Problem_3_3_S

November 15, 2021

Names:
1) Klaus Kades (Uni-id: fw448 Matrikelnr.: 3408463)

2) Lucas-Raphael Müller (Uni-id: al413 Matrikelnr.: 3205638)

3) Melanie Schellenberg (Uni-id: qh400 Matrikelnr.: 3146390)

4) Shuhan Xiao (Uni-id: kg410 Matrikelnr.: 3160697)

[4]:
```python
import pandas as pd
import networkx as nx
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
```

1 Assignment 3

1.1 Problem 3-3 Differences between real and random networks

1.1.1 Load data and construct graph

[6]:
```python
data = pd.read_csv('./FAOSTAT_data_10-28-2021.csv')
```

[7]:
```python
data.shape
```

[7]:
```
(17592, 16)
```

[10]:
```python
data = data.fillna('NULL')
print(data.shape)
data = data[data.Flag=='NULL']
print(data.shape)
```

```
(16619, 16)
(16619, 16)
```

[11]:
```python
data.head()
```

[11]:
```
  Domain Code                 Domain  …  Flag Flag Description
0          FT   Forestry Trade Flows  …  NULL    Official data
```

```
1           FT  Forestry Trade Flows  …  NULL    Official data
2           FT  Forestry Trade Flows  …  NULL    Official data
3           FT  Forestry Trade Flows  …  NULL    Official data
4           FT  Forestry Trade Flows  …  NULL    Official data

[5 rows x 16 columns]
```

[ ]:
```python
data['Reporter Countries'].unique()
```

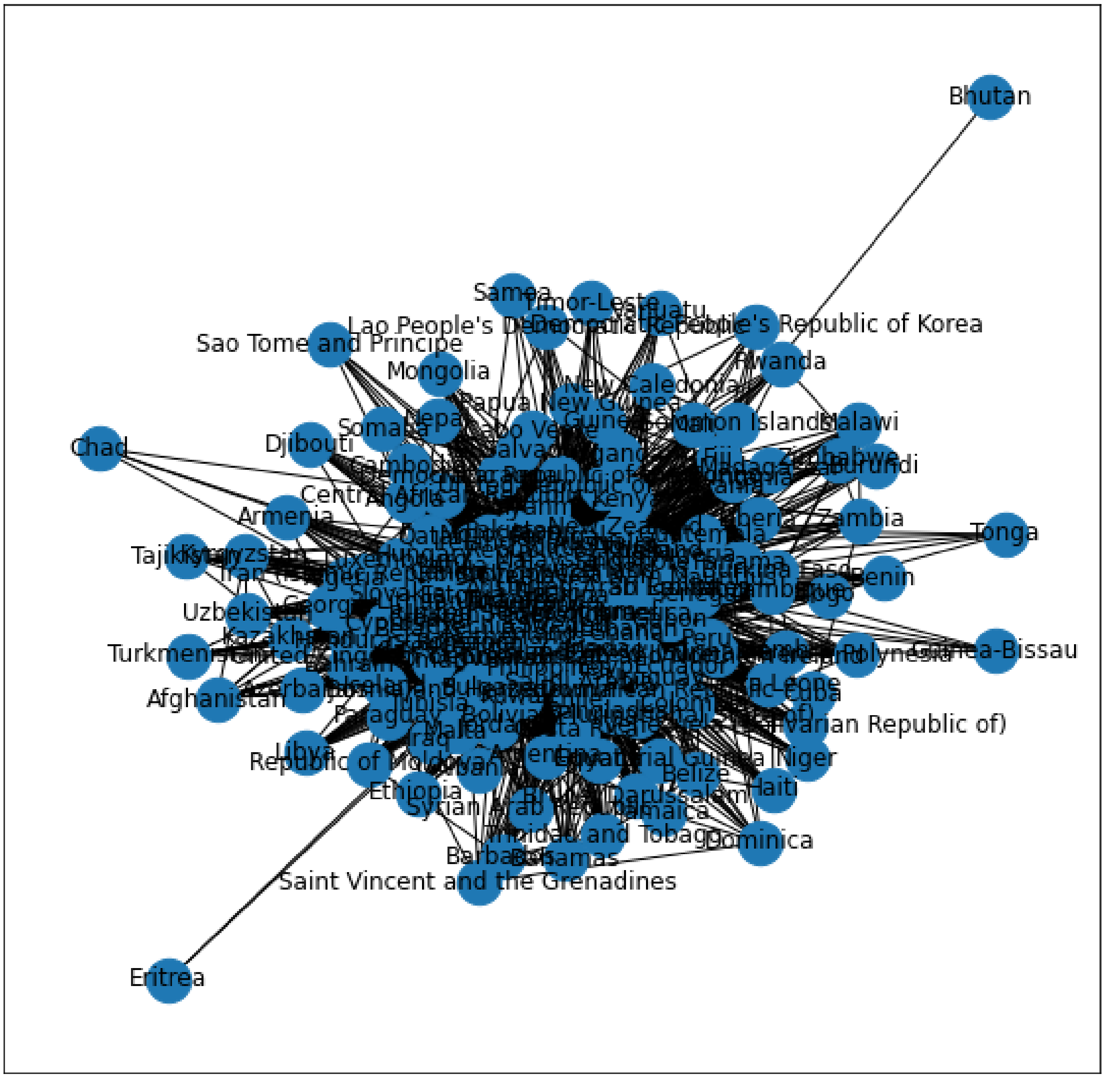
[ ]:
```python
data['Partner Countries'].unique()
```

