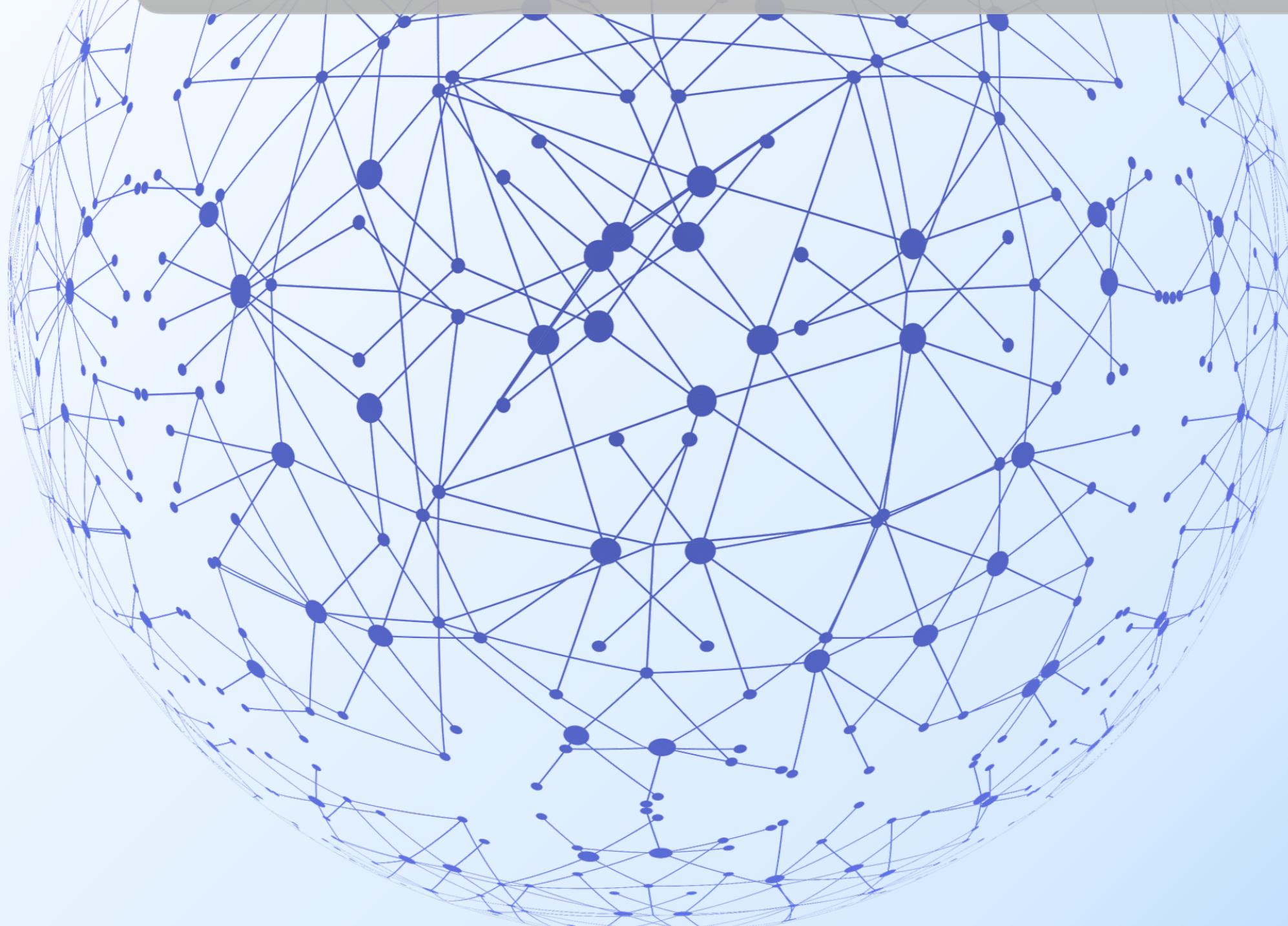


# Graphs For Data Science

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](http://www.data4sci.com)
- **Twitter:** @data4sci
- **Email:** [bgoncalves@data4sci.com](mailto:bgoncalves@data4sci.com)



## Table of Contents

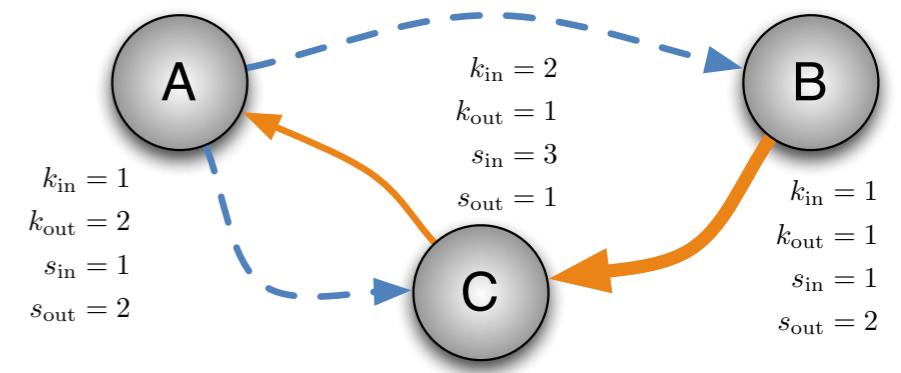
1. Graphs 101  
Airline Transportation Network
2. Searching in Graphs  
OpenStreetMap
3. Graph Connectivity  
BitCoin Transaction Network



## 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

The screenshot shows the Bureau of Transportation Statistics (BTS) website. At the top, there's a green banner with a link to the BTS Covid-19 landing page. Below it, the header includes the United States Department of Transportation logo, Ask-A-Librarian, and an A-Z Index. The main navigation menu has links for Topics and Geography, Statistical Products and Data, National Transportation Library, Newsroom, and About BTS. A breadcrumb trail shows the user is at BTS > TranStats. On the left, there's a sidebar with the TranStats logo, a search bar, and sections for Resources (Database Directory, Glossary, Upcoming Releases, Data Release History) and Data Tools (Analysis, Table Profile, Table Contents, Carrier Release Status, Data Tables, Database Profile, Databases). The main content area is titled "Air Carriers : T-100 Domestic Segment (All Carriers)". It features a table of fields with descriptions and checkboxes for selecting all fields or missing values. The table is organized into sections: Summaries, Data Tools, and Carrier. The "Carrier" section includes a table with rows for UniqueCarrier and AirlineID, each with a "Get Lookup Table" link.

Field Name	Description	Support Table
Summaries		
DepScheduled	Departures Scheduled	
DepPerformed	Departures Performed	
Payload	Available Payload (pounds)	
Seats	Available Seats	
Passengers	Non-Stop Segment Passengers Transported	
Freight	Non-Stop Segment Freight Transported (pounds)	
Mail	Non-Stop Segment Mail Transported (pounds)	
Distance	Distance between airports (miles)	
RampTime	Ramp to Ramp Time (minutes)	
AirTime	Airborne Time (minutes)	
Carrier		
UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.	Get Lookup Table
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline ( <i>carrier</i> ) is defined as	Get Lookup Table

