Graphs and Networks

Nik Bear Brown

In this lesson we'll learn the how to implement and analyze graphs and networks in R.

Additional packages needed

To run the code you may need additional packages.

- If necessary install the followings packages.

```
install.packages("igraph");
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
dev.off()
```

```
## null device
##           1
```

Data

We will be creating graphs with the library(igraph).

```
# create a undirected unnamed graph
graph.1 <- make_graph( c(1,2, 2,3, 3,5, 7,6, 1,8, 2,3, 5,7, 3,6, 3,6,
4,2, 8,8), n=8, directed=F)
```

Graphs and Networks

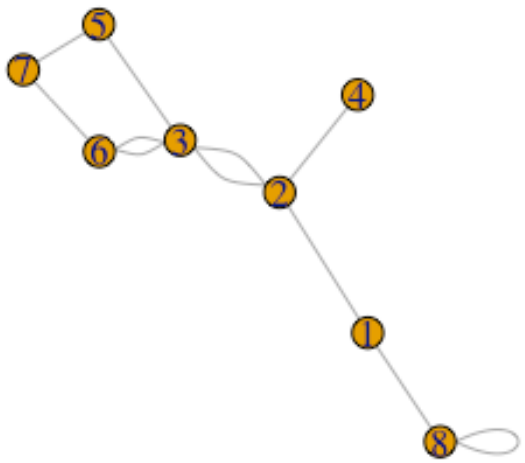
A network is a graph. That is:

- A set of vertices (or nodes) joined by edges.
- Vertices and edges can have properties (feature values).
- Edges can have weights and be directed.

```
# create a undirected unnamed graph
graph.1 <- make_graph( c(1,2, 2,3, 3,5, 7,6, 1,8, 2,3, 5,7, 3,6, 3,6,
4,2, 8,8), n=8, directed=F)
graph.1
```

```
## IGRAPH U--- 8 11 --
## + edges:
##  [1] 1--2 2--3 3--5 6--7 1--8 2--3 5--7 3--6 3--6 2--4 8--8
```

```
plot(graph.1)
```




```
# graph summary
str(graph.1)
```

```
## IGRAPH U--- 8 11 --
## + edges:
##  [1] 1--2 2--3 3--5 6--7 1--8 2--3 5--7 3--6 3--6 2--4 8--8
```

```r
# number of vertices
vcount(graph.1)
```

```
## [1] 8
```

```r
# number of edges
ecount(graph.1)
```

```
## [1] 11
```

```r
# check if directed, and convert
is.directed(graph.1)
```

```
## [1] FALSE
```

```r
graph.1 <- as.directed(graph.1) # to convert back, use -
as.undirected(graph.1)
str(graph.1)
```

```
## IGRAPH D--- 8 22 --
## + edges:
##  [1] 1->2 2->3 3->5 6->7 1->8 2->3 5->7 3->6 3->6 2->4 8->8 2->1 3-
>2 5->3
## [15] 7->6 8->1 3->2 7->5 6->3 6->3 4->2 8->8
```

```r
# check for multiple, loop edges and remove them
is_simple(graph.1)
```

```
## [1] FALSE
```

```r
which_multiple(graph.1)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
FALSE
## [12] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
TRUE
```

```r
which_loop(graph.1)
```

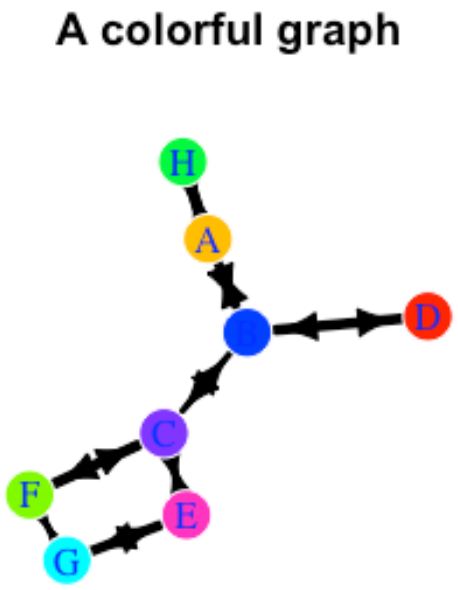
```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
TRUE
## [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
TRUE
```

```r
graph.1 <- simplify(graph.1)
is_simple(graph.1)
```

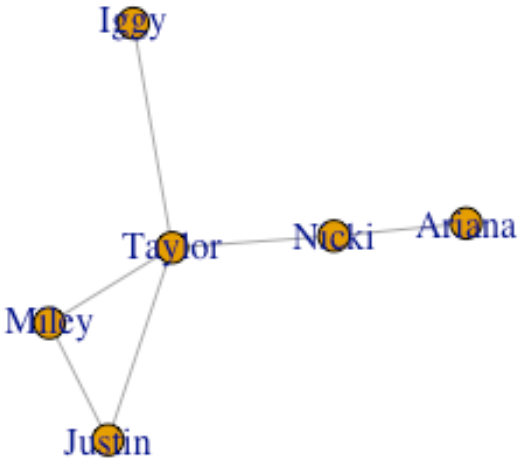
```
## [1] TRUE
```

```r
# name vertices using uppercase
V(graph.1)$name <- toupper(letters[1:8])
# assign attributes to the graph
graph.1$name <- "A colorful graph"
```