[15]:
```python
data=data[(data['Partner Countries']!='Others (adjustment)')
         &(data['Partner Countries']!='Total FAO')
         &(data['Partner Countries']!='Unspecified Area')]

data.shape
```

[15]:
```
(15402, 16)
```

[23]:
```python
G = nx.Graph()
G.add_weighted_edges_from(data[["Reporter Countries", "Partner␣
 ↪Countries","Value"]].itertuples(index=False),weight='Value')
plt.figure(figsize=(10,10))
nx.draw_networkx(G, node_size=500)
plt.show()
```



[18]:

```
print(nx.info(G))
```

```
Graph with 168 nodes and 4102 edges
```

1.1.2 1.

extract ego graph G_{France}

[40]:

```
GFr = nx.ego_graph(G,"France")
print(nx.info(GFr))
#ignore direction of edges
GFr = GFr.to_undirected()
```

```
Graph with 122 nodes and 3380 edges
```

Filling out table

N	L	L_{min}	L_{max}	k_{min}	k_{max}

[49]:
```
degrees = [val for (node, val) in GFr.degree()]
N = GFr.number_of_nodes()
table = pd.DataFrame.from_dict({'N': [N],
                                'L': [GFr.number_of_edges()],
                                'Lmin': [N-1],
                                'Lmax': [N*(N-1)/2],
                                'kmin': [min(degrees)],
                                'kmax': [max(degrees)]})
display(table)
```

	N	L	Lmin	Lmax	kmin	kmax
0	122	3380	121	7381.0	4	121

1.1.3 2.

random graph metrics * $<k>=\frac{\sum_i k_i}{N}$

[50]:
```
k = sum(degrees)/N
print(f"<k> = {k}")
```

```
<k> = 55.40983606557377
```

- $p=\frac{<k>}{N-1}$

[56]:
```
p = k/(N-1)
print(f"p = {p}")
```

```
p = 0.4579325294675518
```

- $<L>$

[57]:
```
L = p*N*(N-1)/2
print(f"<L> = {L}")
```

```
<L> = 3380.0
```

- $<k>=\frac{2<L>}{N}$

[53]:
```
k_ = 2*L/N
print(f"<k> = {k_}")
```

```
<k> = 55.40983606557377
```

- $p=\frac{2<L>}{N(N-1)}$

[58]:
```
p_ = 2*L/(N*(N-1))
print(f"p = {p_}")
```

```
p = 0.4579325294675518
```

1.1.4 3.

