Complex Networks 2022

Node Vector Distances: Methods and Applications

Michele Coscia
ITU København
November 7th, 2022

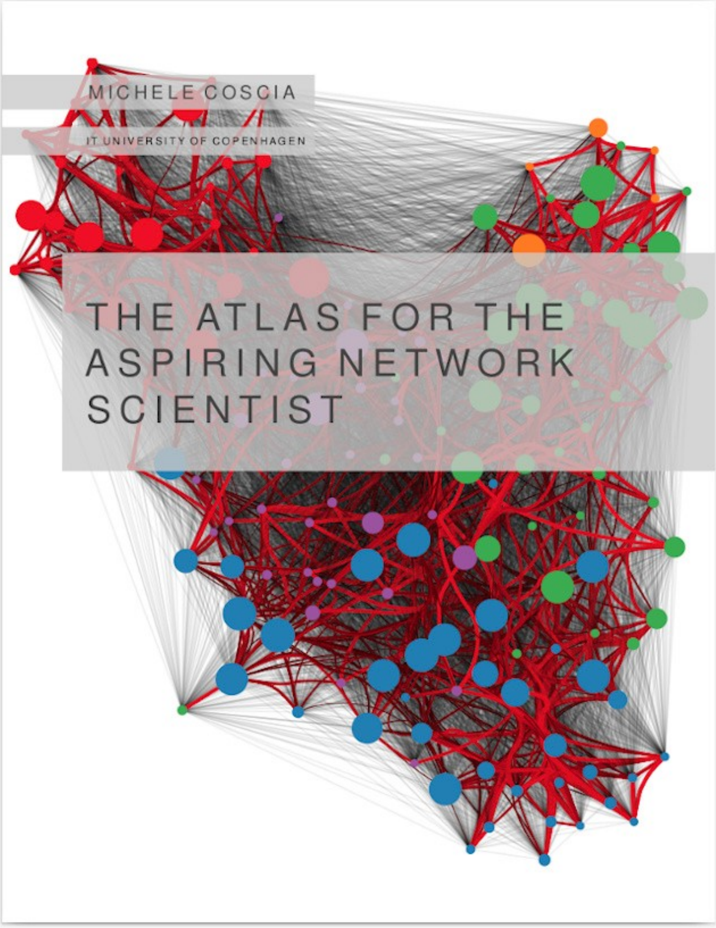
IT UNIVERSITY OF COPENHAGEN

Download the Material

https://github.com/mikk-c/complexnetworks22-tutorial

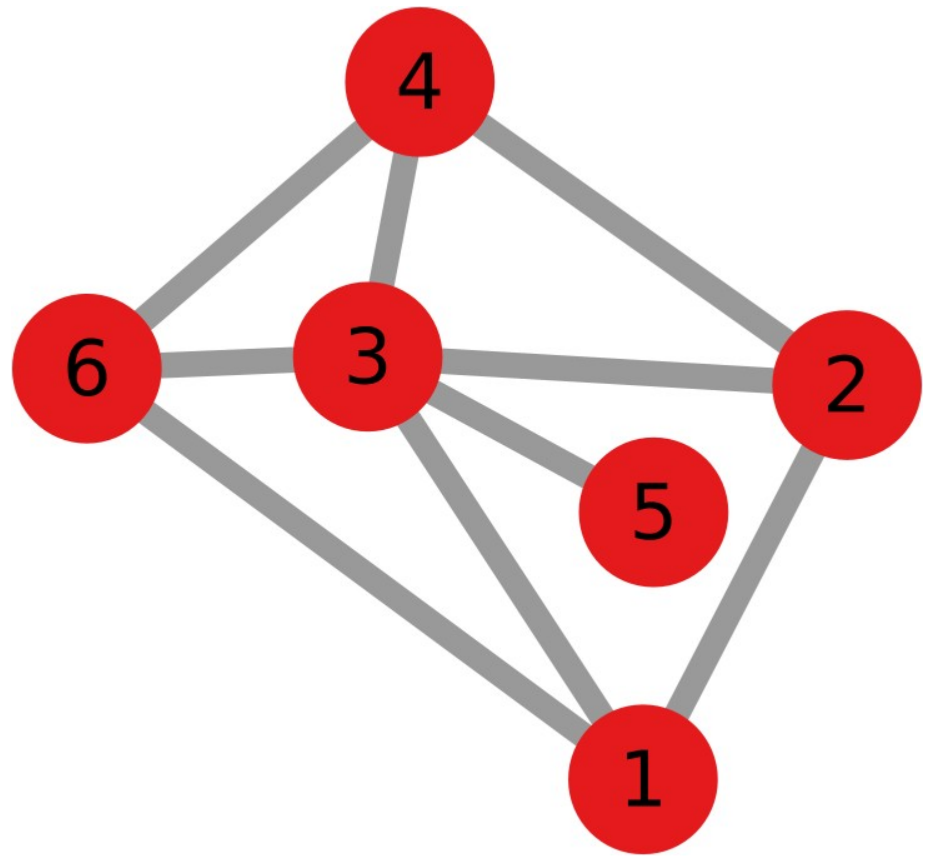
Readings

- The Atlas for the Aspiring Network Scientist
  - Chapter 40
  - https://www.networkatlas.eu/
- The node vector distance problem in complex networks, ACM Computing Surveys (CSUR) 53 (6), 1-27



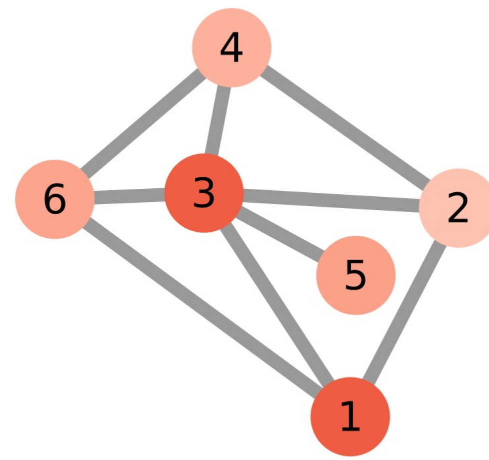
NVD: What?

- A graph G



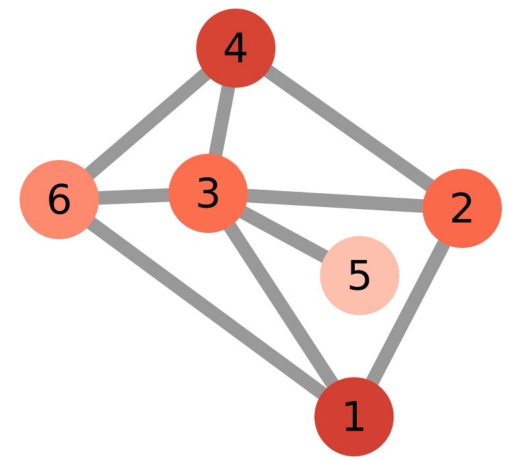
NVD: What?

- A graph G
- Two vectors v_1 and v_2
  - One value per node
- How far is v_1 from v_2?