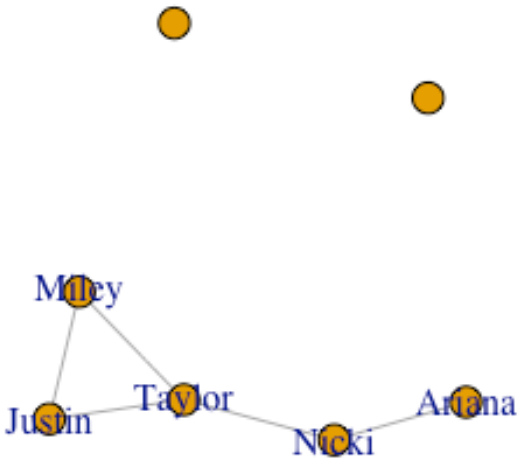
```
# assign attributes to the graph's vertices
V(graph.1)$color <- sample(rainbow(8),8,replace=F)
# assign attributes to the edges
E(graph.1)$weight <- runif(length(E(graph.1)),.75,5)
# plot the graph with additional parameters
# Note: plot=igraph.plot
plot(graph.1, layout = layout.fruchterman.reingold,
     main = graph.1$name,
     vertex.label = V(graph.1)$name,
     vertex.size = 25,
     vertex.color= V(graph.1)$color,
     vertex.frame.color= "white",
     vertex.label.color = "blue",
     edge.width=E(graph.1)$weight,
     edge.color="black")
```



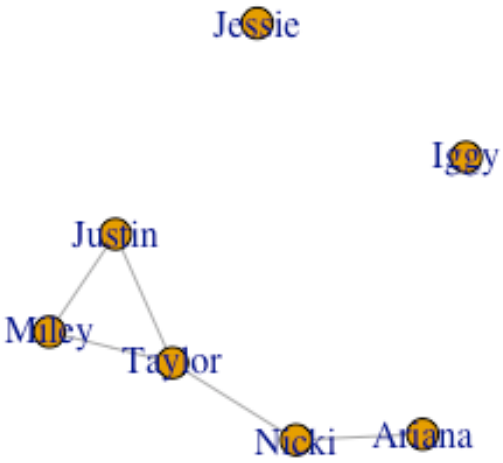

```
# create a undirected named graph
graph.2 <- graph.formula(Miley-Justin-Taylor-Miley, Ariana:Taylor-
Nicki,
                         Taylor-Iggy)
plot(graph.2)
```



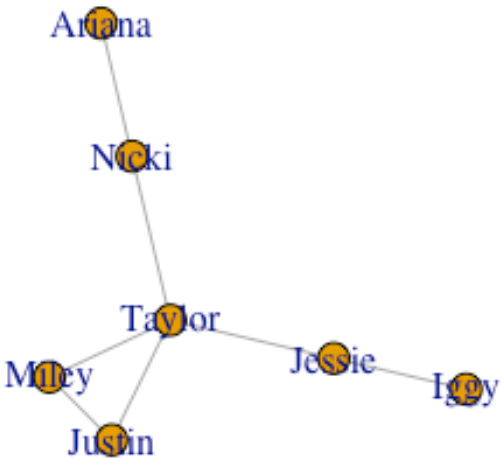

```
# remove Iggy
graph.2 <- delete_vertices(graph.2, match("Iggy", V(graph.2)$name))
# add two vertices
graph.2.1 <- add_vertices(graph.2, 2)
plot(graph.2.1)
```



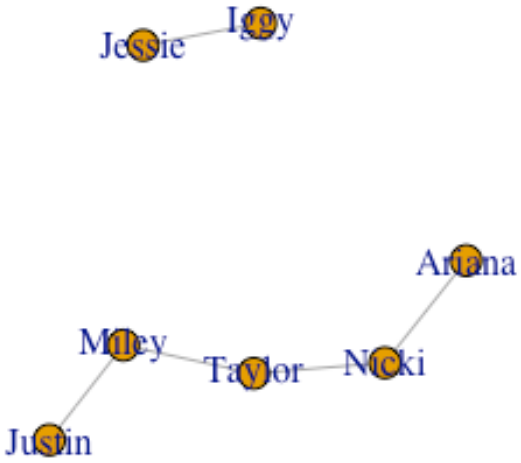

```
# add two vertices with names
graph.2.2 <- add_vertices(graph.2, 2, attr=list(name=c("Iggy",
"Jessie")))
plot(graph.2.2)
```