generate Erdős-Rényi graph realization G_{random}

[60]:
```
Grand = nx.erdos_renyi_graph(n=N, p=p)
print(nx.info(Grand))
```

```
Graph with 122 nodes and 3398 edges
```

visualise adjacency matrices

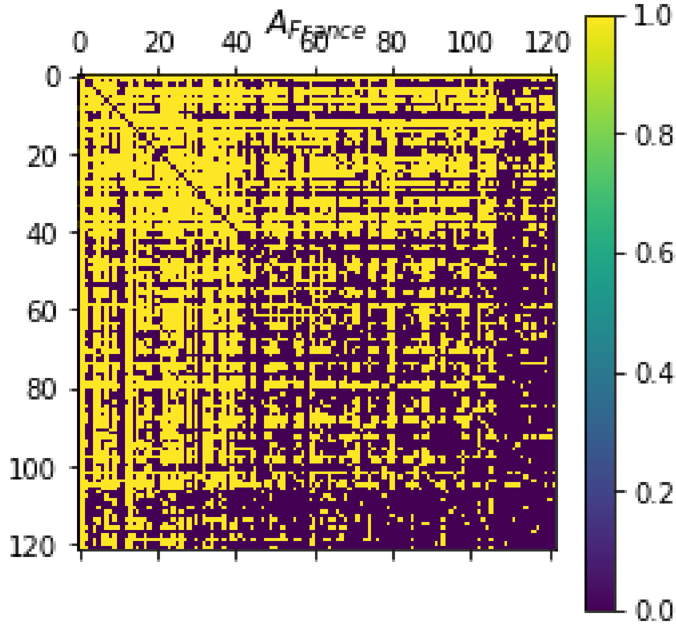
- A_{France}

[85]:
```
AFr = nx.adjacency_matrix(GFr)
print(AFr.todense())
print(AFr.shape)
AFr = AFr.toarray()
plt.figure()
plt.matshow(AFr)
plt.title("$A_{France}$")
plt.colorbar()
```

```
[[0 1 1 … 0 0 0]
 [1 0 1 … 1 1 1]
 [1 1 0 … 1 0 1]
 …
 [0 1 1 … 0 0 0]
 [0 1 0 … 0 0 0]
 [0 1 1 … 0 0 0]]
(122, 122)
```

[85]:
```
<matplotlib.colorbar.Colorbar at 0x7f82e2988590>
```

```
<Figure size 432x288 with 0 Axes>
```



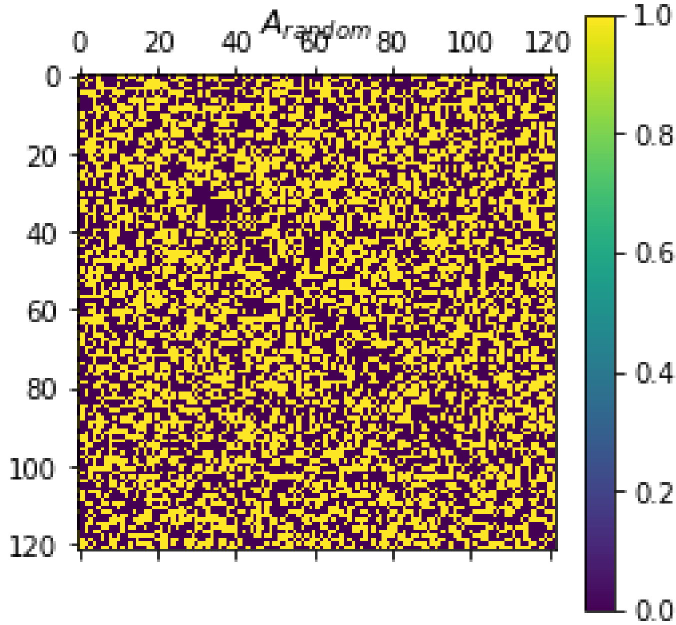
- A_{random}

[82]:
```
Arand = nx.adjacency_matrix(Grand)
print(Arand.todense())
print(Arand.shape)
Arand = Arand.toarray()
plt.figure()
plt.matshow(Arand)
plt.title("$A_{random}$")
plt.colorbar()
```

```
[[0 1 1 … 1 1 1]
 [1 0 1 … 1 1 1]
 [1 1 0 … 0 1 0]
 …
 [1 1 0 … 0 1 0]
 [1 1 1 … 1 0 0]
 [1 1 0 … 0 0 0]]
(122, 122)
```

