Module II - Why do SNA in NetworkX

Drew Conway and Aric Hagberg

June 29, 2010

Agenda for Module II

Speed, Scalability & Graph Types

- Why speed and scalability matter
- Comparing `NetworkX` to other SNA tools
- What can be a "graph" in `NetworkX`

Agenda for Module II

Speed, Scalability & Graph Types

- Why speed and scalability matter
- Comparing `NetworkX` to other SNA tools
- What can be a "graph" in `NetworkX`

How `NetworkX` complements Python's scientific computing suite

- `SciPy/NumPy`
- `Matplotlib`
- GraphViz

Agenda for Module II

Speed, Scalability & Graph Types

- Why speed and scalability matter
- Comparing `NetworkX` to other SNA tools
- What can be a "graph" in `NetworkX`

How `NetworkX` complements Python's scientific computing suite

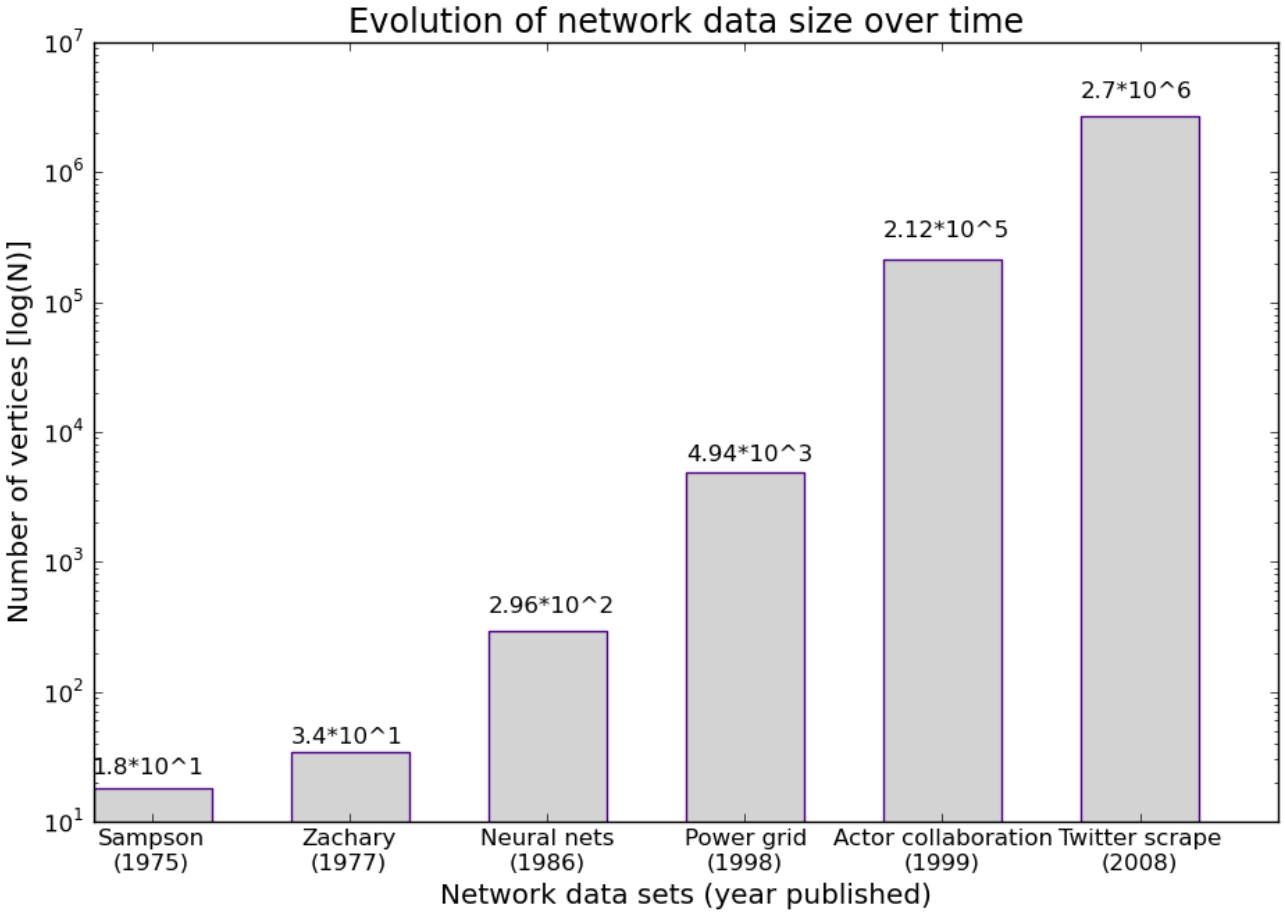
- `SciPy/NumPy`
- `Matplotlib`
- GraphViz

