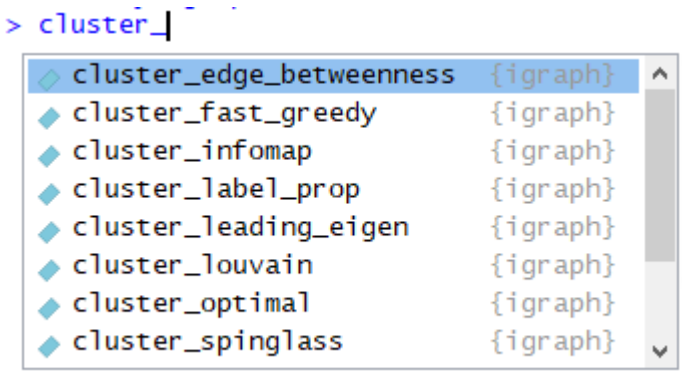
Hľadanie komunít v sieťach - časť II. (slajdy siete04.pdf)

1/ Úvod

- V predchádzajúcej časti: komunity, ktoré sú presne definované: klika (každý je prepojený s každým), jadro (každý je prepojený s aspoň K členmi jadra)
- Teraz:
  - intuitívny pojem komunity v zmysle „silno prepojená skupina vrcholov, ktorá je však slabo prepojená s ostatnými skupinami“, pričom sa snažíme všetky vrcholy rozdeliť do komunít
  - analógia zhlukovania vektorov (clustering), aj preto budeme používať pojem *zhlukovanie (clustering)* ako synonymum pre *určovanie komunít (community detection)*, funkcie v R-ku majú tiež v názve slovo *cluster*
- Algoritmus *walktrap* sme už videli v úvodnom prvom cvičení, v prvej domácej úlohe (pre tých, ktorí si vyberú túto možnosť hodnotenia) sa budú porovnávať rôzne algoritmy, využitie zhlukovania (ako hlavná téma alebo súčasť inej analýzy) môže byť aj v projektoch
- Zopakujeme si podrobnejšie *walktrap* algoritmus a ukážeme si *label propagation* algoritmus (a povieme si, z čoho pochádza ten názov), ako námet na projekt je daná možnosť vysvetliť a samostatne naprogramovať prvý krok *eigenvalue* algoritmu (a doplniť ten výpočet ešte niečím ďalším podľa vlastného výberu), R-ko má viac takýchto algoritmov.



- Medzi algoritmami je aj *cluster_optimal*, takže si vysvetlíme, o akú optimalitu ide (a prečo máme veľa ďalších algoritmov, keď máme aj tento „optimálny“

2/ Váhy v prípade vážených sietí

- Budeme pracovať s neorientovanými sieťami.
- Tieto môžu byť nevážené alebo vážené. V prípade vážených sietí tieto algoritmy budú vyžadovať, aby váhy vyjadrovali silu väzby medzi vrcholmi.
  - Je to teda naopak ako pri počítaní centrality blízkosti a medzipolohy, tam sme potrebovali váhy vyjadrujúce vzdialenosti. Ak je váhou hrany vzdialenosť (v nejakom zmysle), tak veľká hodnota váhy znamená, že vrcholy si nie sú blízke. V prípade zhlukovania potrebujeme, aby veľká hodnota hrany znamenala veľkú blízkosť vrcholov, ich silné spojenie, silnú väzbu, …
  - Teda napríklad v prípade karate klubu, kedy váha vyjadruje, koľkými rôznymi spôsobmi sa ľudia stretávali mimo klubu, môžeme tieto váhy použiť pri zhlukovaní (kým pri výpočte spomínaných centralít sme ich museli transformovať alebo pracovať s neváženou sieťou).
  - Ak si zobrazíme help k funkcii napríklad *cluster_optimal*, nájdeme to aj tam :

`weights` Optional positive weight vector for optimizing weighted modularity. If the graph has a `weight` edge attribute, then this is used by default. Supply `NA` to ignore the weights of a weighted graph. Larger edge weights correspond to stronger connections.