[82]: <matplotlib.colorbar.Colorbar at 0x7f82e2a0f350>

<Figure size 432x288 with 0 Axes>



visual differences * The zeroth row and column of A_{France} is completely yellow (all elements are 1), which makes sense as all nodes are connected with the node "France" in G_{France}, "France" is a central hub in this case. * Compared to A_{France} where a somewhat grid-like pattern is visible, the elements where the adjacency matrix is 1 (yellow) and 0 (dark blue) are more evenly distributed for A_{random}, no particular pattern is visible. There are more yellow elements in the upper left corner compared to the lower right corner, where gradually more and more dark blue pixels are visible for columns and rows with a higher number. This can be explained by the fact that G_{France} has nodes that are connected with a lot more nodes than others, i.e. other hubs exist, while the nodes of G_{random} are connected with an even probability. Its degree distribution is more even. The gradual change of colors in A_{France} is due to the ordering of row and column numbers.

1.1.5 4.

- histogram of node degrees

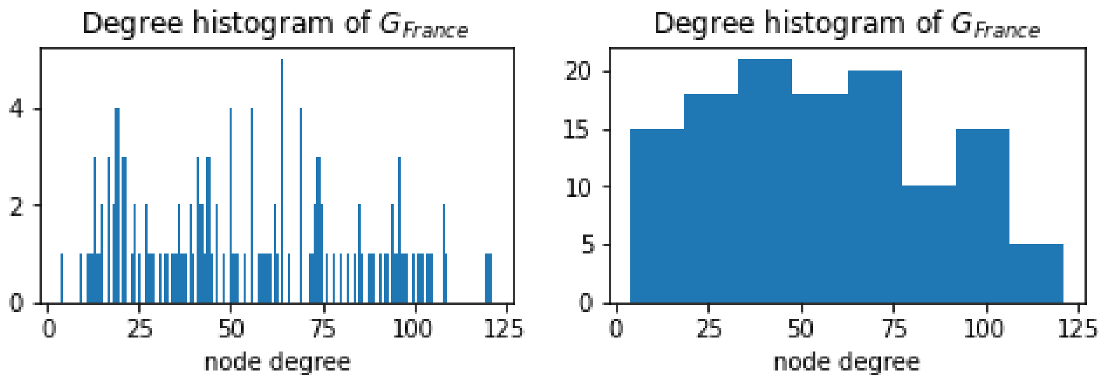
[123]:
```
from scipy.stats import norm
```

[166]:
```
degFr = sorted([d for n, d in GFr.degree()], reverse=True)
print(min(degFr))
print(max(degFr))
plt.figure(figsize=(8,2))
plt.subplot(1,2,1)
plt.bar(*np.unique(degFr, return_counts=True))
plt.title("Degree histogram of $G_{France}$")
plt.xlabel("node degree")
plt.subplot(1,2,2)
```

```
plt.hist(degFr, bins=8)
plt.title("Degree histogram of $G_{France}$")
plt.xlabel("node degree")
```

```
4
121
```

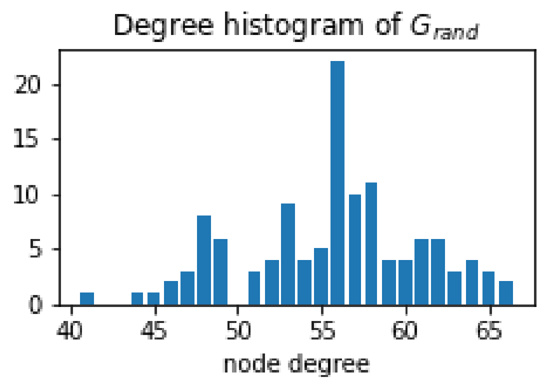
[166]: Text(0.5, 0, 'node degree')

