

# Informatika pre biológov

**Broňa Brejová**

**28.9.2017**

## Formulácia problému, algoritmus

- **Formulácia problému:** jasne zadefinujeme vstupné a výstupné dáta a aký výstup očakávame pre každý vstup.
- Vo formulácii nehovoríme **akým spôsobom** vypočítame výstupy zo vstupov.
- **Správny algoritmus:** Postup, ktorý určuje spôsob, akým pre každý vstup vypočítame príslušný výstup.

**Biologický problém:** Pomocou hmotnostného spektrometra (mass spectrometer) sme odmerali vo vzorke peptid s hmotnosťou  $K$ . Máme databázu proteínov a chceme zistiť, ktorý z proteínov obsahuje peptid s touto hmotnosťou.

**Informatický problém:** Vstup je postupnosť  $n$  kladných čísel  $a[1], \dots, a[n]$  a číslo  $K$ . Nájdite súvislý úsek tejto postupnosti  $a[i], a[i + 1], \dots, a[j]$ , ktorý svojim súčtom dáva číslo  $K$ .

**Príklad:**

$K=19$

3 4 6 3 6 4 9 2 8  
    ^ ^ ^ ^ ^ ^ ^

**Informatický problém:** Vstup je postupnost  $n$  kladných čísel  $a[1], \dots, a[n]$  a číslo  $K$ . Nájdiť súvislý úsek tejto postupnosti  $a[i], a[i + 1], \dots, a[j]$ , ktorý svojim súčtom dáva číslo  $K$ .

**Triviálne riešenie:** skúšame všetky možnosti

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |   suma := suma + a[u]
|   |   ak suma = K, vypíš i,j
```

K=19

3 4 6 3 6 4 9 2 8

    i      j

## Ako dlho takýto program pobeží?

- Naimplementovať do počítača a odmerať
- Na akom počítači? Na akých vstupoch?
- **Časová zložitosť:** (označujeme  $O(f(n))$ )
  - Zvolíme si parameter charakterizujúci množstvo dát napr. počet prvkov vstupnej postupnosti  $n$
  - Pre každú veľkosť vstupu **odhadneme najhorší prípad**
  - Zanedbáme konštanty

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |   suma := suma + a[u]
|   |   ak suma = K, vypíš i,j
```

**Časová zložitosť:** kubická, alebo  $O(n^3)$

## Prečo je časová zložitosť dôležitá a konštanty nie?

		$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
Čas na	10	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$
vyriešenie	50	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	2 weeks
problému	100	$\varepsilon$	$\varepsilon$	$\varepsilon$	$\varepsilon$	2800 univ.
veľkosti	1000	$\varepsilon$	$\varepsilon$	0.02s	4.5s	—
...	10000	$\varepsilon$	0.01s	2.1s	75m	—
	100000	0.04s	0.12s	3.5m	52d	—
	1 mil.	0.42s	1.4s	5.8h	142yr	—
	10 mil.	4.2s	16.1s	24.3d	140000yr	—
Max veľkosť	1s	2.3 mil.	740000	6900	610	33
problému	1m	140 mil.	34 mil.	53000	2400	39
vyriešená za	1d	200 bil.	35 bil.	2 mil.	26000	49
Zvýšenie	+1	—	—	—	—	$\times 2$
času so	$\times 2$	$\times 2$	$\times 2+$	$\times 4$	$\times 8$	—
zvýšeným $n$						

## Efektívnejší algorimus

- Najprv si predpočítame pre každý začiatok postupnosti jej súčet

$$S[i] = a[1] + a[2] + \dots + a[i]$$

$$S[0] := 0$$

pre každé  $i$  od 1 po  $n$

$$| \quad S[i] = S[i-1] + a[i]$$

a: 3 4 6 3 6 4 9 2 8

S: 3 7 13 16 22 26 35 ...

- Potom súčet podpostupnosti od  $i$  po  $j$  vieme spočítať jednoducho ako  $S[j] - S[i - 1]$

pre každé  $i$  od 1 po  $n$

| pre každé  $j$  od  $i$  po  $n$

| | ak  $S[j] - S[i-1] = K$ , vypíš  $i, j$

- **Časová zložitosť:** kvadratická, alebo  $O(n^2)$
- Ak sú všetky čísla kladné, dá sa aj v lineárnom čase  $O(n)$

## Ďalší příklad infromatického problému z prednášky

### Najkratšie spoločné nadslovo

- Vstup: niekoľko reťazcov
- Výstup: najkratší reťazec, ktorý obsahuje všetky vstupné reťazce ako súvislé podreťazce

### Príklad:

Vstup: GCCAAC, CCTGCC, ACCTTC

Výstup: CCTGCCAACCTTC (najkratšie možné)

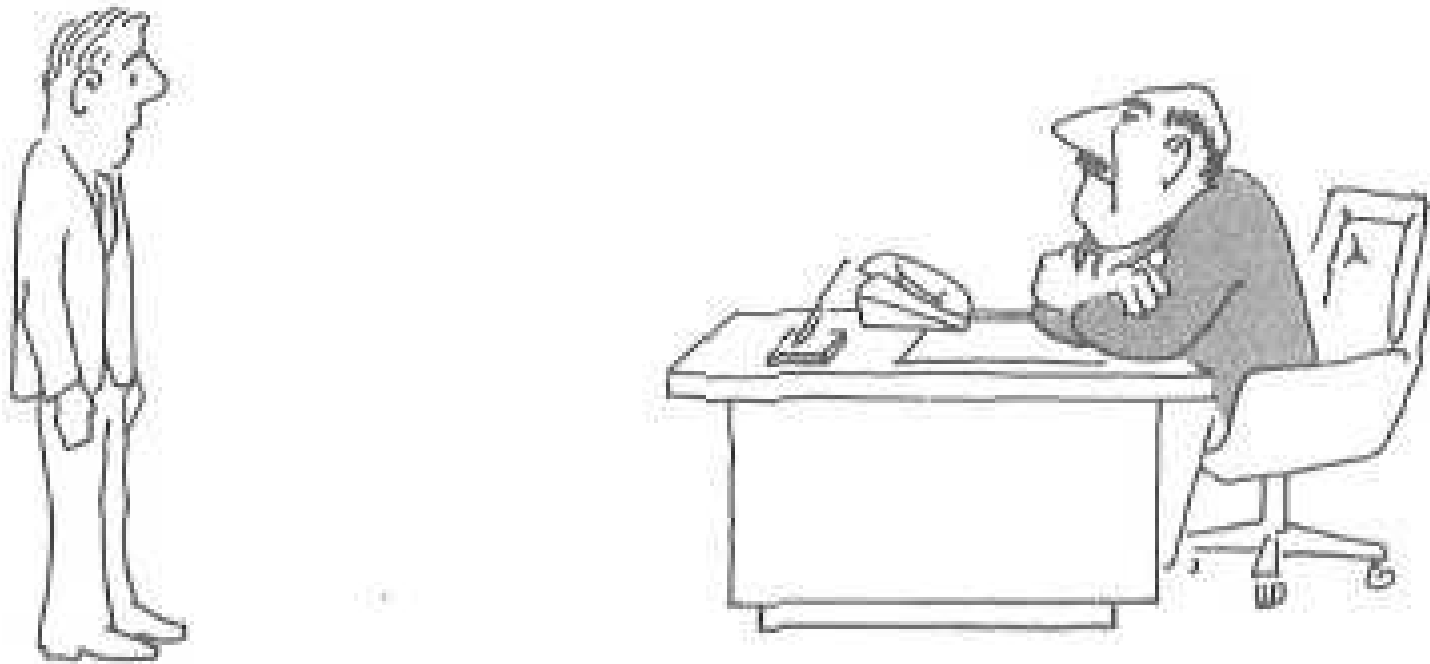


**Ako rýchly algoritmus poznáme pre tento problém?**

**Najkratšie spoločné nadslovo**

Nepoznáme algoritmus, ktorý by bežal v polynomiálnom čase  
t.j.  $O(n^k)$  pre nejakú konštantu  $k$

Tento problém je **NP-ťažký**.



“I can’t find an efficient algorithm, I guess I’m just too dumb.”



"I can't find an efficient algorithm, because no such algorithm is possible!"



"I can't find an efficient algorithm, but neither can all these famous people."

## Ako so vysporiadať s NP-ťažkými problémami?

### Heuristické algoritmy

- Nájde **aspoň nejaké riešenie**, aj keď nie nutne optimálne
- Nejde teda o správny algoritmus riešiaci náš problém, lebo pre niektoré vstupy dáva zlé odpovede
- Radšej ale horšia odpoveď rýchlo, ako perfektná o milión rokov

**Príklad:** Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Príklad: CATATAT, TATATA, ATATATC

Optimum: CATATATATC, dĺžka 10

Heuristika: CATATATCTATATA, dĺžka 14

## Ako so vysporiadať s NP-ťažkými problémami?

### Aproximačný algoritmus

- Často vieme dokázať, že nejaká heuristika sa vždy priblíži k optimálnemu riešeniu aspoň po určitú hranicu

**Príklad:** Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Je dokázané, že vždy nájde najviac 3,5-krát dlhší reťazec ako najlepšie riešenie.

Informatici predpokladajú, že v skutočnosti najviac 2-krát dlhší, ale nevieme to dokázať.

## Ako so vysporiadať s NP-ťažkými problémami?

### Exaktný výpočet pomocou iného problému

- Preformulovať do podoby jedného z dobre známych NP-ťažkých problémov (napr. celočíselné lineárne programovanie, a pod.)
- Múdri ľudia napísali programy, ktoré vedia riešiť tieto známe problémy **aspoň v niektorých prípadoch** (CONCORD, CPLEX, a pod.)

### Preformulovať problém

- Je toto skutočne jediná rozumná formulácia biologického problému ktorý chceme vyriešiť?

## Zhrnutie

- Problémy zo skutočného života je dobré najskôr sformulovať tak, aby bolo jasné, aké výsledky očakávame pre každý možný vstup.
- Takáto formulácia by mala byť oddelená od postupu (algoritmu) riešenia.
- Informatici merajú čas v O-čkach, ktoré abstrahujú od detailov konkrétneho počítača.
- Vytvorenie efektívneho algoritmu je umenie! Časť z toho sú finty (ako napr. dynamické programovanie).
- Pre niektoré problémy poznáme iba Nechutne Pomalé algoritmy (NP-ťažké).
- Aj napriek tomu vo veľa prípadoch vieme pomôcť.



# Úvod do dynamického programovania (cvičenie)

**Broňa Brejová**  
**5.10.2017**

## Problém platenia minimálnym počtom mincí

**Vstup:** hodnoty  $k$  mincí  $m_1, m_2, \dots, m_k$  a cieľová suma  $X$  (všetko kladné celé čísla)

**Výstup:** najmenší počet mincí, ktoré potrebujeme na zaplatenie  $X$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 2, m_3 = 5, X = 13$

**Odbočka: ešte matematickejšia formulácia bez slov minca, suma,...**

**Vstup:** kladné celé čísla  $m_1, m_2, \dots, m_k$  a  $X$

**Výstup:** celé číslo  $n$  a  $n$  čísel  $x_1, \dots, x_n$  také že platia nasledujúce podmienky:

- $x_i \in \{m_1, m_2, \dots, m_k\}$  pre každé  $i = 1, 2, \dots, n$
- $\sum_{i=1}^n x_i = X$
- $n$  je najmenšie možné.

## Problém platenia minimálnym počtom mincí

**Vstup:** hodnoty  $k$  mincí  $m_1, m_2, \dots, m_k$  a cieľová suma  $X$  (všetko kladné celé čísla)

**Výstup:** najmenší počet mincí, ktoré potrebujeme na zaplatenie  $X$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 2, m_3 = 5, X = 13$

**Príklad:**  $k = 3, m_1 = 1, m_2 = 3, m_3 = 4, X = 6$

## Algoritmus pre všeobecnú sústavu $k$ mincí $m_1, m_2, \dots, m_k$

$$A[i] = 1 + \min\{A[i - m_1], A[i - m_2], \dots, A[i - m_k]\}$$

$A[0] = 0;$

pre každé  $i$  od 1 po  $X$

$\text{min} = \text{nekonecno}$

    pre každé  $j$  od 1 po  $k$

        ak  $i \geq m[j]$  a  $A[i - m[j]] < \text{min}$

$\text{min} = A[i - m[j]]$

$A[i] = 1 + \text{min}$

vypis  $A[X]$

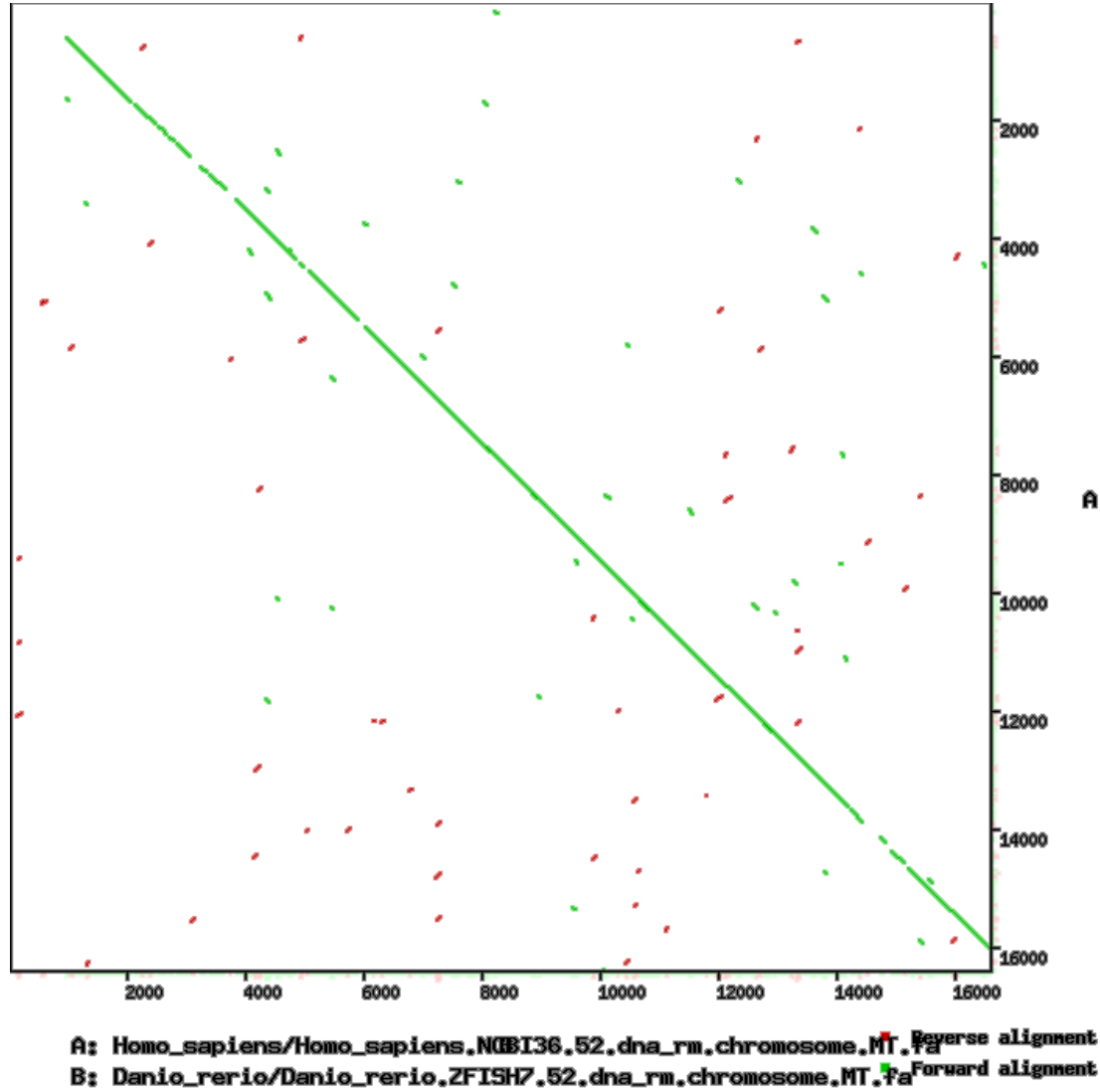
## Dynamické programovanie vo všeobecnosti

- Okrem riešenia celého problému riešime aj menšie problémy (nazývame ich podproblémy)
- Riešenia podproblémov ukladáme do tabuľky a používame pri riešení väčších podproblémov
- Technika dynamického programovania sa používa na viacero problémov v bioinformatike