0.57
0.13
0.57
0.20
0.27
0.25

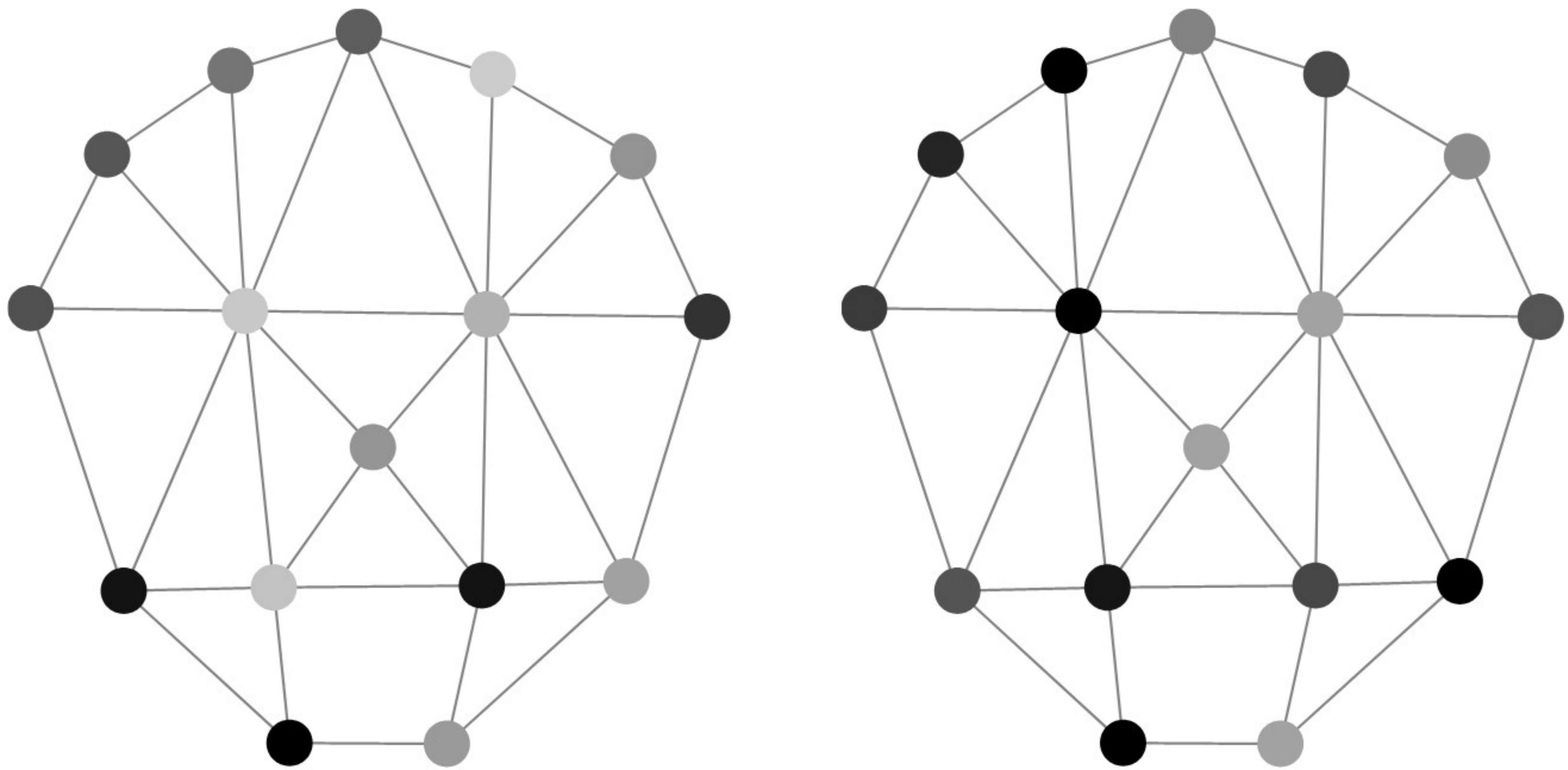
v_1



0.73
0.51
0.48
0.71
0.13
0.37

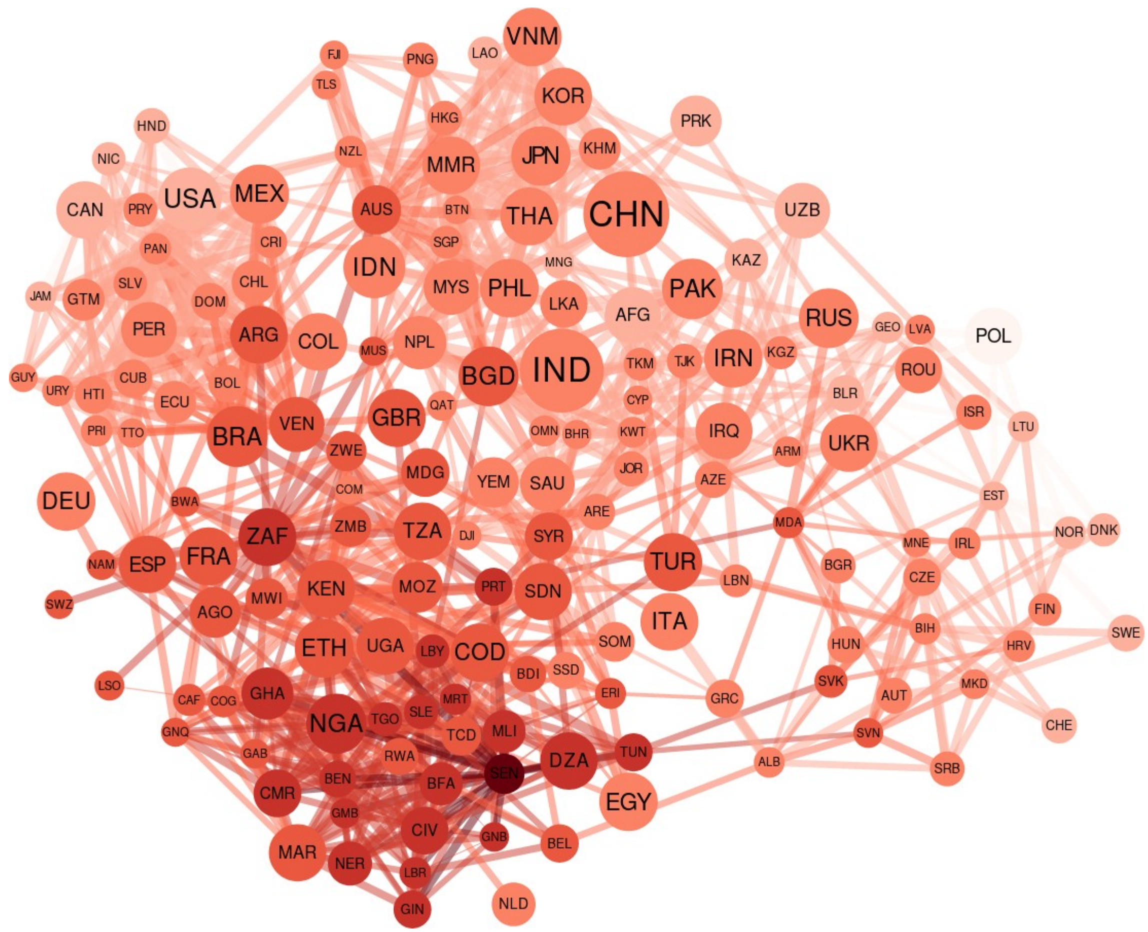
v_2

NVD: Why?

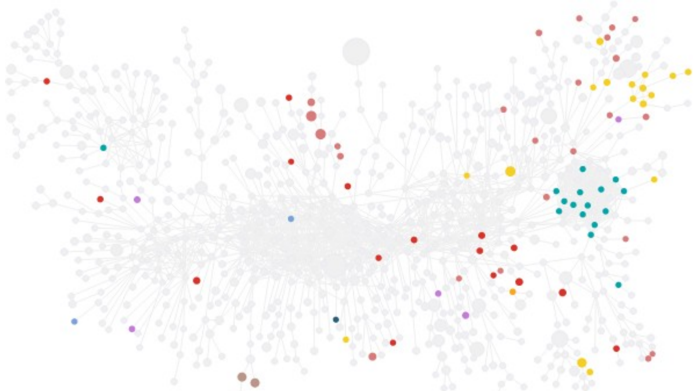


(a) Face #1 (b) Face #2

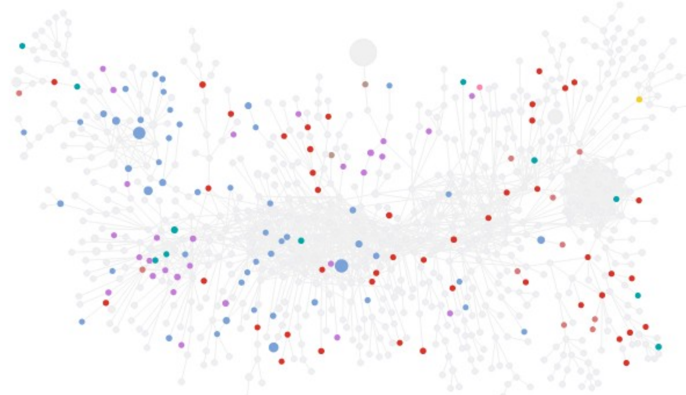
NVD: Why?


VNM
FJI
PNG
LAO
TLS
KOR
HND
HKG
PRK
NIC
NZL
KHM
JPN
MMR
CAN
PRY
USA
MEX
AUS
BTN
THA
CHN
UZB
CRI
SGP
PAN
IDN
MNG
KAZ
JAM
SLV
CHL
MYS
PHL
PAK
GTM
DOM
LKA
AFG
RUS
GEO
LVA
POL
PER
ARG
COL
NPL
MUS
IRN
KGZ
TKM
TJK
BGD
IND
ROU
CUB
GUY
BOL
URY
HTI
ECU
CYP
BLR
QAT
GBR
ISR
LTU
PRI
TTO
VEN
OMN
BHR
KWT
IRQ
BRA
ARM
UKR
ZWE
MDG
YEM
SAU
JOR
AZE
COM
DEU
BWA
EST
ARE
ZMB
MDA
TZA
DJI
SYR
NOR
DNK
ZAF
MNE
IRL
FRA
ESP
NAM
TUR
LBN
BGR
CZE
KEN
MOZ
PRT
SDN
MWI
SWZ
AGO
FIN
ITA
BIH
SWE
ETH
UGA
LBY
COD
SOM
HUN
HRV
BDI
SSD
LSO
GHA
CAF
COG
MRT
ERI
GRC
SVK
AUT
MKD
NGA
TGO
SLE
CHE
GNQ
TCD
MLI
DZA
SVN
GAB
RWA
TUN
ALB
SEN
BEN
BFA
SRB
CMR
EGY
GMB
CIV
GNB
BEL
MAR
NER
LBR
GIN
NLD

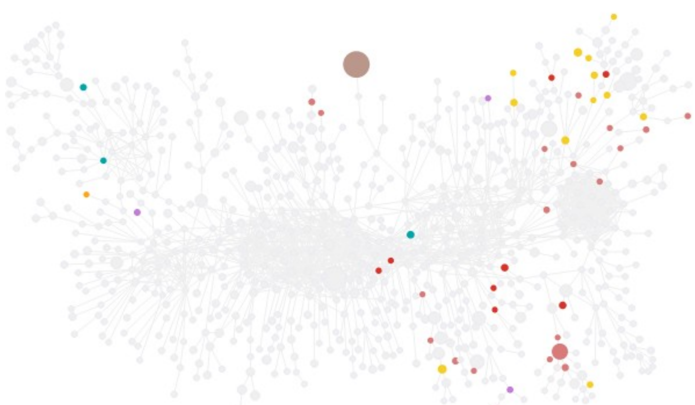
NVD: Why?