  -

3/ Modularita

- Najskôr uvažujme neváženú sieť.
- Na str. 5 v slajdoch:

▶ Inituitívne: modularita bude *(počet hrán vrámci komunít) mínus (očakávaný počet hrán vrámci komunít)* a budeme ju maximalizovať

- Teda modularita vyjadruje, o čo viac hrán je v komunitách, ako by sme to očakávali na základe nejakého modelu. Budeme potom hľadať také rozdelenie na komunity, kedy je tento rozdiel čo najväčší
- Na základe „nejakého modelu“ - treba špecifikovať, akého. Označme najskôr vo všeobecnosti $\mathbf{P_{ij}}$ pravdepodobnosť, že medzi vrcholmi **i, j** vznikne na základe modelu hrana. Nech **A** je matica susednosti. Definujeme na základe predchádzajúcej úvahy modularitu

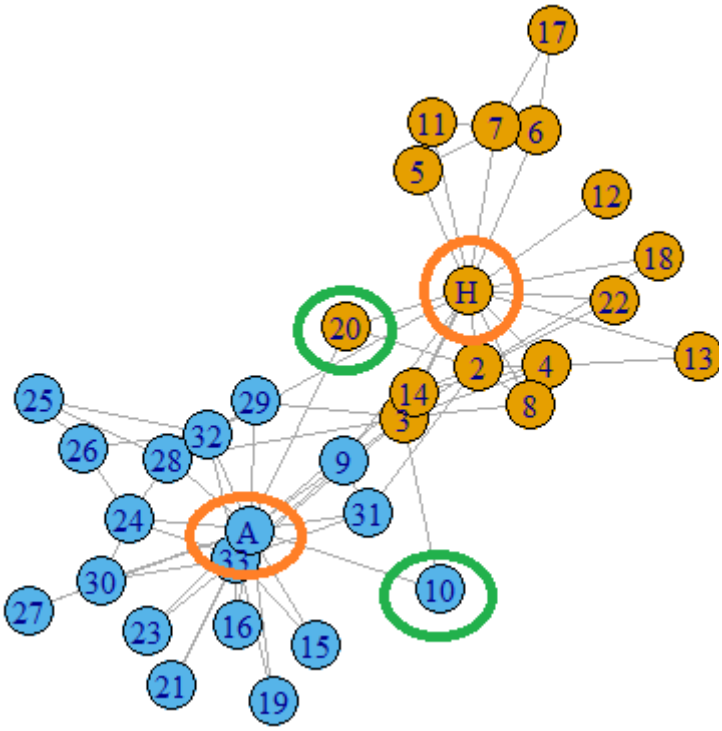
▶ Definujme modularitu ako

$$Q = \frac{1}{2m} \sum_{i,j} [A_{ij} - P_{ij}] \delta(g_i, g_j),$$

kde g_i je komunita, do ktorej patrí vrchol i a $\delta(x, y)$ je 1 pre $x = y$ a inak 0. Optimálne zhlukovanie ju bude maximalizovať.

a budeme sa zamýšľať nad tým, aké majú byť tie pravdepodobnosti **P.**

- Na prvom cvičení sme sa stretli s modelom Erdösa a Rényiho
  - Tu každá hrana vzniká s pravdepodobnosťou *p* a hrany vznikajú nezávisle.
  - To by znamenalo, že každé $\mathbf{P_{ij}} = \mathbf{p}$.
  - Prečo to nemusí byť dobrá voľba – prečo Erdösov a Rényiho model nemusí byť dobrým modelom pre siete, ktoré pozorujeme?



- Zoberme karate klub.
- ER model by znamenal, že počet hrán – teda počet kontaktov – by bol rovnaký napríklad pre inštruktora ako pre každého člena klubu.
- Ten vyšší počet hrán práve pre A a H (značenie z obrázka) nie je zrejme dôsledkom náhody (aj pri ER modeli vznikajú vrcholy s vyšším stupňom, ale je to dôsledok náhodnosti).
- Preto by sme chceli taký model pre náhodný vznik hrán, ktorý by predpokladal, že tieto konkrétne vrcholy budú mať veľa hrán.