**Zarovňávanie sekvencií  
(cvičenie)**

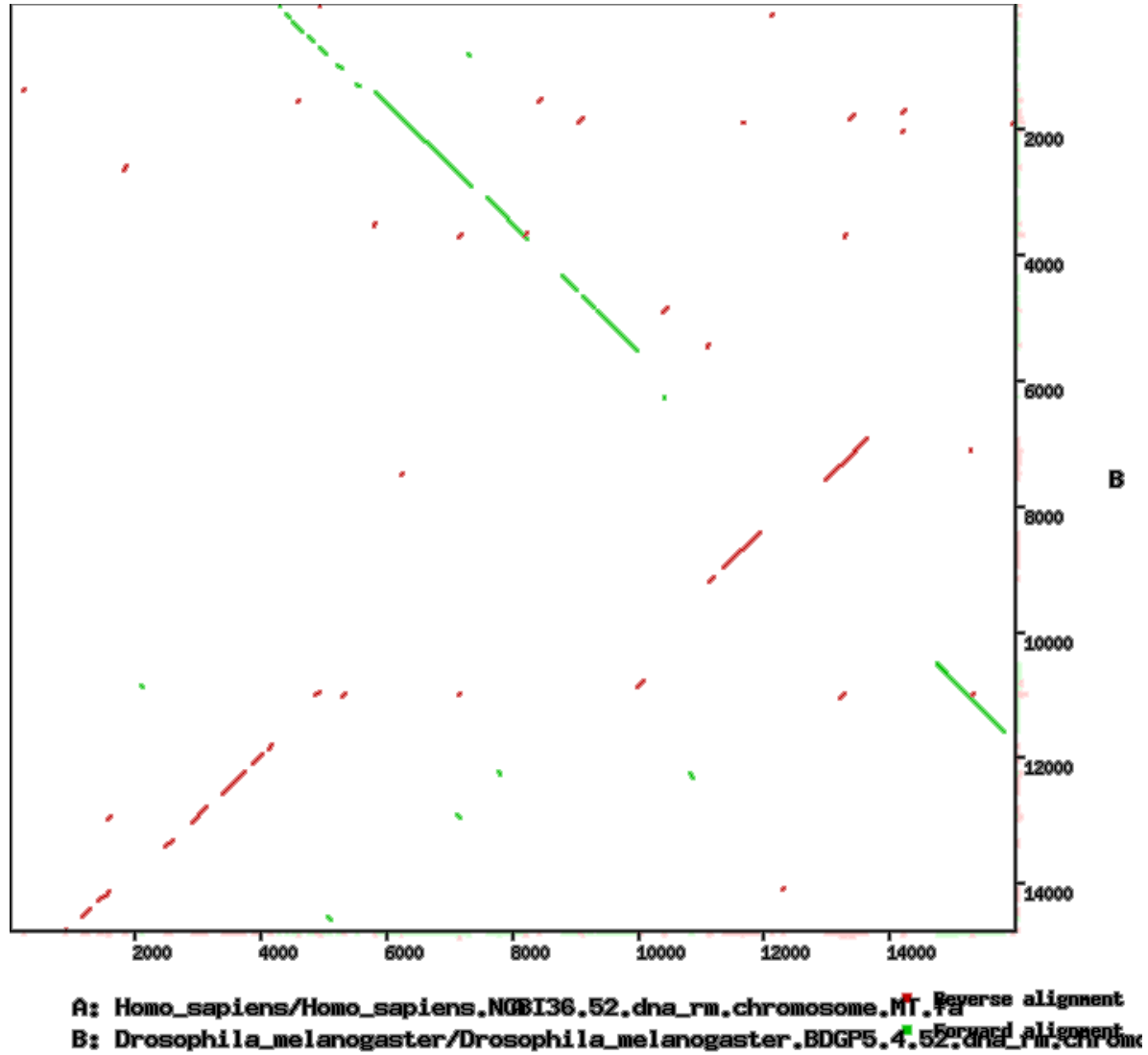
**Broňa Brejová  
12.10.2017**

# Mitochondriálny genóm človeka vs. ryba *Danio rerio*

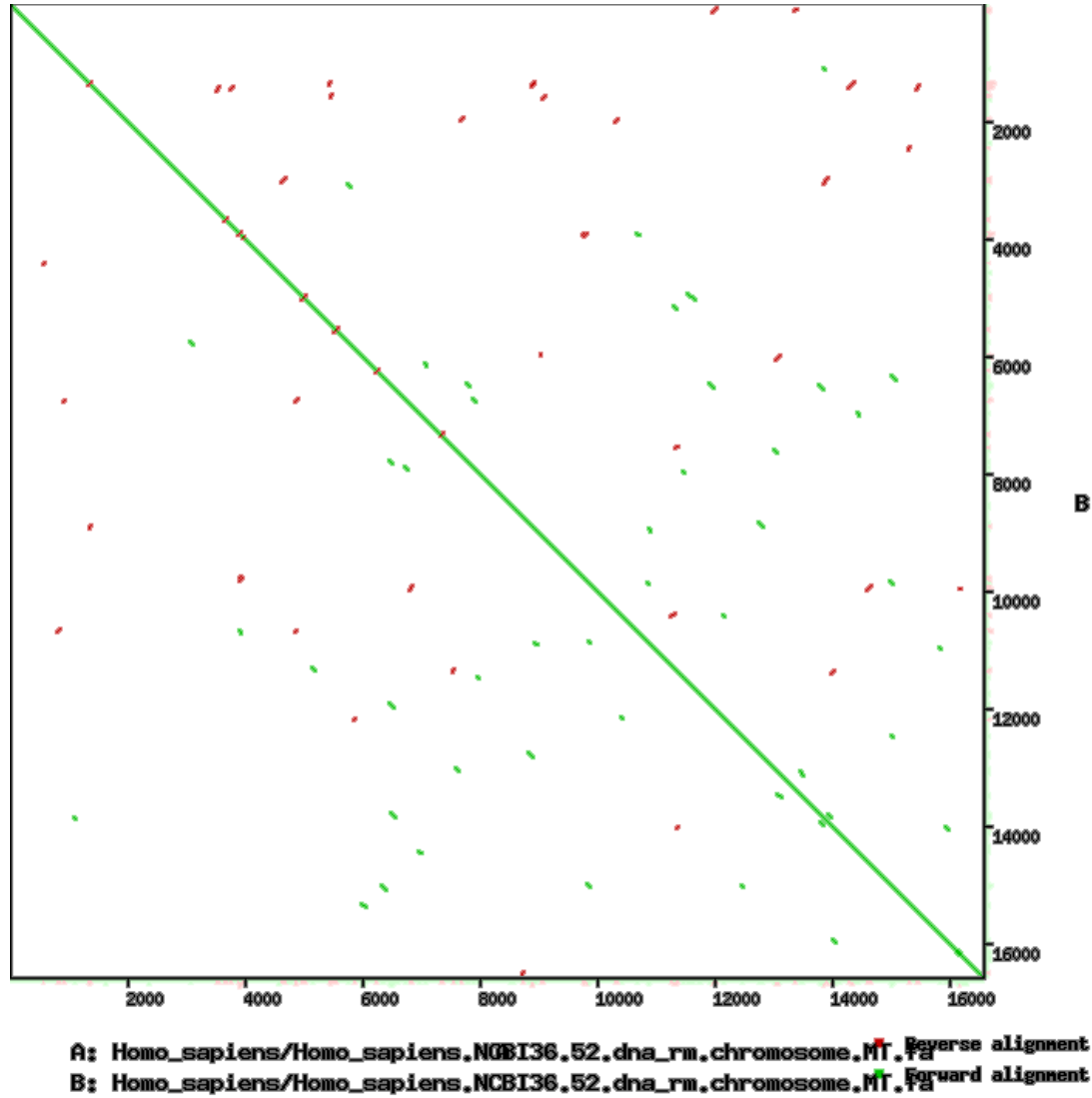


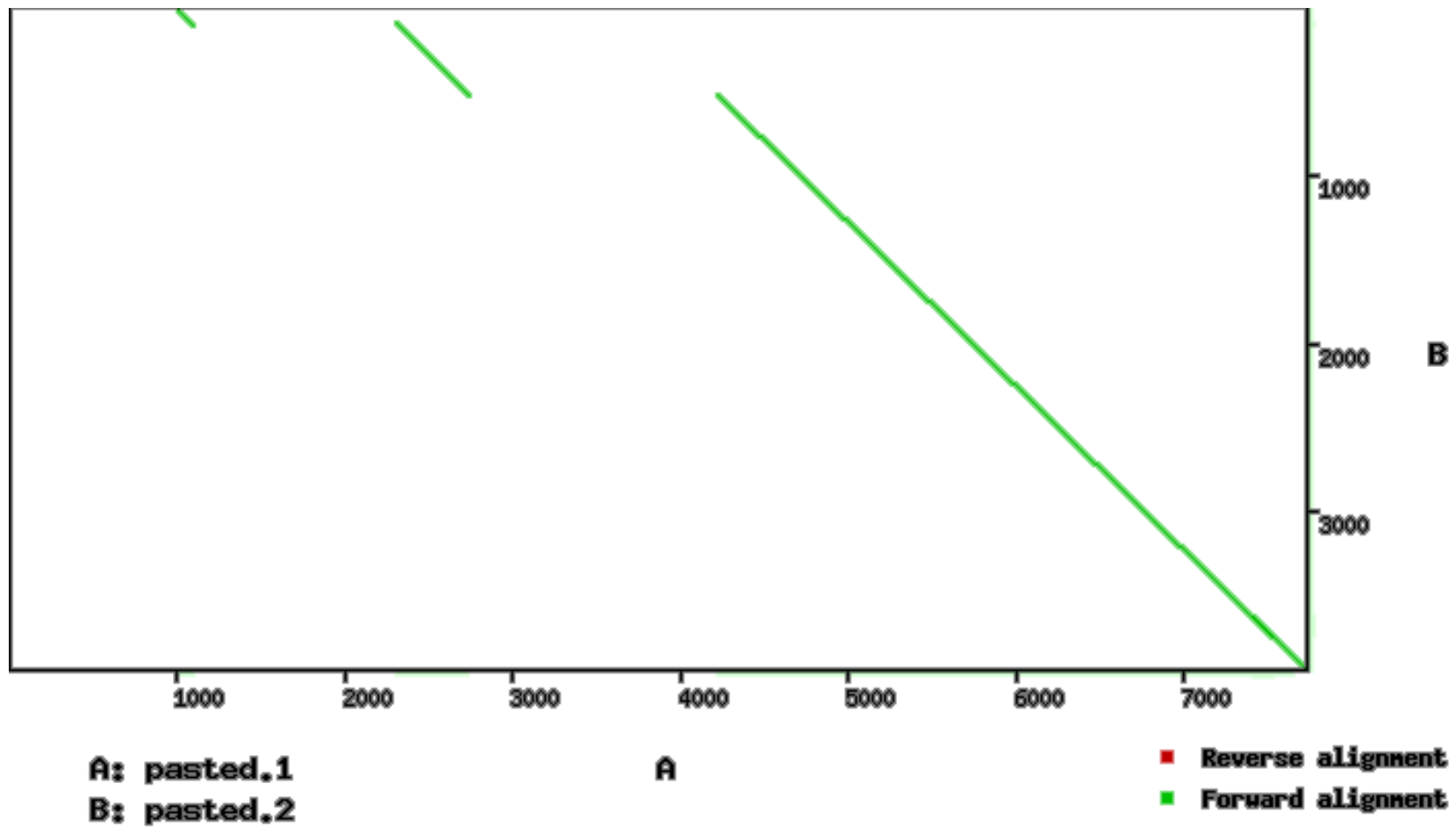


# Mitochondriálny genóm človeka vs. Drosophila melanogaster

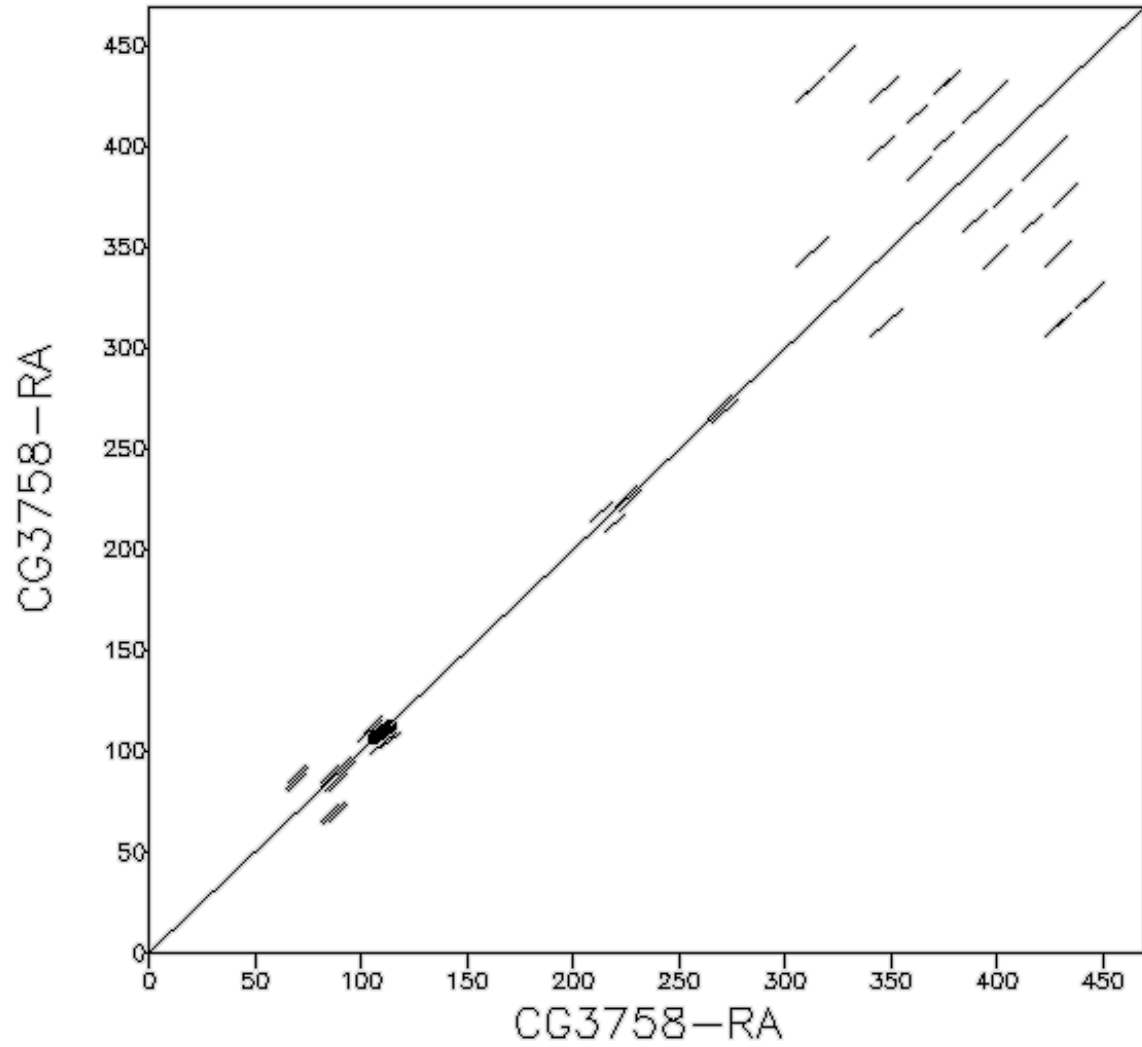


# Mitochondriálny genóm človeka vs. to isté





# Drosophila protein Escargot zinc finger vs. to isté



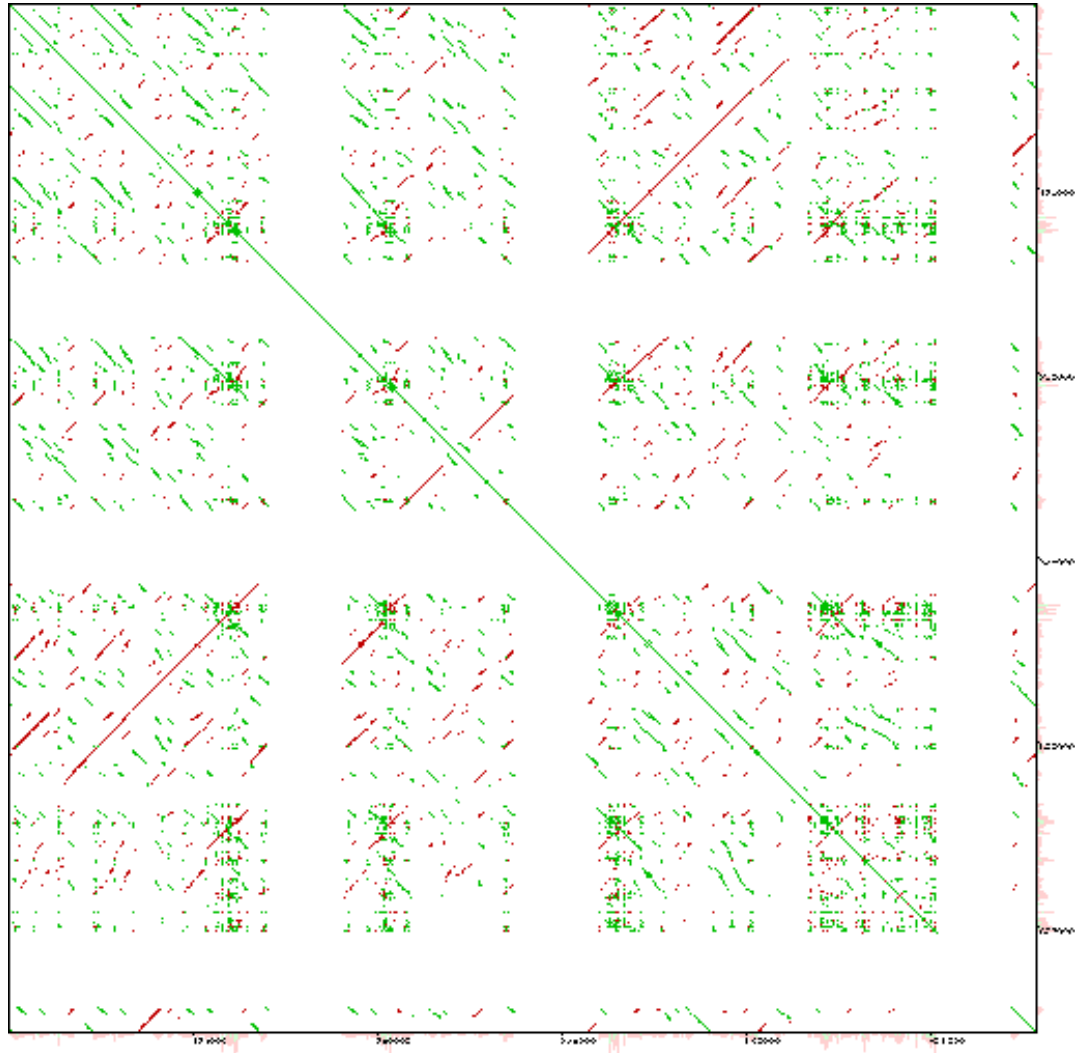
# Drosophila protein Escargot zinc finger

<b>Description:</b>	Protein escargot
<b>Source organism:</b>	<a href="#">Drosophila melanogaster (Fruit fly)</a> <a href="#">View Pfam proteome data.</a>
<b>Length:</b>	470 amino acids

## Pfam domains



Source	Domain	Start ^	End
Pfam B	<a href="#">Pfam-B 18487</a>	8	270
low_complexity		71	95
low_complexity		101	129
low_complexity		163	177
low_complexity		244	263
low_complexity		264	274
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	309	332
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	344	366
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	370	392
<b>Pfam A</b>	<a href="#">zf-C2H2</a>	398	420
low_complexity		445	460



**Pravdepodobnosť a E-value  
(cvičenie)**

**Tomáš Vinar̄  
26.10.2017**

## Hračkársky prípad

**Dotaz:** ATGCTCAAAC (dĺžka  $m = 10$ )

**Databáza:** (dĺžka  $n = 300$ )

accacttgcgcacgatttccagattcggtttccctgggcgcacgaagggc  
ccacgaagcgGCTCAACccggagccttagttagaaggggggtctccgtca  
agagagacggtaagttggaggggtcactagcggaggactccgaatggaaac  
actgaatagtggcagaacctaaccctcgttttggatttcctgaaaaaggc  
aggcgctagaggaagaggcacgactgtgctagagataatcacttgtaaga  
ccttgggggatgggcttcgtatgcagaacgcgataaggtatcgaaaacgtg

**Skórovacia schéma:** zhoda  $+1$ , nezhoda  $-1$ , medzera  $-1$

**Lokálne zarovnanie** so skóre  $S = 6$

GCTCAAAC

GCTCA-AC

**E-value:** koľko očakávame lokálnych zarovnaní so skóre aspoň  $S$   
v náhodnej databáze dĺžky  $n$  pri náhodnom dotaze dĺžky  $m$



## Náhodný dotaz a databáza

Dotaz: GTGCCTGCAG

Databáza:

cctctgatagccttgaaccgggcgagactcatacagacagtgctcctcgg  
gcgataaccatgagatgacaggtccgatgctaattgtaacggacctacag  
tgacatgttaaagtgtccattaagtttataaccggaatcaacgagtgtccc  
ccagcgcggcgaccgatggagccCCTGCAGgtatactcacttcaaggatt  
accgctcggtgtaagttagtgttcagtcagactataactaagtattcagtt  
atagagcgttagtaggtcgaccatgagcgggtaggGTGCCGAGatgtgaa

Počet výskytov: 2

## Náhodný dotaz a dazabáza

Dotaz: TCGACCGAAA

Databáza:

```
tactccattagggattataacgactaaagcccgtcgtggcgggatcactt
tgagattcaactttaacgcatcacagaggaatctgagacaaagcaaaacc
gatcataatgatcgatccaggtaataagtctccttgatggcgtagactg
gaaataacagttgacttccgactatagtttaatgaacgttcgtaattaga
cgatcgtgtaacttaaccaaaggctgcccccaaactagctgagtaatagc
tcgtcctgagcatgtaagagtcagcctccacggaacactgcaacgttctt
```

Počet výskytov: 0

## Náhodný dotaz a databáza

Dotaz: CCCGTCGTAG

Databáza:

cagcattagccccgttat~~tt~~CGTCGTtctccaacgggtctgcctttctgg  
aacgtggcgaaccttcacaggtcagtctgtcatcgcctgagccttagagcg  
gacggtactcgaaaggtcgggtcagtgtggcgctggaaagaagaatagca  
acacatgcactaatggaaggtcccagtggtgtgggacattctggaCCCGT  
GTgtgccaacctatgtgagctccggcgctgactcggaggatgttaacaag  
atcaagctgtaggagcagatccccgccgggtttcctctactgcctcgagc

Počet výskytov: 2

## Náhodný dotaz a databáza

Dotaz: AGGATGAGGA

Databáza:

ttatcgattctccggtgcgccagtacagcacaaggctcggatcctgtaaa  
acactacaccttaaaaactaagtcAGGATGtgatctcccttaaGATGAGa  
cagtctctaatacgggcgtagtgggaccctcgtgaccgagctaagcagttc  
acaatgggcgctctgagcgattggctggagaccttgacttcccggtaggt  
gtggtgtagttctgtgcccagagataaccatccaccgtaatggatctcg  
taactttacGATGAAGAccggcatcatctcagttatatttctaggacggg

Počet výskytov: 3

## Celkovo opakujeme 100 krát

$S = 6$ ,  $m = 10$ ,  $n = 300$ , obsah GC 50%

Počet výskytov: 2, 0, 2, 3, 3, 1, 0, 1, 1, 1, 0, 0, 4, 2, 0, 1, 0, 1, 0, 0, 1,  
0, 0, 4, 3, 1, 1, 0, 0, 0, 2, 3, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0, 4, 1, 1, 0, 0, 1,  
1, 1, 2, 2, 2, 0, 0, 2, 0, 1, 1, 0, 1, 2, 2, 1, 0, 0, 1, 1, 2, 0, 1, 0, 0, 1, 0,  
3, 2, 0, 2, 2, 1, 0, 0, 2, 0, 0, 1, 2, 1, 1, 3, 2, 2, 1, 1, 0, 2, 0, 1, 3

Priemerný počet výskytov: 1.05

Keď celé opakujeme viackrát, dostávame hodnoty 0.99, 1.15, 1.02,  
1.07, 0.98, ...

Správna hodnota E-value: 0.99

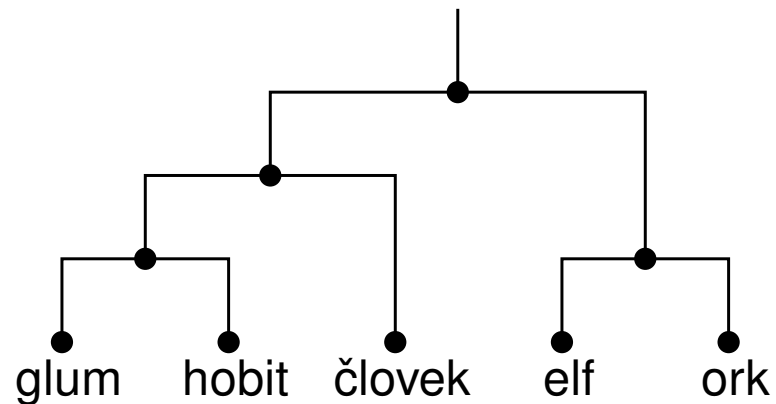
# Fylogenetické stromy

**Tomáš Vinař**

**2.11.2017**

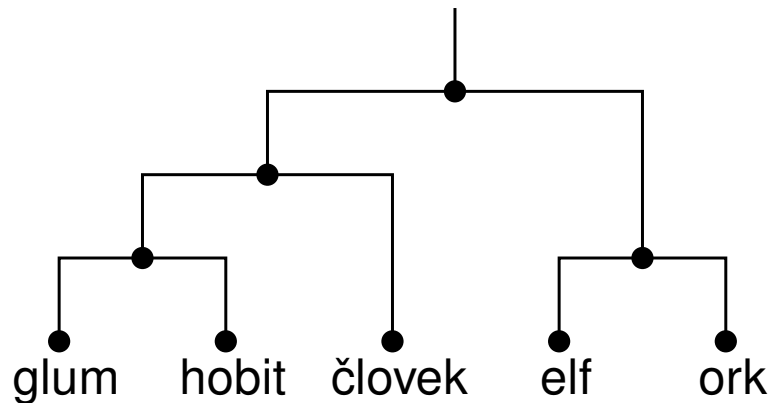
## Terminol6gia

- zakorenený strom, rooted tree
- nezakorenený strom, unrooted tree
- hrana, vetva, edge, branch
- vrchol, uzol, vertex, node
- list, leaf, leaf node, tip, terminal node
- vnútorný vrchol, internal node
- koreň, root
- podstrom, subtree, clade



## Zopár faktov o stromoch

- Majme zakorenený strom s  $n$  listami, v ktorom má každý vnútorný vrchol 2 deti. Takýto strom vždy má  $n - 1$  vnútorných vrcholov a  $2n - 2$  vetiev (prečo?)
- Majme nezakorenený strom s  $n$  listami, v ktorom má každý vnútorný vrchol 3 susedov. Takýto strom vždy má  $n - 2$  vnútorných vrcholov a  $2n - 3$  vetiev.
- Koľkými spôsobmi môžeme zakoreniť nezakorenený strom s  $n$  listami?



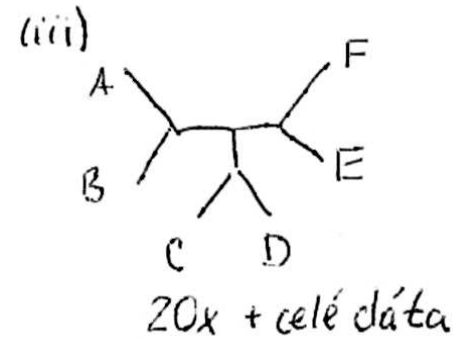
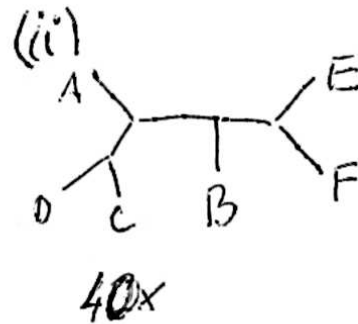
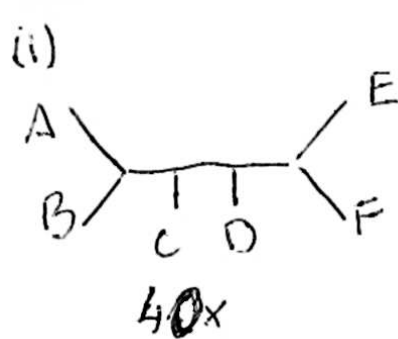


## Bootstrap

- Náhodne vyberieme niektoré stĺpce zarovnaní, zostrojíme strom
- Celé to opakujeme veľa krát
- Značíme si, koľkokrát sa ktorá hrana opakuje v stromoch (v nezakorenenom strome je hrana rozdelenie listov na dve skupiny)
- Nakoniec zostavíme strom z celých dát a pozrieme sa ako často sa ktorá jeho hrana vyskytovala
- Môžeme zostaviť aj strom z často sa vyskytujúcich hrán
- Bootstrap hodnoty sú odhadom spoľahlivosti, hlavne ak máme celkovo málo dát (krátke zarovnanie)
- Ak však dáta nezodpovedajú vybranej metóde/modelu, tak aj pre zlý strom môžeme dostať vysoký bootstrap

## Bootstrap

Robili sme 100× bootstrap, dostali sme tieto výsledky:



Doplňte bootstrap hodnoty hranám výsledného stromu (iii)

Ktoré ďalšie vetvy majú podporu aspoň 20%?

Aký strom by sme dostali, ak by sme chceli nechať iba vetvy s podporou aspoň 80%?