Getting data in and out of `NetworkX`

- I/O basics
- Pulling non-local data
  - Directly from the web
  - External databases

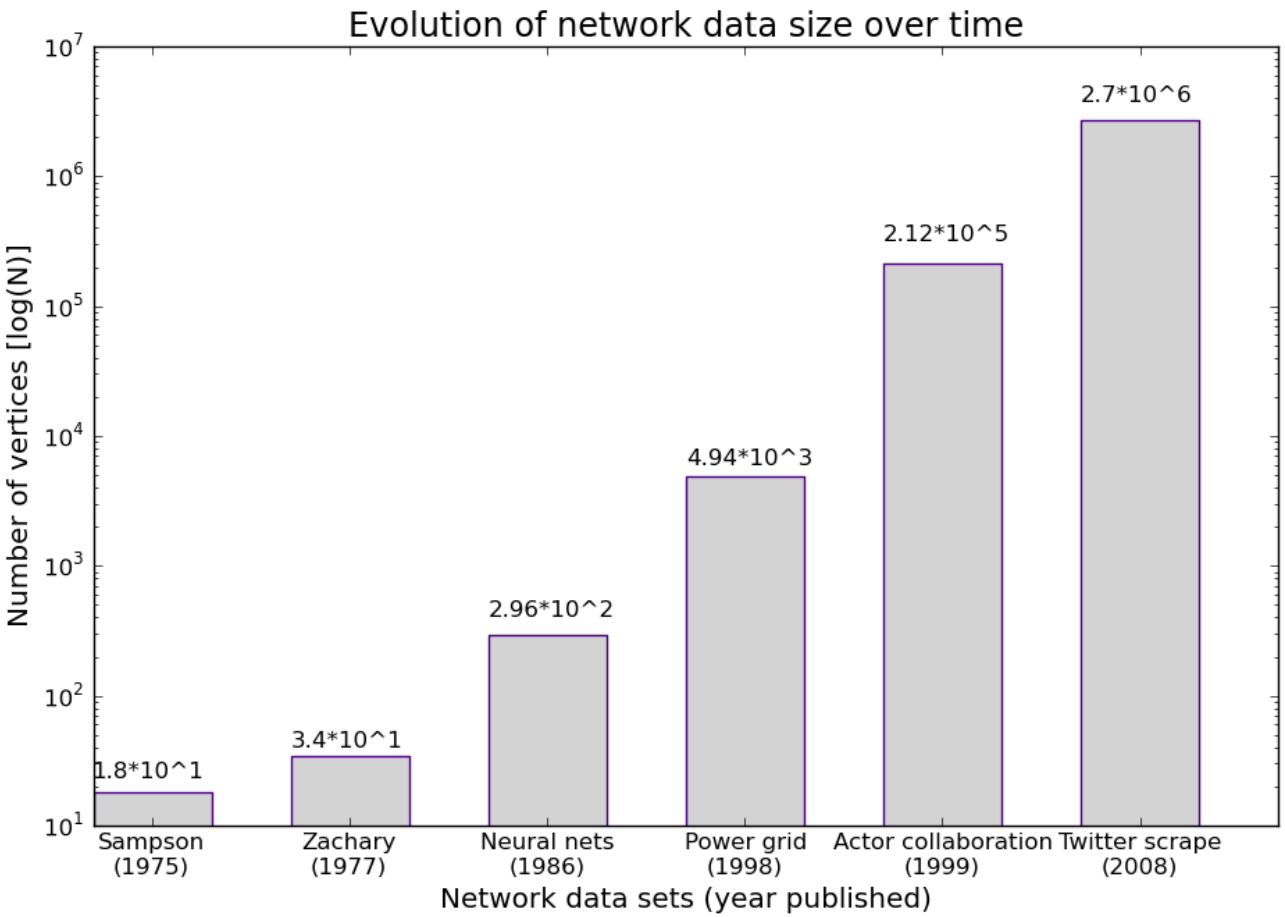
Why should we worry about scalability?