# 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

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

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

Weighted Directed Graph

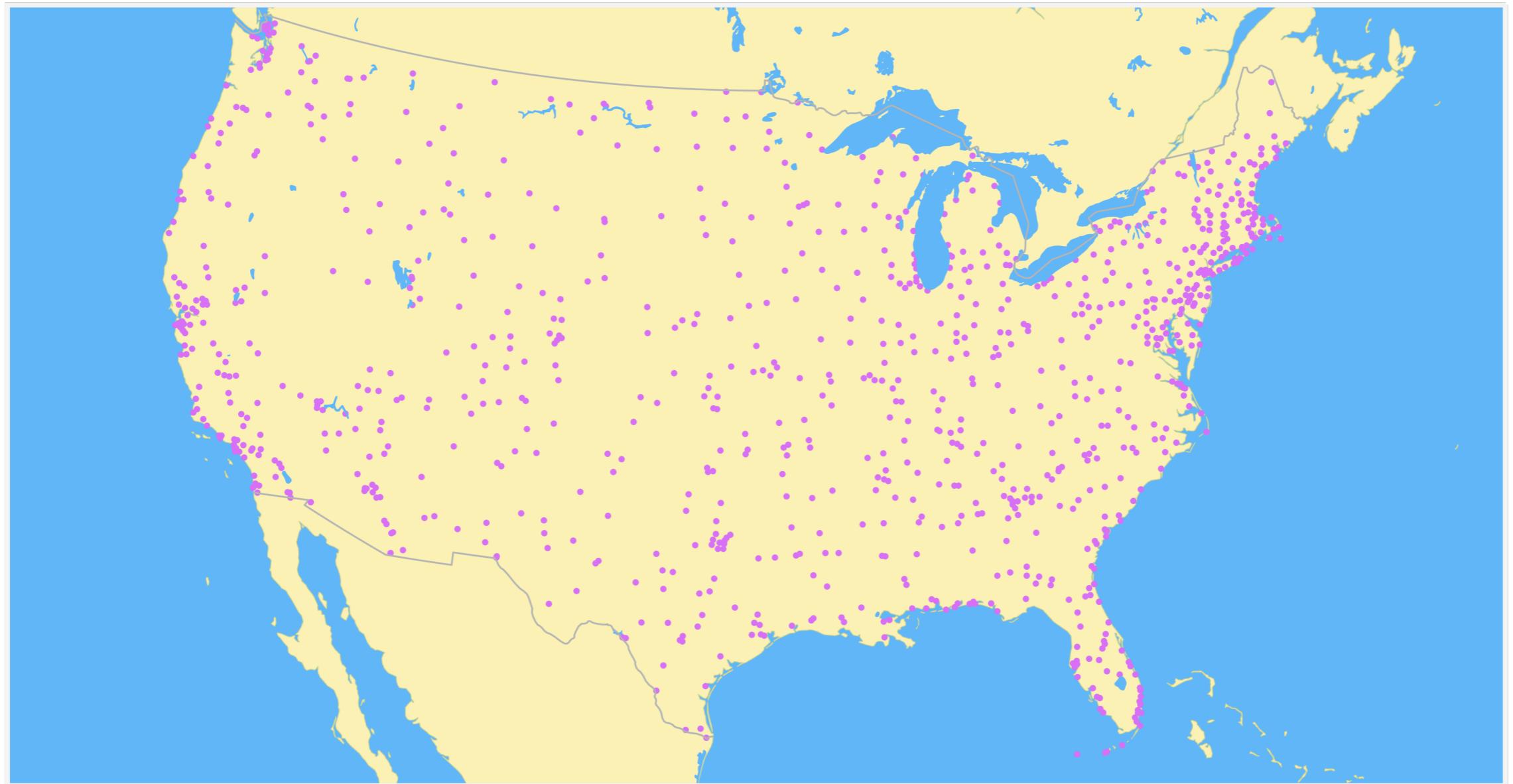
# Nodes

- Nodes are airports



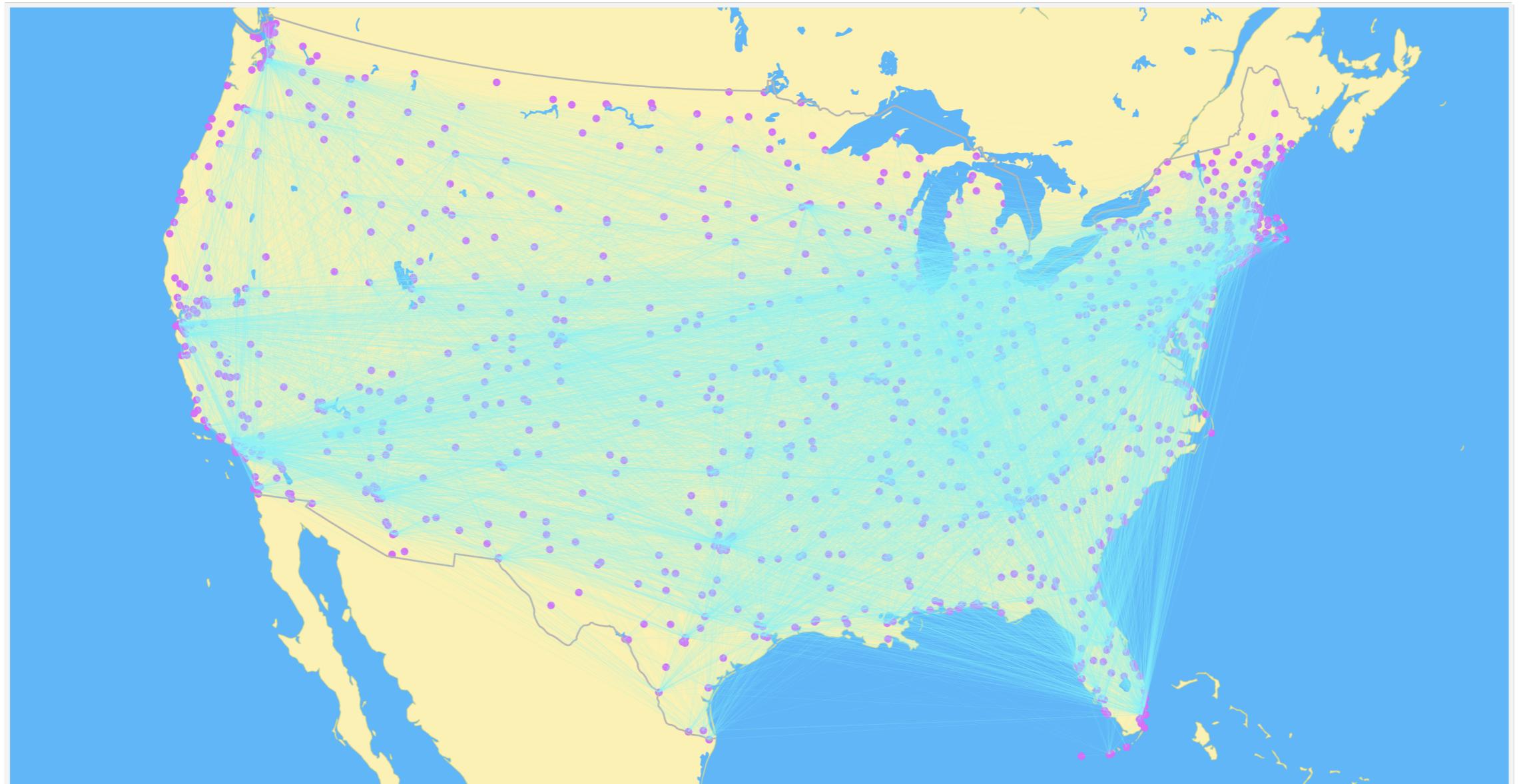
# Nodes

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



# Edges

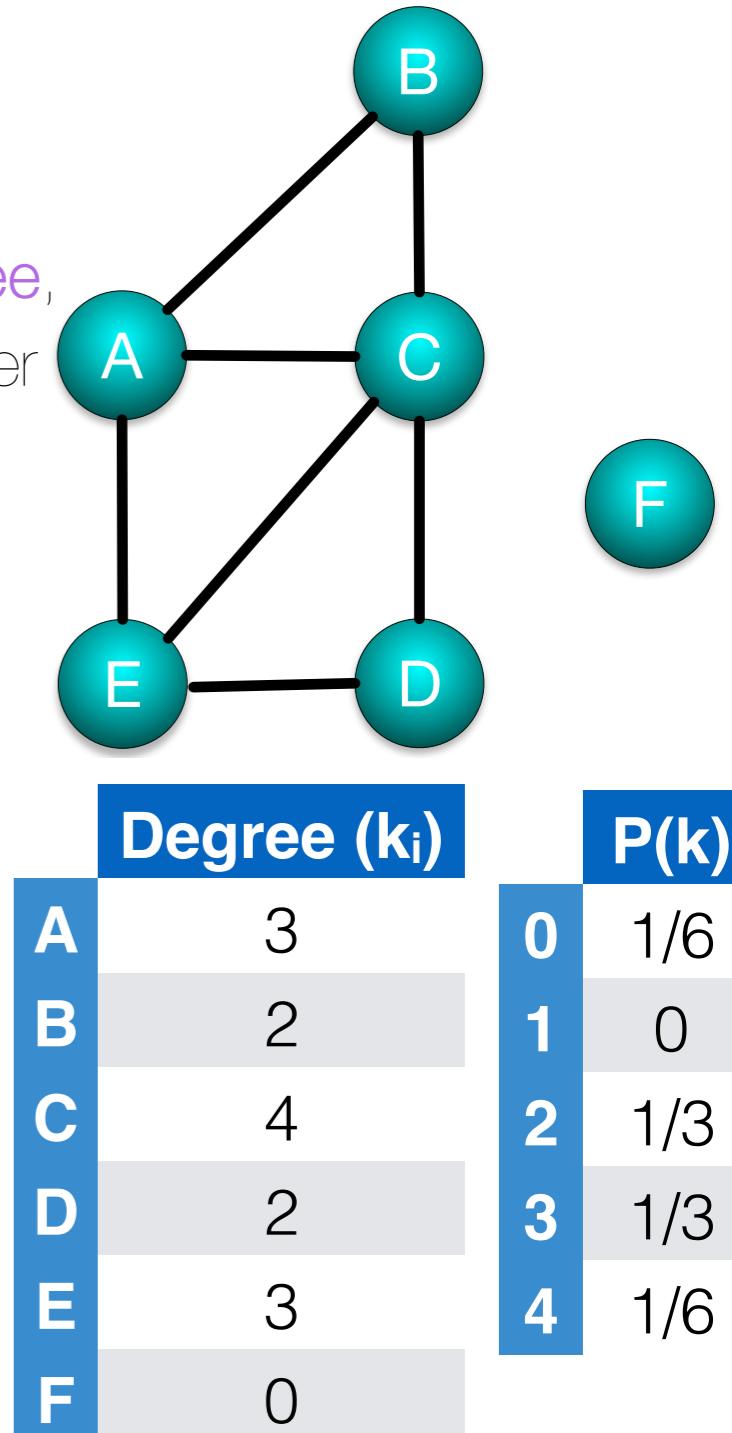
- Edges are flights
- We aggregate all the flights going from airport A to airport B



# Degrees

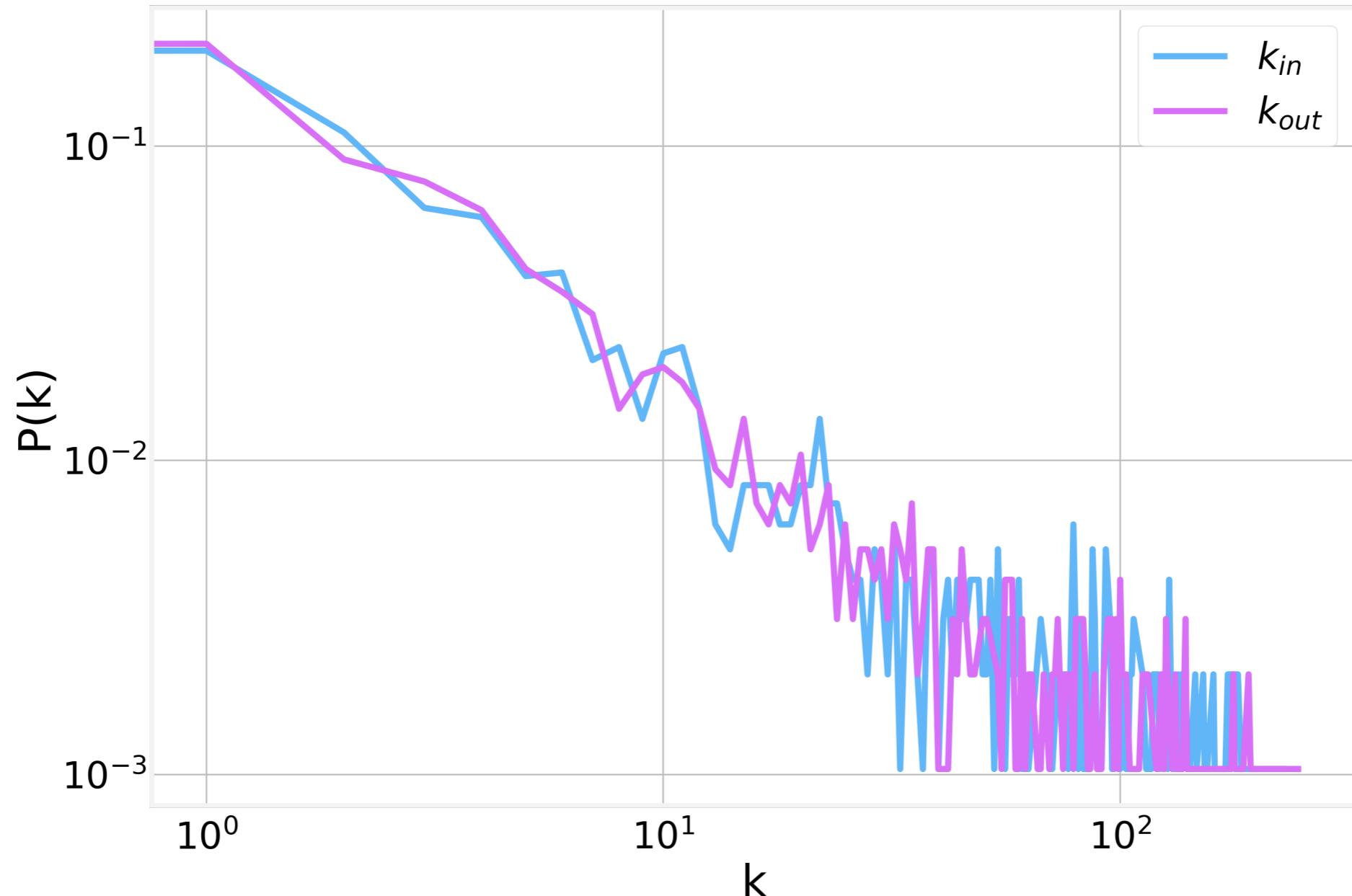
- The most fundamental property of a node is its degree,  $k$  - the number of connections of a node
- In the case of **directed networks**, we distinguish between **in-degree**,  $k_{in}$  (number of incoming connections) and **out-degree**  $k_{out}$  (number of outgoing connections).
- **Degree distribution** - The relative frequency of each degree value:

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



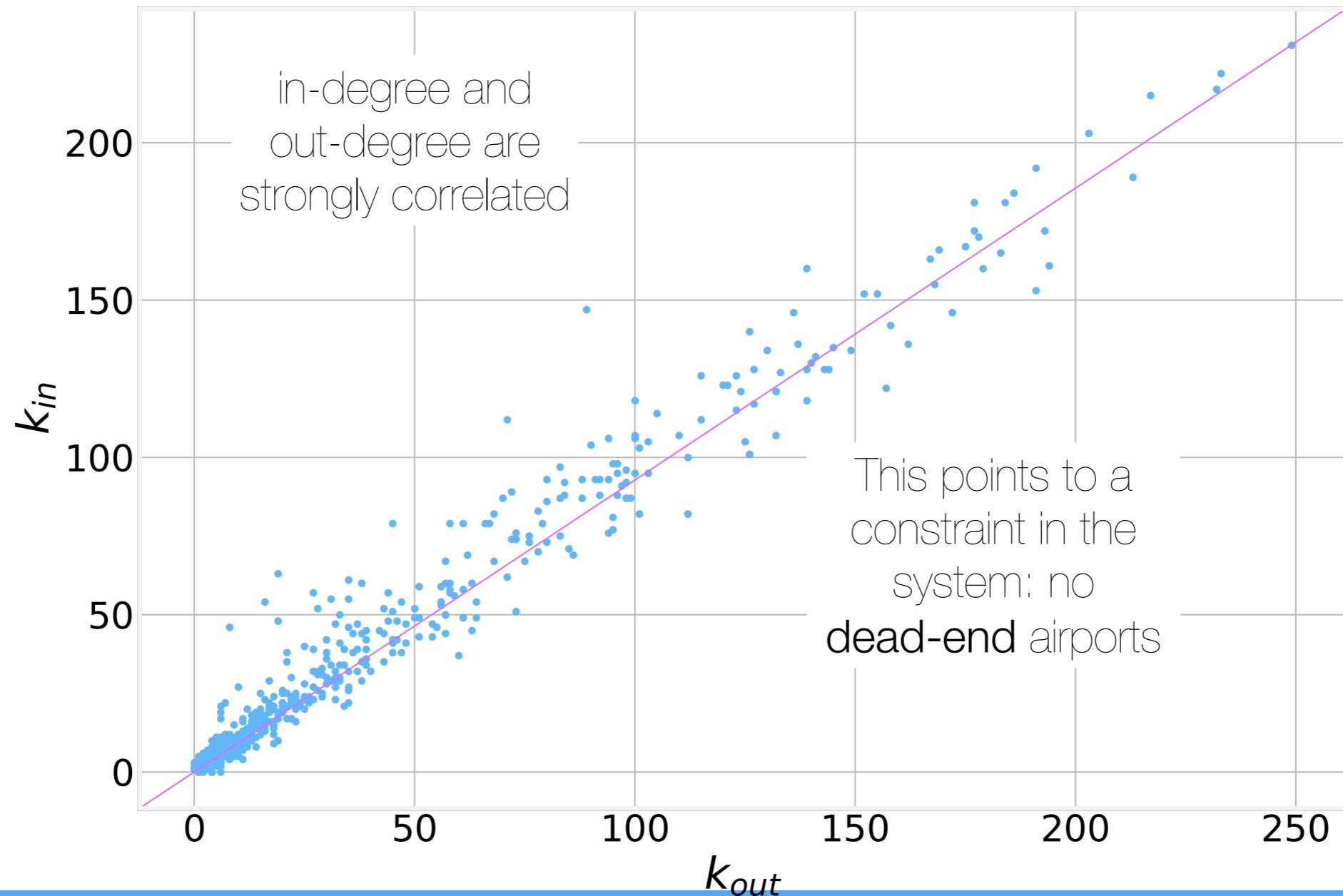
# Degree Distribution

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



# in-out degree correlations

- Different networks will display different behaviors



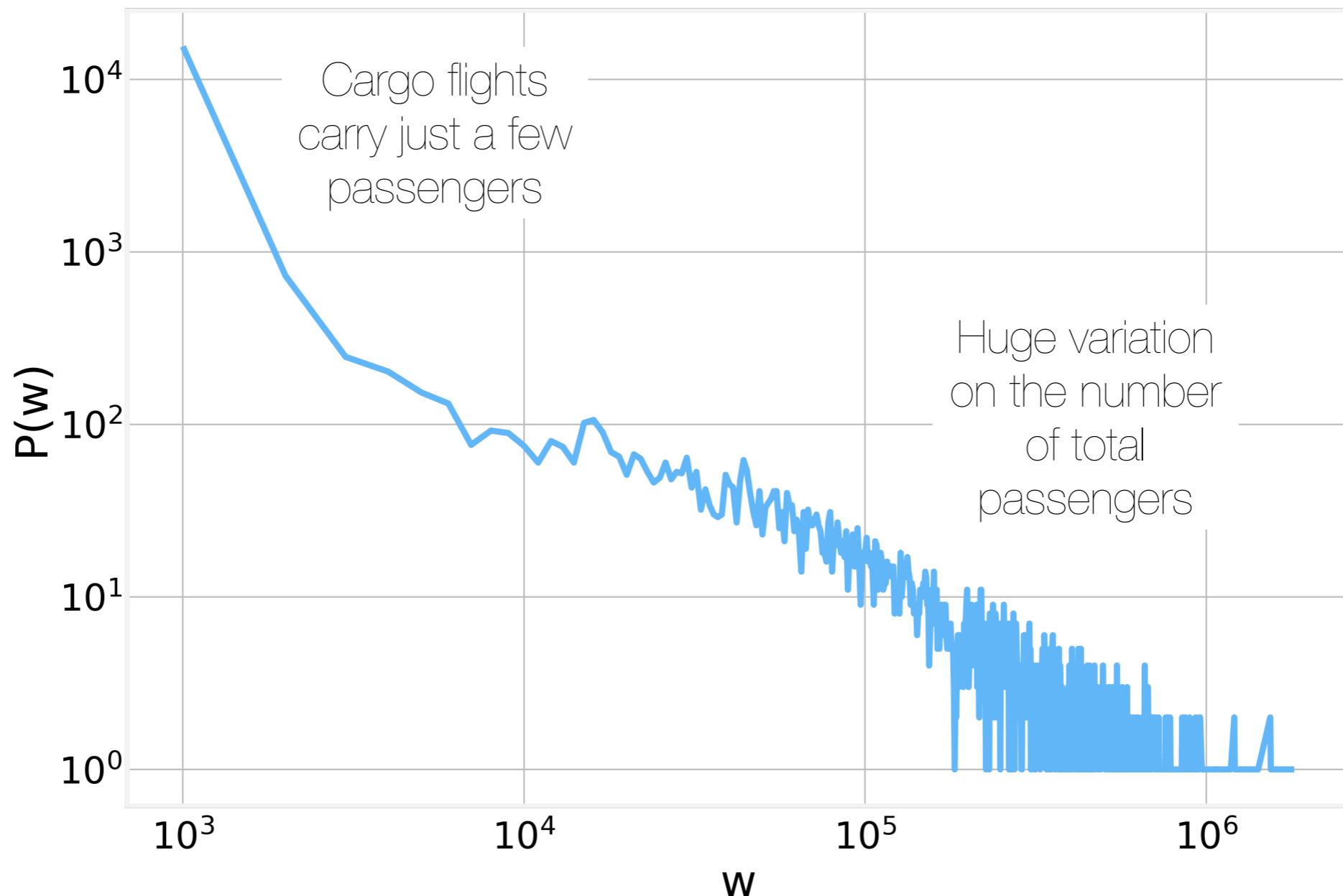
# Edge Weight

- 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!