# Komparatívna genomika (cvičenie pre biológov)

**Broňa Brejová**

**9.11.2017**

## Objavenie génu HAR1 pomocou komparatívnej genomiky

Hľadáme úseky genómu, ktoré sa:

- dlho vyvíjali pomaly (purifikačná selekcia)
- v človeku sa vyvíjajú prekvapivo pomaly (pozitívna selekcia)

Postup: [Pollard et al. (2006) Nature]

- Všetky regióny dĺžky  $\geq 100$  s  $> 96\%$  podobnosťou medzi šimpanzom a myšou/potkanom (35,000)
- Porovnali s ostatnými cicavcami, zistili, ktoré majú veľa mutácií v človeku, ale málo inde (pravdepodobnostný model)
- 49 štatisticky významných regiónov, 96% v nekódujúcich oblastiach
- Štatisticky najvýznamnejší HAR1 (Human Accelerated Region)

## Human Accelerated Regions: HAR1

Oblasť dĺžky 118 báz

18 zmien medzi človekom a šimpanzom, 2 zmeny medzi šimpanzom a sliepkou

```
Clovek C T G A A A T G A T G G G C G T A G A C G C A C G T C A G C G G C G G A A A T G G T T T C T A T C A
Simpanz C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C A G T G G A A A T A G T T T C T A T C A
Gorila C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C A G T G G A A A T A G T T T C T A T C A
Rezus C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C A G T G G A A A T A G T T T C T A T C A
Mys C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C C G T G G A A A T G G T T T C T A T C A
Krava C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C A G T G G A A A C C G T T T C T A T C A
Pes C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C G G T G C A A A C A G T T T C T A T C A
Sliepka C T G A A A T T A T A G G T G T A G A C A C A T G T C A G C A G T A G A A A C A G T T T C T A T C A
```

- Prekrývajúce sa RNA gény HAR1R a HAR1F
- HAR1F je exprimovaný v neokortexe u 7 a 9 týždenných embrií, neskôr aj v iných častiach mozgu (u človeka aj iných primátov)
- Všetky substitúcie v človeku A/T->C/G, stabilnejšia RNA štruktúra (ale tiež sú blízko k telomére, kde takéto mutácie časté)

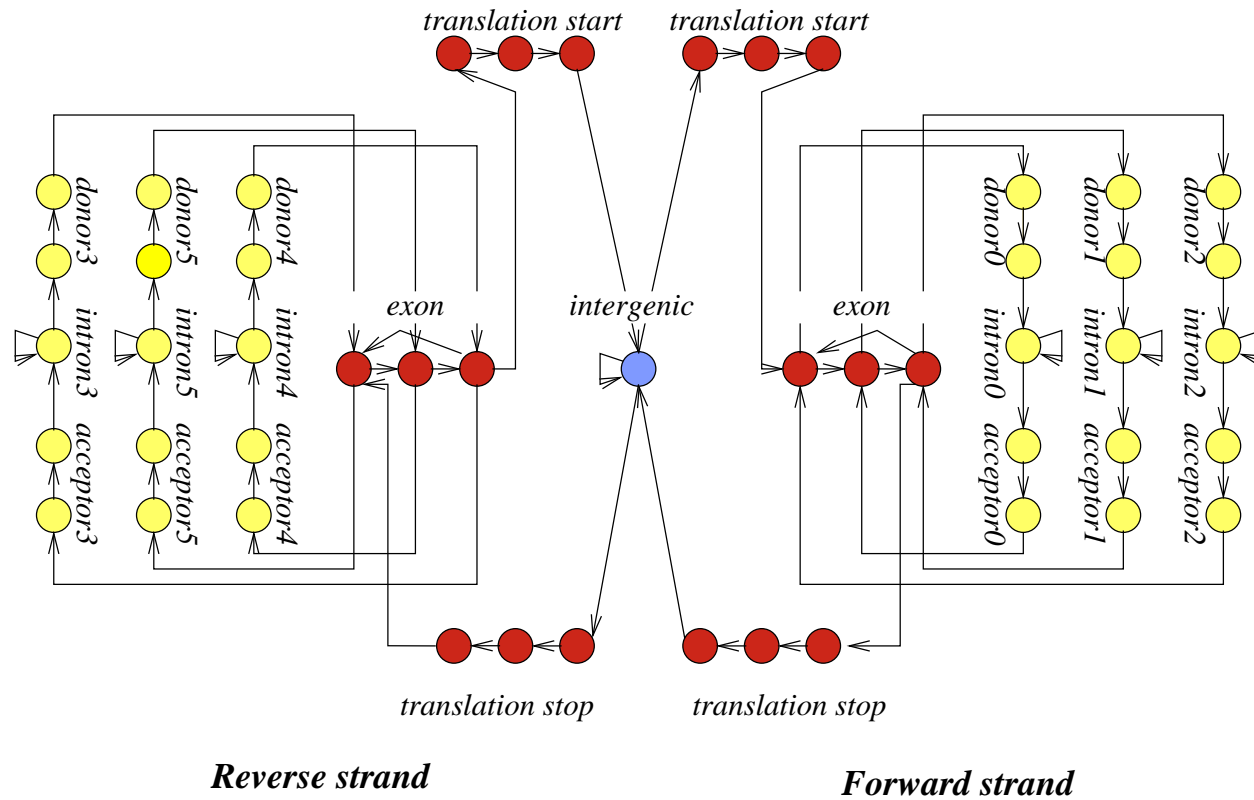
# Hľadanie génov (cvičenie)

**Broňa Brejová**

**9.11.2017**

# Hľadanie prokaryotických génov

## HMM pre eukaryotické gény:



**Prokaryotické gény jednoduchšie:** zväčša nemajú intróny

## Hľadanie prokaryotických génov

**ORF:** open reading frame, otvorený čítací rámec

- hľadanie ORFov je ľahké
- problémy:
  - ako nájsť začiatok,
  - ako rozlíšiť pseudogény a náhodné ORF-y

## Anotácia prokaryotických génov nie je triviálny problém

- E.coli sekvenovaná a anotovaná 1997
- Porovnanie s verziou 2005 (4464 génov) [Riley et al NAR 2005]  
(oprava sekvenovacích chýb aj chýb v anotácii)  
682 zmien v štart kodóne  
31 génov zrušených  
48 nových génov



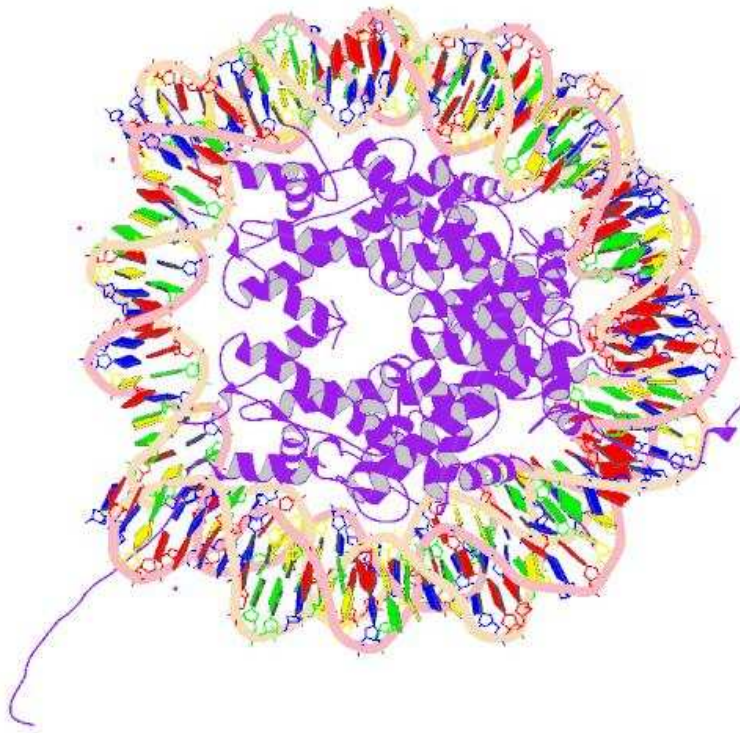
## Hľadanie génov

Ideálne kombinácia výpočtových modelov a experimentálnej informácie

- Sekvenovanie RNA-seq; RT-PCR
- Metódy na detekciu proteínov
- Komparatívna genomika
- Stav chromatinu, histónové modifikácie

## Históny a nukleozómy

- DNA v chromozómoch ovinutá okolo nukleozómov pozostávajúcich z histónov H2A, H2B, H3, H4
- 146 báz ovinutých okolo nukleozómu, cca 50 báz medzi nukleozómami

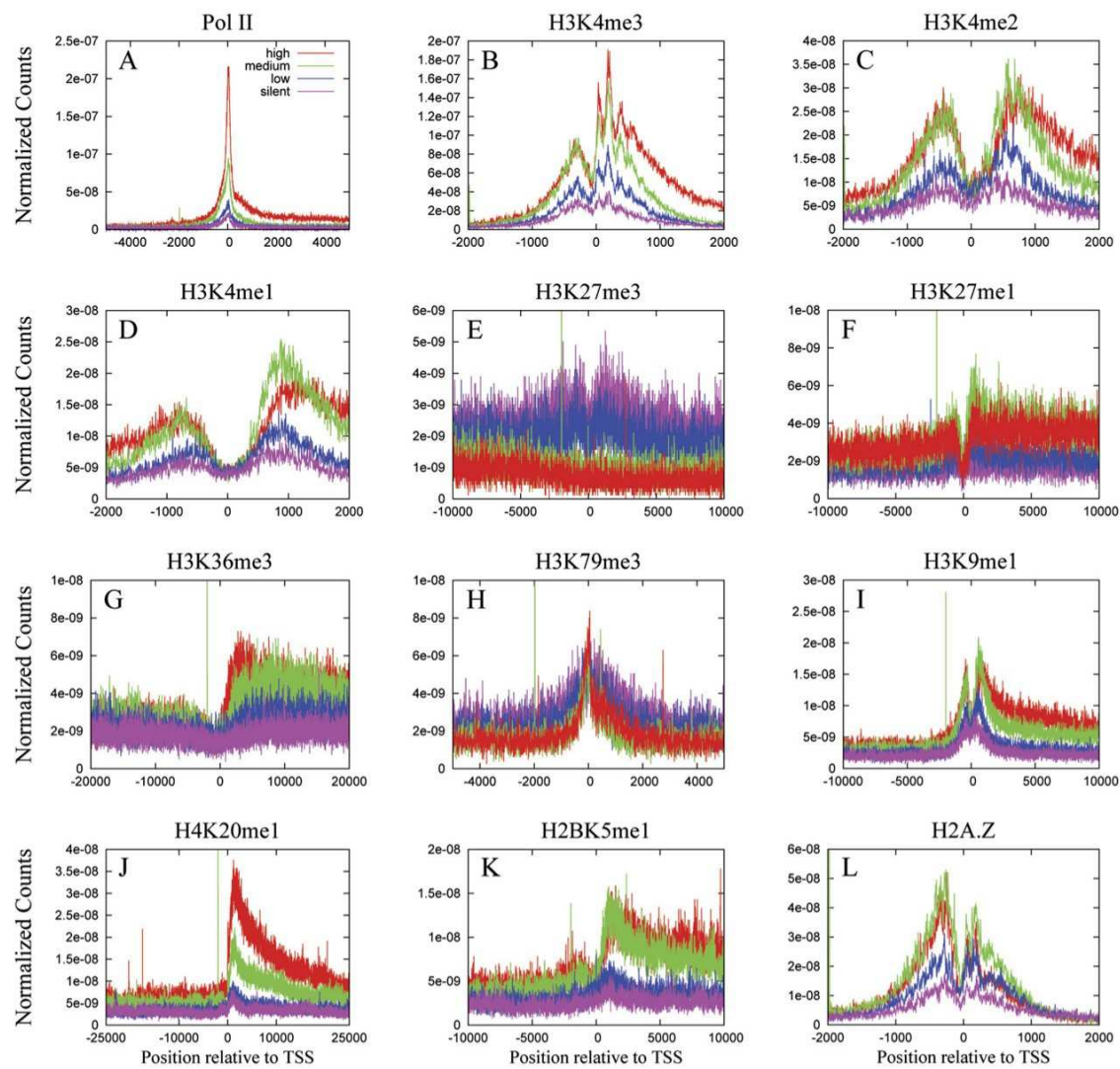


## Histónové modifikácie

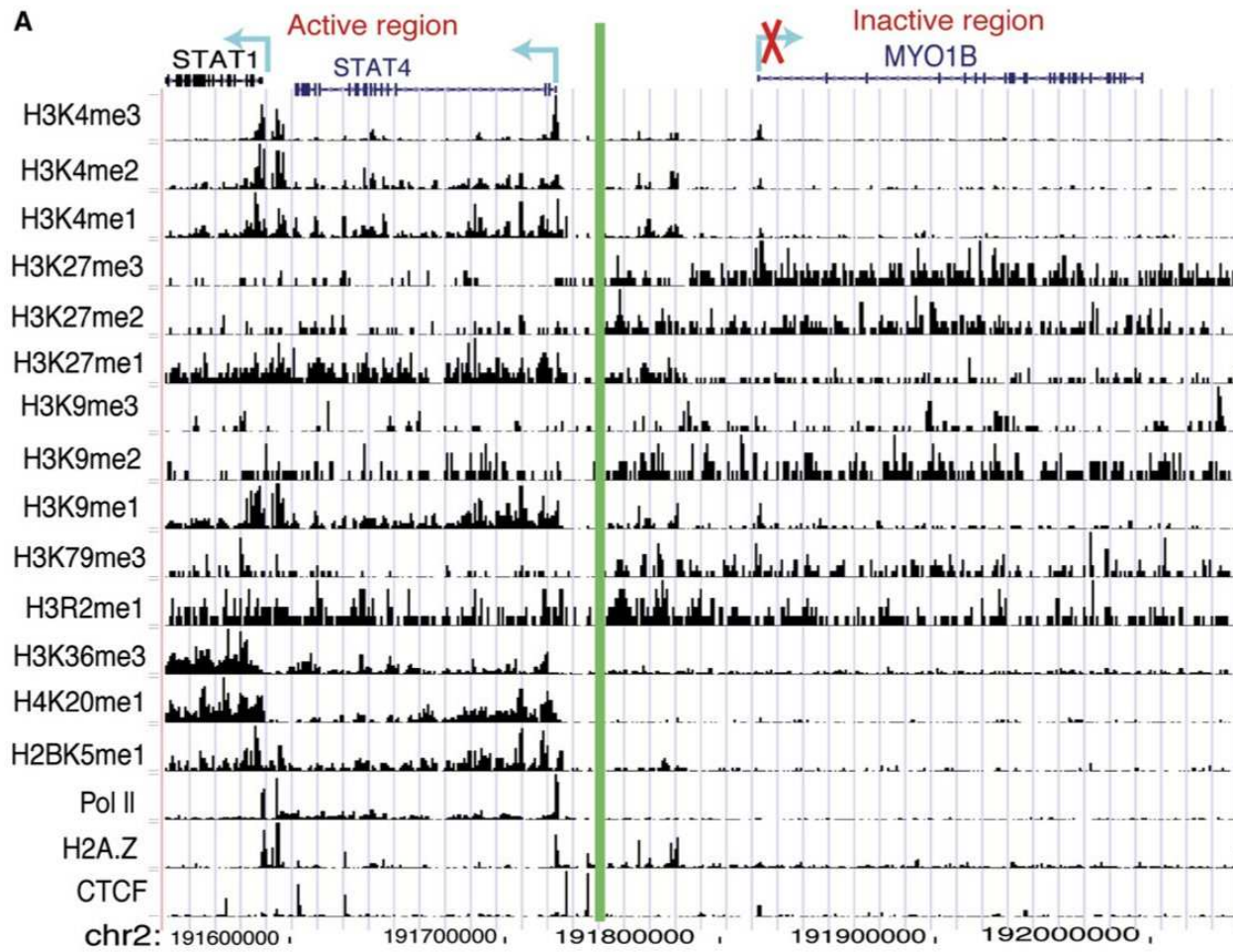
- Posttranslačné modifikácie, napr. metylácia
- Označenie napr. H3K4me1 je (mono-)metylácia štvrtej amino kyseliny (lyzínu) v proteíne H3

## Zisťovanie v genóme

- Enzýmom nasekáme DNA medzi nukleozómami
- Nukleozómy s danou modifikáciou extrahujeme pomocou protilátky (chromatin immunoprecipitation, ChIP)
- Extrahovanú DNA identifikujeme pomocou microarray alebo sekvenovaním a mapovaním na genóm (ChIP-chip alebo ChIP-seq)



Priemerné profily okolo začiatku transkripcie, Barski et al 2007

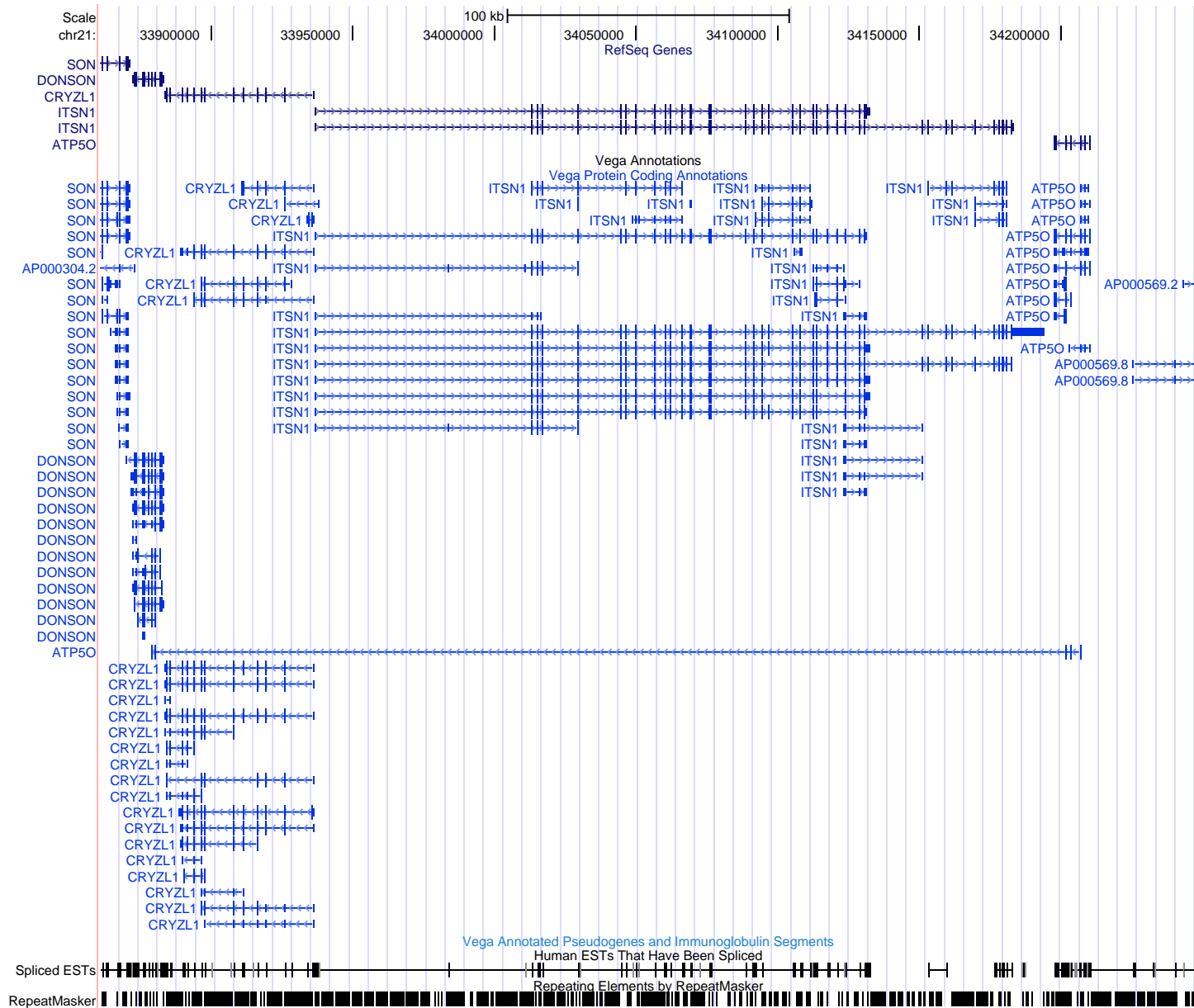


Konkrétne gény, Barski et al, The Cell, 2007

## ENCODE

- ENCODE pilot: podrobne preskúmať 1% ľudského genómu 2007
- ENCODE 2. fáza: rozšírenie na celý genóm, 2012
- Zistili, že väčšina báz ľudského genómu transkribovaná
- Veľa transkriptov nekódujúcich proteíny, prekrývajúce sa transkripty
- Veľa začiatkov transkripcie, alternatívnych transkriptov

**Príklad:** Kúsok ENCODE regiónu, chr21:33,860,819-34,247,408  
transkript spájajúci DONSON a ATP50



# K-means clustering

**Tomáš Vinař**

**16.11.2017**



## Formulácia problému

**Vstup:**  $n$ -rozmerné vektory  $x_1, x_2, \dots, x_t$  a počet zhlukov  $k$

**Výstup:** Rozdelenie vektorov do  $k$  zhlukov:

- priradenie vstupných vektorov do zhlukov zapísané ako čísla  $c_1, c_2, \dots, c_t$ , kde  $c_i \in \{1, 2, \dots, k\}$  je číslo zhľuku pre  $x_i$
- centrum každého zhľuku, t.j.  $n$ -rozmerné vektory  $\mu_1, \mu_2, \dots, \mu_k$

Hodnoty  $c_1, \dots, c_t$  a  $\mu_1, \dots, \mu_k$  volíme tak, aby sme minimalizovali súčet štvorcov vzdialeností od každého vektoru k centru jeho zhľuku:

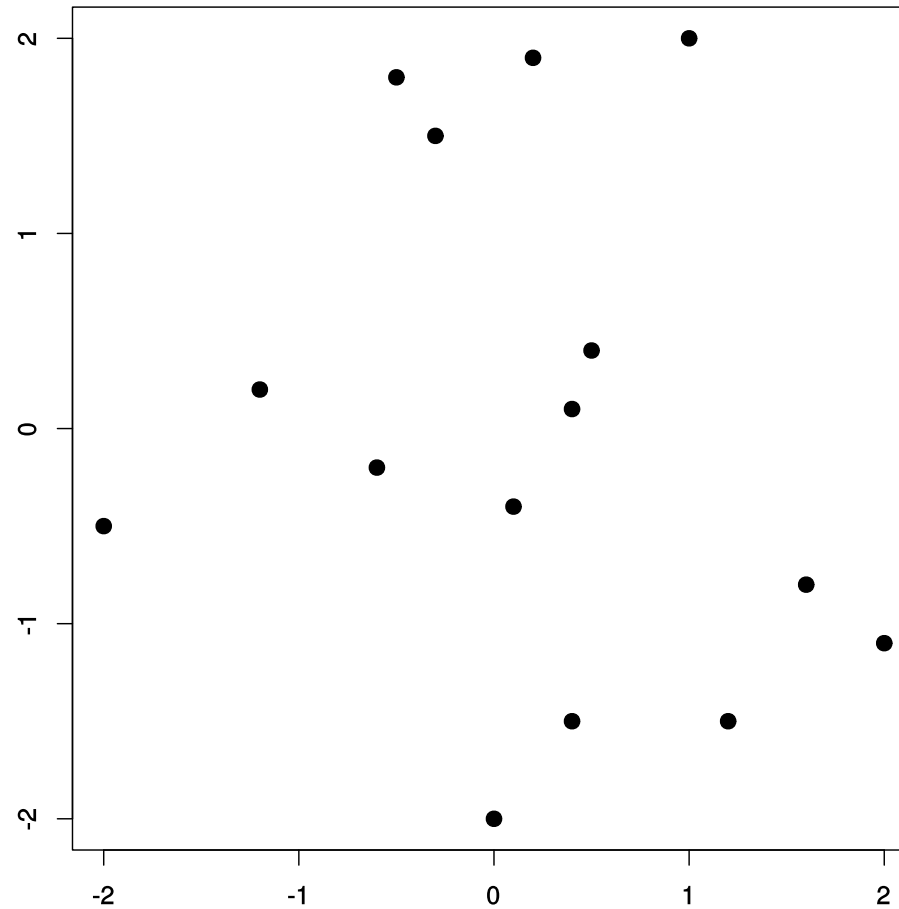
$$\sum_{i=1}^t \|x_i - \mu_{c_i}\|_2^2$$

Pre vektory  $a = (a_1, \dots, a_n)$  a  $b = (b_1, \dots, b_n)$  je druhá mocnina vzdialenosti  $\|a - b\|_2^2 = \sum_{i=1}^n (a_i - b_i)^2$

## Príklad vstupu

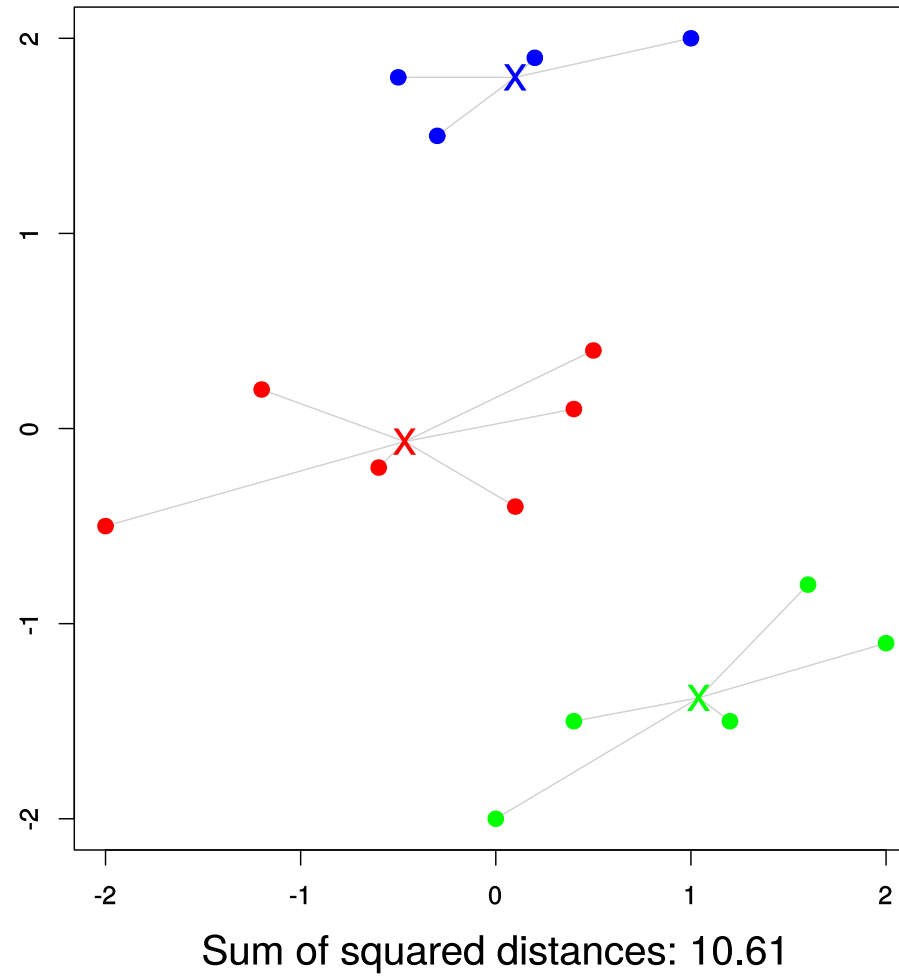
$x_1$	-2.00	-0.50
$x_2$	-1.20	0.20
$x_3$	-0.60	-0.20
$x_4$	-0.50	1.80
$x_5$	-0.30	1.50
$x_6$	0.00	-2.00
$x_7$	0.10	-0.40
$x_8$	0.20	1.90
$x_9$	0.40	0.10
$x_{10}$	0.40	-1.50
$x_{11}$	0.50	0.40
$x_{12}$	1.00	2.00
$x_{13}$	1.20	-1.50
$x_{14}$	1.60	-0.80
$x_{15}$	2.00	-1.10

$$k = 3$$



## Príklad výstupu

$x_1$	-2.00	-0.50	<b>1</b>
$x_2$	-1.20	0.20	<b>1</b>
$x_3$	-0.60	-0.20	<b>1</b>
$x_4$	-0.50	1.80	<b>3</b>
$x_5$	-0.30	1.50	<b>3</b>
$x_6$	0.00	-2.00	<b>2</b>
$x_7$	0.10	-0.40	<b>1</b>
$x_8$	0.20	1.90	<b>3</b>
$x_9$	0.40	0.10	<b>1</b>
$x_{10}$	0.40	-1.50	<b>2</b>
$x_{11}$	0.50	0.40	<b>1</b>
$x_{12}$	1.00	2.00	<b>3</b>
$x_{13}$	1.20	-1.50	<b>2</b>
$x_{14}$	1.60	-0.80	<b>2</b>
$x_{15}$	2.00	-1.10	<b>2</b>
$\mu_1$	<b>-0.47</b>	<b>-0.07</b>	
$\mu_2$	<b>1.04</b>	<b>-1.38</b>	
$\mu_3$	<b>0.10</b>	<b>1.80</b>	



## Algoritmus

Heuristika, ktorá nenájde vždy najlepšie zhľukovanie.  
Začne z nejakého zhľukovania a postupne ho zlepšuje.