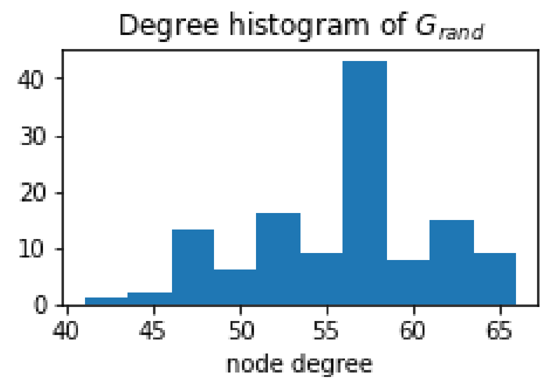



```
[165]: degrand = sorted([d for n, d in Grand.degree()], reverse=True)
print(min(degrand))
print(max(degrand))
plt.figure(figsize=(8,2))
plt.subplot(1,2,1)
plt.bar(*np.unique(degrand, return_counts=True))
plt.title("Degree histogram of $G_{rand}$")
plt.xlabel("node degree")
plt.subplot(1,2,2)
plt.hist(degrand)
plt.title("Degree histogram of $G_{rand}$")
plt.xlabel("node degree")
```

```
41
66
```

[165]: Text(0.5, 0, 'node degree')



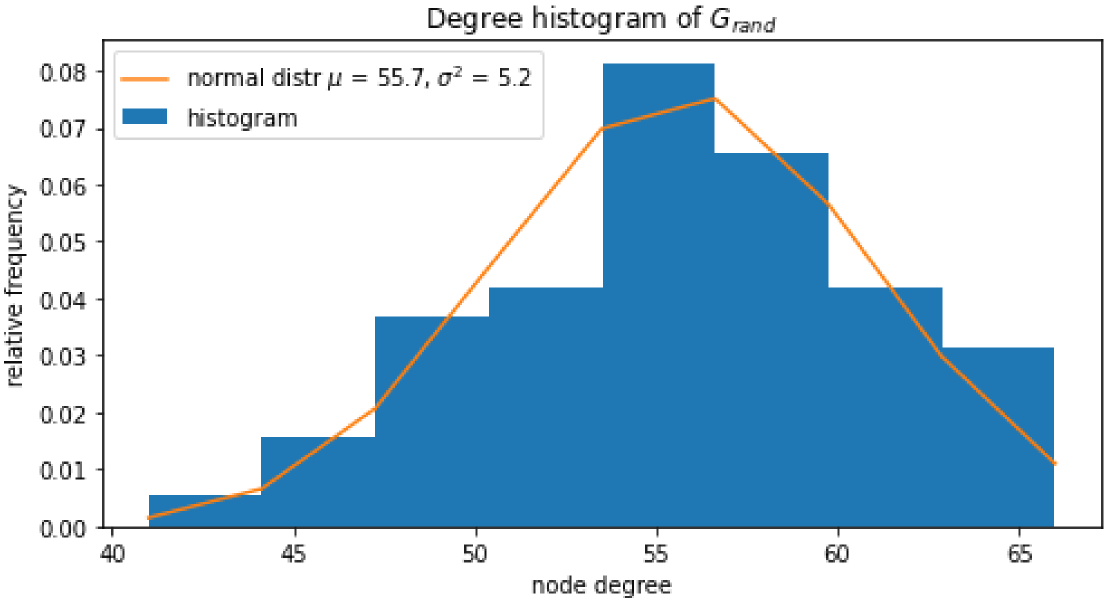


- normal distribution

[152]:
```
# normalise count to frequency
plt.figure(figsize=(8,4))
_, binsrand, _ = plt.hist(degrand, density=1, bins=8)
plt.title("Degree histogram of $G_{rand}$")
plt.ylabel("relative frequency")
plt.xlabel("node degree")
mu, sigma = norm.fit(degrand)
print(mu, sigma)
best_fit_linerand = norm.pdf(binsrand, mu, sigma)
plt.plot(binsrand, best_fit_linerand)
plt.legend([f"normal distr $\mu$ = {mu:.1f}, $\sigma^2$ = {sigma:.
 ↪1f}","histogram"])
```

```
55.704918032786885 5.228685890433883
```