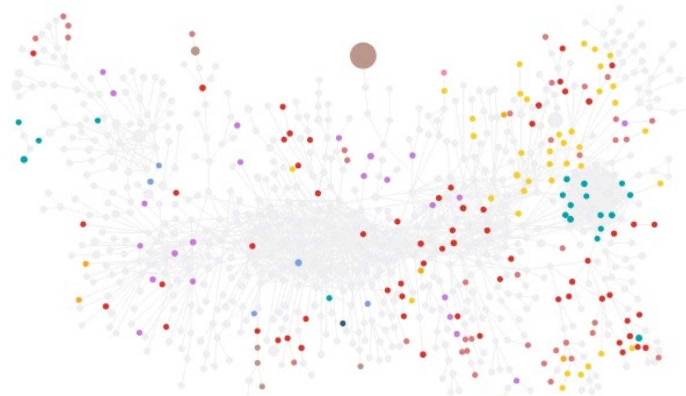
(a) Korea 1962



(b) Korea 2013

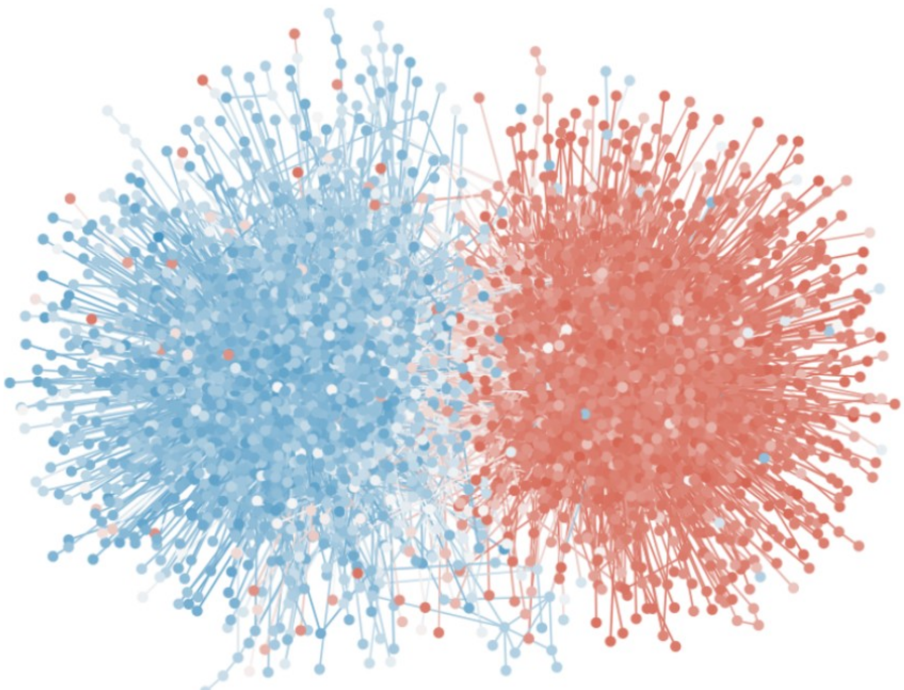


(c) Egypt 1962



(d) Egypt 2013

NVD: Why?



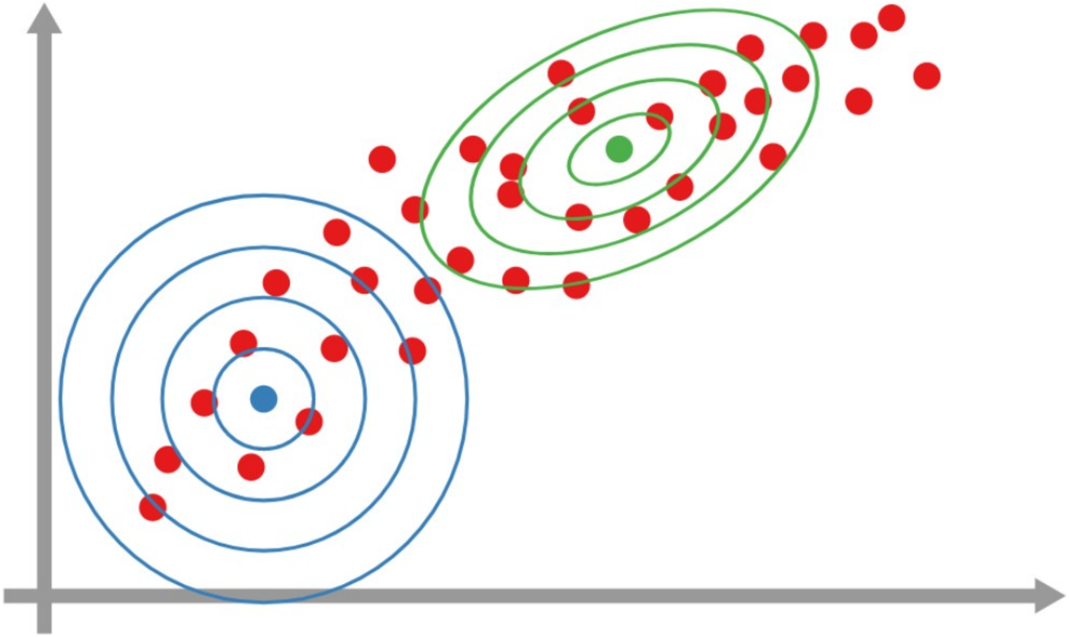
Re-Cap on Distances

- Euclidean “straight line”