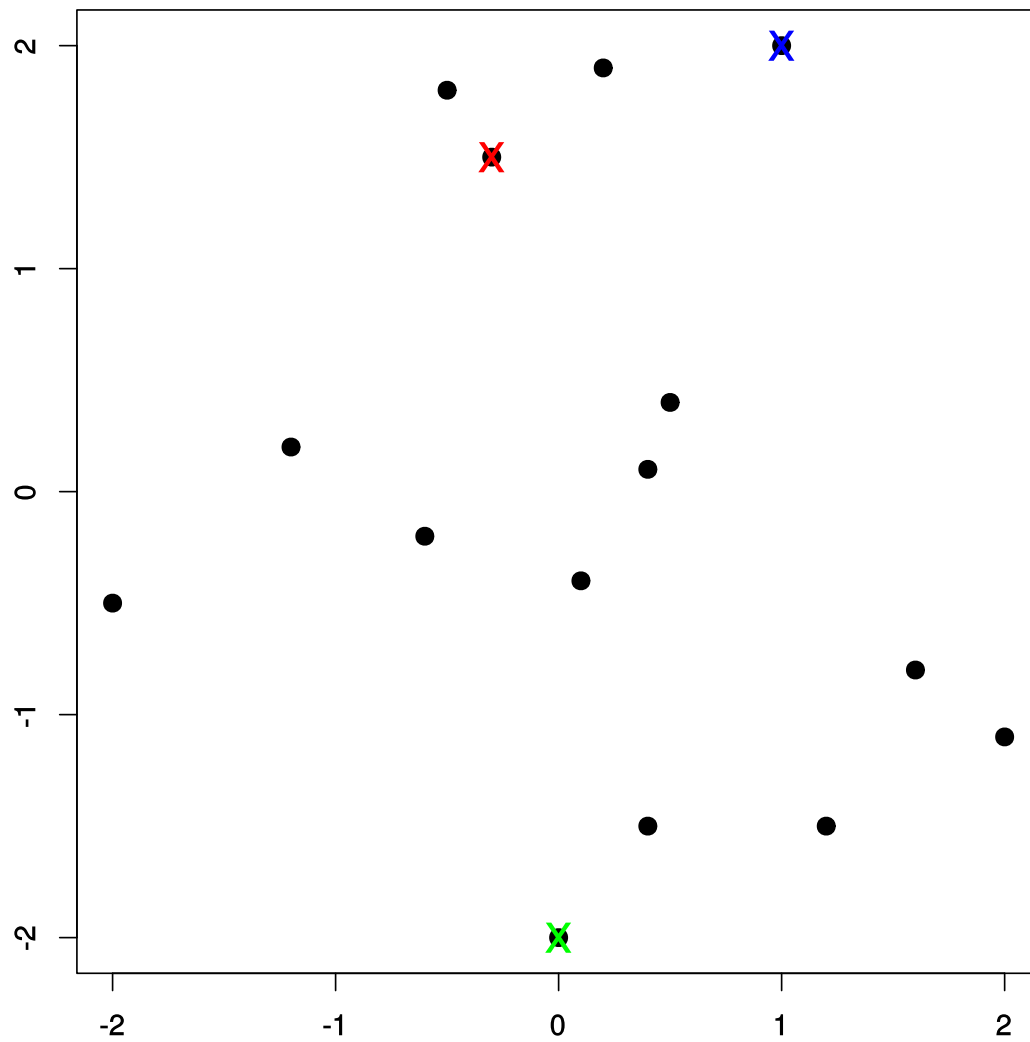
### Inicializácia:

náhodne vyber  $k$  centier  $\mu_1, \mu_2, \dots, \mu_k$  spomedzi vstupných vektorov

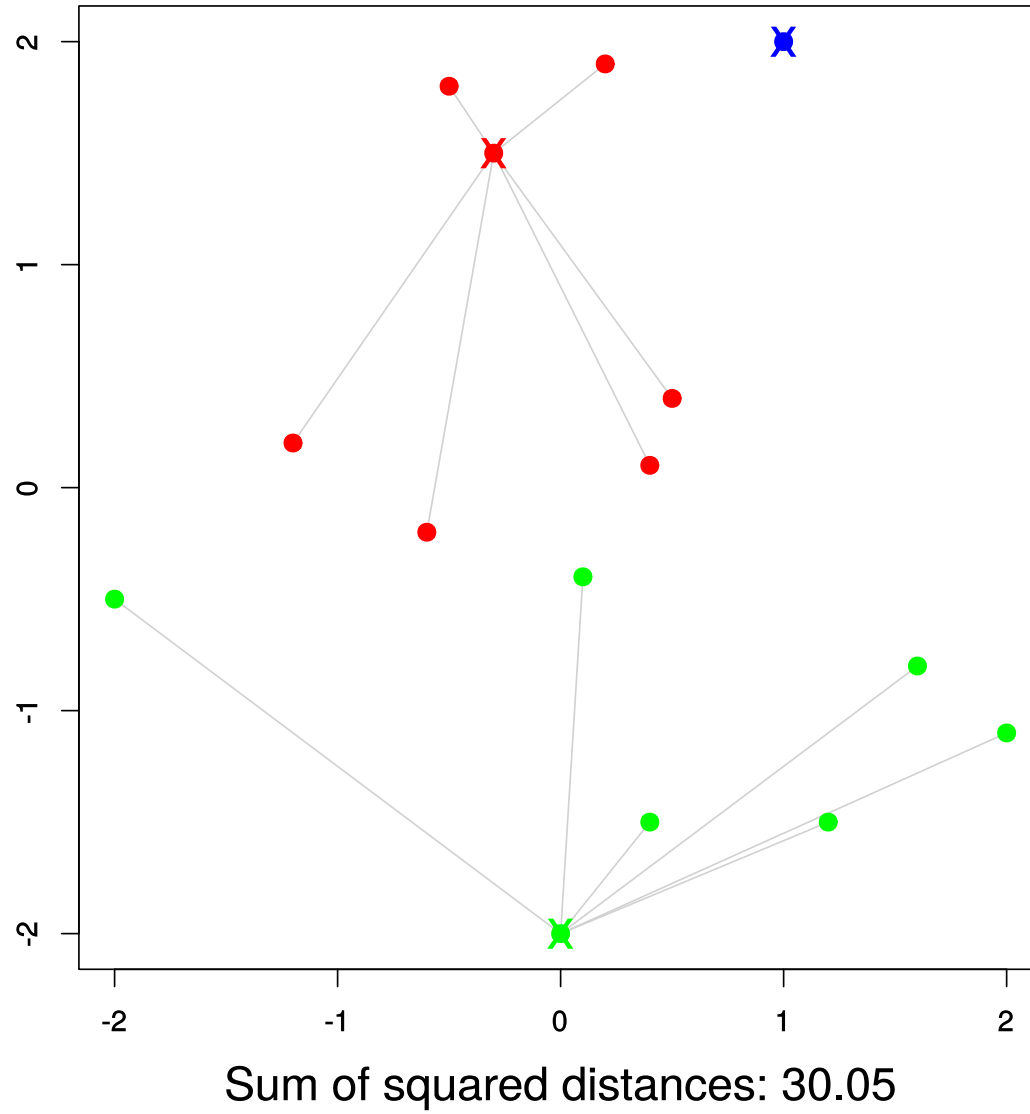
### Opakuj, kým sa niečo mení:

- priradiť každý bod najbližšiemu centru:  $c_i = \arg \min_j \|x_i - \mu_j\|_2$
- vypočítaj nové centroidy:  $\mu_j$  bude priemerom (po zložkách) z vektorov  $x_i$ , pre ktoré  $c_i = j$

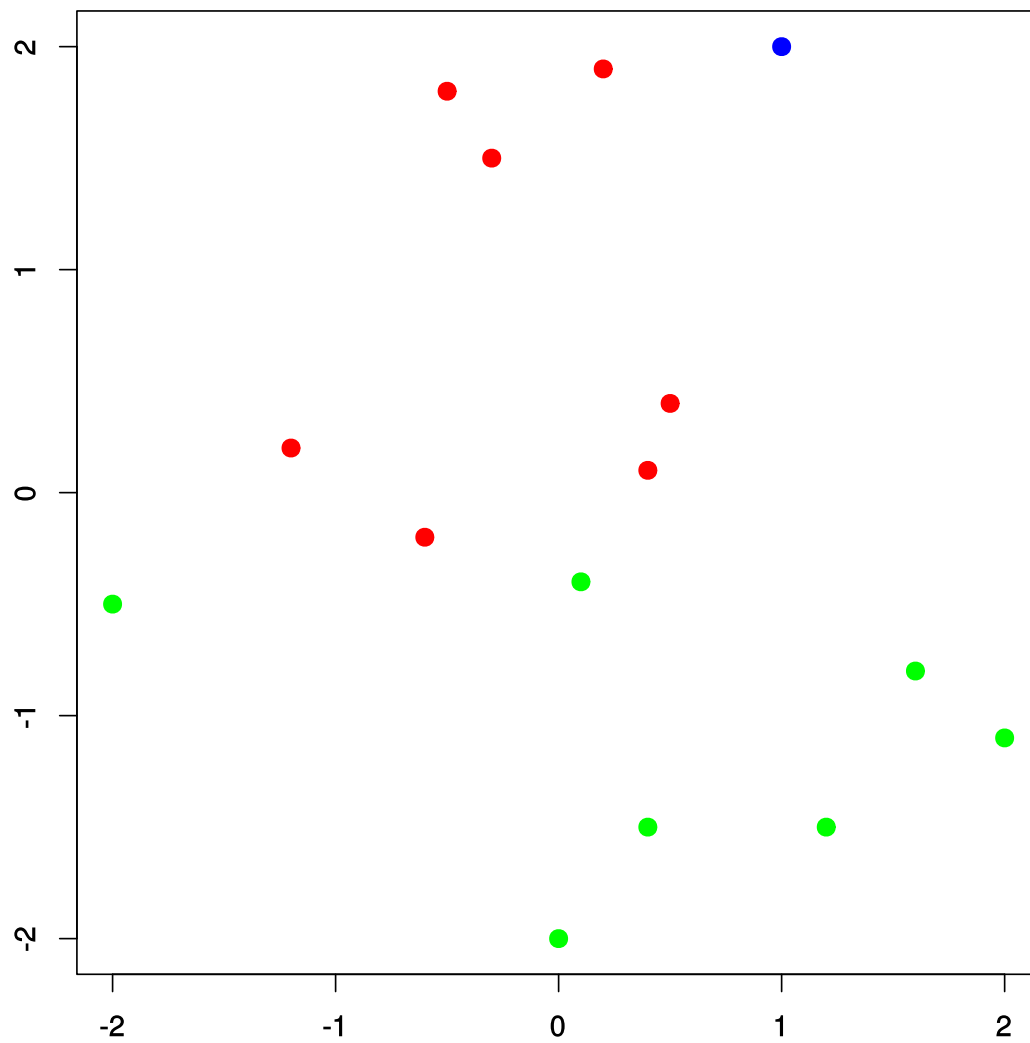
Zvolíme náhodné centrá  $\mu_i$



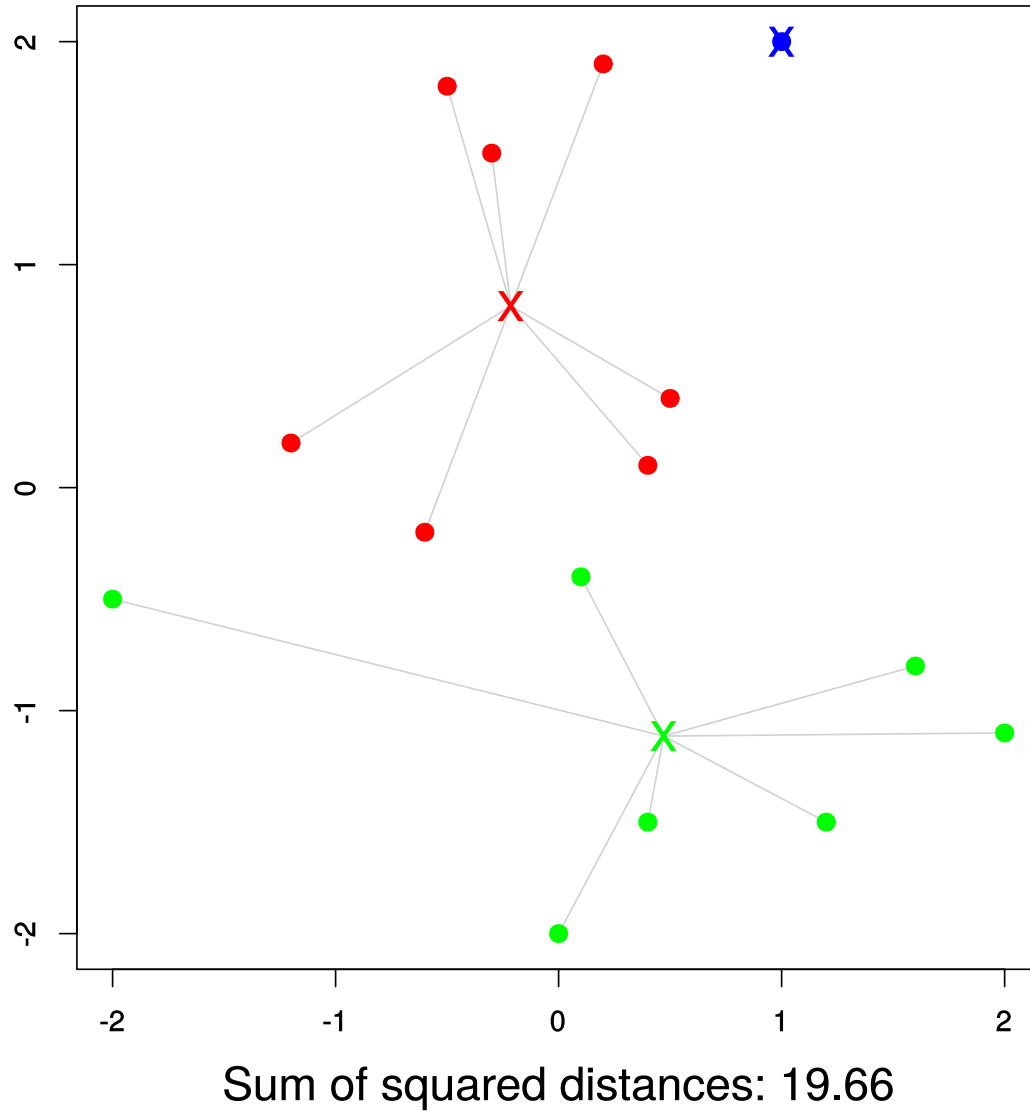
# Vektory priradíme do zhlukov (hodnoty $c_i$ )



Zabudneme  $\mu_i$

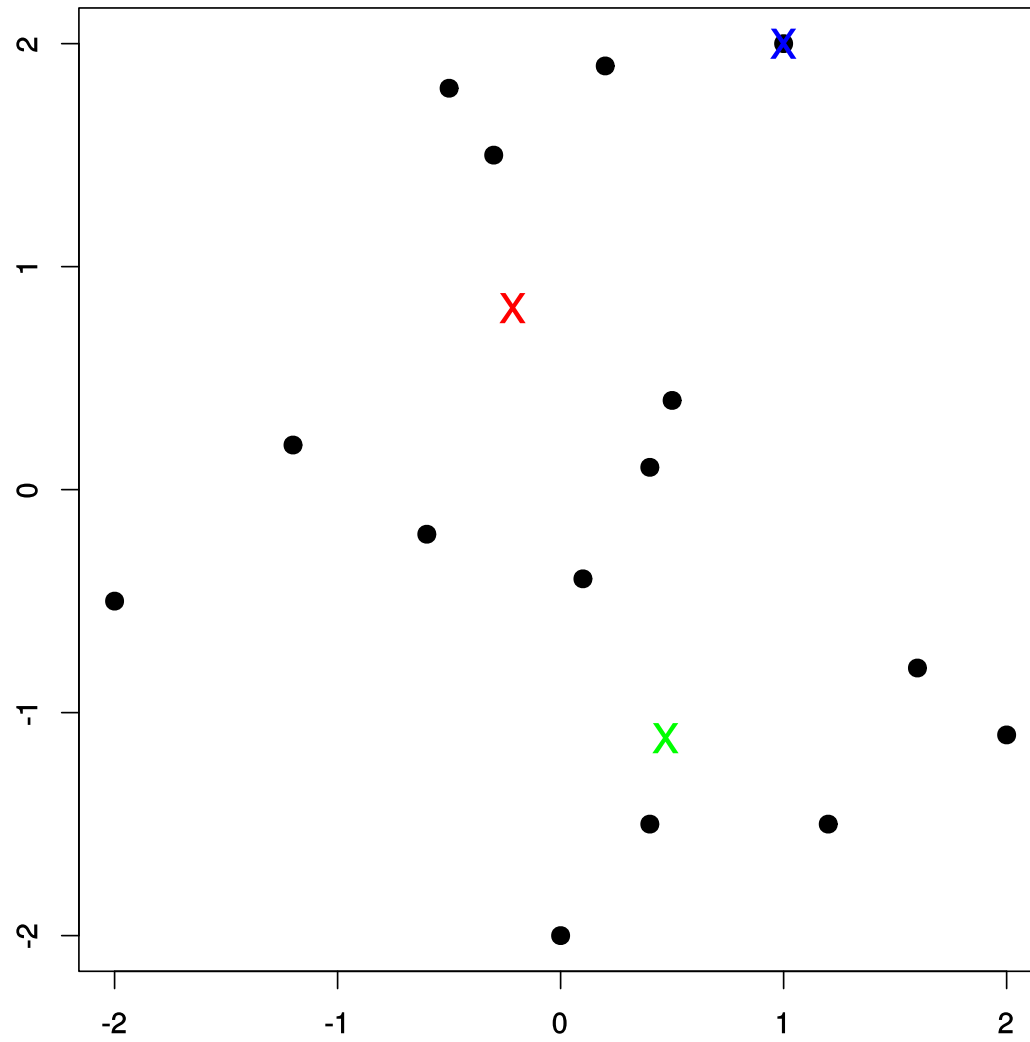


Dopocítame nové  $\mu_i$  (suma klesla z 30.05 na 19.66)

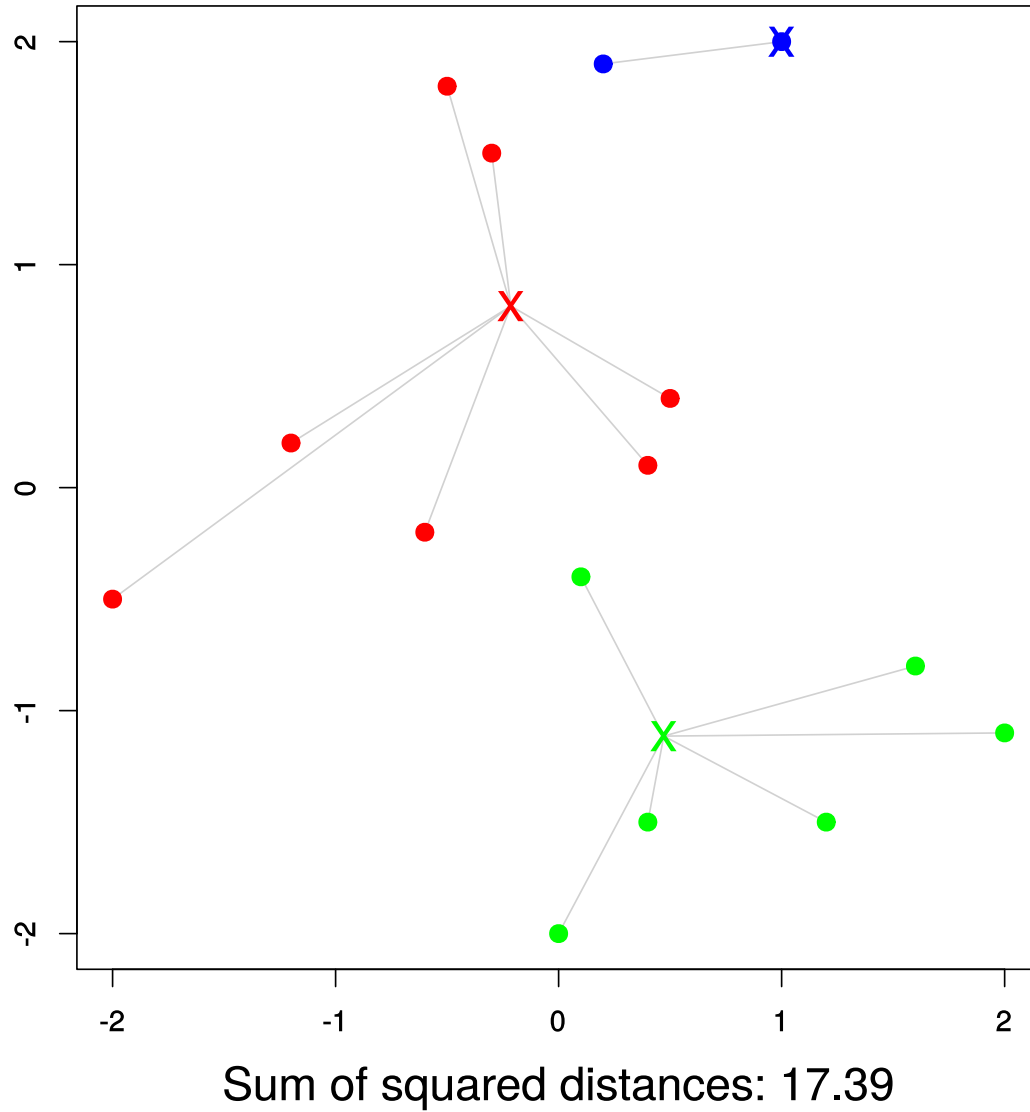




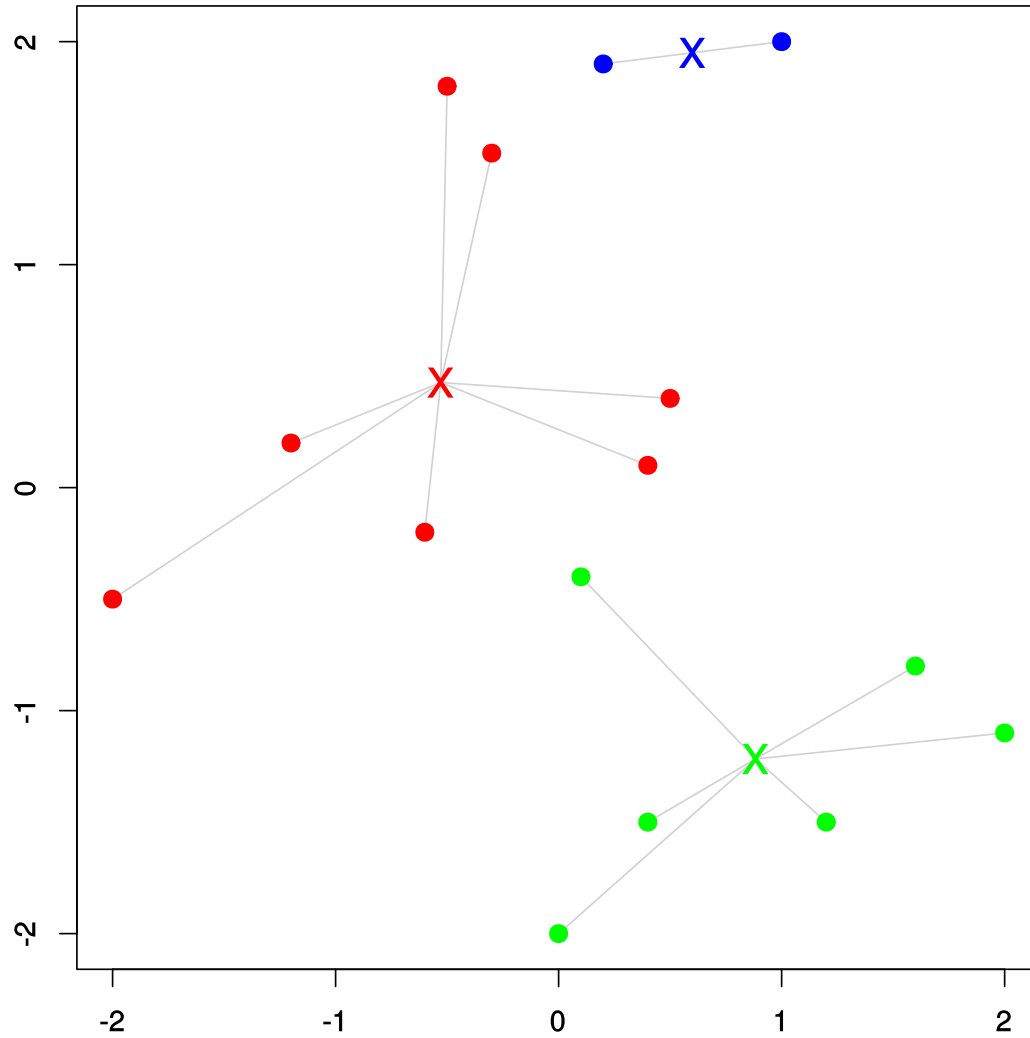
Zabudneme  $c_i$



Dopocítame nové  $c_i$  (suma klesla z 19.66 na 17.39)

