
- each non-terminal node has two daughter nodes
- each terminal node represents one of the input instances ( $n$  terminal nodes)

# Dendrogram



# How to read a dendrogram

- location of instances on the horizontal axis says nothing about the similarity
- location on the vertical axis: dissimilarity between the clusters when they were merged
- cutting the dendrogram  $\sim$  getting clusters

# Hierarchical clustering methods

## Agglomerative (bottom-up) clustering

- 1 Start with each instance in its own singleton cluster
- 2 At each step, greedily merge 2 most similar clusters
- 3 Stop when there is a single cluster of all examples, else go to 2

## Divisive (top-down) clustering

- 1 Start with all instances in the same cluster
- 2 At each step, remove the “outsiders” from the least cohesive cluster
- 3 Stop when each example is in its own singleton cluster, else go to 2

# Agglomerative (bottom-up) hierarchical methods

- 1 for  $i := 1$  to  $n$  do  $C_i := \{\mathbf{x}_i\}$  end
- 2  $C := \{C_1, C_2, \dots, C_n\}$
- 3  $j := n + 1$
- 4 while  $|C| > 1$ 
  - 1  $(C_{n_1}, C_{n_2}) := \operatorname{argmax}_{C_u, C_v \in C \times C} \operatorname{sim}(C_u, C_v)$
  - 2  $C_j = C_{n_1} \cup C_{n_2}$
  - 3  $C := C \setminus \{C_{n_1}, C_{n_2}\} \cup C_j$
  - 4  $j := j + 1$

# Agglomerative hierarchical methods

Assume using a distance measure as a dissimilarity measure

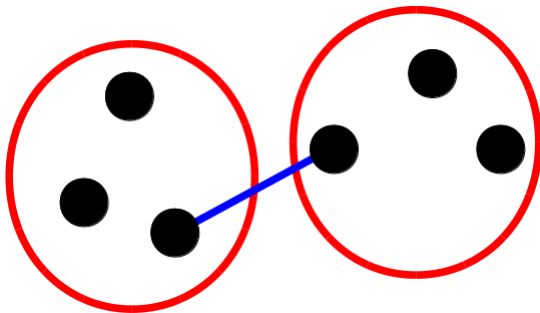
- **Dissimilarity between instances**  $d(\mathbf{x}_i, \mathbf{x}_j)$
- **Dissimilarity between clusters**  $d(C_u, C_v)$ 
  - step 4.1 of the algorithm  $(C_{n_1}, C_{n_2}) := \operatorname{argmin}_{C_u, C_v \in C \times C} d(C_u, C_v)$
  - types of linkage

# Dissimilarity between clusters

## Single linkage clustering

The minimum dissimilarity between instances of each cluster

$$d(C_u, C_v) = \min_{\mathbf{x}_i \in C_u, \mathbf{x}_j \in C_v} d(\mathbf{x}_i, \mathbf{x}_j)$$



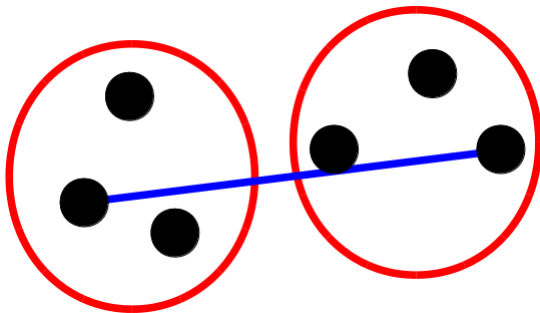


# Dissimilarity between clusters

## Complete linkage clustering

The maximum dissimilarity between instances of each cluster

$$d(C_u, C_v) = \max_{\mathbf{x}_i \in C_u, \mathbf{x}_j \in C_v} d(\mathbf{x}_i, \mathbf{x}_j)$$