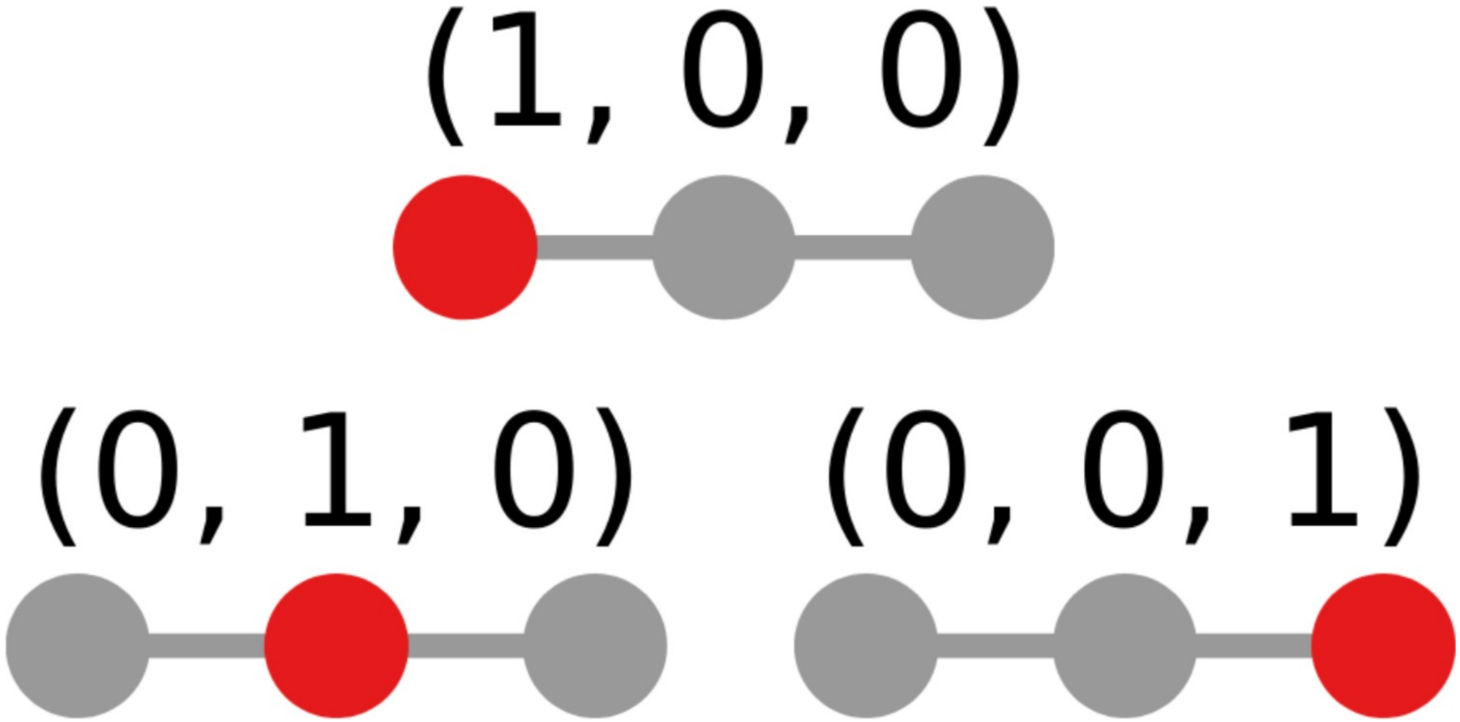
$$\sqrt{(p-q)^T I (p-q)}$$

- Mahalanobis “bendy” space

$$\sqrt{(p-q)^T \operatorname{cov}(p,q)^{-1} (p-q)}$$

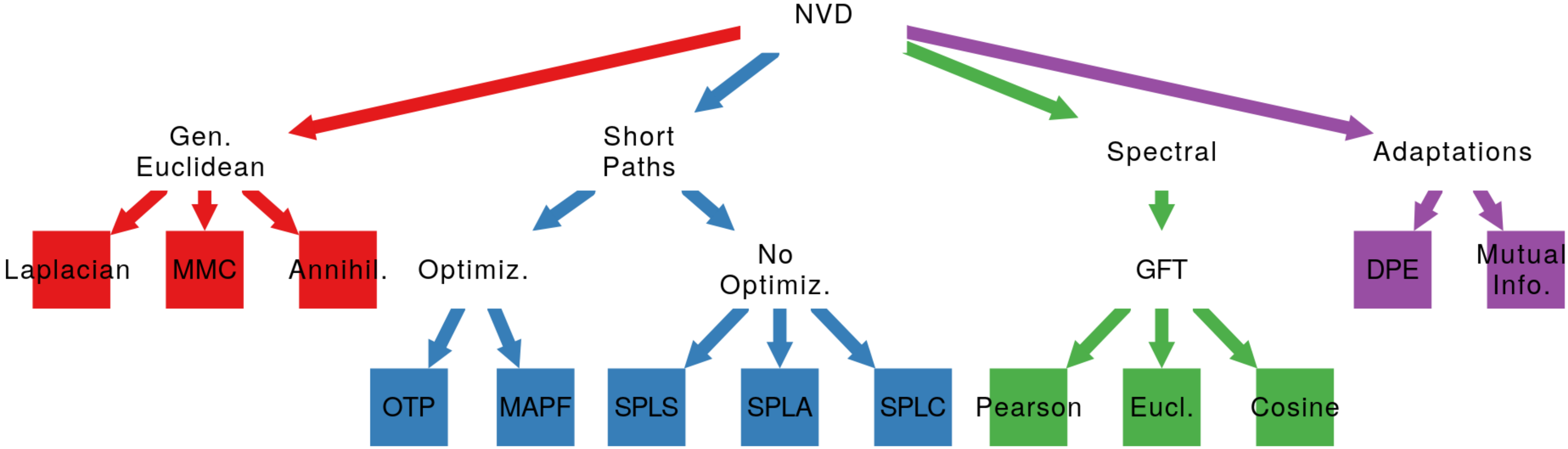


Euclidean vs Network



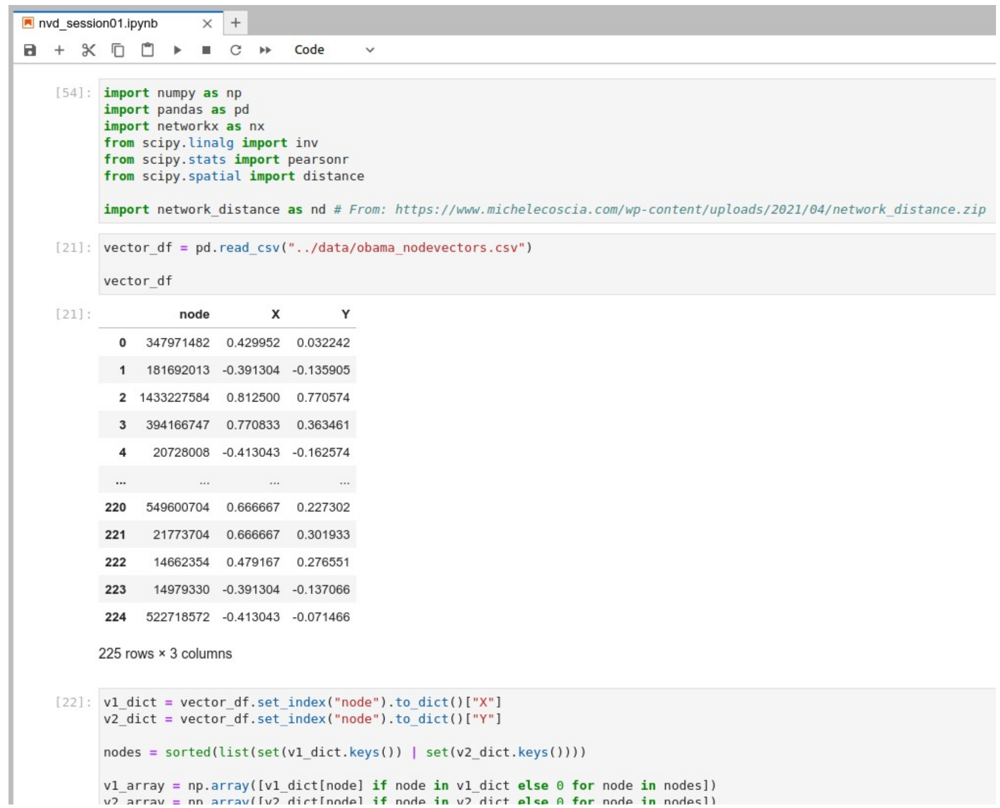
Same Euclidean distances, different network distances
The network “bends” the space, like in Mahalanobis

Current Methods



Tutorial Part #1

- Objectives:
  - Refresher on vector distances
  - Understand I/O of library
  - Set up the data




```python
import numpy as np
import pandas as pd
import networkx as nx
from scipy.linalg import inv
from scipy.stats import pearsonr
from scipy.spatial import distance

import network_distance as nd # From: https://www.michelecoscia.com/wp-content/uploads/2021/04/network_distance.zip
```



```python
vector_df = pd.read_csv("../data/obama_nodevectors.csv")

vector_df
```


	node	X	Y
0	347971482	0.429952	0.032242
1	181692013	-0.391304	-0.135905
2	1433227584	0.812500	0.770574
3	394166747	0.770833	0.363461
4	20728008	-0.413043	-0.162574
...	...	...	...
220	549600704	0.666667	0.227302
221	21773704	0.666667	0.301933
222	14662354	0.479167	0.276551
223	14979330	-0.391304	-0.137066
224	522718572	-0.413043	-0.071466

225 rows × 3 columns

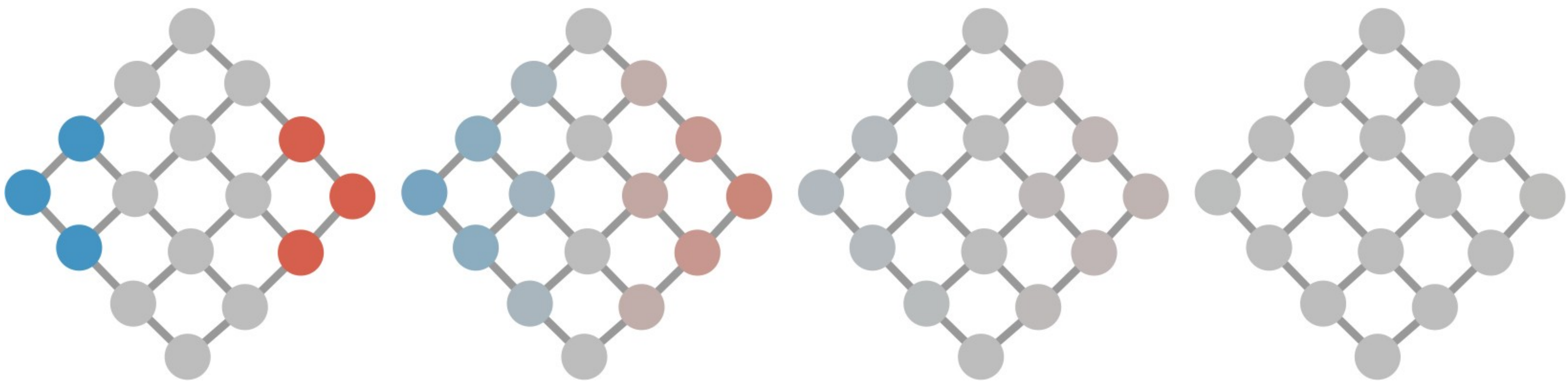

```python
v1_dict = vector_df.set_index("node").to_dict()["X"]
v2_dict = vector_df.set_index("node").to_dict()["Y"]

nodes = sorted(list(set(v1_dict.keys()) | set(v2_dict.keys())))

v1_array = np.array([v1_dict[node] if node in v1_dict else 0 for node in nodes])
v2_array = np.array([v2_dict[node] if node in v2_dict else 0 for node in nodes])
```

Generalized Euclideans

Heat Diffusion



- Imagine each node as a thermometer
- Connected to other thermometers to pass heat
- If we know the heat h at time t
- What would be the heat at t + 1?

$$\frac{\partial h}{\partial t} = -Lh$$

Laplacian

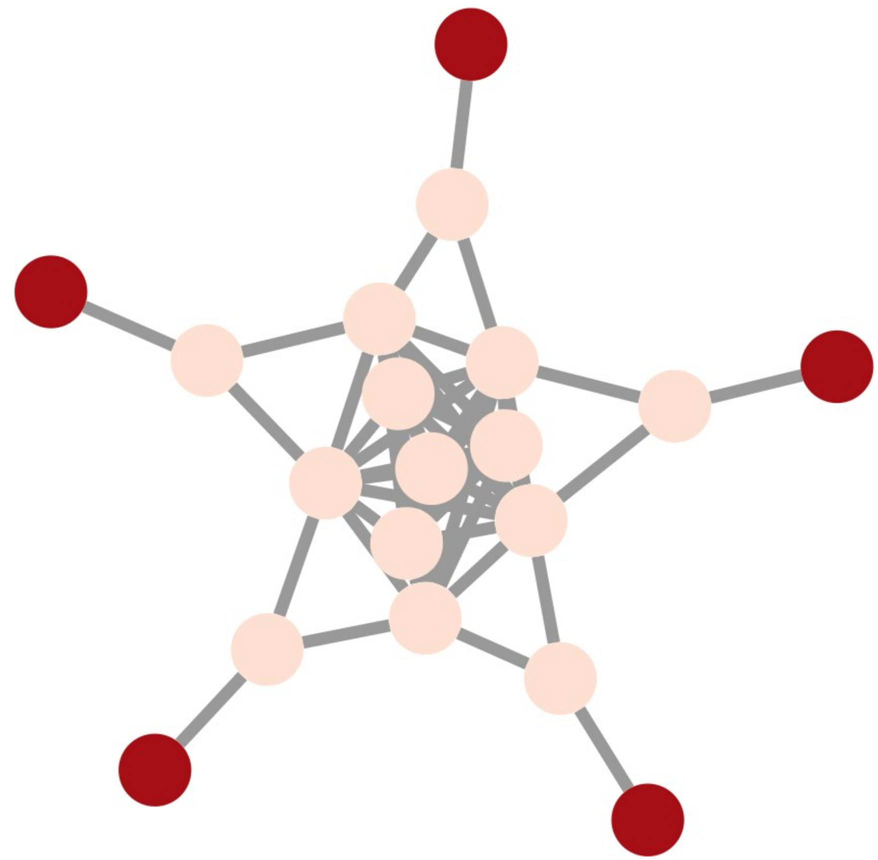
Generalized Euclidean: Laplacian

- L tells us how easy it is to pass heat between nodes
- Which is via random walks
- Thus its inverse gives a sense of distance