- Model – str. 7 v slajdoch:

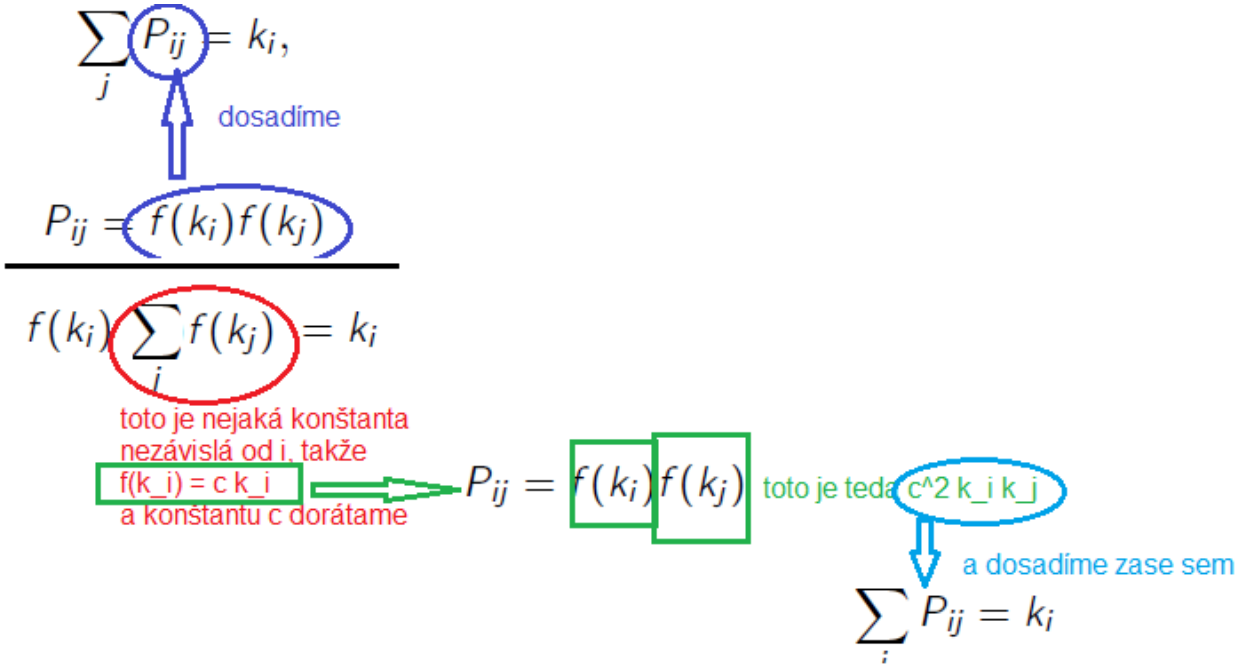
  - Označme $k_i = degree_i$ stupeň vrchola i
  - Očakávaný stupeň každého vrchola sa bude rovnať pozorovanému stupňu:

$$\sum_j P_{ij} = k_i,$$

  - Suma na ľavej strane je očakávaný počet hrán vrchola **i** – premyslite si, prečo (základná myšlienka je použiť indikátory, pre každý iný vrchol **j** definujme náhodnú premennú s hodnotami 0 a 1 vyjadrujúcu, či je **i** s ním spojený – počet hrán je súčet týchto náhodných premenných a stredná hodnota počtu je súčet stredných hodnôt indikátorových premenných – doplňte podrobnosti výpočtu)
  - Táto požiadavka teda hovorí, že očakávaný počet hrán vrchola sa má rovnať pozorovanému počtu.
  - Hľadajme riešenie v separovanom tvare: $P_{ij} = f(k_i)f(k_j)$
  - Riešenie zo slajdov:

  - Dostaneme $P_{ij} = \frac{k_i k_j}{2m}$

    Ako ho dostaneme:



Už je to skoro hotové:
- na ľavej strane je c^2 * k_i * sum(k_j) = c^2 * k_i * 2 *m
  (ak sčitujeme stupne všetkých vrcholov, tak každú hranu zarátame 2x)
- to má byť rovné k_i, teda c = 1/sqrt(2*m)
- a teda f(k_i) = c k_i = k_i / sqrt(2*m), a napokon P_{ij} = k_i k_j / (2*m)

- Takže v definícii modularity

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - P_{ij}]\delta(g_i, g_j),$$ budeme mať toto $$P_{ij} = \frac{k_i k_j}{2m}$$

- V prípade váženej siete sa namiesto matice susednosti **A** berie matica, v ktorej sú váhy hrán (a nula, ak tam hrana nie je) a **k** namiesto stupňa vrcholov vyjadruje súčet váh hrán, ktoré vychádzajú z daného vrchola.

