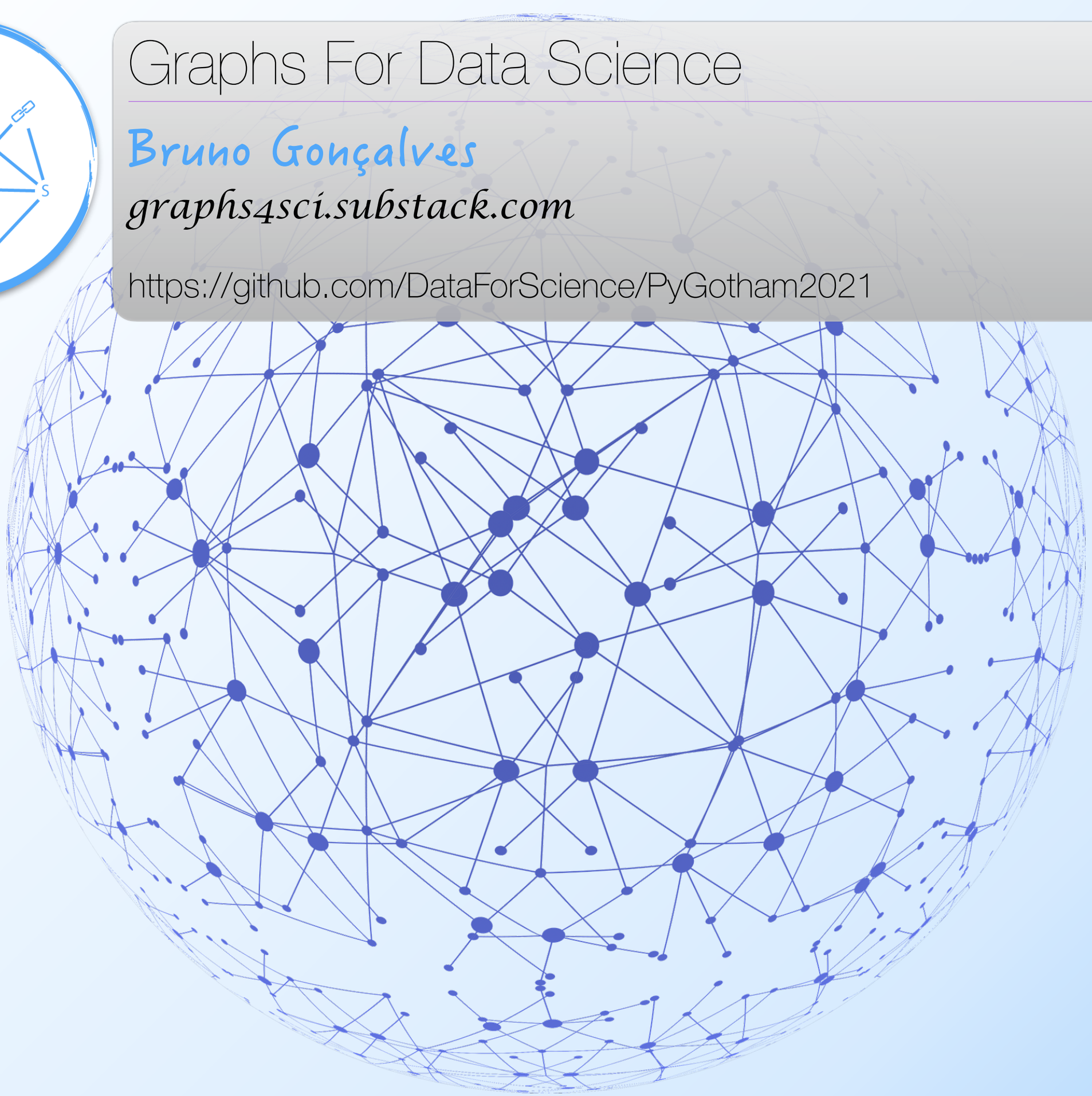
Graphs For Data Science

Bruno Gonçalves

graphs4sci.substack.com

https://github.com/DataForScience/PyGotham2021



Who Am I?

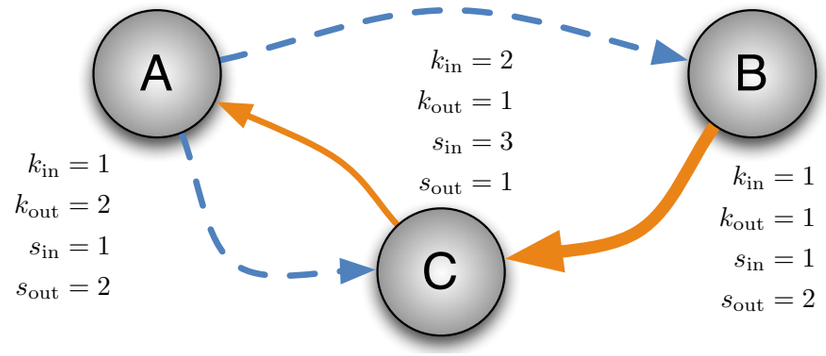
- Name: Bruno Gonçalves
- Work: Senior Data Scientist
- Background: Physics (PhD) and Computer Science (MS)
- Experience: 10+ years using large scale datasets to analyze individual human behavior.
- Website: www.data4sci.com
- Twitter: @data4sci
- Email: bgoncalves@data4sci.com

Table of Contents

1. Graphs 101

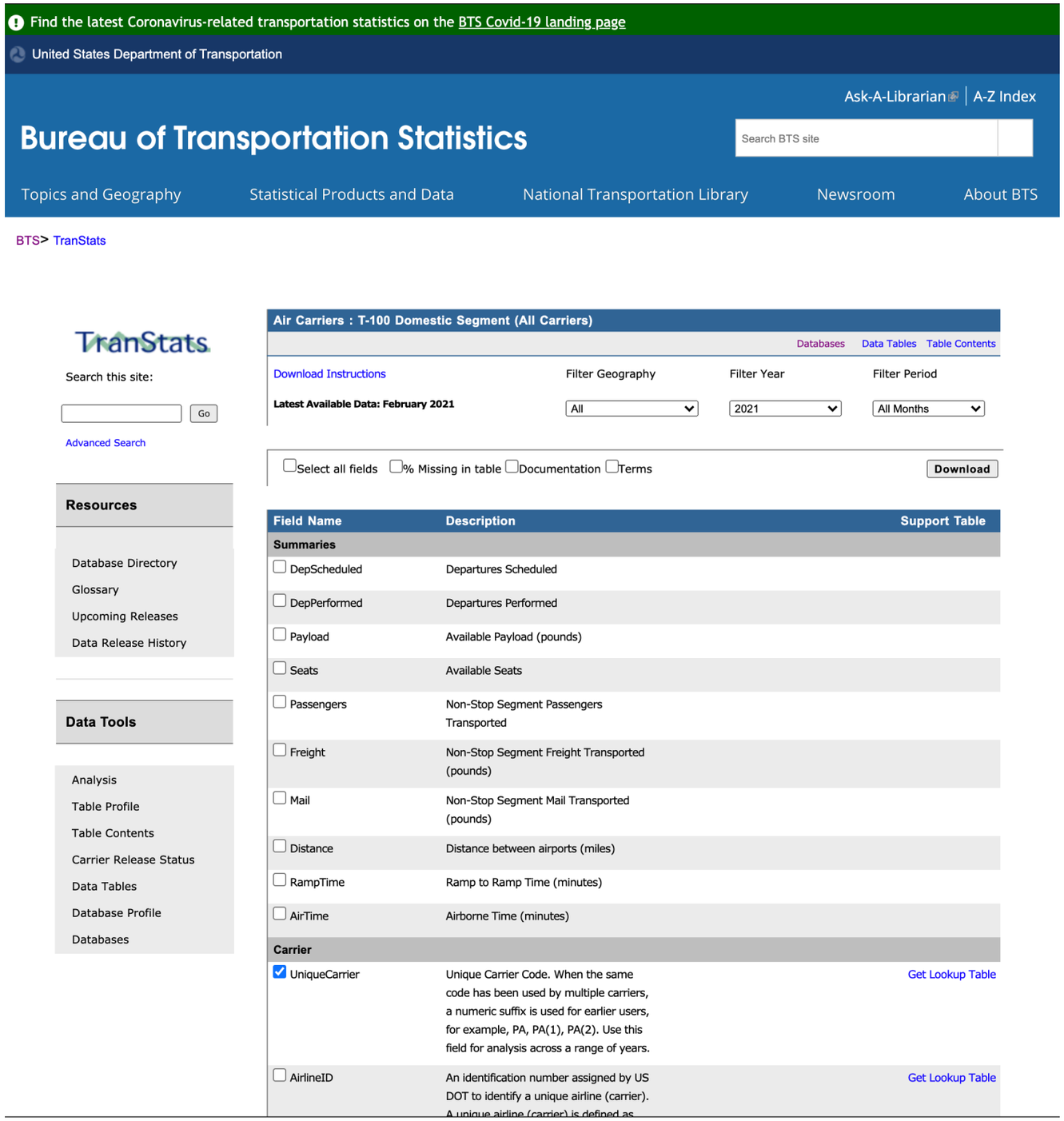
Mathematical Details

- Mathematical object, with set of nodes and edges
- Node - Individual Element
- Edge - Connection between element
- Degree - Number of edges connected to a node
- Weighted Edge - Edge with a weight associated
- Directed Edge - “One way street”



Airline Network

- The Bureau of Transportation Statistics provides a wealth of information about transportation in the US
- We'll look at the 'T-100 Domestic Segment table' that contains information on flights between any two airports serviced by a US Carrier



Airline Network

- Each row in this DataFrame corresponds to one edge in an 'edge list' format
- Airports correspond to individual nodes
- The database contains one row per individual flight so there are many repeated edges (which we can aggregate)
- Edges are directed as the number of people traveling from A to B is not the same that are traveling from B to A

	ORIGIN	DEST	PASSENGERS
9400	GCN	GCN	41012
2982	BLD	BLD	22881
19516	PGA	PGA	22466
14527	LKE	LKE	9876
15	1G4	1G4	3101
...	...	...	...
11877	ILI	ILI	0
11840	IGM	IGM	0
10628	HOU	HOU	0
10499	HNL	HNL	0
27230	YIP	YIP	0

Airline Network

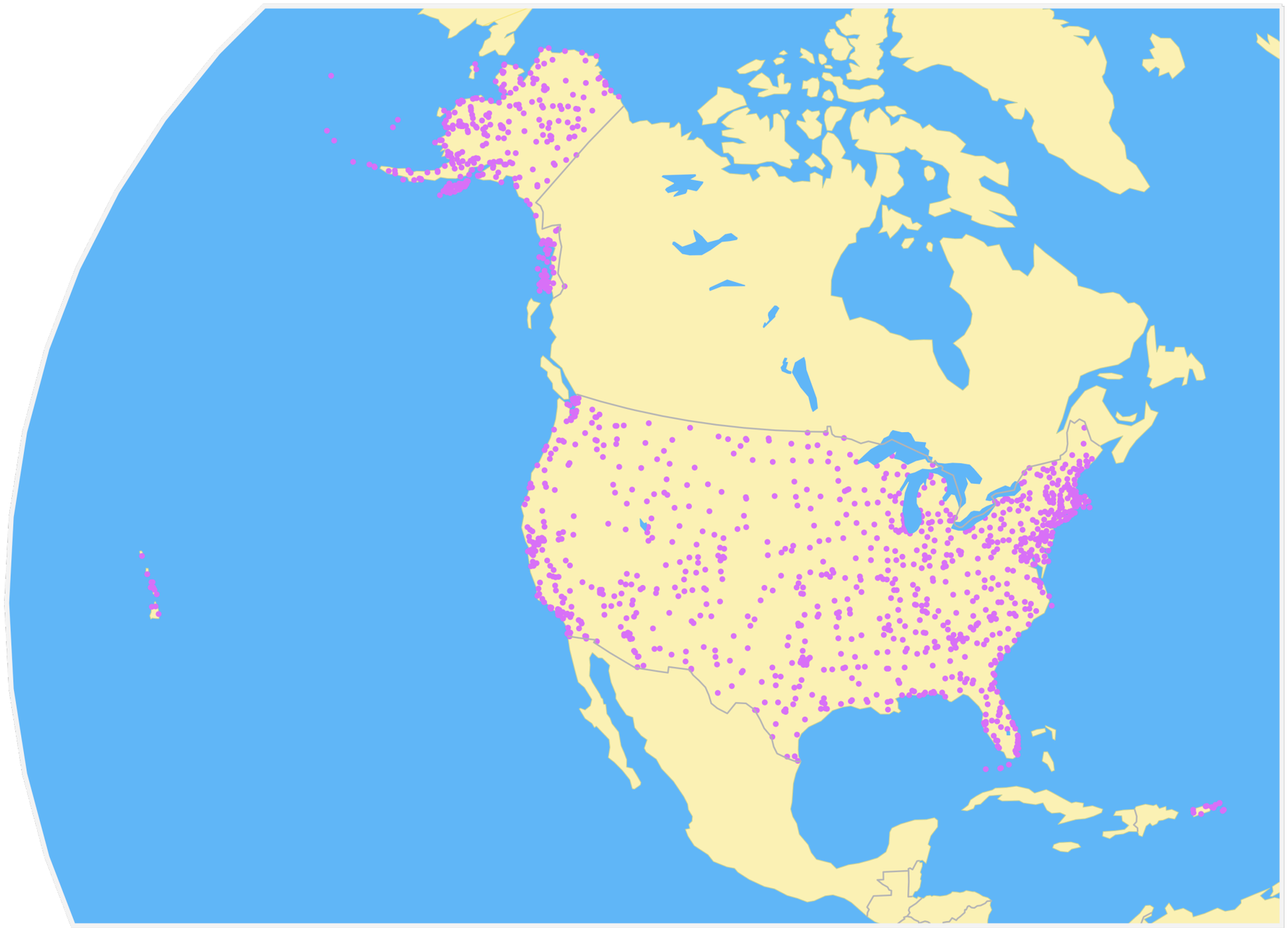
- Each row in this DataFrame corresponds to one edge in an 'edge list' format
- Airports correspond to individual nodes
- The database contains one row per individual flight so there are many repeated edges (which we can aggregate)
- Edges are directed as the number of people traveling from A to B is not the same that are traveling from B to A