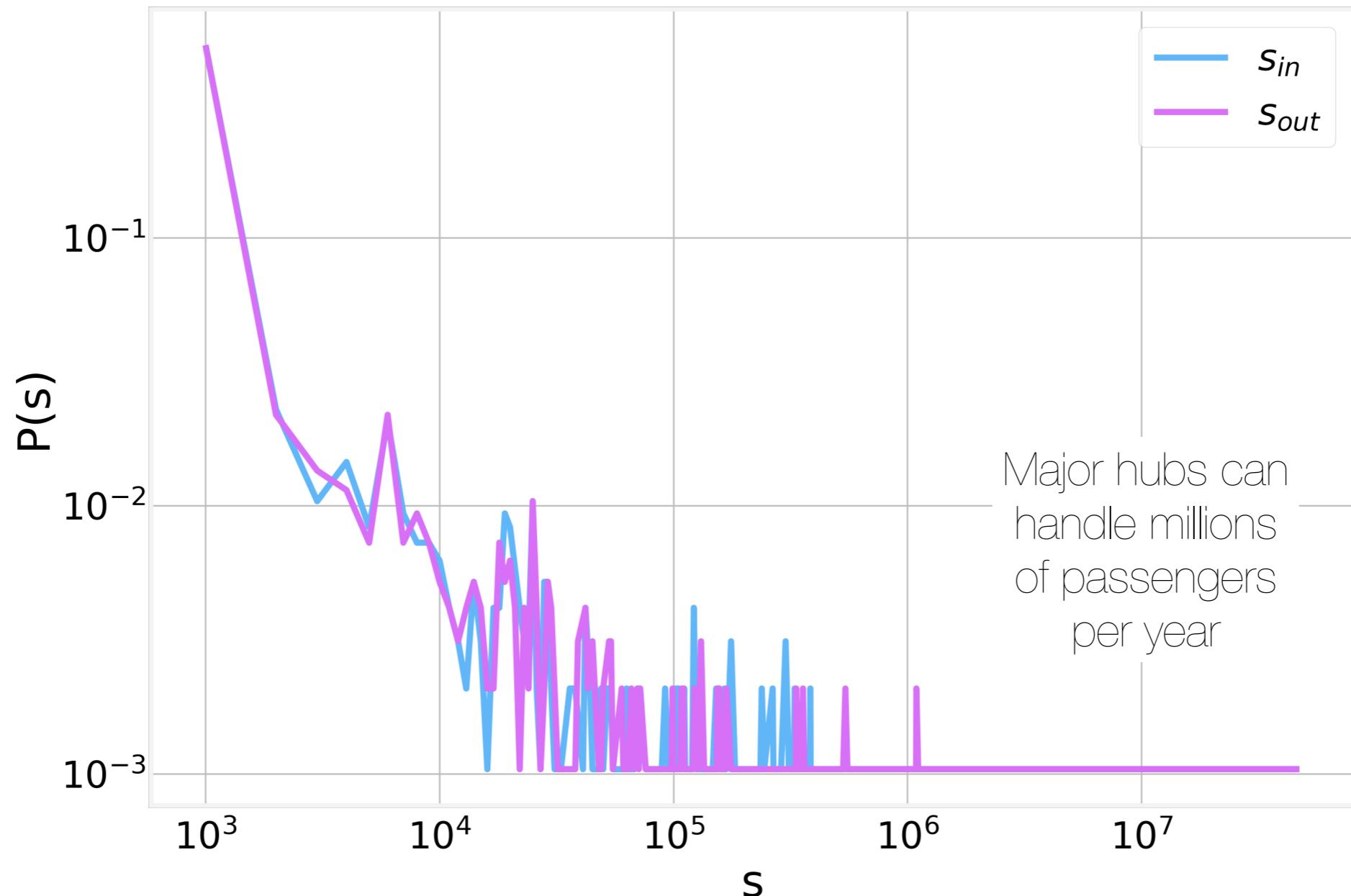
# Edge Weight

- 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



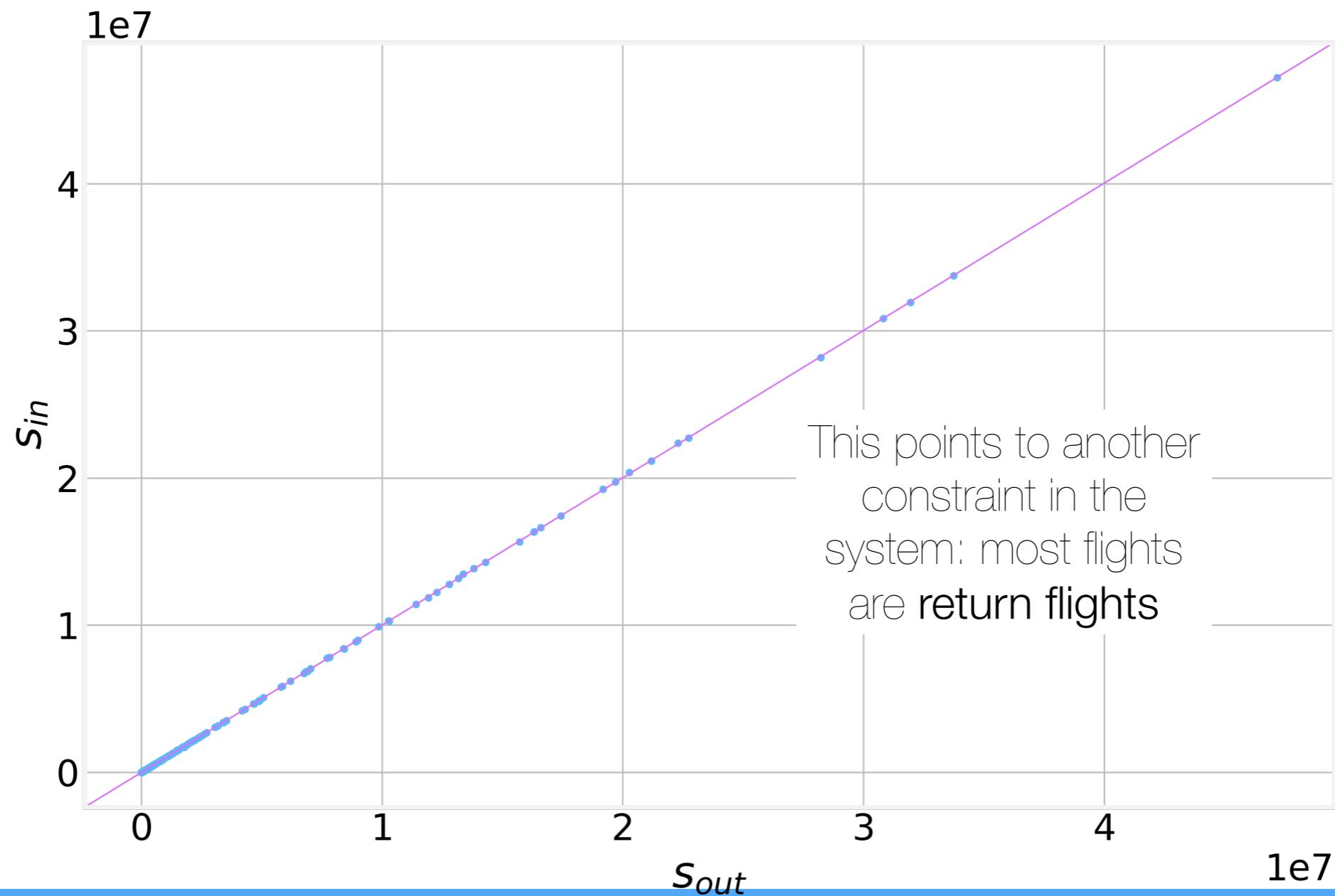
# Strength

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



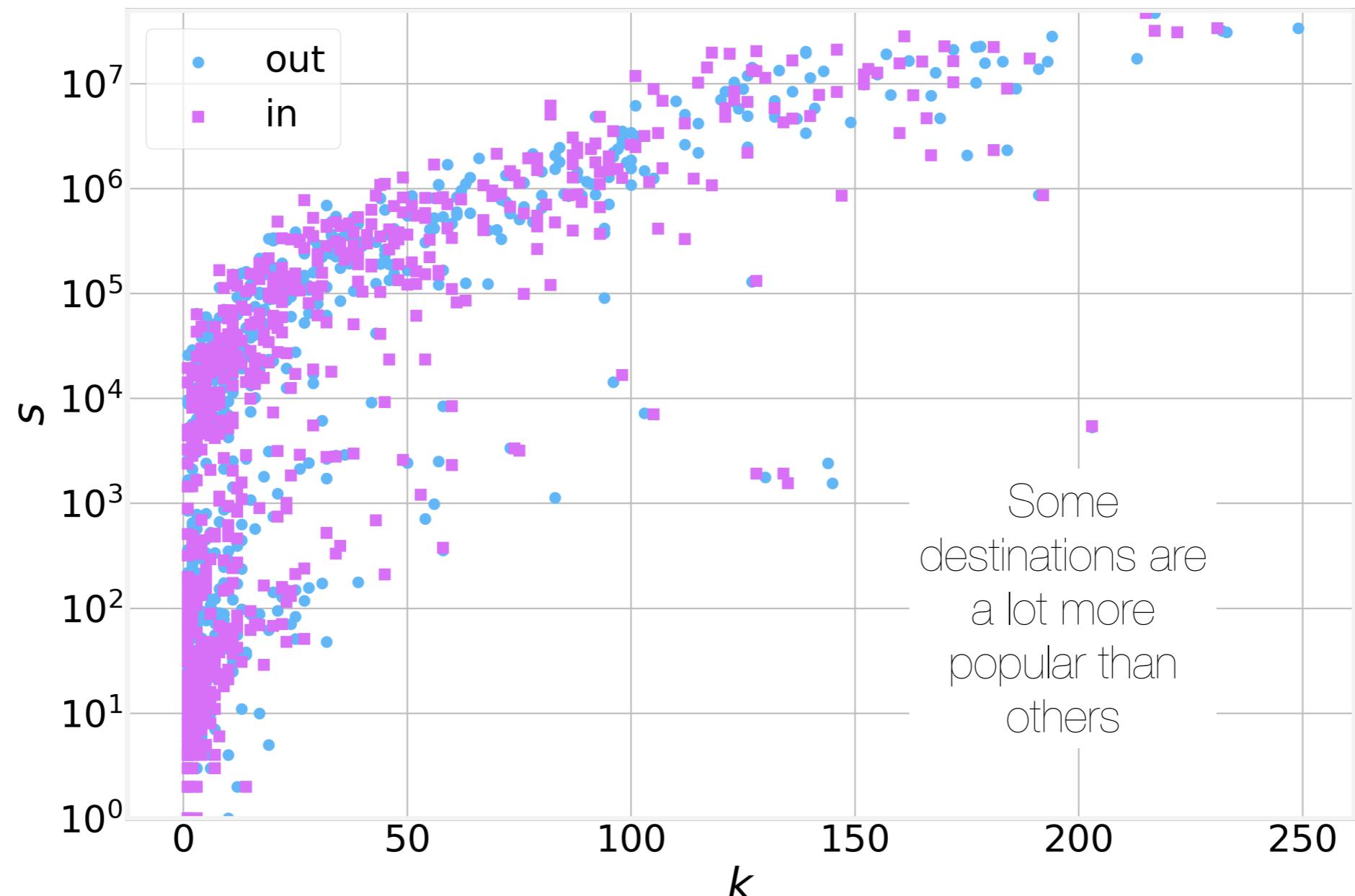
# Strength

- 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



WIKIPEDIA

The Free Encyclopedia

Article

Talk

Read

Edit

View history

Search Wikipedia



## 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.<sup>[5]</sup>

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.<sup>[6][7]</sup> Since then, it has grown to over two million registered users.<sup>[8]</sup> 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.<sup>[9]</sup> 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.<sup>[10]</sup> OpenStreetMap data has been favourably compared with proprietary datasources,<sup>[11]</sup> although as of 2009 data quality varied across the world.<sup>[12][13][needs update]</sup>

### Contents [hide]

1 History
2 Map production
2.1 Software for editing maps
2.2 Contributors
2.3 Surveys and personal knowledge
2.4 Street-level image data
2.5 Government data
2.6 Route planning
3 Usage
3.1 Software for viewing maps
3.2 Humanitarian aid
3.3 Scientific research
4 "State of the Map" annual conference
5 Legal aspects
5.1 Licensing terms
5.2 Commercial data contributions
6 Operation
6.1 Data format
6.2 Data storage
6.3 Popular services
7 See also
8 References
9 Further reading
10 External links