$$\sqrt{(p-q)^T L^{-1} (p-q)}$$

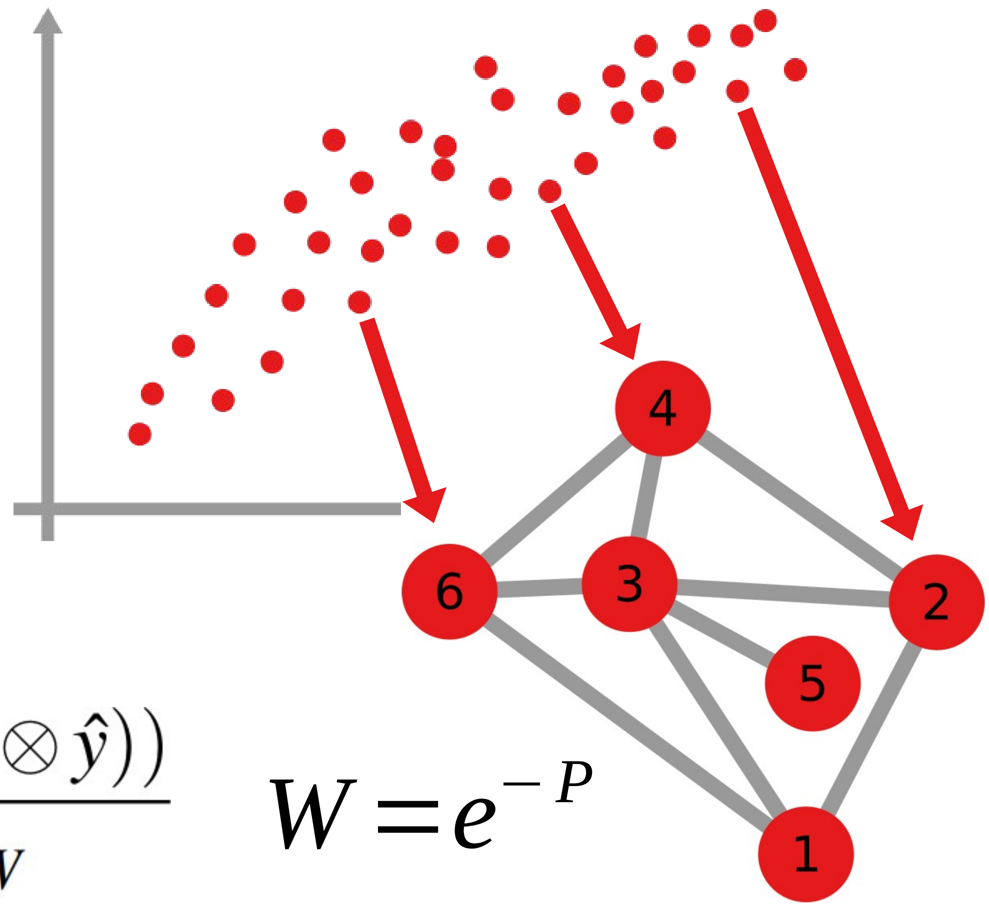
Network Variance

- Not only about distances!
- Vector v: how spread out is on the network?
- Same as variance, now network variance



Network Correlation

- How are two variables related?
- Pearson works in a flat space
- What if we have a network?
- E.g. correlation between age and sharing activity in a social network



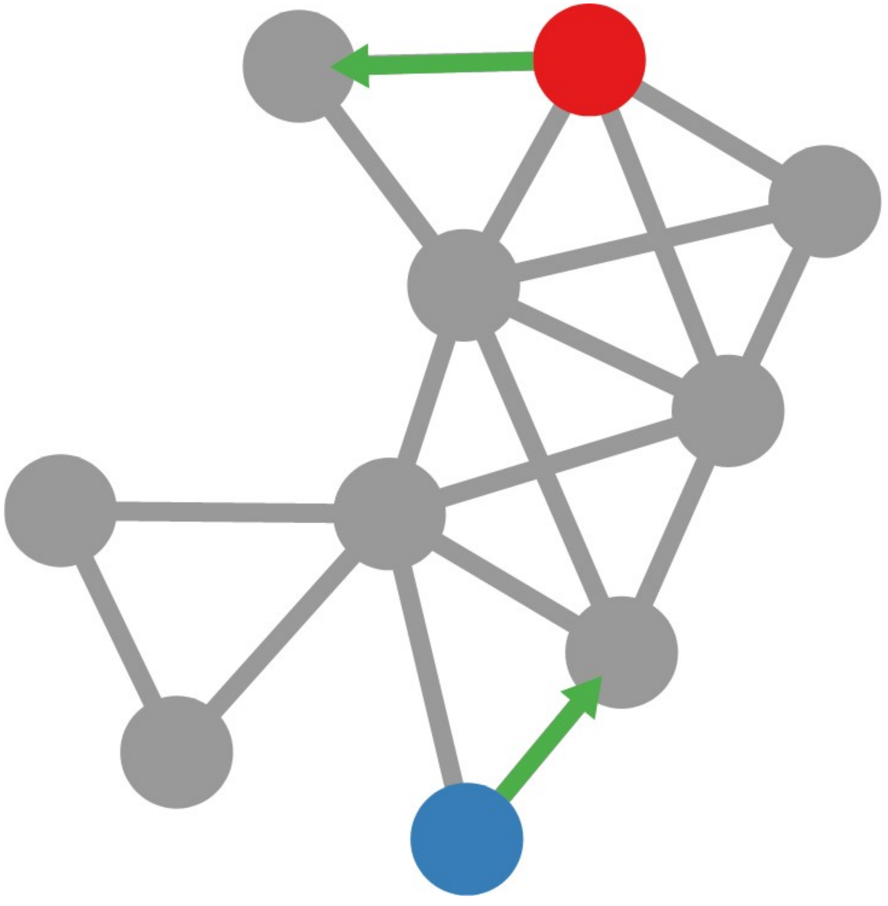
$$\rho_{x,y,G} = \frac{\mathrm{sum}(W \times (\hat{x} \otimes \hat{y}))}{\sigma_{x,W}\sigma_{y,W}}$$

$$W = e^{-P}$$

Random Walks

- If we move randomly, we expect bump into each other
- The later we do so from what we expected, the farther we were
- Z-scores!

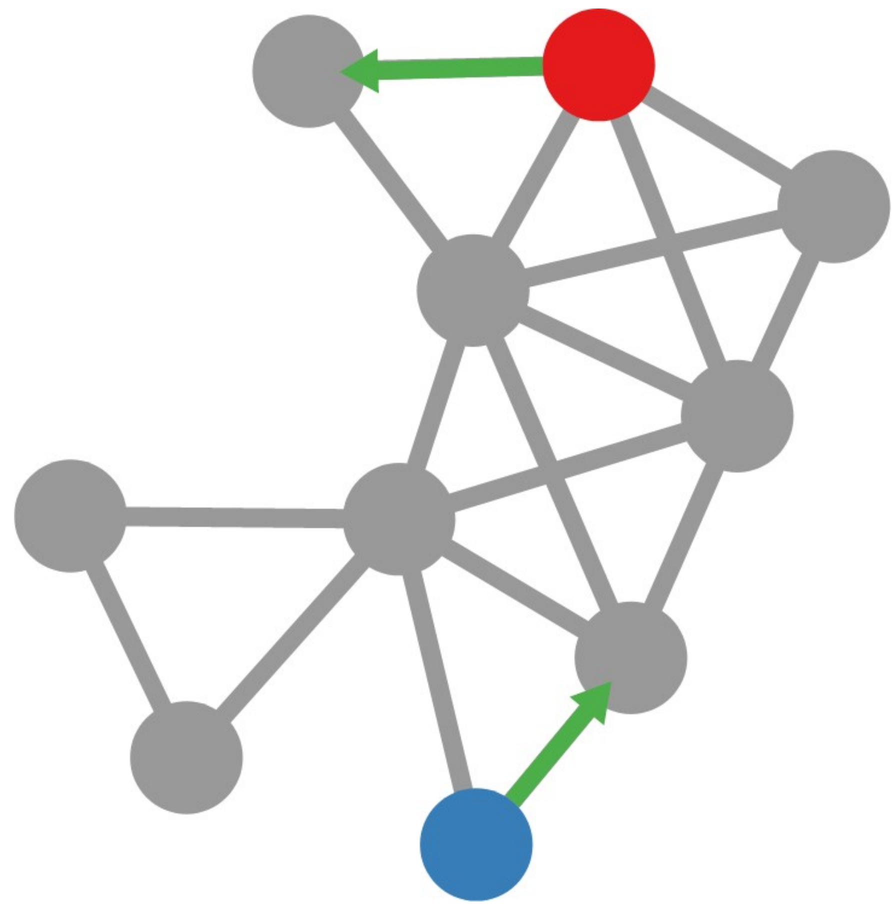
$$\sqrt{(p-q)^T Z^{-1} (p-q)}$$



Random Walks #2

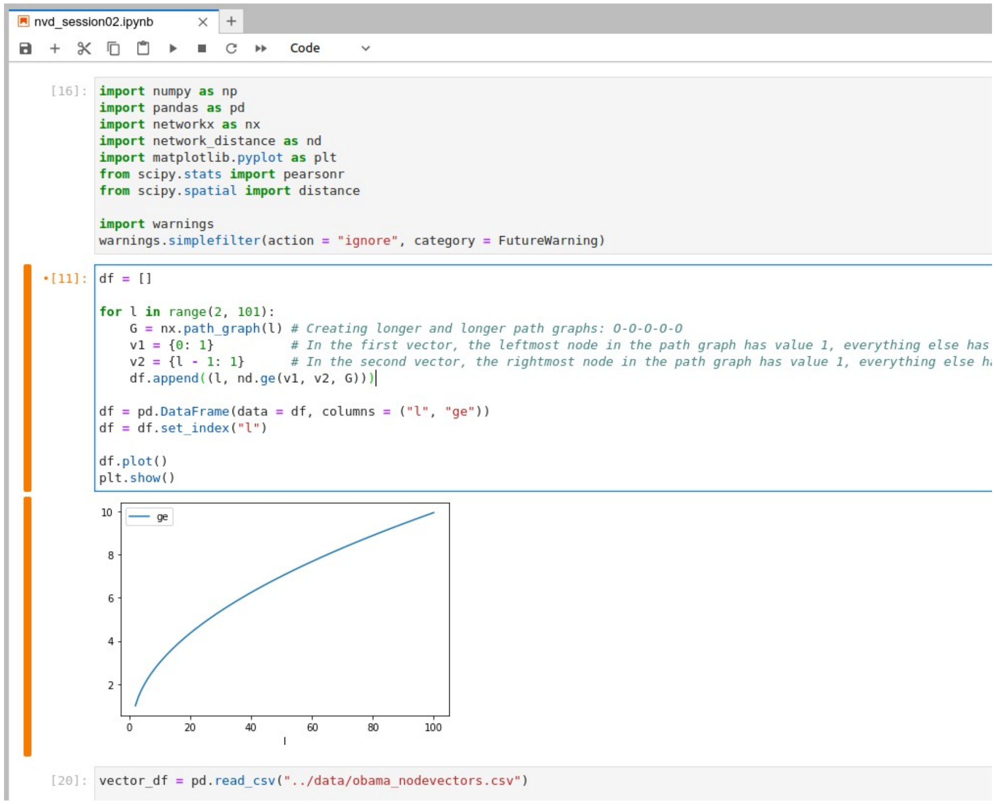
- A^k: p of going from v_1 to v_2 in k steps (RW)
- For which k would our vectors completely overlap each other?