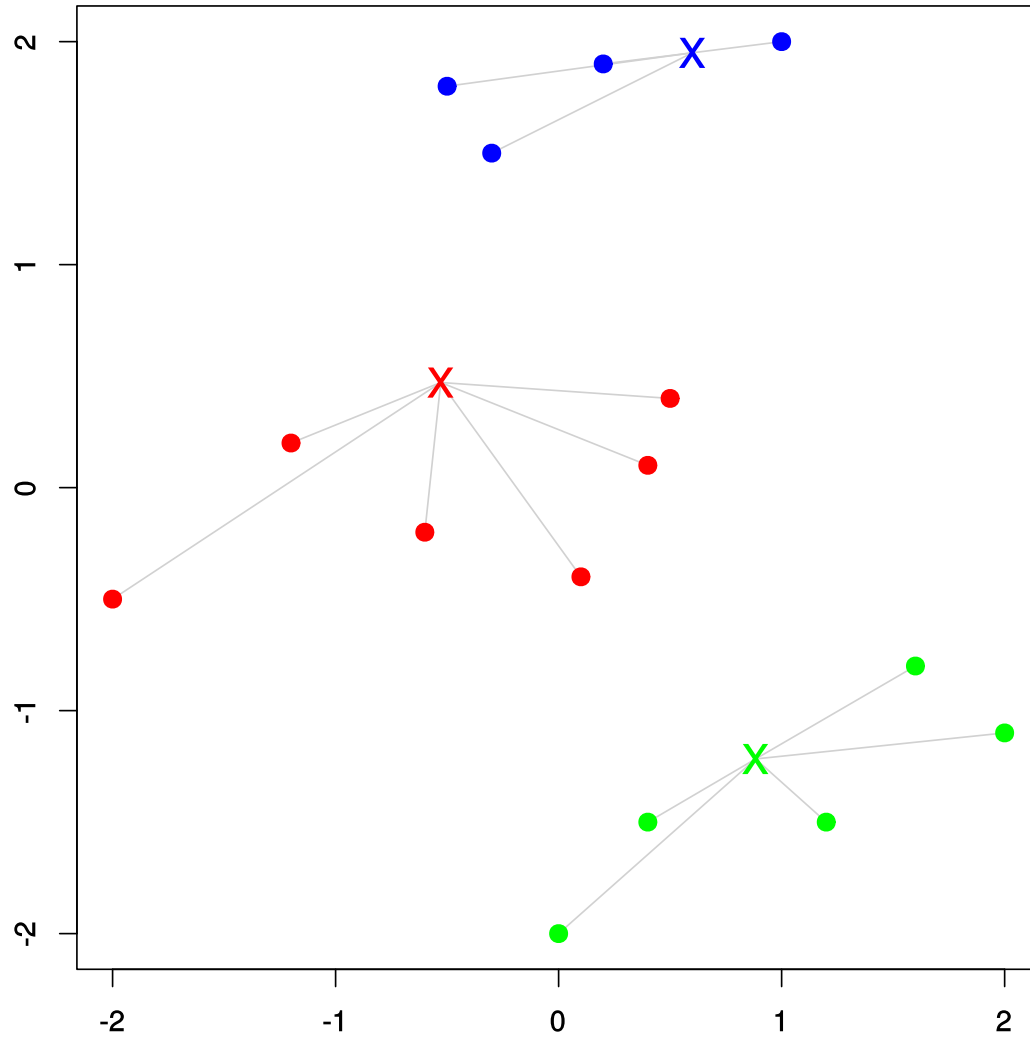


Prepočítame  $\mu_i$



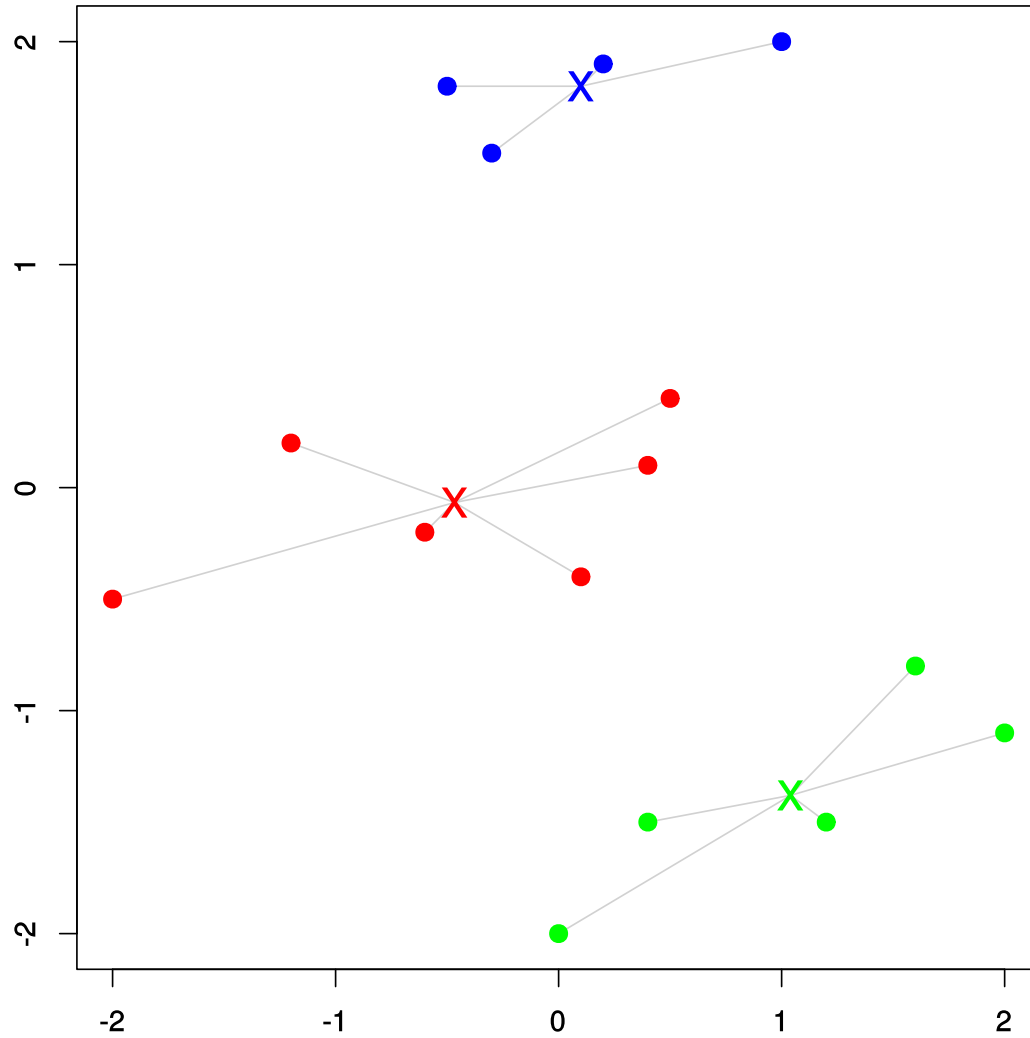
Sum of squared distances: 14.47

Prepočítame  $c_i$

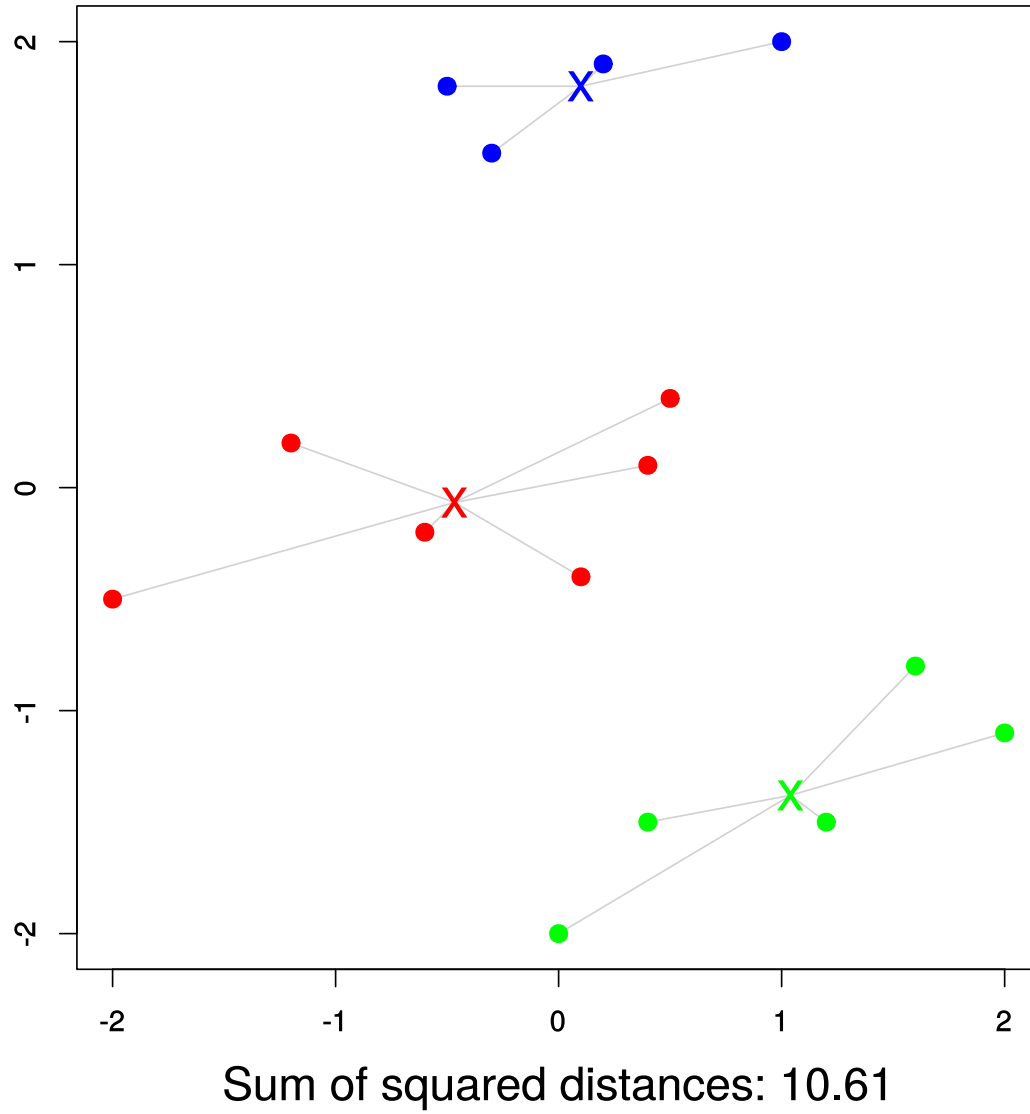


Sum of squared distances: 13.71

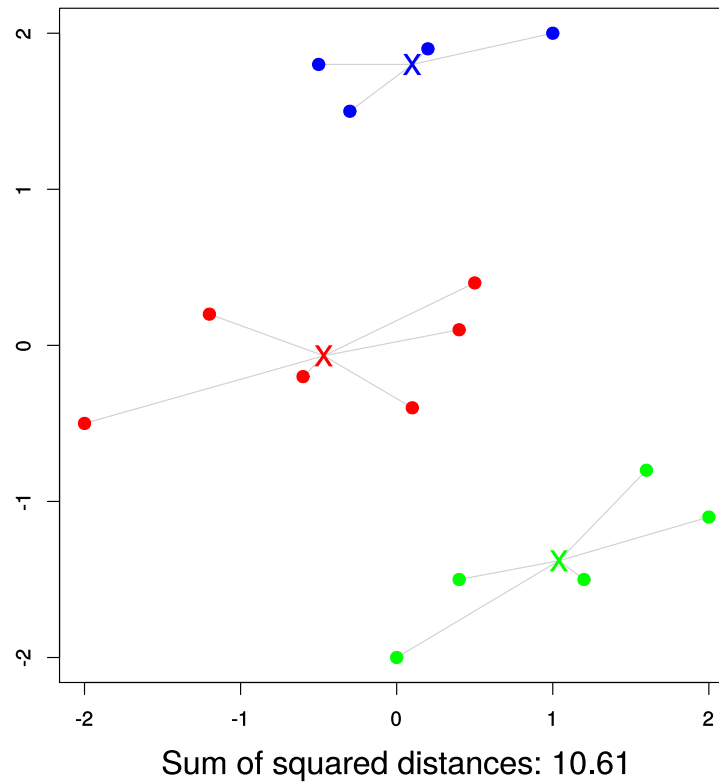
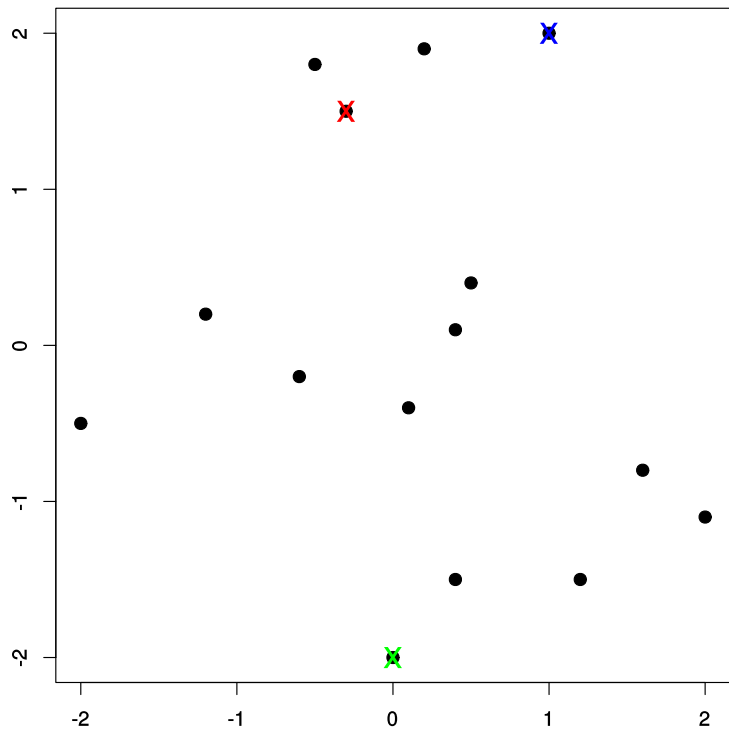
Prepočítame  $\mu_i$



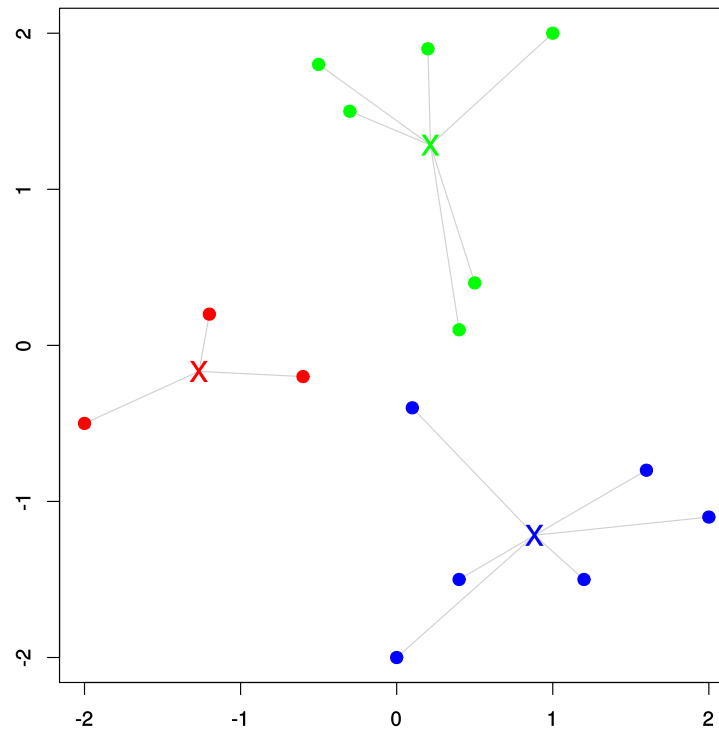
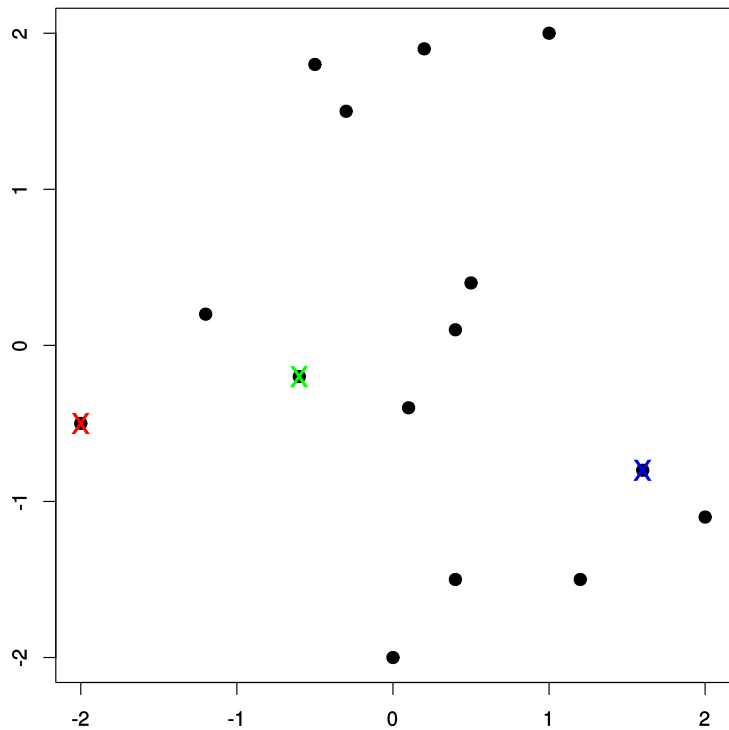
Prepočítame  $c_i$  (žiadna zmena, končíme)



# Príklady niekoľkých behov programu



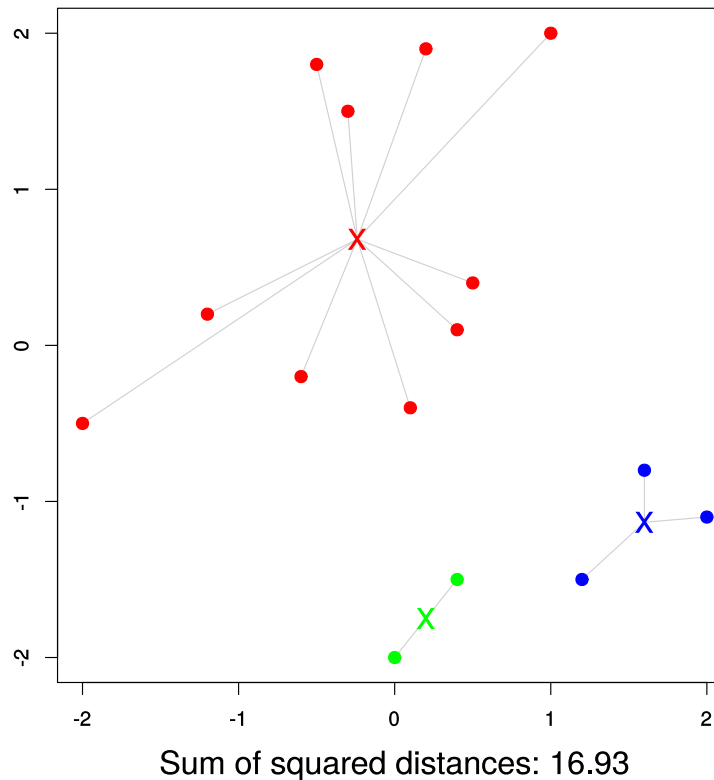
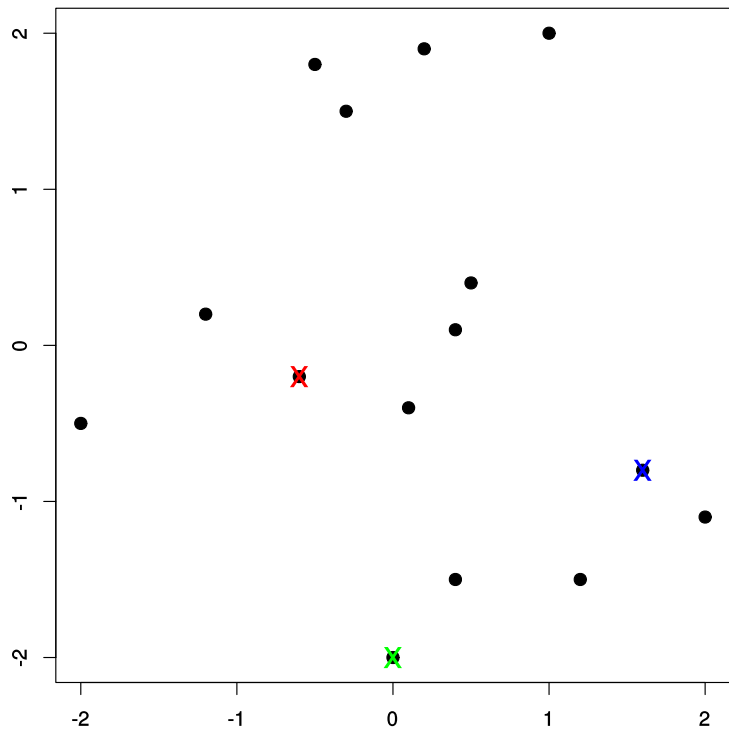
# Príklady niekoľkých behov programu



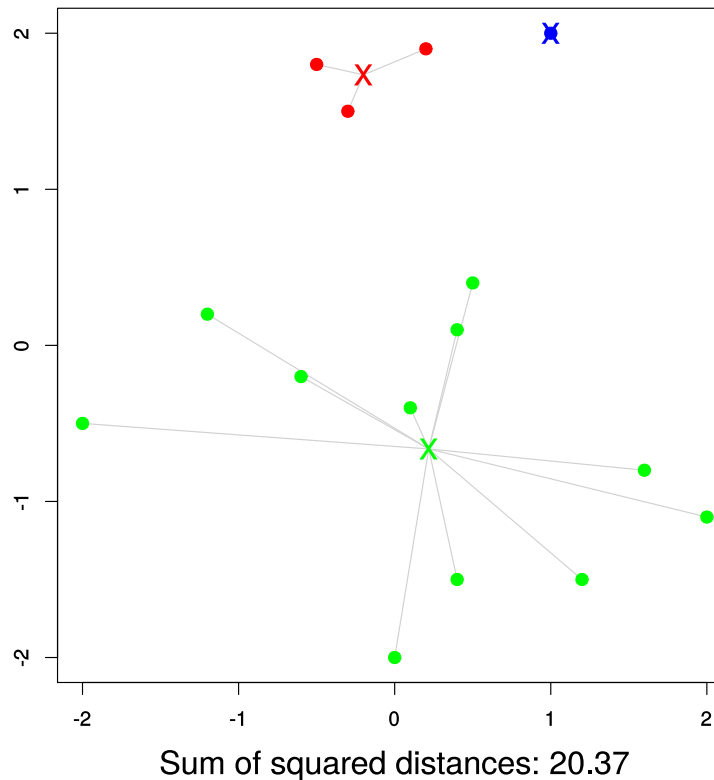
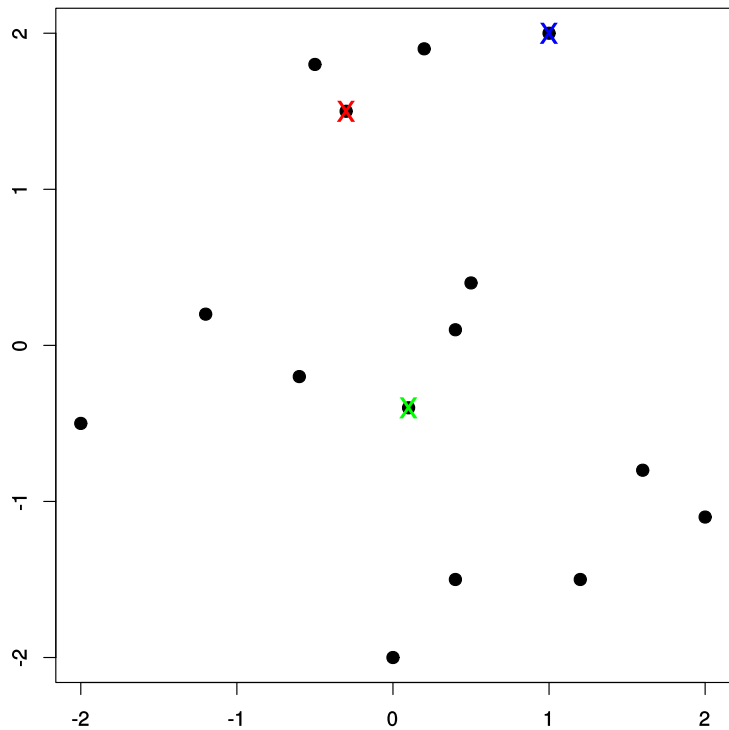
Sum of squared distances: 11.25