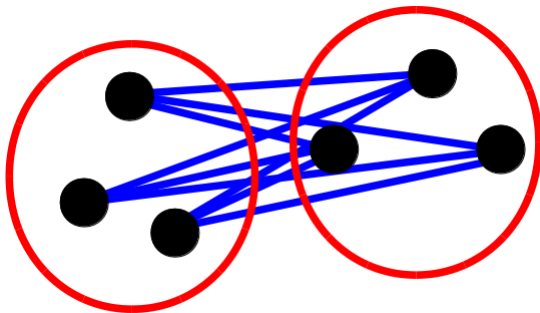


# Dissimilarity between clusters

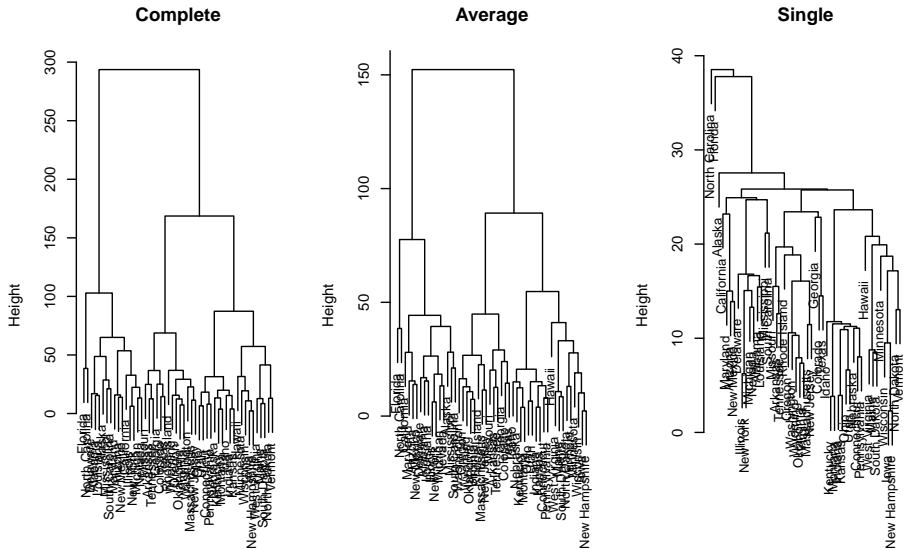
## Average linkage clustering

The mean dissimilarity between instances of each cluster

$$d(C_u, C_v) = \frac{1}{|C_u||C_v|} \sum_{\mathbf{x}_i \in C_u} \sum_{\mathbf{x}_j \in C_v} d(\mathbf{x}_i, \mathbf{x}_j)$$



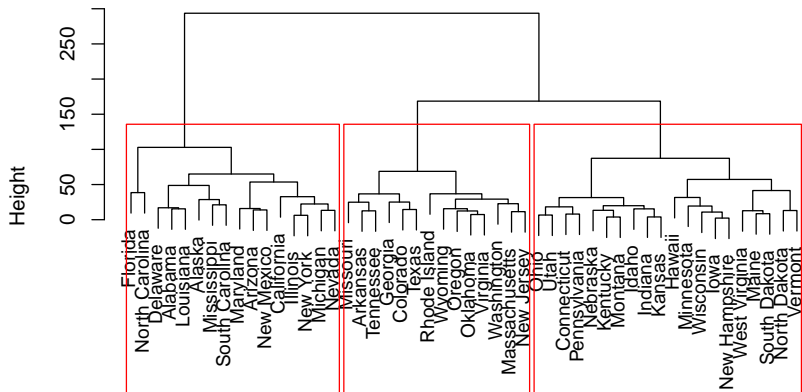
## USArrest and the linkage methods



# Cutting the dendrogram

```
> hc.complete <- hclust(dist(d), method = "complete")  
# draw dendrogram with red borders around the 3 clusters  
> rect.hclust(hc.complete, k=3, border="red")
```

Complete



# Cutting the dendrogram

```
# cut tree into 3 clusters
```

```
> cutree(hc.complete, 3)
```

Alabama	Alaska	Arizona	Arkansas	California
1	1	1	2	1
Colorado	Connecticut	Delaware	Florida	Georgia
2	3	1	1	2
Hawaii	Idaho	Illinois	Indiana	Iowa
3	3	1	3	3
Kansas	Kentucky	Louisiana	Maine	Maryland
3	3	1	3	1
Massachusetts	Michigan	Minnesota	Mississippi	Missouri
2	1	3	1	2
Montana	Nebraska	Nevada	New Hampshire	New Jersey
3	3	1	3	2
New Mexico	New York	North Carolina	North Dakota	Ohio
1	1	1	3	3
Oklahoma	Oregon	Pennsylvania	Rhode Island	South Carolina
2	2	3	2	1
South Dakota	Tennessee	Texas	Utah	Vermont
3	2	2	3	3
Virginia	Washington	West Virginia	Wisconsin	Wyoming
2	2	3	3	2

