Networks: structure, evolution & processes

Internet Analytics - Lab 2

Group: *P*

Names:

- *Albert Koppelmaa*
- *Edouard Lacroix*
- *Guillem Pruñonosa Soler*
- *Gaëtan Schwartz*

Instructions

This is a template for part 1 of the lab. Clearly write your answers, comments and interpretations in Markodown cells. Don't forget that you can add $\LaTeX$ equations in these cells. Feel free to add or remove any cell.

Please properly comment your code. Code readability will be considered for grading. To avoid long cells of codes in the notebook, you can also embed long python functions and classes in a separate module. Don't forget to hand in your module if that is the case. In multiple exercises, you are required to come up with your own method to solve various problems. Be creative and clearly motivate and explain your methods. Creativity and clarity will be considered for grading.

2.1 Exploration of real networks

Exercise 2.1

In [1]:
```
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [4]:
```
# we import the graph (ignoring the comments (starts with #))
Data = open('..//data//network1.csv', "r")
Gnet1 = nx.parse_edgelist(Data, comments='#', delimiter=',', create_using=nx.Graph()
                      nodetype=int, data=(('weight', float),))
```

In [5]:
```
nNodes = np.unique(Gnet1.nodes).shape[0]
print(f'The number of nodes of Network1 is: {nNodes}')
nEdges = (np.array(Gnet1.edges)).shape[0]
print(f'The number of edges of Network1 is: {nEdges}')
```

```
The number of nodes of Network1 is: 13033
The number of edges of Network1 is: 18584
```

Exercise 2.2: Node degree distribution

In [6]:
```
ds = sorted([d for n, d in Gnet1.degree()])
```

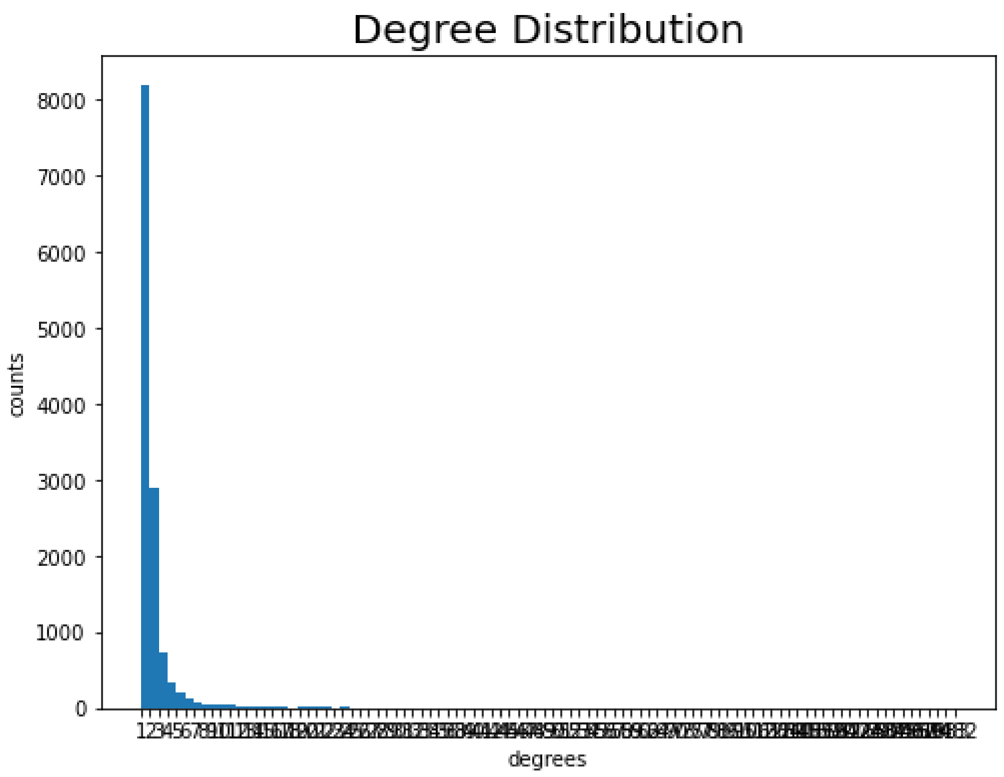
In [7]:
```
np.bincount(ds)
```

Out[7]:
```
array([   0, 8192, 2889, ...,    0,    0,    1], dtype=int64)
```

In [8]:
```
ds_s = [str(d) for d in ds]
```

1-. The most useful plot for this task is the histogram because it represents well the degree distribution, taking in account the number of nodes with every degree.

In [9]:
```
plt.figure(figsize=(8,6))
plt.hist(ds_s,bins=len(np.unique(ds)))
plt.title("Degree Distribution", fontsize=20)
plt.xlabel("degrees", fontsize=10)
plt.ylabel("counts", fontsize=10)
# Display the result
plt.show()
```



In [10]:
```
mean = np.mean(ds)
mean
```

Out[10]:
```
2.8518376429064682
```

In [11]:
```
# Pareto parameters
xm = np.min(ds) # == 1
b = mean / (mean - 1) # mean = (b * xm) / (b - 1)
b
```

Out[11]:
```
1.5400041433602651
```

In [12]:
```
print('Some of the properties of the degree distribution are:')
print(f'- mean = {mean}')
print(f'- min value degree = {np.min(ds)}')
print(f'- max value degree = {np.max(ds)}')
```