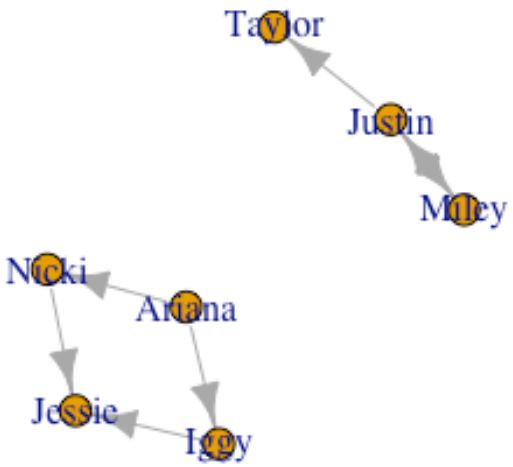

```
# add edges
graph.2 <- add_edges(graph.2.2, match(c("Jessie", "Iggy",
                                        "Taylor", "Jessie"),
V(graph.2.2)$name ))
plot(graph.2)
```




```
# delete edge
graph.2 <- delete_edges(graph.2, match(c("Taylor", "Jessie"),
V(graph.2)$name))
plot(graph.2)
```




```
# create a directed named graph
# the (+) sign signifies the direction of the arrow
graph.3 <- graph.formula(Miley+-+Justin--+Taylor, Ariana--+Nicki:Iggy--
+Jessie)
plot(graph.3)
```



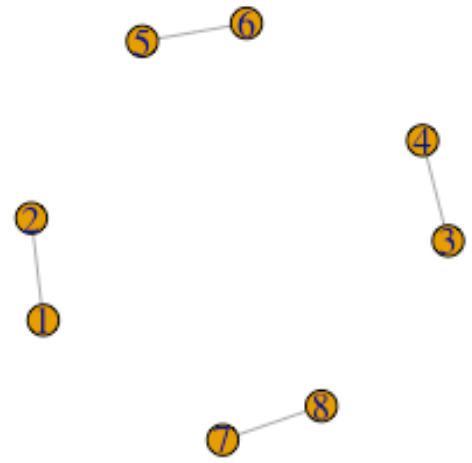

```
# names of vertices
V(graph.3)

## + 7/7 vertices, named:
## [1] Miley  Justin Taylor Ariana Nicki  Iggy   Jessie

# ids of vertices
as.vector(V(graph.3))

## [1] 1 2 3 4 5 6 7

# sequence of edges
E(graph.3)

## + 7/7 edges (vertex names):
## [1] Miley ->Justin Justin->Miley  Justin->Taylor Ariana->Nicki
## [5] Ariana->Iggy   Nicki ->Jessie Iggy  ->Jessie

# edge from vertex 1 to vertex 2
E(graph.3, P=c(1,2))

## + 1/7 edge (vertex names):
## [1] Miley->Justin
```

```
# id of the edge from vertex 1 to vertex 2
as.vector(E(graph.3, P=c(1,2)))

## [1] 1

# all adjacent edges of vertex 4
E(graph.3)[adj(4)]

## + 2/7 edges (vertex names):
## [1] Ariana->Nicki Ariana->Iggy

# all adjacent edges of vertices 4 and 1
E(graph.3)[adj(c(4,1))]

## + 4/7 edges (vertex names):
## [1] Miley ->Justin Justin->Miley  Ariana->Nicki  Ariana->Iggy

# all outgoing edges from vertex 2
E(graph.3)[from(2)]

## + 2/7 edges (vertex names):
## [1] Justin->Miley  Justin->Taylor

# all incoming edges to vertex 7
E(graph.3)[to(7)]

## + 2/7 edges (vertex names):
## [1] Nicki->Jessie Iggy ->Jessie
```

Bipartite graphs

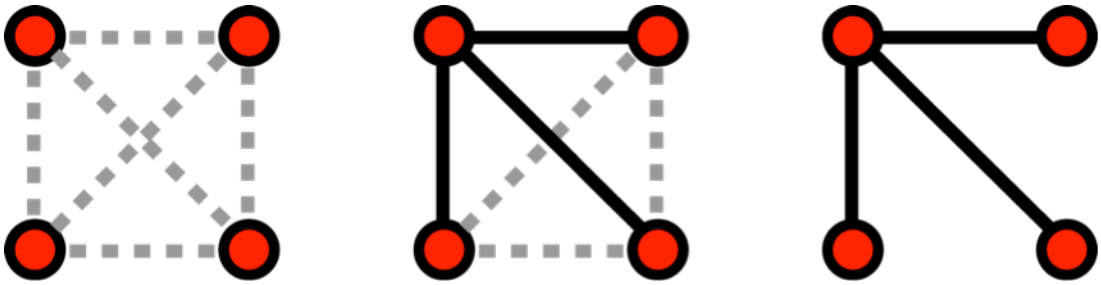
In graph theory, a bipartite graph is a graph whose vertices can be divided into two disjoint sets and (that is, and are each independent sets) such that every edge connects a vertex in one disjoint set to one in in the other disjoint set. That is, there are only between set and no within set edges.

```
# create a bipartite graph
graph.4 <- make_bipartite_graph( rep(0:1,length=8), c(1:8))
plot(graph.4)
```



Erdos-Renyi Random Graphs

In graph theory, the Erdos-Renyi model is either of two closely related models for generating random graphs. They are named after Paul Erdős and Alfréd Rényi.