	ORIGIN	DEST	PASSENGERS
9400	GCN	GCN	41012
2982	BLD	BLD	22881
19516	PGA	PGA	22466
14527	LKE	LKE	9876
		G4	3101
			...
11877	ILI	ILI	0
11840	IGM	IGM	0
10628	HOU	HOU	0
10499	HNL	HNL	0
27230	YIP	YIP	0

Weighted Directed Graph

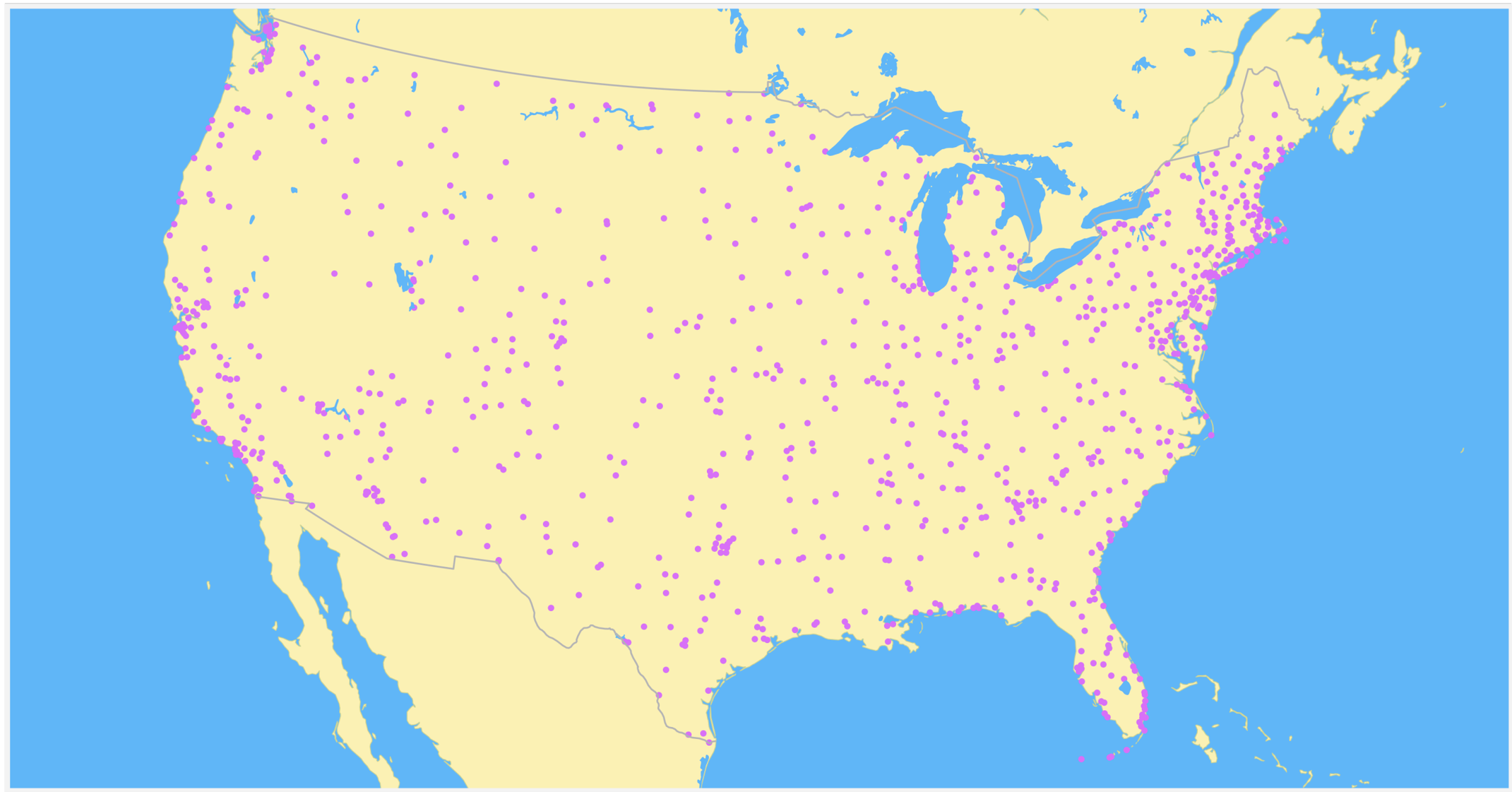
Nodes

- Nodes are airports



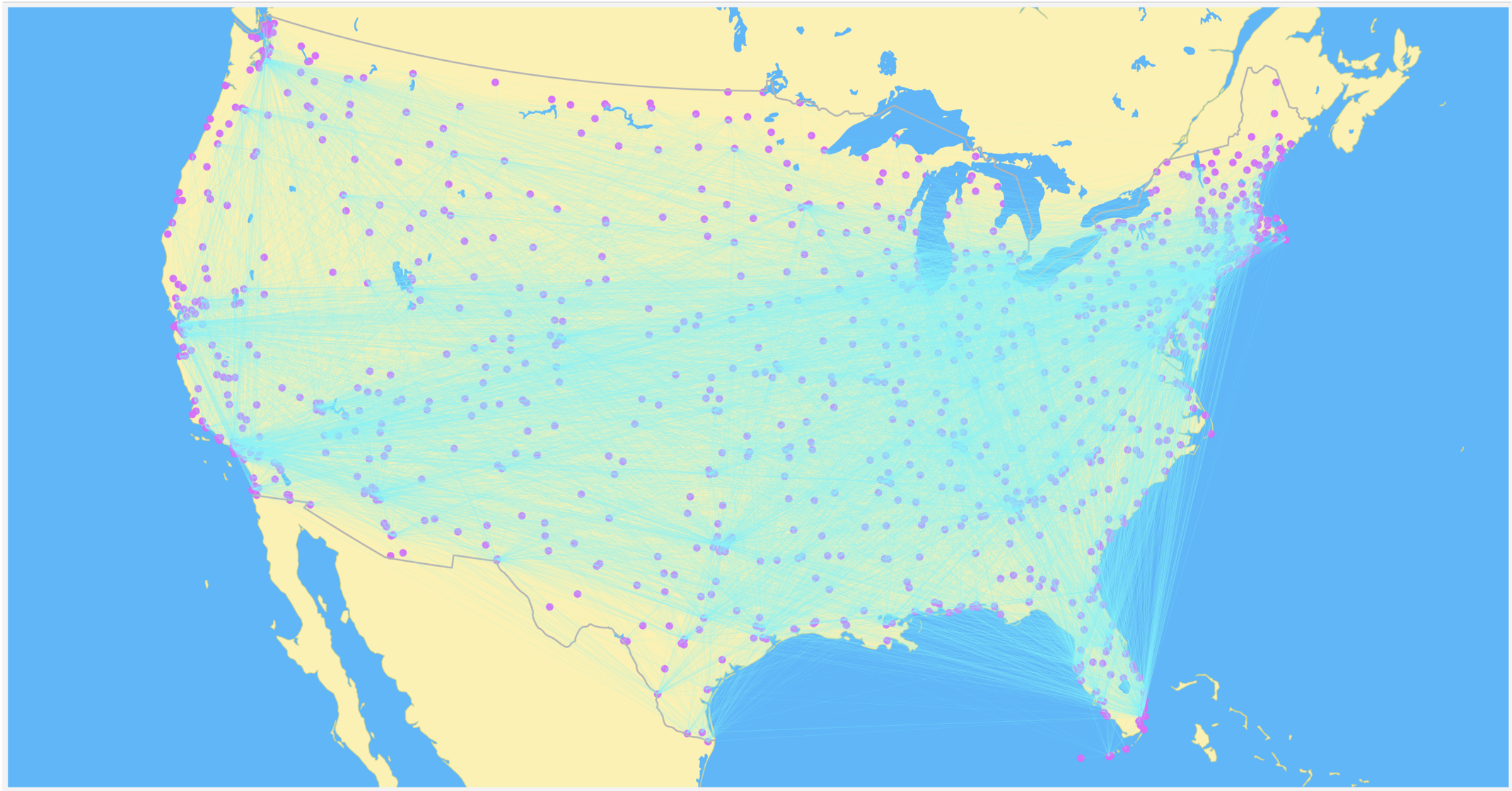
Nodes

- Nodes are airports
- We'll focus on the contiguous 48 states



Edges

- Edges are flights
- We aggregate all the flights going from airport A to airport B



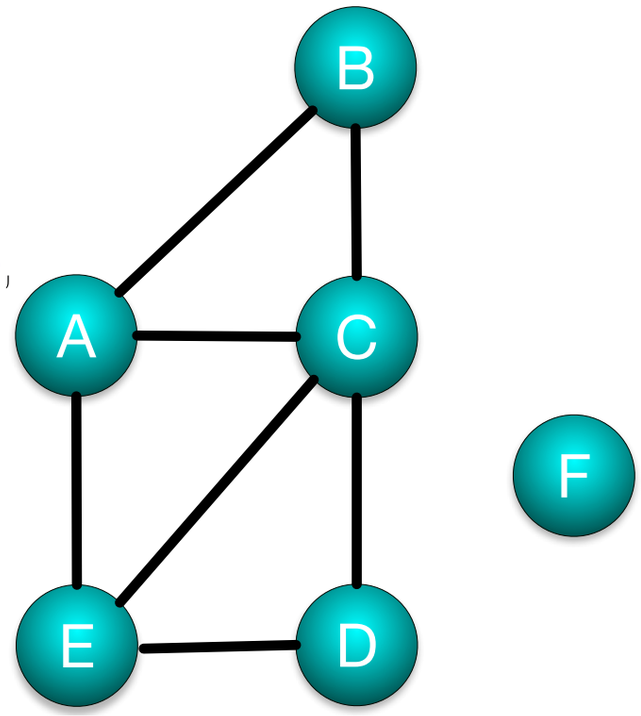
Degrees

- The most fundamental property of a node is its degree, k - the number of connections of a node

- In the case of **directed networks**, we distinguish between in-degree, k_{in} (number of incoming connections) and out-degree k_{out} (number of outgoing connections).

