Using igraph with Python and R

python™

igraph – The network analysis package

"igraph is a collection of network analysis tools with the emphasis on efficiency, portability and ease of use. igraph is open source and free. igraph can be programmed in R, Python, Mathematica and C/C++."

www.igraph.org

Installing Python

- Go to https://www.anaconda.com/distribution/#download-section and pick the installer for your platform (for the Anaconda distribution)
- Or https://www.python.org/downloads/ (for just Python)
- If you are using Windows, you may run into some difficulty getting the igraph setup if there has been a recent update to Python. If you have difficulties, you may wish to use R instead.
- Aside from that, this part will be easy, just follow the instructions

Installing R

- Go to https://cran.r-project.org/mirrors.html (for standard R)
- You may also wish to install R studio https://www.rstudio.com/products/rstudio/download/
- Pick the installer for your platform
- Follow the instructions therein
- It's that easy

Installing the igraph package

- R
  - Run the command > install.packages('igraph')
    - Pick whatever mirror you like, it doesn't matter
- Python
  - Mac/Linux
    - In the command prompt run the command > pip install python-igraph
  - Windows (this one is a bit of a pain, the command line DOES NOT WORK)
    - Go to http://www.lfd.uci.edu/~gohlke/pythonlibs/#python-igraph
      - Pick the .whl for your architecture and download it
    - In the command prompt, go to wherever you put the .whl
    - Run the command > pip install [full-file-name-here].whl

Installing visualization packages (Python)

- This is also a bit tedious, but if you want to get visualizations working in Python, it's necessary (in R it works by default)
- Install Cairo, following the platform specific instructions here: http://cairographics.org/download/
- Mac
  - Run > sudo port install cairo
- Linux
  - Distro dependent (Follow instructions on the download page)
- Windows
  - Go to http://www.lfd.uci.edu/~gohlke/pythonlibs/#pycairo
    - Like before, install it with > pip install [full-file-name-here].whl

Unique features: Python

- Syntax
  - Whitespace is part of the language (no brackets on loops)
  - Language is dynamically typed: a=1 a='yes!' is a valid series of commands
    - a will be an int, and later a string
  - Generally concise, not a lot of wasted characters
- Libraries
  - Has an excellent selection of libraries for almost any data analysis task
    - Cleaning, parsing, machine learning, etc.
  - Many libraries exist, but there are some only available in 2.x (fewer, now that 2.x is officially losing support)

Unique features: R

- Syntax
  - Declaration <- or = (almost always interchangeable)
  - Accessing array elements List$index == List[[‘index’]] (List[‘index’] returns a list)
- Data structures
  - Lists are vectors with items of different types
  - data.frame is a database-like data structure unique to R
- Libraries
  - R has a LOT of useful libraries for data processing, analysis, etc.
- Bulk operations
  - c <- a + b works whether a and b are variables or vectors of variables
    - If vector, each value will be added in order, so [1,2,3]+[2,3,4]=[3,5,7]

Code Sample: make a basic graph object

```
library("igraph")
numNodes <- 5
edges <- c(1,2, 3,2, 2,4)
g<-graph(edges, n=numNodes, directed=TRUE)
print (g)
```

```
> IGRAPH D--- 5 3 --
+ edges:[1] 1->2 3->2 2->4
```

```
from igraph import *
numNodes = 5
edges = [(1,2), (3,2), (2,4)]
g=Graph(edges, n=numNodes, directed=True)
print (g)
```

```
> IGRAPH D--- 5 3 --
+ edges:1->2 3->2 2->4
```

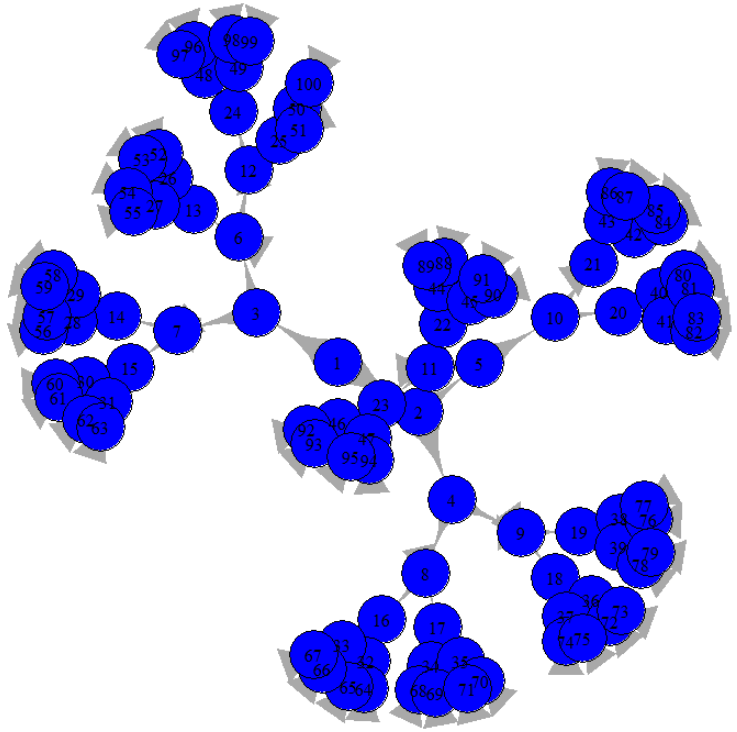
Sample exercise: Generating graphs

```
t <- graph.tree(100, 2)
```