- Erdos-Renyi model is generated with N = 4 nodes. from en.wikipedia.org/wiki/Network_science#/media/File:ER_model.png

The G(n, M) model, * n is the the number of vertices or nodes
* M is the the number of edges * $0 \leq p \leq 1$ * for each pair (i,j), generate the edge (i,j) independently with probability p

Equivalently, all graphs with n nodes and M edges have equal probability of

$$p^M(1-p)^{\binom{n}{2}-M}.$$

The parameter p in this model can be thought of as a weighting function; as p increases from 0 to 1, the model becomes more and more likely to include graphs with more edges and less and less likely to include graphs with fewer edges.

Random graphs degree distribution follows a binomial.

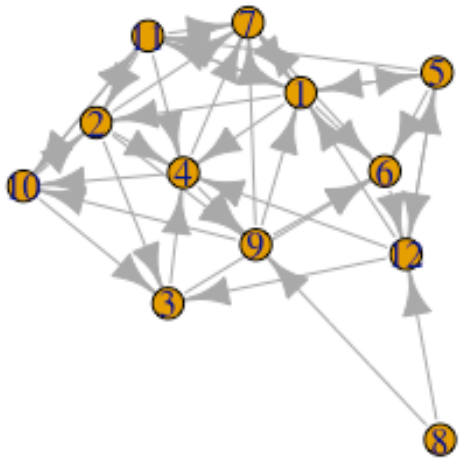
$$P(\deg(v)=k)=\binom{n-1}{k}p^k(1-p)^{n-1-k},$$

where n is the total number of vertices in the graph. Since

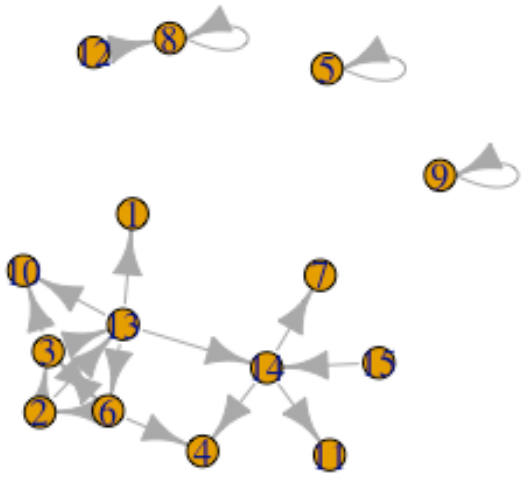
$$P(\deg(v)=k)\to\frac{(np)^k e^{-np}}{k!}\quad \text{as } n\to\infty \text{ and } np=\text{const},$$

this distribution is Poisson for large n and np = const.

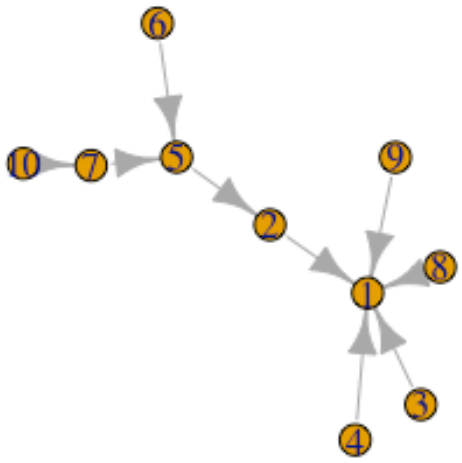
```
  # create a random graph
random.graph.1 <- sample_gnp(n=12, p=.3, directed=T)
plot.igraph(random.graph.1)
```