- Degree distribution - The relative frequency of each degree value:

$$P(k) = \frac{1}{N} \sum_{k_i \equiv k} 1$$

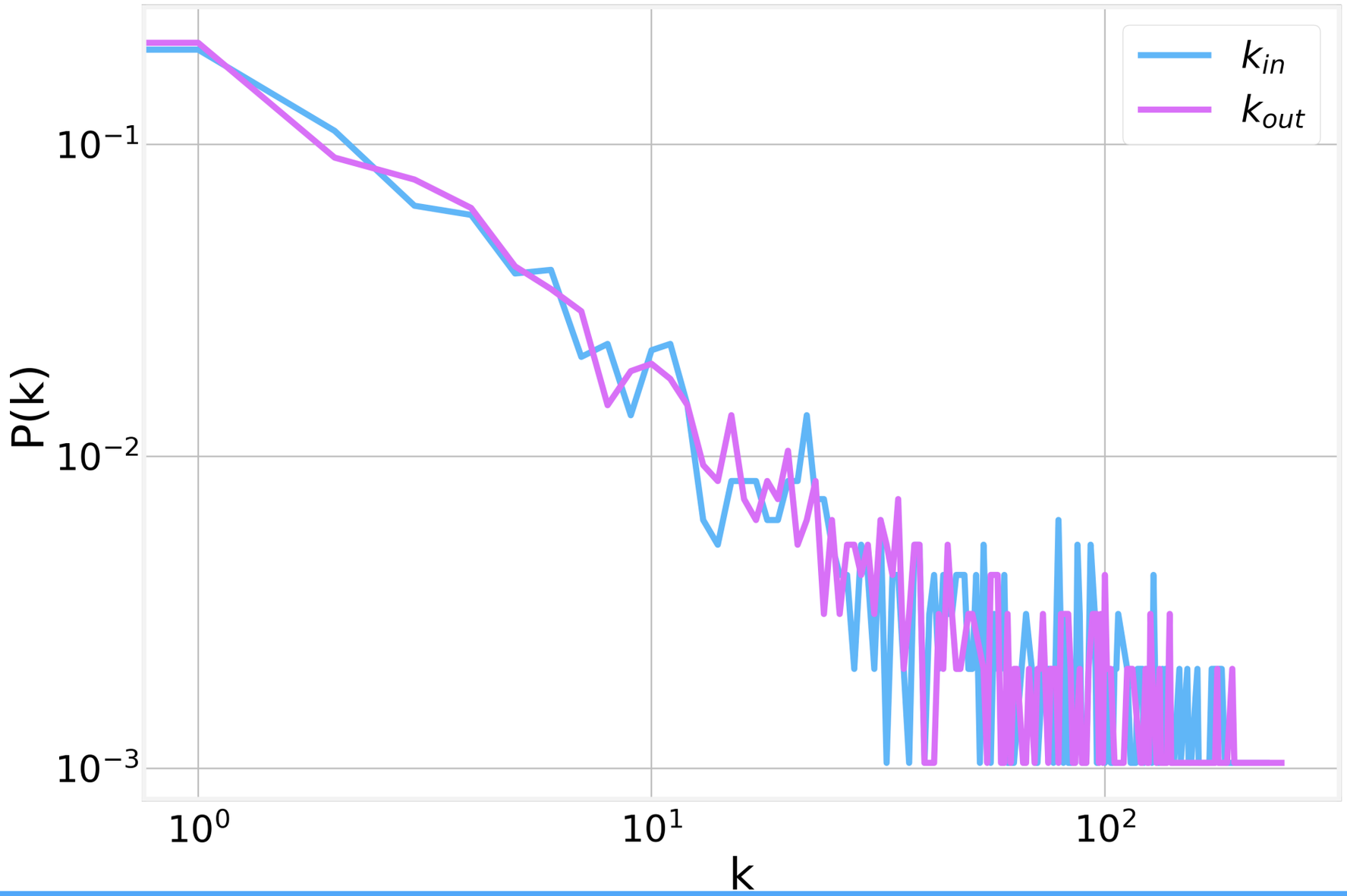


	Degree (k_i)
A	3
B	2
C	4
D	2
E	3
F	0

	P(k)
0	1/6
1	0
2	1/3
3	1/3
4	1/6

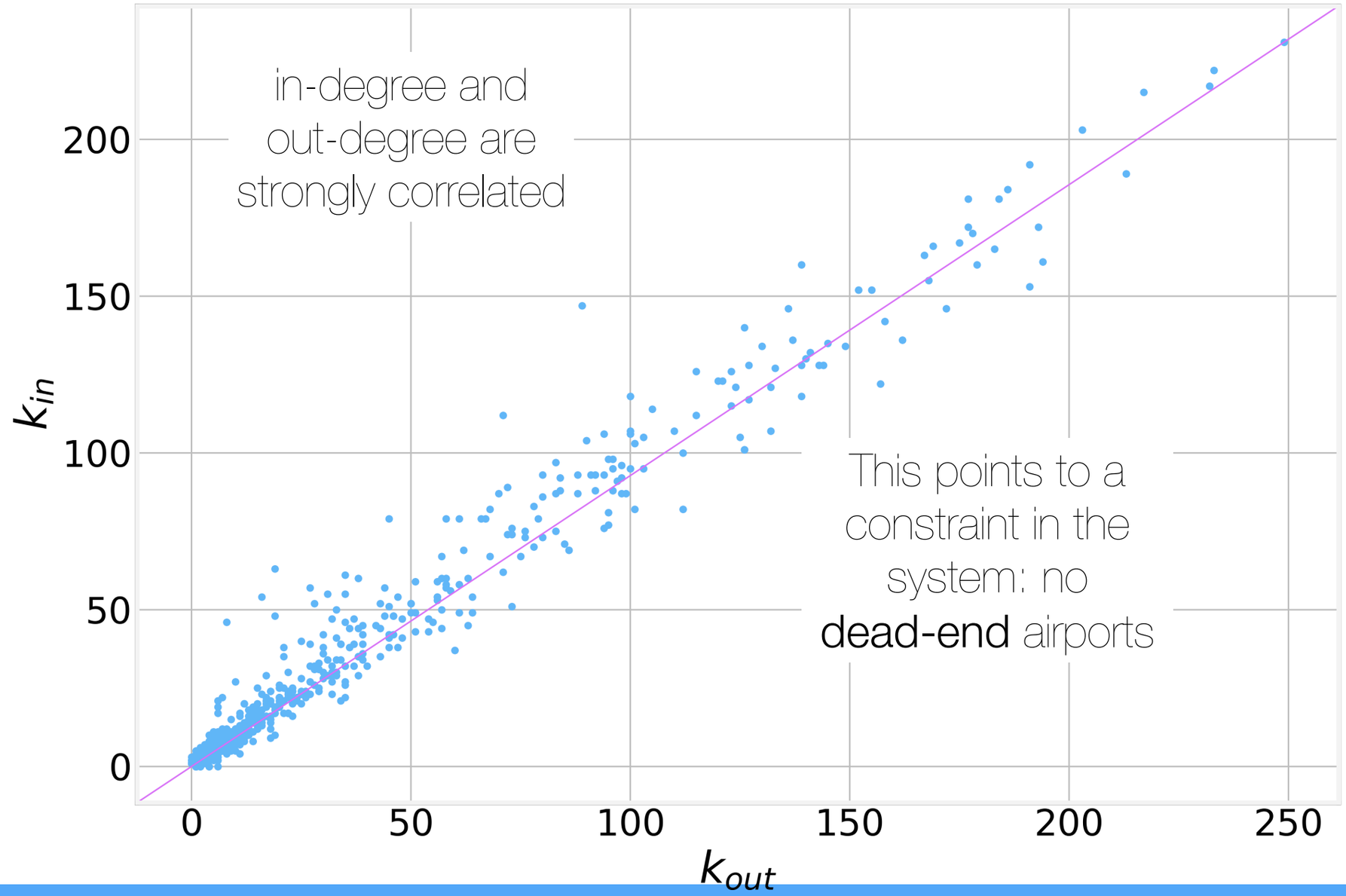
Degree Distribution

- Power-law distribution for both in- and out-degrees



in-out degree correlations

- Different networks will display different behaviors



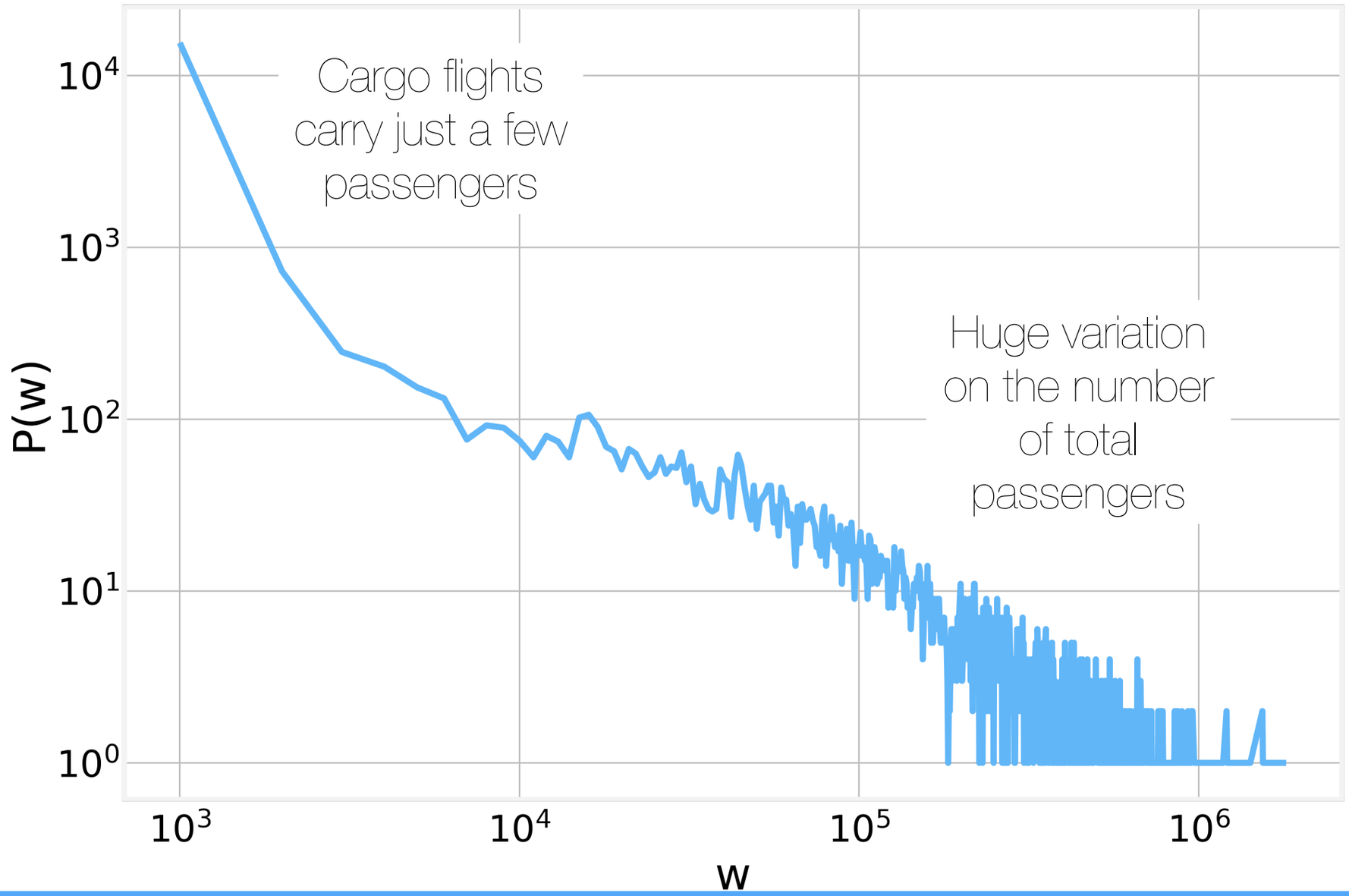
Edge Weight

- So far we have focused exclusively on the number of edges, but edges can also have weights

- Edge weights can represent:
  - connection strength (close friend vs acquaintance)
  - connection frequency (daily calls vs yearly meetings)
  - physical distance (flight time between cities)
  - etc…
- Node Strength:
  - The sum of the weights of all the nodes connected to the node
  - Weighted equivalent to node degree
- Weights are strongly related with network topology!