4/ Výpočet modularity pre dané rozdelenie vrcholov do komunít

- Pripomeňme si, že vyššia modularita znamená lepšie vytvorené komunity.
- Môžeme teda porovnávať rôzne rozdelenia vrcholov do komunít – tie, ktoré nám dali rôzne algoritmy alebo ktoré sme si vytvorili my
- Môžeme sa tiež snažiť maximalizovať modularitu – riešiť optimalizačnú úlohu, kde maximalizujeme vzhľadom na všetky možné rozdelenia vrcholov do komnunít.

<u>Výpočet modularity pre dané komunity v R-ku:</u>

- Funkcia **modularity**
- Vstupom je:
  - výsledok niektorej funkcie typu **cluster_,** napr. **cluster_walktrap** – môžeme sa presvedčiť, že to objekt špeciálneho typu „communities“ - obsahuje informáciu o sieti aj o rozdelení vrcholov, takže nič iné na výpočet modularity nepotrebujeme

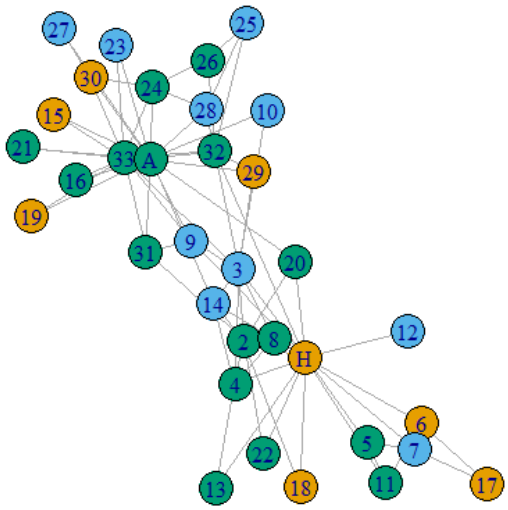
```
> com <- cluster_walktrap(karate)
> class(com)
[1] "communities"
```

  - alebo môžeme zadať sieť a vektor, ktorého i-ta zložka vyjadruje číslo komunity, do ktorej je zaradený i-ty vrchol siete
- Napríklad pre karate klub
  - Zoberieme výstup z **cluster_walktrap**
  - Náš vektor pre zaradenie do komunít, pričom každý z vrcholov zaradíme náhodne do jednej z troch komunít
  - Zobrazte si výstup zhlukovanie z *walktrap* algoritmu:

```
> com <- cluster_walktrap(karate)
> plot(com, karate)
```

  - Pre naše zhlukovanie môžeme spraviť

```
> set.seed(123)
> indexy <- sample(1:3, size = 34, replace = TRUE)
> plot(karate, vertex.color = indexy)
```



a vidíme, že to rozdelenie je nezmyselné.

  - Rozdiel v kvalite zhlukovania vidieť na modularite:

```
> modularity(com)
[1] 0.440181
> modularity(karate, indexy)
[1] -0.008136095
```

- Spravte si podobné porovnanie zhlukovaní pre sieť **kite** (z knižnice **datasets**) podľa slajdu 8.

Poznámky:

- Slajd 9: Ak si vytvoríme vlastné rozdelenie vrcholov do komunít, môžeme z neho tiež vytvoriť objekt triedy **communities** a pracovať s ním tak, ako s výstupom z R-kovskej funkcie pre zhlukovanie (napríklad sa to môže hodiť pri kreslení obrázkov).
- Slajd 10: Vyznačovanie skupiny vrcholov podobným formátovaním ako pri nakreslení komunít sa nám môže hodiť aj v iných súvislostiach, keď chceme zvýrazniť nejakú skupinu vrcholov – na to slúži parameter **mark.groups** pri volaní funkcie **plot.**