# Cutting the dendrogram

Cut the dendrogram at height  $h$ . Then

- complete linkage: for each instance  $\mathbf{x}_i$ , EVERY other instance  $\mathbf{x}_j$  in its cluster satisfies  $\text{sim}(\mathbf{x}_i, \mathbf{x}_j) \leq h$
- single linkage: for each instance  $\mathbf{x}_i$ , there is ANOTHER instance  $\mathbf{x}_j$  in its cluster satisfies  $\text{sim}(\mathbf{x}_i, \mathbf{x}_j) \leq h$
- average linkage: no interpretation

## Stopping criteria

- **Distance criterion**

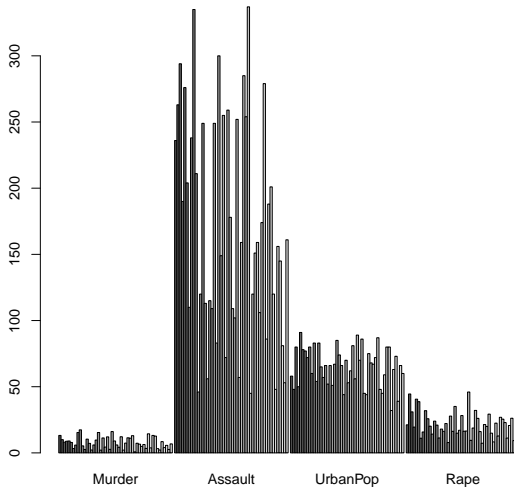
When the clusters are too far apart to be merged

- **Number criterion**

When there is sufficiently small number of clusters

# Scaling

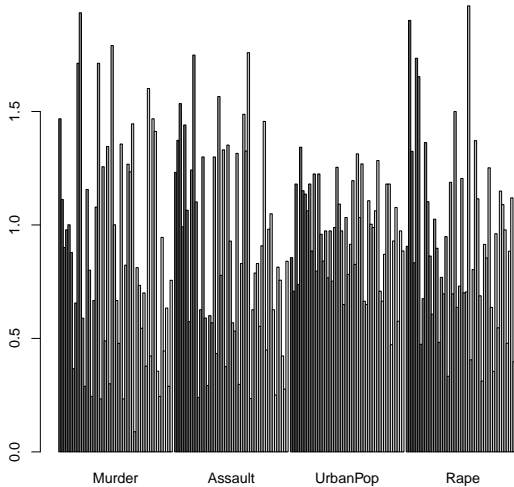
No scaling





# Scaling

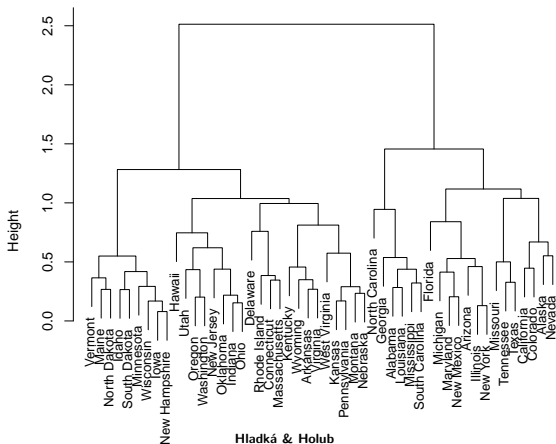
Each feature by its standard deviation



# Clustering

## Scaling each feature by its standard deviation

```
> sd.d. <- scale(as.matrix(d), center = FALSE, scale = TRUE)
> hc.complete.sd <- hclust(dist(sd.d), method = "complete")
> plot(hc.complete.sd, main = "", xlab="", sub="", cex=.9)
```



# Clustering

## Scaling each feature by its standard deviation

```
> hc <- cutree(hc.complete,3)
> hc.sd <- cutree(hc.complete.sd, 3)
> table(hc, hc.sd)
      hc.sd
hc      1      2      3
  1      5     10      1
  2      1      4      9
  3      0      0     20
> table(hc)
hc
  1      2      3
16 14 20
> table(hc.sd)
hc.sd
  1      2      3
 6 14 30
```