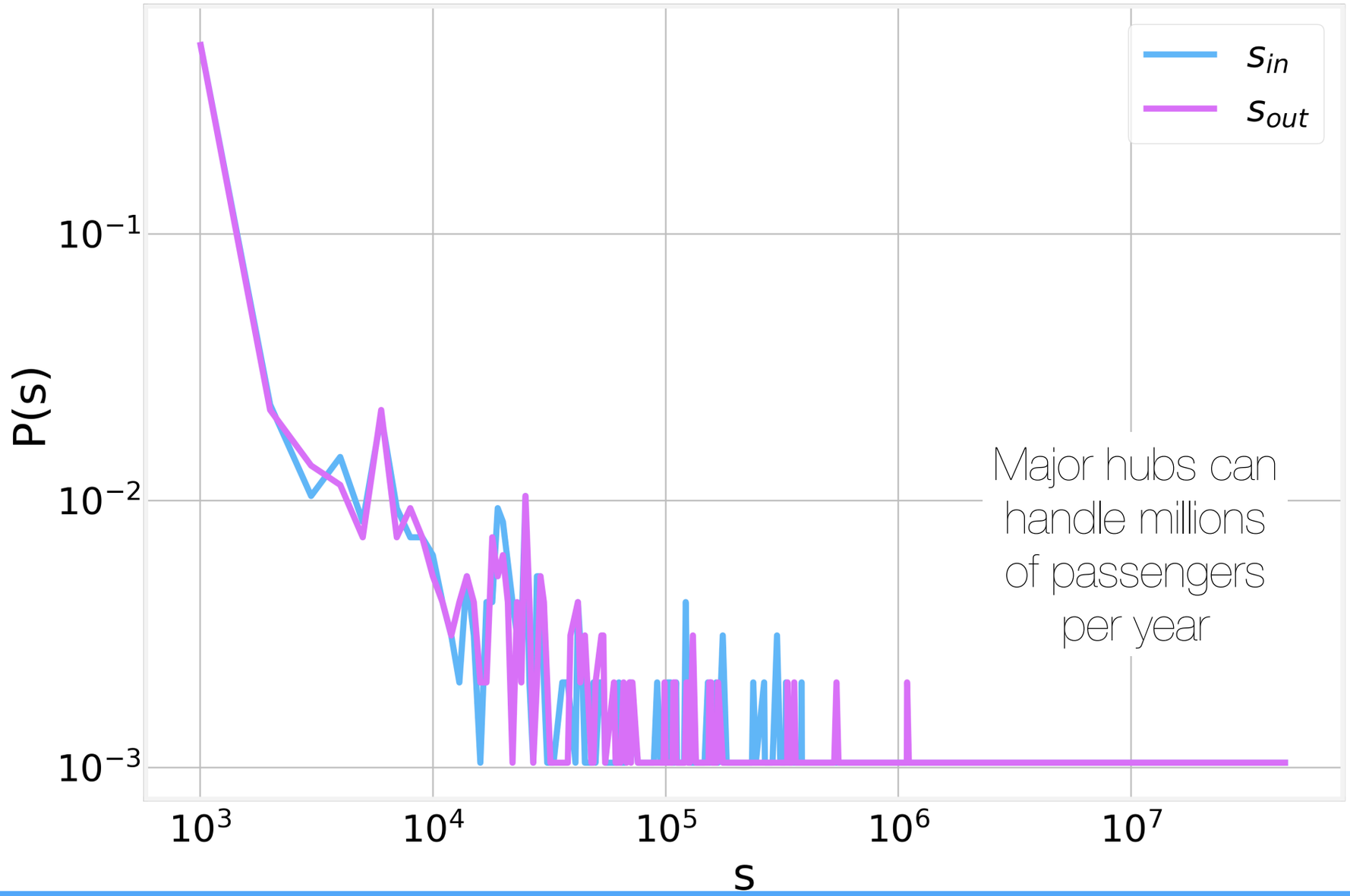
Edge Weight

- Weights are the number of passengers traveling from A to B
- Weight distribution is broad-tailed. Not necessarily power-law but allowing for large weights



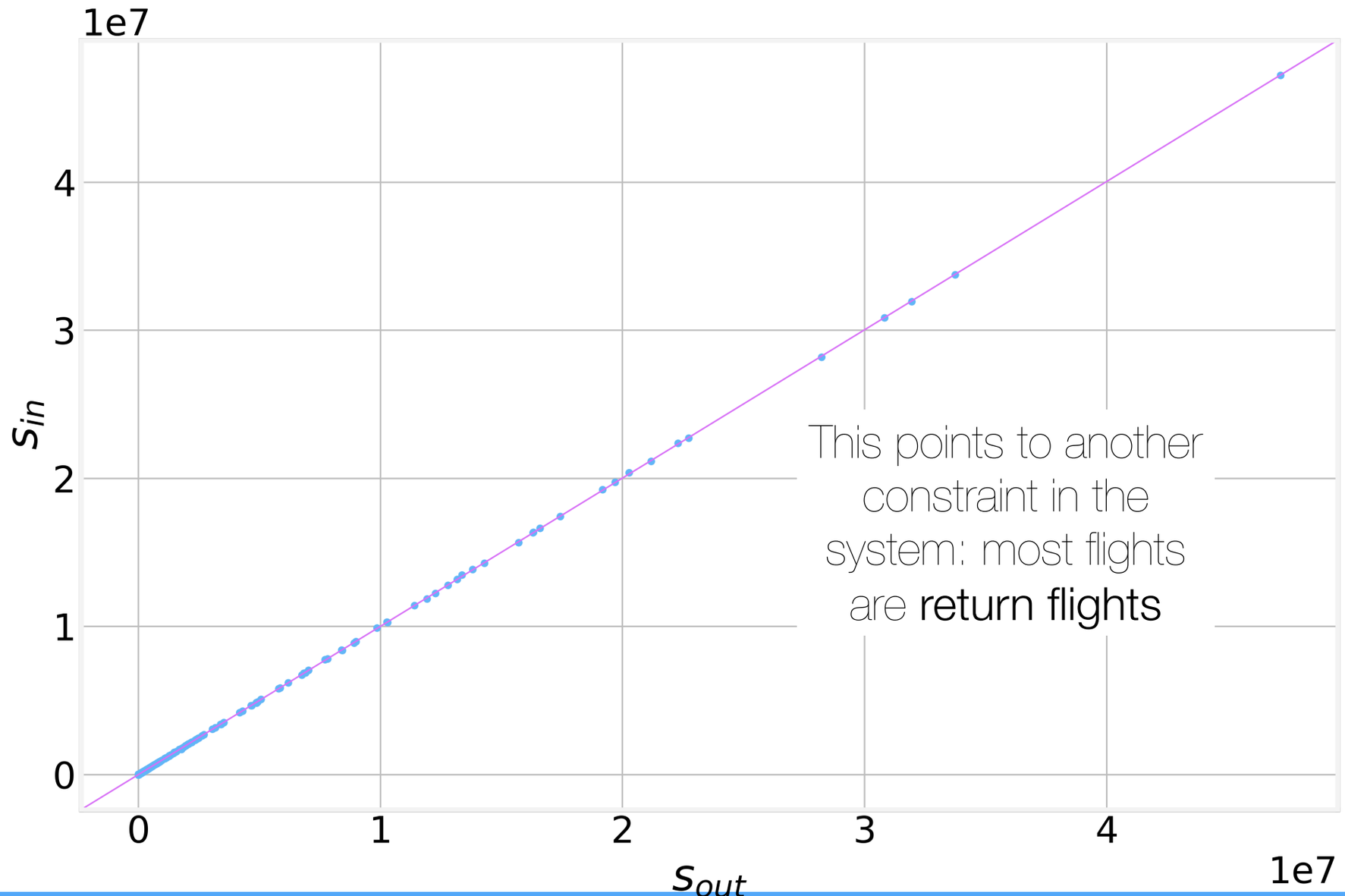
Strength

- Strength is the weighted equivalent of the degree
- We consider both in-strength and out-strength



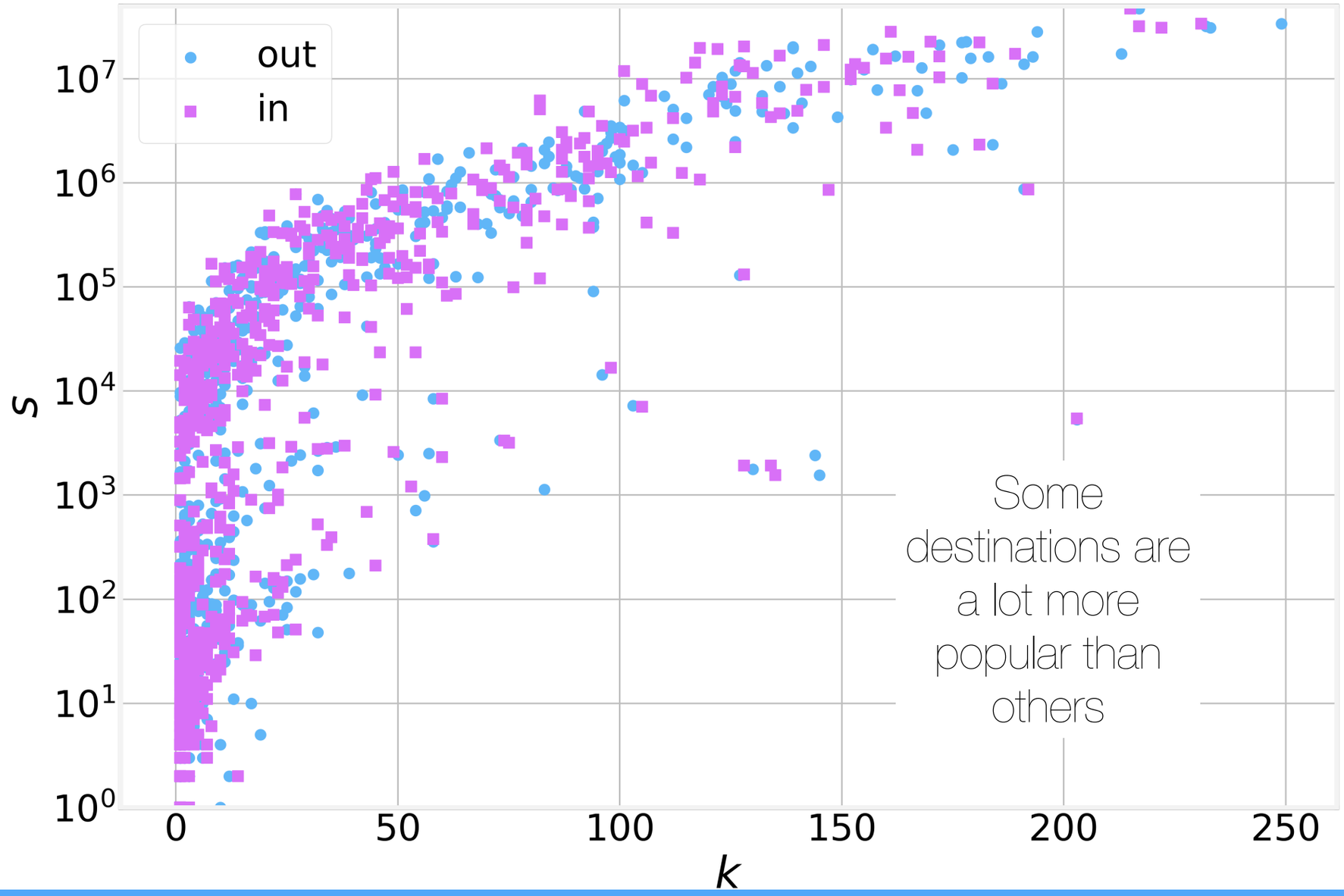
Strength

- Strong correlations between the in and out strength



Degree-Strength Correlations

- Airports with higher in- or out-degree also have higher in- out- strength but the dependency is not trivial



Code - Graphs 101

https://github.com/DataForScience/PyGotham2021

2. Searching in Graphs

OpenStreetMap

openstreetmap.org

- The Wikipedia of Maps
- Represents roads and intersections as edges and nodes, respectively

OpenStreetMap

From Wikipedia, the free encyclopedia

OpenStreetMap (**OSM**) is a collaborative project to create a free editable geographic database of the world. The geodata underlying the maps is considered the primary output of the project. The creation and growth of OSM has been motivated by restrictions on use or availability of map data across much of the world, and the advent of inexpensive portable satellite navigation devices.[5]

Created by Steve Coast in the UK in 2004, it was inspired by the success of Wikipedia and the predominance of proprietary map data in the UK and elsewhere.[6][7] Since then, it has grown to over two million registered users.[8] Users may collect data using manual survey, GPS devices, aerial photography, and other free sources, or use their own local knowledge of the area. This crowdsourced data is then made available under the Open Database License. The site is supported by the OpenStreetMap Foundation, a non-profit organisation registered in England and Wales.