$$\sqrt{(p-q)^T \sum_{k=0}^{\infty} A^k (p-q)}$$



Tutorial Part #2

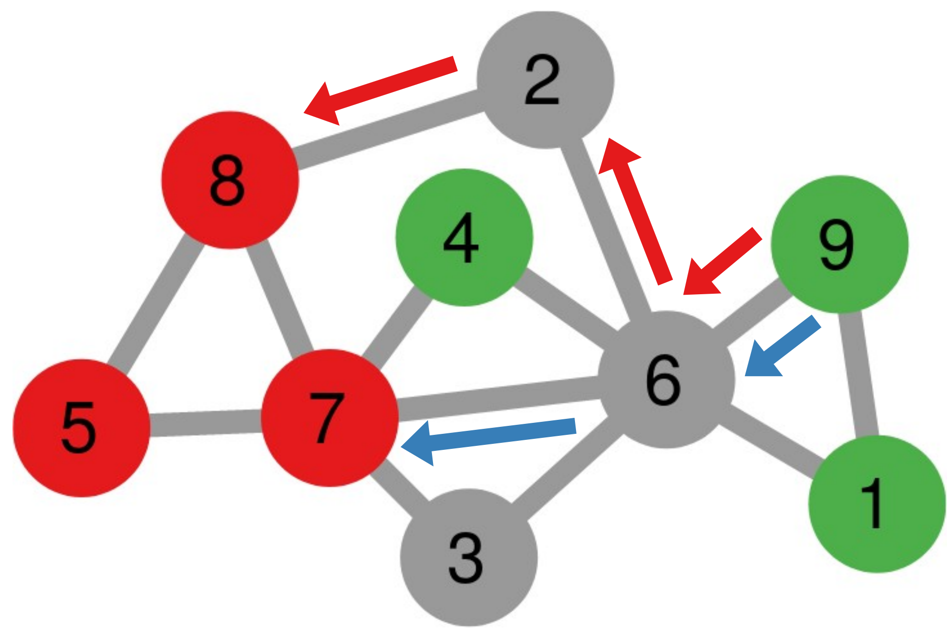
- Objectives:
  - Build an intuition of the measure with a simple toy experiment
  - Calculate Euclidean, variance, and correlation on a network
  - Calculate alternative Euclideans



Shortest Path Distances

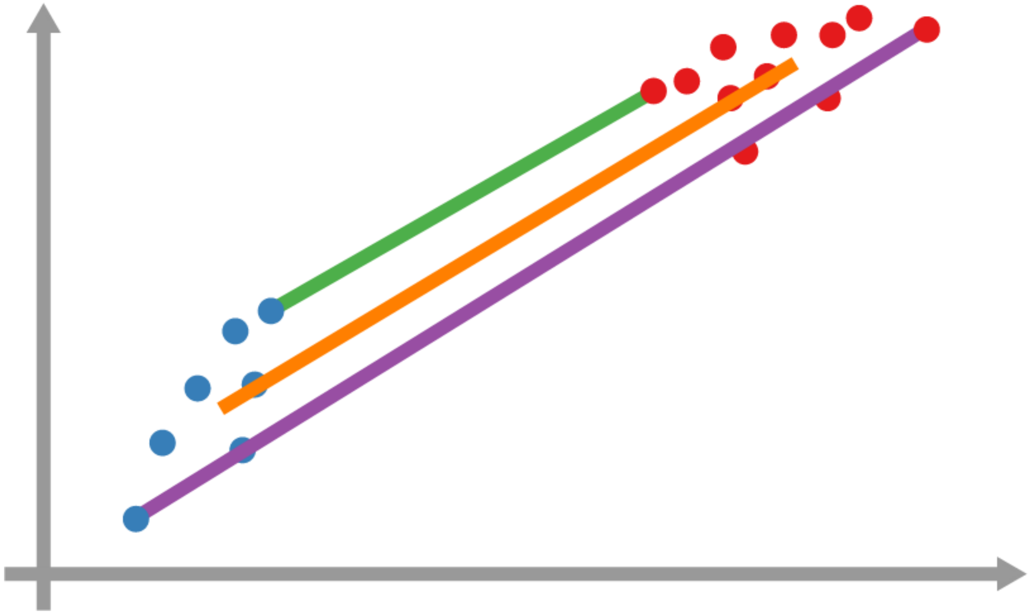
NVD Seems Easy

- Just calculate shortest paths!
- But then: which ones do we choose?



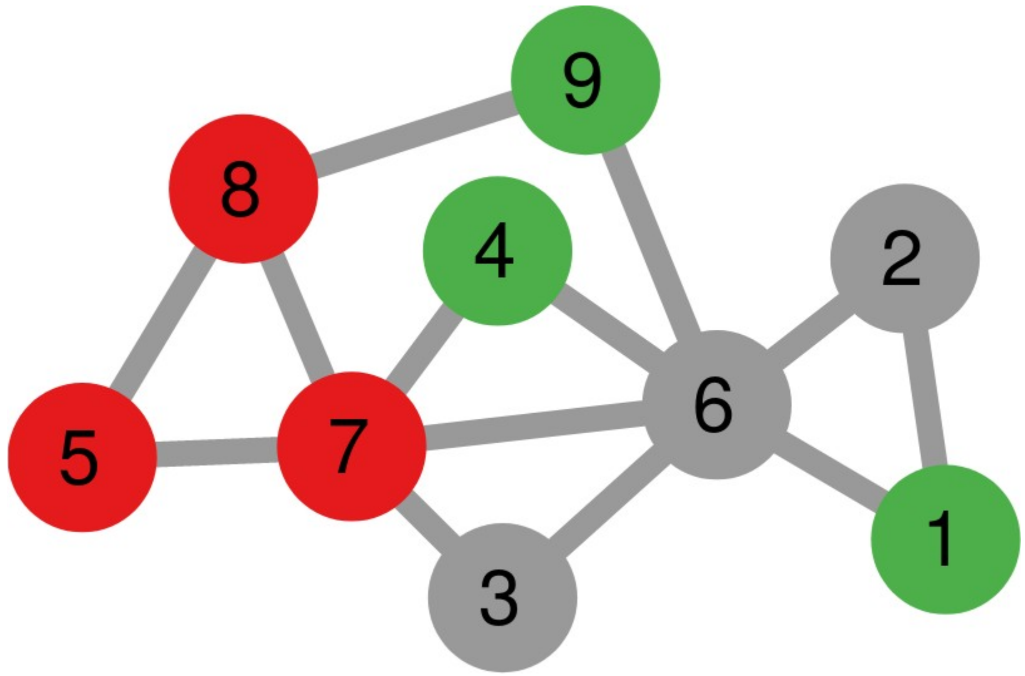
Non-Optimized

- **Single** linkage
  - Always pick the shortest available
- **Complete** Linkage
  - Always pick the longest available
- **Average** Linkage
  - Pick them all and weigh the alternatives



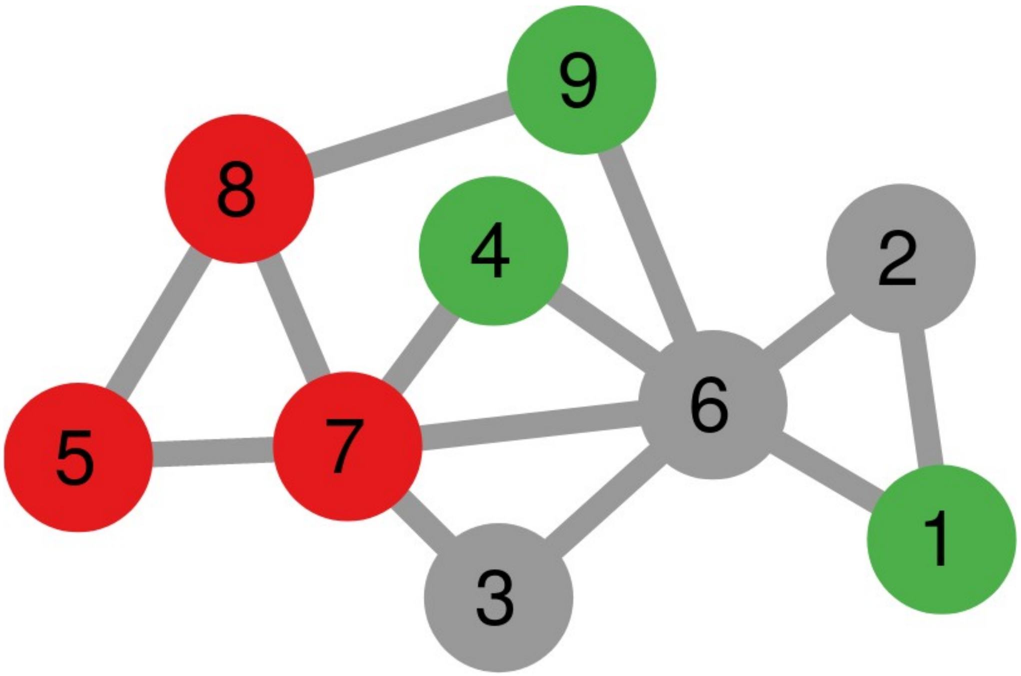
Non-Optimized

- Single Linkage
  - 1 + 1 + 3
- Complete Linkage
  - 3 + 2 + 2
- Average Linkage
  - 18 / 3