```
Some of the properties of the degree distribution are:
- mean = 2.8518376429064682
- min value degree = 1
- max value degree = 1482
```

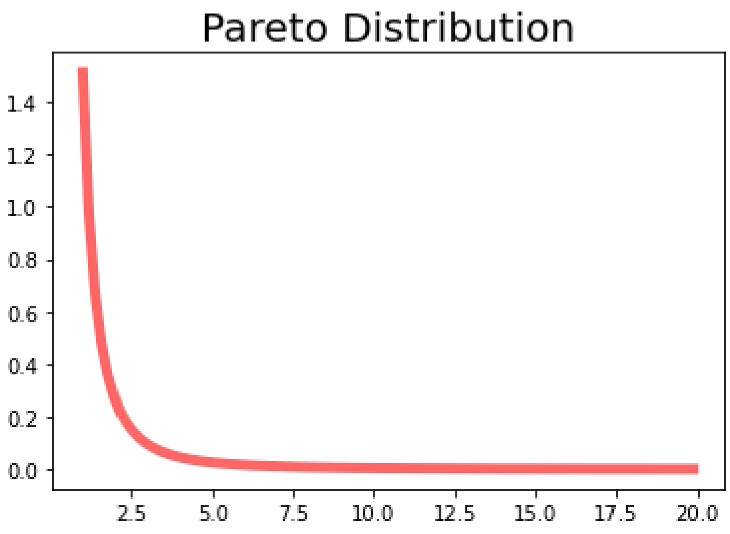
In [13]:
```
from scipy.stats import pareto
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1,1)
# mean, var, skew, kurt = pareto.stats(b, moments='mvsk')

plt.title("Pareto Distribution", fontsize=20)

x = np.linspace(pareto.ppf(0.01, b),
                pareto.ppf(0.99, b), 100)

ax.plot(x, pareto.pdf(x, b),
       'r-', lw=5, alpha=0.6, label='pareto pdf')
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x23d28e506d0>]
```



The degree of the Network1 Graph follows a Pareto(β,xmin) distribution with β = 1.54 and xmin = 1, that it is the minimum degree value.

Both graphs have a very similar shape. We have estimated β given the mean of the degree distribution.

Exercise 2.3: Giant component

In [14]:
```
ncc = nx.number_connected_components(Gnet1)
ncc
```

Out[14]:
```
192
```

The number of connected components is 192.

In [15]:
```
sorted([len(cc) for cc in nx.connected_components(Gnet1)],reverse=True)[:5]
```

Out[15]:
```
[12576, 16, 7, 6, 5]
```

There is one giant component and his size is 12576 nodes.

Exercise 2.4: Short paths & Small-worlds

In [16]:
```
connected_components = sorted([cc for cc in nx.connected_components(Gnet1)])
G0 = Gnet1.subgraph(connected_components[0]) # G0 = giant component
```

In [17]:
```
len(G0.nodes)
```

Out[17]:
```
12576
```

In [18]:
```
[1,2,3,4,5,6,7][-2:]
```

Out[18]:
```
[6, 7]
```

In [22]:
```
nodesToVisit
```

Out[22]:
```
array([ 3748,  4162,  2126, ...,   221, 12986,   222])
```

In [20]:
```
distances = []
np.random.seed(0)
nodesToVisit = np.random.permutation(list(Gnet1.nodes))[:2000]
assert len(nodesToVisit[:1000]) == len(nodesToVisit[-1000:])