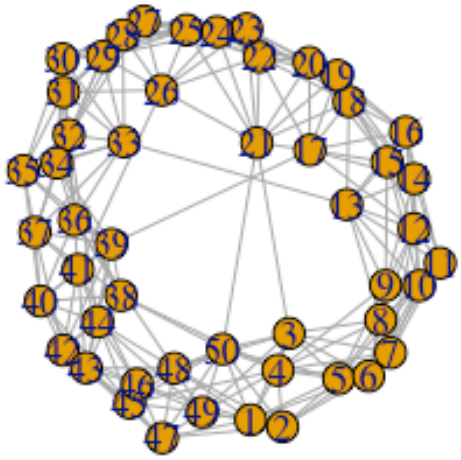

```
random.graph.2 <- sample_gnm(n=15, m=20, directed=T, loops=T)
plot.igraph(random.graph.2)
```




```
# create a scale-free graph
scale.free.graph <- sample_pa(n=10,power=1)
plot.igraph(scale.free.graph)
```




```
# create a small-world graph
small.world.graph <- sample_smallworld(1, 50, 5, 0.025, loops=T,
multiple=F)
plot.igraph(small.world.graph)
```



```r
# convert a graph to an adjacency matrix
matrix.3 <- as_adjacency_matrix(graph.3, sparse=F)
matrix.3
```

```
##        Miley Justin Taylor Ariana Nicki Iggy Jessie
## Miley      0      1      0      0     0    0      0
## Justin     1      0      1      0     0    0      0
## Taylor     0      0      0      0     0    0      0
## Ariana     0      0      0      0     1    1      0
## Nicki      0      0      0      0     0    0      1
## Iggy       0      0      0      0     0    0      1
## Jessie     0      0      0      0     0    0      0
```

```r
# convert a graph to an adjacency list
adjlist.3 <- as_adj_list(graph.3)
adjlist.3
```

```
## $Miley
## + 2/7 vertices, named:
## [1] Justin Justin
##
## $Justin
## + 3/7 vertices, named:
## [1] Miley  Miley  Taylor
```

```
##
## $Taylor
## + 1/7 vertex, named:
## [1] Justin
##
## $Ariana
## + 2/7 vertices, named:
## [1] Nicki Iggy
##
## $Nicki
## + 2/7 vertices, named:
## [1] Ariana Jessie
##
## $Iggy
## + 2/7 vertices, named:
## [1] Ariana Jessie
##
## $Jessie
## + 2/7 vertices, named:
## [1] Nicki Iggy
```

```
# convert a graph to an edge list
edgelist.3 <- as_edgelist(graph.3, names = TRUE)
edgelist.3
```

```
##      [,1]     [,2]
## [1,] "Miley"  "Justin"
## [2,] "Justin" "Miley"
## [3,] "Justin" "Taylor"
## [4,] "Ariana" "Nicki"
## [5,] "Ariana" "Iggy"
## [6,] "Nicki"  "Jessie"
## [7,] "Iggy"   "Jessie"
```

```
# convert a graph to an incidence matrix
# only bipartite graphs can be converted to an incidence matrix
incidence.4 <- as_incidence_matrix(graph.4, sparse=F)
incidence.4
```

```
##   2 4 6 8
## 1 1 0 0 0
## 3 0 1 0 0
## 5 0 0 1 0
## 7 0 0 0 1
```

Network Motifs

Network motifs, are recurrent and statistically significant sub-graphs or patterns.

```r
# Motifs
# create a directed scale-free graph
graph.5 <- sample_pa(n=20,power=1)
plot.igraph(graph.5)
```

```r
# total number of motifs of size 3
count_motifs(graph.5, size = 3)
```

```
## [1] 53
```

```r
# number of occurences of each motif in a graph
motifs(graph.5, size = 3)
```

```
##  [1] NA NA 35 NA 18  0  0  0  0  0  0  0  0  0  0  0
```

```r
# estimate for the total number of motifs in a graph
sample_motifs(graph.5, size = 3, cut.prob=c(0.25,0.5,0.75),
              sample.size=15)
```

```
## [1] 4
```

```r
# total number of motifs of size 4
count_motifs(graph.5, size = 4)
```

```
## [1] 150
```

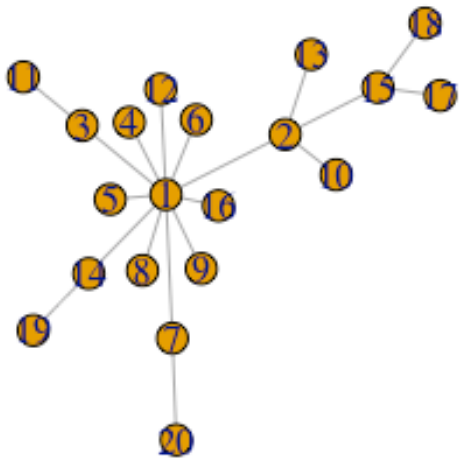
```r
# number of occurences of each motif in a graph
motifs(graph.5, size = 4)
```

```
##   [1] NA NA NA 46 NA NA NA 35  0 NA NA NA 58  0  0 NA  0  0  0  0  0
0 NA
##  [24] NA  0  0  0 NA NA 11  0  0  0 NA NA  0  0  0  0 NA  0  0  0  0
0  0
##  [47]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 NA  0  0  0  0
0  0
##  [70]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
##  [93]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [116]  0  0  0  0  0 NA  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [139]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [162]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [185]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0
## [208]  0  0  0  0  0  0  0  0  0  0  0  0
```

```r
# estimate for the total number of motifs in a graph
sample_motifs(graph.5, size = 4, cut.prob=c(0.2,0.4,0.6,0.8),
sample.size=15)
```