5/ Optimalizácia modularity – optimálne komunity

- Funkcia **cluster_optimal** nájde také rozdelenie vrcholov do komunít, ktoré maximalizuje modularitu.
- Zbehne pre nie veľmi veľké siete (nie je to jednoduchá optimalizačná úloha)
- Aká úloha sa rieši:

Z pôvodného článku *U. Brandes et al.: On Finding Graph Clusterings with Maximum Modularity*:

ak sú vrcholy u, v v tej istej komunite, tak X_{uv} = 1, inak 0

The problem of maximizing modularity can be cast into a very simple and intuitive integer linear program (ILP). Given a graph $G = (V, E)$ with $n := |V|$ nodes, we define n^2 decision variables $X_{uv} \in \{0, 1\}$, one for every pair of nodes $u, v \in V$. The key idea is that these variables can be interpreted as an equivalence relation (over V) and thus form a clustering. In order to ensure consistency, we need the following constraints, which guarantee

reflexivity $\forall\, u\colon X_{uu} = 1$ — vrchol je sám so sebou v tej istej komunite

symmetry $\forall\, u, v\colon X_{uv} = X_{vu}$ and — či sú vrcholy u, v v tej istej komunite, je to isté, ako či sú v jednej komunite vrcholy v, u

transitivity $\forall\, u, v, w\colon \begin{cases} X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1 \\ X_{uw} + X_{uv} - 2 \cdot X_{vw} \leq 1 \\ X_{vw} + X_{uw} - 2 \cdot X_{uv} \leq 1 \end{cases}$ — týmto vyjadrujeme tranzitivitu: ak sú vrcholy u, v v rovnakej komunite a aj vrcholy v, w sú v rovnakej komunite, tak aj vrcholy u,w sú v rovnakej komunite

The objective function of modularity then becomes

$$\frac{1}{2m} \sum_{(u,v) \in V^2} \left(E_{uv} - \frac{\deg(u) \deg(v)}{2m} \right) X_{uv} \ ,$$

$$\text{with} \quad E_{uv} = \begin{cases} 1 & , \text{if } (u, v) \in E \\ 0 & , \text{otherwise} \end{cases} .$$

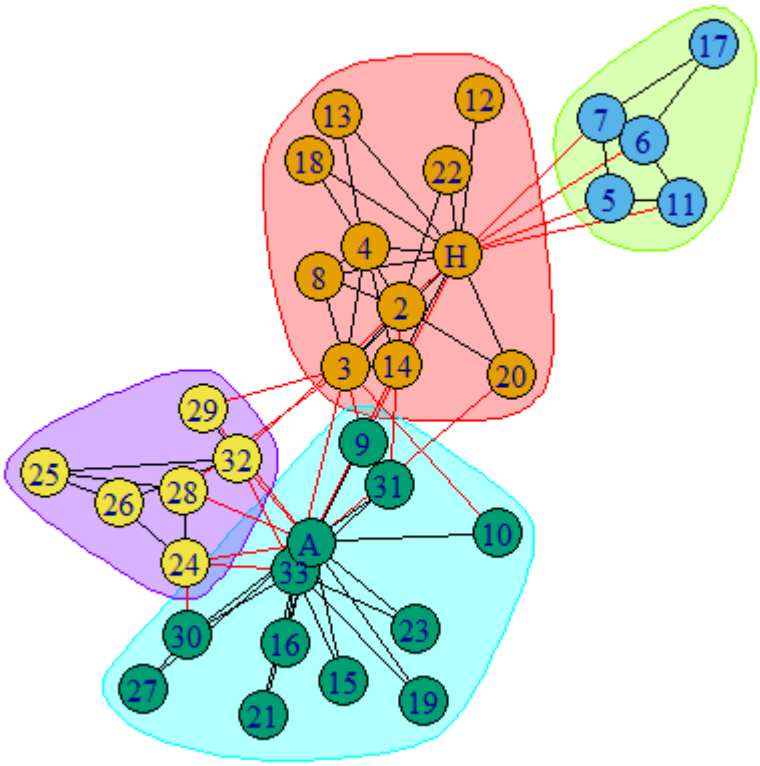
- Prečo funguje ten zápis podmienky tranzitivity: Keďže X nadobúda hodnoty iba 1 a 0, tak nerovnosť tvaru