### OpenStreetMap



OpenStreetMap's logo

#### Screenshot [show]

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

# osmnx

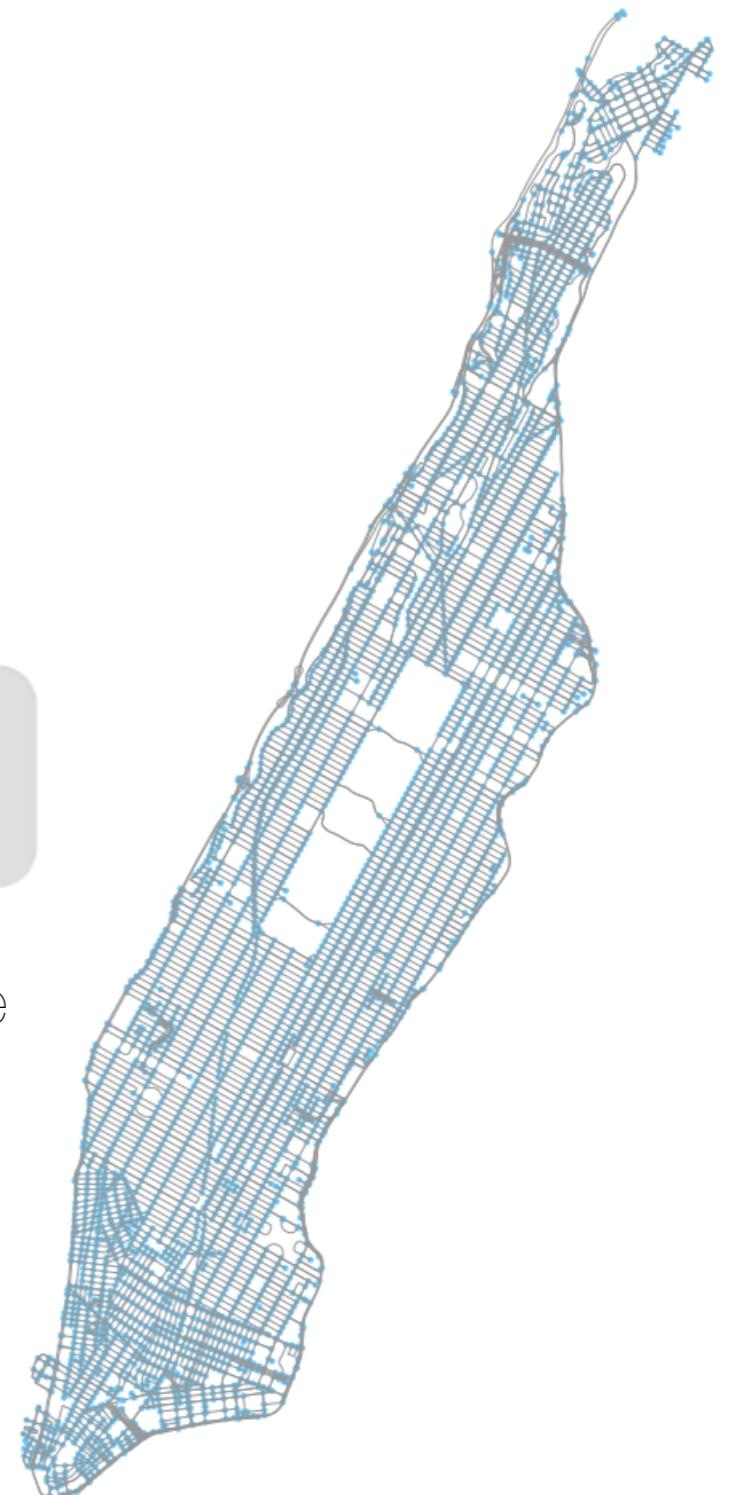
<https://github.com/gboeing/osmnx>

- 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')
```

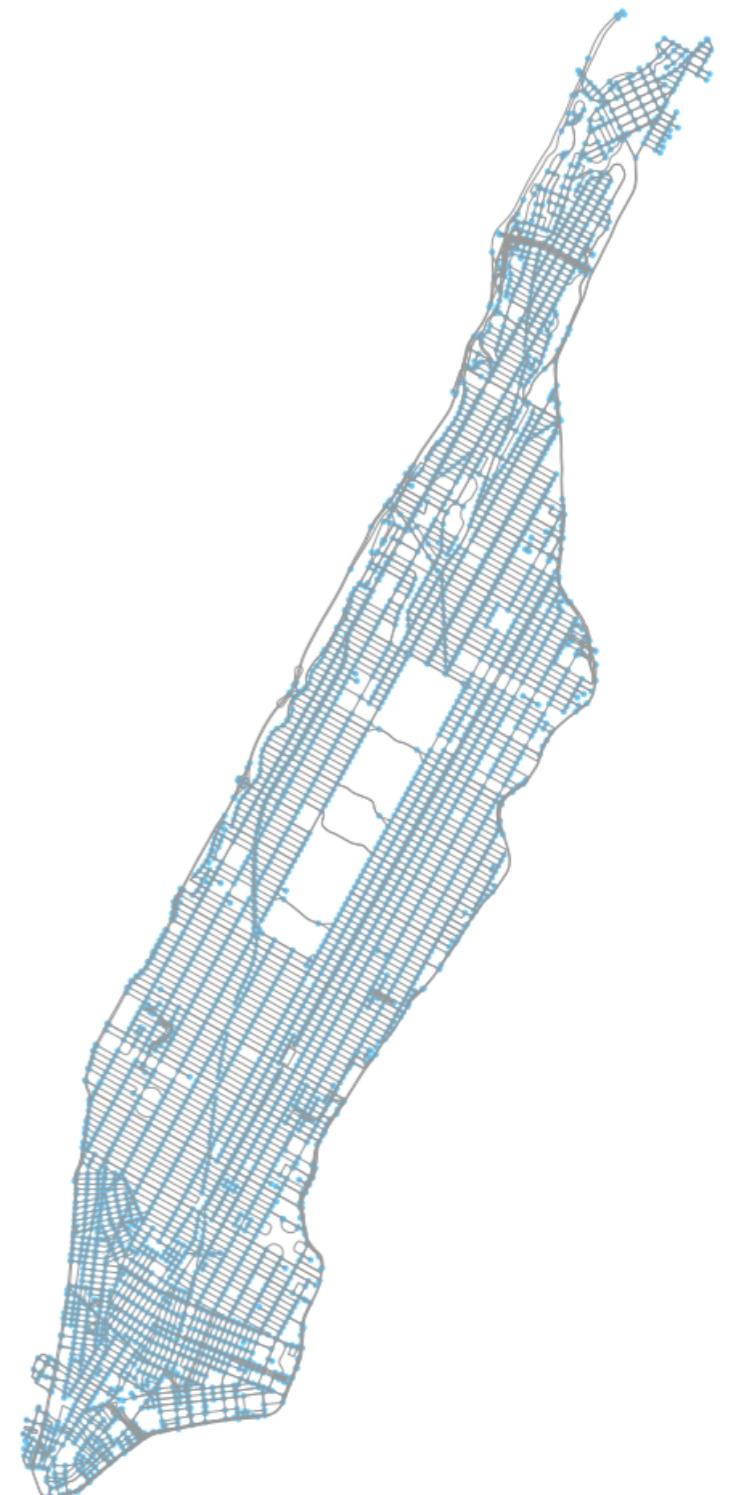
- Returns a **NetworkX** object that provides us with a simple interface to manipulate the graph



# 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](https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

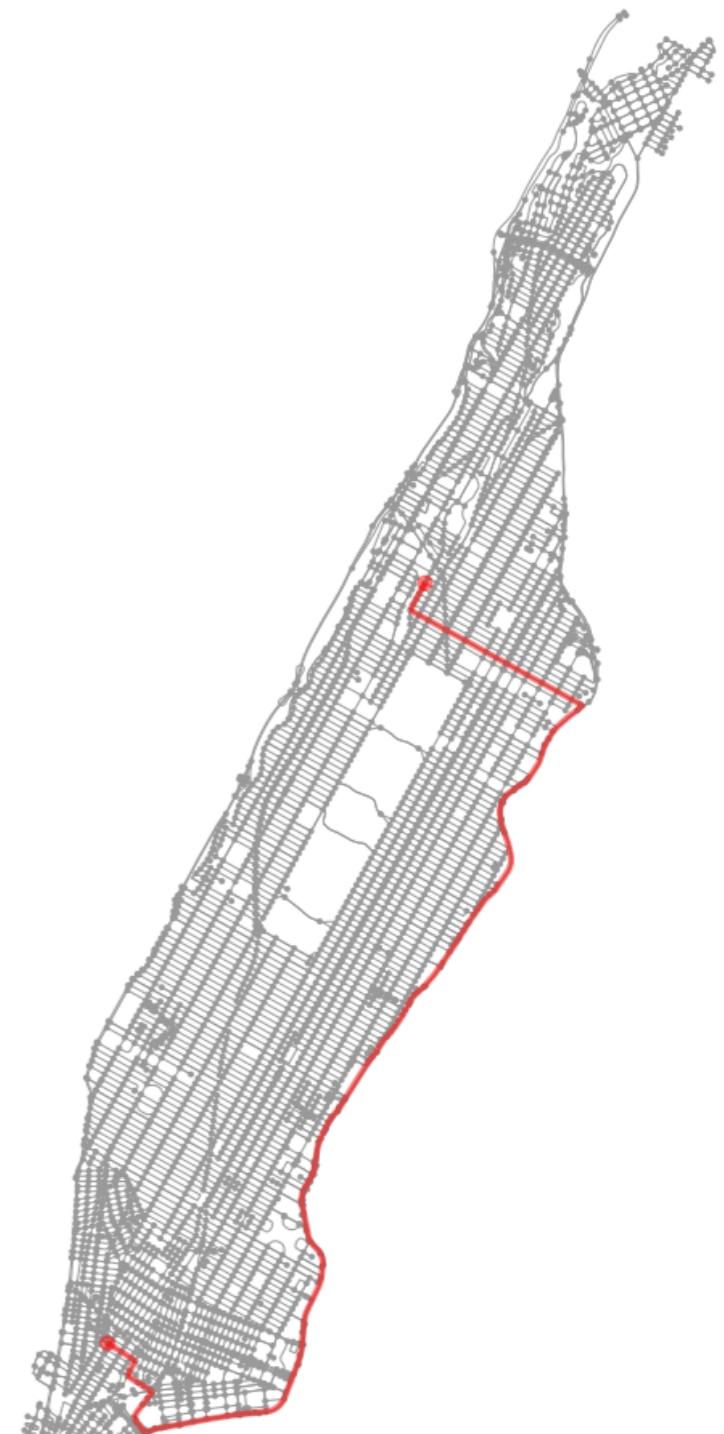
# Dijkstra's Algorithm

- At each step:
  - Find the node  $i$  with the **smallest current distance** to the origin
  - Update the distance to each of its nearest neighbors,  $j$ 
$$dist(node_j) = \min \left[ dist(node_j), dist(node_i) + w_{ij} \right]$$
  - 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](https://en.wikipedia.org/wiki/Dijkstra%27s_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**

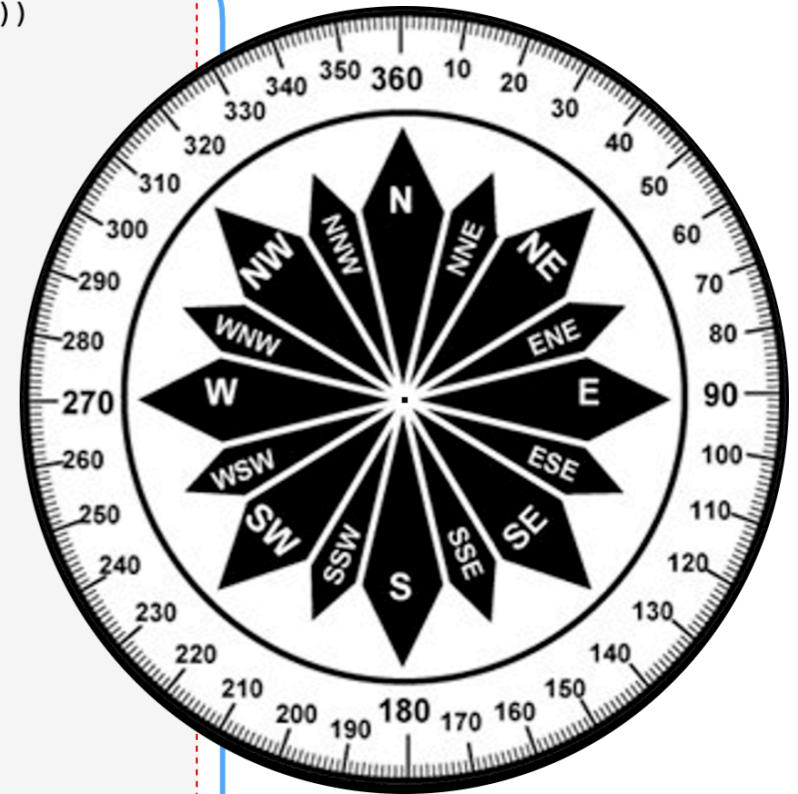


[en.wikipedia.org/wiki/Dijkstra's\\_algorithm](https://en.wikipedia.org/wiki/Dijkstra's_algorithm)

# 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