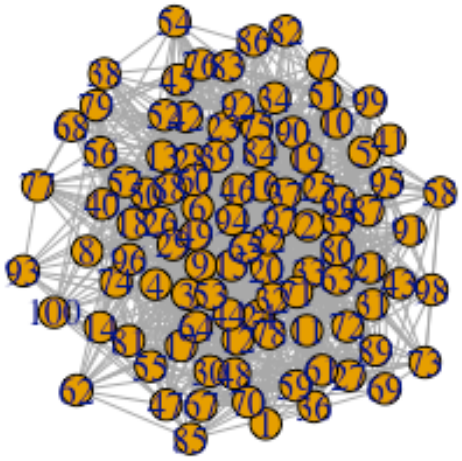
```
## [1] 0
```

```r
# create an undirected scale-free graph
graph.6 <- sample_pa(n=20,power=1, directed=F)
plot.igraph(graph.6)
```



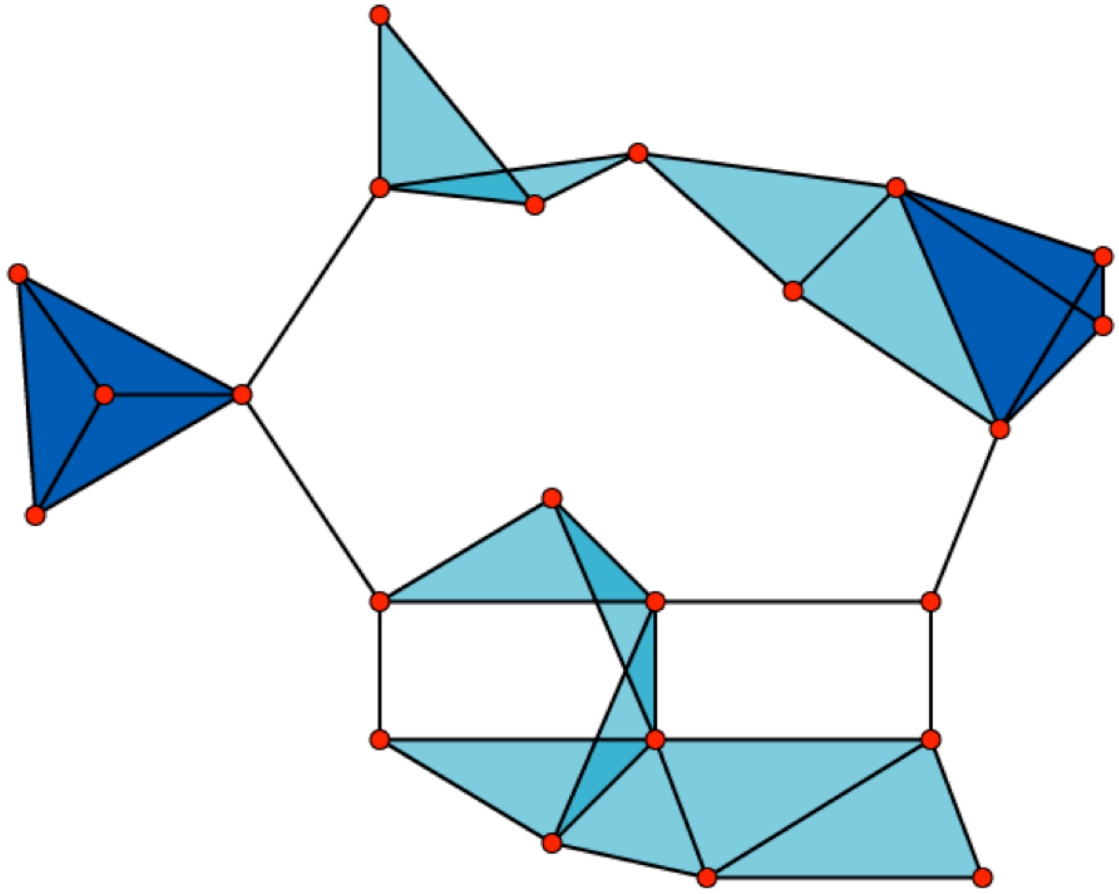
```
# total number of motifs of size 3
count_motifs(graph.6, size = 3)

## [1] 67

# number of occurences of each motif in a graph
motifs(graph.6, size = 3)

## [1] NA NA 67  0

# estimate for the total number of motifs in a graph
sample_motifs(graph.6, size = 3, cut.prob=c(0.25,0.5,0.75),
sample.size=15)

## [1] 0

# total number of motifs of size 4
count_motifs(graph.6, size = 4)

## [1] 236

# number of occurences of each motif in a graph
motifs(graph.6, size = 4)

##  [1]  NA  NA  NA  NA 170  NA  66   0   0   0   0
```

```
# estimate for the total number of motifs in a graph
sample_motifs(graph.6, size = 4, cut.prob=c(0.2,0.4,0.6,0.8),
sample.size=15)

## [1] 0

# Cliques - only for undirected graphs
# If calculated for directed graphs - directions are ignored
# create a random undirected graph
graph.7 <- sample_gnm(100,1000)
plot(graph.7)
```



Cliques

A clique is a fully connected graph. A graph that has all possible n(n-1)/2 edges.