The data from OSM can be used in various ways including production of paper maps and electronic maps, geocoding of address and place names, and route planning.[9] Prominent users include Facebook, Wikimedia Maps, Apple, Microsoft, Amazon Logistics, Uber, Craigslist, Snapchat, OsmAnd, Maps.me, MapQuest Open, JMP statistical software, and Foursquare. Many users of GPS devices use OSM data to replace the built-in map data on their devices.[10] OpenStreetMap data has been favourably compared with proprietary datasources,[11] although as of 2009 data quality varied across the world.[12][13][*needs update*]

OpenStreetMap

OpenStreetMap's logo

Screenshot	[show]
Type of site	Collaborative mapping
Available in	UI: 96 languages and variants[1] Map data: Local languages
Owner	Community-owned; supported by OpenStreetMap Foundation[2]
Created by	Steve Coast (User page in OSM)
Products	Geographic data
URL	www.openstreetmap.org
Commercial	No
Registration	Required for contributors, not required for viewing
Users	7,450,000[3]
Launched	9 August 2004; 17 years ago[4]
Current status	Active (details)
Content license	Open Database License (ODbL)

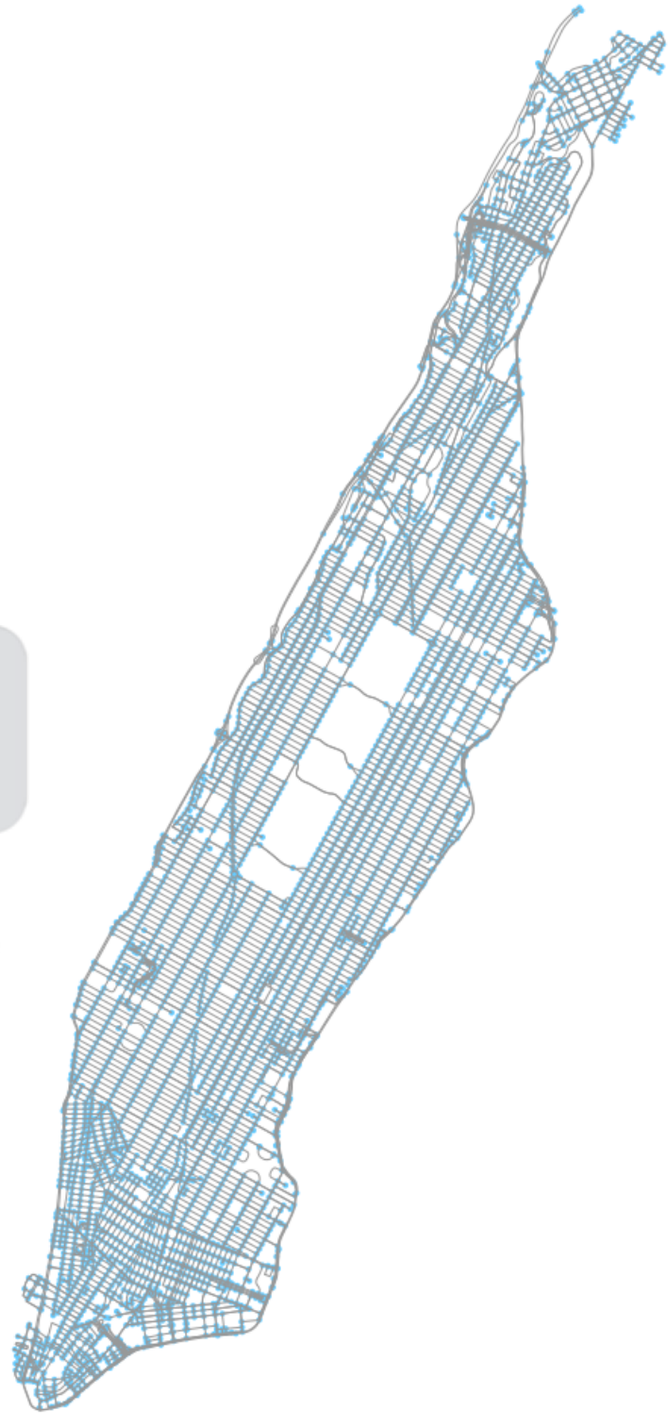
osmnx

https://github.com/gboeing/osmnx

- Python package to easily download OpenStreetMap data
- Builds on top of GeoPandas, NetworkX, and matplotlib
- Supports different modes of transportation (driving, walking, etc)
- We can obtain the driving network for a specific region with just a couple lines of code

```
import osmnx as ox

place = 'Manhattan, NY, USA'
G_roads = ox.graph_from_place(place, network_type='drive')
```

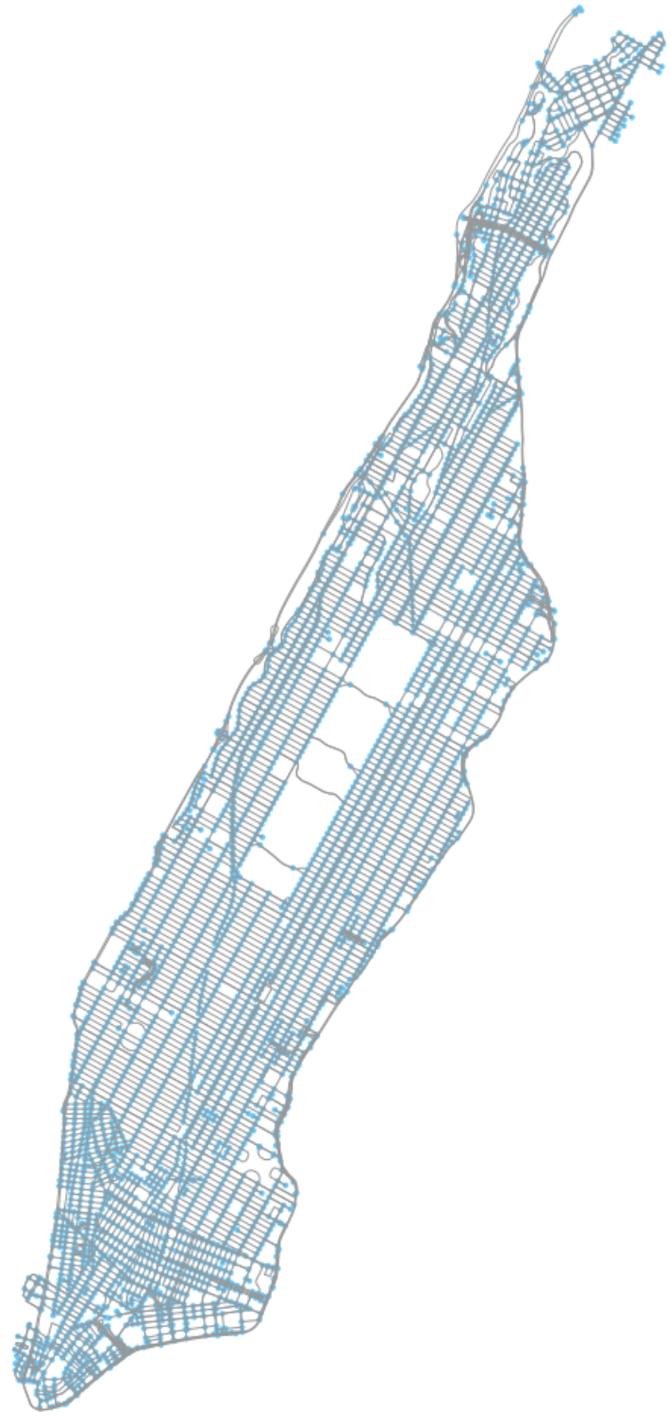
- Returns a NetworkX object that provides us with a simple interface to manipulate the graph



Shortest paths and Distances

https://github.com/gboeing/osmnx

- A common problem in graph analysis is to determine the shortest path between two given nodes:
  - Minimum number of hops on a computer network
  - Fastest drive path between two different locations on a city
  - etc
- There's a very rich literature on algorithms to identify the shortest paths and to measure the shortest distance between two nodes on a graph:
  - Breadth-First Search - Explore all the local neighbors first
  - Depth-First Search - Keep going until you hit a dead-end then backtrack
  - Dijkstra's Algorithm - Keep Extending the current shortest path



Dijkstra's Algorithm

- Algorithm created by Edsger W. Dijkstra, a Dutch computer scientist, in 1956 to find the shortest paths between nodes on a weighted graph

- Most common formulation finds the shortest path between a source node and every other node on the graph

- Dijkstra's algorithm proceeds in an incremental way in which we continuously update our best estimate for the distance between the origin and every other node. Define each node to be at an unknown (or infinite) distance from the origin. The origin is defined to be at distance 0 from itself.

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

- At each step:
  - Find the node i with the smallest current distance to the origin
  - Update the distance to each of it's nearest neighbors, j

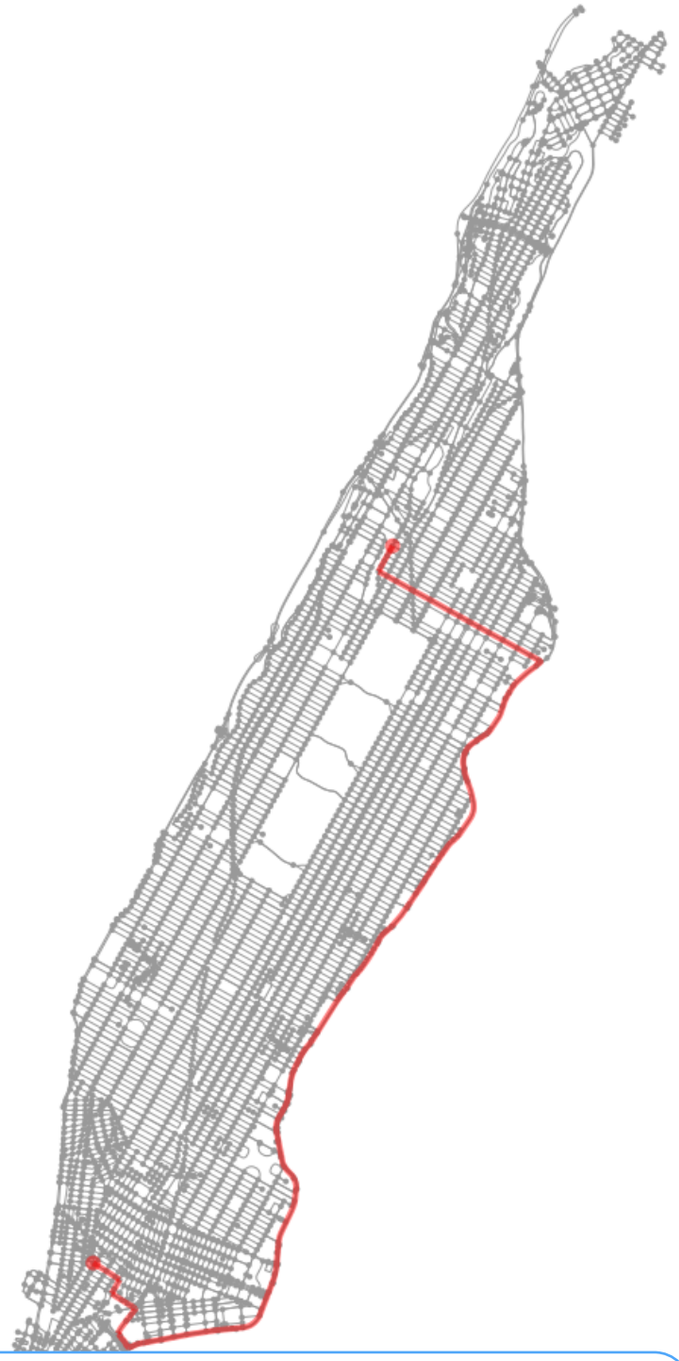
$$dist\left(node_j\right) = \min\left[dist\left(node_j\right), dist\left(node_i\right) + w_{ij}\right]$$