# Príklady niekoľkých behov programu



# Príklady niekoľkých behov programu



# K-means clustering

**Tomáš Vinař**

**16.11.2017**

## Formulácia problému

**Vstup:**  $n$ -rozmerné vektory  $x_1, x_2, \dots, x_t$  a počet zhlukov  $k$

**Výstup:** Rozdelenie vektorov do  $k$  zhlukov:

- priradenie vstupných vektorov do zhlukov zapísané ako čísla  $c_1, c_2, \dots, c_t$ , kde  $c_i \in \{1, 2, \dots, k\}$  je číslo zhľuku pre  $x_i$
- centrum každého zhľuku, t.j.  $n$ -rozmerné vektory  $\mu_1, \mu_2, \dots, \mu_k$

Hodnoty  $c_1, \dots, c_t$  a  $\mu_1, \dots, \mu_k$  volíme tak, aby sme minimalizovali súčet štvorcov vzdialeností od každého vektoru k centru jeho zhľuku:

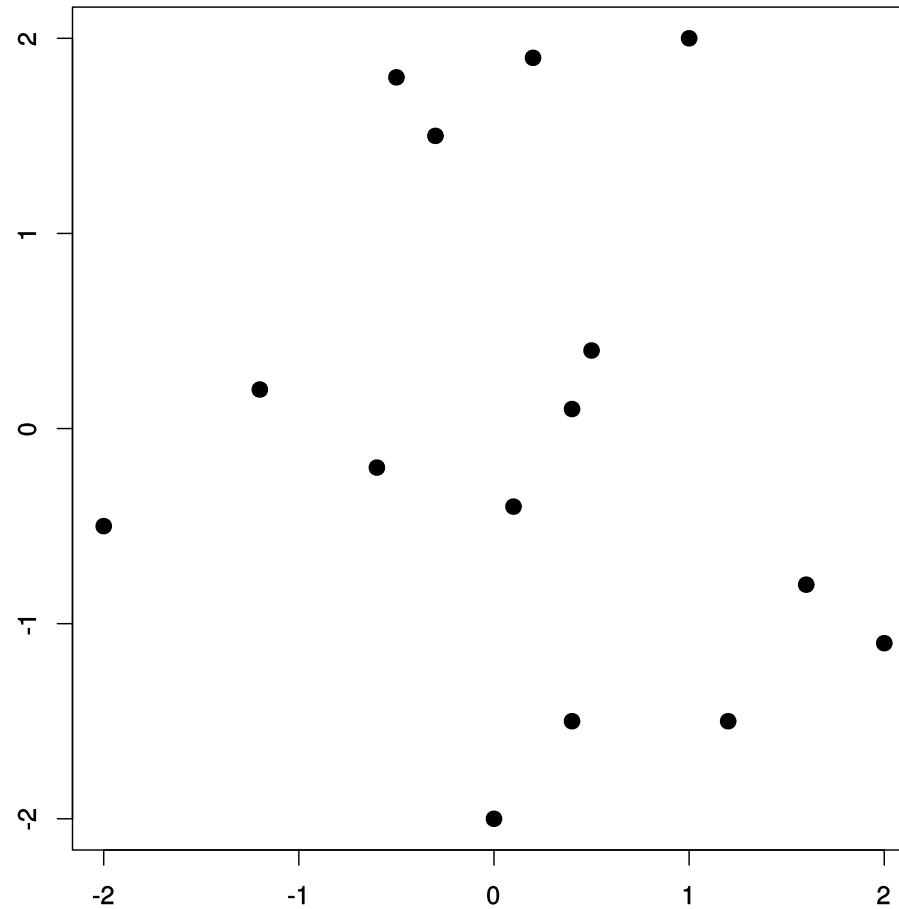
$$\sum_{i=1}^t \|x_i - \mu_{c_i}\|_2^2$$

Pre vektory  $a = (a_1, \dots, a_n)$  a  $b = (b_1, \dots, b_n)$  je druhá mocnina vzdialenosti  $\|a - b\|_2^2 = \sum_{i=1}^n (a_i - b_i)^2$

## Príklad vstupu

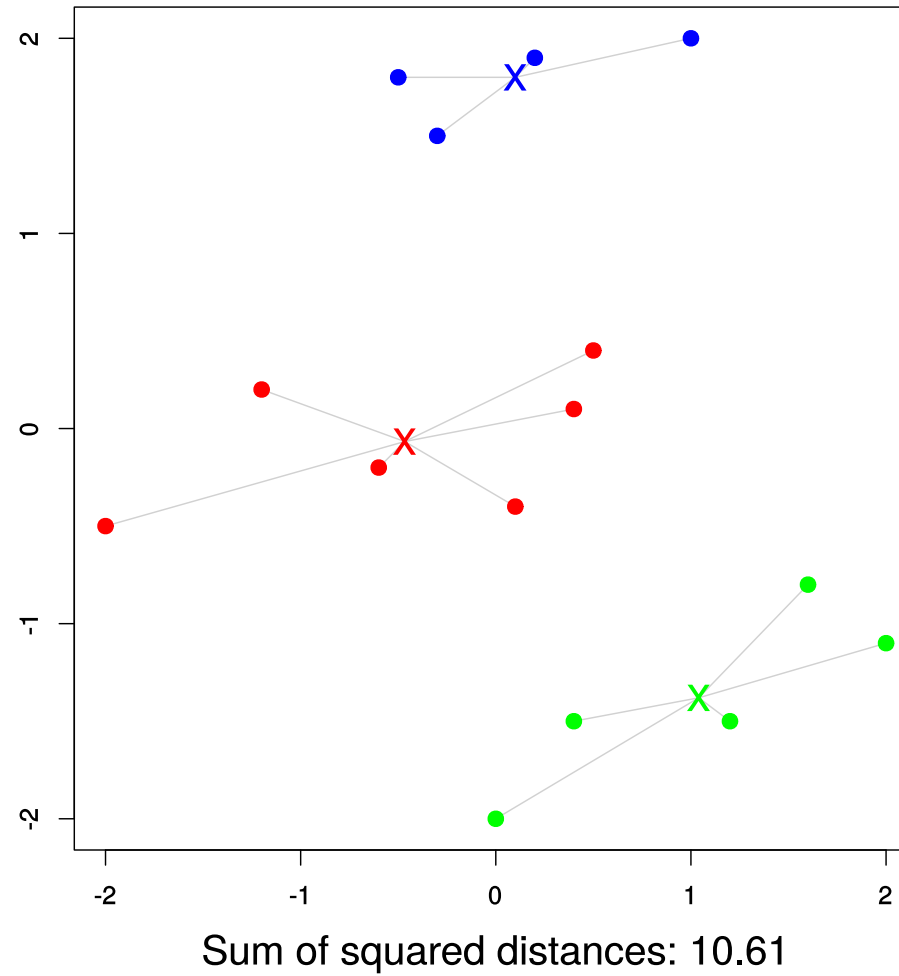
$x_1$	-2.00	-0.50
$x_2$	-1.20	0.20
$x_3$	-0.60	-0.20
$x_4$	-0.50	1.80
$x_5$	-0.30	1.50
$x_6$	0.00	-2.00
$x_7$	0.10	-0.40
$x_8$	0.20	1.90
$x_9$	0.40	0.10
$x_{10}$	0.40	-1.50
$x_{11}$	0.50	0.40
$x_{12}$	1.00	2.00
$x_{13}$	1.20	-1.50
$x_{14}$	1.60	-0.80
$x_{15}$	2.00	-1.10

$$k = 3$$



## Príklad výstupu

$x_1$	-2.00	-0.50	<b>1</b>
$x_2$	-1.20	0.20	<b>1</b>
$x_3$	-0.60	-0.20	<b>1</b>
$x_4$	-0.50	1.80	<b>3</b>
$x_5$	-0.30	1.50	<b>3</b>
$x_6$	0.00	-2.00	<b>2</b>
$x_7$	0.10	-0.40	<b>1</b>
$x_8$	0.20	1.90	<b>3</b>
$x_9$	0.40	0.10	<b>1</b>
$x_{10}$	0.40	-1.50	<b>2</b>
$x_{11}$	0.50	0.40	<b>1</b>
$x_{12}$	1.00	2.00	<b>3</b>
$x_{13}$	1.20	-1.50	<b>2</b>
$x_{14}$	1.60	-0.80	<b>2</b>
$x_{15}$	2.00	-1.10	<b>2</b>
$\mu_1$	<b>-0.47</b>	<b>-0.07</b>	
$\mu_2$	<b>1.04</b>	<b>-1.38</b>	
$\mu_3$	<b>0.10</b>	<b>1.80</b>	



## Algoritmus

Heuristika, ktorá nenájde vždy najlepšie zhlukovanie.  
Začne z nejakého zhlukovania a postupne ho zlepšuje.

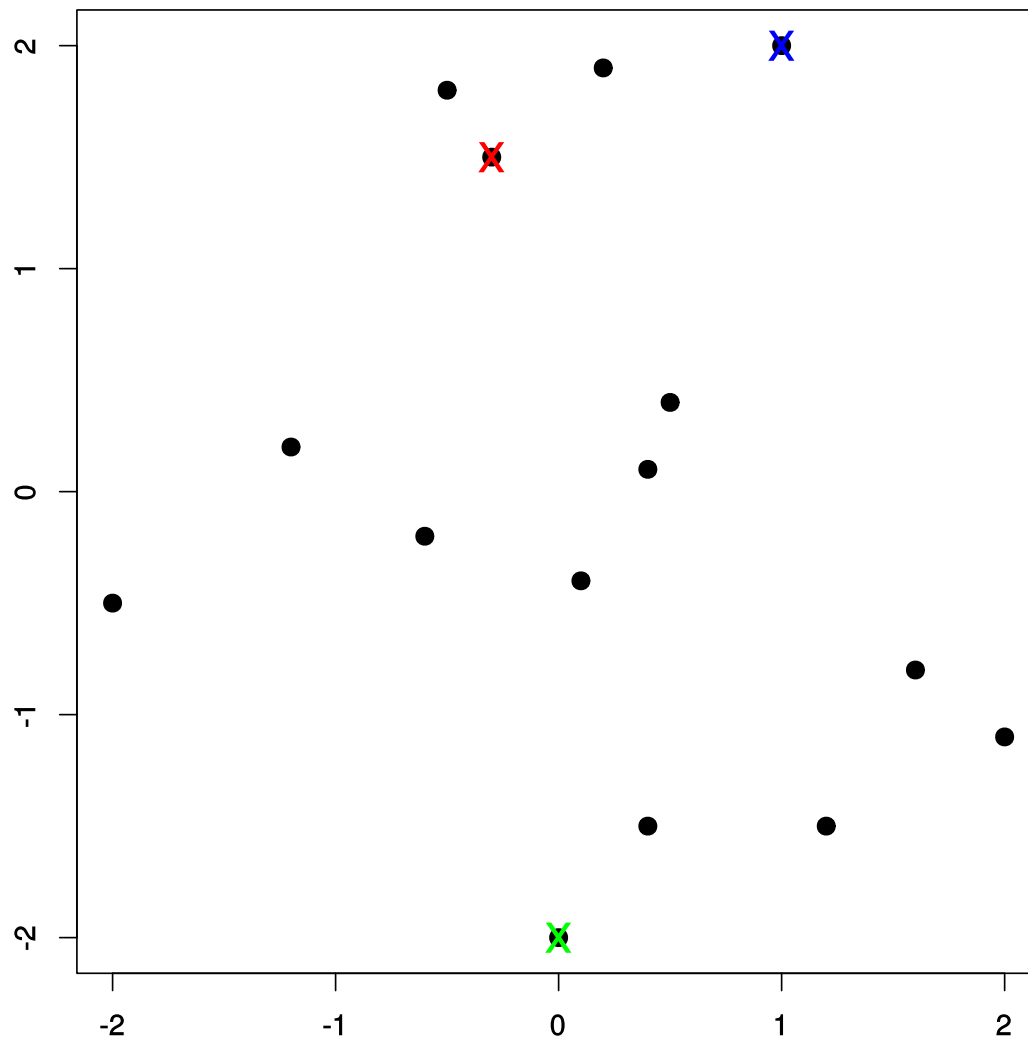
### Inicializácia:

náhodne vyber  $k$  centier  $\mu_1, \mu_2, \dots, \mu_k$  spomedzi vstupných vektorov

### Opakuj, kým sa niečo mení:

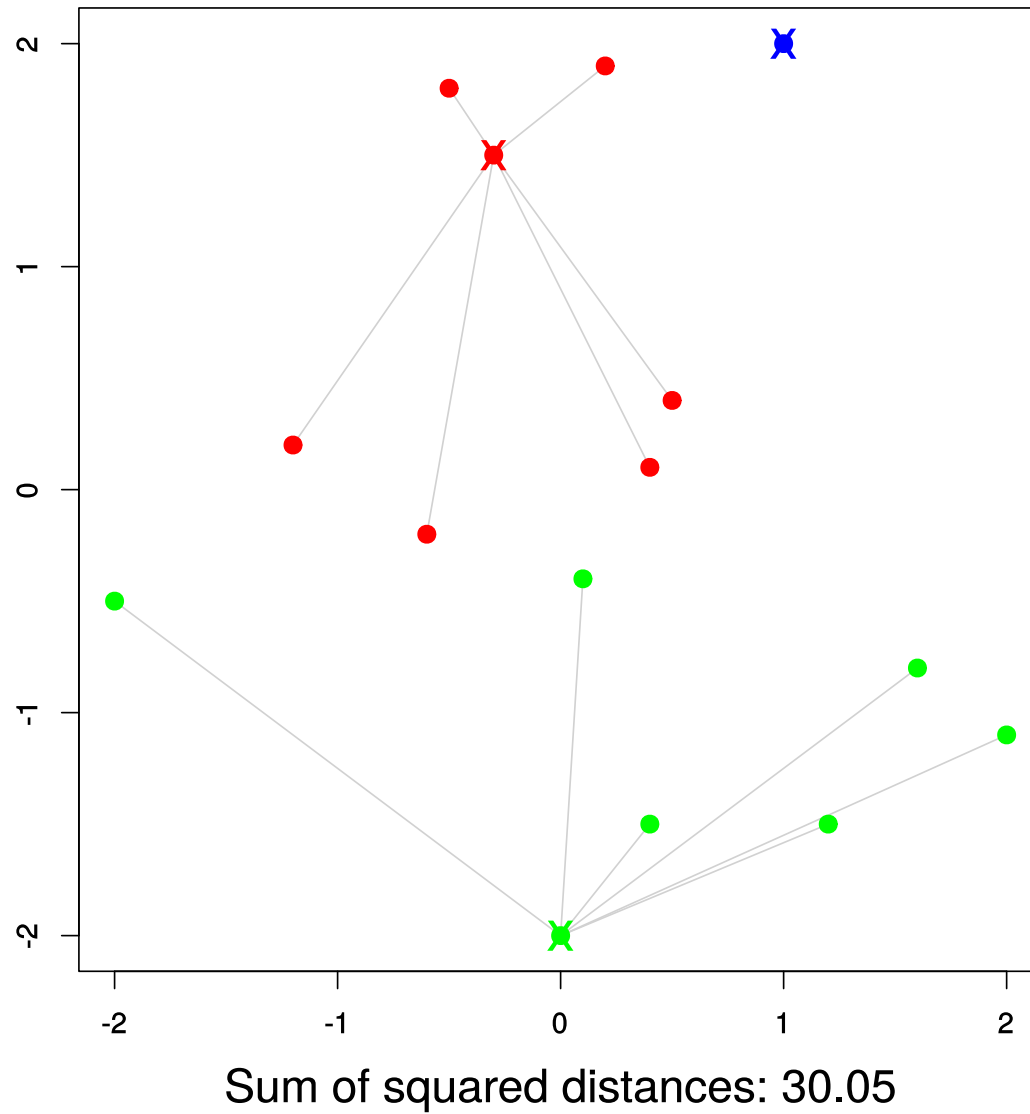
- priradiť každý bod najbližšiemu centru:  $c_i = \arg \min_j \|x_i - \mu_j\|_2$
- vypočítaj nové centroidy:  $\mu_j$  bude priemerom (po zložkách) z vektorov  $x_i$ , pre ktoré  $c_i = j$

Zvolíme náhodné centrá  $\mu_i$

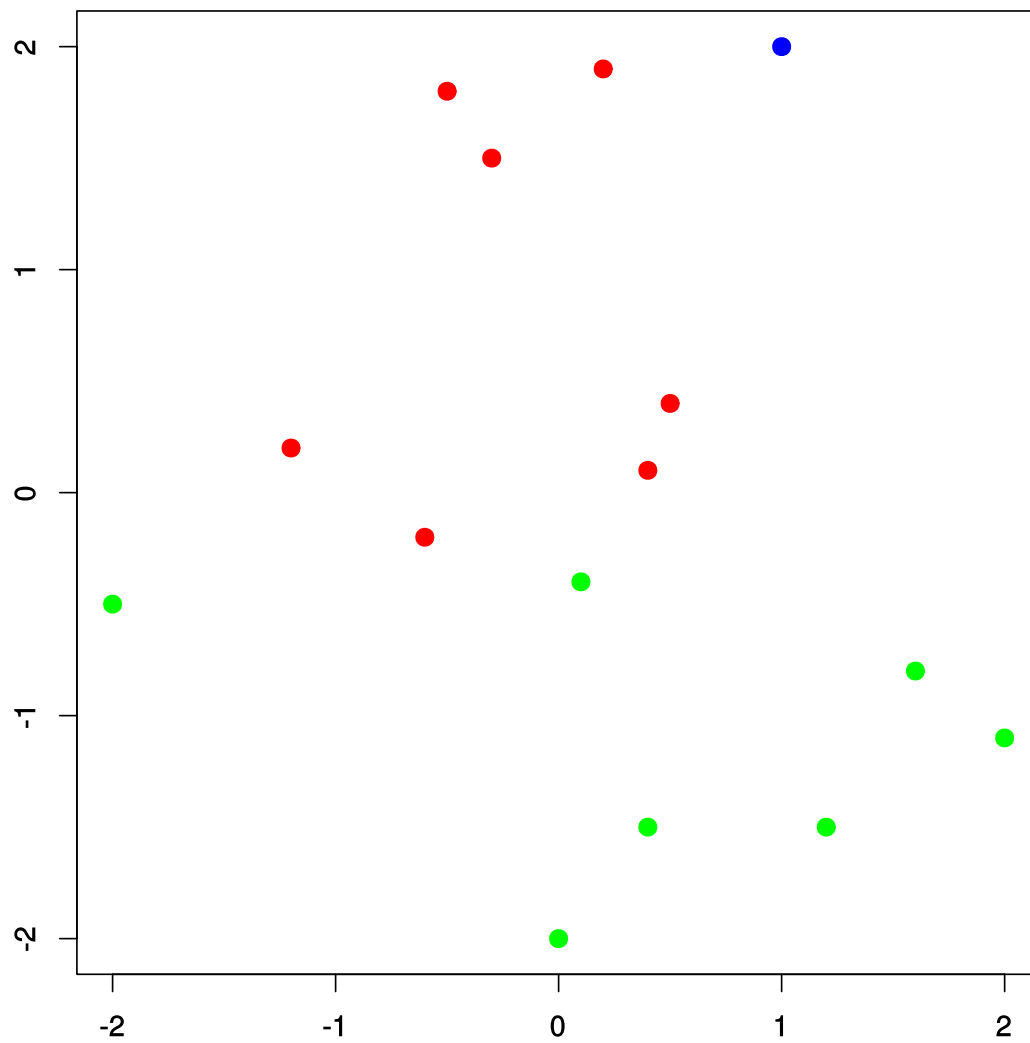




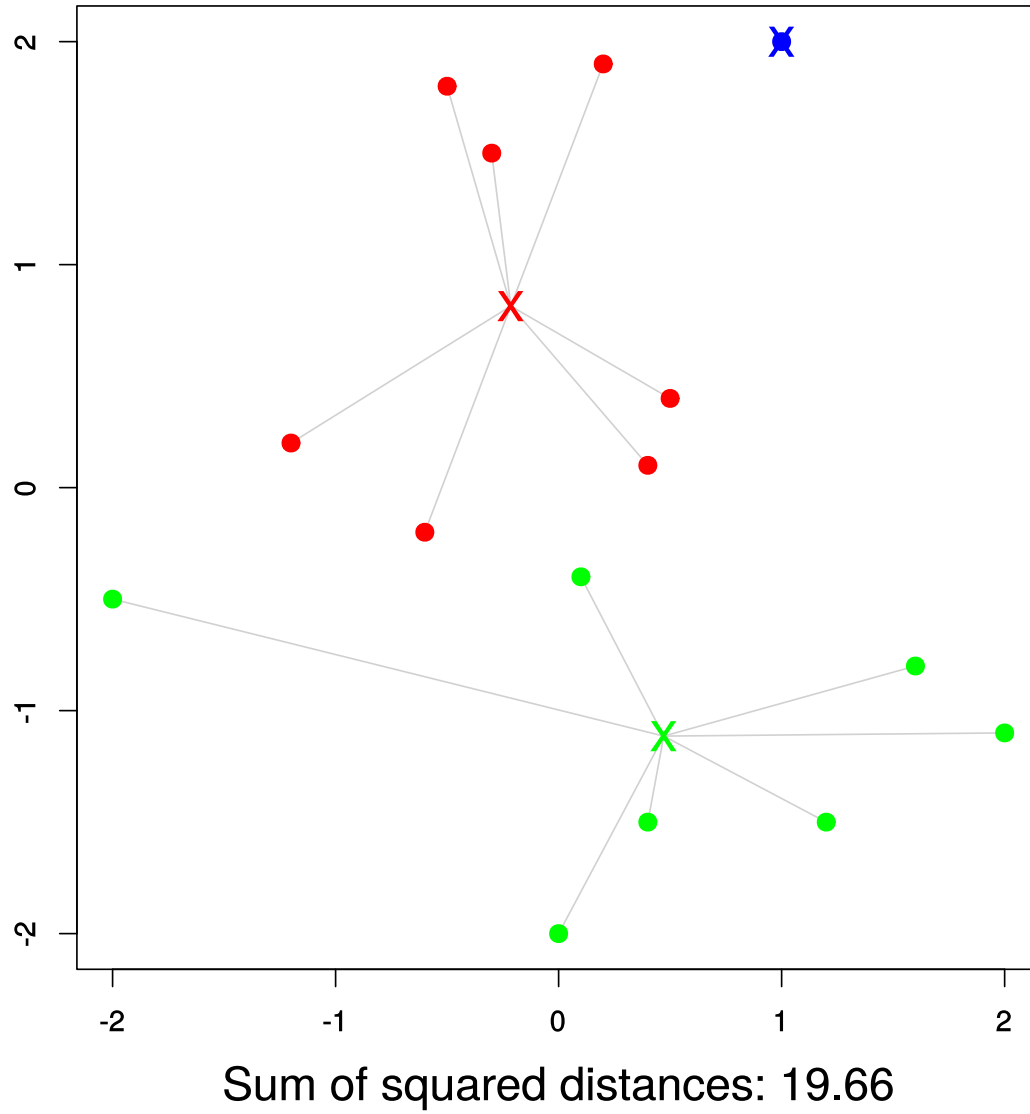
# Vektory priradíme do zhlukov (hodnoty $c_i$ )