Clique (graph theory)

A graph with * 23 x 1-vertex cliques (the vertices), * 42 x 2-vertex cliques (the edges), * 19 x 3-vertex cliques (the light and dark blue triangles), and * 2 x 4-vertex cliques (just the dark blue areas).

The 11 light blue triangles form maximal cliques. The two dark blue 4-cliques are both maximum and maximal, and the clique number of the graph is 4.

from Clique (graph theory)

```
# find all complete subgraphs - i.e. cliques
all_cliq <- cliques(graph.7)
length(all_cliq) # this is the total number of cliques

## [1] 2757

# head(all_cliq)
# tail(all_cliq)
# size of the largest clique(s)
large_cliq_size <- clique_num(graph.7)
large_cliq_size

## [1] 5
```

```
# find all largest cliques
large_cliq <- largest_cliques(graph.7)
# alternate way of finding all largest cliques
large_cliq <- cliques(graph.7, min=large_cliq_size)
length(large_cliq) # this is the total number of largest cliques
```

```
## [1] 10
```

```
# head(large_cliq)
# tail(large_cliq)
# find all maximal cliques
max_cliq <- max_cliques(graph.7)
length(max_cliq) # this is the total number of maximal cliques
```

```
## [1] 807
```

```
count_max_cliques(graph.7) # alterate way
```

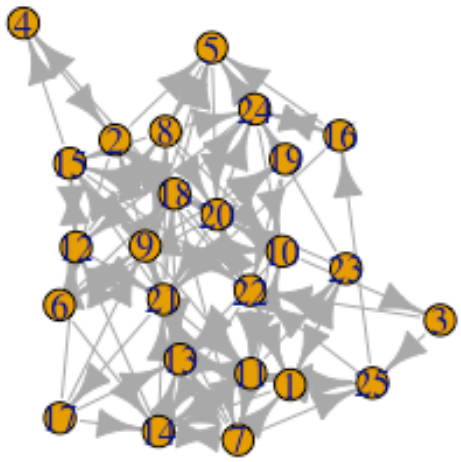
```
## [1] 807
```

```
# head(max_cliq)
# tail(max_cliq)
```

PageRank

PageRank counts the number weighted by the quality of links to a node from other nodes to determine a numeric estimate of the importance the node.

```
# PageRank
# create a random directed graph
graph.8 <- sample_gnm(25,100, directed=T)
plot(graph.8)
```



```
# calculate PageRank for all the vertices in the graph
unweighted_rank <- page_rank(graph.8)
unweighted_rank

## $vector
##  [1] 0.04605462 0.05483732 0.01411234 0.02061032 0.04819744
0.03426307
##  [7] 0.04278549 0.03841993 0.04820532 0.03817573 0.07089383
0.03851107
## [13] 0.05415951 0.05961126 0.03786037 0.02435347 0.01385252
0.05406463
## [19] 0.01915211 0.04082408 0.06466782 0.03899980 0.03261099
0.04019352
## [25] 0.02458343
##
## $value
## [1] 1
##
## $options
## NULL

which.max(unweighted_rank$vector)

## [1] 11
```

```r
# calculate the PageRank of vertices 1 through 5, using the power
algorithm
page_rank(graph.8, algo=c("power"), vids=c(1:5))
```

```
## Warning in .Call("R_igraph_personalized_pagerank", graph, algo, vids
- 1, :
## At structural_properties.c:1379 :igraph_pagerank_old is deprecated
from
## igraph 0.7, use igraph_pagerank instead

## $vector
## [1] 0.04615115 0.05497201 0.01414209 0.02067163 0.04829088
##
## $value
## [1] 2.251334e-314
##
## $options
## $options$niter
## [1] 1000
##
## $options$eps
## [1] 0.001
```

```r
# the parameter algo can take three values - prpack(default), arpack
and power
# add weights to all the edges and then calculate PageRank
# damping - the damping factor - can be between 0 and 1
# 0.85 is generally the most accepted value and also the default here
# $vector gives only the PageRank vector
weighted_rank <- page_rank(graph.8, damping=0.85,
weights=runif(ecount(graph.8)))$vector
weighted_rank
```

```
##  [1] 0.030133391 0.045098401 0.009399175 0.021359492 0.040032608
##  [6] 0.032036619 0.047735059 0.048925489 0.049363210 0.034417749
## [11] 0.072035092 0.036545348 0.046671880 0.069466292 0.043554555
## [16] 0.035601859 0.016658766 0.052722306 0.022200767 0.038212310
## [21] 0.074698909 0.025486600 0.032512191 0.053636346 0.021495584
```

```r
which.max(weighted_rank)
```

```
## [1] 21
```

```r
# Personalized PageRank
personalProbability <- runif(vcount(graph.8))
personalized_rank <- page_rank(graph.8,
personalized=personalProbability)
personalized_rank$vector
```

```
##  [1] 0.04647270 0.05470722 0.01883314 0.02502730 0.04479117
0.03624188
##  [7] 0.03575532 0.03647921 0.04647589 0.04132948 0.06900015
```

```
0.04227732
## [13] 0.05428811 0.05452446 0.03556873 0.02633643 0.01534525
0.05329480
## [19] 0.02257017 0.04146449 0.06342205 0.04042483 0.02856420
0.04134910
## [25] 0.02545660

which.max(personalized_rank$vector)

## [1] 11
```