The size of networks being studying has increased rapidly over the years...



Why should we worry about scalability?

The size of networks being studying has increased rapidly over the years...



As network data becomes more readily available this trend will continue!

How network size affects tools

While the data continues to scale up, many tools have not kept pace

Standard Network Analysis Tools

	Tool	**Base Algorithms**	**Platforms**
Stand alone	UCINet	V=10K limit	Windows only
	Pajek	V=100K limit	Windows only
	ORA	C++/Java	Windows & Linux
	NetworkWorkbench	Java	Multi-platform
Libraries	Statnet	R	Multi-platform
	JUNG	Java	Multi-platform
	igraph	C/Fortran	Multi-platform
	NetworkX	C/Fortran	Multi-platform

How network size affects tools

While the data continues to scale up, many tools have not kept pace

Standard Network Analysis Tools

	Tool	**Base Algorithms**	**Platforms**
Stand alone	UCINet	V=10K limit	Windows only
	Pajek	V=100K limit	Windows only
	ORA	C++/Java	Windows & Linux
	NetworkWorkbench	Java	Multi-platform
Libraries	Statnet	R	Multi-platform
	JUNG	Java	Multi-platform
	igraph	C/Fortran	Multi-platform
	NetworkX	C/Fortran	Multi-platform

How network size affects tools

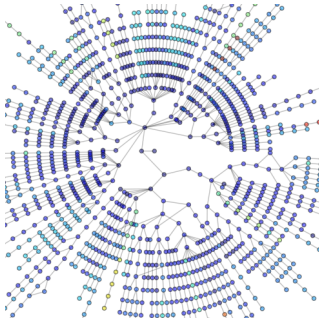
While the data continues to scale up, many tools have not kept pace

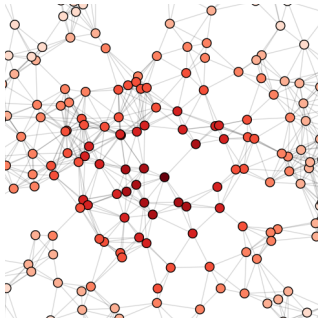
Standard Network Analysis Tools

	Tool	Base Algorithms	Platforms
Stand alone	UCINet	V=10K limit	Windows only
	Pajek	V=100K limit	Windows only
	ORA	C++/Java	Windows & Linux
	NetworkWorkbench	Java	Multi-platform
Libraries	Statnet	R	Multi-platform
	JUNG	Java	Multi-platform
	igraph	C/Fortran	Multi-platform
	NetworkX	C/Fortran	Multi-platform

`NetworkX` is designed to handle data sets of the scale being generated today

- 10M's nodes and 100M's edges
- Read network data from local files, or from external sources
- Inherently multi-platform





Moving beyond basic concepts of the "graph"

In a more fundamental way, however, most network tools are limited in their concept of what can be a network

- Networks are collections of nodes and edges
- Nodes are static integers or strings, and edges are binary or continuous values

NetworkX can represent **ANY** relationship supported by Python data types

Suppose we had data, or a data generating process, that was a time-series

- Current tools need kludges or hacks to add this data
- In NetworkX, we simply use the built-in Python datetime package to create a network of time-stamps

```
G=nx.DiGraph()
# Create datetime object nodes
for v in xrange(num_nodes):
    G.add_node(datetime.now())
time_nodes=G.nodes()
# Add edges with 'time' attribut
for i in xrange(num_nodes):
    draws=random.uniform(0,1,num_nodes)
    for j in xrange(num_nodes):
        if i!=j and draws[j]<=p:
            G.add_edge(time_nodes[i],time_nodes[j],time=datetime.now())
...
# target source datetime_created
2010-05-25 13:38:42.515323 2010-05-25 13:38:42.515492
    {'time': datetime.datetime(2010, 5, 25, 13, 38, 42, 515752)}
...
```

Moving beyond basic concepts of the "graph"

In a more fundamental way, however, most network tools are limited in their concept of what can be a network

- Networks are collections of nodes and edges
- Nodes are static integers or strings, and edges are binary or continuous values