```
1 print("Drive %1.1fm down %s" % (directions[0][1], directions[0][0]))
2
3 for i in range(1, len(directions)):
4     change = directions[i][2]-directions[i-1][2]
5
6     # Find the shortest difference between the two bearings
7     if change < -180:
8         change += 360
9     elif change > 180:
10        change -= 360
11
12     if np.abs(change) < 5:
13         msg = 'Continue onto'
14     elif change > 0:
15         if change > 25:
16             msg = 'Turn right onto'
17         else:
18             msg = 'Turn slight right onto'
19     elif change < 0:
20         if change < -25:
21             msg = 'Turn left onto'
22         else:
23             msg = 'Turn slight left onto'
24
25     print('%s %s and drive for %1.1fm' % (msg, directions[i][0],
26                                         directions[i][1]))
27
28 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

Cryptos: 11,746 Exchanges: 405 Market Cap: \$2,079,319,605,905 24h Vol: \$227,858,546,811 Dominance: BTC: 42.5% ETH: 19.5% ETH Gas: 208 Gwei ▾

English ▾ USD ▾

**CoinMarketCap** Cryptocurrencies Exchanges NFT Portfolio Watchlist Calendars Products Learn

Log In Sign up Search /

Want to take a sneak peek of our enhanced homepage? Yes, switch to new home Maybe later

### Today's Cryptocurrency Prices by Market Cap

The global crypto market cap is \$2.08T, a **-11.83%** decrease over the last day. [Read More](#)

News of the Day Sept. 7: Bitcoin Crashes! Free Airdrops Join \$21,000 SHREW Airdrop!

#	Name	Price	24h %	7d %	Market Cap	Volume(24h)	Circulating Supply	Last 7 Days
1	Bitcoin BTC	\$46,927.64	-9.30%	-0.90%	\$881,962,716,418	\$65,141,664,514 1,389,263 BTC	18,809,437 BTC	
2	Ethereum ETH	\$3,438.36	-12.78%	+0.73%	\$403,801,281,682	\$37,572,720,761 10,926,484 ETH	117,429,028 ETH	
3	Cardano ADA	\$2.40	-15.33%	-13.53%	\$76,700,799,428	\$10,901,856,290 4,552,056,680 ADA	32,026,324,425 ADA	

# Kaggle Bitcoin Transaction Dataset

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

The screenshot shows the Kaggle dataset page for the "Bitcoin Partial Transaction Dataset". The page has a dark blue header with the title "Bitcoin Partial Transaction Dataset" in white. Below the header, there's a section for "XBlock" with a green icon, updated a year ago (Version 1). The main navigation bar includes "Data" (which is underlined), "Tasks", "Code (2)", "Discussion", "Activity", and "Metadata". To the right of the navigation are buttons for "Download (2 GB)" and "New Notebook", along with a three-dot menu. Below the navigation, there are two sections: "Usability 4.7" and "Tags currencies and foreign exchange". The main content area is titled "Description" and contains several paragraphs of text about the dataset, its snapshots, and its use for mixing service detection. It also mentions the file structure and provides a link for more details.

**Bitcoin Partial Transaction Dataset**

XBlock • updated a year ago (Version 1)

Data Tasks Code (2) Discussion Activity Metadata

Download (2 GB) New Notebook :

Usability 4.7 Tags currencies and foreign exchange

Description

The Bitcoin Partial Transaction Datasets contain three snapshots of Bitcoin transaction data for easier analysis, namely dataset12014111500000, dataset2201561500000 and dataset320161\_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, namely 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.

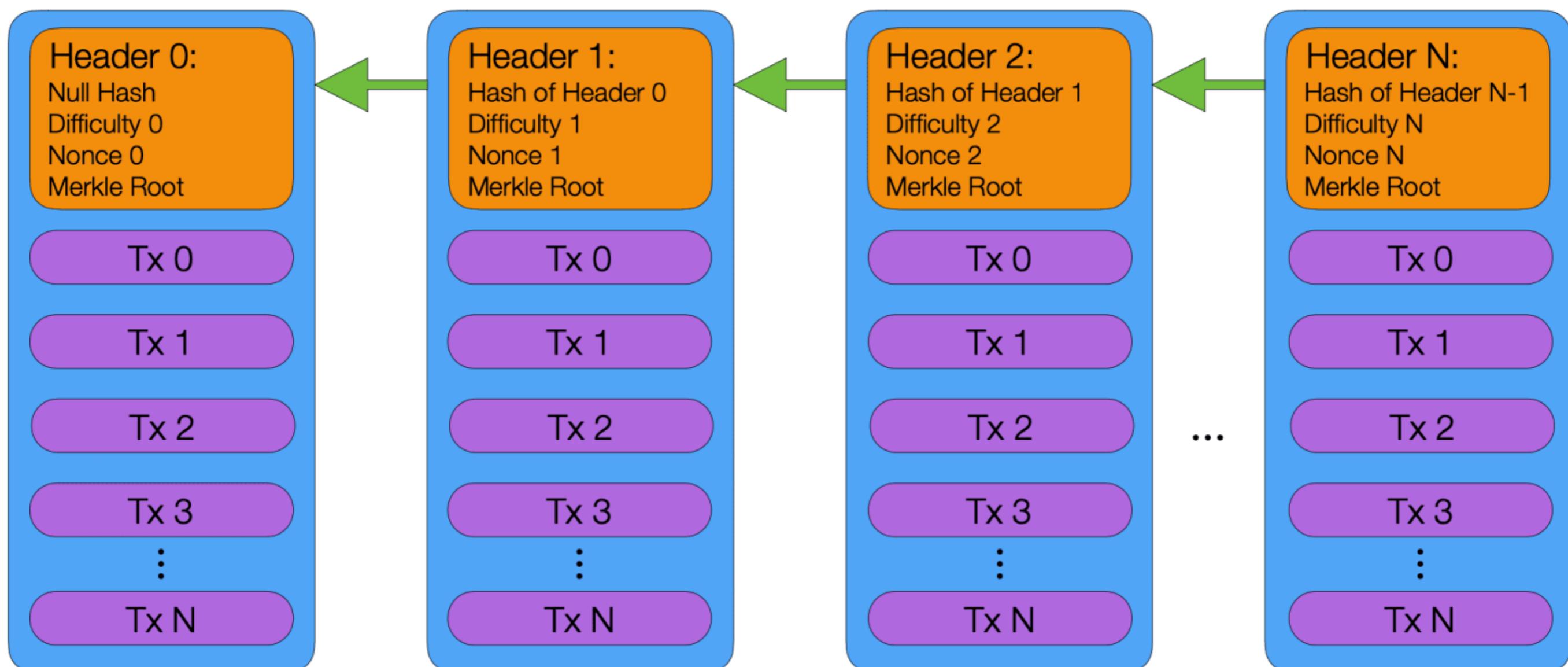
In our work, we study mixing service detection with this dataset. For further study, we can chase up users involved in criminal activities by analyzing users who take part in Bitcoin mixing.

The details of dataset12014111500000 are described below. The file structure of dataset2201561500000 and dataset320161\_1500000 are similar to EthereumG1. You can know more information from the README file.

For more details about blockchain dataset, please click [here](#).