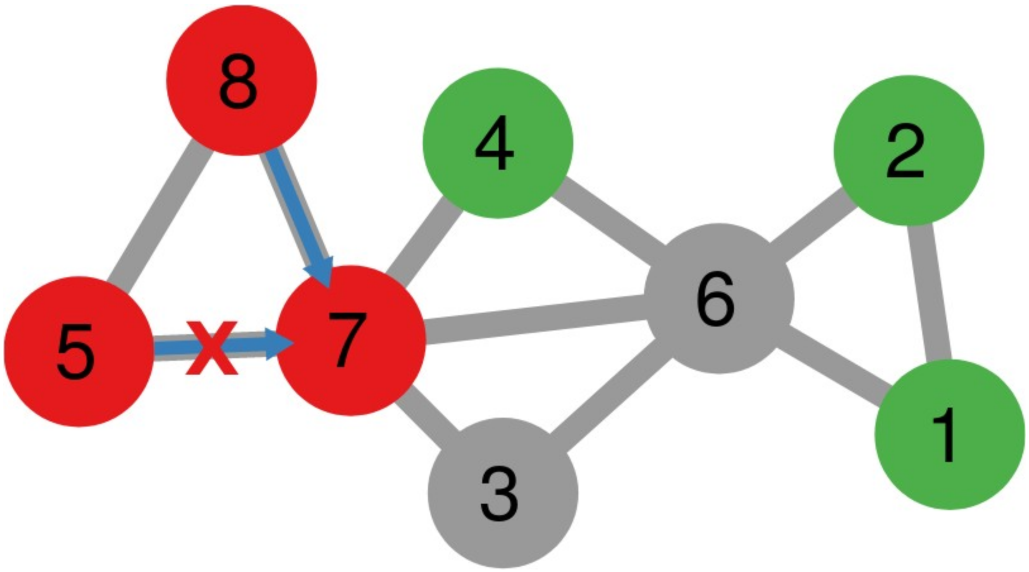
Earth Mover Distance

- Find the best set of paths
- Minimize the cost
- Similar to single linkage



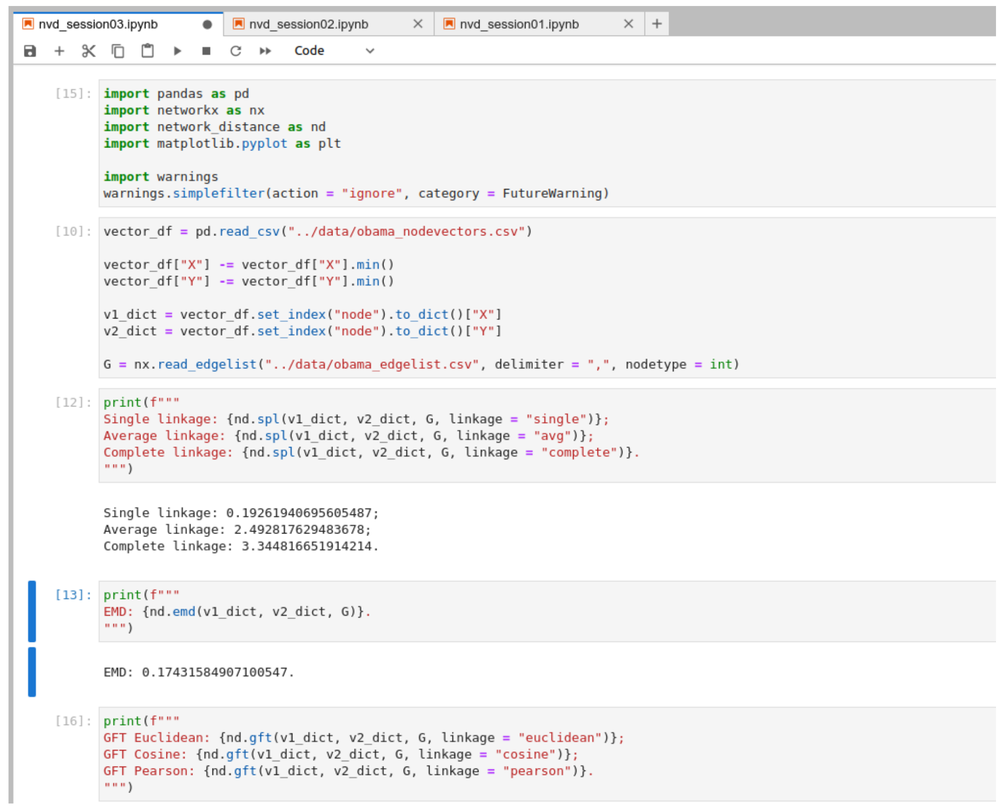
Multi Agent Path Finding

- Similar to EMD
- With constraints
- Nodes & edges have capacity limits
- E.g. cannot use the same link at the same time



Tutorial Part #3

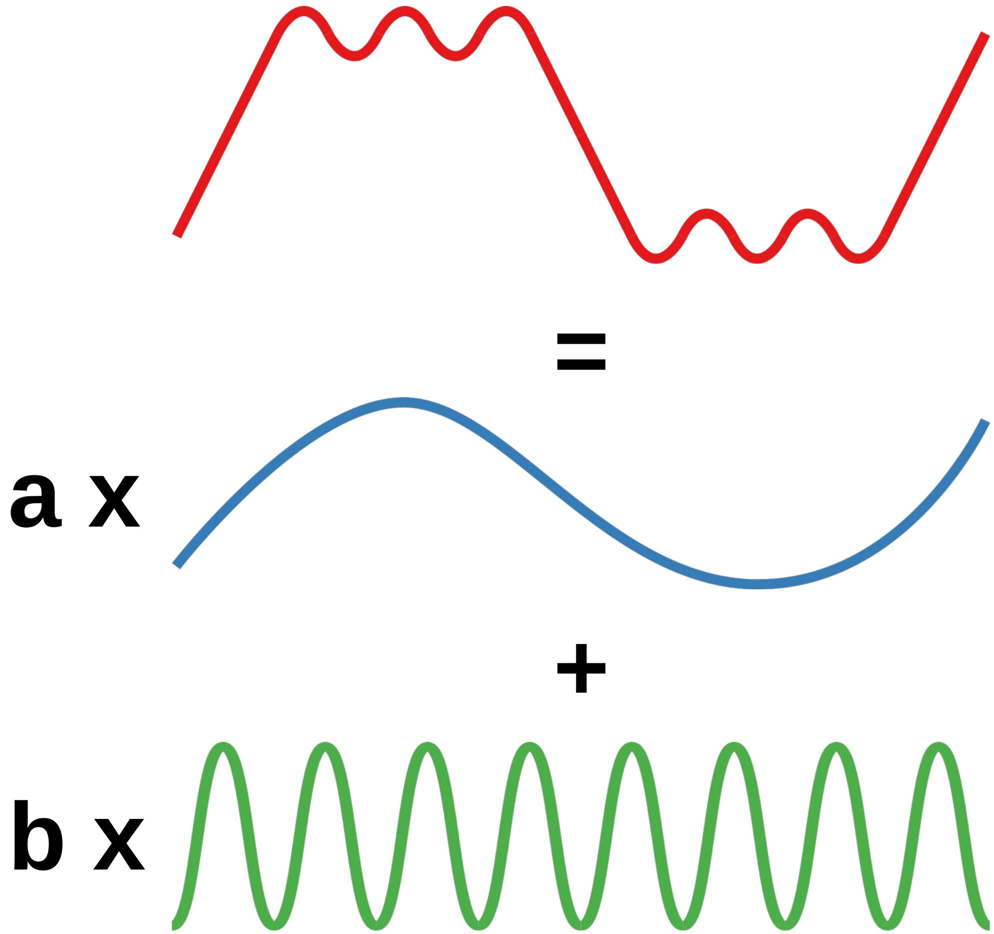
- Objectives:
  - Experiment with different linkage criteria
  - Use optimal EMD solution
  - (Skipping ahead) Test spectral distance



Spectral Methods

Graph Fourier Transform

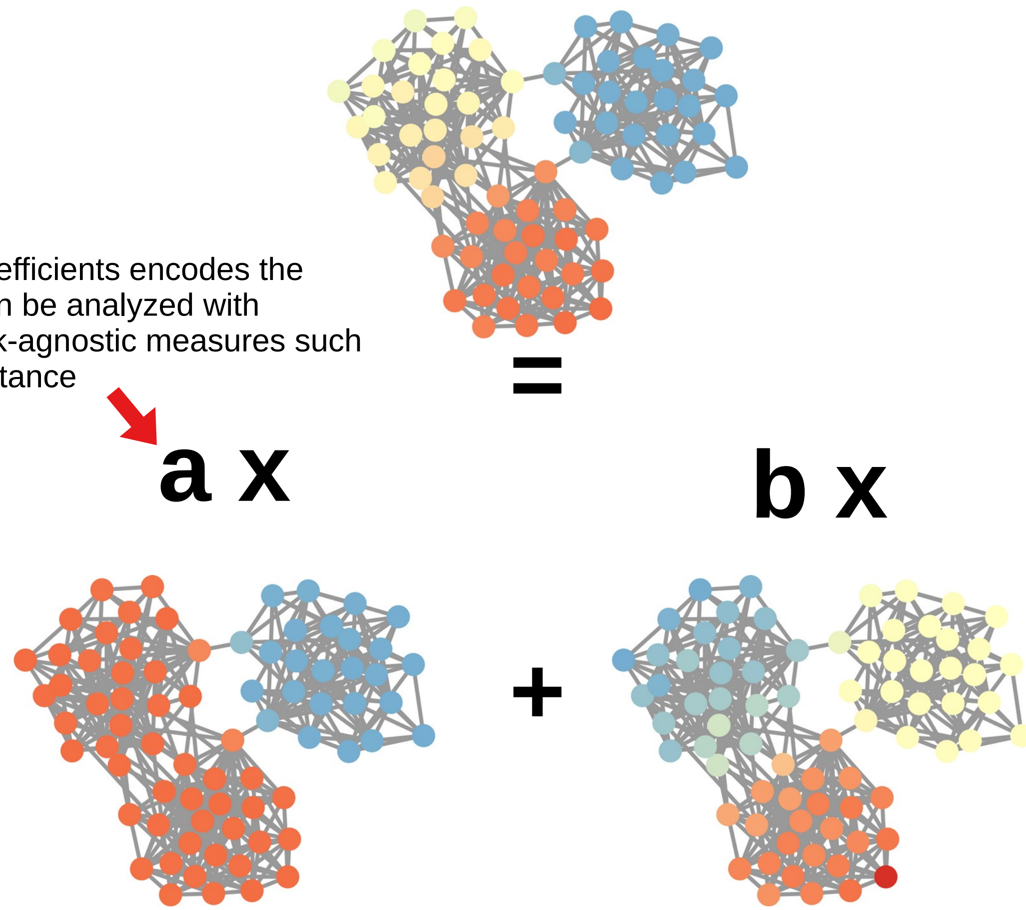
- FT: decompose any signal into its component frequencies