NetworkX can represent **ANY** relationship supported by Python data types

Suppose we had data, or a data generating process, that was a time-series

- Current tools need kludges or hacks to add this data
- In NetworkX, we simply use the built-in Python datetime package to create a network of time-stamps

```
G=nx.DiGraph()
# Create datetime object nodes
for v in xrange(num_nodes):
    G.add_node(datetime.now())
time_nodes=G.nodes()
# Add edges with 'time' attribut
for i in xrange(num_nodes):
    draws=random.uniform(0,1,num_nodes)
    for j in xrange(num_nodes):
        if i!=j and draws[j]<=p:
            G.add_edge(time_nodes[i],time_nodes[j],time=datetime.now())
...
# target source datetime_created
2010-05-25 13:38:42.515323 2010-05-25 13:38:42.515492
    {'time': datetime.datetime(2010, 5, 25, 13, 38, 42, 515752)}
...
```

Moving beyond basic concepts of the "graph"

In a more fundamental way, however, most network tools are limited in their concept of what can be a network

- Networks are collections of nodes and edges
- Nodes are static integers or strings, and edges are binary or continuous values

`NetworkX` can represent **ANY** relationship supported by Python data types

Suppose we had data, or a data generating process, that was a time-series

- Current tools need kludges or hacks to add this data
- In `NetworkX`, we simply use the built-in Python `datetime` package to create a network of time-stamps

```
G=nx.DiGraph()
# Create datetime object nodes
for v in xrange(num_nodes):
    G.add_node(datetime.now())
time_nodes=G.nodes()
# Add edges with 'time' attribut
for i in xrange(num_nodes):
    draws=random.uniform(0,1,num_nodes)
    for j in xrange(num_nodes):
        if i!=j and draws[j]<=p:
            G.add_edge(time_nodes[i],time_nodes[j],time=datetime.now())
...
# target source datetime_created
2010-05-25 13:38:42.515323 2010-05-25 13:38:42.515492
    {'time': datetime.datetime(2010, 5, 25, 13, 38, 42, 515752)}
...
```

Moving beyond basic concepts of the "graph"

In a more fundamental way, however, most network tools are limited in their concept of what can be a network

- Networks are collections of nodes and edges
- Nodes are static integers or strings, and edges are binary or continuous values

NetworkX can represent **ANY** relationship supported by Python data types

Suppose we had data, or a data generating process, that was a time-series

- Current tools need kludges or hacks to add this data
- In NetworkX, we simply use the built-in Python datetime package to create a network of time-stamps

```
G=nx.DiGraph()
# Create datetime object nodes
for v in xrange(num_nodes):
    G.add_node(datetime.now())
time_nodes=G.nodes()
# Add edges with 'time' attribut
for i in xrange(num_nodes):
    draws=random.uniform(0,1,num_nodes)
    for j in xrange(num_nodes):
        if i!=j and draws[j]<=p:
            G.add_edge(time_nodes[i],time_nodes[j],time=datetime.now())
...
# target source datetime_created
2010-05-25 13:38:42.515323 2010-05-25 13:38:42.515492
    {'time': datetime.datetime(2010, 5, 25, 13, 38, 42, 515752)}
...
```

Python's scientific computing holy trinity

SciPy.org Sponsored By ENTHOUGHT

NumPy

matplotlib

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

`NumPy` is an extension of the `SciPy` data type to include **multidimensional arrays and matrices**

- Provides many functions for working on arrays and matrices
- Very useful for representing relational data

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

`NumPy` is an extension of the `SciPy` data type to include **multidimensional arrays and matrices**

- Provides many functions for working on arrays and matrices
- Very useful for representing relational data

Both SciPy and `NumPy` rely on the C library `LAPACK` for very fast implementation

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

`NumPy` is an extension of the `SciPy` data type to include **multidimensional arrays and matrices**

- Provides many functions for working on arrays and matrices
- Very useful for representing relational data

Both SciPy and `NumPy` rely on the C library `LAPACK` for very fast implementation

`matplotlib` is **primary plotting library in Python**

- Supports 2- and 3-D plotting
- API allows embedding in apps

Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

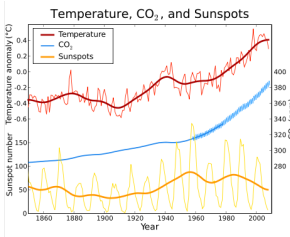
`NumPy` is an extension of the `SciPy` data type to include **multidimensional arrays and matrices**