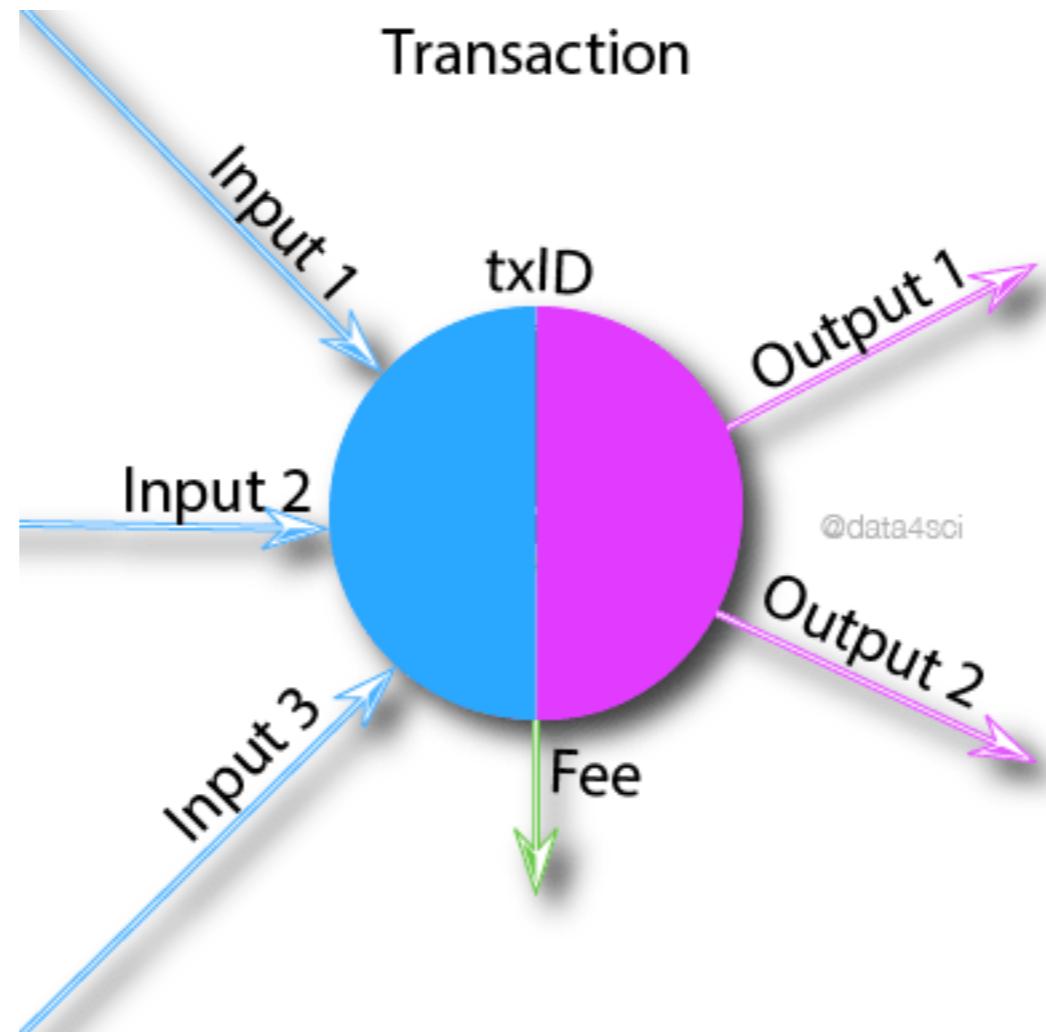
# Blockchain

- Each Block contains a



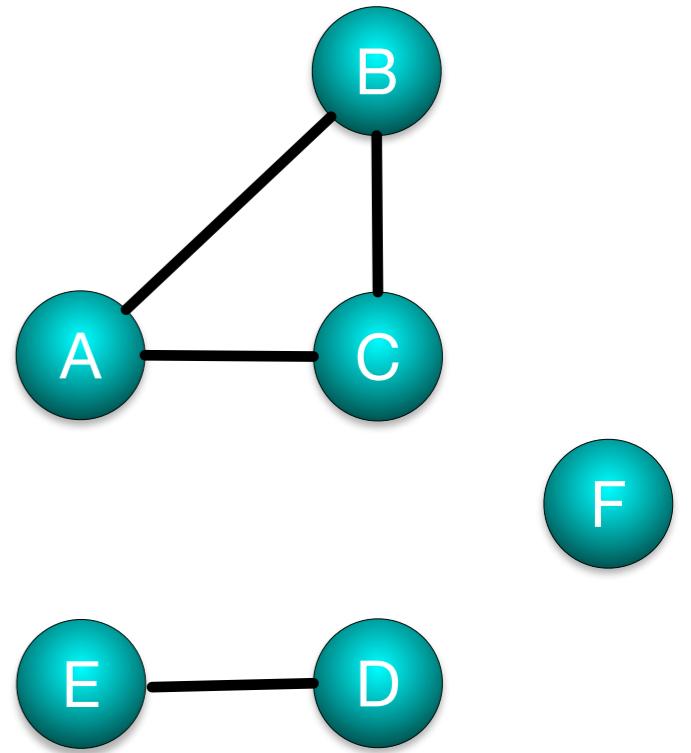
# Transactions

- 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



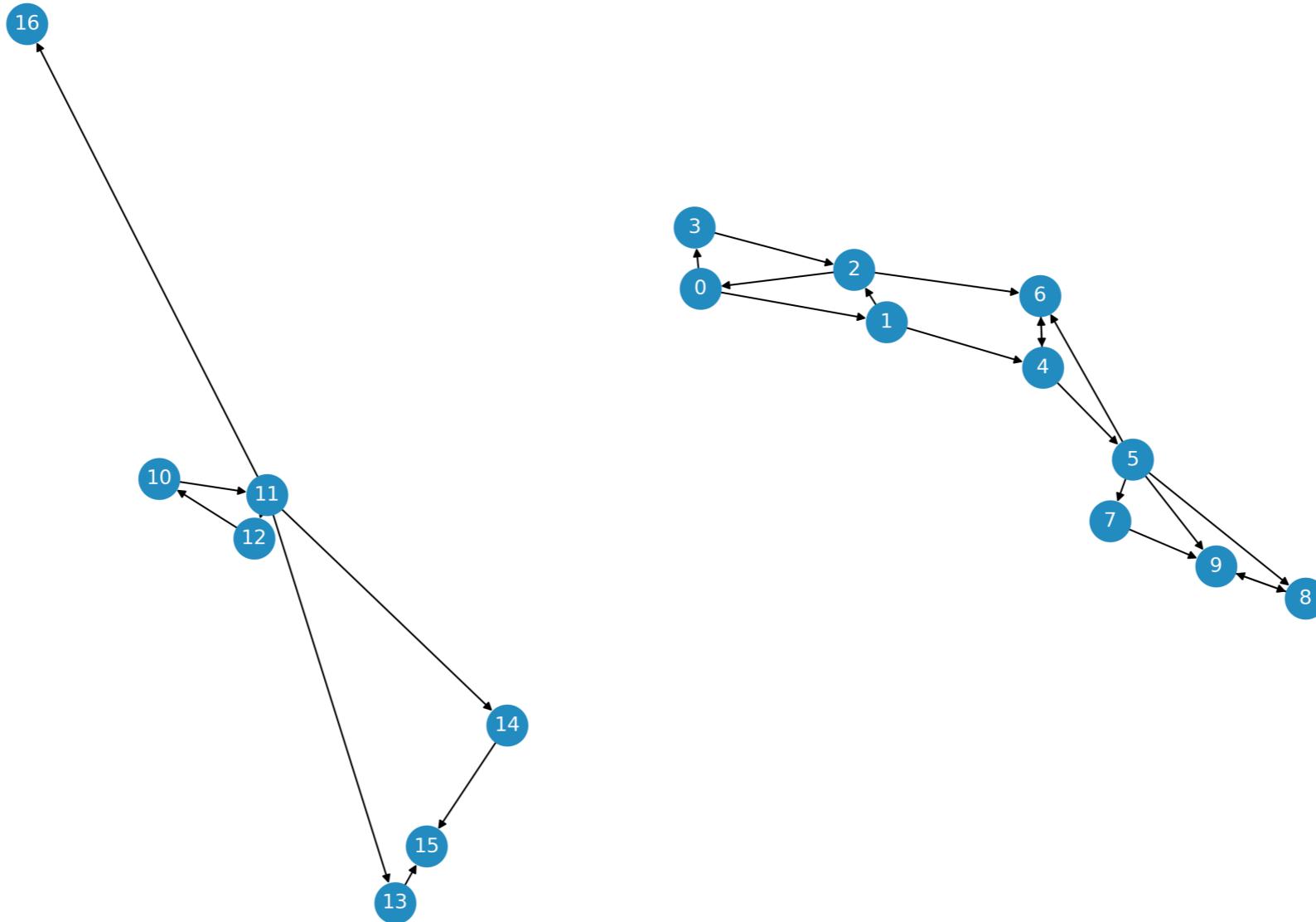
# Components

- 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



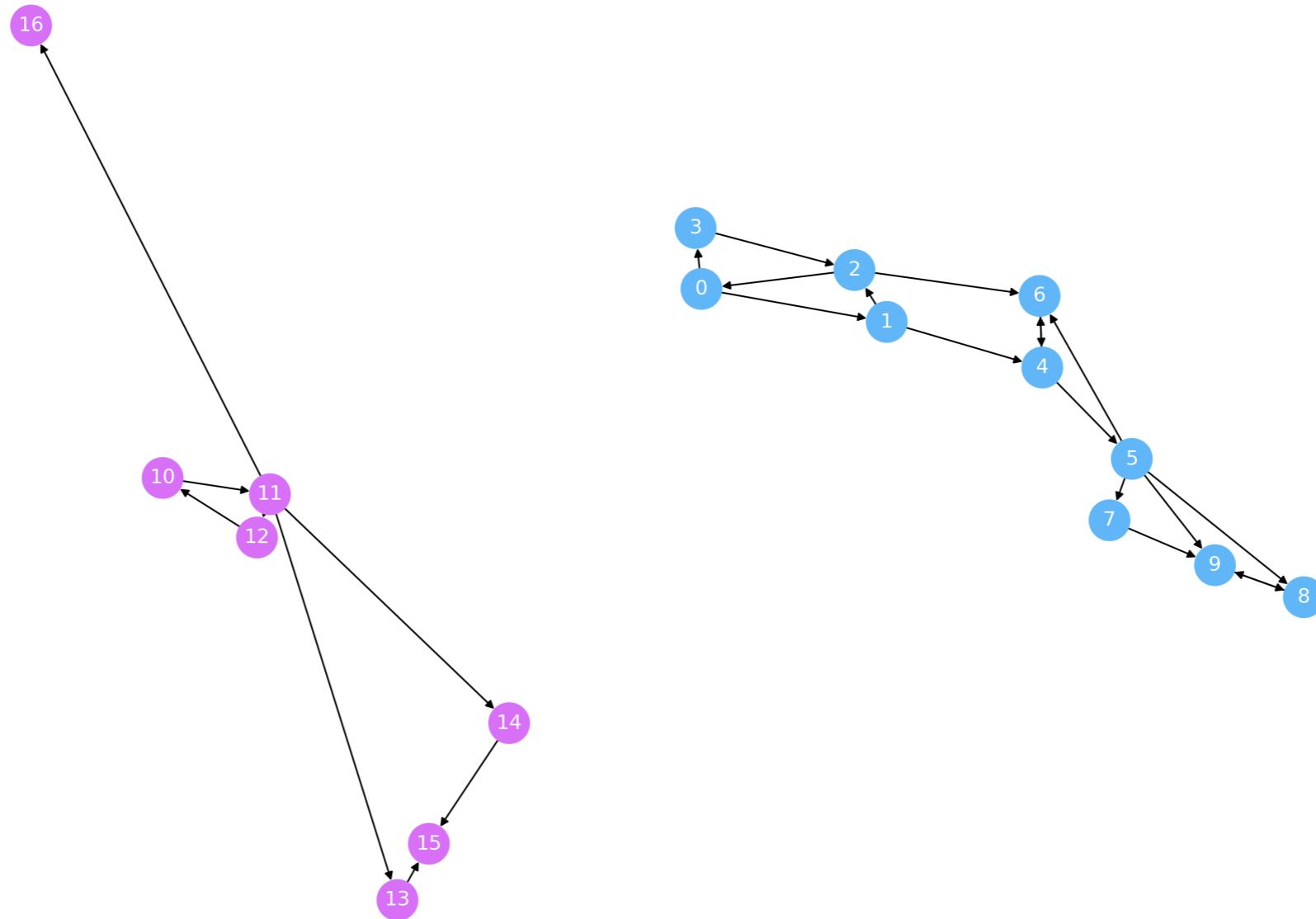
# Components

- Let us consider a simple network



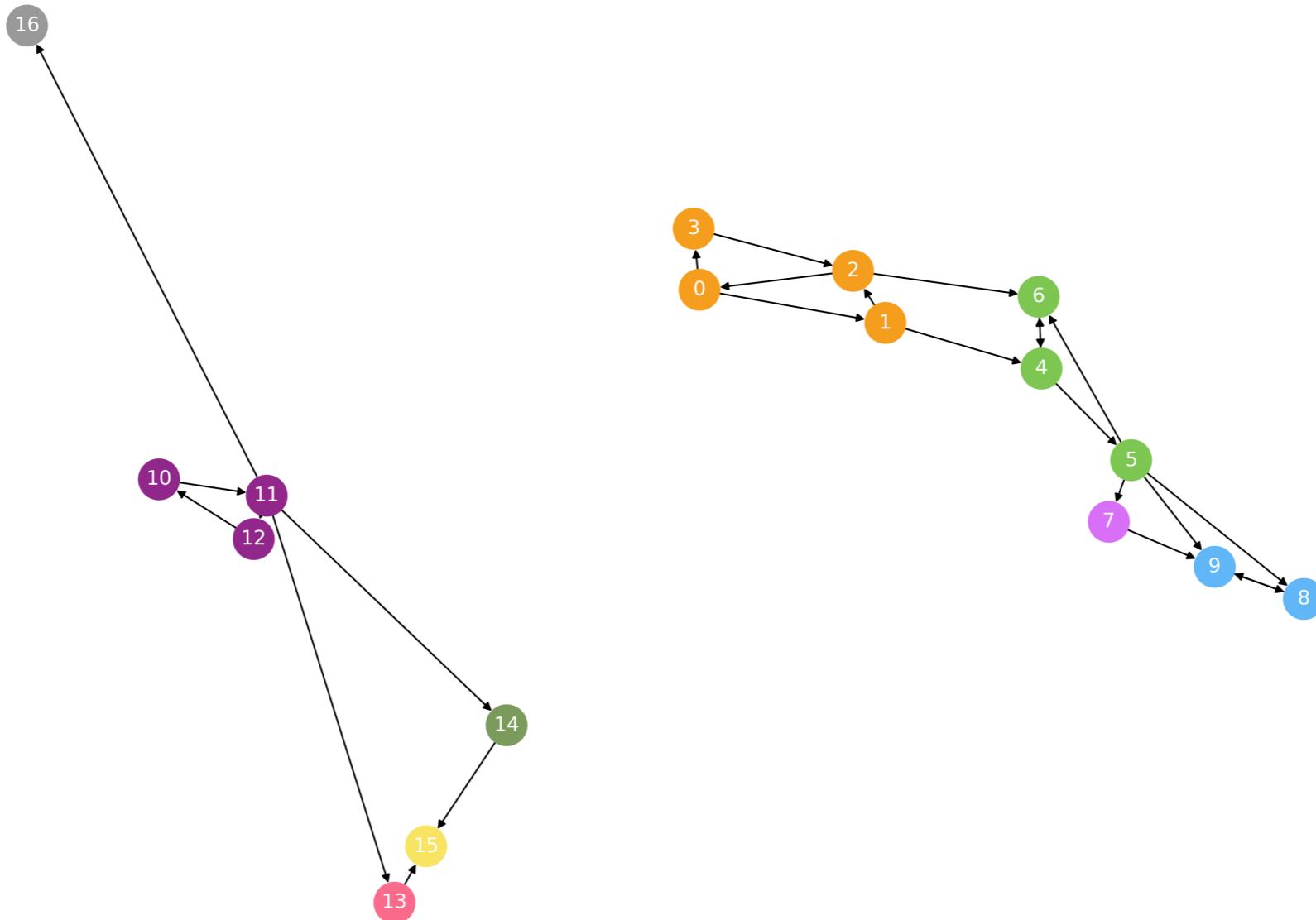
# Weakly Connected Components

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



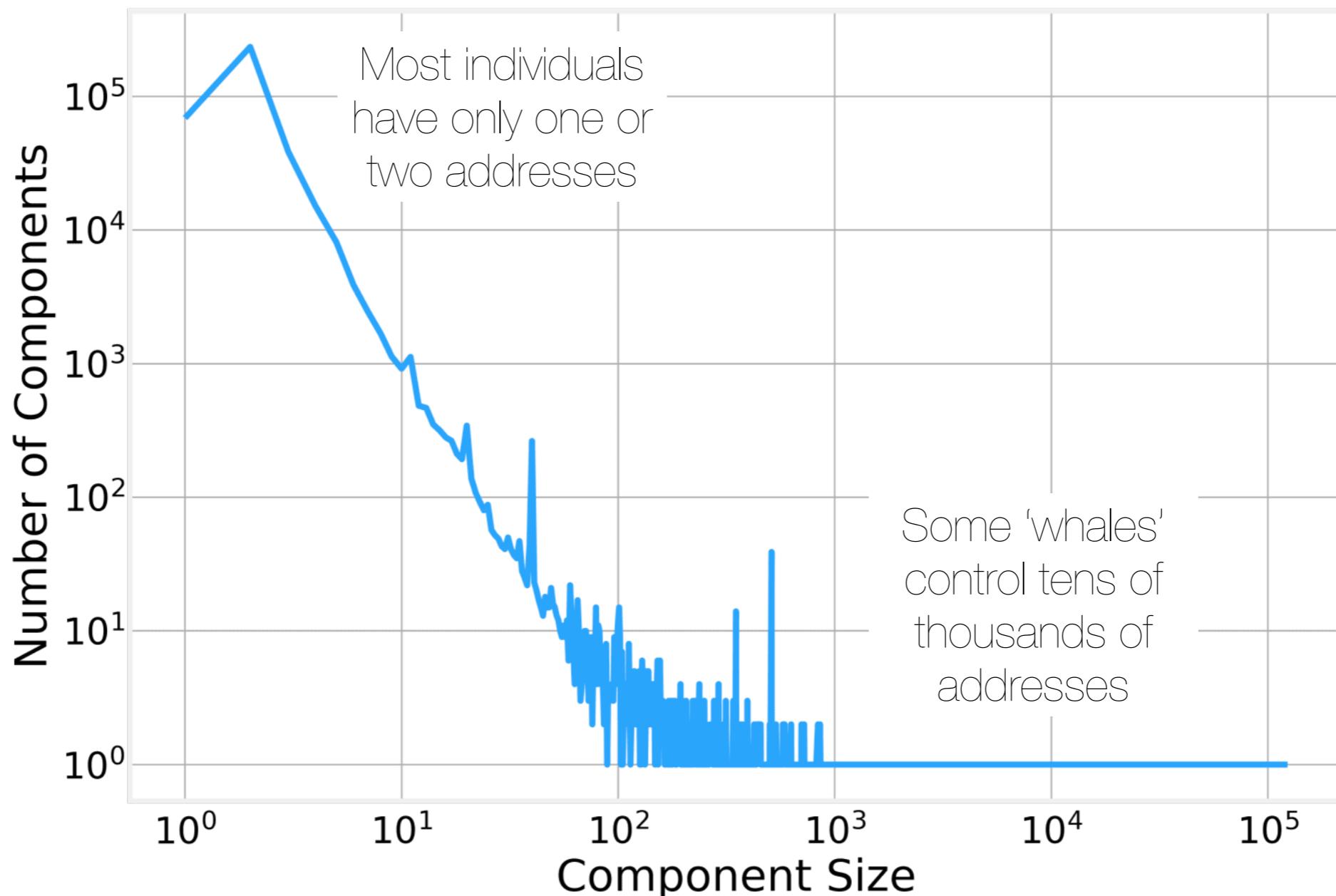
# Strongly Connected Components

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