  - Continue until the destination has been reached or we have calculated the distance to every other node
  - The shortest path can be calculated as well by tracking which was the node that resulted in a distance update

en.wikipedia.org/wiki/Dijkstra's_algorithm

Dijkstra's Algorithm

- Assumes weighted graphs
    - Unweighted graphs can be handled by assigning a weight of 1 to each edge
- All weights must be positive. Negative weights will cause infinite loops as you can always reduce your current "distance" estimate by traversing a negative weight edge.
- Can be particularly slow on some topologies.
- In unweighted graphs, it becomes similar to Breath First Search



en.wikipedia.org/wiki/Dijkstra's_algorithm

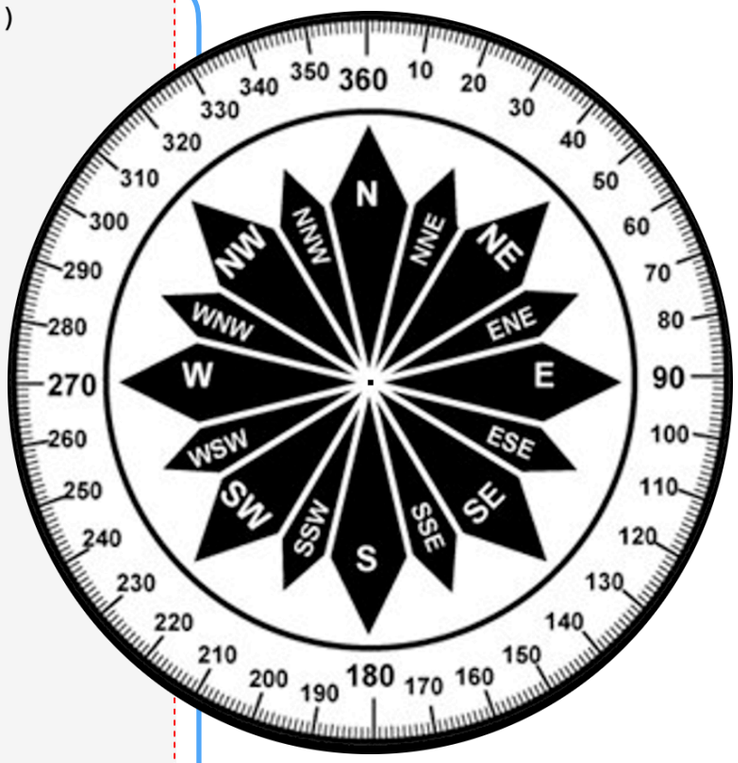
Bearings

- osmnx also provides us with the "bearing" of each edge, the angle the edge is traveling with respect to North

- By checking the change in bearing from one edge to another, we can detect when we turn left or right

- The combination of changes in bearing with changes in street names allow us to provide turn by turn directions

```
print("Drive %1.1fm down %s" % (directions[0][1], directions[0][0]))

for i in range(1, len(directions)):
    change = directions[i][2]-directions[i-1][2]

    # Find the shortest difference between the two bearings
    if change < -180:
        change += 360
    elif change > 180:
        change -= 360

    if np.abs(change) < 5:
        msg = 'Continue onto'
    elif change > 0:
        if change > 25:
            msg = 'Turn right onto'
        else:
            msg = 'Turn slight right onto'
    elif change < 0:
        if change < -25:
            msg = 'Turn left onto'
        else:
            msg = 'Turn slight left onto'

    print('%s %s and drive for %1.1fm' % (msg, directions[i][0],
                                          directions[i][1]))

print("Arrive at your destination")
```



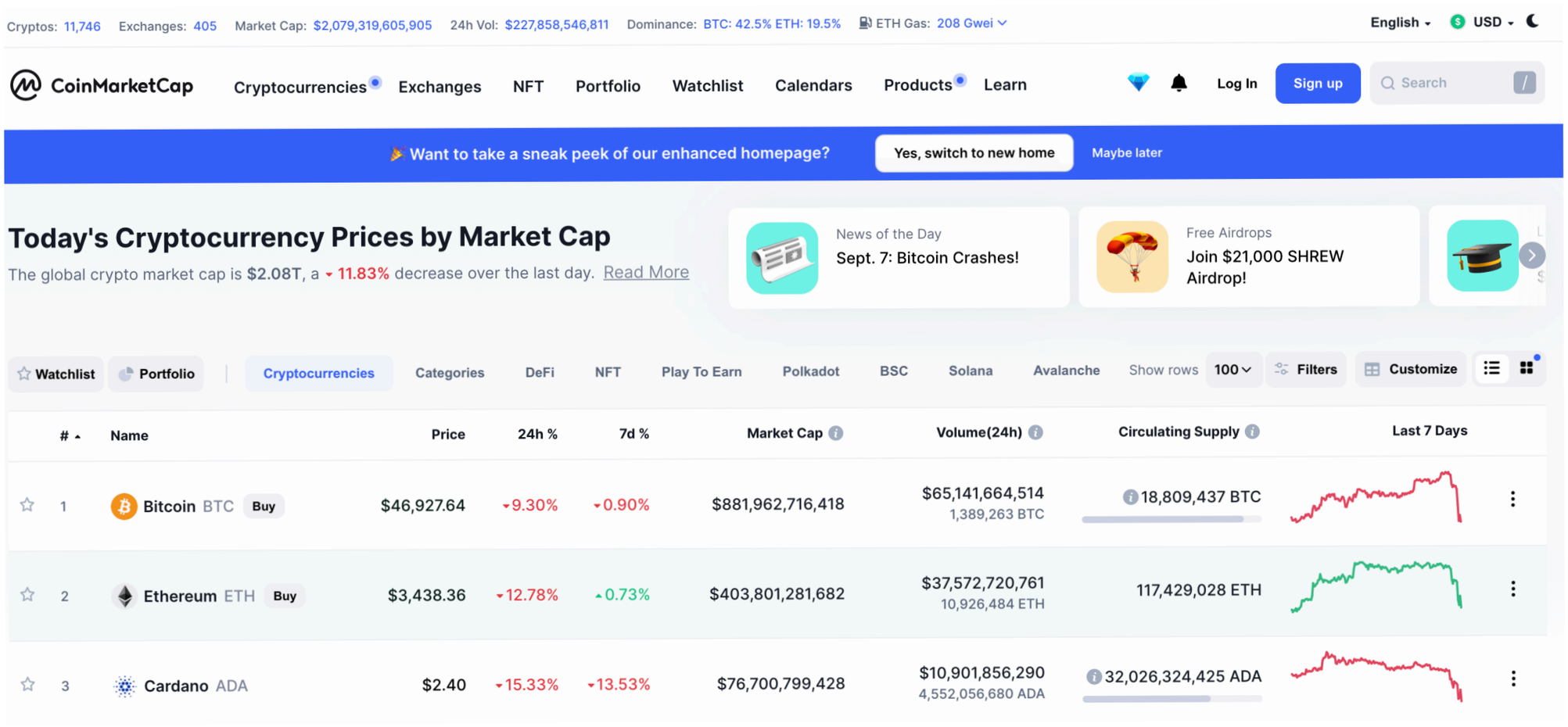
Code - Searching in Graphs

https://github.com/DataForScience/PyGotham2021

3. Graph Connectivity

BitCoin

- Introduced by Satoshi Nakamoto in 2009
- Provided the first anonymous peer-to-peer blockchain based crypto-currency, initiating the crypto-currency revolution
- Currently still the dominant crypto-currency with the largest market capitalization



Kaggle Bitcoin Transaction Dataset

https://www.kaggle.com/xblock/bitcoin-partial-transaction-dataset

Description

The Bitcoin Partial Transaction Datasets contain three snapshots of Bitcoin transaction data for easier analysis, namly dataset1*2014*11*1500000,* *dataset2*2015*6*1500000 and dataset3*2016*1_1500000. We sample the snapshots from November 2014 to January 2016 with six months as the sampling interval. Each snapshot contains the first 1,500,000 transaction records in its corresponding month, namly Nov. 2014, Jun. 2015 and Jan. 2016.

We also provide a file including the labeled addresses belonging to mixing services, and these addresses were active during the observing time of our snapshots.

Due to the pseudonymous requirements of Bitcoin, it is unlikely to enforce Know-Your-Customer (KYC) processes, which are guidelines in anti-money laundering. However, mixing services in Bitcoin, originally designed to enhance transaction anonymity, have been widely employed for money laundry to complicate trailing illicit fund.

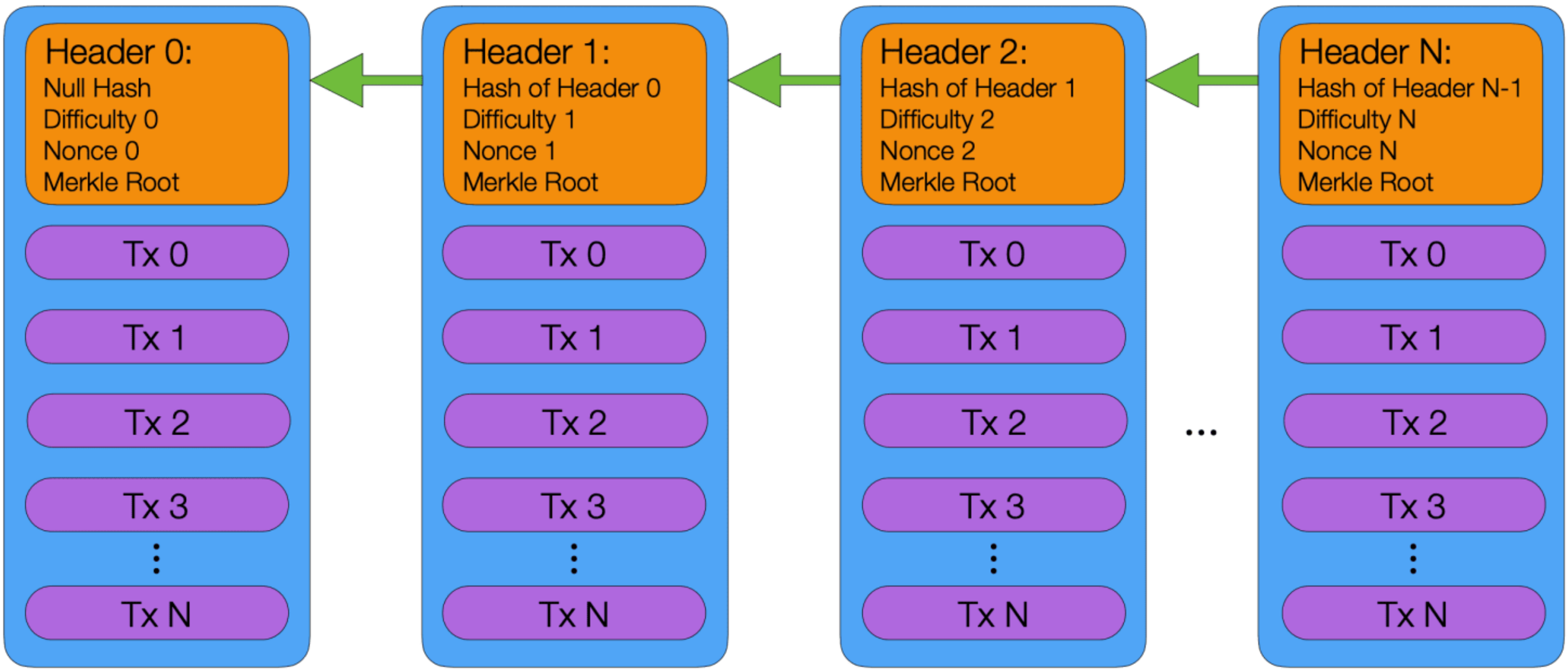
In our work, we study mixing service detection with this dataset. For further study, we can chase up users involved in criminal activities by analyzing users who take part in Bitcoin mixing.

The details of dataset1*2014*11*1500000 are described below. The file structure of dataset2*2015*6*1500000 and dataset3*2016*1_1500000 are similar to EthereumG1. You can know more information from the README file.

For more details about blockchain dataset, please click here.