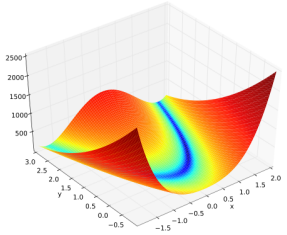
- Provides many functions for working on arrays and matrices
- Very useful for representing relational data

Both SciPy and `NumPy` rely on the C library `LAPACK` for very fast implementation

`matplotlib` is **primary plotting library in Python**

- Supports 2- and 3-D plotting
- API allows embedding in apps





Python's scientific computing holy trinity

Python's primary library for **mathematical and statistical** computing. Containing sub-libs for

- Numeric optimization
- Clustering
- Linear algebra
- ..and many others

The primary data type in `SciPy` is an array

- Data manipulation is similar to that of MATLAB

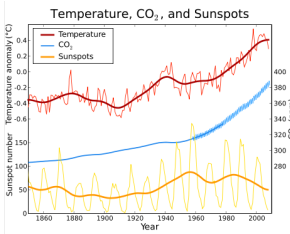
`NumPy` is an extension of the `SciPy` data type to include **multidimensional arrays and matrices**

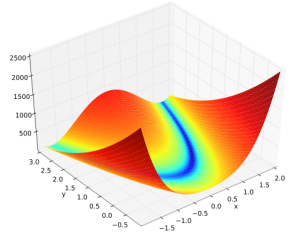
- Provides many functions for working on arrays and matrices
- Very useful for representing relational data

Both SciPy and `NumPy` rely on the C library `LAPACK` for very fast implementation

`matplotlib` is **primary plotting library in Python**

- Supports 2- and 3-D plotting
- API allows embedding in apps





All graphics are highly customizable and professional publication ready

Exporting to GraphViz in NetworkX

NetworkX is designed to be an open-source all-purpose network manipulation and analysis tool

- Historically, the focus has not been on visualization

While there are several options for visualization in NetworkX, perhaps the best is its ability to read and write GraphViz files

- GraphViz is an open-source tool designed specifically for drawing graphs from the DOT language
- NetworkX works directly with GV using the pygraphviz package

```
# Load Sampson monastery data from edgelist
>> g2=nx.read_edgelist("samp_like_el.txt",create_using=nx.DiGraph())
>> nx.info(g2)
Name:
Type:                    DiGraph
Number of nodes:         18
Number of edges:         55
Average in degree:       3.0556
Average out degree:      3.0556
# Convert to pygraphviz type
>> g2_gv=nx.to_agraph(g2)
# Output DOT file and draw using dot layout
>> g2_gv.write(“1samp_like_dot.dot”)
>> g2_gv.draw(“samp_like.png”,prog=“dot”)
```

Exporting to GraphViz in NetworkX

NetworkX is designed to be an open-source all-purpose network manipulation and analysis tool

- Historically, the focus has not been on visualization

While there are several options for visualization in NetworkX, perhaps the best is its ability to read and write GraphViz files

- GraphViz is an open-source tool designed specifically for drawing graphs from the DOT language
- NetworkX works directly with GV using the pygraphviz package

```
# Load Sampson monastery data from edgelist
>> g2=nx.read_edgelist("samp_like_el.txt",create_using=nx.DiGraph())
>> nx.info(g2)
Name:
Type:                    DiGraph
Number of nodes:         18
Number of edges:         55
Average in degree:       3.0556
Average out degree:      3.0556
# Convert to pygraphviz type
>> g2_gv=nx.to_agraph(g2)
# Output DOT file and draw using dot layout
>> g2_gv.write("1samp_like_dot.dot")
>> g2_gv.draw("samp_like.png",prog="dot")
```

Exporting to GraphViz in NetworkX

NetworkX is designed to be an open-source all-purpose network manipulation and analysis tool

- Historically, the focus has not been on visualization

While there are several options for visualization in NetworkX, perhaps the best is its ability to read and write GraphViz files

- GraphViz is an open-source tool designed specifically for drawing graphs from the DOT language
- NetworkX works directly with GV using the pygraphviz package

```
# Load Sampson monastery data from edgelist
>> g2=nx.read_edgelist("samp_like_el.txt",create_using=nx.DiGraph())
>> nx.info(g2)
Name:
Type:                    DiGraph
Number of nodes:         18
Number of edges:         55
Average in degree:       3.0556
Average out degree:      3.0556
# Convert to pygraphviz type
>> g2_gv=nx.to_agraph(g2)
# Output DOT file and draw using dot layout
>> g2_gv.write("1samp_like_dot.dot")
>> g2_gv.draw("samp_like.png",prog="dot")
```