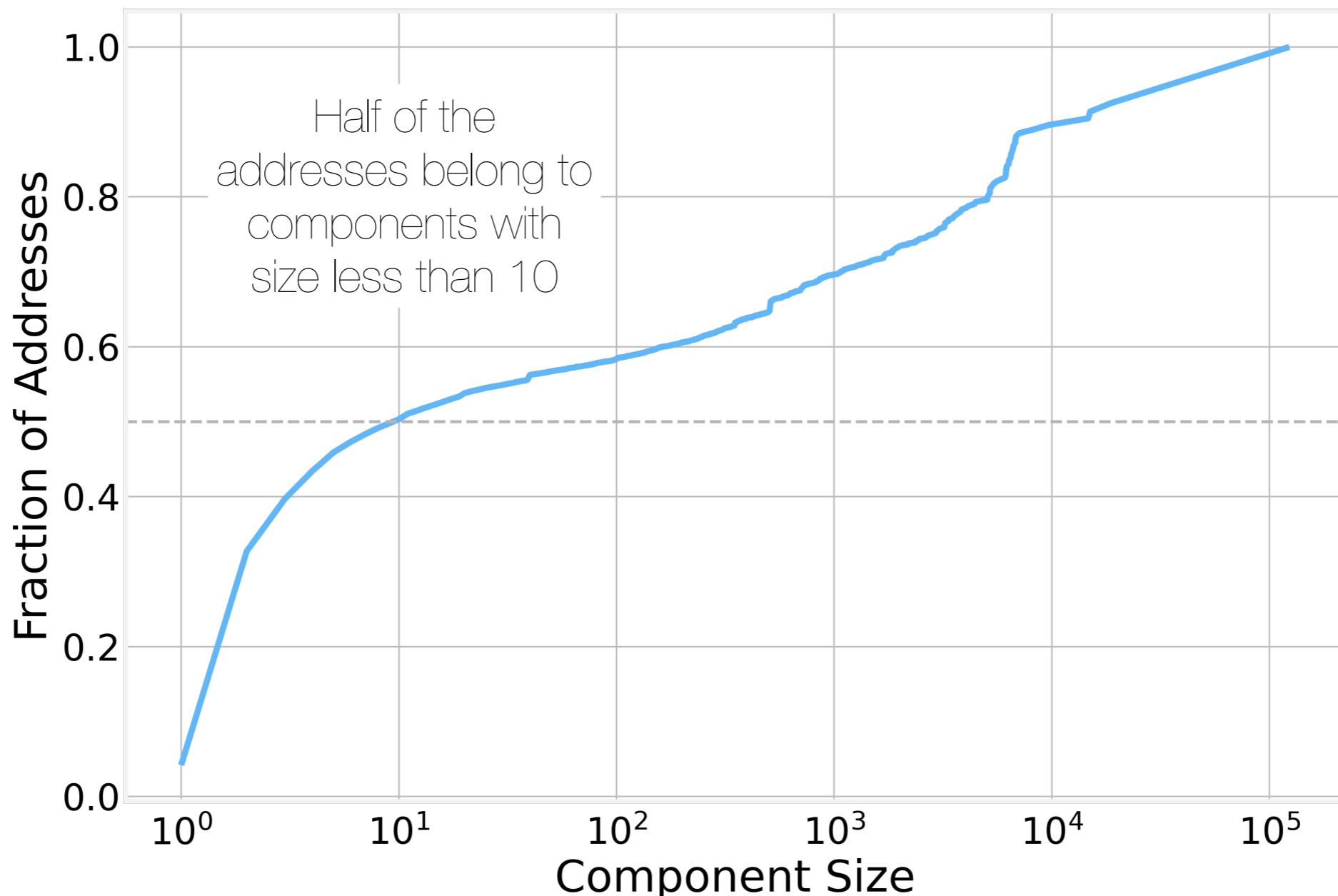
# Wallets

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



# Wallets

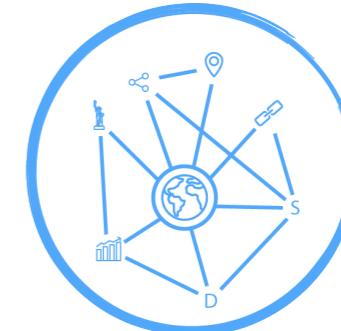
- Wallets are addresses controlled by the same individual
- We can identify them as the Weekly Connected Components in our transaction network





Code - Graph Connectivity  
<https://github.com/DataForScience/PyGotham2021>

# Thank you!



[graphs4sci.substack.com](https://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](http://www.data4sci.com)
- [Twitter](#): @data4sci
- [Email](#): [bgoncalves@data4sci.com](mailto:bgoncalves@data4sci.com)