Blockchain

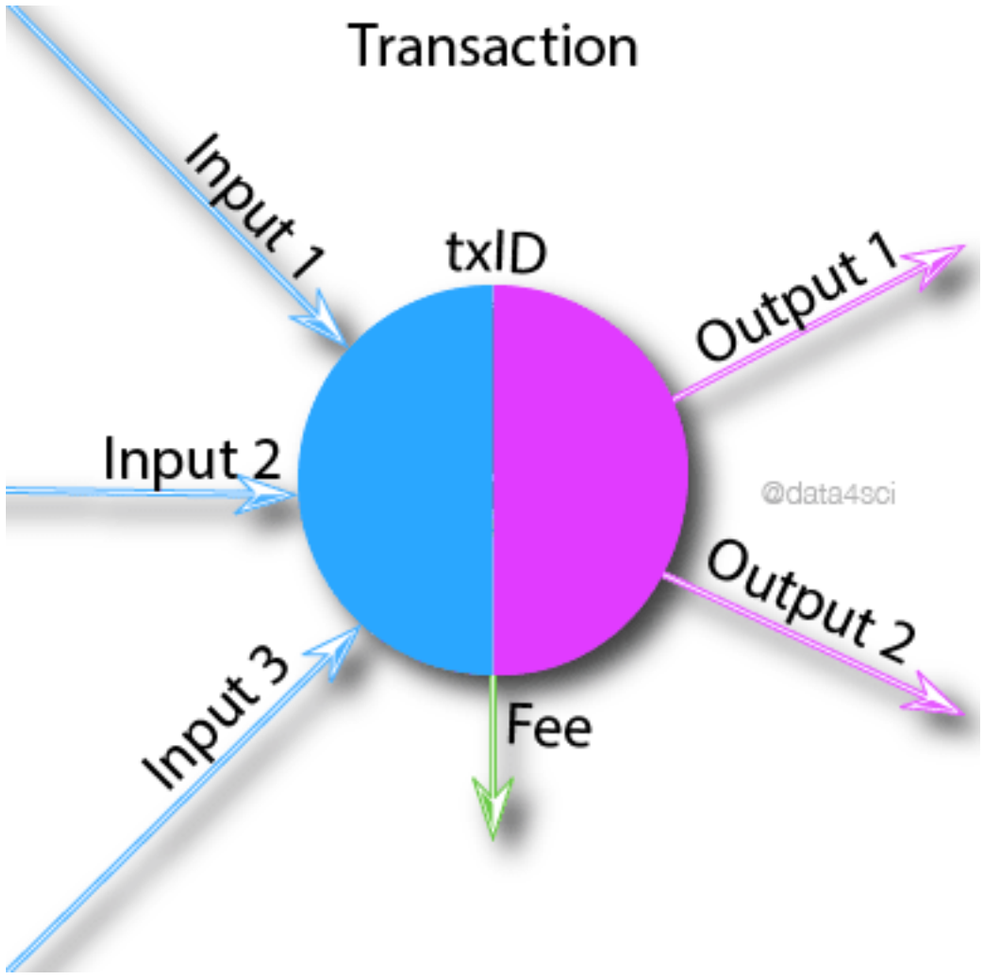
- Each Block contains a



Transactions

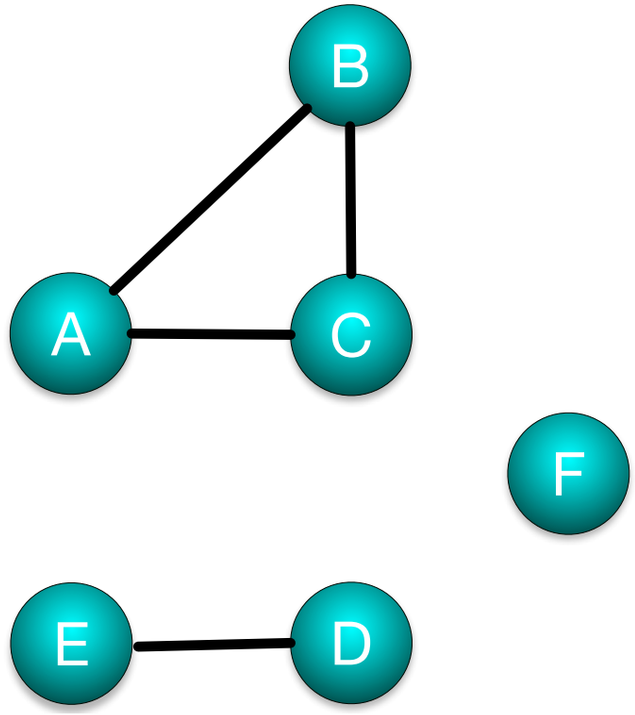
- A transaction (node) combines amounts (weights) associated with inputs (incoming edges) to produce outputs (outgoing edges)

- Outputs of one transaction can be spent as inputs to another



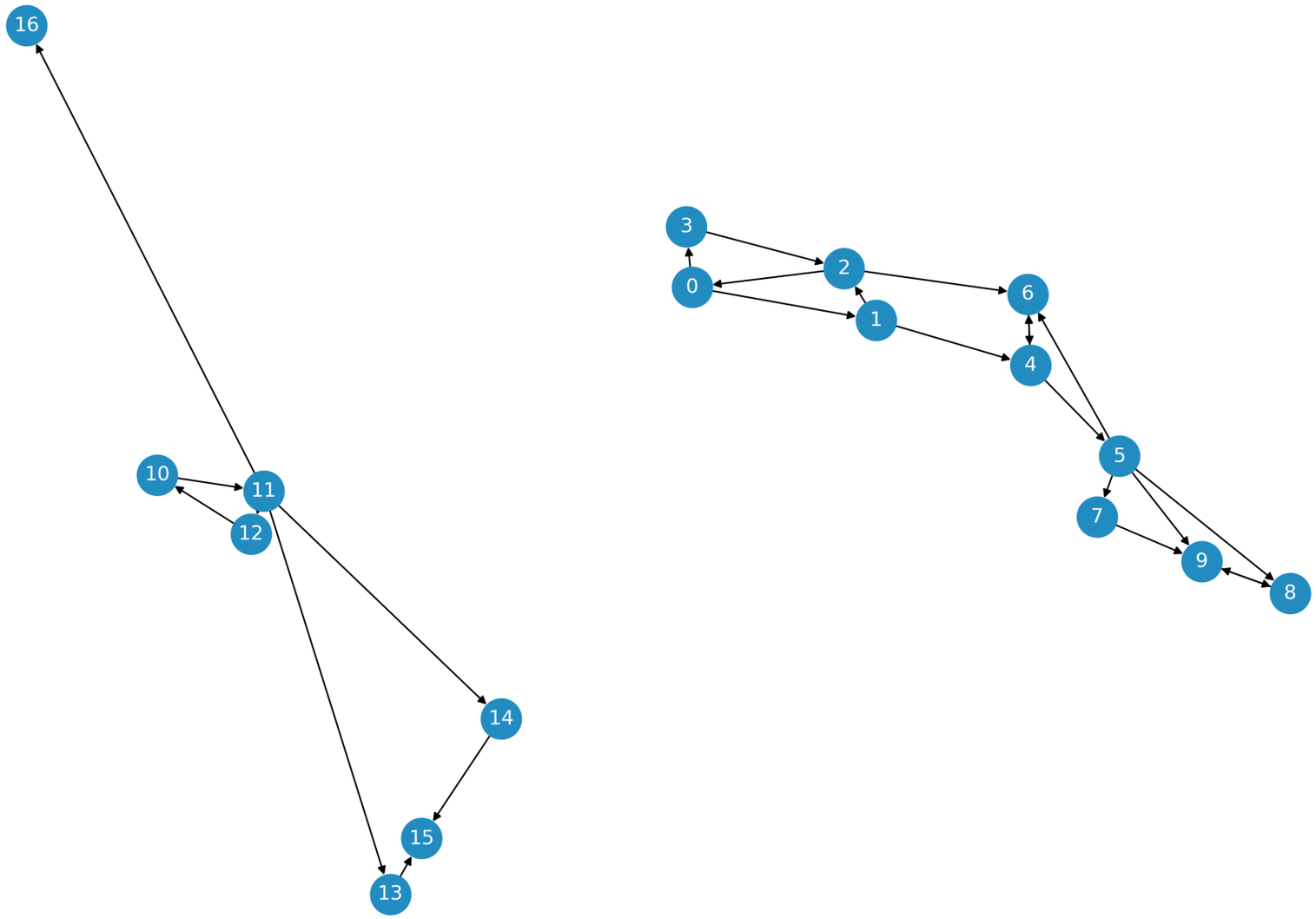
Components

- As we saw, a graph is a set of nodes connected by edges
- However, it is possible to have disconnected nodes (F) and even larger disconnected parts of the network (E and D)
- Each piece of the graph is called a Component.
- When the largest component accounts for a substantial part of the nodes of the graph, it's called the Giant Connected Component.
- Graphs with a single component are called Connected
- In the case of directed graphs, we consider two cases:
  - Strongly-connected - Every node is accessible from every other node following the direction of the edges
  - Weakly-connected - Every node is accessible by disregarding edge directions