Graph Fourier Transform

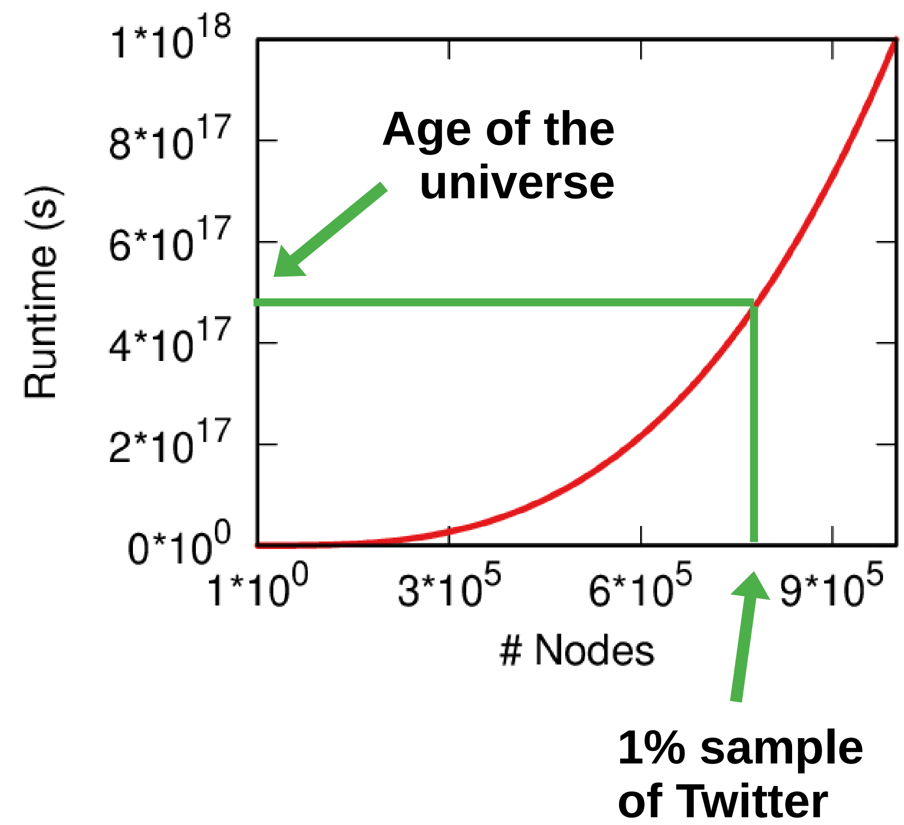
- FT: decompose any signal into its component frequencies
- GFT: same, but the components are the eigenvectors of the graph
- Remember: eigenvectors encode structure

The vector of coefficients encodes the structure and can be analyzed with classical network-agnostic measures such as Euclidean distance



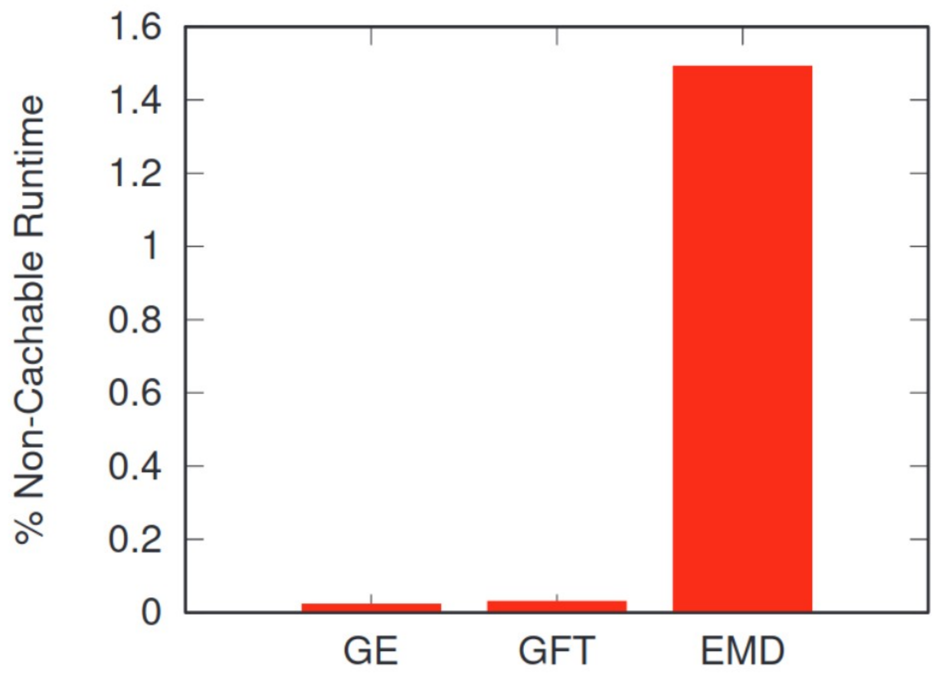
Computational Efficiency

- Expensive steps can be $O(|V|^3)$…
  - Pseudoinverse for GE
  - Shortest paths
  - Eigenvectors of Laplacian



Some Tricks

- Expensive part needs to be done once per network
- You can re-use it for all node vector pairs you have
- E.g. disease spreading on an unchanging social network



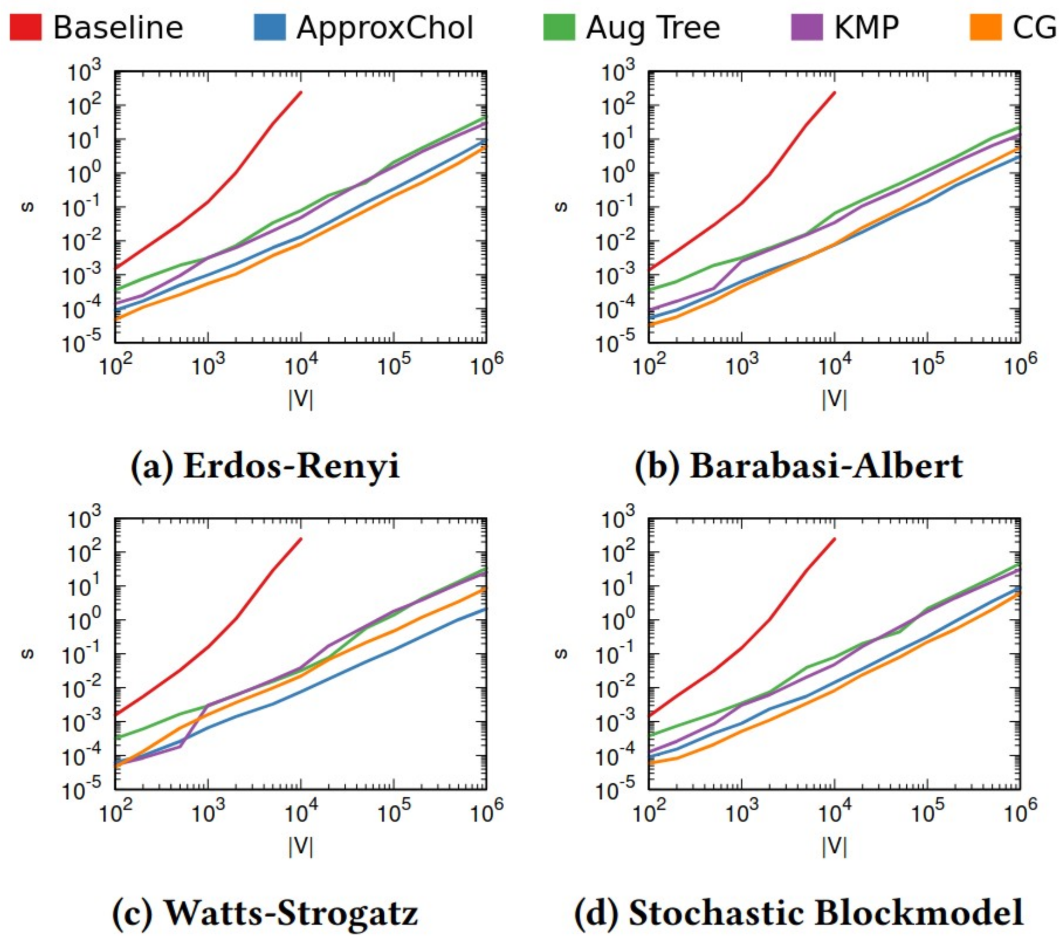
Laplacian Solvers

$$\sqrt{(p-q)^T \boxed{L^{-1}(p-q)}}$$

This part can be calculated efficiently

Without pseudo-inverting the Laplacian!

With Laplacian solvers (in Julia)



Tutorial Parts #4 & #5

- Objectives:
  - Testing runtimes
  - Experimenting with Laplacian solvers
  - Trying out some application scenarios