$$X_{uv} + X_{vw} - 2 \cdot X_{uw} \leq 1$$

môže byť porušená len tak, že prvé dva sčítance sú 1 (teda u, v sú v rovnakej komunite a v, w sú v rovnakej komunite) a odpočítavame nulu (teda u, w nie sú v rovnakej komunite).
- Podmienky reflexivity a symetrie by sme mohli odstrániť a prepísať – po očíslovaní vrcholov 1, 2, …, n – pomocou X definovaných len pre i menšie ako j. Charakter úlohy a jej zložitosť by sa tým nezmenila.
- Ide o úlohu celočíselného lineárneho programovania (všetky X v ohraničeniach aj účelovej funkcii vystupujú lineárne).
- Použitie pre karate klub:

```
> com <- cluster_optimal(karate)
> plot(com, karate)
```

Dostaneme:



6/ Motivácia pre iné algoritmy

- Teda: Načo sú nám iné algoritmy, keď vieme optimalizovať priamo modularitu?
- 3 odpovede:
  - pre veľké siete ju optimalizovať nevieme, funkcia nezbehne
  - výsledky optimalizácie modularity nemusia byť vždy v súlade s intuíciou a teda to nemusí byť naše jediné kritérium – konkrétne príklady z citovaného článku sú na slajdoch 15, 16, 17
  - môže chcieť zadať vlastný počet komunít (funkcia **cluster_optimal** to neumožňuje, niektoré iné algoritmy áno)

7/ Walktrap algoritmus

- Str. 19:

▶ *walk* - prechádzka, *trap* - pasca

▶ pri náhodnej prechádzke po vrcholoch siete by sme mali zostať s veľkou pravdepodobnosť v tej istej komunite

▶ v R-ku `cluster_walktrap`

- Použitie výstupu cluster_walktrap sme už videli – znovu na str. 20 v slajdoch

- Ako získať zhlukovanie s nami zvoleným počtom komunít - funkcia **cut_at**
  - vstupom je výstup zo zhlukovania a parameter **no** udávajúci počet komunít, ktoré chceme
  - výstupom je vektor, i-ta zložka vyjadruje index komunity, do ktorej je zaradený i-ty vektor

```
nase_rozdelenie_3 <- cut_at(komunity, no = 3)
nase_rozdelenie_3
```

```
##  [1] 2 2 2 2 3 3 3 2 1 1 3 2 2 2 1 1 3 2 1
```

Teda napríklad vrcholy 1, 2, 3, 4 sú v druhej komunite, potom vrcholy 5, 6, 7 sú v tretej, atď.

- Tento vektor môžeme použiť na kreslenie obrázku, na porovnávanie rozdelení do komunít a pod.
- Str. 23 – porovnanie modularity pre rôzny počet komunít:

```
for (i in 2:6){
  nase_rozdelenie <- cut_at(komunity, i)
  print(modularity(karate, nase_rozdelenie))
}
```

```
## [1] 0.3714661
## [1] 0.3990796
## [1] 0.4111604
## [1] 0.398833
## [1] 0.3908613
```

8/ Label propagation algoritmus

- Str. 25:

- label - nálepka, značka, propagation - šírenie
- vrcholy dostanú značky a potom sa značky menia tak, aby mal vrchol takú značku ako väčšina jeho susedov
- v R-ku: cluster_label_prop

- Pracuje sa s ním analogicky ako s predchádzajúcimi algoritmami
- Na rozdiel od *walktrap* algoritmu nemáme možnosť voliť počet zhlukov – toto je chyba, ktorú dostaneme, ak sa pokúsime zadať počet zhlukov pre výstup, pre ktorý sa to nedá:

```
> nase_rozdelenie_3 <- cut_at(komunity, no = 3)
Error in cut_at(komunity, no = 3) :
  Not a hierarchical communitity structure
```

9/ Ďalšie algoritmy

- Ešte raz obrázok z úvodnej časti týchto poznámok →
- Rôzne algoritmy, založené na rôznych myšlienkach
- Niekedy sa dá voliť počet zhlukov, niekedy nie
- Niektoré obsahujú náhodnosť, pri viacnásobnom spustení môžu dať rôzny výsledok – oplatí sa vtedy použiť set.seed.