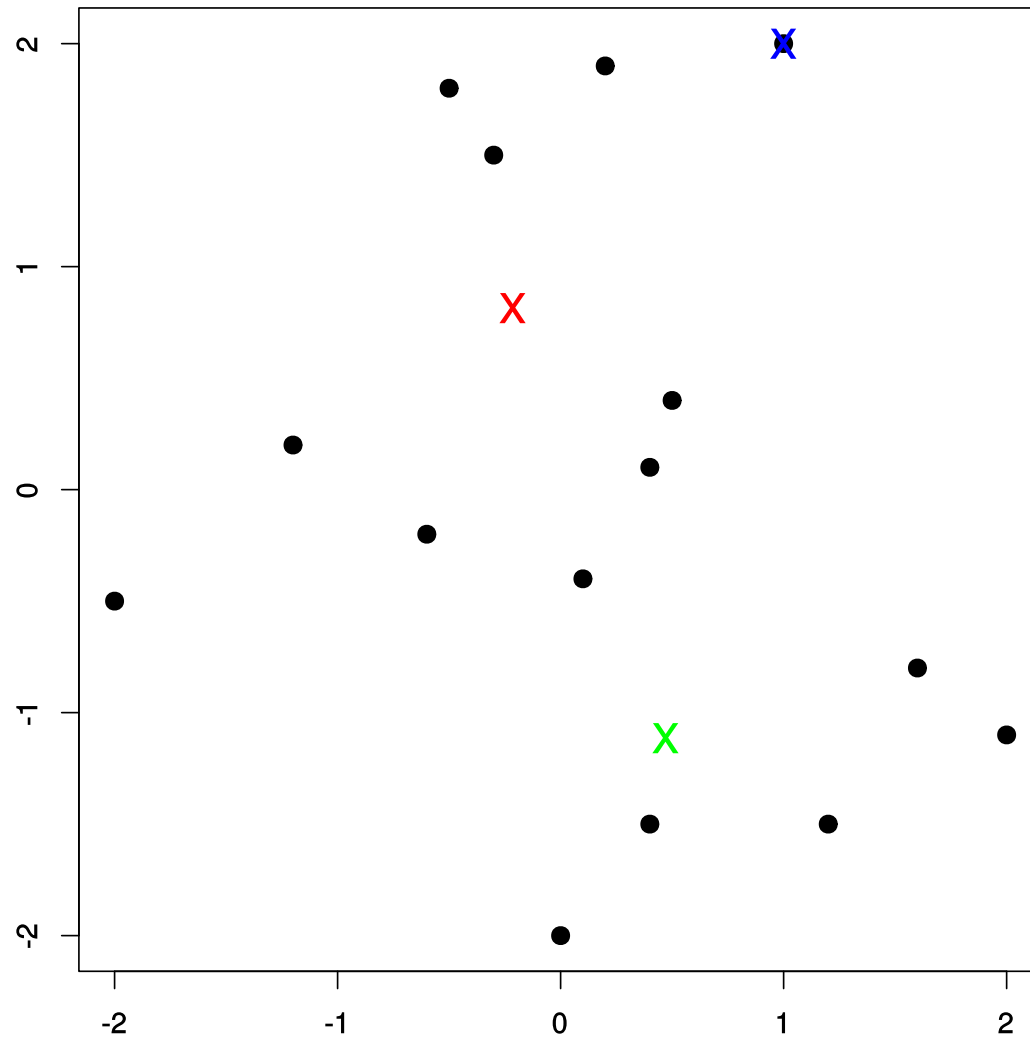
Zabudneme  $\mu_i$



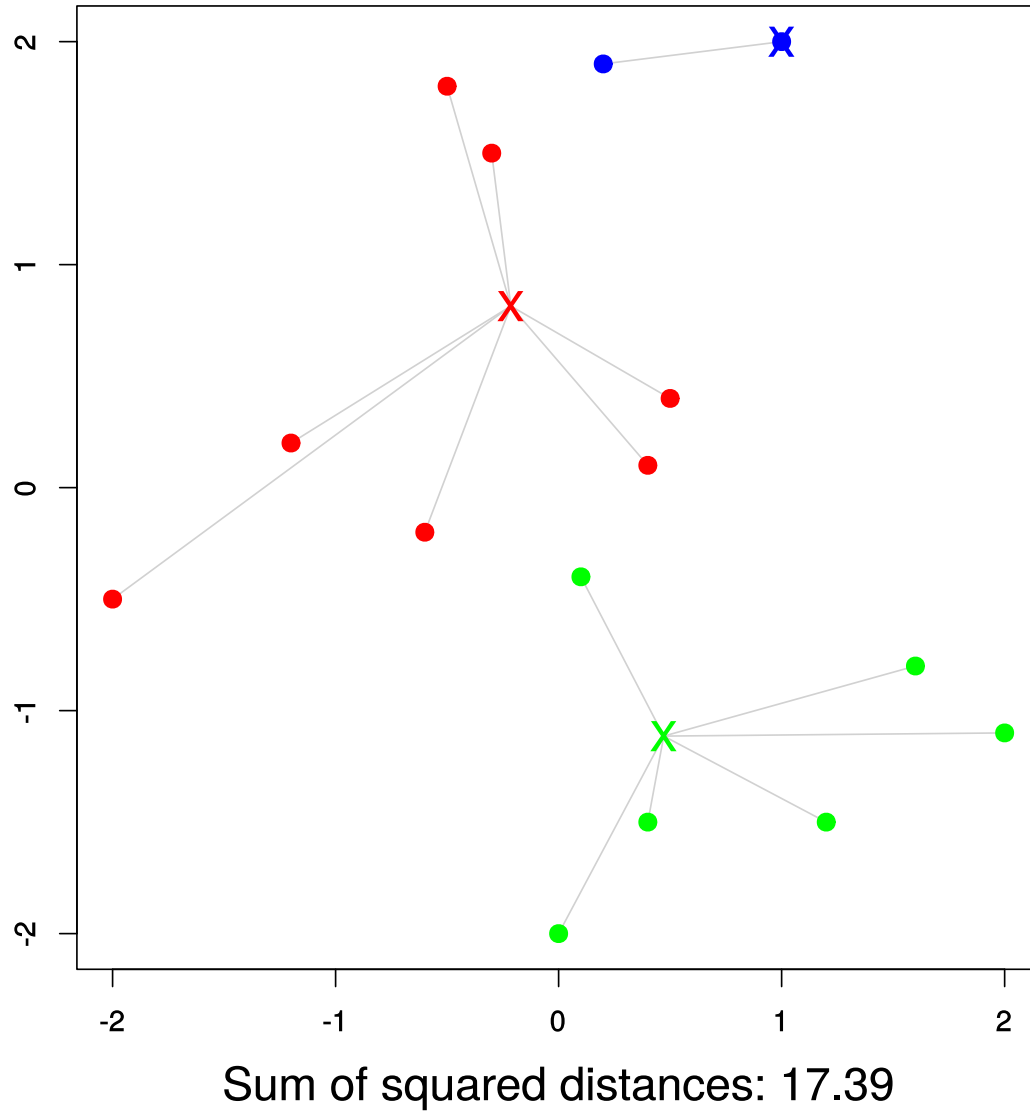
Dopocítame nové  $\mu_i$  (suma klesla z 30.05 na 19.66)



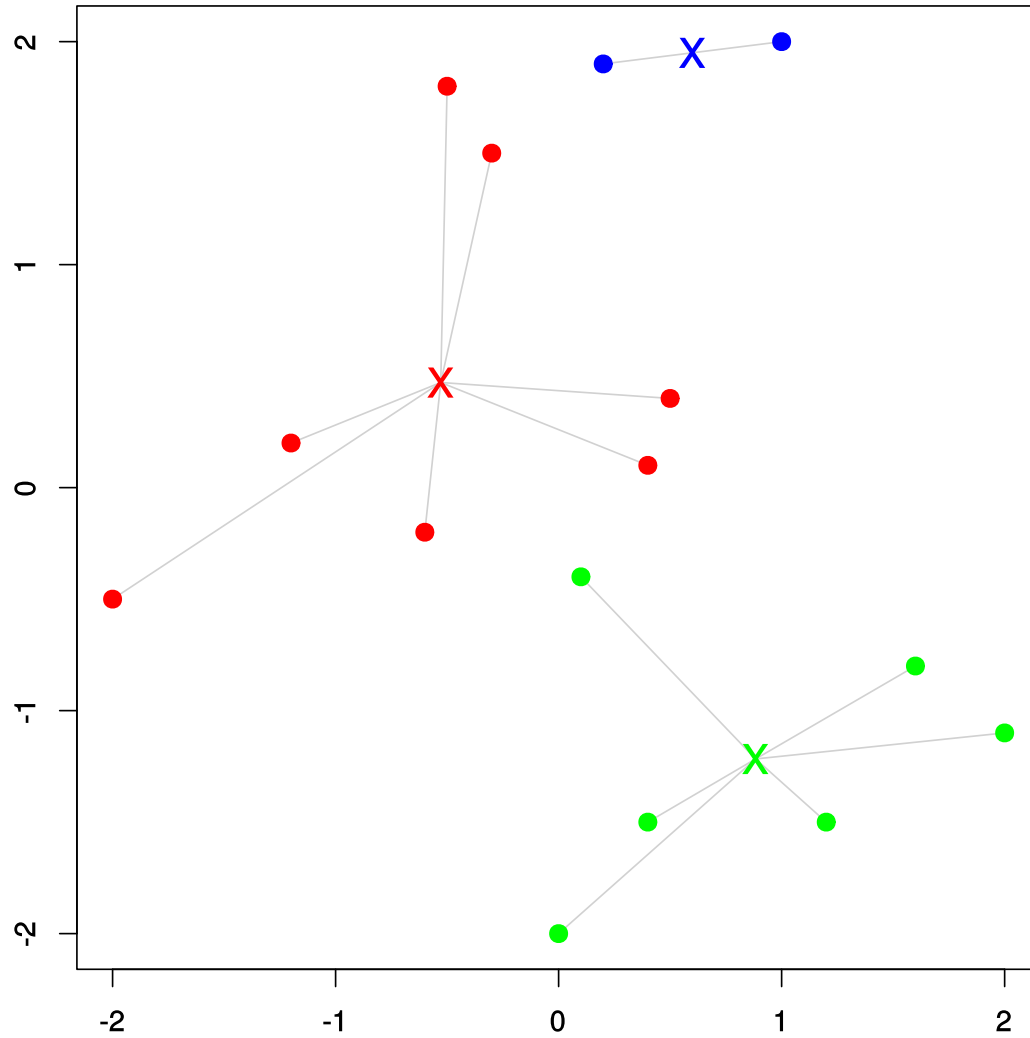
Zabudneme  $c_i$



Dopocítame nové  $c_i$  (suma klesla z 19.66 na 17.39)

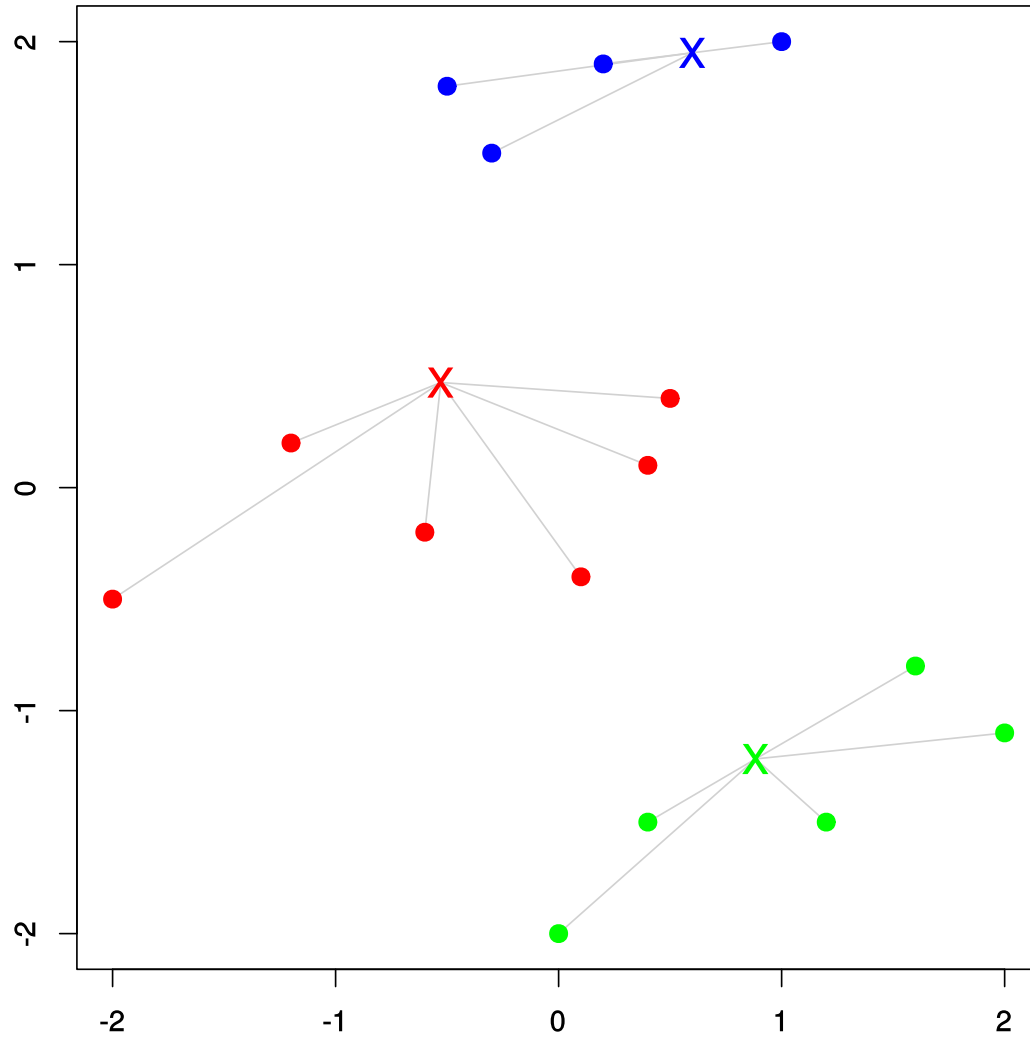


Prepočítame  $\mu_i$



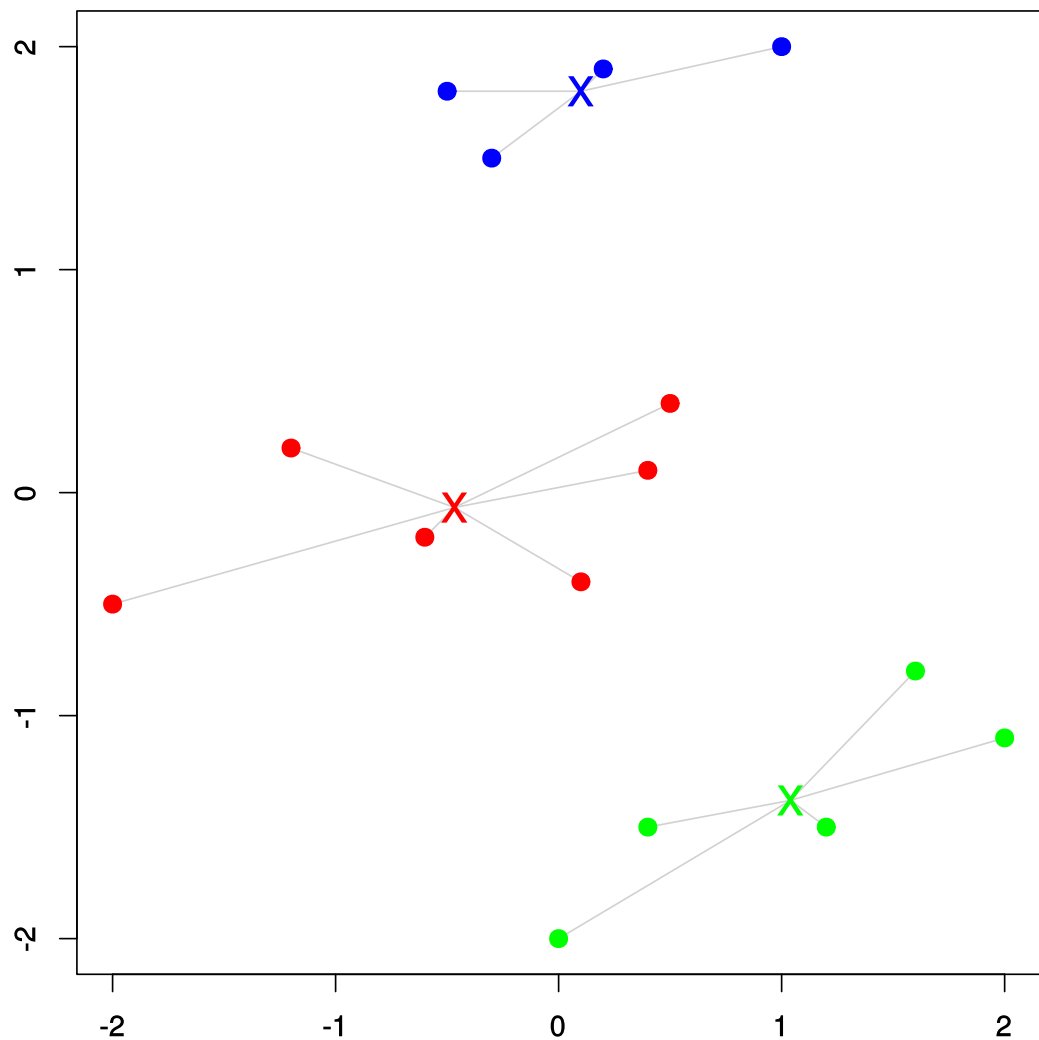
Sum of squared distances: 14.47

Prepočítame  $c_i$



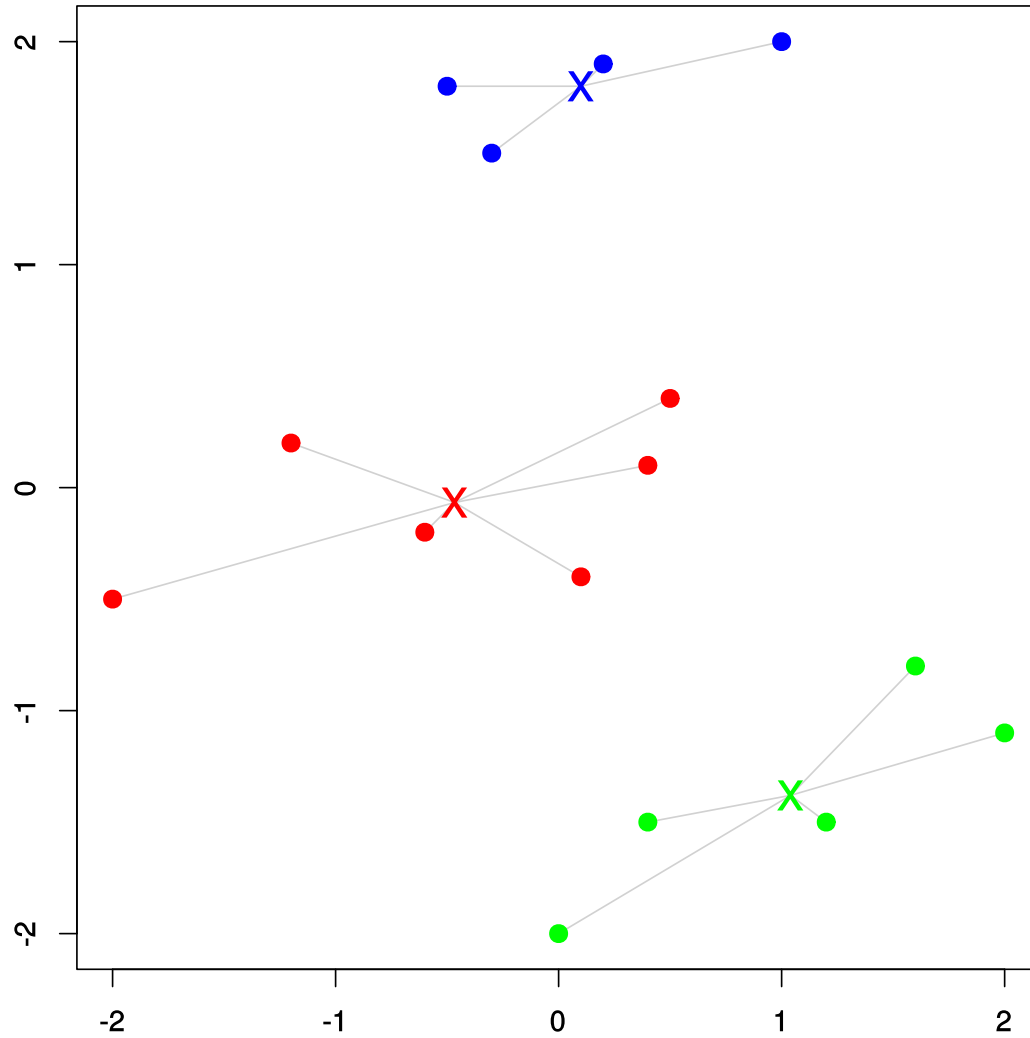
Sum of squared distances: 13.71

Prepočítame  $\mu_i$



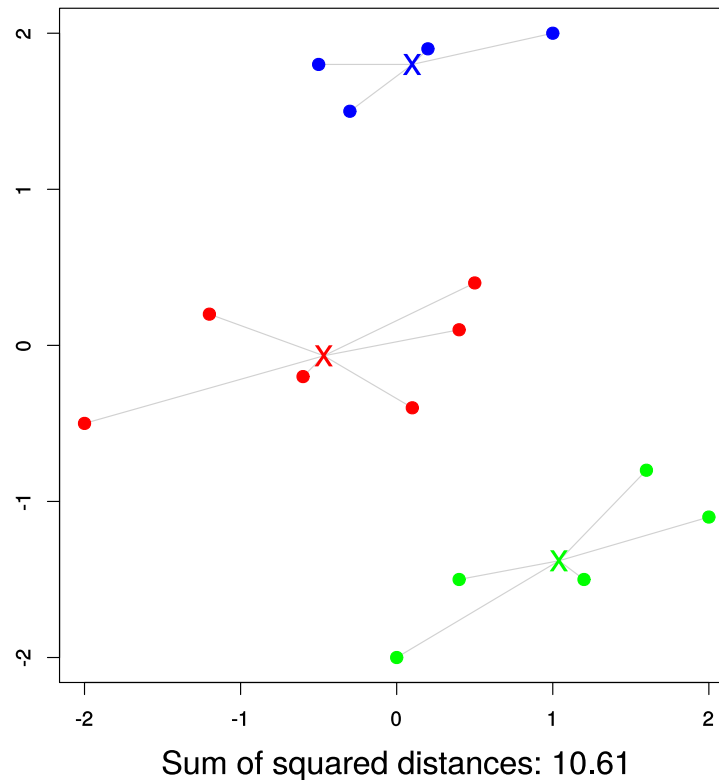
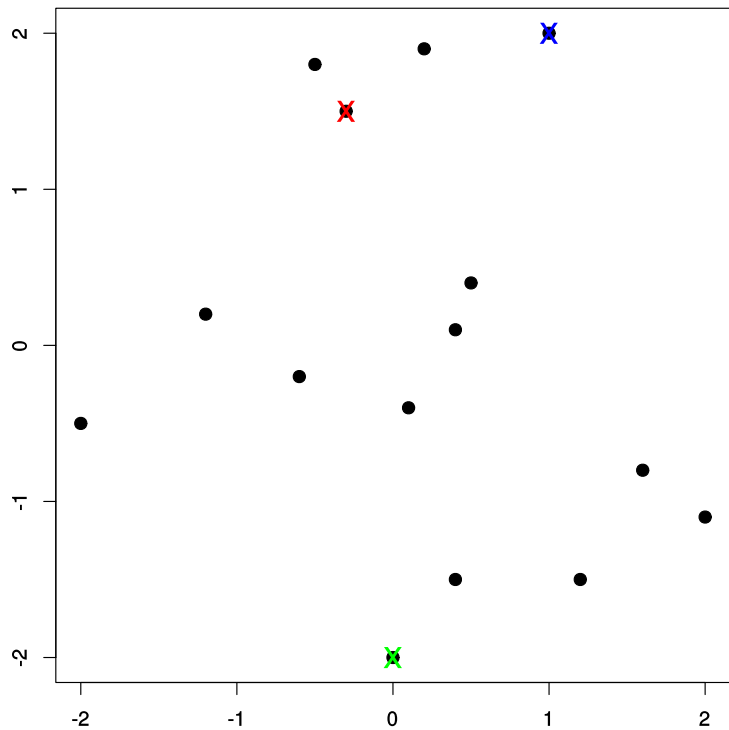


Prepočítame  $c_i$  (žiadna zmena, končíme)