[152]:
```
<matplotlib.legend.Legend at 0x7f82d3236310>
```



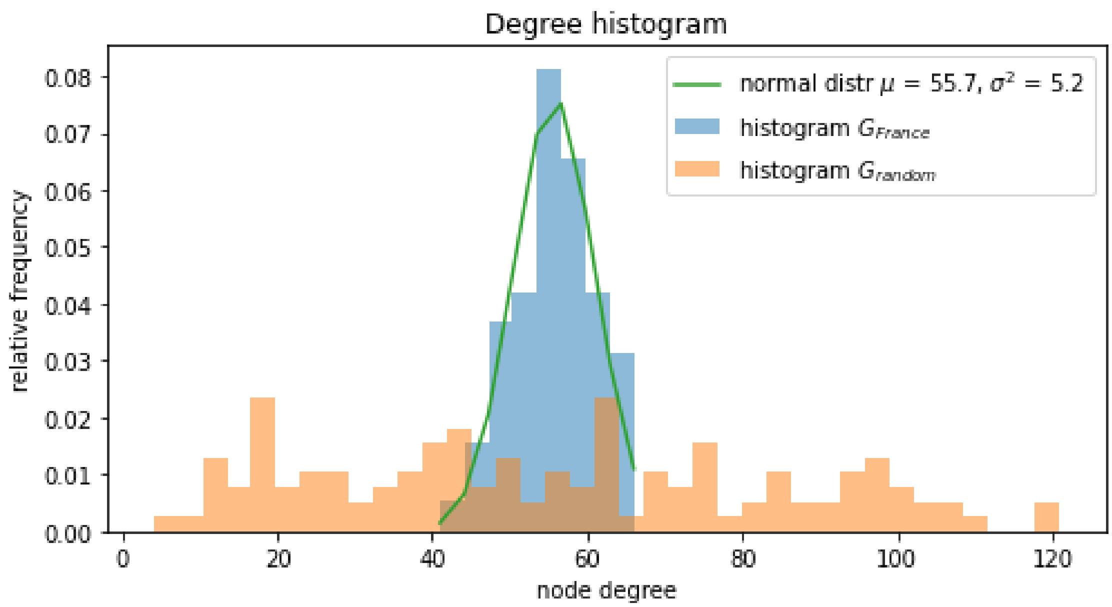
overlay:

[179]:
```
# normalise count to frequency
plt.figure(figsize=(8,4))
bins=8
_, binsrand, _ = plt.hist(degrand, density=1, alpha = 0.5, bins=bins)
binsFr = (max(degFr)-min(degFr))/((max(degrand)-min(degrand))/bins)
plt.hist(degFr, density=1, alpha = 0.5, bins=int(binsFr))


plt.title("Degree histogram")
plt.ylabel("relative frequency")
plt.xlabel("node degree")
mu, sigma = norm.fit(degrand)
print(mu, sigma)
best_fit_linerand = norm.pdf(binsrand, mu, sigma)
plt.plot(binsrand, best_fit_linerand)
plt.legend([f"normal distr $\mu$ = {mu:.1f}, $\sigma^2$ = {sigma:.
↪1f}","histogram $G_{France}$", "histogram $G_{random}$"])
```

```
55.704918032786885 5.228685890433883
```

[179]:
```
<matplotlib.legend.Legend at 0x7f82d27c4850>
```



- Visual differences: For G_{France}, the frequency of nodes generally decreases for increasing node degrees (meaning that there are fewer nodes with very high node degrees), whereas the node degree frequency of G_{random} increases and reaches a peak around $\mu = 55.7$ before decreasing again. The shape of the histogram resembles a Gaussian curve. The range of the histogram of G_{France} is much larger. Its minimun node degree $k_{min} = 4$ is much smaller and maximum node degree $k_{max} = 121$ is much larger than that of G_{random} with $k_{min} = 41$ and $k_{max} = 66$.
- Explanations: As explained in the material from the lecture, random networks as a model for real networks underestimate the number of low degree nodes and and the number of high degree. They do not contain hubs, whereas real networks do. The degree distribution of random networks is a a binomial distribution, but approximates a normal distribution for large N, which is why we can fit the histogram of the random network with a normal distribution.
- Does the Erdős-Rényi ensemble realisation of $G(n, p)$ provide a good approximation for G_{France}? No - as we have seen here both adjacency matrices and degree distributions are too different, the Erdős-Rényi is therefore not a sufficiently good approximation