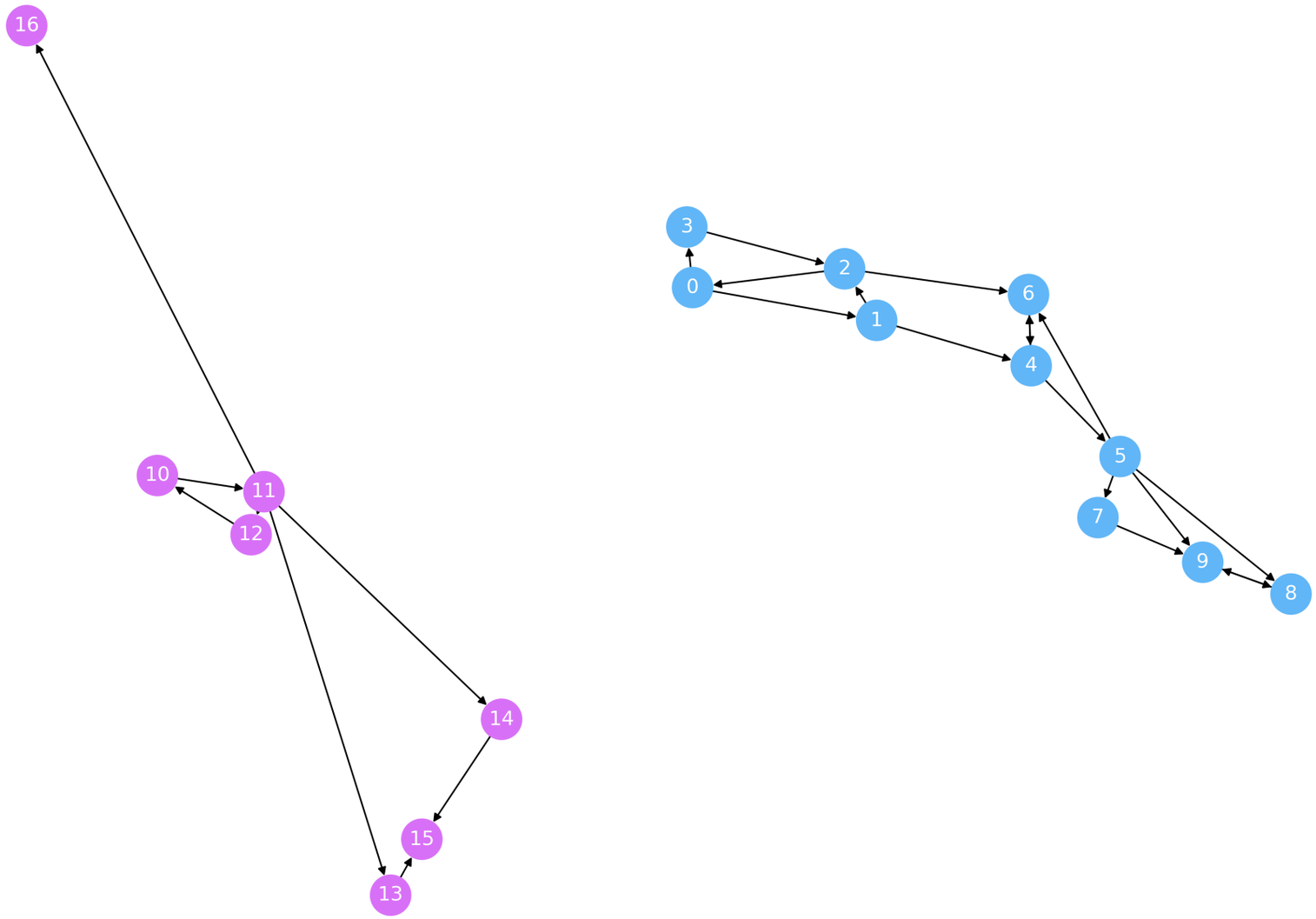
Components

- Let us consider a simple network



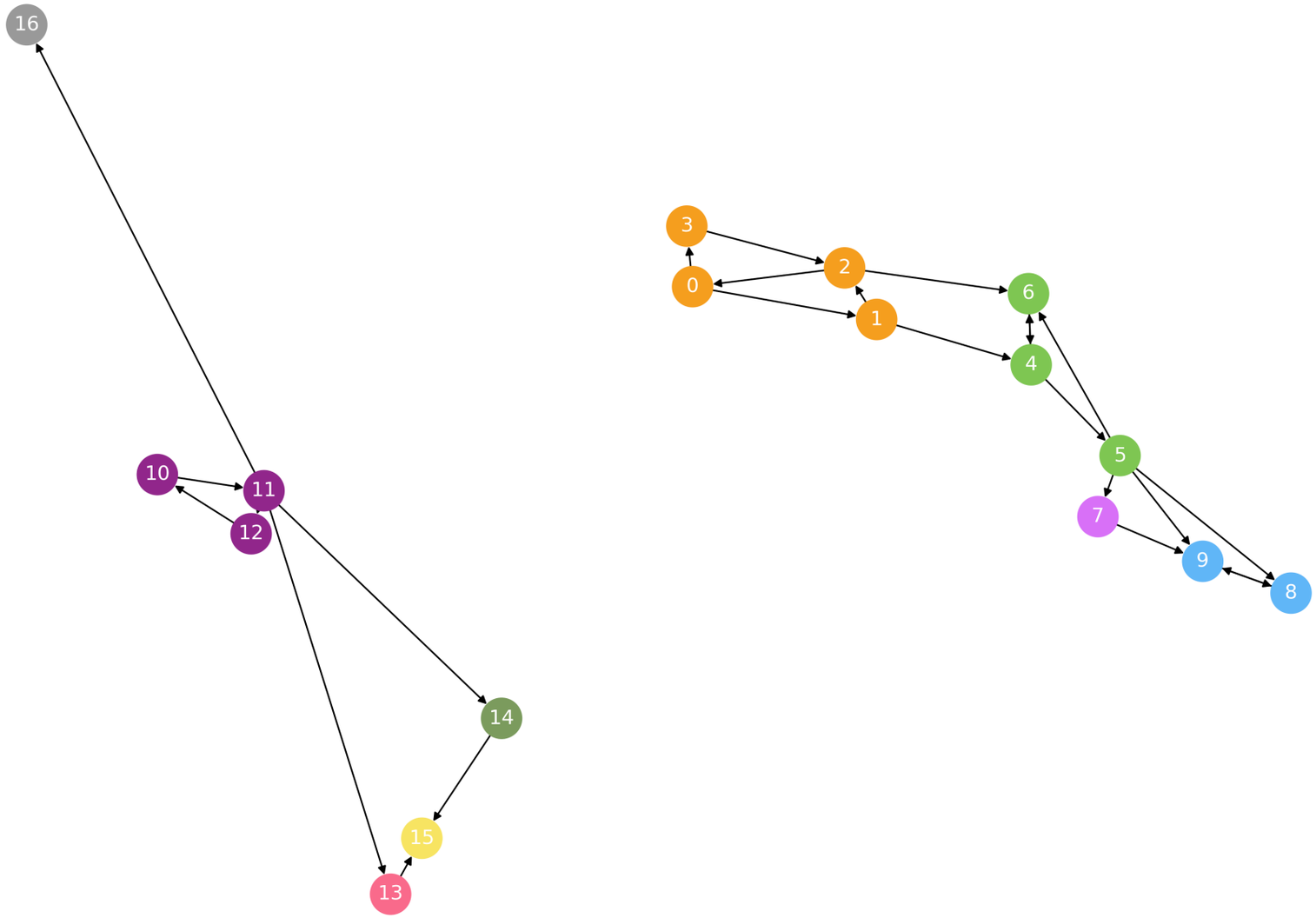
Weakly Connected Components

- The weakly Connected Components are just the components that are isolated from each other



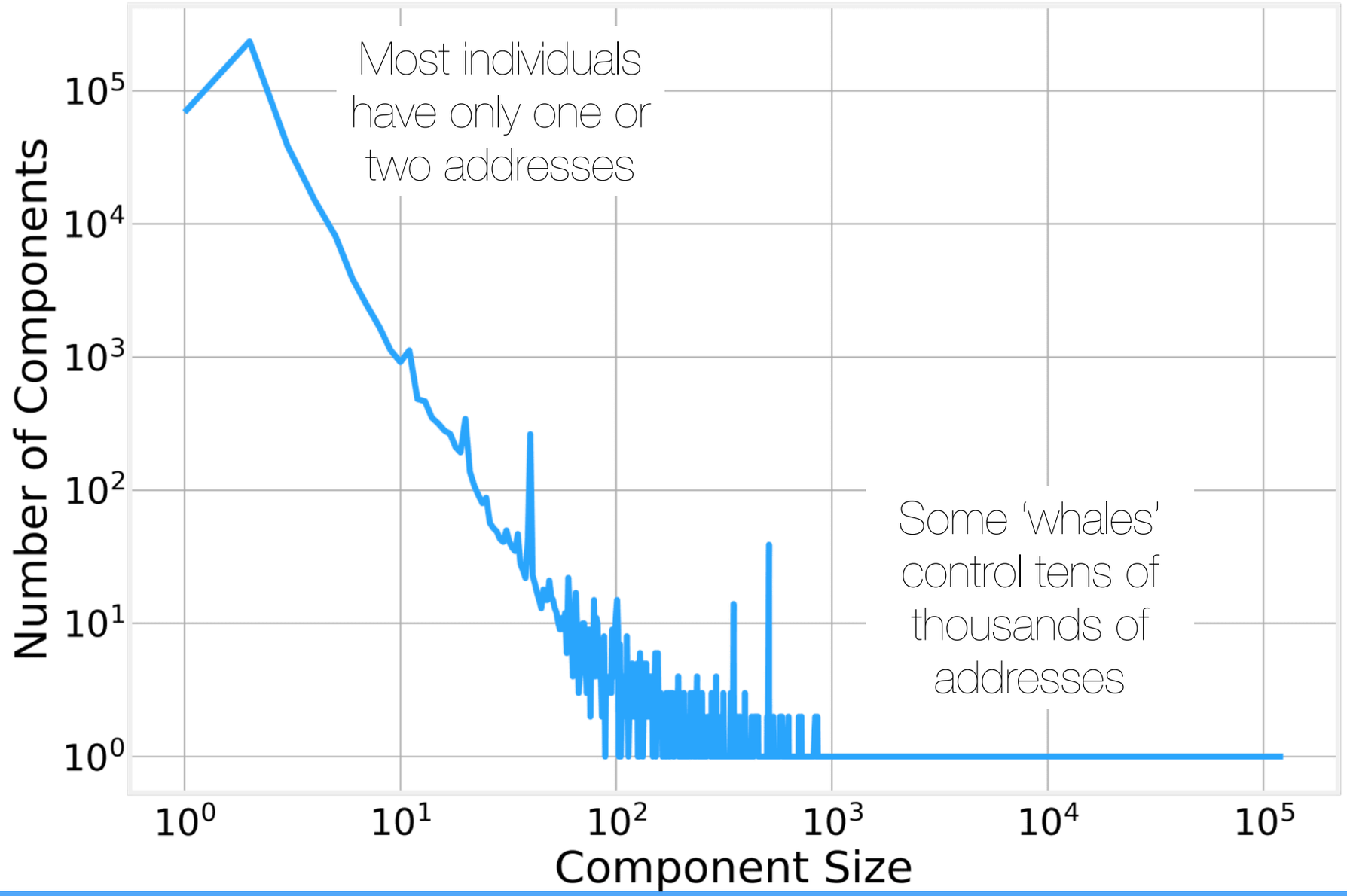
Strongly Connected Components

- Each Weekly Connected Component contains one or more Strongly Connected Components where each node is



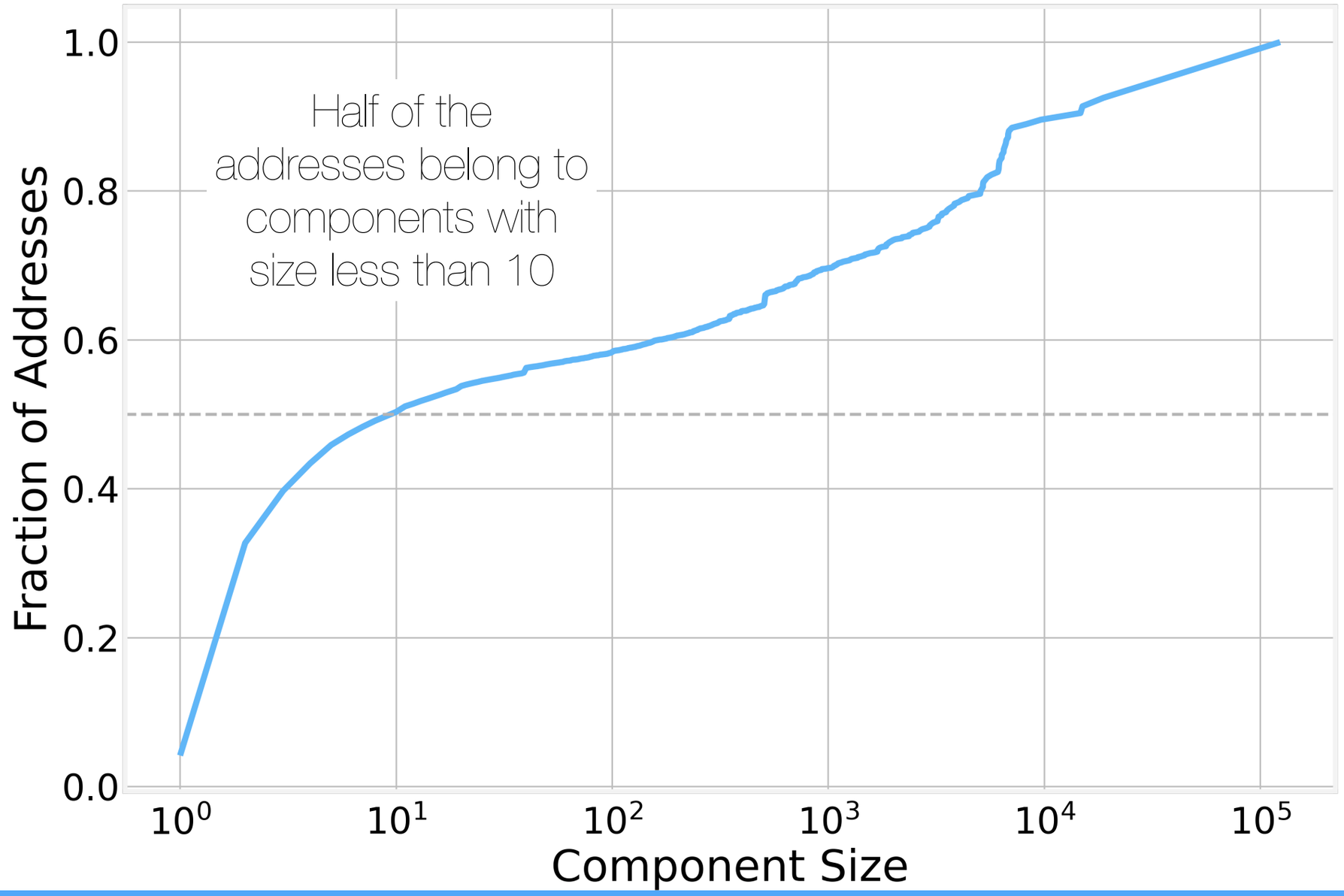
Wallets

- Wallets are addresses controlled by the same individual
- We can identified them as the Weekly Connected Components in our transaction network



Wallets

- Wallets are addresses controlled by the same individual
- We can identified them as the Weekly Connected Components in our transaction network



Code - Graph Connectivity

https://github.com/DataForScience/PyGotham2021

Thank you!

graphs4sci.substack.com

- Hope you enjoyed this talk. Looking forward to your questions and thoughts.
- You can subscribe to the graphs4sci substack to receive related content in your inbox every couple of weeks
- Don't forget to download the Jupyter notebooks and the slide deck from GitHub: https://github.com/DataForScience/PyGotham2021
- Please feel free to reach out on Social Media!
- Website: www.data4sci.com
- Twitter: @data4sci
- Email: bgoncalves@data4sci.com