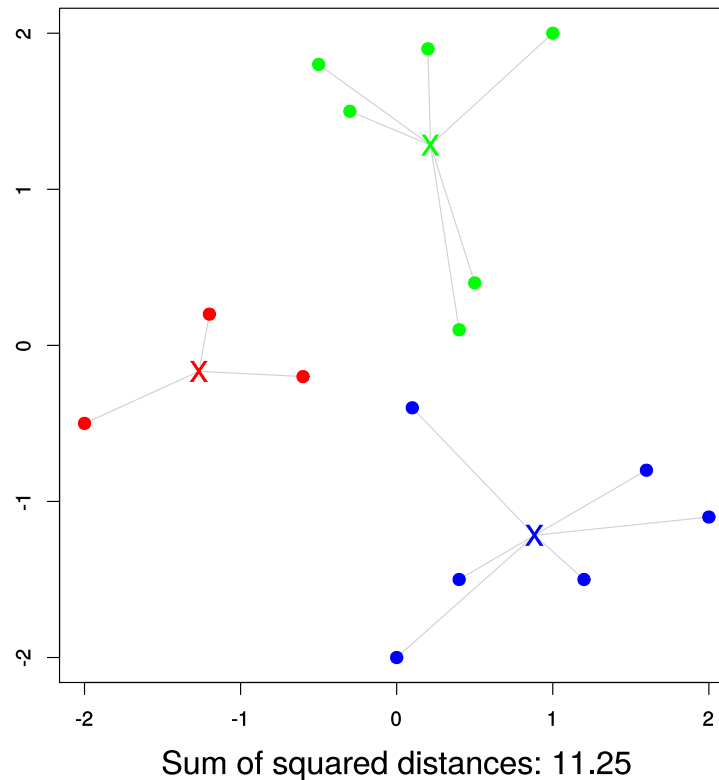
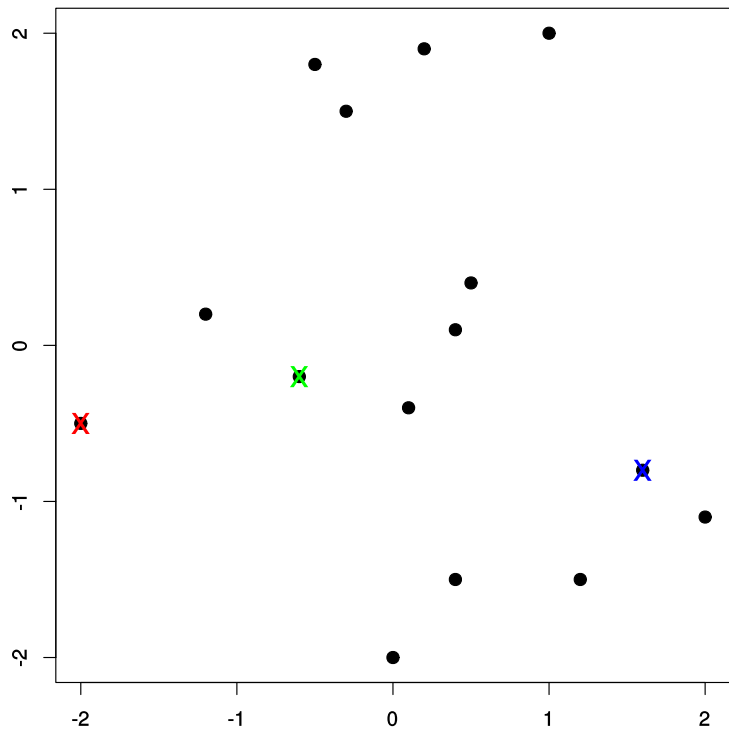


Sum of squared distances: 10.61

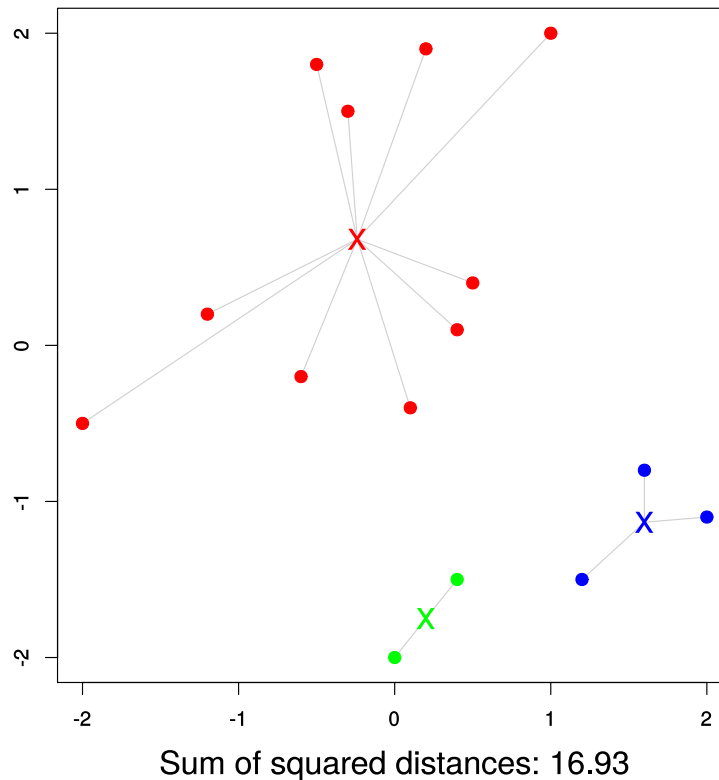
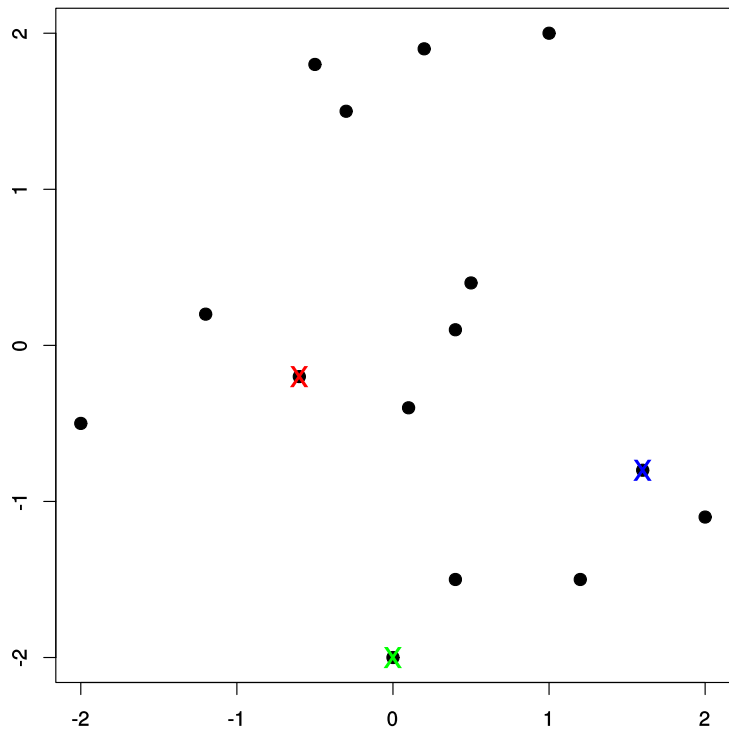
# Príklady niekoľkých behov programu



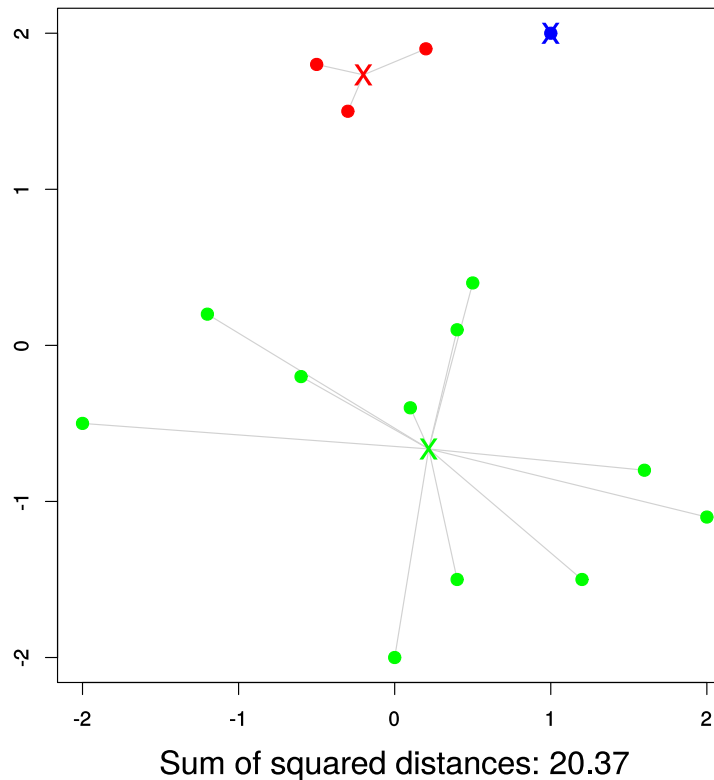
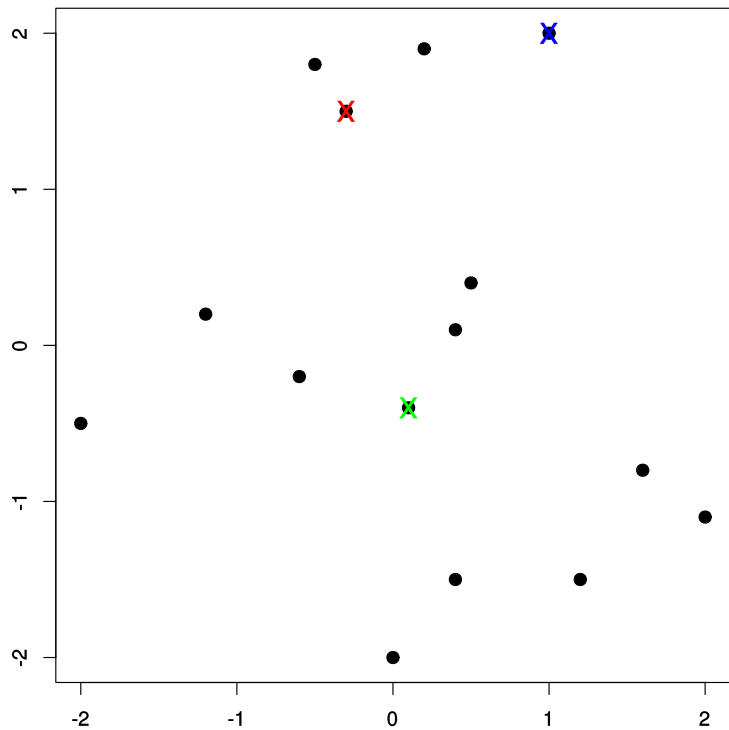
# Príklady niekoľkých behov programu



# Príklady niekoľkých behov programu



# Príklady niekoľkých behov programu



**Cvičenia pre biológov, 14.12.2017**  
**Zhrnutie semestra**

## Tvorba bioinformatického nástroja

- Sformulujeme biologické ciele  
(aké máme dáta, aké typy otázok sa chceme pýtať).
- Sformulujeme informaticky/matematicky  
(napr. ako pravdepodobnostný model).  
Dostaneme informatické zadanie problému, v ktorom je presne daný vzťah medzi vstupom a želaným výstupom  
(napr. nájsť zarovnanie s max. skóre v určitej skórovacej schéme).
- Hľadáme efektívne algoritmy na riešenie informatického problému.
- Ak sa nám nepodarí nájsť dosť rýchly algoritmus, použijeme heuristiky, ktoré dávajú približné riešenia.
- Testujeme na reálnych dátach, či sú výsledky biologicky správne  
(či bol model dobre zvolený, či heuristiky dobre fungujú).

## Použitie bioinformatického nástroja

- Sformulujeme biologické ciele (aké máme dáta, aké typy otázok sa chceme pýtať).
- Porozmýšľame, aký typ nástroja, resp. ich kombinácia by nám mohli pomôcť
- Alebo hľadáme v literatúre nástroj na typ problému, s ktorým sme sa ešte nestretli
- Pre správne nastavenie parametrov a interpretovanie výsledkov je dôležité poznať model, predpoklady, ktoré autori nástroja použili, resp. zdroj dát v príslušnej databáze
- Konkrétne nástroje a webstránky sa rýchlo menia, celkové princípy sa menia pomalšie



## Prehľad preberaných tém

- Zostavovanie genómov (najkratšie spoločné nadslovo, heuristiky, de Bruijnov graf)
- Zarovnanie (skórovanie ako pravdepodobnostný model, dynamické programovanie, heuristické zarovnávanie, E-value a P-value, lokálne vs. globálne, párové vs. viacnásobné, celogenómové)
- Evolúcia (pravdepodobnostné modely substitúcií, metóda maximálnej vierohodnosti, metóda maximálnej úspornosti, metóda spájania susedov)
- Hľadanie génov (skryté Markovove modely)
- Komparatívna genomika (hľadanie konzervovaných oblastí, komparatívne hľadanie génov, pozitívny výber, fylogenetické HMM, kodónové matice)

## Prehľad preberaných tém (pokračovanie)

- Expresia génov (zhlukovanie, klasifikácia, regulačné siete, transkripčné faktory, hľadanie motívov)
- Proteíny (predikcia štruktúry, profily a profilové HMM rodín/domén, protein threading)
- RNA štruktúra (dynamické programovanie, stochastické bezkontextové gramatiky)
- Populačná genetika (mapovanie asociácií, väzbová nerovnováha, genetický drift, štruktúra a história populácie)

## Nahliadli sme do sveta informatiky

- Algoritmus, časová zložitosť
- NP-ťažké problémy, presné algoritmy, heuristiky, aproximačné algoritmy
- Dynamické programovanie
- Stromy, grafy
- Skryté Markovove modely a bezkontextové gramatiky

## Ďalšie predmety

- **Genomika** N-mCBI-303, Nosek a kol. (LS, 2P, 3kr)
- **Seminár z bioinformatiky (1)-(4)** 2-AIN-503, 2-AIN-504, 2-AIN-251, 2-AIN-252, Vinař (ZS/LS, 2S, 2kr)
- **Linux pre používateľov** 1-AIN-500, Nagy (LS, 2K, 2kr)
- **Programovanie (1)** 1-MAT-130, Salanci (ZS, 2P/2C, 5kr)
- <http://compbio.fmph.uniba.sk/vyuka/>

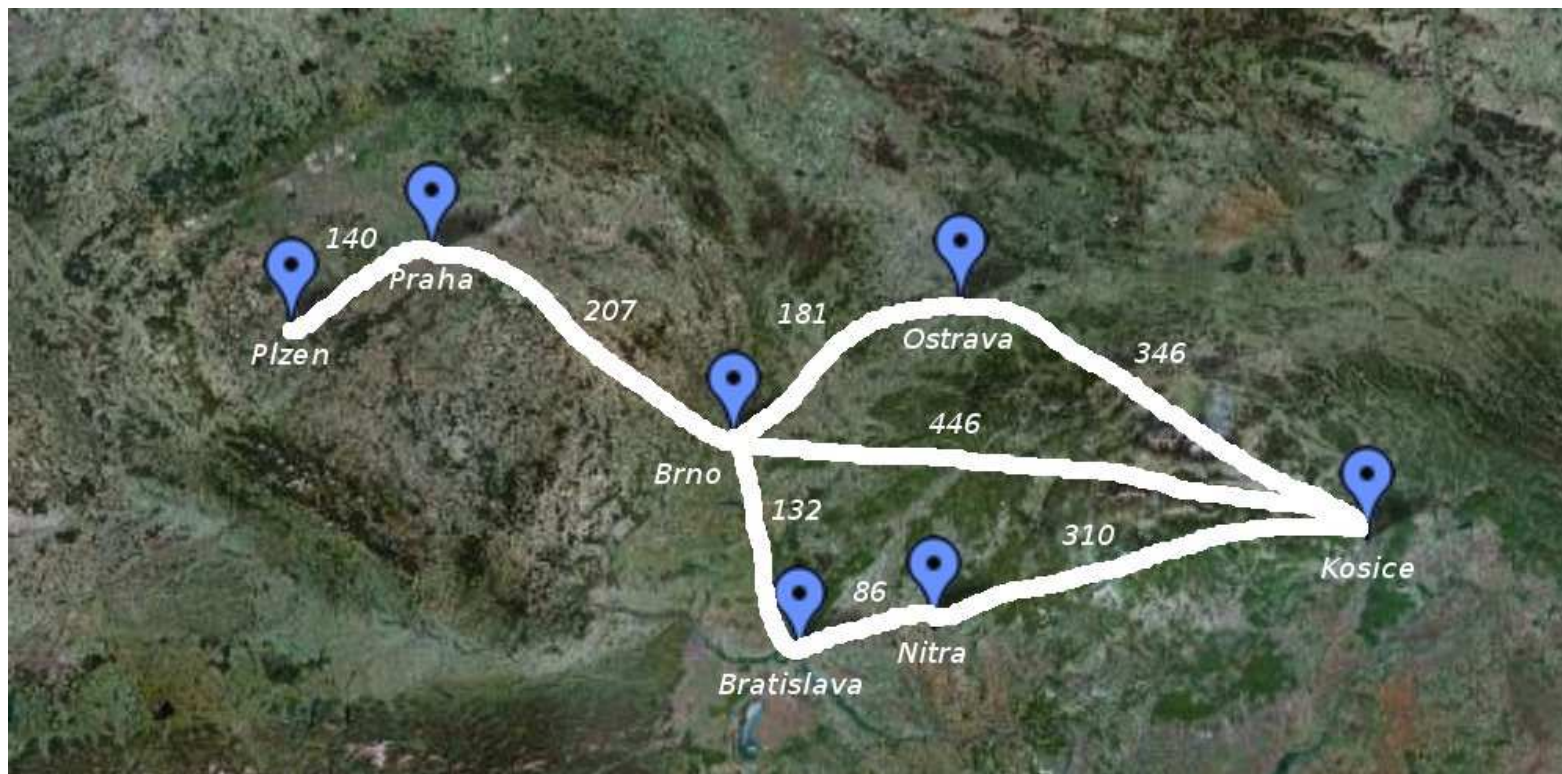
# Teória grafov

Broňa Brejová

14.12.2017

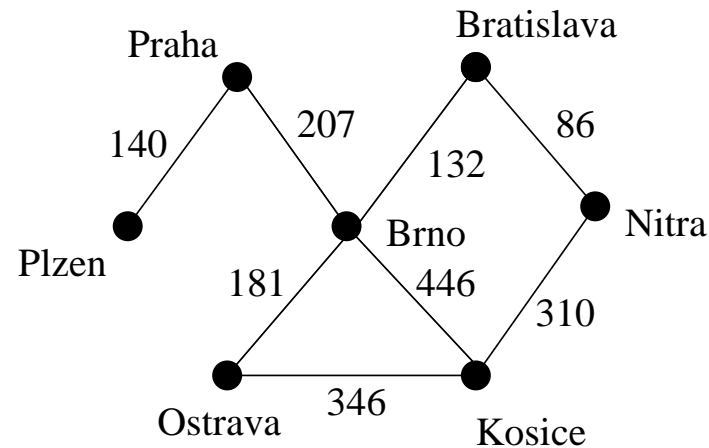
## Grafy a grafové algoritmy

**Graf:** 7 vrcholov (mestá), 8 hrán (cestné spojenia)



Počet vrcholov  $n$ , počet hrán  $m$   
Nezáleží na rozmiestnení vrcholov

**Cesta:** Postupnosť nadväzujúcich hrán, žiadny vrchol sa neopakuje



Napr. Plzeň–Praha–Brno–Bratislava je cesta  
Brno–Ostrava–Košice–Brno–Praha nie je cesta

**Najkratšia cesta z  $a$  do  $b$ :** Cesta spájajúca vrcholy  $a$  a  $b$  s najmenším súčtom vzdialeností na hranách

Možno spočítať v čase  $O(n^2)$  **Dijkstrovym algoritmom.**

**Cyklus:** Postupnosť nadväzujúcich hrán, ktorá sa vracia do východzieho bodu, nemá žiadne iné opakujúce sa vrcholy.





Procter and Gamble sůťaž, 1962



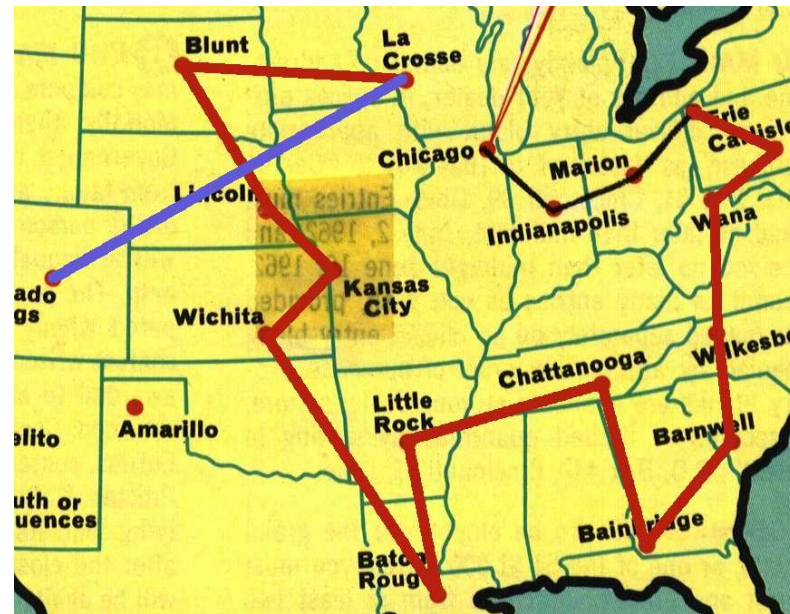
## Problém obchodného cestujúceho

**Vrcholy:** mestá na mape

**Hrany:** medzi každými dvoma vrcholmi, váha je vzdušná vzdialenosť

**Úloha:** obcestovať všetky mestá tak, aby celková vzdušná vzdialenosť bola minimálna (**Hamiltonovská kružnica**)

**Jednoduchá heuristika:** Vždy pokračuj v najbližšom meste, ktoré sme ešte nenavštívili.



Správny a efektívny algoritmus? Nanešťastie, obchodný cestujúci je **NP-ťažký problém**.

## Príklad: Sieť interakcií proteínov

**Vrcholy:** proteíny

**Hrany:** priame interakcie

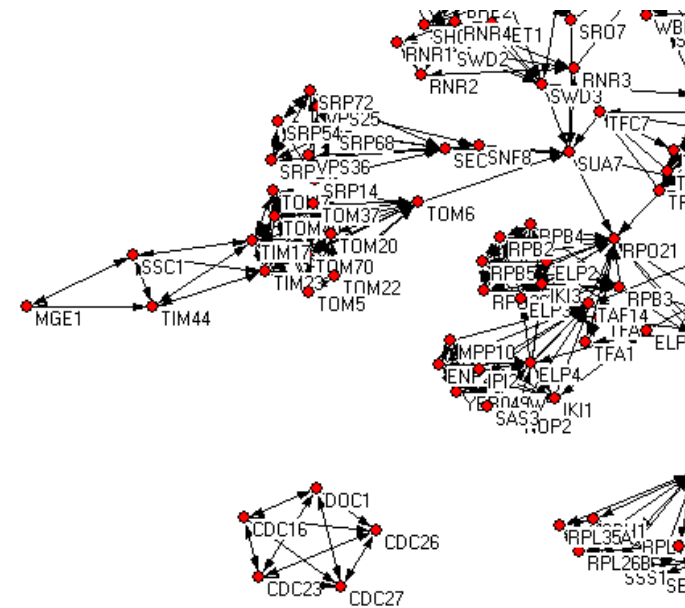
Metabolické dráhy zodp. **cestám**

Metabolické cykly zodp. **cyklom**

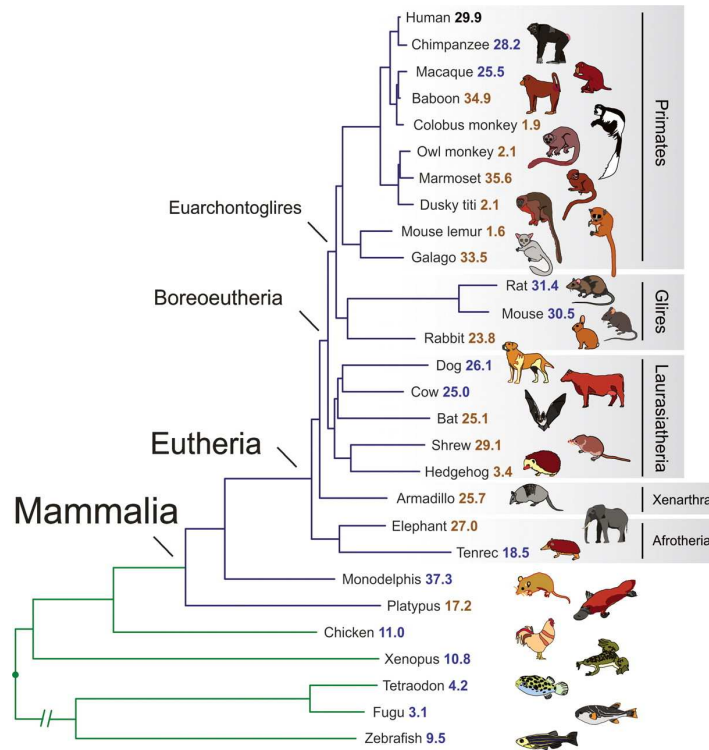
**Kliky:** Skupiny vrcholov priamo prepojené každý s každým

Komplexy zodpovedajú **klikám**

**Komponenty súvislosti:** Najväčšie skupiny vrcholov tak, aby sa v každom komponente dalo dostať z každého vrcholu do každého.



## Príklad: Fylogenetický strom



- **Stromy** sú špeciálna podtrieda grafov (acyklické, súvislé)
- Vrcholy: listy, vnútorné (spolu  $n$ )
- Hrany:  $n - 1$
- **Binárny strom:** každý vnútorný vrchol má 2 synov

**Ďalšie príklady stromov:** hierarchické zhlukovanie, dátové štruktúry na rýchle vyhľadávanie

**Ďalšie príklady grafov:** de Bruijnov graf, fylogenetická sieť (evolúcia s horizontálnym prenosom génov alebo rekombináciou), regulačné siete, hierarchia GO (gene ontology)