Exporting to GraphViz in NetworkX

NetworkX is designed to be an open-source all-purpose network manipulation and analysis tool

- Historically, the focus has not been on visualization

While there are several options for visualization in NetworkX, perhaps the best is its ability to read and write GraphViz files

- GraphViz is an open-source tool designed specifically for drawing graphs from the DOT language
- NetworkX works directly with GV using the pygraphviz package

```
# Load Sampson monastery data from edgelist
>> g2=nx.read_edgelist("samp_like_el.txt",create_using=nx.DiGraph())
>> nx.info(g2)
Name:
Type:                    DiGraph
Number of nodes:         18
Number of edges:         55
Average in degree:       3.0556
Average out degree:      3.0556
# Convert to pygraphviz type
>> g2_gv=nx.to_agraph(g2)
# Output DOT file and draw using dot layout
>> g2_gv.write("1samp_like_dot.dot")
>> g2_gv.draw("samp_like.png",prog="dot")
```

Exporting to GraphViz in NetworkX

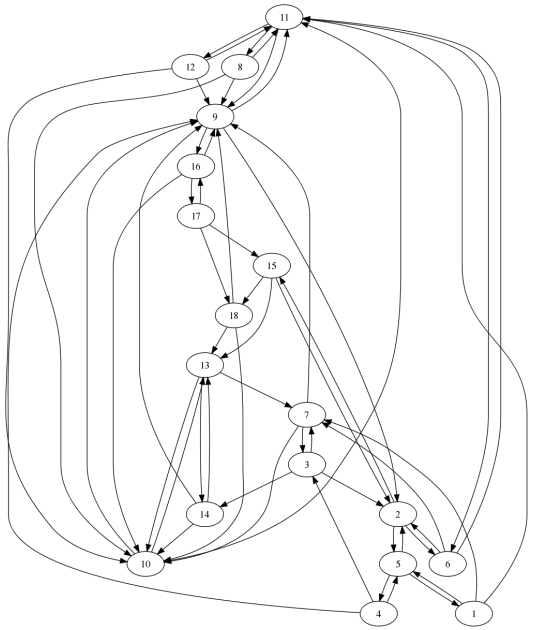
NetworkX is designed to be an open-source all-purpose network manipulation and analysis tool

- Historically, the focus has not been on visualization

While there are several options for visualization in NetworkX, perhaps the best is its ability to read and write GraphViz files

- GraphViz is an open-source tool designed specifically for drawing graphs from the DOT language
- NetworkX works directly with GV using the pygraphviz package

```
# Load Sampson monastery data from edgelist
>> g2=nx.read_edgelist("samp_like_el.txt",create_using=nx.DiGraph())
>> nx.info(g2)
Name:
Type:                    DiGraph
Number of nodes:         18
Number of edges:         55
Average in degree:       3.0556
Average out degree:      3.0556
# Convert to pygraphviz type
>> g2_gv=nx.to_agraph(g2)
# Output DOT file and draw using dot layout
>> g2_gv.write("1samp_like_dot.dot")
>> g2_gv.draw("samp_like.png",prog="dot")
```



Getting local data into NetworkX

Getting data into NetworkX is as simple as a single line of code:

Loading local data file

```
>>> G=read_edgelist("my_data.txt")
```

Getting local data into NetworkX

Getting data into NetworkX is as simple as a single line of code:

Loading local data file

```
>> G=read_edgelist("my_data.txt")
```

Like many other network analysis platforms, NetworkX can parse a wide variety of network data types

Readable and Writeable Formats in NX

	Format	Description
Standard	Edge list	2 column, source→ target
	Adjacency list	Each row 1st column as out-degree
	Pajek	Edge list + node and edge attr
Exotic	GML	Similar to DOT
	GraphML	XML implementation
	Pickle	Standard Python text output
	LEDA	Between edge list and Pajek
	YAML	Readable data serialization
	SparseGraph6	Adjacency list variant