for nA in nodesToVisit[:1000]:
    for nB in nodesToVisit[-1000:]:
        if not nA == nB:
            try:
                dist = nx.shortest_path_length(Gnet1,source=nA,target=nB)
                distances.append(dist)
            except:
                pass
```

In [21]:
```
np.mean(distances)
```

Out[21]:
```
4.155957541235
```

The network is indeed a small world as we have a node distance between them of 4.156. So, our network is compact.
In order to calculate this, we have chosen two groups of 1000 random nodes each one from the network and we have compute the shortest_path between every node of each group. We have also had to manage the exception that occurs in case there is no path between two nodes. Another option would have been to compute the mean only in one connected component instead of in the whole graph to guarantee there is always a path between 2 nodes.

Exercise 2.5: Network comparison

In [2]:
```
Data = open('..//data//network2.csv', "r")
Gnet2 = nx.parse_edgelist(Data, comments='#', delimiter=',', create_using=nx.Graph()
                          nodetype=int, data=(('weight', float),))
```

In [3]:
```
nNodes = np.unique(Gnet2.nodes).shape[0]
print(f'The number of nodes of Network2 is: {nNodes}')
nEdges = (np.array(Gnet2.edges)).shape[0]
print(f'The number of edges of Network2 is: {nEdges}')
```

```
The number of nodes of Network2 is: 26542
The number of edges of Network2 is: 43516
```

This (Network2) graph is much bigger than the previous one (Network1).

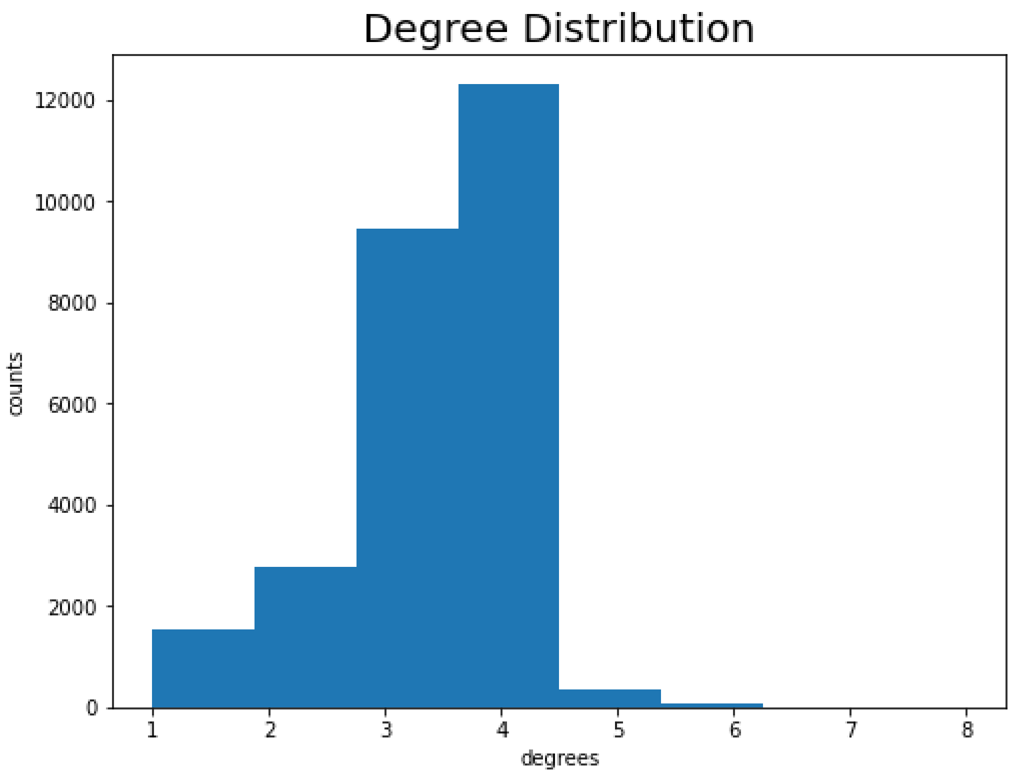
In [4]:
```
ds = sorted([d for n, d in Gnet2.degree()])
```

In [5]:
```
np.bincount(ds)
```

Out[5]:
```
array([    0,  1556,  2782,  9455, 12305,   347,    88,     8,     1],
      dtype=int64)
```

In [6]:
```
ds_s = [str(d) for d in ds]
```

In [7]:
```
plt.figure(figsize=(8,6))
plt.hist(ds_s,bins=len(set(ds)))
plt.title("Degree Distribution", fontsize=20)
plt.xlabel("degrees", fontsize=10)
plt.ylabel("counts", fontsize=10)
# Display the result
plt.show()
```



In Network2 the degree is distributed among all nodes. So, there is no node that is super connected to other nodes.

In [8]:
```
mean = np.mean(ds)
```

In [9]:
```
print('Some of the properties of the degree distribution are:')
print(f'- mean = {mean}')
print(f'- min value = {np.min(ds)}')
print(f'- max value = {np.max(ds)}')
```