Assingment

Answer the following questions:
* Can some form of Social Network analysis help in your research project?
* If it can apply Social Network analysis to your research project? Does it help?
* If (and only if) you can't use some form of Social Network analysis help in your research project then apply a form of Social Network analysis to the data the Twiiter time series data set
* Note you only need to use ONE Social Network analysis approach from Module 12, so there will be only ONE assingment for all the modules and the same assingment for all the modules.

Resources

- Network visualization in R with the igraph package
- [Making prettier network graphs with sna and igraph via @rbloggers](http://www.r-bloggers.com/making-prettier-network-graphs-with-sna-and-igraph/)
- igraph R manual pages

References

The data, R code and lessons are based upon:

Graph theory/data structures:
* http://math.tut.fi/~ruohonen/GT_English.pdf
* http://www.cl.cam.ac.uk/teaching/1011/PrincComm/slides-lpr/graph_theory_1-11.pdf
*
http://www.researchgate.net/publication/228300013_Graph_Theory_A_Primer_for_Using_R_Visualization_Techniques_in_the_Applications_of_the_Adjacency_Matrix
*
http://www.boost.org/doc/libs/1_59_0/libs/graph/doc/graph_theory_review.html

SNA:
* http://files.meetup.com/1406240/sna_in_R.pdf
* http://www2.unb.ca/~ddu/6634/Lecture_notes/Lec1_intro_handout.pdf
* http://www.faculty.ucr.edu/~hanneman/nettext/
*
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.372.1960&rep=rep1&type=pdf
* http://www.rdatamining.com/examples/social-network-analysis

igraph:
* http://statmath.wu.ac.at/research/friday/resources_WS0708_SS08/igraph.pdf
* http://blog.revolutionanalytics.com/2014/11/a-look-at-the-igraph-package.html
* http://www.r-bloggers.com/igraph-and-sna-an-amateurs-dabbling/
* http://www.r-bloggers.com/going-viral-with-rs-igraph-package/
* https://cran.r-project.org/web/packages/igraph/igraph.pdf

Other famous packages for SNA:
* http://www.r-bloggers.com/must-have-r-packages-for-social-scientists/
* https://cran.r-project.org/web/views/SocialSciences.html
* https://cran.r-project.org/web/packages/sna/sna.pdf
* https://cran.r-project.org/web/packages/RSiena/RSiena.pdf
* https://cran.r-project.org/web/packages/network/network.pdf
*
https://www.bioconductor.org/packages/release/bioc/manuals/graph/man/graph.pdf
* http://www.statnet.org/

In-depth SNA tutorials:
* http://sna.stanford.edu/rlabs.php
* http://www.stats.ox.ac.uk/~snijders/sna_course.htm
* http://www.shizukalab.com/toolkits

Sample projects:

- http://www.orgnet.com/cases.html

Motifs:
* http://igraph.org/r/doc/motifs.html
* https://en.wikipedia.org/wiki/Network_motif
* http://www.cs.columbia.edu/4761/notes07/chapter8.2-topology.pdf
* https://sites.google.com/site/networkanalysisacourse/schedule/networkmotifs
Cliques:
* http://igraph.org/r/doc/cliques.html
* http://faculty.ucr.edu/~hanneman/nettext/C11_Cliques.html
* https://courses.cs.washington.edu/courses/cse527/01au/oct25/oct25.html
* ttp://www.mathcove.net/petersen/lessons/get-lesson?les=29
* http://news.stanford.edu/news/2014/november/cliques-high-school-110514.html

PageRank:
* http://igraph.org/r/doc/page_rank.html
* http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
* http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm
* http://www.stat.cmu.edu/~ryantibs/datamining/lectures/03-pr-marked.pdf
* http://smallstats.blogspot.com/2014/04/from-random-walks-to-personalized.html
* http://blog.revolutionanalytics.com/2014/12/a-reproducible-r-example-finding-the-most-popular-packages-using-the-pagerank-algorithm.html
* Mining Massive Datasets on Coursera - Week 1, Videos 5 through 11 explain PageRank elegantly. The course maybe unavailable (or archived) by the time this module is out

Dataset:
* http://moreno.ss.uci.edu/data.html#padgett
* http://home.uchicago.edu/~jpadgett/papers/unpublished/maelite.pdf

Other SNA:
* http://www.r-bloggers.com/experiments-with-igraph/
* http://cran.us.r-project.org/doc/contrib/Zhao_R_and_data_mining.pdf