# Divisive (top-down) hierarchical methods

- 1  $C := \{Data\}$
- 2  $j := 1$
- 3 while  $\exists C_i \in C$  s.t.  $|C_i| > 1$
- 4  $C_u := \operatorname{argmin}_{C_v \in C} \operatorname{coh}(C_v)$ , where  $\operatorname{coh}(C_v) = \max d(\mathbf{x}_k, \mathbf{x}_l), \mathbf{x}_k, \mathbf{x}_l \in C_v$  (to obtain homogenous clusters)
- 5  $(C_{j+1}, C_{j+2}) = \operatorname{split}(C_u) = \min_{h \neq k} \max_{i \in C_h, j \in C_k} d(\mathbf{x}_i, \mathbf{x}_j)$  - minimum dissimilarity between an object of the given cluster and an object of a different cluster
- 6  $C := C \setminus \{C_u\} \cup \{C_{j+1}, C_{j+2}\}$
- 7  $j := j + 2$

# Divisive (top-down) hierarchical methods

## Divisive Coefficient (DC)

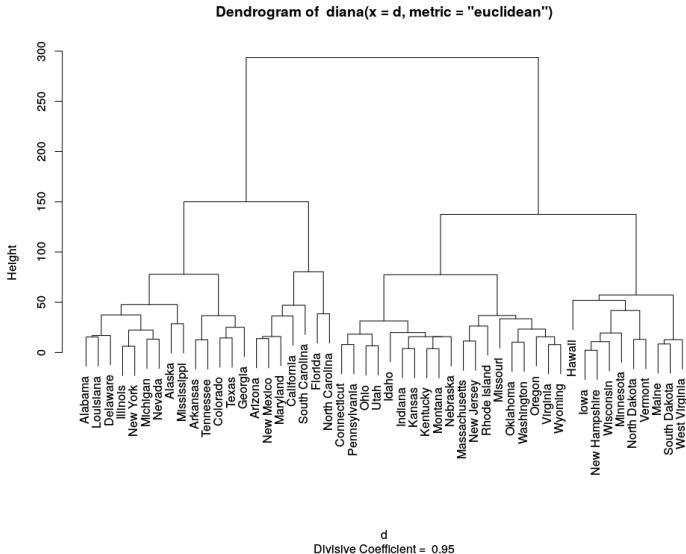
- The **diameter of a cluster** is the maximum dissimilarity between pairs of objects in the cluster.
- For each instance  $\mathbf{x}_i$ , let  $d(\mathbf{x}_i)$  denote the diameter of the last cluster to which it belongs (before being split off as a single object), divided by the diameter of the whole data set.
- The **divisive coefficient** (DC) indicates the strength of the clustering structure found by the algorithm:

$$DC(Data) = \sum_{i=1}^n d(\mathbf{x}_i) / n$$

# Divisive (top-down) hierarchical methods

```
> library(cluster)
> dc <- diana(d, metric = "euclidean"); print(dc)
Order of objects:
 [1] Alabama      Louisiana    Delaware    Illinois    New York
 [6] Michigan     Nevada      Alaska      Mississippi Arkansas
[11] Tennessee    Colorado    Texas       Georgia     Arizona
[16] New Mexico    Maryland    California  South Carolina Florida
[21] North Carolina Connecticut Pennsylvania Ohio        Utah
[26] Idaho         Indiana     Kansas      Kentucky    Montana
[31] Nebraska      Massachusetts New Jersey  Rhode Island Missouri
[36] Oklahoma      Washington  Oregon      Virginia    Wyoming
[41] Hawaii        Iowa        New Hampshire Wisconsin   Minnesota
[46] North Dakota  Vermont     Maine       South Dakota West Virginia
Height:
 [1] 15.454449 16.976749 37.430469 6.236986 22.366046 13.297368
...
[19] 80.332123 38.527912 293.622751 8.027453 18.264994 6.637771
[25] 31.477135 19.904271 3.929377 15.766420 3.834058 15.766420
...
[43] 10.860018 19.437592 41.784447 13.044922 57.271022 8.537564
[49] 12.775367
Divisive coefficient:
[1] 0.9464692
```

# Divisive (top-down) hierarchical methods



# Summary of Examination Requirements

- Methods for basic data exploration – plotting and summarizing
- Association between features
- Clustering algorithms: K-means, hierarchical agglomerative