```
Some of the properties of the degree distribution are:
- mean = 3.2790294627383014
- min value = 1
- max value = 8
```

The average degree of Network2 is slighly higher than the Network1 avg degree, but with another distribution.

In [10]:
```
ncc = nx.number_connected_components(Gnet2)
ncc
```

Out[10]:
```
5
```

In [11]:
```
connected_components = [cc for cc in nx.connected_components(Gnet2)]
print([len(cc) for cc in connected_components])
big_cc = np.argmax([len(cc) for cc in connected_components])
G0 = Gnet2.subgraph(connected_components[big_cc]) # G0 = giant component
print(big_cc)
```

```
[26481, 24, 30, 2, 5]
0
```

The number of connected components is 5. So this graph has a bigger giant component and less small components, so the graph is more compacted. Here we show the length of each connected component.

In [14]:
```
distances = []
np.random.seed(0)
nodesToVisit = np.random.permutation(list(Gnet2.nodes))

for nA in nodesToVisit[:100]:
    i += 1
    for nB in nodesToVisit[-100:]:
        if nA != nB:
            try:
                dist = nx.shortest_path_length(Gnet2,source=nA,target=nB)
                distances.append(dist)
            except:
                pass
```

In [15]:
```
np.mean(distances)
```

Out[15]:
```
100.14727272727272
```

In this case the network is very sparsed, because the mean distance from two nodes is 100.
In order to calculate this, we have chosen two groups of 100 random nodes (after some trials, we have decided it is a reasonable value) each one from the network and we have compute the shortest_path between every node of each group. We have also had to manage the exception that occurs in case there is no path between two nodes. Another option would have been to compute the mean only in one connected component instead of in the whole graph to guarantee there is always a path between 2 nodes.

Exercise 2.6: Network identification

Considering the degree distributions of both graphs, we have concluded that the first network is related to the Internet subnetwork of routers due to the number of connections of the major router (+8000 connections), and also due to the high compactness (short paths) of the network that suits with the idea of a Internet subnetwork, that you need few jumps from one node to another.

On the other hand, the other degree distribution (network2) is related with the network of roads around NYC, as the maximum of roads intersections is 8. We have also concluded this because of the spread of the second network, that it makes sense to the structure of a city such big as NYC.