IGRAPH D--- 100 99 -- Tree

+ attr: name (g/c), children (g/n), mode (g/c)

```
t = Graph.Tree(100, 2)
```



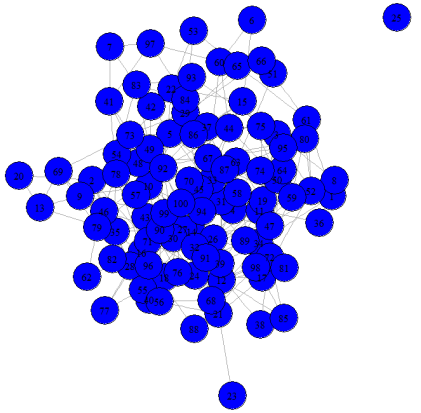
Sample exercise: Generating graphs

```
erg <- erdos.renyi.game (100, .05)
```

IGRAPH U--- 100 253 -- Erdos renyi (gnp) graph

+ attr: name (g/c), type (g/c), loops (g/l), p (g/n)

```
erg = Graph.Erdos_Renyi(100, .05)
```

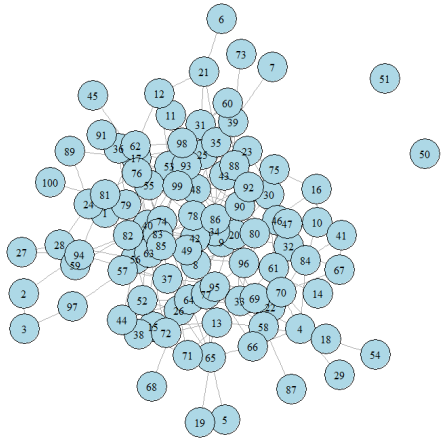


```
erg2 <- erdos.renyi.game (100, 200, type="gnm")
```

IGRAPH U--- 100 200 -- Erdos renyi (gnm) graph

+ attr: name (g/c), type (g/c), loops (g/l), m (g/n)

```
erg2 = Graph.Erdos_Renyi(100, m=200)
```



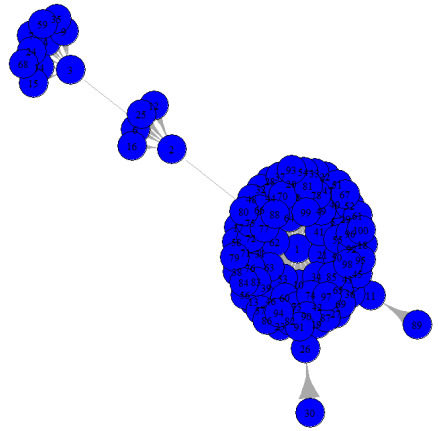
Sample exercise: Generating graphs

```
b <- barabasi.game(100, 2)
```

IGRAPH D--- 100 99 -- Barabasi graph

+ attr: name (g/c), power (g/n), m (g/n), zero.appeal (g/n), algorithm (g/c)

```
b = Graph.Barabasi(100, 2)
```

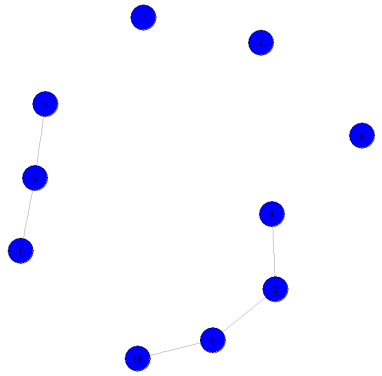


```
e <- induced_subgraph(erg, 1:10)
```

IGRAPH U--- 10 5 -- Erdos renyi (gnp) graph

+ attr: name (g/c), type (g/c), loops (g/l), p (g/n)

```
e = erg.induced_subgraph(range(10))
```



Sample exercise: structural properties

degree(g)	g.degree()
diameter(g)	g.diameter()
radius(g)	g.radius()
girth(g)$girth	g.girth()
is_connected(g)	g.is_connected()
is_dag(g)	g.is_dag()
components(g, 'strong')$no	len(g.components('strong'))

Sample exercise: more advanced properties

degree(g, mode="in")	g.indegree()
degree(g, mode="out")	g.outdegree()
betweenness(g)	g.betweenness()
evcent(g)$vector	g.evcent()
page_rank(g)$vector	g.personalized_pagerank()

Sample exercise: visualization

```
plot(g)

plot(g, layout=layout_randomly)
plot(g, layout=layout_on_grid)

plot(degree.distribution(g))
```

```
plot(g)

plot(g, layout='random')
plot(g, layout='grid')

plot(g.degree_distribution())
```

Wrap up

- This is only a tiny segment of what igraph can do
  - Tutorial (Python): http://igraph.org/python/doc/tutorial/tutorial.html
  - Documentation: http://igraph.org/r/doc/ and http://igraph.org/python/doc/igraph-module.html
- Python Tutorial:
  - https://www.w3schools.com/python/
- Python Documentation:
  - https://docs.python.org/3/
- R Documentation
  - https://cran.r-project.org/manuals.html