Network data available on the Internet

Recently, there has been an explosion of resources for scraping social graph

Service	Data	API Docs
twitter	Following(ers), @-replies, date/time/geo	http://apiwiki.twitter.com/
facebook	Friends, Wall Posts, date/time	http://developers.facebook.com/docs/api
Google	All SocialGraph relationships	http://code.google.com/apis/socialgraph/
foursquare	Friends, Check-ins	http://foursquare.com/developers/
hunch	“Taste graph", recommendations	http://hunch.com/developers/
The New York Times	Congressional votes, campaign finance	http://developer.nytimes.com/docs

Network data available on the Internet

Recently, there has been an explosion of resources for scraping social graph

Service	Data	API Docs
twitter	Following(ers), @-replies, date/time/geo	http://apiwiki.twitter.com/
facebook	Friends, Wall Posts, date/time	http://developers.facebook.com/docs/api
Google	All SocialGraph relationships	http://code.google.com/apis/socialgraph/
foursquare	Friends, Check-ins	http://foursquare.com/developers/
hunch	"Taste graph", recommendations	http://hunch.com/developers/
The New York Times	Congressional votes, campaign finance	http://developer.nytimes.com/docs

There is clearly no shortage of data

- Each service provides different relational context
- Data formats are generally JSON, Atom, XML, or some combination
- Python has built-in parsers for all of these data types, which can easily be represented in `NetworkX`

Network data available on the Internet

Recently, there has been an explosion of resources for scraping social graph

Service	Data	API Docs
twitter	Following(ers), @-replies, date/time/geo	http://apiwiki.twitter.com/
facebook	Friends, Wall Posts, date/time	http://developers.facebook.com/docs/api
Google	All SocialGraph relationships	http://code.google.com/apis/socialgraph/
foursquare	Friends, Check-ins	http://foursquare.com/developers/
hunch	"Taste graph", recommendations	http://hunch.com/developers/
The New York Times	Congressional votes, campaign finance	http://developer.nytimes.com/docs

There is clearly no shortage of data

- Each service provides different relational context
- Data formats are generally JSON, Atom, XML, or some combination
- Python has built-in parsers for all of these data types, which can easily be represented in `NetworkX`

Next, we will go over an example of building network data using Google's SocialGraph API

Load data from databases

Along with the ability to parse data from online API's, NetworkX can create graphs from network data stored in various database formats

- All database platforms have either native or third-party libraries that allow read and write access from Python

Load data from databases

Along with the ability to parse data from online API's, NetworkX can create graphs from network data stored in various database formats

- All database platforms have either native or third-party libraries that allow read and write access from Python

Ope-Source DB's Supported in Python

	Database	**Python Library**
SQL	MySQL	MySQLdb
	PosgreSQL	PyGreSQL
	SQLite	sqlite3
NoSQL	Neo4j	Neo4j.py
	MongoDB	PyMongo
	CouchDB	couchdb-python

Load data from databases

Along with the ability to parse data from online API's, NetworkX can create graphs from network data stored in various database formats

- All database platforms have either native or third-party libraries that allow read and write access from Python

Ope-Source DB's Supported in Python

	Database	**Python Library**
SQL	MySQL	MySQLdb
	PosgreSQL	PyGreSQL
	SQLite	sqlite3
NoSQL	Neo4j	Neo4j.py
	MongoDB	PyMongo
	CouchDB	couchdb-python

- This is just a small glance of all possible Python→ DB bindings

Module II Review

Why use NetworkX to do SNA?

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems
2. Most of the core algorithms in NX rely on extremely fast legacy code

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems
2. Most of the core algorithms in NX rely on extremely fast legacy code
3. Highly flexible graph implementations (a graph can be anything!)

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems
2. Most of the core algorithms in NX rely on extremely fast legacy code
3. Highly flexible graph implementations (a graph can be anything!)
4. Extensive set of native readable and writable formats

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems
2. Most of the core algorithms in NX rely on extremely fast legacy code
3. Highly flexible graph implementations (a graph can be anything!)
4. Extensive set of native readable and writable formats
5. Takes advantage of Python's ability to pull data from the Internet or databases

Module II Review

Why use `NetworkX` to do SNA?

1. Unlike many other tools, NX is designed to handle data on a scale relevant to modern problems
2. Most of the core algorithms in NX rely on extremely fast legacy code
3. Highly flexible graph implementations (a graph can be anything!)
4. Extensive set of native readable and writable formats
5. Takes advantage of Python's ability to pull data from the Internet or databases

Questions?